МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

ДИСКРЕТНА МАТЕМАТИКА

Рекомендовано вченою радою Київського національного університету будівництва і архітектури як навчальний посібник для здобувачів галузі знань 12 «Інформаційні технології» спеціальності 122 «Комп'ютерні науки» та 126 «Інформаційні системи і технології» освітньо-кваліфікаційного рівня «бакалавр»

Київ 2024

ББК 22.176 я73

Д48

Автори: Терентьєв О. О., д-р техн. наук, професор;
Серпінська О. І., асистент;
Баліна О. І., канд. техн. наук, доцент;
Безклубенко І. С., канд. техн. наук, доцент;
Рябчун Ю. В., Ph.D.

Рецензенти: *Є. В. Бородавка,* д-р техн. наук, професор, КНУБА;
Т. А. Гончаренко, канд. техн. наук, доцент, КНУБА;
Л. О. Терейковська, д-р техн. наук, КНУБА;
Ю. П. Буценко, канд.фіз.-мат.наук, доцент,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»;
Б. М. Єременко, канд. техн. наук,
Київський національний університет імені Тараса Шевченка

Затверджено на засіданні вченої ради Київського національного університету будівництва і архітектури, протокол № 10 від 31 травня 2024 року.

Д48 **Дискретна математика** [Електронний ресурс] : навчальний посібник / О. І. Серпінська та ін. – Київ : КНУБА, 2024. – 344 с.

ISBN 978-966-627-265-5

Створено за матеріалами лекцій із курсу «Дискретна математика», що є базовою дисципліною для підготовки бакалаврів зі спеціальностей 122 «Комп’ютерні науки» і 126 «Інформаційні системи і технології». Отримані знання будуть використані студентами під час вивчення дисциплін «Теорія алгоритмів», «Прикладне програмування», «Об’єктно-орієнтоване програмування», «Числові методи», «Комп’ютерні мережі» та ін. Містить основні положення теорії множин, комбінаторики, теорії графів і булевої алгебри. Матеріал проілюстровано численними прикладами. До кожного розділу подано задачі та вправи.

Призначено для здобувачів галузі знань 12 «Інформаційні технології» спеціальності 122 «Комп’ютерні науки» та 126 «Інформаційні системи і технології» освітньо-кваліфікаційного рівня «бакалавр».

ББК 22.176 я73

ISBN 978-966-627-265-5

ЗМІСТ

Вступ

Навчальний посібник створений за матеріалами лекцій із курсу «Дискретна математика». Дисципліна «Дискретна математика» є базовою для підготовки бакалаврів зі спеціальностей 122 «Комп’ютерні науки» і 126 «Інформаційні системи і технології».

Отримані знання будуть використані студентами під час вивчення дисциплін «Теорія алгоритмів», «Прикладне програмування», «Об’єктно-орієнтоване програмування», «Числові методи», «Комп’ютерні мережі» та ін.

Навчальний посібник містить основні положення теорії множин, комбінаторики та булевої алгебри. Зокрема, розглянуті основи теорії множин, відповідностей і відношень на множинах. Представлені відношення еквівалентності та порядку, детально описані їх властивості. Основні положення комбінаторики представлені законами комбінаторики, комбінаторними вибірками й типовими комбінаторними алгоритмами. Значну увагу приділено теорії графів. Розглянуті базові визначення теорії графів, способи створення та властивості графів, відношення й відображення на графах. Окремо розглянуто дерева, їх властивості та ліс. Представлені основні алгоритми на графах. У розділі «Булева алгебра» надані канонічні форми булевих функцій і розглянуті проблеми мінімізації формул алгебри логіки.

Посібник може бути корисним для інженерів і студентів технічних спеціальностей.

Розділ 1. Множини, операції над множинами

1.1. Основні положення теорії множин

Канторівський вираз «Множина – це зібрання в єдине ціле певних об'єктів, чітко розрізнюваних нашою інтуїцією чи нашою думкою», безумовно, не можна вважати строгим математичним означенням. Це радше пояснення поняття множини.

Прикладами множини можуть бути множини десяткових цифр, літер українського алфавіту, мешканців Києва, парних чисел, розв'язків якогось рівняння тощо. На письмі множини позначають зазвичай великими літерами. Для деяких множин у математиці використовують сталі позначення, наприклад: Z – множина цілих чисел, N – множина натуральних чисел, Q – множина раціональних чисел, R – множина дійсних чисел.

Основні визначення

Під ***множиною*** будемо розуміти довільну сукупність об'єктів, об'єднаних деякою спільною ознакою.

Синонімами поняття «множина» є поняття «сукупність», «клас», «збірка» тощо. Предмети, об'єкти, які містить множина, називають її елементами. Наприклад, дочка, син є елементами множини сім'ї.

Множини позначають великими літерами латинського алфавіту $A, B, C, D, \ldots$, а елементи множин – малими буквами цього алфавіту – $a, b, c, d, \ldots$

Приклад 1.1. Можливі позначення елементів множини: $a, b, c, \ldots, x, y, z$. Елементом множини може бути інша множина. У цьому випадку елементом множини є множина, яка представлена своєю назвою або елементами у фігурних дужках.

Приклад 1.2. Нехай $B = \{d, c, n\}$. Тоді $A = \{a, b, c, B\}$ або $A = \{a, b, c, \{d, c, n\}\}$.

Про елементи цієї множини говорять, що вони належать цій множині, і символічно записують так: $a \in A$.

Читають: «елемент a належить множині A» або «множина A містить елемент a».

Знак належності елемента множини – це стилізація першої літери грецького слова бути. Те, що елемент a не належить множині A, позначають так: $a \notin A$.

Запис $a, b, c, \ldots \in A$ використовують для скорочення запису $a \in A$, $b \in A$, $c \in A$ ...

Приклад 1.3. Позначення $x \in X$ показує, що елемент x належить множині X, тобто x є одним з елементів множини X. Позначення $\{d, c, n\} \in X$ показує, що множина $\{d, c, n\}$ є елементом множини X.

Позначення $a \notin X$ показує, що елемент a не належить множині X.

Вправа 1. Назвіть три елементи множини: а) навчальних предметів, що вивчаються в початковій школі; б) парних натуральних чисел; в) чотирикутників.

Вправа 2. Надані числа: 325, 0, –17, –3, 8, 7. Установіть, які з них належать множині: 1) натуральних чисел; 2) цілих чисел; 3) раціональних чисел; 4) дійсних чисел.

Вправа 3. Прочитайте ці висловлювання і вкажіть серед них істинні: 1) 100 $\in N$; 2) –8 $\in Z$; 3) –8 $\notin N$; 4) 5,36 $\in$ Q; 5) 102 $\notin$ R; 6) $\sqrt{2} \notin$ Q; 7) –7 $\in$ R; 8) $\frac{3}{4} \in N$; 9) 0 $\in Z$.

Вправа 4. Р – множина натуральних чисел, більших за 7 і менших за 14. З'ясуйте, які з чисел 13, 10, 5, 7, 14 йому належать, а які – не належать. Відповідь запишіть, використовуючи знаки $\in$ і $\notin$.

Способи задавання множини

Для ***задання множини***, утвореної з будь-яких елементів, застосовуватимемо способи, в основі яких лежить позначення множини за допомогою фігурних дужок.

Якщо $a_1, a_2, \ldots, a_n$ – якісь об'єкти, то їх множину позначають як $\{a_1, a_2, \ldots, a_n\}$, де у фігурних дужках міститься перелік усіх елементів відповідної множини. Порядок запису елементів множини в такому позначенні неістотний.

Приклад 1.4 Множину всіх десяткових цифр записують $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, множину всіх основних математичних операцій $\{+, -, \times, \div\}$, множину розв'язків нерівності $(x-1)^2 \leq 0 - \{1\}$.

Одна з основних ідей канторівської теорії множин – розгляд множини як нового самостійного об'єкта математичного дослідження. Тому потрібно розрізняти такі два різні об'єкти, як елемент a та множина $\{a\}$, що складається з єдиного елемента a. Зокрема, множини можуть бути елементами якоїсь іншої множини. Наприклад, множина

$D = \{\{a, b\}, \{a, c\}, \{b, c\}\}$ всіх можливих пар з елементів a, b, c складається з трьох елементів, і її задано цілком коректно.

Такий спосіб задавання множини громіздкий і не використовується в разі великої кількості елементів.

Другий спосіб задання множин ґрунтується на зазначенні загальної властивості чи породжувальної процедури для всіх об’єктів, що утворюють описувану множину.

У загальному випадку задання множини М має вигляд $M = \{a|P(a)\}$. Цей вираз слід читати так: «M – це множина всіх тих і тільки тих елементів a, для яких виконується умова P». Через Р(а) позначено або властивість, яку мають елементи множини M, або якусь породжувальну процедуру, що описує спосіб отримання елементів множини M з уже відомих її елементів чи інших об’єктів. Замість вертикальної риски іноді пишуть двокрапку.

Приклад 1.6. $S = \{n|n - \text{непарне число}\}$або $S = \{n|n = 2k + 1, k \in Z\}$, $X = \{x|x = \pi k, k \in Z\}$

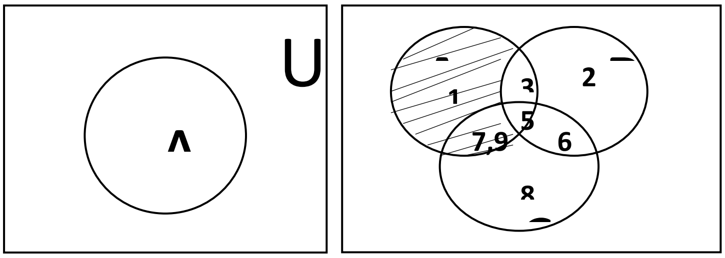
Другий спосіб задання множин більш загальний. Наприклад, уведену вище множину D всіх пар з елементів a, b, c можна задати так: $D = \{\{x, y\}|x \in \{a, b, c\}, y \in \{a. b. c\}, x \neq y\}$.

Третій спосіб задання множин носить назву діаграми Ейлера – Венна.

З визначення витікає, що рівні множини й відношення з множинами зручно ілюструвати за допомогою графічних схем, у яких множини представляються у вигляді кіл, овалів або будь-яких інших геометричних фігур і передбачається, що в цих геометричних фігурах містяться всі елементи цієї множини. Такі геометричні фігури називаються ***колами Ейлера***, від імені німецького математика Леонарда Ейлера, який у 1762 році застосував цю геометричну фігуру для логічних цілей.

Аналітичний, за допомогою символів операцій над множинами та дужок.

$$C = \overline{(B \cup A) \cap \overline{\overline{A}} \cup (A \cap \overline{B})} \cup B$$



Визначення підмножини

Дві множини A і B називають ***рівними*** (записують $A = B$), якщо вони складаються з тих самих елементів.

Множину A називають ***підмножиною*** множини B (записують $A \subseteq$ В чи $B \supseteq A$) тоді й тільки тоді, коли кожний елемент множини A належить також множині B. Знаки $і$ називають ***знаками включення***.

Неважко переконатися, що $A = B$ тоді й тільки тоді, коли одночасно виконуються два включення: $A \subseteq B$ та $B \subseteq A$. Крім того, якщо $A \subseteq B$ та $B \subseteq C$, то $A \subseteq C$. Останні два факти часто використовують у доведеннях тверджень про рівність двох заданих множин.

Якщо $A \subseteq B$, однак $A \neq B$, то пишуть $A \subset B$ і називають множину A ***власною*** (строгою або істинною) ***підмножиною*** множини B. Знаки $\subset і \supset$, на відміну від знаків $\subseteq\ і \supseteq$, називають ***знаками строгого включення***.

Очевидно, що для будь-якої множини A виконується включення
$A \subseteq A$.

Слід чітко розуміти різницю між знаками ∈ і ⊆, не плутати ситуації їх використання. Якщо $\{a\} \subseteq M$, то $a \in M$ і навпаки. Однак із включення $\{a\} \subseteq M$, взагалі кажучи, не випливає $\{a\} \in M$. Наприклад, для множини $D = \big\{\{a,b\},\{a,c\},\{b,c\}\big\}$ і її елементів виконуються такі співвідношення: $\{a,b\} \in D$, $\big\{\{a,b\},\{bc\}\big\} \subseteq D$, $a \in \{a,b\}$, $\{c\} \notin \{a,c\}$, $\{a\} \subseteq \{a,b\}$.

Визначення рівності множин

Множина X дорівнює множині Y у випадку, якщо будь-який елемент a належить множині X ($a \in X$) тоді й тільки тоді, коли $a \in Y$. Інакше кажучи, $X = Y$ (X дорівнює Y) тоді й тільки тоді, коли $X \subseteq Y$ і $Y \subseteq X$.

Приклад 1.7. Визначити, чи є рівними множини X та Y за умови, що $X = \{a, b, c, d\}$, $Y = \{c, a, b, d\}$.

Розв'язок. Перевіримо приналежність елементів множин X та Y. Будь-який елемент, що належить множині X, також належить і множині Y. Отже, $X = Y$.

Визначення порожньої множини

Порожня множина – це множина, яка не містить елементів. Позначення порожньої множини: ∅ або { }.

Крім того, вважають, що порожня множина є підмножиною будь-якої множини А, тобто $\emptyset \subseteq A$ (зокрема, $\emptyset \subseteq \emptyset$). Для будь-якого об’єкта x виконується $x \notin \emptyset$.

Вправа 5. Які з наведених співвідношень правильні?

(а) $\emptyset = \{0\}$; (б) $\emptyset = \{\square\}$; (в) $\{1, \emptyset\} = \{1\}$.

Визначення універсальної множини

Множина, яка вміщує в собі всі множини, що розглядаються, називається ***універсальною*** множиною, або ***універсумом***, і позначається U.

Властивості універсальної множини

Властивість 1. Будь-який об’єкт, яка б не була його природа, є елементом універсальної множини, зокрема $U \in U$.

Властивість 2. Будь-яка множина є підмножиною універсальної множини, зокрема $U \subseteq U$.

Універсальна множина має прикладний характер.

Множина U, незважаючи на те що названа універсальною, не може бути однозначно визначена, якщо не названа предметна область, тобто не зазначена властивість об’єктів, за якою ця множина формується.

1. Універсальна множина не є множиною всіх множин.
2. Універсальна множина є єдиною для певної чітко окресленої загальної ознаки її елементів.
3. Універсальна множина має прикладний характер.

Приклад. 1.8.

1. Універсальна множина в теорії чисел – множина цілих чисел.
2. Універсальна множина в математичному аналізі – множина дійсних чисел.

Скінченні й нескінченні множини

Множину називають ***скінченною,*** якщо кількість її елементів скінченна, тобто існує натуральне число k , що є кількістю елементів цієї множини. В іншому разі множина є ***нескінченною***. Кількість елементів скінченної множини А традиційно позначають $|A|$.

Кількість елементів скінченої множини називається ***потужністю*** множини.

Наприклад: $|\emptyset| = 0$, якщо $a = \{0, 1, 3\}$, тоді потужність $|A| = 3$.

Якщо $|A| = |B|$, то множини A та B називають ***рівнопотужними***.

Приклад 1.9. Навести приклад скінченної множини.

Розв'язок. Задамо скінченну множину перерахуванням $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9,10\}$. У цій множині $k = 10$.

Нескінченна множина – множина, що містить нескінченну кількість елементів.

Якщо не існує невід'ємного цілого числа k, то множину називають нескінченною.

Приклад 1.10. Навести приклад нескінченної множини.

Розв'язок. Задамо нескінченну множину предикатом $B = \{b\}$, b − *ціле число*. Для цієї множини не існує цілого невід'ємного числа k.

Зліченна множина – нескінченна множина, елементи якої принципово можливо пронумерувати натуральними числами.

Незліченна множина – така нескінченна множина, яка не є зліченною.

Приклад 1.11. Дати приклади нескінченних множин, до яких входять зліченні множини.

Розв'язок. Множини натуральних чисел N, цілих чисел Z. Множини раціональних чисел Q і дійсних чисел R. Множина комплексних чисел C.

Зліченна множина – нескінченна множина, елементи якої можна пронумерувати натуральними числами.

Зліченна множина – нескінченна множина X, у якій існує взаємо-однозначна відповідність $X \leftrightarrow N$, де N позначає множину всіх натуральних чисел.

Зліченна множина є «найменшою» нескінченною множиною, тобто в будь-якій нескінченній множині знайдеться зліченна підмножина.

Властивість 1. Будь-яка підмножина зліченної множини скінченна.

Властивість 2. Множина всіх скінченних підмножин зліченної множини зліченна.

Приклад 1.12. Приклад нескінченної зліченної множини. Зліченна множина – зірки, які принципово можливо порахувати. Скінченна множина, яка є підмножиною зліченної множини – це зірки, які занесені до зіркового каталогу.

Незліченна множина – така нескінченна множина, яка не є зліченною. Таким чином, будь-яка множина або скінченна, або зліченна, або незліченна.

n-множина – це множина, яка має *n* елементів.

Булеан

Визначення. Множину всіх підмножин множини *A* називають ***булеаном*** і позначають 2^A або P(A). У деяких джерелах булеан позначають як $\beta(A)$.

$$2^A = \{N | N \subseteq A\}$$

Якщо множини *A* має *n* елементів, то булеан *P(A)* міститиме 2^n елементів, через що його називають ***множиною-степенем*** множини *A*. Для скінченної множини *A* з *n* = *A* кількість її підмножин (потужність булеана) дорівнює $2^n = 2^{|A|}$. Отже, $|2^A| = 2^{|A|}$. Булеан включає порожню множину й саму множину *A*.

Приклад 1.13. Для множини $A = \{a, b\}$ маємо $P(A) = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$, а для $A = \{\varnothing, \{\varnothing\}\}$ $P(A) = \{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \{\{\varnothing, \{\varnothing\}\}\}\}$.

Вправа 6. Нехай *A* – множина студентів вашої групи, яка її потужність? Яка потужність булеана вашої групи?

Вправа 7. Визначити множину: (а) $\beta(\beta(\{1,2\}))$; (б) $\beta(\beta(\beta(\varnothing)))$.

<u>Обов'язкові завдання</u>

1. Які з наведених співвідношень правильні?

(а) $\{1,2,3\} = \{1,2,2,3\}$; (в) $\{1,2,3\} = \{1,3,2\}$;

(б) $\{1,2,3\} = \{1, \{2\}, 3\}$; (г) $\{1,2,3\} = \{(1,2), (2,3)\}$

2. З яких елементів складається множина *B*, якщо $A = \{1,2,3\}$?

(а) $B = \{y \mid y = x + z, x, z \in A\}$; (б) $B = \{y \mid x = y + z, x, z \in A\}$

(в) $B = \{y \mid y = xz, x, z \in A\}$.

3. Які з наведених співвідношень правильні?

(а) $\varnothing = \{0\}$; (в) $\{1, \varnothing\} = \{1\}$; (д) $|\{\varnothing\}| = 0$; (є) $|\{\varnothing, \{\varnothing\}\}| = 2$;

(б) $\varnothing = \{\ \}$; (г) $|\varnothing| = 0$; (е) $|\{\varnothing\}| = 1$; (ж) $|\{\{\varnothing\}\}| = 2$.

4. Які з наведених співвідношень правильні?

(а) $1 \in \{1,2,3\}$; (в) $\{1\} \in \{1,2,3\}$; (д) $\{1,2\} \in \{1,2,3\}$ (є) $\{1,2\} \in \{\{1,2\}\}$; (з) $a \in \{a\}$;

(б) $1 \in \{\{1\},\{2\},\{3\}\}$; (г) $\{1\} \in \{\{1\},\{2\},\{3\}\}$; (е) $\{1,2\} \in \{1,2\}$; (ж) $\{1,2\} \in \{\{1\},\{2\},\{3\}\}$.

5. Які з наведених співвідношень правильні?

(а) $0 \in \varnothing$; (в) $\varnothing \in \{1\}$; (д) $\varnothing \in \{\{\varnothing\}\}$;

(б) $\varnothing \in \varnothing$; (г) $\varnothing \in \{\varnothing\}$; (е) $\{\varnothing\} \in \{\varnothing, \{\varnothing\}\}$.

6. Які з наведених співвідношень правильні?

(а) $1 \subseteq \{1,2,3\}$; (в) $\{1,2,3\} \subseteq \{1,2,3\}$; (д) $\{1,2\} \subseteq \{\{1\},\{1,2\}\{3\}\}$;

(б) $\{1\} \subseteq \{1,2,3\}$; (г) $\{1\} \subseteq \{\{1\},\{2,3\}\}$; (е) $\varnothing \subseteq \{1,2,3\}$.

7. Нехай $A = \{1,2,\{1\}\}$. Які з наведених співвідношень правильні?

(а) $1 \in A$; (б) $\{1\} \in A$; (в) $\{\{1\}\} \in A$; (г) $\{1\} \subseteq A$;

(д) $\{\{1\}\} \subseteq A$; (ж) $\{\{2\}\} \subseteq A$; (і) $\varnothing \subseteq A$; (к) $\{\varnothing \subseteq A\}$;

(е) $\{2\} \in A$; (з) $\{1,2\} \in A$; (ї) $\varnothing \subseteq A$; (л) $\{\varnothing,1\} \subseteq A$;

(є) $\{2\} \subseteq A$; (и) $\{1,2\} \subseteq A$; (й) $\{\varnothing\} \in A$; (м) $\{\varnothing,\{1\}\} \subseteq A$.

8. Які з наведених співвідношень правильні?

(а) $\varnothing \subseteq \varnothing$; (г) $\{\varnothing\} \subseteq \{\{\varnothing\}\}$; (є) $\{\{\varnothing\}\} \subseteq \{\varnothing,\{\varnothing\}\}$;

(б) $\varnothing \subseteq \{\varnothing\}$; (д) $\varnothing \subseteq \{1\}$; (ж) $\{\{\varnothing\}\} \subseteq \{\varnothing\}$;

(в) $\{\varnothing\} \subseteq \varnothing$; (е) $\{\varnothing\} \subseteq \{\varnothing\}$; (з) $\{\{\varnothing\}\} \subseteq \{\{\{\varnothing\}\}\}$.

9. Чи існує така одноелементна множина B, що для якоїсь множини A одночасно виконуються співвідношення $A \in B$ і $A \subseteq B$?

10. Для множини A побудувати множину всіх її підмножин, тобто булеан P(A):

(а) $A = \{1,2,3\}$; (в) $A = \{1,\{2\},\{1,2\}\}$;

(б) $A = \{\varnothing\}$; (г) $A = \{\varnothing,\{1,2\}\}$.

11. Визначити множину:

(а) $\beta(\beta(\{1,2\}))$; (б) $\beta(\beta(\beta(\varnothing)))$.

12. Назвіть три елементи множини:

а) навчальних предметів, що вивчаються в початковій школі;

б) парних натуральних чисел;

в) чотирикутників.

13. B – множина парних чисел. Запишіть за допомогою символів такі висловлювання: 1) число 20 парне; 2) число 17 не є парним.

14. Запишіть, використовуючи символи: а) число 14 – натуральне; б) число – 7 не є натуральним; в) число 0 – раціональне; г) $\sqrt{7}$ – число дійсне.

15. Надані числа: 325, 0, –17, –3, 8, 7. Установіть, які з них належать множині: 1) натуральних чисел; 2) цілих чисел; 3) раціональних чисел; 4) дійсних чисел.

16. Прочитайте ці висловлювання і вкажіть серед них істинні:

1) $100 \in N$; 2) $-8 \in Z$; 3) $-8 \notin N$; 4) $5{,}36 \in Q$; 5) $102 \notin R$; 6) $\sqrt{2} \in Q$;

7) $-7 \in R$; 8) $\frac{3}{4} \in N$; 9) $0 \in Z$.

17. P – множина натуральних чисел, більших за 7 і менших за 14. З'ясуйте, які з чисел 13, 10, 5, 7, 14 йому належать, а які не належать. Відповідь запишіть, використовуючи знаки $\in$ і $\notin$.

18. A – множина рішень рівняння $x^2 + 1 = 0$. Чи правильно, що A – порожня множина? Наведіть приклади рівняння, множина рішень якого складається з: а) одного елемента; б) двох елементів; в) трьох елементів.

19. Запишіть за допомогою знака «дорівнює» і фігурних дужок речення: 1) X – множна чисел 0, 1, 2, 3, 4, 5; 2) Y – множина літер в слові «математика».

20. Множина C складається з квадрата, кола і трикутника. Чи належить цій множині діагональ квадрата?

21. Перерахуйте елементи такої множини: A – множина непарних однозначних чисел; B – множина натуральних чисел, не менших за 5; C – множина двозначних чисел, що діляться на 10.

22. Укажіть характеристичну властивість елементів множини:

а) {а, е, е, і, о, у, є, ю, я, и}; б) {23, 22, 21, 20, 19, 18, 17, 16, 15};

в) {11, 22, 33, 44, 55, 66, 77, 88, 99}.

23. A – множина двозначних чисел, запис яких закінчується цифрою 1. Чи належать цій множині числа 28, 31, 321, 61?

24. Надана множина A = {5, 10, 15, 25}. Укажіть дві підмножини, що рівні множині A.

25. Відомо, що елемент міститься в множині A і в множині B. Чи випливає із цього, що: 1) $A \subset B$; 2) $B \subset A$; 3) $A = B$?

26. Відомо, що кожен елемент множини A міститься в множині B. Чи правильно, що: 1) $A \subset B$; 2) $A = B$?

27. З множини К = {216, 546, 153, 171, 234} випишіть числа, які: 1) діляться на 3; 2) діляться на 9; 3) не діляться на 4; 4) не діляться на 5. Чи є серед отриманих підмножин таке, яке дорівнює множині К?

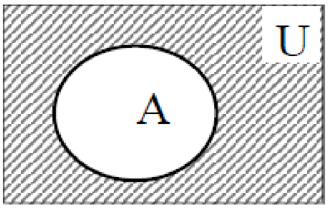
28. Зобразіть за допомогою діаграм Ейлера – Венна відношення між множинами А і В, якщо: 1) А – множина парних чисел, В – множина чисел, що кратні 3; 2) А – множина квадратів, В – множина прямокутників; 3) А – множина квадратів, В – множина прямокутних трикутників; 4) А – множина квадратів, В – множина прямокутників з однаковими сторонами.

29. Зобразіть за допомогою діаграм Ейлера – Венна відношення між множинами А, В і С, якщо відомо, що: 1) $A \subset B$ і $B \subset A$; 2) $A \subset B$, С перетинається з В, але не перетинається з А; 3) А, В і С перетинаються, але жодна не є підмножиною іншої.

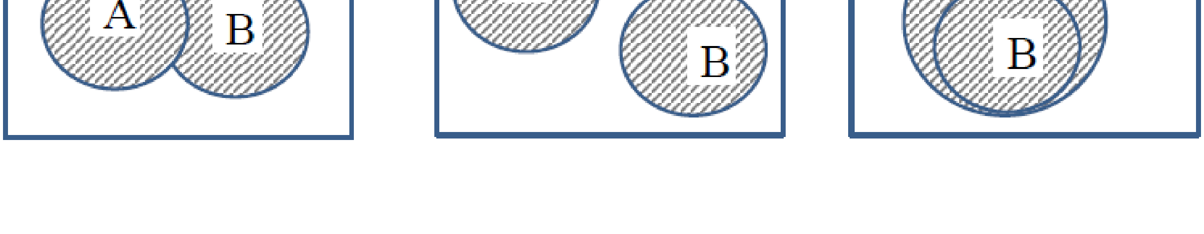
Операції над множинами

Операції над множинами дають змогу будувати нові множини, використовуючи вже існуючі.

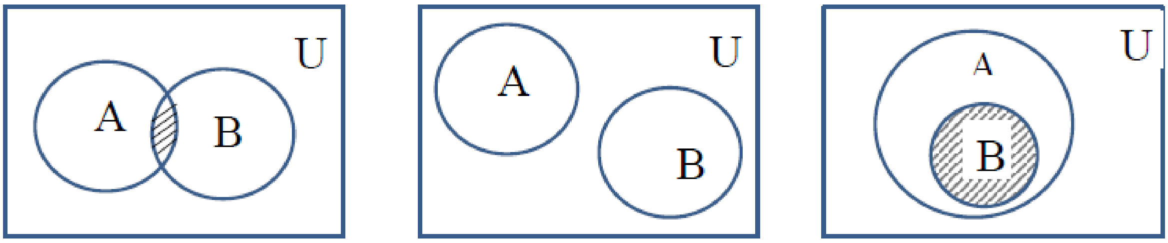
Доповненням до множини А, позначається (читається «не А» або «доповнення до А») є множина $\overline{A} = \{x | x \notin A, x \in U\}$.



Об'єднанням множин А та В називають множину С, яка складається з елементів А або В, а саме: $C = A \cup B = A + B = \{x \mid x \in A \text{ або } x \in B\}$.

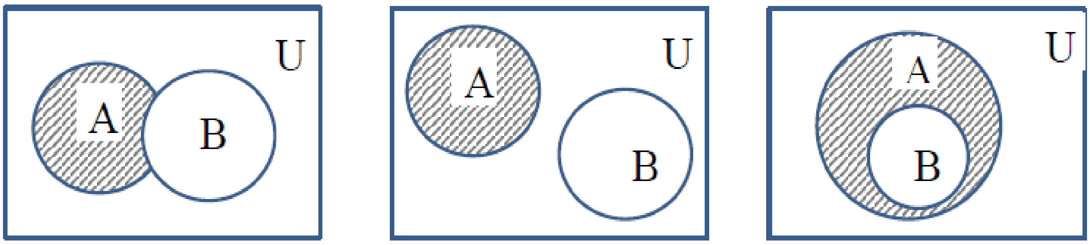


Перетином двох множин є третя множина C = A $\cap$ B, яка складається з елементів як A так і B: C = $\{x \mid x \cap A$ та $x \cap B\}$.



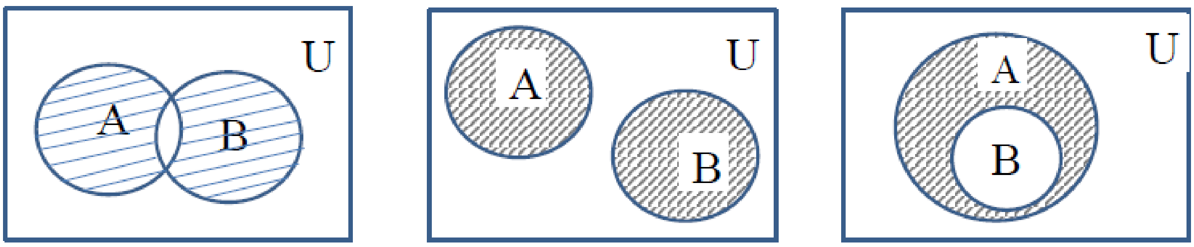
Різницею двох множин A та B позначається A\B або A-B називають множину C, яка складається з елементів A, що не належать

B : C =A\B =$\{x \mid x \in A$ та $x \notin B\}$.



Симетричною різницею множин A та B називають множину:

A $\div$ B = $\{x \mid x \in A$ та $x \notin B$, або $x \in B$ та $x \notin A\}$ або A $\div$ B = ((A\B) $\cup$ (B\A)).



Приклад 1.14. $\{a, b\} \cup \{c, d\} = \{a, b, c, d\}$, $\{a, c\} \cup \emptyset = \{a, c\}$, $\{a, b, c\} \cup \{a, c, d, e\} = \{a, b, c, d, e\}$.

Приклад 1.15. $\{a, b, c\} \cap \{a, c, d, e\} = \{a, c\}$, $\{a, b\} \cap \{c, d\} = \emptyset$**.**

Кажуть, що A і B не перетинаються, якщо $A \cap B = \emptyset$.

Вправа 8.

1. Найстарший математик серед шахістів і найстарший шахіст серед математиків – це та сама людина чи (можливо) різні?

2. Найкращий математик серед шахістів і найкращий шахіст серед математиків – це та сама людина чи (можливо) різні?

Приклад 1.16. $\{b,c\}\backslash\{a,d,c\} = \{b\}$**,** $\{a,c,d,e\}\backslash\{a,b,c\} = \{d,e\}$, $\{a,b\}\backslash\emptyset = \{a,b\}$**,** $\{a,b\}\backslash\{a,b,c,d\} = \emptyset$**.**

Приклад 1.17. $\{a,b,c\} \oplus \{a,c,d,e\} = \{\boldsymbol{b},d,e\}$**,** $\{a,b\} \oplus \{a,b\} = \emptyset$**,** $\{a,b\} \oplus \emptyset = \{a,b\}$**.**

Приклад 1.18. Якщо як універсальну множину взяти множину N всіх натуральних чисел, то доповненням P множини P всіх парних натуральних чисел буде множина всіх непарних натуральних чисел.

Діаграми Венна (Ейлера)

Діаграми Венна (Ейлера) – зручний інструмент, що дає змогу зображувати множини й ілюструвати операції над ними. Множини в діаграмах Венна зображують внутрішніми частинами кіл, їх перетинами, об’єднаннями тощо. На рис. 1 наведено діаграму Венна для множини *X*, яка зображена внутрішньою частиною кола. Зовнішня частина кола зображує $\overline{X}$.

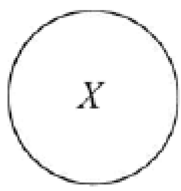


Рис. 1. Діаграма Венна для множини *X*

На рис. 2 наведено діаграма Венна для двох множин: *X* і *Y*. Кожна множина зображена колом, і кола перетинаються.

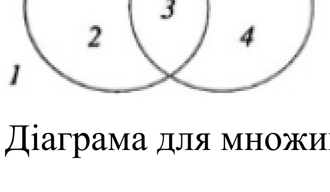


Рис. 2. Діаграма для множин *X* і *Y*

На рис. 2 бачимо 4 області: 1 – область універсальної множини, 2 – область, що належить тільки множині *X*, 3 – область, що належить спільно множинам *X* і *Y*, 4 – область, що належить тільки множині *Y* .

На рис. 3 наведені ілюстрації операцій над множинами.

Для вирішення багатьох задач, пов’язаних із множинами, незамінним виявляється прийом, що заснований на використанні так званих діаграм Ейлера – Венна. Ці діаграми вперше з’явилися в роботах одного з величніших математиків в історії Леонарда Ейлера. Використання кіл Ейлера додає наочності під час вирішення складних задач, роблячи деякі речі буквально очевидними. Пропоную вам у цьому впевнитися самостійно на прикладі вирішення такої задачі.

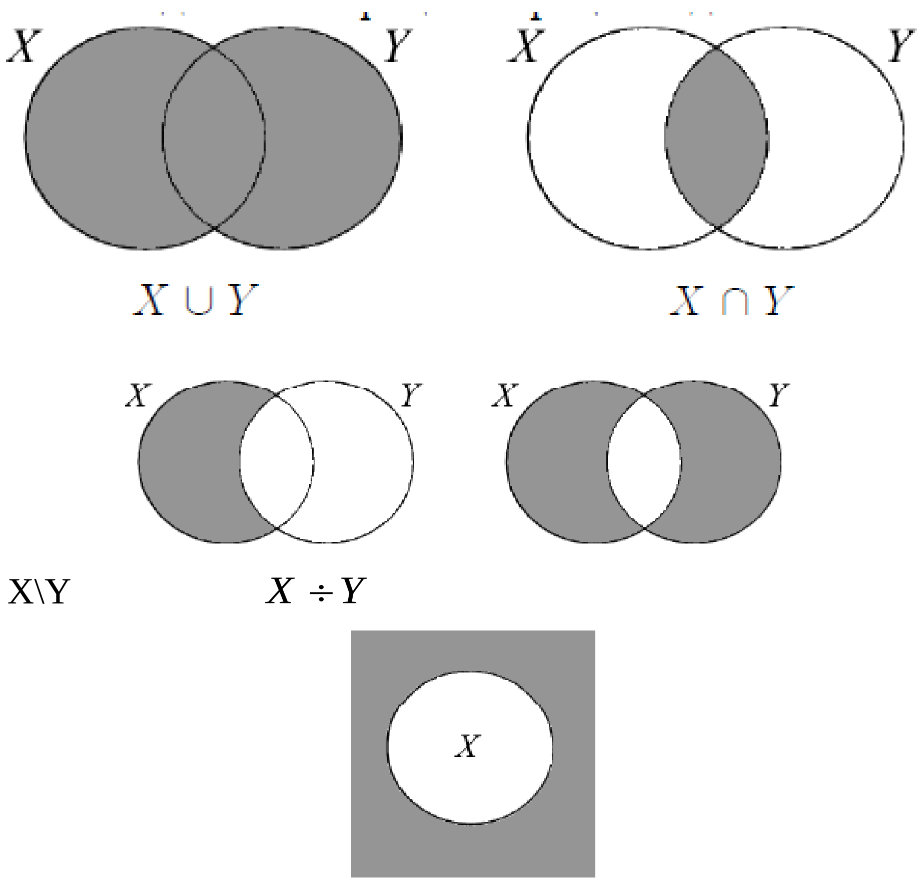


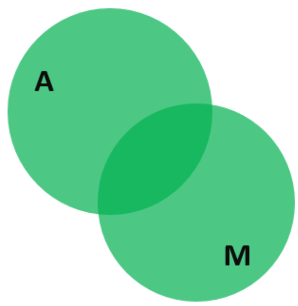
Рис. 3. Діаграми Венна операцій над множинами

Приклад 1.19 58 людей щоденно дістаються до роботи міським транспортом: на автобусі, трамваї або метро. Кожен користується хоча б одним з видів транспорту. 42 людини з них користуються метро, 32 –

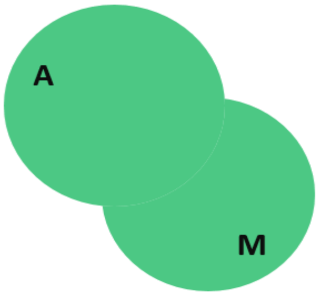
трамваєм, 44 – автобусом. 21 людина з них користується метро і трамваєм, 31 – метро й автобусом, 22 – трамваєм і автобусом. Скільки серед них людей, які використовують усі три види транспорту, щоб дістатися до роботи?

Тут треба розуміти, що якщо сказано, що «42 людини користуються метро», то це зовсім не означає, що, крім метро, вони не користуються ніяким іншим видом транспорту. Хто-небудь з них, може, і користується. Може бути ще якийсь один вид транспорту: трамвай або автобус. А може відразу й обидва! Вирішення задачі якраз і складається з того, щоб порахувати людей, які користуються всіма трьома видами транспорту.

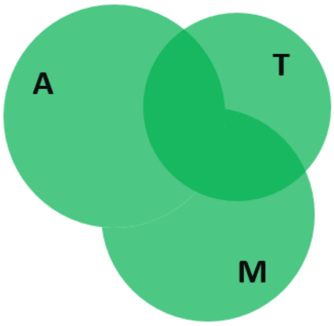
З першого погляду незрозуміло, із чого починати рішення. Але якщо подумати, стає ясно, що діяти треба за таким алгоритмом. Намагатимемося розписати всіх людей (58) через відомі з умови дані. Нам відомо, що автобусом користуються 44 людини. Додамо до цієї кількості людей, які користуються метро. Їх всього 42 людини. За допомогою діаграм Ейлера – Венна цю операцію можна зобразити наочно в такому вигляді:



Тобто поки що ми маємо діло з виразом 58 = 44 + 42… Знак «…» означає, що вираз ще не закінчений. Проблема в тому, що ми порахували людей на перетині цих кіл двічі. Відповідна область на діаграмі виділена темно-зеленим кольором. Тому один раз їх треба відняти. Це люди, які користуються автобусом і метро. Їх, як відомо, 31. Тобто наш «незакінчений» вираз набуває вигляду: 58 = 44 + 42 — 31… І на діаграмі при цьому зникає темно-зелений колір:

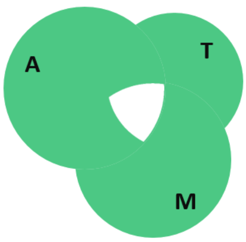


Поки все добре. Додаємо тепер людей, які їздять на трамваї. Таких людей 32. Вираз набуває вигляду: 58 = 44 + 42 — 31 + 32… Діаграма з колами Ейлера, зі свого боку, стає такою:

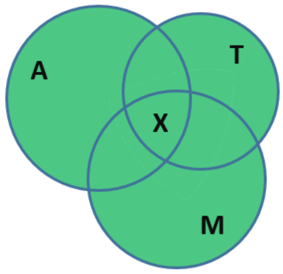


Проблема в тому, що знову ми схопили зайвого. Люди, яких ми знову порахували двічі, відмічені на діаграмі темно-зеленим кольором. Ця область перебуває на перетині множин, які ми отримали на попередньому етапі, і множини людей, що користуються трамваєм.

Треба відняти людей, яких ми порахували двічі. Але як це зробити? Єдине, що ми можемо зробити, це разом відняти людей, які користуються трамваєм і автобусом (їх 22 людини), а також трамваєм і метро (таких людей 21). Після цього наш незакінчений вираз для загальної кількості людей набуває вигляду: 58 = 44 + 42 — 31 + 32 — 22 — 21…, а діаграма з колами Ейлера виявиться з діркою в центрі, тому що центральну частину ми відняли двічі:



На щастя, у незафарбованій області якраз і перебувають ті люди, число яких нам потрібно порахувати. Дійсно, ці бідолахи користуються щоденно всіма трьома видами транспорту для того, щоб дістатися до роботи, бо вони перебувають на перетині всіх трьох множин. Позначимо кількість цих бідолах за x. Тоді діаграма набуде такого вигляду:

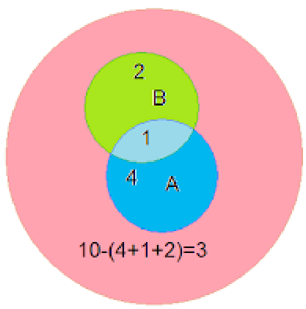


А рівняння стане таким: $58 = 44 + 42 – 31 + 32 – 22 – 21 + x$.

Розрахунки дають x = 14. Це і є відповіддю задачі. Стільки людей користуються всіма трьома видами транспорту щодня, щоб дістатися до роботи.

Ось таке просте рішення. Фактично в одне рівняння. Просто чудово, чи не так?! А тепер уявіть, як прийшлося б вирішувати цю задачу без використання кіл Ейлера. Це був би жах. Отже, вкотре пересвідчуємося, що будь-які методи візуалізації дуже корисні у вирішенні задач із математики. Використовуйте їх, це допоможе вам у вирішенні таких складних задач, як на олімпіадах.

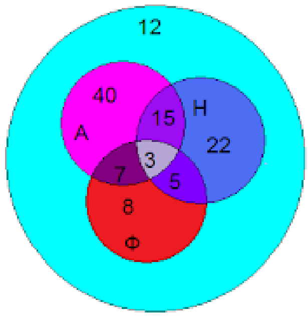
Приклад 1.20. Скільки натуральних чисел з першого десятка не діляться ані на 2, ані на 3?



Розв’язок. Для вирішення задачі зручно скористуватися колами Ейлера. У нашому випадку три кола: велике коло – це множина чисел від 1 до 10, усередині великого – два менших кола, що перетинаються один з одним. Нехай множина чисел, що кратні 2, – це множина А, а множина чисел, що кратні 3, – множина В. Розмірковуємо. На 2 ділиться кожне друге число. Отже, таких чисел буде 10 : 2 = 5. На 3 діляться 3 числа (10 : 3). На 2 і 3 діляться ті числа, що діляться на 6. Таке число тільки одне. Тому множина А складається з 5 – 1 = 4 чисел, множина В – 3 – 1 = 2 чисел. Звідси слідує, що в першому десятку міститься 10 – (4 + 1 + 2) = 3 числа.

Приклад 1.21. За допомогою кіл Ейлера можна відповісти на безліч питань, що поставлені до однієї умови задачі.

Розв'язок. Нехай коло А відображає всіх учнів, що говорять англійською, коло Н – що говорять німецькою мовою, коло Ф – що говорять французькою. Усього досліджуваних учнів 112. Скільки учнів говорять: а) усіма трьома мовами; б) англійською і німецькою; в) французькою? Скільки всього учнів, що розмовляють іноземними мовами? Скільки з них не говорять французькою? Скільки з них не говорять німецькою? Скільки з них не говорять іноземними мовами?



Відповідь: а) на всіх трьох мовах говорять 3 учні; б) англійською і німецькою – 15 учнів; в) тільки французькою – 8 учнів. Усього 100 (40 + 7 + 3 + 15 + 5 + 22 + 8) дітей, що розмовляють іноземними мовами. Французькою не говорять 89 учнів (112 – (8 + 5 + 7 + 3).

Тотожності алгебри множин

Тотожності алгебри множин, сформульовані в наведених нижче теоремах, можуть бути перевірені шляхом формальних доведень або на діаграмах Венна.

Основні тотожності алгебри множин

1. $A \cup A = A$ 1′. $A \cap A = A$

ідемпотентність

2. $A \cup B = B \cup A$ 2′. $A \cap B = B \cap A$

комутативність

3. $A \cup (B \cup C) = (A \cup B) \cup C$ 3′. $A \cap (B \cap C) = (A \cap B) \cap C$

асоціативність

4. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ 4′ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

дистрибутивність

5. $A \cup \emptyset = A$ 5′. $A \cap \emptyset = \emptyset$

6. $A \cup \bar{A} = U$ 6′. $A \cap \bar{A} = \emptyset$

7. $A \cup U = U$ 7′. $A \cap U = A$

8. $\overline{A \cup B} = \overline{A} \cap \overline{B}$ 8′. $\overline{A \cap B} = \overline{A} \cup \overline{B}$

закони де Моргана

9. $A \cup (A \cap B) = A$ 9′. $A \cap (A \cup B) = A$

Закони Порецького

10. $A \cup \left(\overline{A} \cap B\right) = A \cup B$ 10′. $A \cap \left(\overline{A} \cup B\right) = A \cap B$

закони поглинання

11. $\overline{U} = \emptyset$ 11′. $\overline{\emptyset} = U$

12. $A \backslash \emptyset = A$

13. $A \backslash A = \emptyset$

14. $U \backslash A = \overline{A}$

15. $A \backslash U = \emptyset$

16. $A \backslash B = A \cap \overline{B}$

17. $A \oplus B = (A \backslash B) \cup (B \backslash A)$

Наведу також інші корисні теоретико-множинні тотожності:

$A \cup \emptyset = A,\ A \cap \emptyset = \emptyset,\ A \cup U = U,\ A \cap U = A,$

$A \cup \overline{A} = U,\ A \cap \overline{A} = \emptyset,\ \overline{U} = \emptyset,\ \ \overline{\emptyset} = U.$

Окремо запишемо властивості операції симетричної різниці:

$A \oplus B = (A \backslash B) \cup (B \backslash A) = (A \cup B) \backslash (A \cap B) = \left(A \cap \overline{B}\right) \cup \left(\overline{A} \cap B\right);$

$(A \oplus B) \oplus C = A \oplus (B \oplus C)$ – асоціативність;

$A \oplus B = B \oplus A$ – комутативність;

$A \cap (B \oplus C) = (A \cap B) \oplus (A \cap C)$ – дистрибутивність перетину;

$A \oplus A = \emptyset;\; A \oplus U = \overline{A};\; A \oplus \emptyset = A.$

У справедливості перерахованих властивостей можна переконатися в різні способи. Наприклад, намалювати діаграми Ейлера для лівої та правої частин тотожностей і переконатися, що вони збігаються, або ж провести формальне міркування для кожної тотожності. Розглянемо, наприклад, першу тотожність: $A \cup A = A$. Візьмемо довільний елемент x, що належить до лівої частини тотожності, $x \in A \cup A$. За визначенням операції об'єднання маємо: $x \in A$або $x \in A$. У кожному разі $x \in A$. Візьмемо знову довільний елемент з множини в лівої частині тотожності. Виявляється, що він належить множині в правій частині. Звідси за визначенням включення множин одержуємо, що $A \cup A \subseteq A$.

Нехай тепер $x \in A$. Тоді, очевидно, правильно, що $x \in A$ або $x \in A$. Звідси за визначенням операції об'єднання маємо $x \subseteq A \cup A$. Таким чином, $A \subseteq A \cup A$.

Отже, за визначенням тотожності множин: $A \cup A = A$. Аналогічні міркування неважко провести й для інших тотожностей. Для доведення можна використовувати тотожності алгебри логіки.

Доведемо в такий спосіб властивість дистрибутивності множин.

Теорема 1.1. Для множин A і B справджується тотожність:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Доведення. Для доведення скористаємося операціями алгебри логіки, оскільки для доведення співвідношень на множинах потрібно довести, що ці співвідношення справедливі для всіх його елементів.

$x \in A \cap (B \cup C) \leftrightarrow (x \in A)$ і $(x \in (B \cup C)) \leftrightarrow$ Визначення перетину

$\leftrightarrow (x \in A)$ і $(x \in ((x \in B)$ або $(x \in C))) \leftrightarrow$ Визначення об'єднання

$\leftrightarrow ((x \in A)$ і $(x \in B))$ або $((x \in A)$ і $(x \in C)) \leftrightarrow$ Закон логіки де Моргана

$\leftrightarrow (x \in (A \cap B))$ або $(x \in (A \cap C)) \leftrightarrow$ Визначення перетину

$\leftrightarrow x \in ((A \cap B) \cup (A \cap C))$ Визначення об'єднання

Для доведення також використовують діаграми Венна (Ейлера). Доведення властивості асоціативності за допомогою діаграм Венна показано на рис. 4.

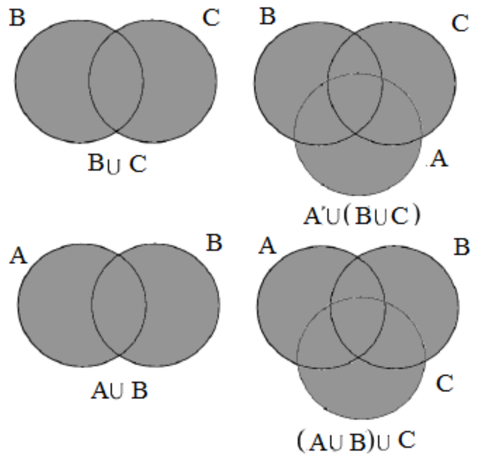


Рис. 4. Графічне доведення властивості асоціативності

Будуємо $(B \cup C)$, а потім $A \cup (B \cup C)$.

Будуємо $(A \cup B)$, а потім $(A \cup B) \cup C$.

Для доведення тотожностей алгебри множин можна використовувати інші тотожності алгебри множин.

Теорема 1.2. Для множин A і B справедлива тотожність (закон склеювання) $(A \cap B) \cup (A \cap \overline{B}) = A$

Доведення. Доведемо цю тотожність двома способами: аналітично (використовуючи алгебру множин) і конструктивно (використовуючи діаграми Ейлера – Венна).

$(A \cap B) \cup (A \cap \overline{B})$ = початковий вираз

$= (A \cup (A \cap \overline{B})) \cap (B \cup (A \cap \overline{B}))$ = застосували закон дистрибутивності відносно $(X \cap \overline{Y})$

$= (A \cup (A \cap \overline{B})) \cap (B \cup A)$ = застосували закон Порецького

$= A \cap (B \cup A)$ = застосували закон склеювання для об'єднання

$= A$ застосували закон склеювання для перетину

На рис. 5 показано діаграми Ейлера – Венна для доведення закону склеювання. $A \cap B$ $A \cap \overline{B}$ $(A \cap B) \cup (A \cap \overline{B}) = A$

Приклад 1.22. Доведемо тотожність: $A \setminus (B \cup C) = (A \setminus B) \setminus C$

Доведення.

1 спосіб

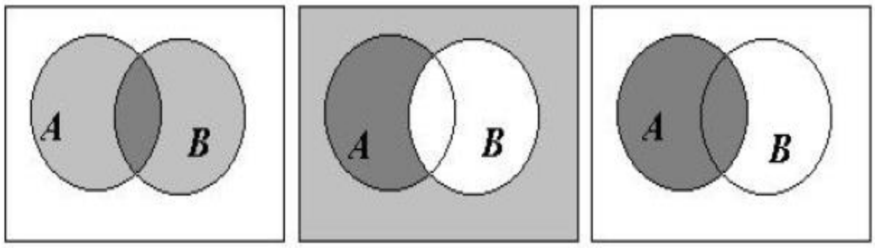


Рис. 5. Графічне доведення закону склеювання

1. Доведемо, що $A\setminus(B\cup C)\subseteq(A\setminus B)\setminus C$. Розглянемо довільний елемент множини
$A\setminus(B\cup C)$: $x\in A\setminus(B\cup C)\Rightarrow x\in A$ і $x\notin B\cup C\Rightarrow x\in A$ і $x\notin B$ і $x\notin C\Rightarrow x\in A\setminus B$ й $X\notin C\Rightarrow x\in(A\setminus B)\setminus C$.

2. Доведемо, що $(A\setminus B)\setminus C\subseteq A\setminus(B\cup C)$. Нехай $x_{\in}(A\setminus B)\setminus C$:
$X\in(A\setminus B)\setminus C\Rightarrow x\in A\setminus B$ і $x\notin C\Rightarrow x\in A$ і $x\notin B$ і $x\notin C\Rightarrow x\in A$ й $x\notin B\cup C\Rightarrow$
$x\in A\setminus(B\cup C)$.

2 спосіб

Перетворимо ліву частину тотожності на праву за допомогою властивостей операцій над множинами:

$A\setminus(B\cup C)=A\cap(\overline{B\cup C})=A\cap(\overline{B}\cap\overline{C})=(A\cap\overline{B})\cap\overline{C}=(A\setminus B)\cap\overline{C}=$
$=(A\setminus B)\setminus C$

На рис. 6 зображені обидві частини тотожності за допомогою діаграм Ейлера – Венна:

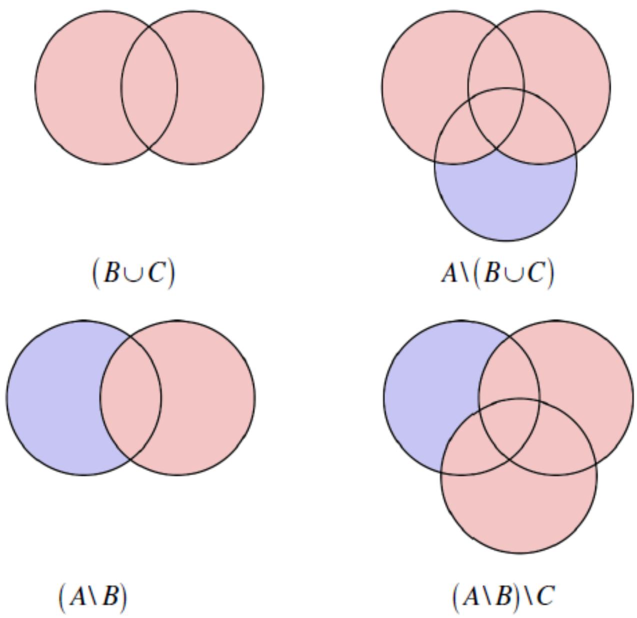


Рис. 6. Графічне доведення тотожності $A\setminus(B\cup C)=(A\setminus B)\setminus C$

<u>Обов'язкові завдання</u>

1. Нехай $A = \{1, 3, 5, 6\}, B = \{1, 2, 3, 5, 7\}, C = \{2, 4, 7\}$. Обчислити
(а) $A \cup B$; (б) $(A \cup C)\backslash B$; (в) $A \cap B \cap C$; (г) $(A\backslash C) \cup (B\backslash A)$;
(д) $A \oplus B$; (е) $(B\backslash C) \cap (A\backslash B)$.

2. За допомогою діаграм Ейлера – Венна перевірити такі теоретико-множинні рівності:
(а) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$; (б) $(A \cap B) \cup A = (A \cup B) \cap A = A$;
(в) $A\backslash(B \cup C) = (A\backslash B) \cap (A\backslash C)$; (г) $(A \cup B)\backslash C = (A\backslash C) \cup (B\backslash C)$;
(д) $A \cap (B \oplus C) = (A \cap B) \oplus (A \cap C)$; (е) $A \oplus B = (A \cup B)\backslash(A \cap B)$.

3. Що можна сказати про множини А та В, якщо
(а) $A \cup B = A \cap B$; (б) $A\backslash B = B\backslash A$; (в) $A \subseteq \overline{B}$ та $\overline{A} \subseteq B$; (г) $A \cup B = \emptyset$;
(д) $A\backslash B = A$; (е) $A\backslash B = \emptyset$; (є) $A\backslash B = B$; (ж) $(A\backslash B) \cup (B\backslash A) = A \cup B$.

4. Що можна сказати про множини А та В, якщо
(а) $A \oplus B = A$; (в) $A \oplus B = \emptyset$; (д) $(A\backslash B) \oplus (B\backslash A) = \emptyset$;
(б) $A \oplus B = \overline{A}$; (г) $A \oplus B = U$; (е) $(A \cup B) \oplus A = B$

5. Шестикласники заповнювали анкету з питаннями про їхні улюблені мультфільми. Виявилося, що більшості з них подобається «Білосніжка і семеро гномів», «Губка Боб Квадратні Штані» і «Вовк і теля». У класі 38 учнів. «Білосніжка і семеро гномів» подобається 21 учню. Причому трьом із них подобається ще і «Вовк і теля», шістьом – «Губка Боб Квадратні Штані», а одна дитина однаково любить усі три мультфільми. У «Вовка і теля» 13 фанатів, п'ятеро з яких назвали в анкеті два мультфільми. Треба визначити, скільком шестикласникам подобається «Губка Боб Квадратні Штані».

Розбиття множин

Множина X може бути розбита на класи множин X_j, які не перетинаються, якщо:

– об'єднання всіх підмножин X_j збігається з множиною $X : X = \bigcup_{j \in J} X_j$:

– перетин двох різних підмножин порожній, тобто для будь-яких двох $i \in J$ і $j \in J$ з $i \neq j$ виконується умова: $X_i \cap X_j = \emptyset$.

Приклад 1.23. Розбиття множини на підмножини.

1. Довільна множина X може бути розбита на дві підмножини, які доповнюють одна одну X_1 і $X_2 = X \setminus X_1$. Для цих підмножин справедливі співвідношення: $X_1 \cup X_2 = X$ і $X_1 \cap X_2 = \emptyset$.

Приклад 1.24. Множину двозначних чисел $X = \{10, 11, 12, \dots, 98, 99\}$ можна розбити на класи за ознакою остачі від ділення на 4:
клас, породжений остачею 0 – $X_0 = \{12, 16, 20, \dots, 96\}$;
клас, породжений остачею 1 – $X_1 = \{13, 17, 21, \dots, 97\}$;
клас, породжений остачею 2 – $X_2 = \{14, 18, 22, \dots, 98\}$;
клас, породжений остачею 3 – $X_3 = \{15, 19, 23, \dots, 99\}$.

Покриття множин

Покриттям множини X називають сімейство множин $C = \{Y_j\}, j \in J$ таких, що їх об'єднання містить множину X: $X \subset \bigcup\limits_{j \in J} Y_j$.

Якщо C – покриття множини X, то будь-яку множину $D \subset C$, що також є покриттям множини X, називають **під покриттям** множини C.

Приклад 1.25. Нехай $X = \{i | i = 2n + 1, n = 1, 2, 3, \dots\}$. Побудувати покриття множини X

Розв'язок. $J = \{1, 2\}$, $C = \{Y_1, Y_2\}$, $Y_1 = \{-k | k = 1, 2, \dots\}$, $Y_2 = \{k | k = 1, 2, \dots\}$. Тоді $X \subset Y_1 \cup Y_2$, а, отже, сімейство множин C є покриттям множини X.

Упорядкований набір або кортеж

Нехай задана деяка множина X. Візьмемо множину натуральних чисел $N_n = \{1, 2, \dots, n\}$ і задамо деякі відображення множини N_n у множину X (числу 1 ставиться у відповідність елемент $x_1 \in X$, числу 2 – елемент $x_2 \in X$, …, числу n – елемент $x_n \in X$).

Отримуємо набір $x_1, x_2, \dots, x_n$ елементів множини X, у якому деякі елементи можуть повторюватися декілька разів (з відображенням N_n у X може трапитися, що різним числам відповідає той самий елемент множини X). Розташовуючи елементи цього набору за порядком номерів, отримуємо **кортеж** $(x_1, x_2, \dots, x_n)$ довжини n, який складено з елементів множини X. Елемент x_k, $1 \leq k \leq n$, називається ***k*-тою компонентою** або ***k*-тою координатою кортежу** $(x_1, x_2, \dots, x_n)$.

Два кортежі $(x_1, x_2, \ldots, x_n)$ і $(y_1, y_2, \ldots, y_n)$ вважаються **рівними**, якщо вони мають однакову довжину і їх компоненти з однаковими номерами рівні.

Позначатимемо кортежі грецькими літерами. Компонентами кортежу можуть бути множини, кортежі і т. д.

Приклад 1.26.

1) Кортежі α = $(3^2, 4^2, 5^2)$ і β = $(\sqrt{81}, \sqrt{256}, \sqrt{625})$ рівні, α = β.

2) Кортежі (a, b, c) і (b, a, c, a) не рівні, оскільки мають різну довжину.

3) Кортежі (1, 2, 3) і (2, 1, 3) мають однакову довжину і складаються з однакових елементів, але вони не рівні, оскільки порядок їх компонент не збігається.

4) Кортежі α = $(\{a, b\}, c, d)$ і β= $(\{b, a\}, c, d)$ рівні, оскільки множини $\{a, b\}$ і $\{b, a\}$ рівні (для множин порядок елементів не грає ролі).

5) Кортежі α = $((a, b), c, d)$ і β= $((b, a), c, d)$ різні, оскільки різні кортежі (a, b) і (b, a).

Декартовий добуток множин

Узагальнимо поняття кортежу й будемо розглядати такі кортежі, у яких компоненти належать різним множинам.

Декартовим добутком множин A і B (позначається $A \times B$) називається множина всіх пар *(a, b)*, у яких перша компонента належить множині A ($a \in A$), а друга – множині B ($b \in B$), тобто $A \times B = \{(a, b) | a \in A, b \in B\}$.

Декартів добуток можна природно узагальнити для довільної скінченної сукупності множин. Якщо $A_1, A_2, \ldots, A_n$ – множини, то їх декартовим добутком називається множина $D = \{(a_1, a_2, \ldots, a_n) | a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n\}$, яка складається з усіх наборів $(a_1, a_2, \ldots, a_n)$, у кожному з яких і-й член, що називається ***і-тою координатою*** , або ***і-тою компонентою*** набору, належить множині $A_i, i = 1,2, \ldots, n$. Декартів добуток позначають $A_1 \times A_2 \times \ldots \times A_n$. Щоб відрізняти набір $(a_1, a_2, \ldots, a_n)$ від множини, що складається з елементів $a_1, a_2, \ldots, a_n$, його записують не у фігурних, а в круглих дужках і називають ***кортежем***, ***вектором*** або ***впорядкованим набором***.

Знайдемо число елементів декартового добутку $A \times B$ у випадку, коли $|A| = k$, а $|B| = m$: $A = \{(a_1, ..., a_k)\}$, $B = \{(b_1, ..., b_m)\}$.

Декартовий добуток $A \times B$ складається з кортежів $(a_i,\ b_j)$, які можна розташувати таким чином:

$$(a_1, b_1), (a_1, b_2), \dots, (a_1. b_m);$$
$$(a_2, b_1), (a_2, b_2), \dots, (a_2, b_m);$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots..$$
$$(a_k, b_1), (a_k, b_2), \dots, (a_k, b_m).$$

Ми отримаємо k рядків з m пар в кожному. Звідси випливає, що загальна кількість кортежів, що містяться в множині $A \times B$, дорівнює $k \times m$, тобто $n(A) \times n(B)$. Має місце формула: $n(A \times B) = n(A) \times n(B)$ – **правило добутку.**

Довжиною кортежу називають кількість його координат.

Два кортежі $(a_1, a_2, \dots, a_n)$ і $(b_1, b_2, \dots, b_n)$ однакової довжини вважають ***рівними*** тоді й тільки тоді, коли рівні відповідні їх координати, тобто $a_i = b_i, i = 1, 2, \dots, n$. Отже, кортежі *(a, b, c)* і *(a, c, b)* різні, а множини $\{a, b, c\}\ i\ \{a, c, b\}$ – рівні між собою.

Декартів добуток множини A на себе n разів, тобто множину $A \times A \times \ldots \times A$ називають ***n-м декартовим (прямим) степенем*** множини A та позначають A^n. Вважають, що $A^0 = \emptyset (n = 0)$ й $A^1 = A (n = 1)$.

Прийнято вважати, що якщо хоча б одна з множин *A, B* порожня, то їх декартовий добуток порожній: $A \times \emptyset = \emptyset \times B = \emptyset$.

Якщо $A = B$, то декартовий добуток $A \times B$ називається **декартовим квадратом множини** A і позначається: $A \times B = A^2 = \{(a, b)\ |a \in A, b \in A\}$

Як універсальна множина для множини $A \times B$ є множина:

1) $U \times U$, якщо $A \in U\ i\ B \in U$;

2) $U_1 \times U_2$, якщо $A \in U_1$ і $B \in U_2$.

Приклад 1.27. Побудувати $C = A \times B$, якщо $A = \{x, y, z\}$, $B = \{1, 2\}$.

Розв'язок. $C = \{(x, 1). (x, 2)(y, 1). (y, 2), (z, 1), (z, 2)\}$.

Графічна інтерпретація декартового добутку

Існує графічна інтерпретація прямого добутку множин. Нехай множина $A = \{x | a \le x \le b\}$ – це інтервал значень змінної x і $B = \{y | c \le y \le d\}$– це інтервал значень змінної y. Ясно, що множини A і B

мають нескінченне число елементів. Тоді прямий декартовий добуток $A \times B$ – це множина точок прямокутника, зображеного на рис. 7.

Отже, $C = A \times B = \{(x, y) | x \in A, y \in B\}$.

Приклад 1.28. Якщо $A = \{a, b\}$ та $B = \{b, c, d\}$, то $A \times B = \{(a, b), (a, c), (a, d), (b, b), (b, c), (b, d)\}$;
$A^2 = \{(a, a), (a, b), (b, a), (b, b)\}$.

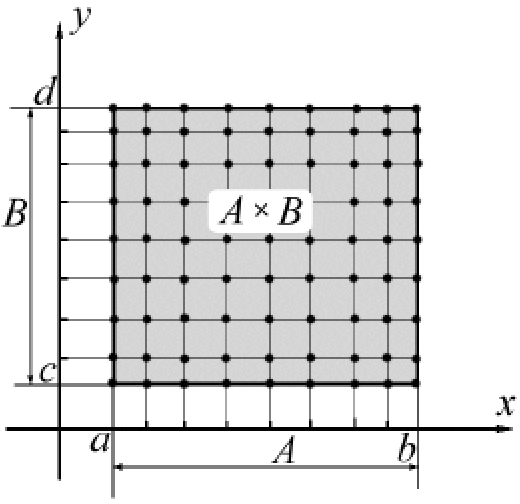


Рис. 7. Геометричне місце точок, що входять до декартового добутку множин A і B

Якщо R – множина дійсних чисел, або множина точок координатної прямої, то R^2 – це множина пар *(a, b)*, де $a, b \in R$, або множина точок координатної площини.

Операція декартового добутку неасоціативна й некомутативна, тобто множини $(A \times B) \times C, A \times (B \times C)$, а також множини $A \times B$ та $B \times A$, узагалі кажучи, різні.

Зв'язок декартового добутку з іншими теоретико-множинними операціями виражають такі тотожності:

$$(A \cup B) \times C = (A \times C) \cup (B \times C);$$
$$(A \cap B) \times C = (A \times C) \cap (B \times C);$$
$$A \times (B \cup C) = (A \times B) \cup (A \times C);$$
$$A \times (B \cap C) = (A \times B) \cap (A \times C).$$
$$\left(\overline{A \times B}\right) = \left(\overline{A} \times \overline{B}\right) \cup \left(\overline{A} \times B\right) \cup \left(A \times \overline{B}\right).$$

Для підмножин будуть правильні твердження:

якщо $A \subseteq B$, то $A \times C \subseteq B \times C$,
якщо $A, B \neq 0$, то $A \times B \subseteq C \times D \Leftrightarrow A \subseteq C$ і $B \subseteq D$.

Зворотний декартовий добуток

Нехай $C = A \times B$– прямий декартовий добуток множин. Тоді $C^{-1} = B \times A$ називатиметься **зворотним** декартовим добутком до прямого добутку.

Приклад 1.29. Побудувати прямий і зворотний декартові добутки для множин A= {1, 2, 3} і B= {x, y, z}.

Розв'язок.

$C = A \times B$ = {(1, x), (1, y), (1, z), (2, x), (2, y), (2, z), (3, x), (3, y), (3, z)}.
$C^{-1} = B \times A = \{(x, 1), (x, 2), (x, 3), (y, 1), (y, 2), (y, 3), (z, 1), (z, 2), (z, 3)\}$.

Проєктування

Проєкцією на і-ту вісь (і-тою проєкцію) кортежу $w = (a_1, a_2, \dots, a_n)$ називається і-та координата a_iкортежу w_δ позначається $Pr_i\, w$.

Проєкцією кортежу $w = (a_1, a_2, \dots, a_n)$ на осі з номерами $i_1, i_2, \dots, i_n$називається кортеж $\left(a_{i_1}, a_{i_2}, \dots, a_{i_n}\right)$; позначається $Pr_{i_1, i_2, \dots, i_n}\, w$.

Нехай V– множина кортежів однакової довжини. Проєкцією множини V на і-ту вісь (позначається $\mathrm{Pr}_i V$) називається множина проєкцій на і-ту вісь усіх кортежів множини V: $Pr_i\, V = \{Pr_i\, v \,|v \in V\}$.

Операція проєктування може застосовуватися тільки до тих множин, які містять кортежі однакової довжини.

Приклад 1.30. $Pr_{i_1, i_2, \dots, i_n}(A_1 \times A_2 \times \dots \times A_n) = A_{i_1} \times A_{i_2} \times \dots \times A_{i_n}$. Якщо $V = \{(a, b, c), (a, c, d), (a, b, d)\}$, то $Pr_1\, V = \{(a)\}$, $Pr_2\, V = \{(b, c)\}$, $Pr_3\, V = \{(c, d)\}$, $Pr_{1,3}\, V = \{(a, c), (a, d)\}$, $Pr_{2,3}\, V = \{(b, c), (c, d), (b, d)\}$.

<u>Обов'язкові завдання</u>

1. Довести тотожності $(A \cap B) \cup A = (A \cup B) \cap A = A$.
2. Для заданих множин $A = \{1, 2\}, B = \{2, 3, 4\}$ визначити
   (а) $A \times B$; (в) B^2; (д) $A \times B \times A$;
   (б) $B \times A$; (г) $(B \backslash A) \times A$; (е) $A \times (A \cup B)$.
3. Довести $(A \times B) \cap (B \times A) = (A \cap B) \times (A \cap B)$.
4. Довести, що коли $B = \emptyset$, то $Pr_1(A \times B) = A$.

1.2. Відповідності та відношення

Розглянемо множини A і B. Елементи цих множин можуть зіставлятися один з одним будь-яким чином, утворюючи впорядковані пари $(a,\ b)$. Якщо спосіб такого зіставлення визначений, тобто для кожного елемента $a \in A$ вказано елемент $b \in B$, з яким зіставляється елемент a, то говорять, що між множинами A і B встановлена відповідність.

Для того щоб задати відповідність, потрібно вказати:

1) множину A, елементи якої зіставляють з елементами іншої множини;

2) множину B, елементи якої ставлять у відповідність елементам першої множини;

3) множину $Q \subseteq A \times B$, що визначає закон (правило), за яким здійснюють відповідність, тобто таке правило, що перераховує всі пари *(a, b)*, які беруть участь у зіставленні.

Таким чином, відповідність (позначимо її через q) є трійкою множин

$$q = \langle A, B, Q \rangle$$

де – $Q \subseteq A \times B$ підмножина декартового добутку множин A і B, яку ще називають графіком відповідності; A – множина відправлення відповідності; B – множина прибуття відповідності.

Крім того, з кожною відповідністю зв’язані нерозривно ще дві множини:

1) множина $Pr_a\, Q$, яку називають **областю визначення** відповідності. До цієї множини входять елементи множини, що беруть участь у зіставленні;

2) множина $Pr_b\, Q$, яку називають **областю значень** відповідності. До цієї множини входять елементи множини, що беруть участь у зіставленні.

Якщо $(a, b) \in Q$, то говорять, що елемент b відповідає елементу a.

Геометрично це зображають у вигляді стрілки, спрямованої від до. На рис. 8 показано дві множини A і B з установленими відповідностями між їх елементами. При цьому графік відповідності має вигляд:

$$Q = \{(a_1, b_6), (a_2, b_7), (a_3, b_1), (a_4, b_2), (a_5, b_3), (a_6, b_4), (a_7, b_5)\}.$$

Для кожної відповідності $q = \langle A, B, Q \rangle$ існує зворотна відповідність, яка утворюється у випадку, коли цю відповідність розглядають у зворотному напрямку, тобто визначають елементи $a \in A$, з якими зіставляються елементи $b \in B$.

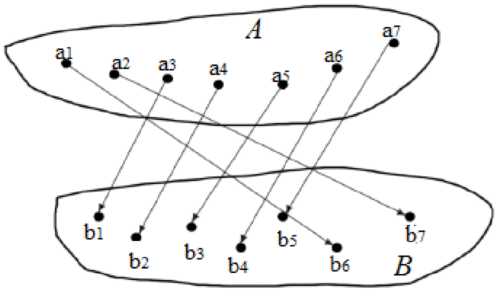


Рис. 8. Графічне представлення відповідності множин A і B

Зворотна відповідність позначається: $q^{-1} = \langle A, B, Q^{-1} \rangle$, де $Q^{-1} = B \times A$. Геометрично зворотну відповідність можна одержати шляхом зміни напрямку стрілок прямої відповідності.

Типи відповідностей

Відповідності можуть бути різних типів: одно-однозначні, одно-багатозначні, багато-однозначні, багато-багатозначні.

Одно-однозначна (або взаємно-однозначна) відповідність

Одно-однозначна (або **взаємно-однозначна)** відповідність – це така попарна відповідність між елементами двох множин A і B, коли один елемент з A зіставлено з єдиним елементом з B і навпаки.

Приклад 1.31. Нехай існує множина натуральних чисел N і множина квадратів натуральних чисел P. Побудувати одно-однозначну відповідність.

Розв'язок. Кожному натуральному числу можна поставити у відповідність його квадрат, і навпаки – кожному квадрату цілого числа відповідає саме натуральне число.

Тому між множинами N й P існує взаємно-однозначна відповідність, як показано на рис. 9.

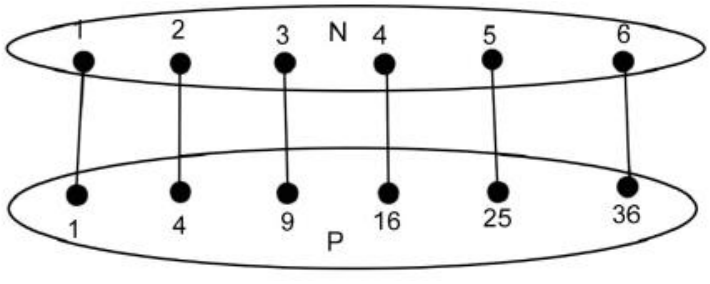


Рис. 9. Графічне представлення відповідності елементів множини натуральних чисел N та множини квадратів цих чисел P

Одно-багатозначна відповідність

Одно-багатозначна відповідність – це така відповідність між елементами двох множин A і B, коли з одним елементом першої множини A зіставлено більше одного елемента другої множини B, але кожний елемент другої множини відповідає тільки одному елементу першої множини.

Приклад 1.32. Нехай існує множина квадратних коренів $G = \{1,2,3,4,5\}$ і множина цілих чисел $N = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, \dots, 25\}$.

Побудувати відповідність між елементами цих множин.

Розв'язок. Кожному елементу множини G однозначно відповідає один елемент множини N. Зворотна відповідність може бути багатозначною за умови, що ми будемо виділяти цілу частину від взяття квадратного кореня для кожного елемента множини N, як показано на рис. 10.

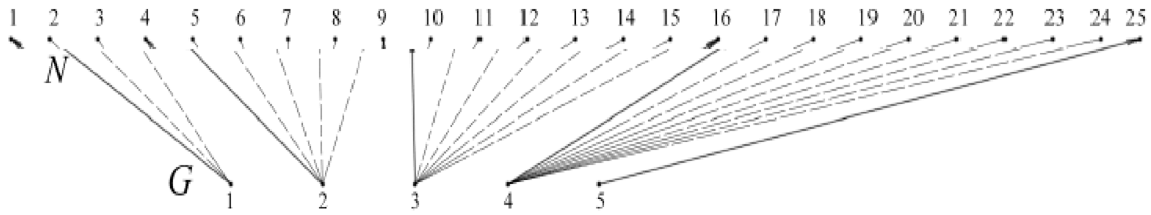


Рис. 10. Графічне представлення одно-багатозначної відповідності між елементами множини N та G

Багато-однозначна відповідність

Багато-однозначна відповідність – це така відповідність між елементами двох множин A і B, коли з елементом першої множини зіставлено тільки один елемент другої множини, але кожний елемент другої множини відповідає більше ніж одному елементу першої множини.

Приклад 1.33. Нехай $A = \{1, 2, 3, 4, 5\}$ – це припустима множина оцінок, а $B = \{1, 2, 3, 4, 5, 6, \dots, 25\}$ – множина студентів у групі. Побудувати відповідність між елементами цих множин.

Розв'язок. Кожний студент під час складання іспиту може одержати тільки одну оцінку. Водночас та сама оцінка може бути поставлена деякій підмножині студентів. Тому між цими множинами можна побудувати багато-однозначну відповідність, як показано на рис. 11.

Багато-багатозначна відповідність

Багато-багатозначна відповідність – це така відповідність між елементами двох множин *A* і *B*, коли з одним елементом першої множини зіставлено більш ніж один елемент другої множини і навпаки.

Приклад 1.34. Нехай *A* – множина театральних постановок, а *B* – множина глядачів. Кожний глядач може подивитися деяку підмножину театральних постановок. Водночас кожну з театральних постановок відвідує деяка підмножина глядачів. Графічне представлення такої відповідності показане на рис. 12.

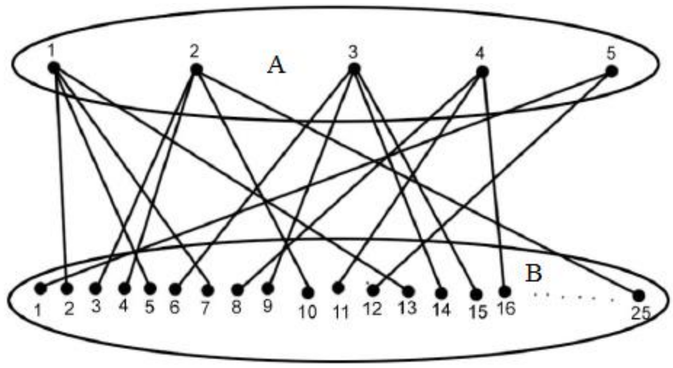


Рис. 11. Графічне представлення багато-однозначної відповідності між елементами множини *A* та *B*

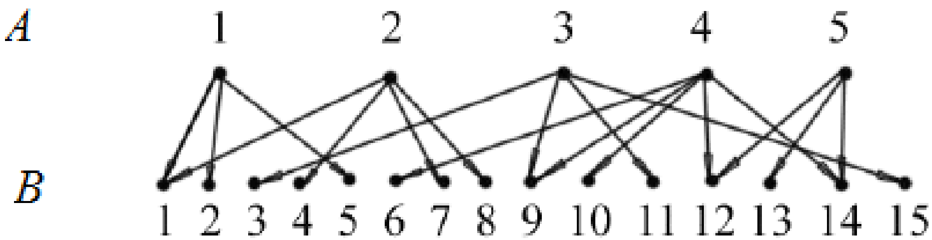


Рис. 12. Графічне представлення багато-багатозначної відповідності між елементами множини *A* та *B*

Поняття відношення

Фундаментальним поняттям дискретної математики є поняття «***відношення***», яке використовують для позначення зв’язку між об’єктами або поняттями.

Наприклад, властивість елемента *a* належати ($a \in A$) множині *A*. На множині людей можна задати родинні відношення, наприклад «*x* сестра *y*».

Причому, якщо взяти конкретних людей і підставляти їх імена замість x і y, то отримаємо або справедливе відношення, або хибне.

Нехай задано декартовий добуток непорожніх множин $A_1 \times A_2 \times \ldots \times A_n$. Підмножина $\rho \subset A_1 \times A_2 \times \ldots \times A_n$, називається n-місним відношенням на множині $A_1 \times A_2 \times \ldots \times A_n$, або кортежем. Інакше кажучи, $a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n$ перебувають у відношенні ρ , якщо $(a_1, a_2, \ldots, a_n) \in \rho$.

Якщо $n = 1$, відношення ρ є підмножинами множини A, їх називають ***одномісними***, або ***унарними***. Властивості унарного відношення ρ – це властивості самої підмножини A, тому термін відношення з $n = 1$ вживається рідко. Якщо $n = 2$, то відношення називають ***бінарним***. Бінарні відношення називаються основними й використовуються найчастіше.

Бінарні відношення. Відношення між парами об'єктів називають двомісними або ***бінарними***. Наприклад «x дільник y», «x студент групи y».

Бінарним відношенням між елементами множин $a \in A$ та $b \in B$ називають підмножину ρ множини $A \times B (\rho \subset A \times B)$. Позначають: $a\rho b$ або $(a, b) \in \rho$. Читається «а перебуває у відношенні з в».

Областю визначення відношення ρ (позначається D_ρ) називають множину перших координат (а) елементів із ρ , ***областю значень*** (позначаються μ_ρ) називають множину других координат (в) елементів з ρ.

Доповненням бінарного відношення R між елементами A та B вважається множина $-R = \overline{R} = (A \times B) \setminus R$.

Оберненим відношенням для бінарного відношення R називається множина $R^{-1} = \{(a, b) \mid (b, a) \in R\}$.

Образом множини A відносно R називають множину $R^{-1}(A)$.

Добутком (композицією) відношень $R_1 \subseteq A \times B$ та $R_2 \subseteq B \times C$ називається відношення $R_1 \circ R_2 = \{(a, b) \mid$ існує c, таке, що $(a, c) \in R_1$ та $(c, b) \in R_2\}$.

Приклад 1.35. $P = \{(a, b), (a, c), (b, a), (c, d)\}$,
$Q = \{(a, b), (b, c), (c, a), (c, d)\}$, то

$$P \circ Q = \{(a, c), (a, a), (a, d), (b, b)\},$$
$$Q \circ P = \{(a, a), (b, d), (c, b), (c, c)\}.$$

Приклад 1.36. Розглянемо відношення ρ на множинах $A = \{1,7,8,9\}$ та $B = \{5, 6, 10\}$, ρ – відношення «а більше за в $(a > b)$.

Тоді $\rho = \{(7,5),(7,6),(8,5),(8,6),(9,5),(9,6)\}$. $D_\rho = \{7, 8, 9\}$. $\mu_\rho = \{5, 6\}$.

Властивості обернених відношень:

1) $(R^{-1})^{-1} = R$;

2) якщо $R \subset S$, то $R^{-1} \subset S^{-1}$;

3) $\overline{R}^{-1} = \overline{R^{-1}}$;

4) $(R \cap S)^{-1} = R^{-1} \cap S^{-1}$;

5) $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$.

Способи задавання бінарних відношень

Оскільки бінарні відношення є множинами, то для їх задавання можна використовувати ті самі способи, що і для множин. Крім того, якщо бінарні відношення задані нескінченними множинами, то їх можна задавати за допомогою ***матриць відношень*** і ***графів*** (діаграм) відношень.

1. Задавання відношення переліком

Бінарне відношення можна задати, перераховуючи всі пари, які до нього входять (якщо відношення складається зі скінченної кількості пар), або вказавши загальну властивість пар, що належать цьому відношенню, тобто предикатом (згадайте способи задавання множин).

Приклад 1.37. Нехай дано множину $A = \{p, r, s, q\}$. Задати відношення $R \subseteq A \times A$ перерахуванням пар.

Розв'язок $R = \{(p,r),(s,q),(r,p),(p,p),(s,r),(p,s)\}$.

Приклад 1.38. Нехай дано множину натуральних чисел *N*. Задати відношення, вказавши загальну властивість пар, що належать відношенню.

Розв'язок. $R = \{(n,m) \in N \times N | n - дільникm\}$.

2. Матричний спосіб

За ***матричного способу задання відношення*** елементам множини *A* ставляться у відповідність рядки матриці *M R*, елементам множини *B* — стовпці. Якщо пара (*a*, *b)* перебуває у відношенні *R*, то на перетині відповідного їм рядка та стовпця матриці записується одиниця, інакше — нуль.

Введемо символ $\delta_{ij} = \begin{cases} 1, aRb \\ 0, a\overline{R}b \end{cases}$.

Бінарне відношення задається двовимірною таблицею – матрицею суміжності, якій взаємоднозначно зіставляють елементи множини *A* та *B*.

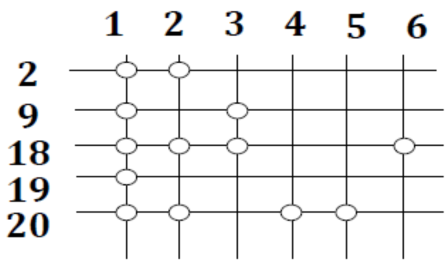
Кожна клітинка (i,j) відповідає елементам множини $A \times B$. Якщо aRb, то в клітинці $(i,\ j)$ ставлять одиницю, якщо $a\overline{R}b$ – нуль.

Приклад 1.39. Задати бінарне відношення на множинах матричним способом $A = \{1, 2, 3, 4, 5, 6\}$, $B = \{2, 9, 18, 19, 20\}$; $R = \{(a,\ b) | a \in A,\ b \in B;$ a –дільник $b\}$.

$$
\begin{array}{cc}
 & \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} \end{array} \\
\mathrm{C} = & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{array}{l} \mathbf{2} \\ \mathbf{9} \\ \mathbf{18} \\ \mathbf{19} \\ \mathbf{20} \end{array}
\end{array}
$$

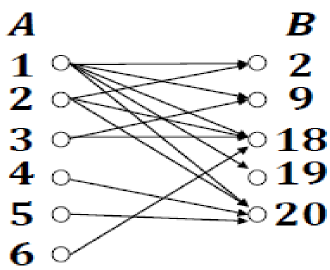
3. Табличний спосіб

Для табличного задання відношення $R \subset A \times B$ проводять вертикалі, кожну з яких позначають деяким елементом з A і горизонталі, позначаючи їх елементами з B. Потім точками позначають перетин тих прямих, які задовольняють відношення R.



4. Стрілочне зображення

У стрілочному зображенні відношення $R \subset A \times B$ елементи A та B позначаються точками, після чого спрямованими від a до b стрілками, з’єднуються ті і тільки ті з них, де $a \in A$ та $b \in B$, для яких $(a,\ b) \in R$.



5. Фактор-множина

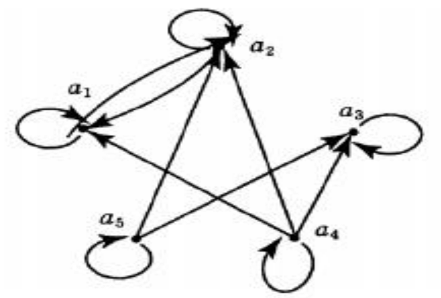
Сукупність всіх елементів $b \in B$, для яких $(a,\ b) \in R$, називають **перерізом (перетином)** відношення R за елементом a. Позначують Pr_a.

Сукупність всіх перерізів відношення R за елементами множини A називають **фактором** або **фактор-множиною** множини B.

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ \{2,9,18,19,20\} & \{2,18,20\} & \{9,18\} & \{20\} & \{20\} & \{18\} \end{pmatrix}$$

6. Графічний спосіб

Цей спосіб передбачає побудову ***графа*** відношення.



Приклад 1.40. Нехай існує відношення R, яке задано переліком: $R = \{(p,r),(s,q),(r,p),(p,p),(s,r),(p,s)\}$.Представити це відношення у вигляді графа.

Розв'язок.

На рис. 13 зображено граф бінарного відношення:

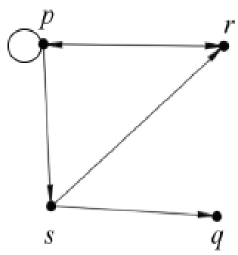


Рис. 13. Граф бінарного відношення R

Операції над відношеннями

Оскільки відношення є множинами, елементами яких є впорядковані пари, то над ними можна виконувати всі відомі операції над множинами.

$P = \{(a,b),(a,c),(b,a),(c,d)\}$, а $Q = \{(a,b),(b,c),(c,a),(c,d)\}$, то

$$P \cup Q = \{(a,b),(a,c),(b,a),(b,c),(c,a),(c,d)\}$$

$$P \cap Q = \{(a,b),(c,d)\}$$

$$P \backslash Q = \{(a,c),(b,a)\}$$

$$P \oplus Q = \{(a,c),(b,a),(b,c),(c,a)\}$$

Крім того, над відношеннями визначені операції знаходження оберненого відношення та композиції відношень.

Бінарне відношення $R^{-1} = \{(y,x)|(x,y) \in R\}$, визначене на множинах B та A, називається ***оберненим*** до відношення $R \subseteq A \times B$.Наприклад, для відношення, діаграму якого наведено на рис. 14, $R^{-1} = \{(1,a),(2,a),(1,d),(4,b),(4,f)\}$.

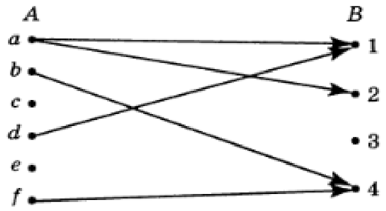


Рис. 14. Граф відношення R

Бінарне відношення $S \circ R = \{(x,z)|\textit{існує такий } y \in B, \textit{що } (x,y) \in R, (y,z) \in S\} \subseteq A \times C$
називається ***добутком*** **(композицією)** відношень $R \subseteq A \times B$ та $S \subseteq B \times C$.

Наприклад, для відношень R та S, зображених на рис. 15, $S \circ R = \{(a,x),(a,y)\}$.

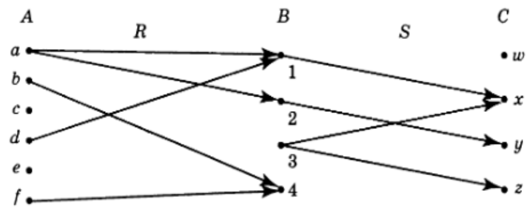


Рис. 15. Граф відношень R та S

Приклад 1.41. Нехай $A = \{2,3,4,5,6,7,8,9\}$, $B = \{23,24,25,26,27\}$, $C = \{a,b,c,d,e,f\}$, Бінарне відношення R визначене на множинах A та B таким чином: тоді й тільки тоді, коли y націло ділиться на x, xRy,
$S = \{(23,f),(24,b),(25,d),(26,d),(27,a),(27,e)\} \subseteq B \times C$.

Знайти першу та другу проєкції бінарного відношення та вказати $T = S \bullet R$ та вказати $T^{-1}[\{a,c,d,f\}]$.

Розв'язок. Задамо бінарне відношення R переліком елементів, які перебувають у цьому відношенні:
$R = \{(2,24),(2,26),(3,24),(3,27),(4,24),(5,25),(6,24),(8,24),(9,27)\}$,
тоді $T = \{(2,b),(2,d),(3,b),(3,a),(3,e),(4,b),(5,d),(6,b),(8,b),(9,a),(9,e)\}$.

Тому $Pr_1\, T = \{2,3,4,5,6,8,9\}$; $Pr_2\, T = \{a, b, d, e\}$. Вкажемо перелік елементів відношення T^{-1}:

$T^{-1} =$
$\{(b, 2), (d, 2), (b, 3), (a, 3), (e, 3), (b, 4), (d, 5), (b, 6), (b. 8), (a, 9), (e, 9)\}$.

Тоді $T^{-1}[\{a, c, d, f\}] = \{2,3,5,9\}$.

Властивості композиції відношень

Композиція відношень **асоціативна**: тобто якщо X, Y, Z, D – множини і якщо $R \subseteq X \times Y, S \subseteq Y \times Z$ і $T \subseteq Z \times D$, тоді
$R \circ (S \circ T) = (R \circ S) \circ T$.

Спеціальні властивості відношень

Бінарне відношення $\{(a, a) | a \in A\}$ називається ***відношенням ідентичності*** на множині A.

Нехай $C = A \times B$, aRb, $R \subseteq C$.

1. ***Рефлексивність.*** Якщо для довільного елемента c виконується cRc, тобто елемент $c \in C$ перебуває у відношенні R до самого себе, відношення R називається **рефлексивним.**

Приклад 1.42. $\boldsymbol{R = \{(1, 1), (1, 2), (2, 2), (3, 3)\}}$ рефлексивне на множині $A = \{1,2,3\}$, проте не рефлексивне на множині $A' = \{1, 2, 3, 4\}$.

Прикладами рефлексивних відношень є:

- = – дорівнює;
- ≤ – менше або дорівнює;
- ≥ – більше або дорівнює;
- $\subseteq$ – є підмножиною або дорівнює.

Бінарне відношення є рефлексивним, якщо на його діаграмі кожна вершина з’єднана «петлею» із самою собою.

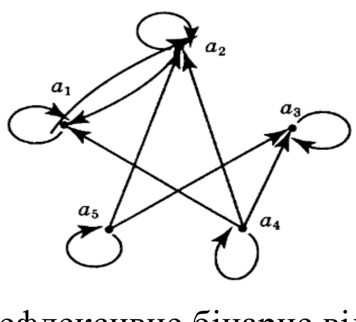


Рис. 16. Рефлексивне бінарне відношення

2. ***Антирефлексивність (іррефлексивність).*** Якщо із $c_1 R c_2$ слідує, що $c_1 \neq c_2$, відношення R називають ***антирефлексивним***, *або* ***іррефлексивним.***

Приклад 1.43. Відношення $S = \{(1,2)(1,3),(2,1),(3,2)\}$ антирефлексивне у множині $A = \{1,2,3\}$.

Прикладами антирефлексивних відношень є:

- $\neq$ – не дорівнює;
- < – менше;
- > – більше;
- $\subset$ – є підмножиною;
- «бути старшим» у множині людей;
- «бути батьком».

Порожнє відношення прийнято вважати як рефлексивним, так і антирефлексивним. Якщо відношення є ні рефлексивним, ні антирефлексивним, то його називають ***нерефлексивним***.

Бінарне відношення, діаграму якого наведено на рис. 17, не є рефлексивним, оскільки на вершинах a_3 та a_5 відсутні петлі.

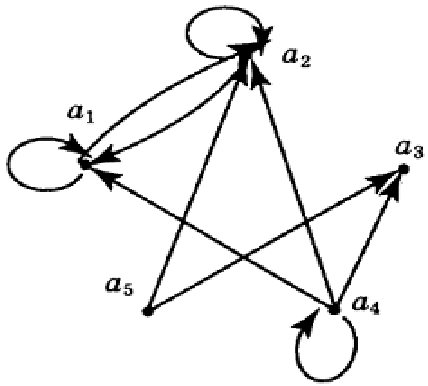


Рис. 17. Приклад діаграми однорідного нерефлексивного відношення

Приклад 1.44. Відношення $P = \{(1,1),(1,3)(2,1),(3,3)\}$ не рефлексивне, оскільки елемент *2*, на відміну від всіх інших, не перебуває у відношенні сам із собою $(2,2) \notin P$.

3. ***Симетричність.*** Якщо для пари $c_1 R c_2$ слідує, що $c_2 R c_1$, то відношення R називають ***симетричним***.

Приклад 1.45. Відношення $R = \{(1,1),(1,2),(2,1),(2,3),(3,2)\}$ симетричне. Симетричними є відношення паралельності, перпендикулярності, подібності, універсальне відношення тощо. Для симетричного відношення його графік симетричний відносно діагоналі – бісектриси координатного кута.

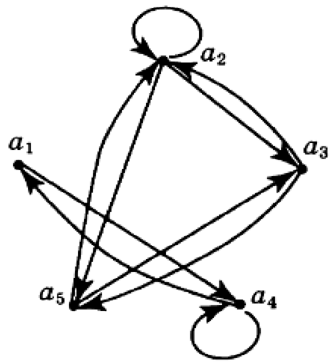


Рис. 18. Приклад діаграми симетричного бінарного відношення

4. ***Антисиметричність.*** Якщо із c_1Rc_2 та c_2Rc_1 слідує $c_1 = c_2$, відношення R називають ***антисиметричним***.

Приклад 1.46. Відношення $R = \{(1,1), (1,2), (1,4), (2,2), (2,4), (4,4)\}$ антисиметричне: є пари (c_1,c_1) і немає пар (c_1,c_2), (c_2,c_1).

Антисиметричними є відношення включення, «менше», «більше», «менше дорівнює» тощо.

5. ***Асиметричність.*** Якщо із c_1Rc_2 не виконується c_2Rc_1, то відношення R називають ***асиметричним***.

Приклад 1.47. Відношення $S = \{(1,2), (2,3), (3,4)\}$ асиметричне. Асиметричними є відношення включення, «менше», «більше» тощо. Відношення рівності, діагональне та порожнє вважають як симетричними, так і антисиметричними.

6. ***Транзитивність***. Якщо c_1Rc_2 та c_2Rc_3 слідує, що c_1Rc_3, то відношення R називають ***транзитивним***.

Приклад 1.48. Відношення $R = \{(1,1), (1,2), (1,4), (2,2), (2,4), (4,4)\}$ транзитивне. Приклади транзитивних відношень:

- відношення часткового порядку:
  - строга нерівність: ($a < b$), ($b < c$) $\Rightarrow$ ($a < c$);
  - нестрога нерівність: $(a \leq b)$, $(b \leq c) \Rightarrow (a \leq c)$;
- включення підмножини:
  - строга підмножина $(A \subset B; B \subset C) \Rightarrow (A \subset C)$;
  - нестрога підмножина $(A \subseteq B; B \subseteq C) \Rightarrow (A \subseteq C)$;

- подільність;
  - $(a|b)$, $(b|c) \Rightarrow (a|c)$;
  - $(a{:}\,b)$, $(b{:}\,c) \Rightarrow (a{:}\,c)$;
- рівність: $(a = b)$, $(b = c) \Rightarrow (a = c)$;
- еквівалентність: $(a \Leftrightarrow b)$, $(b \Leftrightarrow c) \Rightarrow (a \Leftrightarrow c)$;
- імплікація: $(a \Rightarrow b)$, $(b \Rightarrow c) \Rightarrow (a \Rightarrow c)$;
- паралельність: $(a||b)$, $(b||c) \Rightarrow (a||c)$;
- відношення подібності геометричних фігур;
- бути предком.

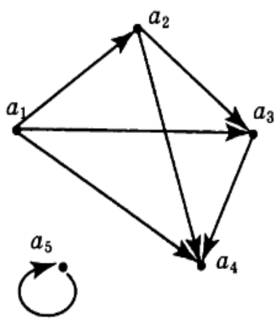


Рис. 19. Приклад діаграми транзитивного відношення

7. ***Антитранзитивність.*** Якщо $c_1 R c_2$ та $c_2 R c_3$ виходить, що не виконується $c_1 R c_3$, то відношення R називають ***антитранзитивним***: для будь-яких трійок a, b, c відсутня транзитивність.

Приклад 1.49.

- Відношення перемогти в турнірах «на виліт»: якщо A переміг гравця B, а B переміг гравця C, то A не грав з C, отже, не міг його перемогти.
- Бути сином (батьком, бабусею).
- Гра «Камінь, ножиці, папір». Камінь перемагає ножиці, ножиці виграють у паперу, але камінь програє паперові і т. д.
- Харчовий ланцюжок – це відношення не завжди є транзитивним (приклад: вовки їдять оленів, олені їдять траву, але вовки не їдять траву).
- Бути переважніше ніж. Якщо ми хочемо яблуко замість апельсина, а замість яблука ми б хотіли кавун, то це не означає, що ми віддамо перевагу кавуну.
- Бути другом.
- Бути колегою по роботі.

- Бути підлеглим. Наприклад, у часи феодального ладу в Західній Європі була в ходу приказка: «Васал мого васала – не мій васал».
- Бути схожим на іншу людину.

Бінарне відношення R називається ***лінійним*** на множині A, якщо для довільних відмінних один від одного a A, b B хоча би одна з пар a, b, b, a є елементом відношення R.

Замиканням бінарного відношення R за властивістю P називається таке мінімальне за числом елементів бінарне відношення $[R]_p$, яке містить у собі відношення R і задовольняє властивість P.

Приклад 1.50. На множині $A = \{1, 2, 3, 4\}$задано відношення:
$R_1 = \{(1,1),(1,2),(2,1),(2,2),(3,4),(4,1),(4,4)\}$;
$R_2 = \{(1,1),(1,2),(2,1)\}$;
$R_3 = \{(1,1),(1,2),(1,4),(2,1),(2,2),(3,3),(4,1),(4,4)\}$;
$R_4 = \{(2,1),(3,1),(3,2),(4,1),(4,2),(4,3)\}$;
$R_5 = \{(1,1),(1,2),(1,3),(1,4),(2,2),(2,3),(2,4),(3,3),(3,4),(4,4)\}$;
$R_6 = \{(3,4)\}$.

Визначити, які з цих відношень (а) рефлексивні; (б) антирефлексивні; (в) симетричні; (г) антисиметричні; (д) транзитивні.

Розв'язок.R_3і R_5– рефлексивні, оскільки вони містять всі пари вигляду (а, а), тобто (1, 1), (2, 2), (3, 3), (4, 4). Усі інші не є рефлексивними. R_1, R_2, R_4, R_6 не містять пару (3, 3).

Відношення R_2, R_3 – симетричні. В R_2є пари (1, 2) і (2, 1). В R_3 є пари (1, 2), (2, 1), (1, 4), (4, 1).

Відношення R_5 – антисиметричні, бо є пари (с1, с1) і немає пар (с1, с2), (с2, с1).

Відношення R_6, R_4 – асиметричні, бо якщо із с1Rс2 не виконується с2Rс1, то відношення R називають *асиметричним.*

Властивості симетричності й антисиметричності не є антагоністичними. Відношення $R = \varnothing$ на множині $A = \{a\}$ одночасно і симетричне, і антисиметричне. R_1 не має властивостей ані симетричності, ані анти симетричності.

Асиметричне відношення є і антисиметричним. Обернене твердження неправильним. Відношення R_5 є антисиметричним відношенням, але не є асиметричним, бо містить пару (1, 1).

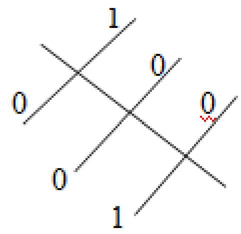
R_4, R_5, R_6 – транзитивні. R_1, R_2, R_3 – не транзитивні, бо

$(3,4) \in R_1; (4,1) \in R_1$, але $(3,1) \notin R_1$; $(2,1) \in R_2; (1,2) \in R_2$ але $(2,2) \notin R_2$; $(2,1) \in R_3$; $(1,4) \in R_3$; але $(2,4) \notin R_3$.

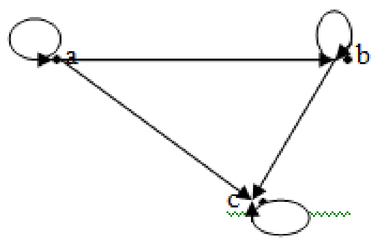
Відношення R на множині A називають ***іррефлексивним***, якщо для будь-якого $a \in A$ $(a,a) \notin R$; R_4, R_6 – іррефлекствні; R_1, R_2 – нерефлексивні і неіррефлексивні.

Якщо відношення рефлексивне, то на головній діагоналі матриці відношення будуть перебувати «1», якщо іррефлексивне, то на головній діагоналі – «0».

Матриця симетричного відношення симетрична відносно головної діагоналі. Матриця антисиметричного відношення має таку властивість: якщо $i \neq j$, то з $m_{i,j} = 1$ випливає $m_{j,i} = 0$.



Граф рефлексивного відношення має петлю в кожній вершині, у графі транзитивного відношення в разі наявності пари дуг (a, b) (b, c) існує дуга (a, c).



1.3. Відношення еквівалентності

Визначення відношення еквівалентності

Деякі елементи множини можна розглядати як еквівалентні в тому випадку, коли кожний із цих елементів за деякого розгляду може бути замінений іншим. У цьому випадку говорять, що ці елементи перебувають у відношенні еквівалентності.

Бінарне відношення називається відношенням ***еквівалентності*** на множині A, якщо воно є рефлексивним, симетричним і транзитивним.

Бінарне відношення еквівалентності *розбиває* множину A на множини, які не мають спільного перетину (***класи еквівалентності***).

Граф відношення еквівалентності $R = \{(a,a),(a,c),(b,b),(c,a),(c,c),(d,d)\}$ наведено на рис 20 (у випадку однорідних відношень кожний елемент множини, на якій визначене відношення, достатньо зображати один раз).

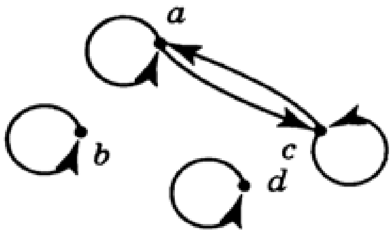


Рис. 20. Граф відношення еквівалентності

Властивості еквівалентних відношень

1. Властивість **рефлексивності** проявляється в тому, що кожний елемент еквівалентний самому собі або $x \equiv x$.

2. Висловлювання про те, що два елементи є еквівалентними, не вимагає уточнення, який з елементів розглядається першим, який – другим, тобто наявною є $x \equiv y \rightarrow y \equiv x$ – властивість **симетричності.**

3. Два елементи, еквівалентні третьому, еквівалентні між собою, або наявною є $x \equiv y$ і $y \equiv z \rightarrow z \equiv z$ – властивість **транзитивності.**

Як загальний символ відношення еквівалентності використовується символ «≡» (іноді символ «~»). Для окремих відношень еквівалентності використовуються інші символи:

«=» – для позначення рівності;

«||» –для позначення паралельності;

«↔»– для позначення логічної еквівалентності.

Приклад 1.51. Визначити, які із зазначених бінарних відношень є відношеннями еквівалентності, та вказати для них класи еквівалентності:

а) перпендикулярність площин у просторі;

б) відношення «бути однакового зросту» на множині людей;

в) відношення «перебувати один від одного на відстані не меншій за 100» на площині;

г) відношення «бути родичем» на множині людей (вважаємо, що людина є родичем сама собі, а дві людини є родичами, якщо одна з них є нащадком іншої, або вони мають спільного предка).

Розв'язок: а) відношення перпендикулярності площин не є відношенням еквівалентності, оскільки воно не є рефлексивним;

б) відношення «бути однакового зросту» на множині людей є відношенням еквівалентності.

Рефлексивність, очевидно, справджується, оскільки відношення рівності чисел є рефлексивним.

Симетричність також виконується з тієї самої причини.

Для перевірки транзитивності досить пересвідчитися у транзитивності відношення рівності чисел. Якщо вважати, що зріст вимірюється в сантиметрах, то класом еквівалентності, який відповідає числу k, є множина людей зросту k см;

в) відношення не є відношенням еквівалентності, оскільки не виконується умова транзитивності. Для того щоб пересвідчитися і цьому, досить розглянути вершини рівнобедреного трикутника з бічною стороною 100 та основою 50. Відстані від кінців основи до вершини задовольняють умову, а довжина основи – не задовольняє;

г) відношення не є відношенням еквівалентності. Рефлексивність і симетричність випливають з означення. Покажемо, що транзитивність не виконується. Нехай різні люди А та В є родичами, і нехай В та С також є родичами, причому А предок В по батьківській лінії, а С – предок В по материнській лінії. Тоді жоден із людей А та С не є родичем іншого. Отже, транзитивність не виконується.

Приклад 1.52. Перевірити, чи є визначене на множині $A = \{x, y, z, t, u, v, w\}$ бінарне відношення $R =$

$$\left\{\begin{matrix}(u,x),(u,u).(y,z),(w,w),(y,y),(z,y),(z,z),(z,w),(y,w),(x,u).(w,y),\\(w,z),(x,x),(v,v),(t,t)\end{matrix}\right\},$$

відношенням еквівалентності. Якщо так, то вказати фактор-множину A/R.

Розв'язок. Оскільки $I_A \subseteq R$, то відношення R є рефлексивним.

Для перевірки симетричності знайдемо обернене відношення: $R^{-1} =$

$$\left\{\begin{matrix}(x,u),(u,u),(z,y),)w,w),(y,y),(y,z),(z,z),(w,z),(w,y),(u,x),(y,w),\\(z,w),(x,x),(y,y),(t,t)\end{matrix}\right\}.$$

Легко переконатися, що $R^{-1} = R$а, отже, відношення R є симетричним.

Для перевірки транзитивності знайдемо другу степінь відношення R: $R^2 =$

$$\left\{\begin{matrix}(x,x),(x,u),(y,y),(y,z),(y,w),(z,y),(z,z),(z,w),(t,t),(u,x),(u,u),\\ (v,v),(w,y),(w,z),(w,w)\end{matrix}\right\}.$$

Нескладно перевірити, що $R^2 = R$. Тому відношення R є транзитивним.

Отже, бінарне відношення R є відношенням еквівалентності. Знайдемо тепер класи еквівалентності. Для цього потрібно вказати одноелементні зрізи відношення R.

$R[x] = \{x,u\}; R[y] = \{y,z,w\}; R[z] = \{y,z,w\}: R[t] = \{t\}"R[u] =$
$= \{x,u\}: R[v] = \{v\}"R[w] = \{y,z,w\}.$
Отже, $A\backslash R = \big\{\{x,u\},\{y,z,w\},\{t\},\{v\}\big\}$.

Класи еквівалентності

Відношення еквівалентності R_e– це відношення на множині *A*, яке розбиває цю множину на підмножини, елементи яких еквівалентні один одному, але не еквівалентні елементам інших підмножин.

Класами еквівалентності називають підмножини, що не перетинаються й отримані внаслідок розбиття множини *A* відношенням еквівалентності R_e. Множину класів еквівалентності множини *A* відносно R_e називають **фактор-множиною** і позначають $[A]_R$.

Приклад 1.53. Розбиття множини на підмножини. Нехай множина *B* – це набір різнокольорових повітряних кульок.

1. Відношення R_1 задамо умовою:

$(a,b) \in R_1$, якщо *«a* одного кольору з *b»*.

Рис. 21. Множина кульок *B*

Одержимо класи еквівалентності з кульок одного кольору.

2. Відношення R_2 задамо умовою:

$(a, b) \in R_2$, якщо «a одного розміру з b».

Одержимо класи еквівалентності з кульок одного розміру.

3. Відношення R_3 задамо умовою: $(a, b) \in R_3$, якщо «a однакової форми з b».

Одержимо класи еквівалентності з кульок однакової форми.

Нехай i $a \in A$ – елемент множини $A = \{a_1, a_2, \ldots, a_i, \ldots, a_n\}$. Тоді $[a_i]$ позначає множину $\{x|xRa_i\} = \{x|(x, a_i) \in R\}$, *яку* називають **класом еквівалентності**, що містить a_i. Символ $[A]_R$ позначає множину всіх класів еквівалентності множини A відносно R. Таким чином, $[A]_R$ – фактор-множина.

Приклад 1.54. Нехай A = {1, 2, 3, 4, 5, 6} і дано відношення еквівалентності $R = \begin{Bmatrix} (1,1),(2,2),(3,3),(4,4),(5,5),(6,6),(1,2), \\ (1,4),(2,1),(2,4),(3,5),(5,3),(4,1),(4,2) \end{Bmatrix}$

Класи еквівалентності відношення R були отримані шляхом визначення класу еквівалентності кожного елемента множини A:

$[1] = \{x|(x,1) \in R\} = \{x|xR1\} = \{1,2,4\}$, де 1 ∈ [1], оскільки (1, 1) ∈ R, 2 ∈ [1], оскільки (2, 1) ∈ R, 4 ∈ [1], оскільки (4, 1) ∈ R, не існує жодного іншого $x \in A$ такого, що (x, 1) ∈ R. Так само одержуємо

[2] = {x|(x, 2) ∈ R} = {x|xR2} = {2, 1, 4}

[3] = {x|(x, 3) ∈ R} = {x|xR3} = {3, 5}

[4] = {x|(x, 4) ∈ R} = {x|xR4 } = {4, 1, 2}

[5] = {x|(x, 5) ∈ R} = {x|xR5} = {5, 3}

[6] = {x|(x, 6) ∈ R} = {x|xR6} = {6}.

Замикання множин

Множину A називають замкнутою відносно деякої операції, якщо результатом виконання цієї операції над елементами множини A завжди буде елемент, який належить множині A.

Приклад 1.55. Нехай множина N – множина натуральних чисел. Розглянемо операцію «+» на множині N. Чи є множина N замкнутою відносно операції «+»?

Розв'язок. Нехай $n \in N$ і $m \in N$. Тоді $n + m = k \in N$ для будь-яких n, $m \in N$.

1.4. Відношення порядку

Визначення відношень порядку

Бінарне відношення R називається **відношенням порядку (порядком) на множині A**, якщо воно є антисиметричним і транзитивним. Пара (A, R) називається **впорядкованою *множиною***. Якщо a та b — елементи впорядкованої множини (A, R) і виконується умова aRb, то кажуть, що елемент a передує елементу b.

Види відношень порядку

Якщо порядок є рефлексивним, то він називається **частковим (нестрогим) порядком.** Прикладом є відношення «≤» на множині дійсних чисел. Антирефлексивний порядок називається **строгим порядком**. Прикладом є відношення «⊂» (відношення строгого включення множин). Відношення R є строгим порядком тоді й тільки тоді, коли воно є одночасно асиметричним і транзитивним. Якщо R – відношення строгого порядку на множині A, то відношення $R' = R \cup I$ називається відношенням часткового порядку, відповідним відношенню R.

Відношення порядку, яке є лінійним, називається відношенням ***лінійного*** порядку.

Прикладом строгого лінійного порядку є відношення «>» на числовій множині.

Діаграма Хассе

Відношення часткового порядку на скінченній множині зручно задавати за допомогою діаграм Хассе. При цьому кожний елемент з'єднується відрізками з усіма його «безпосередніми попередниками» й розташовується на діаграмі вище за них.

На рис. 22 зображено діаграму Хассе для відношення подільності $(x, y) \in R$ тоді й тільки тоді, коли число x є дільником числа y на множині {2, 4, 5, 10, 12, 20, 25}.

Елемент a називається **мінімальним елементом** впорядкованої множини (A, R), якщо не існує такого елемента $b \in A$, що виконуються умови $b \neq a$ і bRa.

Аналогічно дається означення **максимального елемента** впорядкованої множини.

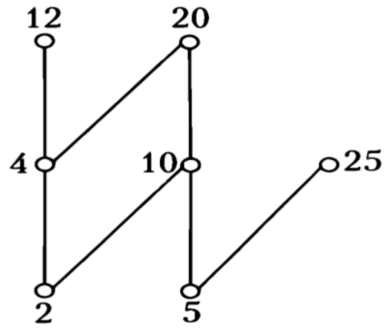


Рис. 22. Діаграма Хассе для відношення подільності на множині
{2, 4, 5, 10, 12, 20, 25}

Для відношення подільності із діаграмою на рис. 22 елементи 2 та 5 є мінімальними, а елементи 12, 20 і 25 — максимальними.

Елемент a називається **найменшим елементом** впорядкованої множини (A, R), якщо для довільного елемента $b \in A$ $(a, b) \in R$. Аналогічно дається означення **найбільшого елемента** впорядкованої множини.

Найбільший, найменший, максимальні та мінімальні елементи називають **екстремальними елементами** впорядкованої множини.

Для часткового впорядкованої множини, діаграму якої наведено на рис. 23, елемент G буде найбільшим, а найменшого елемента взагалі не існує (елементи A, C та E —мінімальні, але не найменші, оскільки жодний із них не передує двом іншим).

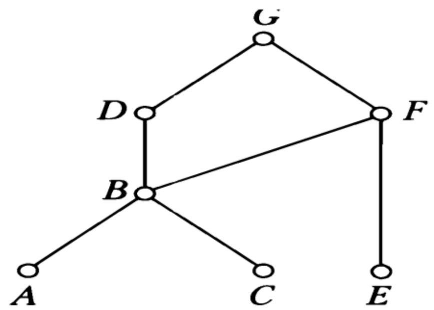


Рис. 23. Діаграма Хассе для відношення часткового порядку

Елемент a впорядкованої множини (A, R) називається **нижньою гранню множини** $M \subseteq A$, якщо для всіх елементів $b \in M$ виконується умова aRb. Аналогічно дається означення верхньої грані.

Найбільша нижня грань множини (якщо вона існує) називається **точною нижньою гранню** множини M і позначається $\inf M$.

Точна верхня грань (найменша верхня грань) множини M позначається sup$M$. Так, наприклад для відношення, діаграму якого наведено на рис. 10, $\sup\{D, E\} = G$, $\inf\{D,F\} = B$, а $\inf\{B, E\}$не існує.

Частково впорядкована множина (A, R) називається ґраткою, якщо для довільних $a \in A$, $b \in A$ існують $\inf\{a, b\}$та $\sup\{a, b\}$.

Приклад 1.56. Перевірити, чи є бінарне відношення $R = \{(a,a),(b,d),(b,b),(a,e),(c,c),(a,b),(a,d),(c,d),(d,d),(b,c),(a,c)\}$ відношенням часткового порядку на множині $A = \{a, b, c, d, e\}$. Якщо так, то зобразити діаграму Хассе впорядкованої множини (A, R), відшукати її екстремальні елементи й перевірити, чи є впорядкована множина A, R ґраткою. Крім того, знайти $\inf\{b, c\}$ та $\sup\{b, c\}$.

Розв'язок. Відношення R є рефлексивним. Перевіримо, чи є воно антисиметричним. Знайдемо обернене йому відношення $R^{-1} = \{(d,b),(e,a),(b,a),(d,a),(d,c),(c,b),(c,a)\}$. Тоді $R^{-1} \cap R = I_A$, а отже, відношення R є антисиметричним. Перевіримо транзитивність.

$R^2 = \{(a,a),(a,b),(a,c),(a,d),(a,e),(b,b),(b,d),(b,e\},(c,c),(c,d),(d,d),(e,e)\}$ $=R$. Тому відношення R є транзитивним.

Отже, відношення R є відношенням часткового порядку. Зобразимо діаграму Хассе частково впорядкованої множини (A, R). Відповідну діаграму наведено на рис. 24.

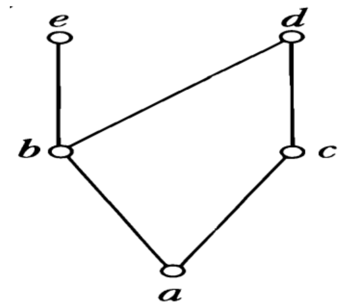


Рис. 24. Діаграма Хассе

З діаграми Хассе видно, що елемент a є найменшим елементом впорядкованої множини (A, R), а отже, єдиним мінімальним елементом. Елементи e та d – максимальні елементи впорядкованої множини (A, R), а найбільшого елемента не існує. Оскільки $\sup\{e, d\}$ не існує, то впорядкована множина (A, R) не є ґраткою. Нарешті, з діаграми Хассе видно, що $\inf\{b, c\} = a$, $\sup\{b, c\} = d$.

Обов'язкові завдання

1. Нехай задано множини $A = \{a, b, c, d\}; B = \{1, 2, 3, 4, 5\}: G = \{\alpha, \beta, \gamma\}$, відповідності між A і B:

$C_1 = \{(a, 2), (a, 3), (b, 1), (c, 4), (c, 5), (d, 2), (d, 3\}$;

$C_2 = \{(b, 3), (b, 5), (c, 1), (c, 4), (d, 1), (d, 4), (d, 5)\}$

і відповідності між B і G:

$D_1 = \{(1, \beta), (1, \gamma), (2, \alpha)(2, \gamma), (3, \gamma), (5, \alpha), (5, \beta)\}$;

$D_2 = \{(1, \alpha), (2, \alpha), (2, \beta), (3, \alpha), (4, \alpha), (4, \gamma)\}$.

Визначити (а) $C_i \circ D_j, i, j = 1, 2$;

(б) $C_i \circ C_j^{-1}, i, j = 1{,}2$; (в) $C_2 \circ (D_1 \circ D_2^{-1})$.

2. На множині $M = \{1, 2, 3, 4\}$ задано відношення

$R_1 = \{(1, 1)(1, 3), (2, 2), (2, 4), (3, 1), (3, 3), (4, 2), (4, 4)\}$;

$R_2 = \{(1, 1), (2, 2), (2, 3), (2, 4), (3, 1), (3, 3), (4, 1), (4, 4)\}$;

$R_3 = \{(1, 3), (1, 4), (2, 1), (1, 2), (3, 1), (3, 3), (4, 1)\}$;

$R_4 =$
$\{(1, 1), (1, 2), (1, 4), (2, 2), (2, 4), (3, 3), (3, 4), (4, 2), (4, 3), (4, 4)\}$;

$R_5 = \{(1, 2), (1, 3), (2, 3), (2, 4), (4, 1), (4, 3)\}$.

Визначити, які із цих відношень

(а) рефлексивні; (б) антирефлексивні; (в) симетричні;

(г) антисиметричні; (д) транзитивні; (е) толерантні.

Відношення R на множині M називається ***толерантним***, якщо воно рефлексивне й симетричне.

1. Нехай $A = \{1, 2, 3, 4, 5\}$. Чи є відношення еквівалентним?

$R = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (1, 3), (1, 5), (3,1), (3, 5), (5, 1), (5, 3)\}$.

4. Дана множина $X = \{1, 2, 3, 6\}$ і відношення $R = \{(x,y)|x,y \in X, x -$ дільник $y\}$. Показати, що відношення R є відношенням порядку. Побудувати діаграму Хассе частково впорядкованої множини (X, R). Чи існує в множині X найбільший і найменший елементи? Чи існують непорівнювальні елементи?

1.5. Функціональні відношення

Відношення R ($R \subset A \times B$) називають **функціональним**, якщо для кожного $x \in A$ переріз R по x містить не більше одного елемента $y \in B$ (або один, або жодного!).

У цьому випадку говорять, що відношення R діє з множини A у множину B і часто використовують позначення $R{:}A \to B$.

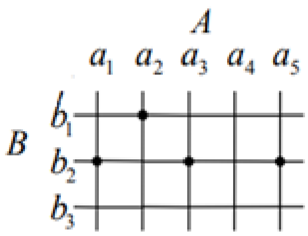
Також використовується позначення функціональної залежності малими латинським буквами $f{:}A \to B$ або $y = f(x)$, а відношення f називають **функцією**.

Функція f може бути задана не на всій множині A, а тільки на деякій її частині $D \subset A$. У цьому випадку множину $D(f)$ називають **областю визначення функції** f, або $domf$ ($domain$ — «область»): $Domf = \{a \in A | b \in B, b = f(a)\}$.

Підмножину $Im \subset B$ називають **областю значень функції** f (Image– «зображення чи образ»: $Im\,f = \{b \in B | a \in A, b = f(a)\}$.

Елемент $b = f(a)$, де $a \in D$, називають **образом** елемента a, а сам елемент a − **прообразом** елемента b. Якщо $D = A$, то функція f називається **<u>всюди визначеною</u>** на A. У цьому разі $Domf = A$.

Приклад 1.57. Відношення f_1, яке задано таблицею



,

є функціональним, але не всюди визначеним. Образом елемента a_3 є елемент b_2, а прообразом елемента b_2 є елементи a_3, a_1, a_5.

Приклад 1.58.

а) Нехай $A = \{1, 2, 3, 4\}$, $B = \{1, 4, 9, 16, 25\}$, тоді $R_1 = \{(1,1), (2,4), (3,4), (4,16)\}$ – функціональне, але відношення $R_2 = \{(1,1), (1,4), (3,9)\}$ не є функціональним за означенням.

б) Англо-український словник встановлює відповідність між множинами англійських і українських слів. Це відношення не є функціональним, оскільки одному англійському слову зазвичай ставиться у відповідність кілька українських слів. До того ж воно не є всюди визначеним, оскільки завжди можна знайти англійське слово, якого немає в цьому словнику.

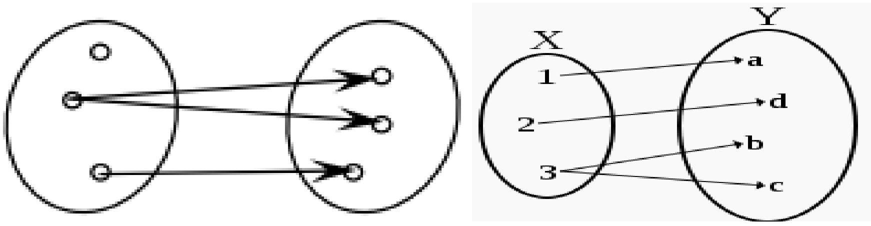


Рис. 25. Відношення, але не функція

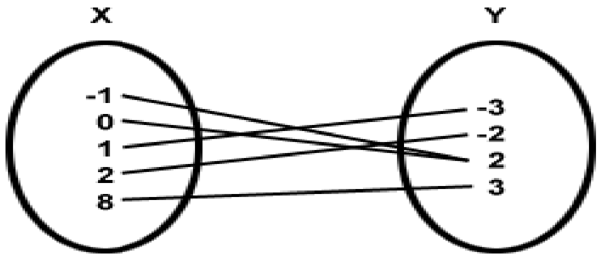
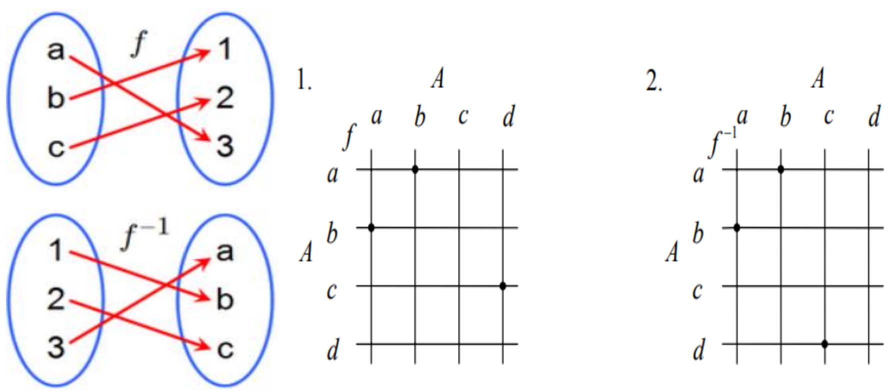


Рис. 26. Відношення і функція

Обернене функціональне відношення

Якщо обернене відношення $R^{-1} \subset B \times A$ також є функціональним відношенням, то це відношення визначає деяку функцію, яку будемо називати оберненою до f функцією і позначати $f^{-1}: B \to A$ або $f(a) = b \Leftrightarrow f^{-1}(b) = a$.



$$f = \begin{bmatrix} a & b & d \\ b & a & c \end{bmatrix}, \quad f^{-1} = \begin{bmatrix} a & b & c \\ b & a & d \end{bmatrix}.$$

Function $f(x)$	Inverse $f^{-1}(y)$	Notes
$x + a$	$y - a$	
$a - x$	$a - y$	
mx	y / m	$m \neq 0$
$1 / x$	$1 / y$	$x, y \neq 0$
x^2	$\sqrt{y}$	$x, y \geq 0$ only
x^3	$\sqrt[3]{y}$	no restriction on x and y
x^p	$y^{1/p}$ (i.e. $\sqrt[p]{y}$)	$x, y \geq 0$ in general, $p \neq 0$
e^x	$\ln y$	$y > 0$
a^x	$\log_a y$	$y > 0$ and $a > 0$

Рис. 27. Приклади функціональних і обернених до них відношень

Якщо функція задана формулою $y = f(x)$, то для знаходження оберненої функції потрібно розв’язати рівняння $f(x) = y$ відносно x, а потім поміняти місцями x і y.

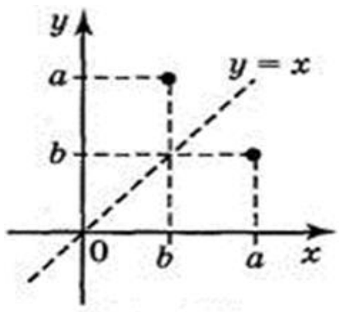
$$f(x) = (2x + 8)^3,$$
$$y = (2x + 8)^3,$$
$$\sqrt[3]{y} = 2x + 8,$$
$$\sqrt[3]{y} - 8 = 2x,$$
$$\frac{\sqrt[3]{y}-8}{2} = x,$$
$$f^{-1}(y) = \frac{\sqrt[3]{y}-8}{2}.$$

Графіки даної функції і оберненої до даної симетричні відносно прямої $y = x$.

Якщо точка $(a; b)$ належить графіку даної функції, то точка $(b; a)$ належить графіку оберненої функції, а ці дві точки симетричні відносно прямої $y = x$.

Якщо функція $y = f(x)$ зростає (спадає) на деякому проміжку, то вона оборотна. Обернена функція до даної визначена в області значень функції $y = f(x)$ і також є зростаючою (спадною).

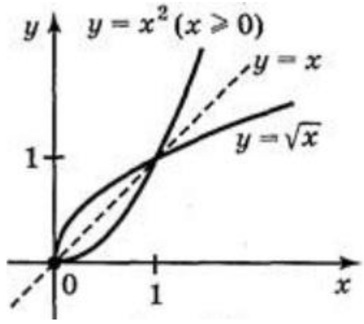
Функція, яка набуває кожного свого значення в єдиній точці області визначення, називається **оборотною,** наприклад функція $y = 2x+1$ – оборотна, а функція $y = x^2$ (визначена на всій числовій осі) не є оборотною.



Якщо рівняння *f(x)* = *y* відносно *x* має більше ніж один корінь, то функція *y* = *f(x)* не має оберненої функції.

Тригонометричні функції *y* = *cosx*, *y* = *sinx* не є монотонними у всій області їх визначення. Тому для утворення обернених функцій виділяють інтервали монотонності.

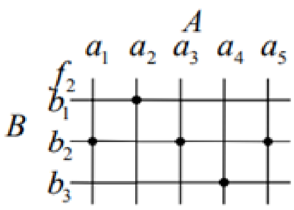
Функція $y = x^2$ не є оборотною в області визначення. Проте функція $y = x^2$, де $x \in [0; +\infty)$ зростає на цьому проміжку, тому має обернену. Оберненою функцію є функція $y = \sqrt{x}$.



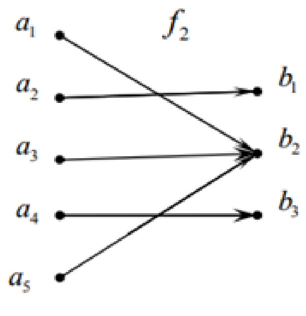
Відображення

Якщо відношення водночас є функціональним та всюди визначеним на множині *A*, то воно називається **відображенням** множини *A* у множину *B*: *f*: *A*→*B*. У стрілочному зображенні відображення *f* з кожної точки повинна виходити лише одна стрілка.

Приклад 1.59. Довизначемо відношення f_1, поклавши $\boldsymbol{a_4 f_1 b_3}$, тоді здобудемо функціональне відношення f_2:



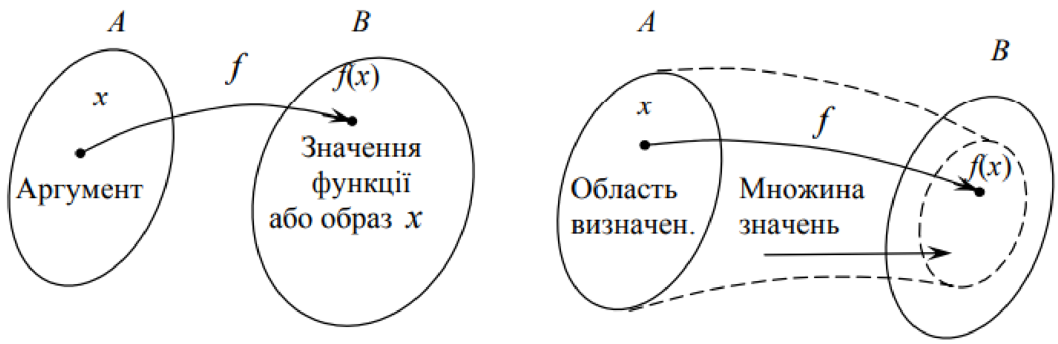
; або



,

яке вже є відображенням.

Нехай fє відображенням множини A на множину B. Переріз $f(x)$ множини f по $x \in A$ є образом елемента x для функції fі позначається як $y = f(x)$.Елемент x називають **аргументом**, $f(x)$ – **значенням функції**. Переріз $f^{-1}(y)$ множини B по $y \in B$ є прообразом елемента y для функції f.



Значення або образ функції f

Відображення f множини A у множину B

Типи відображень

Відображення f називається сюр'єктивним, або просто сюр'єкцією, якщо область значень f збігається з усією множиною B або $f(A) = B$, тобто якщо кожний елемент з множини B є образом хоча б одного елемента з множини A. Тобто f: $A \to B$, $\forall b \in B\ \exists a \in A : b = f(a)$.

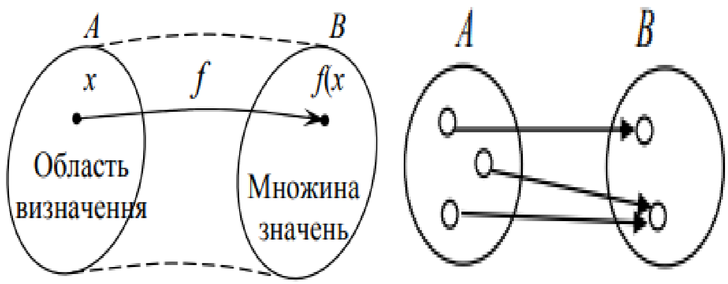
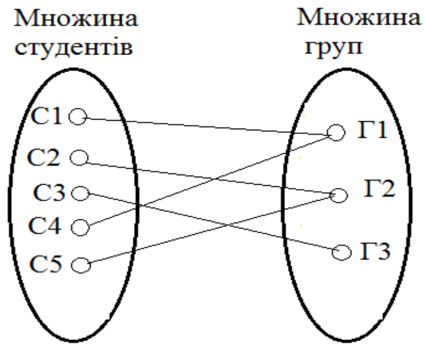
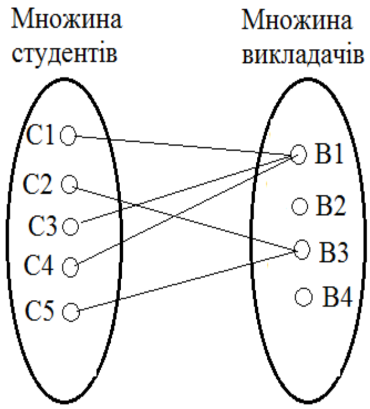


Рис. 28. Тип відображення – сюр'єкція

Приклад 1.60. Відповідність між множиною всіх студентів університету і множиною груп. Це відношення сюр'єктивне, оскільки кожній групі відповідає хоча б один студент.



Приклад 1.61. Відповідність між множиною студентів 2-го курсу університету і множиною викладачів. Це відношення не сюр'єктивне, оскільки на другому курсі викладають не всі викладачі.



Відображення f називається ін'єктивним, або просто **ін'єкцією**, якщо відношення f^{-1} є функціональне, тобто різні елементи множини A переводяться в різні елементи множини B. У цьому разі кожний елемент з області значень f має єдиний прообраз, тобто $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$.

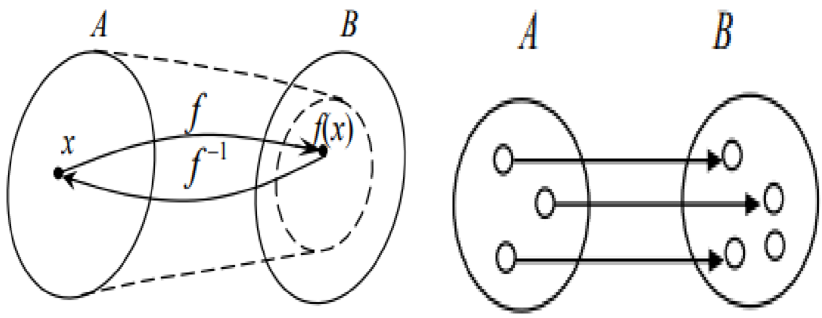
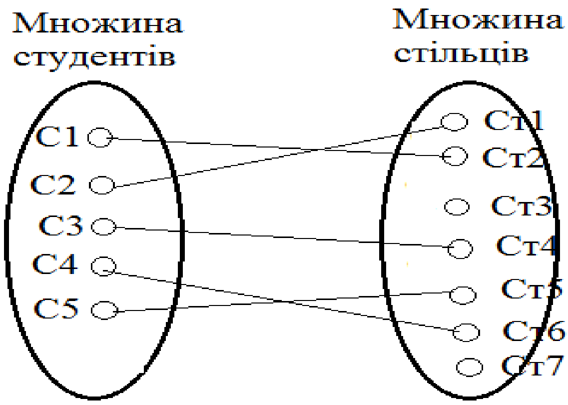
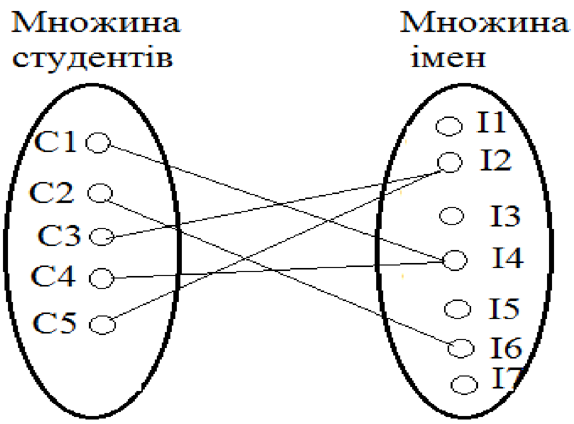


Рис. 29. Тип відображення – ін'єкція

Приклад 1.62. Відповідність між множиною студентів і стільців в аудиторії. Це відношення ін'єктивне, оскільки різні студенти сидять на різних стільцях.



Приклад 1.63. Відображення множини студентів університету на множину імен. Це відношення не ін'єктивне, оскільки є студенти з однаковими іменами.



Відображення називається взаємнооднозначним, або бієктивним, або просто **бієкцією**, якщо воно є сюр'єктивне й ін'єктивне, інакше кажучи, якщо обернене відношення є відображенням.

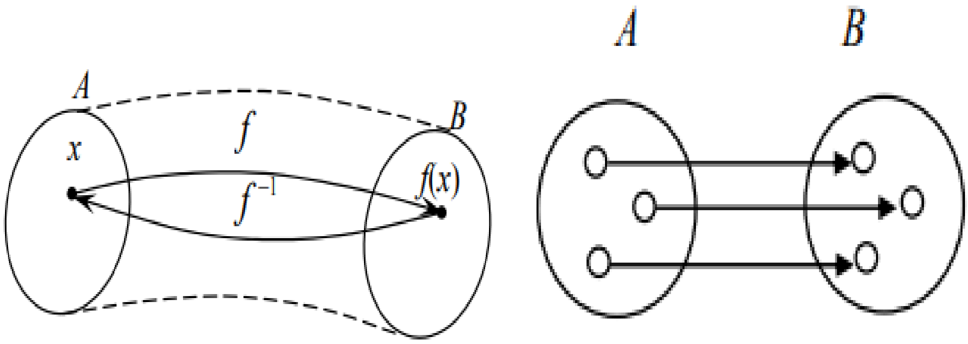
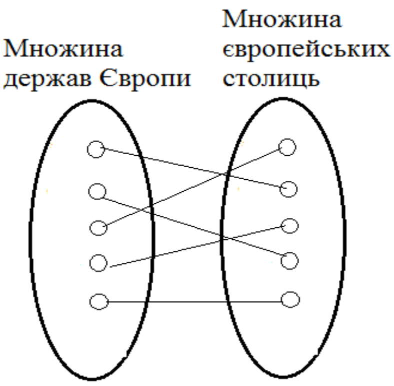


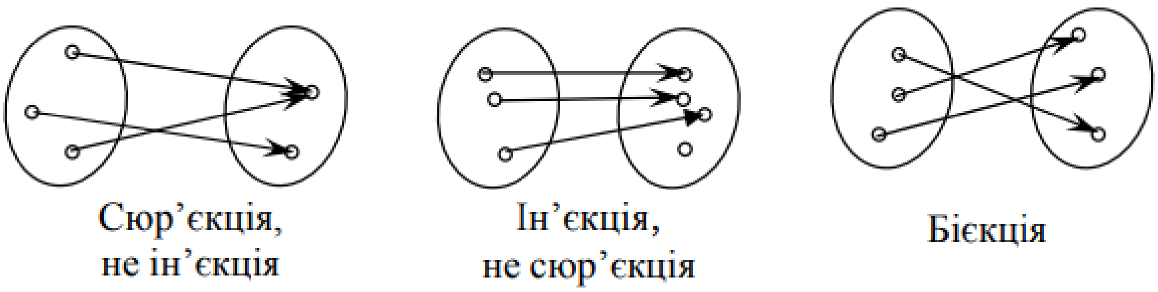
Рис. 30. Тип відображення – бієкція

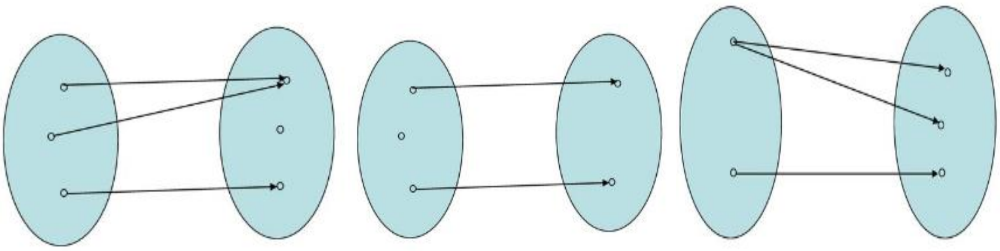
Приклад 1.64.



- Список студентів у журналі (номер→прізвище)
- Нумерація сторінок у книзі (сторінка →номер)

Приклади відношень





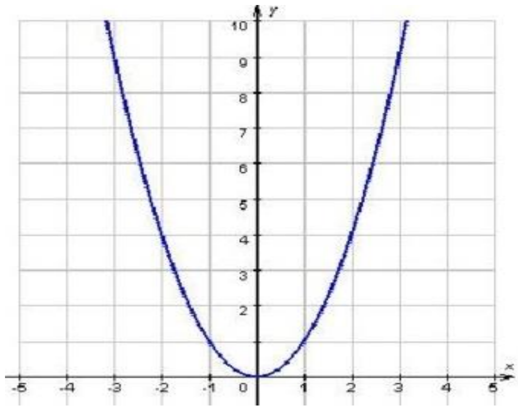
Не ін'єкція Не відображення Не сюр'єкція

Приклад 1.65. Визначити множини на яких відображення є бієкцією. $f(x) = x^2$.

а) R→R не ін'єкція, не сюр'єкція (не використовуються від'ємні значення *y*, і для двох різних *x* одне значення *y*);

б) R→[0;+∞) не ін'єкція, сюр'єкція (використовуються всі значення *x* та *y*, і для двох різних *x* одне значення *y*);

в) [0;+∞) → [0;+∞) ін'єкція, сюр'єкція – бієкція (для кожного x одне значення y).



Приклад 1.66. Нехай R − множина дійсних чисел, + R − множина дійсних додатних чисел, а функція f: $A \to B$.
1) Якщо $A = B = R$, то функція f: $x \to x^2$ задає відображення A у B (не сюр'єктивне, тому що від'ємні числа не є образами).
2) Якщо $A = B = R$, то функція f: $x \to 4x - 3$ задає відображення A на B (сюр'єктивне).
3) Якщо $A = R$, $B =$ R+ , то функція f: $x \to 3^x$ − ін'єктивне відображення, тому що воно є взаємнооднозначне: f^{-1}: $x \to log_3 x$.

Зауваження. 1. Якщо функція f: $A \to B$ є бієкцією, то функція f^{-1}: $B \to A$ також буде бієкцією $(f^{-1})^{-1} = f$.

2. Бієкція скінченної множини А на себе називається **підстановкою**. Якщо множина має n елементів, то можна розглядати множину n! всіх підстановок, пов'язаних із цією множиною А.

Визначення. Елемент x називається нерухомою точкою відображення f, якщо $f(x) = x$.

Зауваження. Оскільки функція є окремим випадком відношення, це означає, що для функцій є також визначена композиція, яка в цьому разі називається суперпозицією функцій. Нехай f − функція, визначена на множині A зі значеннями в множині B, g − функція, визначена на множині B зі значеннями в множині C, тоді композиція $g \circ f$ є функція, яка діє з множини A в множину C.

Таким чином, суперпозиція функцій знову є функцією. З означення суперпозиції маємо $(g \circ f)(x) = g(f(x))$.

Приклад 1.67. Нехай задано функції

$$f=\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ l & m & n & l \end{bmatrix} \quad \text{та} \quad g=\begin{bmatrix} l & m & n \\ c_1 & c_2 & c_3 \end{bmatrix}.$$

$$gf=\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ c_1 & c_2 & c_3 & c_1 \end{bmatrix}.$$

Зауваження. Для відображень g і f справедлива формула $(gf)^{-1} = f^{-1}g^{-1}$. Нехай задано множину $A = \{a_1, a_2, \dots, a_n\}$.

Визначення. ***Тотожним*** відображенням називається відображення, яке кожному елементові $a \in A$ ставить у відповідність цей же елемент (позначається символом 1_A). Таким чином:

$$1_A=\begin{bmatrix} a_1 \; a_2 \; \dots \; a_n \\ a_1 \; a_2 \; \dots \; a_n \end{bmatrix}.$$

Визначення. Якщо f та f^{-1} – відображення, визначені на множині A зі значеннями в цій же самій множині A, то відображення f називається ***відображенням на себе*** (бієкцією на себе) і мають місце рівності: $1_A \circ f = f \circ 1_A = f; f \circ f^{-1} = f^{-1} \circ f = 1_A$.

Приклад 1.68. Нехай відображення f задано таблицею

f	1	2	3	4	5
1				•	
2					•
3		•			
4	•				
5			•		

.

Тоді відображення $f^{-1} \subset A \times A$ визначається таблицею

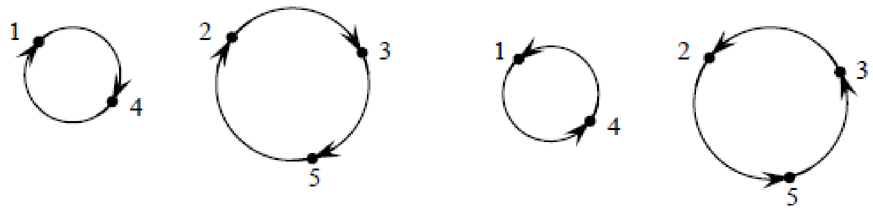
f^{-1}	1	2	3	4	5
1				•	
2			•		
3					•
4	•				
5		•			

.

Функції f і f^{-1} запишемо у вигляді:

$$f=\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 5 & 1 & 2 \end{bmatrix}, \quad f^{-1}=\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 2 & 1 & 3 \end{bmatrix}.$$

Зображення відображень f й f^{-1} стрілками складається із циклів:



Перевіримо виконання умови:

$$1_A \cdot f = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 5 & 1 & 2 \end{bmatrix} = f; \quad f \cdot 1_A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 5 & 1 & 2 \end{bmatrix} = f.$$

Звідси випливає, що відображення 1_A є тотожним. Знайдемо композицію відображень f та f^{-1}:

$$f \cdot f^{-1} = f^{-1} \cdot f = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} = 1_A.$$

Визначення. Функція f : $A_1 \times A_2 \times \ldots \times A_n \to B$ $(f: A^n \to B)$ називається ***функцією n аргументів.*** Така функція відображає кортеж $(a_1, a_2, \ldots, a_n) \in A_1 \times A_2 \times \ldots \times A_n$ у елемент $b \in B$.

<u>Обов'язкові завдання</u>

1. Нехай задано такі відношення між множинами $A = \{a, b, c, d, e\}$ і $B = \{1, 2, 3, 4, 5\}$:

$C_1 = \{(a, 2), (a, 5), (b, 1), (b, 5), (c, 2), (d, 3), (d, 5)\}$;

$C_2 = \{(a, 3), (b, 2), (c, 3), (e, 3)\}$;

$C_3 = \{(a, 2), (b, 3), (c, 4), (d, 1), (e, 5)\}$;

$C_4 = \{(a, 2), (a, 3), (b, 1), (c, 4), (c, 5), (d, 1), (e, 2), (e, 4)\}$;

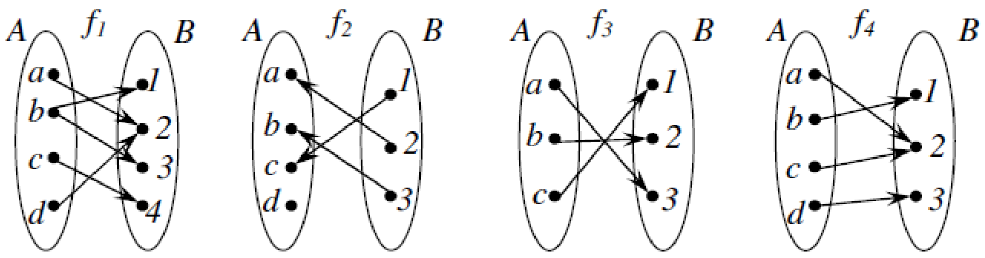
$C_5 = \{(a, 4), (b, 3), (c, 5), (d, 2), (e, 1)\}$.

Визначити, які з цих відношень

(а) всюди визначені; б) функціональні; (в) ін'єктивні; (г) сюр'активні; (д) бієктивні (взаємно однозначні).

2. Для бінарного відношення ρ між елементами множин $A = \{1, 2, 3, 4, 5\}$, $B = \{\{1\}, \{1, 2\}, \{2, 5\}, \{3\}\}$, $a\rho X \Leftrightarrow a \notin X$ знайдіть область визначення $D\rho$ і область значень $R\rho$.

3. Які з відношень, графіки яких зображені на рис., є функціями? Знайдіть для кожної функції її область визначення і область значення. Які з відношень, є ін'єктивними, сюр'єктивними, бієктивними функціями?



1.6. Алгебраїчні структури

Алгебраїчні операції

Алгебра вивчає множини, для елементів яких введено відношення, які називаються алгебраїчними операціями. Під n-арною алгебраїчною операцією (внутрішнім законом композиції) на множині M розуміють відображення множини $\underbrace{M \times M \times \ldots \times M}_{n}$ в M.

Поняття n-арної алгебраїчної операції є рівносильним до поняття відношення R: $(a_1, a_2, \ldots, a_n, b) \in R$, якщо $(a_1, a_2, \ldots, a_n) \to b$.

Під бінарною операцією на множині M розумітимемо закон, за яким кожним двом елементам a та b множини M ставиться у відповідність певний елемент d цієї множини: $(a, b) \to d$.

Для запису композиції елементів a та b їх позначають спеціальним знаком. Закон композиції, який позначається знаком «+», переважно називають додаванням і стверджують, що для нього прийнято адитивне позначення. Закон композиції, який позначається знаком «•», переважно називають множенням і стверджують, що для нього прийнято мультиплікативне позначення.

Під час вивчення загальних властивостей бінарних операцій використовують символ «*».

Прикладом бінарної операції є операція векторного множення векторів.

Операція скалярного добутку векторів не є бінарною алгебраїчною операцією, оскільки скалярний добуток є число, а не вектор.

Для задання бінарної алгебраїчної операції складають таблицю операції (**таблицю Келлі**), рядки і стовпці якої позначають елементами множини M, а на перетинанні рядка і стовпця ставиться відповідний результат операції:

*	a_1	a_2	…	a_n
a_1	$a_1 * a_1$	$a_1 * a_2$	…	$a_1 * a_n$
…	…	…	…	…
a_n	$a_n * a_1$	…	…	$a_n * a_n$

Кількість бінарних операцій на множині з n елементів можна визначити в такий спосіб: маючи n^2 клітинок таблиці, до кожної з них слід записати будь-який з n елементів множини M. Звідси випливає, що кількість бінарних операцій на множині з n елементів дорівнює n^{n^2}. Якщо множина складається з елементів a і b, то існує $2^{2^2} = 16$ операцій.

*	a	b
a	b	a
b	a	b

Використання таблиць має велике значення, оскільки деякі операції в комп'ютерній математиці не придатні для словесного завдання.

Множина, у якій введено бінарну алгебраїчну операцію, називається ***групоїдом.***

Властивості алгебраїчних операцій

Бінарна операція називається ***асоціативною***, якщо для будь-яких елементів a, b, c множини M

$$(a * b) * c = a * (b * c).$$

Прикладом асоціативної операції є операція множення матриць.

Прикладом неасоціативної операції є операція векторного добутку векторів, тому що, наприклад, $(i \times j) \times j \neq i \times (j \times j)$.

Множина, для якої введена асоціативна операція, називається ***півгрупою.***

Бінарна операція називається ***комутативною***, якщо для будь-яких елементів множини M. $a * b = b * a$.

Прикладом комутативної операції є операція додавання матриць, а прикладом некомутативної операції є операція множення матриць.

Векторний добуток векторів є антикомутативною операцією: $a \times b = -b \times a$.

Бінарна операція * називається ***дистрибутивною зліва*** відносно операції $\circ$, якщо для будь-яких елементів a, b, c.

$a * (b \circ c) = (a * b) \circ (a * c)$.

Операція називається **дистрибутивною справа**, якщо $(a \circ b) * c = (a * c) \circ (b * c)$.

Операція множення чисел є дистрибутивною відносно додавання чисел: $(a + b)\, c = a \cdot c + b \cdot c$.

Але операція додавання не є дистрибутивною відносно множення: $a + (b \cdot c) \neq (a + b) \cdot (a + c)$.

Операції перерізу й об'єднання множин є дистрибутивною відносно одна одної і зліва, і справа.

Елемент e називається ***нейтральним елементом*** відносно операції $\circ$, якщо для кожного елемента a

$a \circ e = a$ і $e \circ a = a$.

Нейтральний елемент є єдиний, оскільки, якщо e' – інший нейтральний елемент, то $e = e \circ e' = e'$.

Нейтральний елемент відносно операції додавання називається ***нульовим*** елементом і позначається символом 0.

Нейтральний елемент відносно операції множення називається ***одиничним*** елементом і позначається символом 1.

Елемент a' називається симетричним елементу a у групоїді з нейтральним елементом e, якщо $a \circ a' = a' \circ a = e$.

Елемент $a,'$ симетричний до a відносно операції додавання, називається протилежним до a й позначається символом $-a$.

Елемент $a,'$ симетричний до a відносно операції множення, називається оберненим до елемента a й позначається символом a^{-1}.

Алгебраїчні структури

Алгебраїчною структурою називається множина M разом із заданими Q операціями, визначеними й замкненими на цій множині. M – носій алгебраїчної структури.

Визначення. Групою називається множина з визначеною на ній бінарною асоціативною операцією, для якої існує обернена операція.

Група називається **скінченною**, якщо вона містить скінчену множину елементів.

Число елементів скінченної групи називається **порядком** групи.

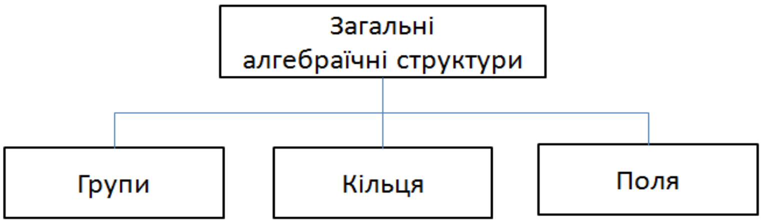


Рис. 31. Алгебраїчні структури

Група

У теорії груп зазвичай використовується мультиплікативна термінологія (тобто групова операція називається множенням, нейтральний елемент – одиничним, симетричний елемент –оберненим).

З визначення випливає, що в кожній групі існує одиничний (нейтральний) елемент і для кожного елемента групи існує обернений елемент.

З іншого боку, можна показати, що якщо асоціативна операція гарантує існування нейтрального й оберненого елементів, то множина з такою операцією є групою. У зв'язку із цим часто користуються іншим визначенням групи, рівносильним до першого.

Визначення. Непорожня множина G, на якій визначена бінарна операція, називається **групою**, якщо виконуються такі умови:

1) операція асоціативна;
2) в G існує нейтральний елемент;
3) для кожного елемента a існує обернений елемент a^{-1}.

Група називається **комутативною**, або **абелевою**, якщо вона має такі властивості:

1) **замкнутість** – якщо a і b — елементи G, то $c = a * b$ — також елемент G;
2) **асоціативність** – якщо a і b — елементи G, то $(a* b) * c = a* (b *c)$;
3) **комутативність** – для всіх a і b в G маємо $a * b = b * a$;
4) **існування нейтрального елемента** – для всіх елементів в G існує елемент e, такий, що $e * a = a * e = a$.
5) **існування інверсії** – для кожного а в G існує елемент a', такий, що $a * a' = a' *a = e$.

Приклад 1.69. Множина цілих чисел є абелевою групою відносно операції додавання.

1. Результатом додавання двох цілих чисел є ціле число.

2. Асоціативність: 4 + (5 + 6) = (4 + 5) + 6.
3. Комутативність: 4 + 5 = 5 + 4.
4. Нейтральним елементом групи є число 0.
5. Симетричним елементом для числа *n* є число *–n (4, –4)*.

Ця група називається адитивною групою цілих чисел і позначається G = <Z_n, +>.

Розглянемо групу $(Z_5;\ \oplus_5)$. Побудуємо для операції таблицю Келлі:

$\oplus_5$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Нейтральним елементом відносно операції $\oplus_5$ e = 0, таблиця симетрична відносно діагоналі – операція комутативна, існують обернені елементи x = 1; $x' = 4$; y = 2; $y' = 3$; x = 3; $x' = 2$.

Кільце

Визначення. Кільцем (*R*,+,×) називається множина *R*, на якій визначені дві бінарні алгебраїчні операції (додавання та множення), при цьому відносно однієї із цих операцій множина є абелевою групою, а друга операція є дистрибутивна відносно першої.

Кільце називається **комутативним**, якщо друга операція є комутативна, і **асоціативним**, якщо вона є асоціативна.

Поле

Визначення. Ненульові елементи кільця можуть утворювати групу відносно операції множення.

Таке кільце називається **кільцем із діленням**, або **тілом**. Комутативне тіло називається **полем**.

Поле являє собою єдність двох абелевих груп – адитивної та мультиплікативної. У криптографії використовують тільки скінчені поля.

Скінчене поле – це поле зі скінченою кількістю елементів. Галуа показав що скінчені поля повинні мати кількість елементів p^n, де *p* – просте, а *n* – додатне ціле число. Скінчені поля називають **полями Галуа** й позначають *GF* (p^n).

Приклад 1.70. Розглянемо алгебраїчну структуру $(Z_n;\ \otimes_n;\ \oplus_n)$, $n = 8$. Побудуємо таблиці Келлі:

$\oplus_8$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

$\otimes_8$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Існують випадки, коли добуток ненульових членів дорівнює 0, а саме (2; 4), (4; 6), (4; 4)…, отже, $(Z_n;\ \otimes_n;\ \oplus_n)$ не є полем, а є комутативним кільцем із нейтральним елементом 0.

Алгебраїчна структура	**Операції**	**Набори цілих чисел**
Група	(+ -) або (× /)	Z_n або $Z^*{}_n$
Кільце	(+ -) та (×)	Z
Поле	(+ -) та (× /)	Z_p

<u>Обов'язкові завдання</u>

1. З'ясувати замкнутість L відносно операцій алгебри $A = \langle N, \{+,\times\}\rangle$, якщо L – це множина:

а) парних чисел; д) чисел виду n^2, де $n \in N$;

б) непарних чисел; е) чисел, більших ніж 1000;

в) чисел, кратних k; є) чисел, менших ніж 1000;

г) чисел виду 2^k, $k \in N$;

2. Які з множин L з попередньої задачі є носіями систем $B = \langle L, \{+,\times\}\rangle$, що утворюють підалгебри алгебри $A = \langle N, \{+,\times\}\rangle$?

3. Написати таблиці Келлі додавання та множення алгебри остач $A = \langle N_{(p)}, \{\otimes,\oplus\}\rangle$ для таких значень p:

а) p=3; б) p=4; в) p=7; г) p=2.

4. Чи є операції $\oplus,\otimes$ алгебри остача:

а) асоціативними;

б) комутативними;

в) дистрибутивними одна відносно одної?

5. З'ясувати, чи є операція конкатенації слів:

а) асоціативною; б) комутативною?

Розділ 2. Комбінаторика

2.1. Теоретичні основи комбінаторики

Вступ у комбінаторику

Комбінаторика – це розділ математики, який вивчає питання вибору або розміщення елементів множин відповідно до заданого правила.

У вузькому сенсі комбінаторика – це підрахунок будь-яких комбінацій, які можна скласти з деякої множини дискретних об’єктів. Під об’єктами розуміють будь-які окремі предмети або живі істоти – людей, звірів, комах, гриби, рослини тощо. При цьому комбінаторику зовсім не хвилює, що множина складається з тарілки манної каші, паяльника чи болотної жаби. Принципово важливим є одне – ці об’єкти піддаються лічбі – їх три (дискретність) і істотним є те, що серед них немає однакових.

Комбінаторика походить від слова **combina**, що в перекладі означає «**сполучати**, **з’єднувати».**

Методи розв’язування задач комбінаторики називають **методами комбінаторного аналізу.** Оскільки комбінаторика має справу зі скінченими множинами, на природу об’єктів яких жодних обмежень не накладають, то її часто називають **теорією скінчених множин.**

Комбінаторика виникла в XVI сторіччі, коли в житті верхнів прошарків суспільства важливе місце займали азартні ігри (карти, кості, пасьянси, лотереї). Це стало рушійною силою для розвитку комбінаторики й теорії імовірностей.

Ряд перших комбінаторних задач розв'язали такі відомі математики, як Паскаль, Ферма, Ейлер, Бернуллі, Лейбніц.

Г. В. Лейбніц у 1666 ройі захистив дисертацію «Про комбінаторне мистецтво» і ввів термін «комбінаторика»

Комбінаторика сьогодні

Комбінаторика (або комбінаторний аналіз, або комбінаторна математика) – це галузь математики, яка вивчає способи побудови підмножин деякої скінченної множини, причому таких, які відповідають заданим обмеженням.

Згадані підмножини часто називають комбінаторними конфігураціями або вибірками. Якщо, наприклад, враховувати порядок розміщення об'єктів, то одержимо різні комбінаторні конфігурації, як показано на рис. 32.

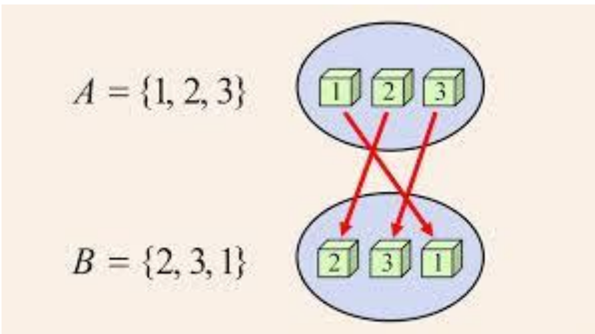


Рис. 32. Формування комбінаторної конфігурації

Комбінаторні методи лежать в основі розв'язку багатьох задач теорії імовірностей та її додатків.

Комбінаторика вивчає такі види задач:

1. Підрахунок числа комбінаторних конфігурацій.

Приклад 2.1. Зі зміною порядку кольорових кульок у стовпцях одержимо скінченну кількість різних стовпців. Кожний стовпець відповідатиме одній комбінаторній конфігурації, як показано на рис. 33.

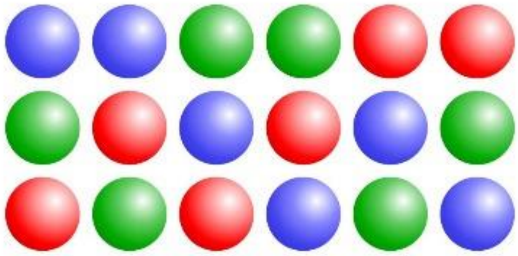


Рис. 33. Задача визначення кількості відмінних стовпців кульок

2. Знаходження умов існування комбінаторної конфігурації.

Приклад 2.2. Задача знаходження умов існування комбінаторної комбінації виникає під час визначення можливості існування певного розфарбування кубика Рубіка (рис. 34).

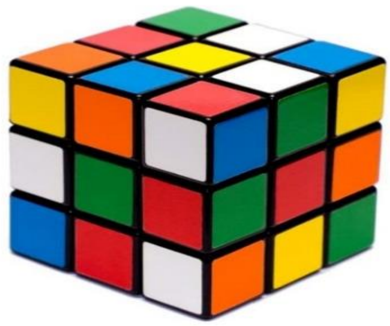


Рис. 34. Задача існування заданого розфарбування

3. Розробка алгоритмів побудови комбінаторних конфігурацій.

Приклад 2.3. Алгоритм побудови транспортних маршрутів, які відповідають заданим параметрам трафіку, дає змогу гнучко розподіляти транспортні потоки.

Рис. 35. Задача перерозподілу транспортних потоків

4. Розв’язування оптимізаційних задач (екстремальних комбінаторних задач).

Приклад 2.4. Перебір і порівняння варіантів вирішення задачі є одним із способів розв’язування оптимізаційної задачі. Така задача може виникнути під час оптимізації WEB-сайту.

Рис. 36. Задача оптимізації технічного рішення

Підрахунок кількості комбінаторних конфігурацій часто зустрічається в програмних засобах. Такі задачі є предметом вивчення рахункової комбінаторики.

Основні поняття комбінаторики

У комбінаториці прийнято говорити про множину, вказуючи кількість її елементів. Наприклад, якщо є множина , яка складається з n елементів, то в цьому випадку її називають n -множиною A.

Визначення. Множину *B* називають **підмножиною** множини *A* і позначають, якщо всі елементи множини *B* є також елементами множини *A*.

Визначення. Якщо множина *C* має кілька екземплярів того самого елемента, то таку множину називають **мультимножиною.**

Вибірка

Визначення. Вибіркою називають довільну мультимножину, елементи якої вибирають з елементів множини А, тобто таку множину, яка в загальному випадку може містити кілька екземплярів того самого елемента множини А, як показано на рис. 37.

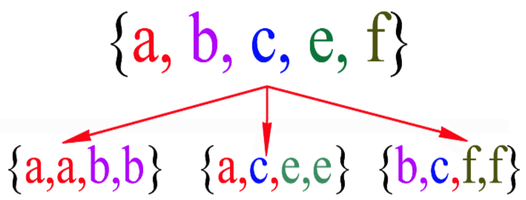


Рис. 37. Приклади 4-вибірок з 5-множини

Обсяг вибірки. Кількість елементів *r* у вибірці (таку вибірку називають також *r* -вибіркою) визначають як її **обсяг**.

Інший зміст поняття «вибірка». Поняття «вибірка» використовують також для позначення **самого процесу відбору** елементів підмножини з початкової множини.

Упорядкована вибірка

Визначення. Вибірку називають **упорядкованою**, якщо порядок слідування елементів в ній заданий. Дві впорядковані вибірки, які різняться лише порядком проходження елементів, вважають різними.

Визначення. Нехай – множина з *n* елементів. Впорядкованою вибіркою обсягом *r* з *n*-множини називають будь-яку впорядковану підмножину з *r* її елементів, як показано на рис. 2.7.

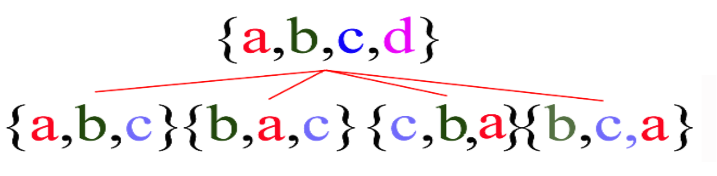


Рис. 38. Упорядковані 3-вибірки з 4-множини

Висновок. Упорядковані вибірки відрізняються порядком слідування одних і тих самих елементів.

Неупорядкована вибірка

Якщо порядок слідування елементів не є істотним, то таку вибірку називають **неупорядкованою**. Приклад такої вибірки показано на рис. 39.

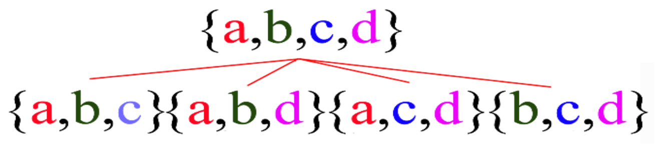


Рис. 39. Приклад неупорядкованої 3-вибірки з 4-множини

Висновок. Неупорядковані вибірки відрізняються елементами, а не порядком їх слідування.

Вибірки з повтореннями та без повторень

Визначення. Вибірки з повтореннями – це вибірки, які допускають повторення елементів. Приклад вибірки з повтореннями показано на рис. 40.

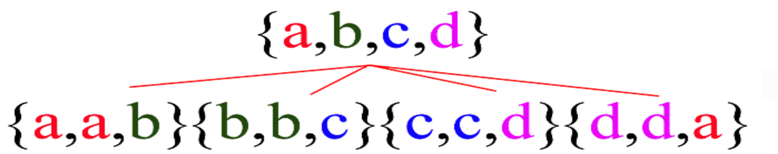


Рис. 40. Приклад 3-вибірок з повтореннями з 4-множини

Визначення. Вибірки без повторень – це вибірки, які не допускають повторення елементів. Приклад вибірки без повторень показаний на рис. 41.

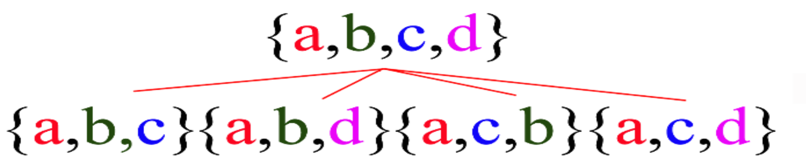


Рис. 41. Приклад 3-вибірок без повторень з 4-множини

Загальноприйняті назви вибірок

1. **Розміщення** (A_n^k – без повторень, $\overline{A_n^k}$ – без повторень) (упорядкована вибірка з повтореннями або без повторень). Набір елементів $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ з множини $X = \{x_1, x_2, \ldots, x_n\}$ називають i-ю вибіркою обсягом k з n елементів $k < n$ або, інакше, (n, k)-розміщенням.

Наприклад: {1, 2, 3, 4}⇒{1, 1, 2},{2, 1, 1},{1, 2, 3}...

{1, 2, 3, 4}⇒{1, 2, 3},{1, 2, 4},{4, 2, 1}...

Поєднання (комбінація) (C_n^m без повторень, $\overline{C_n^m}$ з повтореннями) (неупорядкована вибірка з повтореннями або без повторень).

Визначення. Вибірки, у яких не враховується порядок запису елементів і які відрізняються між собою хоча б одним елементом, називають **поєднанням або комбінаціями**.

Наприклад: {1, 2, 3, 4} ⇒ {1, 2, 3}, {1, 2, 4}, {2, 3, 4} ...

{1, 2, 3, 4} ⇒ {1, 1, 3}, {1, 1, 4}, {4, 2, 1} ...

Перестановка (P_n без повторень, $P(k_1, \ldots, k_m)$ з повтореннями) (упорядкована повна вибірка без повторень і з повтореннями).

Визначення. Вибірки, які складаються з тих самих елементів і які відрізняються тільки порядком їх слідування, називають **перестановками**. Наприклад: {1, 2, 3}⇒{1, 2, 3},{1, 3, 2},{3, 2, 1}...

{1, 2, 3}⇒{1, 1, 3},{3, 1, 1},{3, 3, 1}...

<u>Обов'язкові завдання</u>

1. У студентській групі 23 студенти. Скількома способами можна обрати старосту і його заступника?

2. Скількома способами з колоди в 36 карт можна вибрати 3 карти?

3. У мами 2 яблука і 3 груші. Кожен день протягом 5 днів підряд вона видає по одному фрукту. Скількома способами це можна зробити?

4. Підприємство може надати роботу за однією спеціальністю 4 жінкам, за іншою – 6 чоловікам, за третьою – 3 робітникам незалежно від статі. Скількома способами можна закрити вакансії, якщо є 14 претендентів: 6 жінок і 8 чоловіків?

5. У пасажирському потягу 9 вагонів. Скількома способами можна розсадити в потягу 4 людини за умови, що всі вони повинні їхати в окремих вагонах?

6. У групі 9 людей. Скільки можна скласти різних підгруп за умови, що до підгрупи входять не менше ніж 2 людини?

Основні правила комбінаторики

Комбінаторика – це наука про те, як можна комбінувати різні об'єкти, як можна їх сполучати. Це, з одного боку, наука про те, як порахувати кількість комбінацій визначеного типу, а з іншого – наука про те, як знайти якусь екстремальну комбінацію з якимись оптимальними властивостями. Комбінувати можна що завгодно, наприклад, з групи студентів факультету автоматизації і інформаційних технологій можна вибрати групу студентів, які займатимуться інформаційними технологіями, або групу, що займатиметься кібербезпекою.

Описані задачі – комбінаторні. Можна комбінувати символи деякого алфавіту, і в цьому випадку проявляється зв'язок з такими об'єктами, як ДНК послідовності й т. ін.

Розглянемо два основні правила, які використовують у розв'язуванні комбінаторних задач.

Правило суми

Визначення. Якщо елементи множини A можна вибрати n способами, а елементи множини B можна вибрати m способами, то за умови, що $A \cap B = \varnothing$, загальне число вибірок становить $n + m$.

Ці правила дуже нагадують **алгебру подій**. Знак «плюс» слід розуміти і читати як АБО.

На рис. 42 показано умови застосування правила суми у випадку, коли множини не перетинаються.

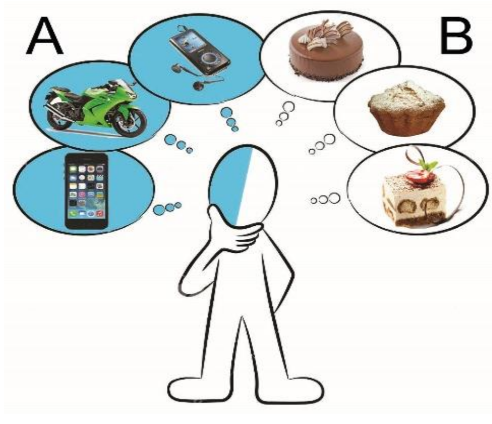


Рис. 42. Демонстрація правила суми

Приклад 2.5. На лекції з дискретної математики присутні 20 студентів. Квитки на концерт Лари Фабіан купили 15 студентів. Скільки всього студентів відвідали лекцію та концерт за умови, що вони відбуваються одночасно?

Розв'язок. Позначимо через X множину студентів, які були присутні на лекції, а через Y – множину студентів, які відвідали концерт. Оскільки $n = |X| = 20$, $m = |Y| = 15$ і $X \cap Y = \emptyset$, то за правилом суми: $m + n = 20 + 15 = 35$.

Приклад 2.6. Студентська група складається з 23 людей, серед яких 10 хлопців і 13 дівчат. Скількома способами можна обрати двох людей однієї статі?

Розв'язок. У цьому випадку підрахунок C_{23}^{2} не підходить, тому що загальна кількість поєднань вміщує і різностатеві пари.

Умова «обрати двох людей однієї статі» передбачає, що потрібно обрати двох хлопців **або** двох дівчат, і вже саме формулювання вказує на правильний шлях вирішення:

$C_{10}^{2} = \frac{10!}{8! \cdot 2!} = \frac{8! \cdot 9 \cdot 10}{8! \cdot 2!} = \frac{9 \cdot 10}{2} = 45$ способами можна обрати 2 хлопців;

$C_{13}^{2} = \frac{13!}{11! \cdot 2!} = \frac{11! \cdot 12 \cdot 13}{11! \cdot 2!} = \frac{12 \cdot 13}{2} = 78$ способами можна обрати 2 дівчат.

Таким чином, двох людей однієї статі (не має різниці – хлопців **або** дівчат) можна обрати: $C_{10}^{2} + C_{13}^{2} = 45 + 78 = 123$ способами.

Відповідь: 123.

Правило добутку комбінацій

Визначення. Кількість способів вибору елементів множини $A \cdot B$ дорівнює $n \cdot m$.

Знак «помножити» слід розуміти і читати як **І**. Розглянемо ту саму студентську групу, яка пішла на танці. Скількома способами можна скласти пару з хлопця і дівчини?

$C_{10}^{1} = 10$ способами можна обрати 1 хлопця;

$C_{13}^{1} = 13$ способами можна обрати 1 дівчину.

Таким чином, одного хлопця й одну дівчину можна обрати: $C_{10}^{1} \cdot C_{13}^{1} = 10 \cdot 13 = 130$ способами.

Коли з кожної множини обирають по 1 об'єкту, то справедливий наступний принцип підрахунку комбінацій: «**кожен** об'єкт з однієї множини може скласти пару з кожним об'єктом іншої множини». Тобто Олег може запросити на танок будь-яку з 13 дівчат, Євген – теж будь-яку з 13, і аналогічний вибір є у всіх інших хлопців. Отже: $10 \cdot 13 = 130$ можливих пар.

Слід зазначити, що в цьому прикладі не має значення «історія» створення пари; однак, якщо взяти до уваги ініціативу, то кількість комбінацій потрібно подвоїти, тому що кожна з 13 дівчат теж може запросити до танцю будь-якого хлопця. Усе залежить від умови тієї чи іншої задачі!

Схожий принцип справедливий і для більш складних комбінацій, наприклад: скількома способами можна обрати двох хлопців **і** двох дівчат для участі в одному епізоді КВК?

Союз **І** однозначно натякає на те, що комбінації потрібно перемножити: $C_{10}^2 \cdot C_{13}^2 = 45 \cdot 78 = 3510$ можливих груп артистів.

Інакше кажучи, **кожна** пара хлопців (45 унікальних пар) може виступати з **будь-якою** парою дівчат (78 унікальних пар). А якщо розглянути розподіл ролей між учасниками, то комбінацій буде ще більше.

Правило множення комбінацій розповсюджується і на більшу кількість множників.

Приклад 2.6. Скільки існує тризначних чисел, які діляться на 5?

Розв'язок. Для наочності позначимо це число трьома зірочками: ******* Комбінації будемо рахувати по розрядах – *зліва направо*:

У *розряд сотень* можна записати будь-яку з $C_9^1 = 9$ цифр (1, 2, 3, 4, 5, 6, 7, 8 або 9). Нуль не підходить, оскільки число перестає бути тризначним. А ось у *розряд десятків* («посередині») можна вибрати будь-яку з 10 цифр: $C_{10}^1 = 10$. За умовою число повинно ділитися на 5, якщо воно закінчується на 5 або на 0. Таким чином, у молодшому розряді нас влаштовують 2 цифри.

Отже, існує: $C_9^1 \cdot C_{10}^1 \cdot 2 = 9 \cdot 10 \cdot 2 = 180$ тризначних чисел, що діляться на 5. При цьому добуток $C_9^1 \cdot C_{10}^1 \cdot 2$ розшифровується так: «9 способами можна вибрати цифру в *розряд сотень* **і** 10 способами вибрати цифру в *розряд десятків* **і** 2 способами в *розряд одиниць*». Або ще простіше: «**кожна з** 9 цифр у *розряді сотень* комбінується **з кожною** з 10 цифр *розряду десятків* **і з кожною** з двох цифр у *розряді одиниць*».

Відповідь: 180.

А тепер… згадаємо задачу з попереднього заняття, у якій Борі, Дімі й Володі можна здати по одній карті $C_{36}^3 \cdot P_3 = 7140 \cdot 6 = 42840$ способами. Множення тут має той самий сенс: $C_{36}^3 = 7140$ способами можна взяти 3 карти з колоди **І в кожній** вибірці представити їх $P_3 = 3! = 6$ способами.

Приклад 2.7. У Васі вдома живуть 4 коти.

а) скількома способами можна розсадити котів по кутках кімнати?
б) скількома способами можна відпустити котів погуляти?
в) скількома способами Вася може взяти на руки двох котів (одного на ліву, іншого – на праву)?

Розв'язок. По-перше, слід звернути увагу на те, що в задачі йдеться про **різні** об'єкти (навіть якщо коти – однояйцеві близнюки). Це дуже важлива умова!

а) Мовчання котів. Цій екзекуції піддаються відразу всі коти + важливе їх розташування, тому тут мають місце перестановки: $P_4 = 4! = 24$ способами можна розсадити котів по кутках кімнати. Ще раз повторюю, що з перестановками значення має лише кількість різних об'єктів і їх взаємне розташування. Залежно від настрою Вася може розсаджувати котів півколом на дивані, у ряд на підвіконні тощо – перестановок у всіх випадках буде 24. Бажаючі для зручності можуть уявити, що коти різнокольорові (наприклад, білий, чорний рудий і смугастий), і перерахувати всі можливі комбінації.

б) Скількома способами можна відпустити котів погуляти?

Передбачається, що коти ходять гуляти тільки через двері, при цьому не має значення кількість тварин – на прогулянку можуть вийти 1, 2, 3 і всі 4 коти.

Рахуємо можливі комбінації:

$C_4^1 = 4$ способами можна відпустити гуляти одного кота (будь-якого з чотирьох);

$C_4^2 = \frac{4!}{2! \cdot 2!} = \frac{24}{2 \cdot 2} = 6$ способами можна відпустити гуляти двох котів (варіанти перерахуйте самостійно);

$C_4^3 = 4$ способами можна відпустити гуляти трьох котів (будь-який один із чотирьох сидить вдома);

$C_4^4 = 1$ способом можна випустити всіх котів.

Мабуть, ви здогадалися, що отримані значення слід скласти: $C_4^1 + C_4^2 + C_4^3 + C_4^4 = 4 + 6 + 4 + 1 = 15$ способами можна відпустити гуляти котів.

Ентузіастам пропоную ускладнену версію задачі – коли будь-який кіт у будь-якій вибірці випадковим чином може вийти на вулицю, як через двері, так і через вікно. Комбінацій значно збільшиться!

в) Скількома способами Вася може взяти на руки двох котів?

Ситуація передбачає не тільки вибір 2 тварин, але й їх розташування на руках: $A_4^2 = 3 \cdot 4 = 12$ способами можна взяти на руки 2 котів. Другий варіант вирішення: $C_4^2 = 6$ способами можна вибрати двох котів **і** $P_2 = 2! = 2$ способами посадити **кожну** пару на руки: $C_4^2 \cdot P_2 = 6 \cdot 2 = 12$.

Відповідь: а) 24, б) 15, в) 12.

І для очищення совісті що-небудь конкретніше на множення комбінацій… Нехай у Васі додатково живуть 5 кішок. Скількома способами можна відпустити погуляти 2 котів і 1 кішку?

$$C_4^2 \cdot C_5^1 = 6 \cdot 5 = 30$$

Тобто з **кожною** парою котів можна відпустити **кожну** кішку.

Правило включень і виключень для двох множин

Визначення. Нехай A і B – скінченні множини. Визначимо, чому дорівнює $|A \cup B|$, якщо відомі потужності $|A|$і $|B|$.

Дотримуючись визначення операції об'єднання*:* $|A \cup B| = |A| + |B| - |A \cap B|$.

Пояснення. Сума $|A| + |B|$ включає всі елементи множини A й множини B. При цьому загальні елементи множин A і B, а їх буде $|A \cap B|$, включаються в суму двічі. Тому для одержання суми об'єднання необхідно відняти один набір спільних елементів. У підсумку одержуємо: $|A \cup B| = |A| + |B| - |A \cap B|$.

Правило включень і виключень для трьох множин

$$|\boldsymbol{A} \cup \boldsymbol{B} \cup \boldsymbol{C}| = |\boldsymbol{A} \cup (\boldsymbol{B} \cup \boldsymbol{C})| = |\boldsymbol{A}| + |\boldsymbol{B} \cup \boldsymbol{C}| - |\boldsymbol{A} \cap (\boldsymbol{B} \cup \boldsymbol{C})| =$$

Застосували правило включень і виключень для трьох множин A і $(B \cup C)$

$$= |A| + |B| + |C| - |B \cap C| - |A \cap (B \cup C)| =$$

Застосували правило включень і виключень для множин $(B \cup C)$

$$= |A| + |B| + |C| - |B \cap C| - |(A \cap B) \cup (A \cap C)| =$$

Застосували дистрибутивний закон

$$= |A| + |B| + |C| - |B \cap C| - (|A \cap B| + |A \cap C| - |A \cap B| \cap (A \cap C)|) =$$

Правило включень і виключень для множин $(A \cap B)$ і $(A \cap C)$

$$= |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

Розкрили дужки.

Правило включень і виключень у загальному вигляді

Розглянемо правило включень і виключень із застосуванням до n множин.

Нехай маємо A_1, A_2, $A_3, \ldots, A_i, \ldots, A_n$ – деякі множини. Тоді формула для визначення потужності множини об'єднання цих множин має такий вигляд: $|A_1 \cup A_2 \cup A_3 \cup \ldots \cup A_n| = \sum_{i=1}^{n} |A_i| - \sum_{1 \le i < j \le n} |A_i \cap A_j| + \sum_{1 \le i < j < k \le n} |A_i \cap A_j \cap A_n| - \ldots$

$$+(-1)^{n-1} \cdot \sum_{1 \le i < j < k .. < l \le n} |A_i \cap A_j \cap \ldots A_k \cap \ldots A_l \cap A_n|$$

Правило підрахунку за цією формулою полягає в послідовному виконанні операцій додавання й віднімання, які чергуються між собою.

Звідси випливає назва: **правило включень і виключень**.

Приклад 2.8. Обчислення за правилом включень і виключень

Нехай дані множини $A = \{1, 2, 3, 4, 9\}$, $B = \{3, 4, 5, 6, 9\}$ і $C = \{5, 6, 7, 8, 9\}$. Обчислити:

1) $|A \cup B|$; 2) $|B \cup C|$; 3) $|A \cup C|$; 4) $|A \cup B \cup C|$.

Розв'язок. Попередньо обчислимо об'єднання трьох множин $A \cup B \cup C = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

1) $A \cap B = \{3, 4, 9\}$; $|A \cap B| = 3$.

Тому $|A \cup B| = |A| + |B| - |A \cap B| = 5 + 5 - 3 = 7$;

2) $B \cap C = \{5, 6, 9\}$; $|B \cap C| = 3$.

Тому $|B \cup C| = |B| + |C| - |B \cap C| = 5 + 5 - 3 = 7$;

3) $A \cap C = \{9\}$; $|A \cap C| = 1$.

Тому $|A \cup C| = |A| + |C| = 5 + 5 - 1 = 9$;

4) $(A \cap B \cap C) = \{9\}$; $|A \cap B \cap C| = 1$;

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C| =$$
$$= 5 + 5 + 5 - 3 - 1 - 3 + 1 = 9.$$

Приклад 2.9. Записати правило включень і виключень для множин *A*, *B*, *C*, *D*.

Розв'язок. Скористаємося загальною формулою:

Перепишемо цю формулу, враховуючи, що n = 4/D. У підсумку одержимо:

$$|A \cup B \cup C \cup D| = |A| + |B| + |C| + |D| - |A \cap B| - |A \cap C| - |A \cap D| - |B \cap C| - |B \cap D| +$$
$$+|A \cap B \cap C| + |A \cap C \cap D| + |B \cap C \cap D| - |A \cap B \cap C \cap D|$$

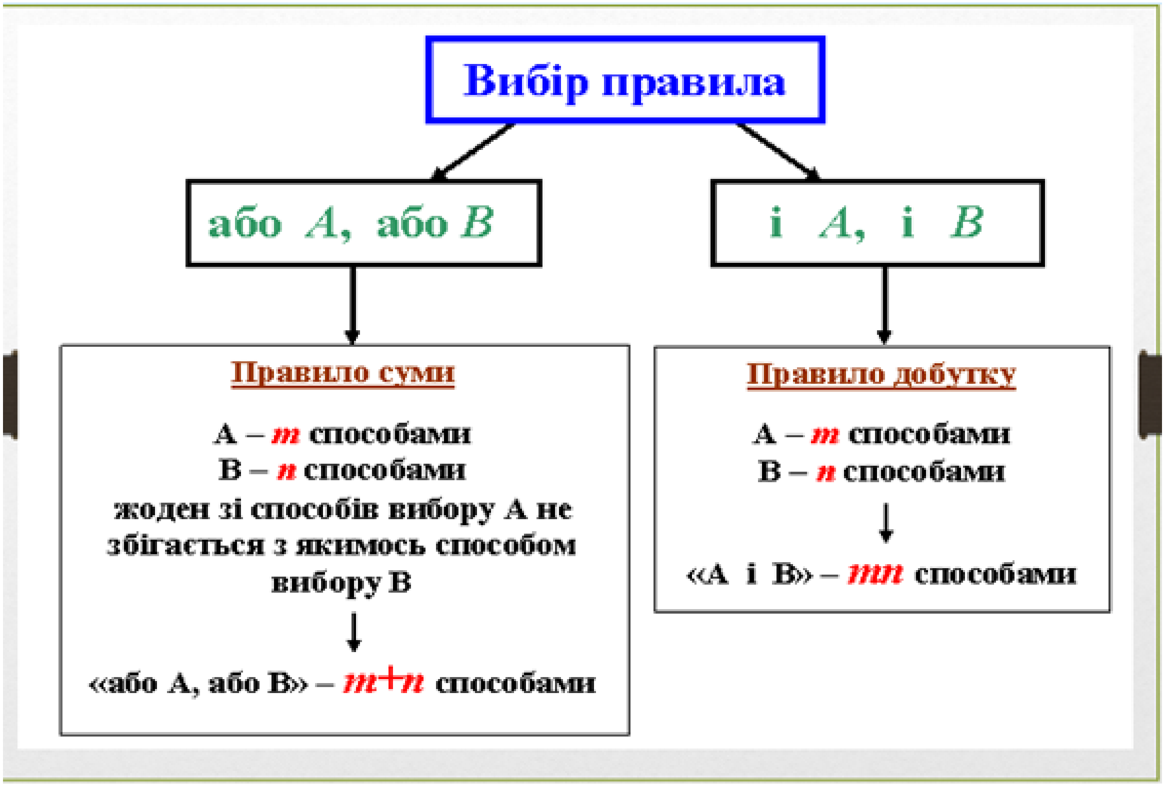


Рис. 43. Схема вибору правила комбінаторики

Принцип Діріхле

Дуже часто буває так, що для розв'язування складних математичних задач використовують різний інструментарій, уже відомі методи, прийоми, принципи, теорії тощо. Завдяки цьому деякі класи задач стають алгоритмічними, а їхні розв'язки – доступними широкому загалу. Жартома цей принцип формулюється так: «Якщо п'ятьох зайців розсадити в чотири клітки, то принаймні в одній із них опиняться два зайці».

В україномовній математичній літературі цей принцип називають **принципом Діріхле** на честь відомого німецького математика Петера Лежена Діріхле (1805–1859), який перший за допомогою такого простого твердження отримав глибокі результати про наближення ірраціональних чисел раціональними (в англомовній літературі цей принцип більше відомий як **pigeonhole principle** – «принцип голубника»). Зауважимо, що задачі цього розділу не претендують на оригінальність, більшість із них уже стала математичним «фольклором», і вже зараз складно встановити їх авторів.

Розглянемо кілька елементарних задач.

Приклад 2.10. У мішку лежать кульки двох різних кольорів – чорного і білого. Яку найменшу кількість кульок потрібно вийняти з мішка, щоб серед них точно дві кульки виявились одного кольору?

Розв'язок. Зрозуміло: узявши три кульки, ми виявимо, що дві з них одного кольору. У цьому випадку роль зайців відіграють кульки, а роль кліток – чорний і білий кольори.

Приклад 2.11. У лісі росте мільйон ялинок. Відомо, що на кожній із них не більше ніж 800 тисяч голок. Доведіть, що в лісі знайдуться дві ялинки з однаковою кількістю голок.

Доведення. Припустімо, що в лісі всі ялинки мають різну кількість голок (на деякій ялинці голок могло не бути зовсім). Тоді в лісі не більше ніж 800 001 ялинка, що суперечить умові. Тут у ролі зайців були ялинки, а в ролі кліток – усі можливі варіанти кількості голок на них.

Приклад 2.12. На 5 поличках книжкової шафи 160 книг, причому на одній із них – 3 книги. Доведіть, що знайдеться поличка, на якій не менше ніж 40 книг.

Доведення. Припустімо, що на кожній із решти 4 поличок не більше ніж 39 книг. Тоді на всіх 5 поличках не більше ніж $3 + 4 \cdot 39 = 159$ книг, що суперечить умові. Отже, на одній із поличок не менше ніж 40 книг.

Проте найчастіше принцип Діріхле використовують в узагальненому формулюванні.

Узагальнений принцип Діріхле. Якщо $nk + 1$ предмет розкладено в n ящиків, то принаймні в одному з ящиків перебуває не менше ніж $k + 1$ предмет.

Доведення. Нехай x_i ($i = 1, \ldots, n$) — кількість предметів, що перебувають в i-му ящику. За умовою $x_1 + x_2 + \ldots + x_n = nk + 1$. Припустімо, що для кожного i виконується $x_i < k$. Тоді $x_1 + x_2 + \ldots + x_n < nk$, що суперечить умові. Отже, наше припущення було хибним, тобто в одному з ящиків є принаймні $k + 1$ предмет. Цей спосіб доведення надалі спрощено також називатимемо принципом Діріхле.

Приклад 2.13. У місті понад 8 мільйонів жителів. Науковці вважають, що в кожної людини менш ніж 200 000 волосин на голові. Доведіть, що є принаймні 41 житель з однаковою кількістю волосин на голові.

Доведення. Оскільки 40 – 200 000 = 8 000 000 (кількість волосин у людини коливається від 0 до 199 999, усього 200 000 варіантів), то згідно з принципом Діріхле знайдеться принаймні 41 житель, котрий має однакову кількість волосин на голові. Тут роль предметів відіграють жителі, а роль ящиків – усі можливі варіанти кількості волосин на голові.

Приклад 2.14. Дано два многочлени від однієї змінної, кожен із яких – сума 9 членів парного степеню, не більшого за 36. Доведіть, що в добутку обов’язково знайдуться три подібні члени.

Доведення. Якщо ми перемножимо ці два многочлени, то отримаємо новий многочлен степеню, не більшого за 72, кожний із 81 одночлена якого має парний степінь. Оскільки парних чисел від 0 до 72 є 37, то, згідно з принципом Діріхле, знайдуться принаймні три подібні члени.

Приклад 2.15. Доведіть, що серед 82 кубиків, кожен із яких помальовано певним кольором, існує 10 кубиків різного кольору або 10 кубиків одного кольору.

Доведення. Якщо для розмалювання 82 кубиків використано не менше ніж 10 кольорів, то зрозуміло, що знайдеться 10 кубиків різного кольору. Якщо ж для розмалювання 82 кубиків використано не більше ніж 9 різних кольорів, то, згідно з принципом Діріхле, знайдеться принаймні 10 кубиків одного кольору.

Тут у ролі предметів виступають кубики, а в ролі ящиків – кольори.

Приклад 2.16. Цифри 1, 2, ..., 9 розбили на три групи. Довести, що добуток цифр в одній із груп не менший за 72.

Доведення. Оскільки 9! = 1·2·...·9 = (8·9)·(3·4·6)·(7·5·2) = 70·722 = (712 – 1)(71 + 1) = 713 + 712 – – 71 – 1 > 713, то, згідно з принципом Діріхле, добуток цифр в одній із груп не менший за 72.

Приклад 2.17. 15 хлопчиків зібрали 100 грибів. Доведіть, що принаймні двоє з них зібрали однакову кількість.

Доведення. Припустімо, що твердження задачі неправильне. Тоді 15 хлопчиків зібрали щонайменше $0 + 1 + 2 + ... + 14 = 14 \cdot 15 : 2 = 105$ грибів. Це суперечить умові.

Узагальнений принцип Діріхле. Інколи буває корисним ще таке переформулювання принципу Діріхле: якщо одне з кількох чисел більше від їх середнього арифметичного, то серед цих чисел знайдеться інше, що є меншим за їх середнє арифметичне.

Приклад 2.18. У бригаді 7 чоловік, і їх сумарний вік становить 322 роки. Доведіть, що з них можна вибрати трьох осіб, сумарний вік яких не менший за 138 років.

Доведення. Оскільки середній вік членів бригади становить 46 років, то сумарний вік трьох найстарших людей не менший за $3 \cdot 46 = 138$ років.

Принцип Діріхле можна також сформулювати зовсім «по-науковому», не використовуючи кліток і зайців.

Нехай A і B – скінчені множини, причому m – кількість елементів множини A, а n – кількість елементів множини B ($m > n$). Тоді за будь-якого відображення множини A в множину B знайдуться два елементи множини A, що мають той самий образ.

Принцип Діріхле допускає також інші переформулювання й узагальнення. Але нас надалі більше цікавитимуть різні способи його застосування. Розглянемо кілька задач про знайомства, зустрічі тощо.

Приклад 2.19. Кілька футбольних команд проводять турнір в одне коло. Доведіть, що в будь-який момент турніру знайдуться дві команди, що зіграли до цього моменту однакову кількість матчів.

Доведення. Нехай n – кількість команд, що проводять турнір. Розгляньмо два випадки:

1) у даний момент турніру знайдеться команда, що не провела жодної гри;

2) протилежний.

Припустімо, що у випадку 1 така команда одна. Якби їх було дві, то все доведено. Тому у випадку 1 не буде жодної команди, що зіграла $n - 1$ матч доданого моменту. Тоді, згідно з принципом Діріхле, серед $n - 1$ команд (крім тієї, що не зіграла жодного матчу) можна вибрати дві, що зіграли однакову кількість матчів. Тут у ролі зайців виступають $n - 1$

команд, а в ролі кліток – можливі кількості матчів від 1 до n – 2, які вони зіграли. У випадку 2 кількість матчів, що провели команди до даного моменту, змінюється від 1 до n – 1. І тому знову, за принципом Діріхле, серед n команд знайдуться дві, що зіграли однакову кількість матчів.

Приклад 2.20. 10 друзів надіслали один одному святкові листівки. Кожний із них надіслав 5 листівок. Доведіть, що якісь двоє друзів відправили листівки один одному.

Доведення. Обчислімо, скільки всього пар людей можна утворити з 10 друзів: 10 · 9 : 2 = 45. Оскільки всього було відправлено 10 · 5 = 50 листівок, то, згідно з принципом Діріхле, на якусь пару друзів припадає дві листівки.

Приклад 2.21. У роботі деякого засідання брали участь 200 чоловік, причому кожен із них був знайомий не менше ніж зі 100 присутніми. Доведіть, що за круглий стіл для 4 осіб можна посадити 4 із присутніх так, щоб кожен із них сидів між своїми знайомими.

Доведення. Припустімо, що серед присутніх є двоє незнайомих A і B, інакше все доведено. Оскільки кожен із A і B знайомий не менше ніж зі 100 присутніми, то вони матимуть принаймні двох спільних знайомих (200 > 198). Тоді ці двоє спільних знайомих разом з A і B утворюватимуть шукану четвірку осіб.

Приклад 2.22. У конгресі брали участь 2000 вчених, одні з них були раніше знайомі один з одним, інші – ні. При цьому виявилося: кожні двоє вчених, які мають однакову кількість знайомих, не мають спільних знайомих. Доведіть, що серед присутніх на конгресі вчених знайдеться вчений, знайомий лише з одним учасником конгресу.

Доведення. Нехай A – це вчений, який має найбільшу кількість знайомих серед присутніх (або одного з них, якщо їх кілька). Усіх знайомих A позначимо $A_1, A_2, ..., A_k$. Згідно з умовою задачі, усі A_i (i = 1, ..., k) мають різну кількість знайомих, що змінюється від 1 до k. Тоді знайдеться такий учений, котрий має рівно одного знайомого.

Приклад 2.23. Уздовж круглого столу рівномірно розміщено таблички з прізвищами дипломатів, які беруть участь у перемовинах. Після початку перемовин виявилося, що жоден із дипломатів не сидить напроти своєї таблички. Чи можна повернути стіл так, щоб принаймні двоє дипломатів сиділи напроти своїх табличок?

Розв'язок. Зауважмо, що серед усіх можливих n положень столу завжди можна вибрати одне, коли якийсь дипломат сидить напроти своєї таблички. Тоді за умови, що за такого положення столу такого дипломата немає, згідно з принципом Діріхле, можна повернути стіл так, щоб принаймні двоє дипломатів сиділи напроти своїх табличок. Часто для розв'язування задач, де застосовують принцип Діріхле, потрібно не лише показати, що чисел, які задовольняють певну властивість, є не більше за деяке k, але й конструктивно вказати множину з k елементів, що таку властивість має.

Приклад 2.24. Картки пронумеровані послідовно цілими числами від 1 до $2n + 1$. Яку найбільшу кількість карток можна вибрати так, щоб жоден із номерів не дорівнював сумі якихось двох інших номерів карток?

Розв'язок. Припустімо, що таких карток існує не менше ніж $n + 2$. Розташувавши вибрані картки за порядком зростання їх номерів, віднімемо від усіх номерів найменший номер картки. Отримаємо $n + 1$ різних чисел, відмінних від 0. Тоді, згідно з принципом Діріхле, отримана множина має принаймні один спільний із початковою номер (без картки з найменшим номером), тобто умови задачі не виконуються. Легко перевірити, що для $n + 1$ картки з непарними номерами $\{1, 3, 5, ..., 2n + 1\}$ умови задачі вже виконуються.

<u>Обов'язкові завдання</u>

1. Скільки існує виграшних комбінацій з 2 карт у грі в «очко»?

Для тих, хто не знає: виграє комбінація 10 + ТУЗ (11 очок) = 21 очко, і давайте вважатимемо виграшною комбінацію з двох тузів *(порядок карт у будь-якій парі не має значення).* До речі, не треба вважати приклад примітивним. «Блек джек» – це мало не єдина гра, для якої існує математично обґрунтований алгоритм, який дає змогу вигравати в казино.

2. У ліфт 12-поверхового дому зайшли 3 пасажири. Кожний незалежно від інших з однаковою імовірністю може вийти на будь-якому (починаючи з 2-го) поверху. Скількома способами:

1) пасажири можуть вийти на тому самому поверсі (порядок виходу не має значення);

2) дві людини можуть вийти на одному поверсі, а третя – на другому;

3) люди можуть вийти на різних поверхах;

4) пасажири можуть вийти з ліфту?

І тут часто перепитують, поточнюю: якщо 2 або 3 людини виходять на одному поверсі, то порядок виходу значення не має. ДУМАЙТЕ, використовуйте формули і правила додавання / добутку комбінацій. У випадку складності корисно надати пасажирам імена й поміркувати, у яких комбінаціях вони можуть вийти з ліфта. Не треба засмучуватися, якщо щось не виходить, так, наприклад, пункт № 2 достатньо підступний.

3. Олексій займається спортом, причому 4 дні на тиждень – легкою атлетикою, 2 дні – силовими вправами і 1 день відпочиває. Скількома способами він може скласти собі графік занять на тиждень?

Формула тут не придатна, оскільки враховує перестановки, що збігаються (наприклад, коли міняються місцями силові вправи в середу із силовими вправами в четвер). І знову – за фактом ті ж 2 силові тренування можуть сильно відрізнятися одне від іншого, але згідно з умовою задачі (з погляду розкладу) вони вважаються однаковими елементами.

4. У гаманці знаходиться досить велика кількість 1-, 2-, 5- і 10-копійчаних монет. Скількома способами можна дістати три монети з гаманця? З метою самоконтролю дайте відповідь на пару простих запитань:
1) Чи можуть всі монети у вибірці бути різними?
2) Назвіть «найдешевшу» і «найдорожчу» комбінацію монет.

5. Автомобільний номерний знак складається з 3 цифр і 3 літер. При цьому недопустимим є номер з трьома нулями, а літери вибираються з набору А, В, Е, К, М, Н, О, Р, С, Т, У, Х (використовуються тільки ті літери кирилиці, написання яких збігається з латинськими літерами). Скільки різних номерних знаків можна скласти для регіонів?

6. У мішку лежать 5 чорних і 5 білих кульок. Яку найменшу кількість кульок потрібно взяти з мішка, щоб серед них точно виявилося 3 кульки одного кольору?

7. Учені дослідили, що кількість голок у їжака не перевищує 200 тисяч. Доведіть, що з 250 тисяч їжаків можна вибрати принаймні дві, що мають однакову кількість голок.

8. У Верховну Раду обрано 336 народних депутатів, причому серед них 123 жінки і 245 особи – представники правих сил. Доведіть, що серед правих є не менше ніж 32 жінки.

9. У школі 30 класів і 1000 учнів. Доведіть, що у школі є клас, у якому не менше ніж 34 учнів.

10. На Землі більше ніж 4 мільярди людей, вік яких не перевищує 100 років. Доведіть, що на Землі є двоє людей, що народилися тієї самої секунди.

11. Яку найменшу кількість карток спорт лото «6 із 49» потрібно купити, щоб на одній із них обов'язково було вгадано хоча б один номер?

12. У магазин завезли 25 ящиків із трьома різними сортами яблук (у кожному ящику яблука лише одного сорту). Доведіть, що серед них є принаймні 9 ящиків одного сорту яблук.

13. У школі навчаються 962 учні. Доведіть, що принаймні у двох учнів збігаються ініціали.

14. У темній коморі лежать черевики одного розміру: 10 пар чорних і 10 пар коричневих. Яку найменшу кількість черевиків потрібно взяти з комори, щоб серед них точно можна було вибрати одну пару одного кольору (у темряві не можна відрізнити не тільки колір черевика, але й лівий від правого)?

15. Із повного набору доміно викинули всі кісточки із шістками. Чи можна викласти в ланцюг усі кісточки, що залишилися?

16. У похід пішли 12 туристів. Наймолодшому з них – 20 років, а найстаршому – 30. Чи є серед них однолітки?

17. У школі навчаються 400 учнів. Доведіть, що принаймні двоє з них народилися в один день.

18. Хлопчик мав 100 табличок із числами 1, 2, ..., 100, але загубив 79 із них. Чи обов'язково серед решти табличок знайдуться чотири такі, що сума чисел на двох із них дорівнюватиме сумі чисел на двох інших?

19. Доведіть, що серед будь-яких шести цілих чисел знайдуться два числа, різниця яких буде кратною 5.

20. У сьомому класі навчаються 30 учнів. У диктанті один учень припустився 12 помилок, а решта – менше. Довести, що принаймні троє учнів припустилися однакової кількості помилок.

21. Дано 12 довільних двоцифрових чисел. Доведіть, що серед них є два, різниця яких дорівнює двоцифровому числу, записаному однаковими цифрами.

Комбінаторні схеми

Нехай $A = \{a_1, a_2, \dots, a_n\}$ – множина об'єктів, число $k \leq n$ (k може бути нулем). Як можна вибрати з множини A рівно k об'єктів? Можна вибрати ці об'єкти послідовно без повернення, один за другим k разів;

1) вибирати з поверненням, а саме вибирати один об'єкт з множини A, повернути назад, вибрати наступний об'єкт, повернути назад і т. д. k разів;
2) вибрати без повернення відразу всі k об'єктів однією купою без повторів (невпорядковано);
3) вибрати невпорядковано k об'єктів, які можуть повторюватися.

Прикладом такого вибору може бути така задача: нехай у магазині є безкінечна кількість лимонів, яблук і груш. Розглянемо множину $A = \{лимон,\ яблуко,\ груша\}$. Ми хочемо купити 5 фруктів. Якою кількістю способів ми зберемо в торбинку 5 фруктів? Оскільки порядок неважливий, то ця кількість дорівнюватиме кількості способів вилучити 5 фруктів невпорядковано з множини $A = \{лимон,\ яблуко,\ груша\}$ з повтореннями.

Розберемо їх детальніше. Попередньо зазначимо, що $\emptyset$ позначає порожню множину і вважається (тут $n!$ читається n-факторіал):
0! = 1, 1! = 1, $n! = n \cdot (n-1)! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$,

Відповідно до чотирьох типів вибірок введемо позначення:

1) число k-розміщень без повторень з n об'єктів позначається, як A_n^k (читається A з n по k);
2) число k-розміщень з повтореннями з n об'єктів позначається $\overline{A_n^k}$;
3) число k-комбінацій без повторень з n об'єктів позначається C_n^k (читається C з n по k, це позначення придумав Паскаль у XVII сторіччі, але зараз в англомовному світі прийнято позначення $\binom{n}{k} = C_n^k$;
4) число k-комбінацій з повтореннями з n об'єктів позначається $\overline{C_n^k}$.

Розміщення без повторень

Вибірки, у яких всі елементи різні, а їх порядок у вибірці є суттєвим, називають **розміщеннями без повторень.**

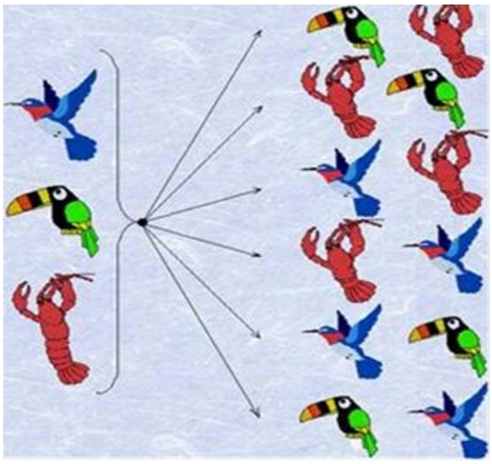
У цьому визначенні виділимо такі характеристичні ознаки розміщень без повторень:

1) елементи вибірки різні;
2) порядок елементів важливий;
3) $\mathbf{0 \le k \le n}$.

Приклад 2.25. З трьохелементної множини можна скласти шість двохелементних вибірок. Кожна така вибірка не містить однакових елементів. Порядок елементів у вибірці є суттєвим, оскільки, наприклад, перша і друга вибірки вважаються різними, хоча складаються з однакових елементів. Отже, кожна така вибірка є розміщенням без повторень з 3 елементів по 2.

Розв'язок. Кількість розміщень без повторень із n елементів по K елементів позначають через A_n^k.

Після введення позначення ми можемо записати, що нам треба знайти A_3^2. Скількома способами ми могли б обрати одного представника живої природи? Оскільки всі вони різні, то трьома різними способами (обравши або першого, або другого, або третього з них). Тому $A_3^1 = 3$. Після нашого вибору першого представника залишилося 2 необраних. З них 2 ми обираємо когось на роль другого у вибірці. Скількома різними способами це можна зробити? Так, двома, адже, кожний із двох ще необраних може стати другим у вибірці. За правилом добутку кількість способів вибору двох представників дорівнює 3×2, тобто $A_3^2 = 3 \times 2 = 6$.



Приклад 2.26. Скількома способами можна скласти розклад занять із 5 різних уроків, якщо у класі вивчається 10 різних предметів?

Розв'язок. Кожний розклад – це вибір 5 різних предметів із списку, у якому 10 предметів. Нам важливий лише набір із 5 предметів, а і їх порядок. Отже, кожен розклад – це впорядкована підмножина з 5 різних елементів множини з 10 елементів, тобто розміщення без повторень з 10 елементів по 5. Нам треба підрахувати кількість таких множин, тобто A_{10}^5.

Скількома способами ми могли б скласти розклад з одного предмета? Оскільки всі вони різні, то 10 різними способами (вибравши або перший, або другий, або…, або десятий предмет). Тому $A_{10}^1 = 10$. Після нашого вибору першого предмета в розкладі у списку залишилося 9 предметів. Із цих 9 ми вибираємо будь-який предмет на роль другого у розкладі. Скількома різними способами це можна зробити? Так, 9, адже, кожний з 9 ще не обраних предметів може стати другим у розкладі. За правилом добутку кількість способів вибору у розклад двох перших предметів дорівнює 10×9, тобто $A_{10}^2 = 10 \times 9$. Тобто, аналогічно міркуючи, можемо підрахувати кількість способів вибору у розклад перших трьох і чотирьох предметів. Це $A_{10}^3 = 10 \times 9 \times 8$ і $A_{10}^4 = 10 \times 9 \times 8 \times 7$ відповідно. Тоді кількість способів скласти розклад із 5 різних предметів дорівнюватиме $A_{10}^5 = 10 \times 9 \times 8 \times 7 \times 6 = 30\ 240$.

Розміщення з повтореннями

Вибірки, у яких елементи можуть бути однаковими і порядок їх розташування є суттєвим (тобто вибірки будуть різними, навіть якщо вони відрізняються тільки порядком розташування в них елементів) називають **розміщенням із повтореннями**. У цьому визначенні виділимо такі характеристичні ознаки розміщень з повтореннями:

1) елементи можуть бути однаковими;
2) порядок елементів важливий.

Кількість розміщень із повтореннями з n елементів по k елементів позначають через $\overline{A_n^k}$, і обчислюють за формулою $\overline{A_n^k} = n^k$. Це випливає з узагальненого правила добутку: на першому місці може бути будь-який з n даних елементів, на другому – також будь-який з n елементів, …, на k-му місці – також будь-який з n даних елементів.

Приклад 2.27. Скільки чотирилітерних слів можна скласти з літер М і А? Випишіть ці слова й перевірте отриманий результат.

Розв'язок. Складемо декілька таких «слів»: ММMA, MAMA, MAAA… Бачимо, що склад вибірки змінюється, порядок елементів у вибірці істотний. Отже, це розміщення з повтореннями з 2 літер М і А по 4. $\overline{A_2^4} = 2^4 = 16$. Випишемо всі ці 16 «слів»: MMMA, MMAA, MAAA, MAMA, AAAM, AAMM, AMMM, AMAM, AMMA, MAAM, AAMA, AMAA, MMAM, MAMM, MMMM, AAAA.

Відповідь: 16.

Приклад 2.28. Уздовж дороги розташовані 6 світлофорів, кожен із них має 3 стани: «червоний», «жовтий», «зелений». Скільки може бути різних ситуацій на дорозі, що спричинені станами світлофорів?

Розв'язок. Випишемо декілька комбінацій: ЧЧЖЗЗЧ, ЖЖЖЖЖЖ, ЗЖЖЗЧЧ…. Ми бачимо, що склад вибірки змінюється і порядок елементів істотний (адже, якщо, наприклад, у вибірці ЧЧЖЗЗЧ поміняти місцями Ж і З, ситуація на дорозі буде іншою). Тому застосовуємо формулу розміщень із повтореннями з 3 по 6:$\overline{A_3^6} = 3^6 = 729$.

Відповідь: 729.

Поєднання (комбінації) без повторень

(C_n^k без повторень, $\overline{C_n^k}$ з повтореннями) (неупорядкована вибірка з повтореннями або без повторень).

У підручниках надається лаконічне й не дуже зрозуміле визначення поєднання, тому моє формулювання буде не дуже раціональним, але, сподіваюсь, зрозумілим.

Поєднаннями називають різні комбінації з *k* об'єктів, що вибрані з множини *n* різних об'єктів, які відрізняються один від одного хоча б одним об'єктом. Інакше кажучи, окреме поєднання – це унікальна вибірка з *k* елементів, у якій неважливий їх порядок (розташування). Загальна ж кількість таких унікальних поєднань розраховується за формулою $C_n^k = \frac{n!}{(n-k)! \cdot k!}$.

Кількість сполук без повторень з *n* елементів по *k* визначають, виходячи з числа розміщень без повторень A_n^k з урахуванням того, що різних неупорядкованих вибірок (підмножин вихідної множини) буде менше в число раз, яке дорівнює перестановці без повторень з *k* елементів P_k: $C_n^k = \frac{A_n^k}{P_k} \frac{n!}{(n-k)! \cdot k!}$.

Приклад 2.29. У ящику є 15 деталей. Скількома способами можна взяти 4 деталі?

Розв'язок. По-перше, знову звертаю увагу на те, що, за логікою умови, деталі вважаються **різними**, навіть якщо вони насправді однотипні й візуально однакові *(у цьому випадку їх можна, наприклад, пронумерувати)*.

У задачі йдеться про вибірку із 4 деталей, у якій не має значення їх «подальша доля» – грубо кажучи, «просто обрали 4 штуки і все». Таким чином, маємо поєднання деталей. Рахуємо їх кількість:

$$C_{15}^{4} = \frac{15!}{(15-4)! \cdot 4!} = \frac{15!}{11! \cdot 4!} = (*)$$

Тут, звісно ж, не треба обраховувати такі великі числа $11! = 39916800, \quad 15! = 1307674368000$

У схожій ситуації я пропоную використовувати такий прийом: у знаменнику вибираємо найбільший із **факторіалів** (у цьому випадку 11!) і скорочуємо на нього дріб. Для цього чисельник потрібно представити у вигляді 15! = $11! \cdot 12 \cdot 13 \cdot 14 \cdot 15$. Розпишу дуже детально:

$$(*) = \frac{11! \cdot 12 \cdot 13 \cdot 14 \cdot 15}{11! \cdot 4!} = \frac{12 \cdot 13 \cdot 14 \cdot 15}{4!} = \frac{12 \cdot 13 \cdot 14 \cdot 15}{24} = 1365$$

способами можна взяти 4 деталі з ящику.

Ще раз: що це означає? Це означає, що з набору 15 різних деталей можна скласти *одну тисячу триста шістдесят п'ять* **унікальних** поєднань із 4 деталей. Тобто кожна така комбінація із чотирьох деталей відрізнятиметься від інших комбінацій хоча б однією деталлю.

Відповідь: 1365 способами.

Формулі $C_n^k = \frac{n!}{(n-k)! \cdot k!}$ потрібно приділити особливу увагу, оскільки вона є «хітом» комбінаторики. При цьому корисно розуміти й без будь-яких обчислень записувати «крайні» значення:

$C_n^0 = 1, \quad C_n^1 = n, \quad C_n^{n-1} = n, \quad C_n^n = 1$. Стосовно розібраної задачі: $C_{15}^0 = 1$ – тільки одним способом можна не вибрати жодної деталі;

$C_{15}^1 = 15$ способами можна взяти 1 деталь (будь-яку з п'ятнадцяти);

$C_{15}^{14} = 15$ способами можна взяти 14 деталей (при цьому якась одна з 15 залишиться в ящику);

$C_{15}^{15} = 1$ – тільки одним способом можна вибрати всі п'ятнадцять деталей.

Рекомендую уважно ознайомитися **з біномом Ньютона і трикутником Паскаля**, за яким, до речі, дуже зручно виконувати перевірку обчислень C_n^m за невеликих значень «*n*».

Приклад 2.30. У шаховому турнірі бере участь k людей і один з одним грає тільки по одній партії. Скільки всього партій зіграно в турнірі?

Відразу ж зорієнтуємося за турнірною таблицею розміром $k \cdot k$ клітинок, у якій результат кожної партії враховується двічі і, крім того, закреслюються клітинки «головної діагоналі» *(оскільки учасники не грають самі із собою)*. Виходячи з приведених міркувань, загальна кількість зіграних партій розраховується за формулою $n = \frac{k \cdot k - k}{2}$. Таке рішення цілком коректне, але колись один зі студентів помітив, що насправді тут можна керуватися самими банальними поєднаннями: $C_k^2 = \frac{k!}{(k-2)! \cdot 2!} = \frac{1 \cdot 2 \cdot 3 \cdot \ldots \cdot (k-2)(k-1)k}{1 \cdot 2 \cdot 3 \cdot \ldots \cdot (k-2) \cdot 2} = \frac{(k-1)k}{2}$ різних пар можна скласти з *k* суперників *(хто грає білими, хто чорними – неважливо)*.

Еквівалентною є задача про рукостискання: у відділі працюють *k* чоловіків і кожний з кожним вітаються за руку, скільки рукостискань вони здійснюють? Ну а висновків тут два: по-перше, не все очевидне є очевидним, по-друге, не бійтеся вирішувати задачі «нестандартно»!

Поєднання (комбінації) з повтореннями

Суму *k* кратностей усіх елементів називають порядком сполуки. Сполуку з повтореннями *k*-го порядку, що складена з множини, яка містить *n* елементів, називають також комбінацією з повторенням з *n* елементів по *k*. Якщо $k_1, k_2, \ldots, k_n$– кратності елементів $a_1, a_2, \ldots, a_n$, то за визначенням $k_1 + k_2 + \ldots + k_n$є порядок комбінації

$$\overbrace{a_1 a_1 a_1 \ldots a_1}^{k_1} \overbrace{a_2 a_2 a_2 \ldots a_2}^{k_2} \ldots \overbrace{a_n a_n a_n \ldots a_n}^{k_n}$$

Кількість вибірок з повтореннями з *n* елементів по *k* виражають формулою:

$$\overline{C_n^k} = C_{n+k-1}^k = C_{n+k-1}^{n-1} = \frac{(n+k-1)!}{k! \cdot (n-1)!}$$

Характерна особливість цього виду комбінацій полягає в тому, що вибірка проводиться з декількох груп, кожна з яких складається з однакових об’єктів.

Приклад 2.31. У студентській столовій продають сосиски в тісті, ватрушки й пончики. Скількома способами можна купити 5 пиріжків?

Розв’язок. Відразу зверніть увагу на типовий критерій поєднань із повтореннями – за умовою на вибір запропонована ні множина об’єктів як така, а **різні види** об'єктів; при цьому мається на увазі, що в продажу є не менш ніж 5 хот-догів, 5 ватрушок і 5 пончиків. Пиріжки в кожній групі, зрозуміло, відрізняються, бо абсолютно ідентичні пончики можна змоделювати хіба що на комп'ютері. Однак фізичні характеристики пиріжків за умовою задачі не суттєві, і хот-доги / ватрушки / пончики у своїх групах вважаються однаковими.

Що може бути у вибірці? По-перше, слід зазначити, що у вибірці обов'язково будуть однакові пиріжки (оскільки вибираємо 5 штук, а на вибір запропоновано 3 види). Варіанти тут на будь-який смак: 5 хот-догів, 5 ватрушок, 5 пончиків, 3 хот-доги + 2 ватрушки, 1 хот-дог + 2 ватрушки + 2 пончики і т.д. Як і за «звичайних» поєднань, порядок вибору й розміщення не має значення – просто вибрали 5 штук пиріжків і все.

Застосуємо формулу $C^m_{n(повт)} = C^m_{n+m-1} = \frac{(n+m-1)!}{(n-1)!\cdot m!}$ кількості поєднань із повтореннями: $C^5_{3(повт)} = C^5_{3+5-1} = C^5_7 = \frac{7!}{2!\cdot 5!} = \frac{6\cdot 7}{2!} = 21$ способом можна купити 5 пиріжків.

Відповідь: 21.

Який висновок можна зробити з багатьох комбінаторних задач? Найскладніше – це розібратися в умові задачі.

Ще декілька задач цього типу:

Приклад 2.32. У магазині продають 4 сорти тістечок: заварне, білкове, «Грибок», «Нічка». Скількома способами можна купити 8 тістечок?

Розв'язок. Розглянемо множину S = {з, б, г, н}, у якій кожен елемент кодує назву тістечка по першій літері. Будь-який вибір 8 тістечок породжує комбінацію елементів множини S, при цьому порядок запису елементів немає значення. Наприклад, вибравши 2 заварні тістечка, 4 білкові й по одному тістечку інших сортів, маємо комбінацію *ззббббгн*, що є поєднанням з повтореннями з 4 по 8. Навпаки, будь-яке таке поєднання однозначно дасть деякий набір тістечок. Отже, число способів купити вказане число тістечок одне $C^8_{4(повт)} = C^8_{4+8-1} = C^8_{11} = \frac{11!}{3!\cdot 8!} = \frac{9\cdot 10\cdot 11}{6} = \frac{3\cdot 10\cdot 11}{2} = 3\cdot 5\cdot 11 = 165.$

Відзначимо, що можна було зробити підрахунок не за формулою, а безпосередньо, застосувавши процедуру, описану під час виведення цієї формули. Кожен набір із 8 тістечок можна закодувати послідовністю з 8 нулів і 3 одиниць. Наприклад, набору *ззббббгн* відповідає послідовність

11011110101, а набору *ггггнннн* послідовність 00111101111 (заварних і білкових тістечок не вибрали жодного). Послідовність 11110011011 однозначно визначає такий набір тістечок: заварні – 4 штуки, «Грибок» – 2 штуки, «Нічка» – 2 штуки, білкових немає. Отже, 8 тістечок можна вибрати (8, 3) = 165 числом способів.

Відповідь: 165 способами.

Приклад 2.33. В умовах попереднього прикладу знайдемо, скількома способами можна купити 8 тістечок за додаткової умови: потрібно купити хоча б по одному тістечку кожного сорту.

Розв'язок. Додаткова умова означає, що у вибірці елементів множини S свідомо повинні бути елементи з, б, г, *н*. Приберемо з вибірки по одному елементу кожного типу. Після цього отримаємо поєднання, що містить тільки 4 елементи з тих же 4 сортів. Число таких поєднань $C_7^3 = \frac{7!}{4!\cdot 3!} = \frac{5\cdot 6\cdot 7}{6} = 35.$

Ототожнення набору об'єктів з послідовністю з нулів і одиниць є важливою ідеєю для вирішення комбінаторних завдань. Цей прийом часто називають *0–1-кодуванням.*

Приклад 2.34. Скількома способами шість однакових олівців можна розподілити між трьома дітьми?

Розв'язок. Оскільки олівці однакові, то різні способи розподілу олівців між дітьми відрізнятимуться лише кількістю олівців у кожної дитини. Застосуємо ідею 0–1-кодування. Зобразимо кожен олівець цифрою 1. Отримаємо набір з 6 одиниць. Для того щоб розподілити їх між трьома дітьми, додамо до набору дві цифри 0, що розіб'є 6 одиниць на три групи. Отриманий набір взаємно однозначно розподіляє олівці між дітьми. Наприклад, набір 11101011 означає, що перша дитина отримала три олівця, друга – один олівець, а третя – два олівця. Залишилося порахувати число наборів із двох нулів і шести одиниць, число яких дорівнює Pg (2,6) або C_8^2, тобто 28.

Відповідь: 28.

У всіх розглянутих вище прикладах завдань цього пункту потрібно порахувати число поєднань із повтореннями. Як було доведено, це число можна знайти за формулою поєднань із повтореннями. Проте замість того, щоб переводити завдання на мову поєднань, можна безпосередньо скористатися ідеєю 0–1-кодування, що було продемонстровано в ряді прикладів. У деяких випадках завдання можна вирішити, не вдаючись до ідеї кодування або за формулою.

Приклад 2.35. Серед азартних ігор поширена гра в доміно. Кістка доміно являє собою комбінацію двох чисел, утворену з 7 елементів: 0, 1, 2, 3, 4, 5, 6. Комбінації чисел невпорядковані (1: 2 і 2: 1 – це та сама кістка), при цьому можливі повторення чисел (0: 0, 1: 1, 2: 2, ... – так звані дублі). Порахуємо, скільки всього існує кісток доміно.

Розв'язок. Неважко бачити, що кістка – це поєднання з повтореннями з 7 по 2. Їх число можна знайти за формулою $C_{7+2-1}^{2} = C_{8}^{2} = \frac{8!}{6!\cdot 2!} = \frac{7\cdot 8}{2} 28.$

Знайдемо це число іншим способом. Спочатку переберемо все дублі: їх 7. Порахуємо кількість кісток, які не є дублями. Їх число дорівнює числу сполучень (без повторень) з 7 по 2, тобто = 21. Усього маємо 7 + 21 = 28 кісток доміно.

Відповідь: 28 кісток.

З мого особистого досвіду можу сказати, що поєднання з повтореннями – найбільш рідкісний «гість» на практиці, чого не скажеш про зазначений нижче вид комбінацій.

Перестановка (P_n без повторень, $P(k_1, \ldots, k_m)$ з повтореннями) (упорядкована повна вибірка без повторень та з повтореннями).

Перестановка без повторень

Розглянемо задачу упорядкування n-елементної множини A (формування впорядкованої вибірки довжини n , складеної з n-елементної множини). Отримані при цьому вибірки відрізнятимуться лише порядком слідування елементів. Такі вибірки називають **перестановками без повторень** із n елементів.

Число перестановок без повторень із n елементів позначається P_n. До перестановок без повторень можна дійти, вважаючи, що здійснюється розміщення без повторень із n елементів по n.

Інакше кажучи, **перестановками** називають комбінації, що складаються з тих самих n **різних** об'єктів, які відрізняються тільки порядком їх розташування. Кількість усіх можливих перестановок виражається формулою $P_n = A_n^n = \frac{n!}{(n-n)!} = n!$

Відмінною рисою перестановок є те, що в кожній з них бере участь **уся** множина, тобто **всі** n об'єктів. Наприклад, дружня родина.

Приклад 2.36. Скількома способами можна розсадити 5 людей за столом?

Розв'язок. Використаємо формулу кількості перестановок: $P_5 = 5! = 120$

Відповідь: 120 способами.

Неймовірно, але факт. Зверніть увагу на те, якої форми стіл або взагалі всі люди сіли (встали, полягали) на лавку вздовж стіни, – важлива лише кількість об'єктів і їх взаємне розташування.

Приклад 2.37. Скільки чотиризначних чисел можна скласти із чотирьох карток із цифрами 0, 5, 7, 9? Для того щоб скласти чотиризначне число, потрібно застосувати всі чотири картки *(цифри на яких **різні!**)*, і це дуже важливо усвідомити для застосування формули $P_n = n!$. Вочевидь, що, переставляючи картки, ми отримуватимемо різні чотиризначні числа… стоп, а тут все в порядку? Взагалі, це характерна риса комбінаторних і ймовірнісних задач – тут **потрібно думати**. І часто думати по-життєвому. Ні, звісно, я не пропоную негайно проробляти інші розділи математики, але й інтеграли можна навчитися вирішувати механічно.

Розв'язок. Знайдемо кількість всіх можливих перестановок 4 карток.

Коли картка з нулем опиняється на 1-му місці, то число стає тризначним, тому ці комбінації слід виключити. Якщо нуль перебуває на 1-му місці, тоді 3 цифри, що залишилися в молодших розрядах, можна переставити способами:

0579
0597
0759
0795
0957
0975

Примітка: оскільки карток небагато, то тут не складно перерахувати всі такі варіанти.

Таким чином, із запропонованого набору можна скласти: 24 – 6 = 18 чотиризначних чисел.

Відповідь: 18.

Перестановки з повтореннями

Під час визначення **перестановок без повторень** ми розглядали ситуацію, коли в початковій n-множині A всі елементи унікальні. Однак існують ситуації, коли множина може містити деяку кількість однотипних елементів.

Визначення. Число різних перестановок, які можна побудувати з n елементів, серед яких k_1 елементів першого типу, k_2 елементів другого типу, …, k_m елементів m-го типу, дорівнює:

$$P(k_1, k_2, \ldots, k_m) = \frac{n!}{k_1! \cdot k_2! \cdot \ldots \cdot k_m!}.$$

У перестановках з повторенням, як і у «звичайних» перестановках, бере участь **відразу вся множина об'єктів**, але є одне але: у цій множині один або більша кількість елементів (об'єктів) повторюються. Зустрічайте ще один стандарт.

Приклад 2.38. Скільки різних сполучень літер можна отримати перестановкою карток із такими літерами: К, О, Л, О, К, О, Л, Ь, Ч, И, К?

Розв'язок. У тому випадку, коли всі літери були б різними, то слід було б застосувати банальну формулу P_n, однак зрозуміло, що для запропонованого набору карток деякі маніпуляції спрацьовуватимуть «вхолосту», так, наприклад, якщо поміняти місцями будь-які дві картки з літерами «К» в будь-якому слові, то отримаємо те ж саме слово. Причому фізично картки можуть сильно відрізнятися одна від одної: одна може бути круглою з надрукованою літерою «К», інша – квадратна з намальованою літерою «К». Але в нашій задачі навіть такі картки **вважаються однаковими**, оскільки за умовою питають про сполучення літер.

Усе дуже просто: усього 11 карток, серед яких літера

К	–	повторюється	3	рази;
О	–	повторюється	3	рази;
Л	–	повторюється	2	рази;
Ь	–	повторюється	1	раз;
Ч	–	повторюється	1	раз;
И	–	повторюється	1	раз.

Перевірка: 3 + 3 + 2 + 1 + 1 + 1 = 11, що потрібно було довести.

За формулою **кількості перестановок з повтореннями**: $P_{11(поем)} = \dfrac{n!}{n_1!\cdot n_2!\cdot n_3!\cdot \ldots \cdot n_k!} = \dfrac{11!}{3!\cdot 3!\cdot 2!\cdot 1!\cdot 1!\cdot 1!} = \dfrac{39916800}{6\cdot 6\cdot 2} = 554400$ різних сполучень літер можна отримати. Понад пів мільйона!

На практиці вважається припустимим не записувати загальну формулу, крім того, опускають одиничні факторіали: $P_{11(поем)} = \dfrac{11!}{3!\cdot 3!\cdot 2!} = \dfrac{39916800}{6\cdot 6\cdot 2} = 554400$

Але попередні коментарі про повторювані літери обов'язкові!

Відповідь: 554400.

Розбиття множини на підмножини

Нехай дана n-множина A. Говорять, що множина A розбита на k підмножин A_i, де (1, 2, ..., k), якщо:

1) $A_i \neq \emptyset$, $i \in \{1,2,\ldots,k\}$;

2) $A_i \cap A_j = \emptyset$, $i, j \in \{1,2,\ldots,k\}$;

3) $\bigcup\limits_{i=1}^{k} A_i = A$.

Позначимо число елементів у підмножині A_i через $n(A_i) = n_i$. Очевидно, що $n_1 + n_2 + \ldots + n_k = n$. Кількість розбиттів множини на підмножини позначимо: $C(n; n_1, n_2, \ldots, n_k)$.

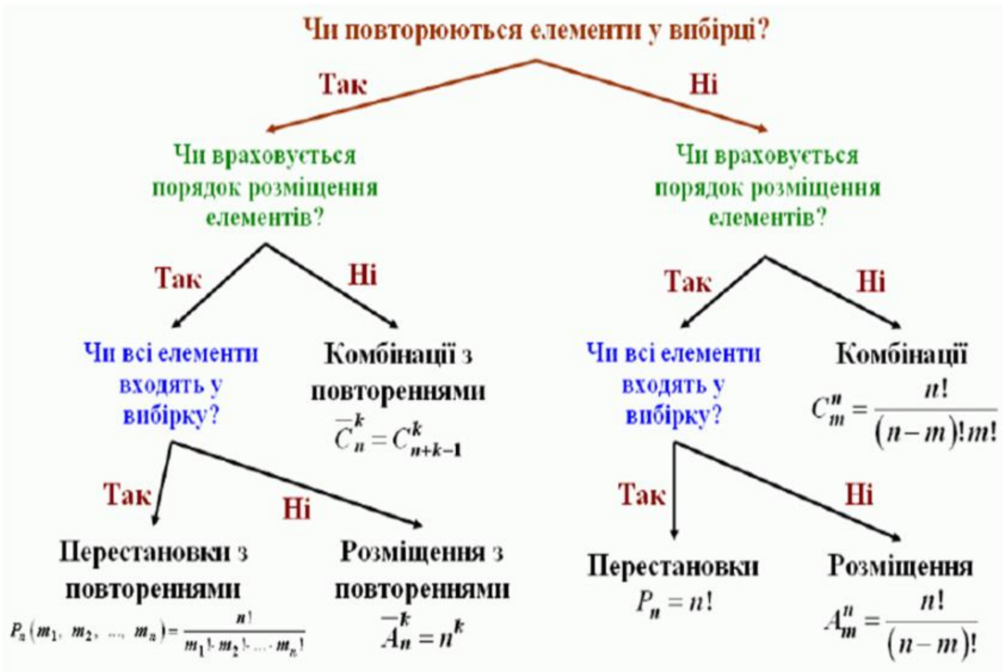


Рис. 44. Алгоритм вибору комбінаторної схеми

Визначимо кількість розбиттів. Кількість способів вибору елементів підмножини A_1 дорівнює кількості сполук $C_n^{n_1}$.

Кількість способів вибору елементів підмножини 2 A дорівнює кількості $C_{n-n_1}^{n_2}$.

Кількість способів вибору цих двох підмножин дорівнює і $C_n^{n_1} \cdot C_{n-n_1}^{n_2}$ і так далі. Таким чином, кількість вибору всіх розбиттів дорівнює: $C_n^{n_1} \cdot C_{n-n_1}^{n_2} \cdot C_{n-n_1-\ldots-n_{k-1}}^{n_k} =$

$= \frac{n!}{n_1!(n-n_1)!} \cdot \frac{(n-1)!}{n_2!(n-n_1-n_2)!} \cdot \ldots \cdot \frac{(n-n_1-\ldots-n_{k-1})!}{n_k!(n-n_1-\ldots-n_{k-1})!} = \frac{n!}{n_1! \cdot n_2! \cdot \ldots \cdot n_k!}$.

$$C_n^{n_1} \cdot C_{n-n_1}^{n_2} \cdot \ldots \cdot C_{n-n_1-\ldots-n_{k-1}}^{n_k} = \frac{n!}{n_1! \cdot n_2! \cdot \ldots \cdot n_k!}$$

Порівняємо цей вираз із формулою для перестановок із повтореннями. $P(n_1, n_2, \ldots, n_k) = \frac{n!}{n_1! \cdot n_2! \cdot \ldots \cdot n_k!}$.

Можна зробити висновок, що **явище розбиття множини на підмножини** й **перестановки з повтореннями** – це та сама комбінаторна дія з різною інтерпретацією.

Приклад 2.39. Із пропорції $C_x^y : C_x^{y-1} : C_x^{y-2} = 3 : 3 : 2$ знайти x і y.

Розв'язок. Записавши окремо відношення першого члена пропорції до другого і другого до третього, після скорочення одержимо:

$$\frac{x!}{y!\,(x-y)!} : \frac{x!}{(y-1)!\,(x-y+1)!} = \frac{(y-1)!\,(x-y+1)!}{y!\,(x-y)!}$$

$$= \frac{(y-1)!\,(x-y)!\,(x-y+1)}{y!\,(x-y)!} = \frac{(x-y+1)}{y}$$

$$\frac{x!}{(y-1)!\,(x-y+1)!} : \frac{x!}{(y-2)!\,(x-y+2)!} = \frac{(y-2)!\,(x-y+2)!}{(y-1)!\,(x-y+1)!}$$

$$= \frac{(y-2)!\,(x-y+1)!\,(x-y+2)}{(y-2)!\,(y-1)(x-y+1)!} = \frac{(x-y+2)}{y-1}$$

З умови задачі отримуємо систему рівнянь: $\begin{cases} \frac{(x-y+1}{y} = 1 \\ \frac{(x-y+2}{(y-1)} = \frac{3}{2} \end{cases}$

Розв'язуємо систему алгебраїчних рівнянь:

$x - y + 1 = y$; $x = 2y + 1$.

$2x - 2y + 4 = 3y - 3$; $2x = 5y - 7$.

$4y - 2 = 5y - 7$; y=5; x = 9.

Тотожності для поєднань (комбінацій)

Основна формула кількості поєднань: $C_n^k = \frac{n!}{k!(n-k)!}$ дає змогу одержати ряд простих тотожностей. Розглянемо деякі з них.

Теорема 2.1. $C_n^k = C_n^{n-k}$.

Доведення. $C_n^{n-k} = \frac{n!}{(n-k)!(n-(n-k))!} = \frac{n!}{(n-k)!k!} = C_n^k$.

Теорема 2.2. $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$.

Доведення $C_{n-1}^k + C_{n-1}^{k-1} = \frac{(n-1)!}{k!(n-k-1)!} + \frac{(n-1)!}{(k-1)!(n-1-(k-1))!} =$

$$= \frac{(n-1)!}{k!\,(n-k-1)!} + \frac{(n-1)!}{(k-1)!\,(n-k)(n-k-1)!} =$$

$$= \frac{(n-k)(n-1)! + k(n-1)!}{k(k-1)!\,(n-k)(n-k-1)!} = \frac{(n-k+k)(n-1)!}{k!\,(n-k)!} = \frac{n!}{k!\,(n-k)!} = C_n^k.$$

Теорема 2.3. $C_n^i C_i^k = C_n^k C_{n-k}^{i-k}$.

Доведення. $C_n^i C_i^k = \frac{n!}{i!(n-i)!} \cdot \frac{i!}{k!(i-k)!} = \frac{n!}{k!(i-k)!(n-i)!} =$

$$= \frac{n!(n-k)!}{k!(i-k)!(n-i)!(n-k)!} = \frac{n!}{k!(n-k)!} \cdot \frac{(n-k)!}{(i-k)!(n-i)!} = C_n^k \cdot C_{n-k}^{i-k}.$$

Теорема 2.4. Біном Ньютона: $(x+y)^n = \sum_{k=0}^{n} C_n^k x^k y^{n-k}$.

$(x+y)^n = C_n^0 x^n + C_n^1 x^{n-1} y + \ldots + C_n^k x^{n-k} y^k + \ldots + C_n^k y^k$,

або $(x+y)^n = x^n + n x^{n-1} y + \ldots + \frac{n(n-1)\ldots(n-k+1)}{k!} x^{n-k} y^k + \ldots + y_n$, де n – натуральне число і $C_n^k x^{n-k} y^k = T_{k+1}$ є $(k+1)$-й член в розкладі біному $(k = 0, 1, 2, \ldots, n)$.

Наслідок 1. Сума біноміальних коефіцієнтів дорівнює 2^n: $\sum_{k=0}^{n} C_n^k = 2^n$.

Наслідок 2. $\sum_{k=0}^{n} (-1)^k C_n^k = 0$.

Теорема 2.5. $\sum_{k0}^{n} k C_n^k = n 2^{n-1}$.

Теорема 2.6. $C_{n+k}^k = \sum_{i=0}^{k} C_n^i C_k^{k-i}$

Приклад 2.40. Команда деякої ЕОМ записується у вигляді набору з восьми цифрових знаків – нулів і одиниць. Яка максимальна кількість різних команд?

Розв'язок. Оскільки для кожного набору можливі лише два значення (0 або 1), то максимальна кількість різних команд є $\underbrace{2\cdot 2\ldots\cdot 2} = 256$. Можна розмірковувати інакше. Розглянути всі двійкові числа від 00000000 до 11111111. Таких чисел також буде 256.

Приклад 2.41. У розкладі $(1+x^n)$ четвертий член дорівнює 0,96. Знайти значення *x* і*n*, якщо сума біноміальних коефіцієнтів дорівнює 1024.

Розв'язок. Оскільки сума біноміальних коефіцієнтів дорівнює 2^n, а $1024 = 2^{10}$, то $n = 10$. Четвертий член розкладу $T_4 = C_n^3 x^3 = \frac{10\cdot 9\cdot 8}{1\cdot 2\cdot 3}x^3 = 120x^3$. Згідно з умовою $120x^3 = 0{,}96$,
Звідки $x^3 = 0{,}008$, тобто $x = 0{,}2$.

Приклад 2.42. За яких значень *x* і *y* можлива рівність $C_y^x : C_{y+2}^x : A_y^x = 1:3:24$?

Розв'язок. Застосовуючи формули, маємо: $\frac{y!}{x!(y-x)!} : \frac{(y+2)!}{x!(y-x+2)!} = \frac{1}{3}; C_y^x : \left(x! \cdot C_y^x\right) = 1:24.$

Із другого рівняння отримаємо $x! = 24$, тобто $x = 4$, оскільки (24 = 1*2*3*4), а з першого рівняння знаходимо $\frac{(y-x+1)(y-x+2)}{(y+1)(y+2)} = \frac{1}{3}$.
Оскільки $x = 4$, то $y^2 - 9y + 8 = 0$, звідки $y = 1$ і $y = 8$, $y = 1$не задовольняє умові $(y > x = 4)$. Отже, $x = 4$; $y = 8$.

Приклад 2.43. Довести тотожність: $P_n = (n-1)(P_{n-1} + P_{n-2})$.

Розв'язок. Маємо $P_{n-1} = (n-1)!; P_{n-2} = (n-2)!$ Таким чином, $(n-1)\big((n-1)! + (n-2)!\big) = (n-1)(n-2)!\,(n-1+1) = n! = P_n$.

Приклад 2.44. Довести тотожність: $C_n^k C_{n-k}^{m-k} = C_m^k C_n^m$.

Розв'язок. Ліва частина шуканої тотожності $\frac{n!}{k!(n-k)!} \cdot \frac{(n-k)!}{(m-k)!(n-m)!} = \frac{n!}{k!(m-k)!(n-m)!}$,
а права частина $\frac{\boldsymbol{m}!}{\boldsymbol{k}!(\boldsymbol{m}-\boldsymbol{k})!} \cdot \frac{\boldsymbol{n}!}{\boldsymbol{m}!(\boldsymbol{n}-\boldsymbol{m})!} = \frac{\boldsymbol{n}!}{\boldsymbol{k}!(\boldsymbol{m}-\boldsymbol{k})!(\boldsymbol{n}-\boldsymbol{m})!}$.

Отже, тотожність доведено.

Приклад 2.45. За якого значення x четвертий доданок розкладу $\left(\sqrt{2^{x-1}}+\sqrt[3]{2^{-x}}\right)^m$ у 20 разів більший за m, якщо біноміальний коефіцієнт 4-го доданку належить до біноміального коефіцієнта 2-го доданку як 5:1?

Розв'язок. Біноміальні коефіцієнти 4-го і 2-го доданків дорівнюють відповідно C_m^3 і m. Отже, $\frac{m(m-1)(m-2)}{3!m}=5$, або $(m-1)(m-2)=30$. Звідки m = 7. Тоді 4-й доданок розкладу має вигляд $T_4 C_3^7 2^{2(x-1)x\frac{1}{2^x}}$, і приходимо до рівняння $C_7^3 2^{x-2}=140$, звідки знаходимо $\frac{7\cdot6\cdot5}{6}2^{x-2}=140$; $2^{x-2}=4$; x = 4.

Приклад 2.46. Вирішити рівняння $\frac{A_{x+1}^{y+1}P_{x-y}}{P_{x-1}}=72.$

Розв'язок. Маємо $A_{x+1}^{y+1}=(x+1)!\,(x-y)!$; $P_{x-y}=(x-y)!$; $P_{x-1}=(x-1)!$, звідки $\frac{(x+1)!(x-y)!}{(x-y)!(x-1)!}=72,$ або $x(x+1)=72,$ тобто x = 8. Враховуючи, що $x-y>0$ і y – ціле число, отримуємо y = 0, y = 1, …, y = 7.

Приклад 2.47. Вирішити систему рівнянь $\begin{cases} A_y^x : P_{x-1}+C_y^{y-x}=126 \\ P_{x+1}=720 \end{cases}.$

Розв'язок. Із другого рівняння маємо: $(x + 1)! = 720$. Оскільки 720 = 6!, то x =5 . Враховуючи, що $C_y^{y-x}=C_y^x$, перепишемо перше рівняння таким чином: $A_y^5 : P_4+C_y^5=126$. Але $A_y^5 : P_4=5C_y^5$, звідки $6C_y^5=126$ або $y(y-1)(y-2)(y-3)(y-4)=21\cdot120.$

Далі маємо $21\cdot120=2^3\cdot3^2\cdot5\cdot7=7\cdot6\cdot5\cdot4\cdot3$, тобто y = 7.

Приклад 2.48. Довести тотожність $C_n^k+C_n^{k-1}=C_{n+1}^k$.

Розв'язок. Перетворимо ліву частину рівності

$$\frac{n!}{k!\,(n-k)!}+\frac{n!}{(k-1)!\,(n-k+1)!}=\frac{n!}{(k-1)!\,(n-k)!}\left(\frac{1}{k}+\frac{1}{n-k+1}\right)$$
$$=\frac{(n+1)!}{k!\,(n-k+1)!}$$

Але $C_{n+1}^k=\frac{(n+1)!}{k!(n-k+1)!}$ і тотожність доведено.

Приклад 2.49. Знайти найбільший біноміальний коефіцієнт розкладу $\left(n+\frac{1}{n}\right)^n$, якщо добуток 4-го від початку і 4-го з кінця доданків дорівнює 14400.

Розв'язок. Четвертий доданок спочатку має вигляд $T_4 = C_n^3 n^{n-3} \frac{1}{n^3}$, а 4-й доданок від кінця – вид $T_{n-2} = C_n^{n-3} n^3 \frac{1}{n-3}$. Відповідно $T_4 T_{n-2} = (C_n^3)^2 = 14400$, звідки $C_n^3 = 120$.

Далі маємо $n(n-1)(n-2) = 720$; $n(n-1)(n-2) = 10 \cdot 9 \cdot 8$, $n = 10$. Отже, найбільший біноміальний коефіцієнт, що входить у доданок, однаково віддалений від кінців розкладу є $C_{10}^5 = 252$.

Приклад 2.50. Сума третього від початку і третього від кінця біноміальних членів розкладу $\sqrt[4]{3} + \sqrt[3]{4}$ дорівнює 9900. Скільки раціональних членів містить цей розклад?

Розв'язок. Указані в умові коефіцієнти дорівнюють C_n^2 . Маємо $\frac{2 \cdot n(n-1)}{2} = 9900$ або $n(n-1) = 100 \cdot 99$, звідки $n = 100$. Тоді $T_{k+1} = C_{100}^k 3^{(100-k)/4} 4^{k/3}$; згідно з умовою $k/3$ і $(100-k){:}\,4$ – цілі числа, тобто k ділиться на 12. Для $n = 100$ таких чисел є $\left[\frac{100}{12}\right] + 1 = 9$.

Рішення комбінаторних рівнянь

Приклад 2.51. $A_x^2 = 42$.

Розв'язок. ОДЗ: $x \in N; x \geq 2$ $\frac{x!}{(x-2)!} = 42$ $\frac{(x-2)!(x-1)x}{(x-2)!} = 42$.

$x^2 - x - 42 = 0$ $x_1 = -6$ (видалити – не входить в ОДЗ); $x_2 = 7$.

Відповідь: 7.

Приклад 2.52. $5C_x^3 = C_{x+2}^4$.

Розв'язок. ОДЗ: $\begin{cases} x \geq 3 \\ x+2 \geq 4 \end{cases} \begin{cases} x \geq 3 \\ x \geq 4-2 \end{cases} \begin{cases} x \geq 3 \\ x \geq 2 \end{cases} x \geq 3$.

$C_x^3 = \frac{x!}{3!(x-3)!}$ $C_{x+2}^4 = \frac{(x+2)!}{4!(x-2)!}$;

$$\frac{5(x-3)!\,(x-2)(x-1)x}{3!\,(x-3)!} = \frac{(x-2)!\,(x-1)x(x+1)(x+2)}{4!\,(x-2)!}$$

$$\frac{5(x-2)(x-1)x}{1 \cdot 2 \cdot 3} = \frac{(x-1)x(x+1)(x+2)}{1 \cdot 2 \cdot 3 \cdot 4}$$

$$20(x-2)(x-1)x = (x-1)x(x+1)(x+2)$$

$$20(x-2) = (x+1)(x+2)$$

$$20x - 40 = x^2 + 3x + 2$$

$x^2 - 17x + 42 = 0$; $x_1 = 0$; $x_2 = 1$ – корені 0 і 1 не входять до ОДЗ; $x_3 = 3$; $x_4 = 14$.

Відповідь: 3; 14.

Приклад 2.53. $\begin{cases} A_x^y : A_x^{y-1} = 10 \\ C_x^y : C_x^{y-1} = \frac{5}{3} \end{cases}$ ОДЗ: $\begin{cases} x \geq y \\ y \geq N \\ x \in N \end{cases}$

$$\begin{cases} \dfrac{x!}{(x-y)!} : \dfrac{x!}{(x-y+1)!} = 10 \\ \dfrac{x!}{(x-y)!\,y!} : \dfrac{x!}{(y-1)!\,(x-y+1)!} = \dfrac{5}{3} \end{cases}$$

$$\begin{cases} \dfrac{x!}{(x-y)!} \cdot \dfrac{(x-y+1)!}{x!} = 10 \\ \dfrac{x!}{(x-y)!\,y!} \cdot \dfrac{(y-1)!\,(x-y+1)!}{x!} = \dfrac{5}{3} \end{cases}$$

$$\begin{cases} \dfrac{(x-y)!\,(x-y+1)}{(x-y)!} = 10 \\ \dfrac{(y-1)!\,(x-y+1)!}{(x-y)!\,y!} = \dfrac{5}{3} \end{cases} \Leftrightarrow \begin{cases} x-y+1 = 10 \\ \dfrac{x-y+1}{y} = \dfrac{5}{3} \end{cases} \Leftrightarrow \begin{cases} x-y = 9 \\ 3x - 8y = -3 \end{cases}$$

$\begin{cases} -3x + 3y = -27 \\ 3x - 8y = -3 \end{cases}$ вирішуємо методом додавання $-5y = -30$; $y = 6$;

$x - 6 = 9$; $x = 15$.

Відповідь: (15; 6).

Приклад 2.54. $\begin{cases} C_x^{y+1} = 2{,}5x \\ C_{x-1}^y = 10 \end{cases}$ ОДЗ: $\begin{cases} x > y+1 \\ x-1 > y \\ x > 0, y > 0, x > 1 \end{cases}$

$$\begin{cases} \dfrac{x!}{(x-(y+1))!\,(y+1)!} = 2{,}5 \\ \dfrac{(x-1)!}{\big(x-(y+1)\big)!\,y!} = 10 \end{cases} \Leftrightarrow \begin{cases} \dfrac{x(x-1)!}{\big(x-(y+1)\big)!\,(y+1)y!} = 2{,}5x \\ \dfrac{(x-1)!}{\big(x-(y+1)\big)!\,y!} = 10 \end{cases}$$

$$\begin{cases} \dfrac{x}{y+1} = 0{,}25x \\ \dfrac{(x-1)!}{\big(x-(y+1)\big)!\,y!} = 10 \end{cases} \Leftrightarrow \begin{cases} y+1 = 4 \\ \dfrac{(x-1)!}{\big(x-(y+1)\big)!\,y!} = 10 \end{cases} \Leftrightarrow \begin{cases} y = 3 \\ \dfrac{(x-1)!}{(x-4)!\,3!} = 10 \end{cases}$$

$$\begin{cases} y = 3 \\ \dfrac{(x-4)!\,(x-3)(x-2)(x-1)}{1 \cdot 2 \cdot 3 \cdot (x-4)!} = 10 \end{cases}$$

$(x-3)\,(x-2)\,(x-1) = 60$

$(x-3)\,(x-2)\,(x-1) = 3 \cdot 4 \cdot 5$

$x - 3 = 3$; $x = 6$.

Відповідь: (6; 3)

Приклад 2.55. $\begin{cases} A_x^y : A_x^{y-1} = 8 \\ C_x^y : C_x^{y-1} = 1{,}6 \end{cases}$.

$$\begin{cases} \dfrac{x!}{(x-y)!} : \dfrac{x!}{\big(x-(y-1)\big)!} = 8 \\ \dfrac{x!}{(x-y)!\,y!} : \dfrac{x!}{(y-1)!\big(x-(y-1)\big)!} = 1{,}6 \end{cases}$$

$$\begin{cases} \dfrac{x!}{(x-y)!} \cdot \dfrac{\big(x-(y-1)\big)!}{x!} = 8 \\ \dfrac{x!}{(x-y)!\,y(y-1)!} \cdot \dfrac{(y-1)!\big(x-(y-1)\big)!}{x!} = 1{,}6 \end{cases}$$

$$\begin{cases} x-(y-1) = 8 \\ \dfrac{(x-y)!\big(x-(y-1)\big)}{(x-y)!\,y} = 1{,}6 \end{cases}$$

$$\begin{cases} x-y+1 = 8 \\ \dfrac{8}{y} = 1{,}6 \end{cases} \Leftrightarrow \begin{cases} x-y = 7 \\ y = 8:1{,}6 \end{cases} \Leftrightarrow \begin{cases} x = 7+5 \\ y = 5 \end{cases} \Leftrightarrow \begin{cases} x = 12 \\ y = 5 \end{cases}$$

Відповідь: (12; 5).

Ще декілька прикладів знаходження коефіцієнтів розкладу многочлена за формулою бінома Ньютона:

Приклад 2.56. Написати розклад за формулою бінома Ньютона і спростити $(a+b)^4$.

Розв'язок: $(a+b)^4 = C_4^0 a^4 b^0 + C_4^1 a^3 b^1 + C_4^2 a^2 b^2 + C_4^3 a^1 b^3 + C_4^4 a^0 b^4 = \frac{4!}{0!4!} a^4 + \frac{4!}{1!3!} a^3 b + \frac{4!}{2!2!} a^2 b^2 + \frac{4!}{3!1!} ab^3 +$

$$+\frac{4!}{4!\,0!} b^4 = a^4 + 4a^3 b + 6a^2 b^2 + 4ab^3 + b^4.$$

Приклад 2.57. Знайти алгебраїчну суму коефіцієнтів многочлена відносно *x*, який отриманий розкладом бінома Ньютона $(3x-4)^{17}$.

Розв'язок.

$$(3x-4)^{17} = C_{17}^0 (3x)^{17}(-4)^0 + C_{17}^1 (3x)^{16}(-4)^1 + C_{17}^2 (3x)^{15}(-4)^2 + C_{17}^3 (3x)^{14}(-4)^3 + \ldots + C_{17}^{17}(3x)^0(-4)^{17}.$$

Ця рівні стиснена за будь-яких значень x.

Якщо $x = 1$, ліва частина дорівнює $(3-4)^{17} = (-1)^{17} = -1$, а в правій частині отримуємо алгебраїчну суму коефіцієнтів:

$C_{17}^{0}3^{17}(1)^{17} + C_{17}^{1}3^{16}1^{16}(-4)^{1} + C_{17}^{2}3^{15}1^{15}(-4)^{2} + \ldots + C_{17}^{17}(-4)^{17} =$
$= C_{17}^{0}3^{17} - 4 \cdot 3^{16}C_{17}^{1} + 16 \cdot 3^{15}C_{17}^{2} + \ldots + (-4)^{17} \cdot C_{17}^{17}$

Отже, алгебраїчна сума коефіцієнтів цього многочлена дорівнює –1.

Приклад 2.58. Знайти 13-й член розкладу бінома $\left(\sqrt[3]{3} + \sqrt{2}\right)^{15}$.

Розв'язок. Згідно з формулою загального члена розкладу бінома,
$T_{13} = T_{12+1} = C_{15}^{12}\left(\sqrt[3]{3}\right)^{3}\left(\sqrt{2}\right)^{12} = C_{5}^{3} \cdot 3 \cdot 2^{6} = \frac{13 \cdot 14 \cdot 15}{1 \cdot 2 \cdot 3} \cdot 3 \cdot 2^{6} = 87360.$

Приклад 2.59. Знайти номер члена розкладу бінома $\left(\sqrt[3]{x} + \frac{1}{x}\right)^{16}$, що не містить x.

Розв'язок. Для загального члена розкладу маємо
$T_{m+1} = C_{16}^{m}\left(\sqrt[3]{x}\right)^{16-m}\left(\frac{1}{x}\right)^{m} = C_{16}^{m}x^{\frac{16-m}{3}}x^{-m} = C_{16}^{m}x^{\frac{16-4m}{3}}.$

Член розкладу не залежить від x; це означає, що показник степені x дорівнює 0, тільки тоді, коли $\frac{16-4m}{3} = 0,$ $16 - 4m = 0$, $m = 4$. Отже, п'ятий член даного розкладу не залежить від x.

Приклад 2.60. Побудувати трикутник Паскаля для знаходження коефіцієнтів розкладу бінома Ньютона $(a+b)^{7}$.

Розв'язок.

n	C_n^m
0	1
1	1 1
2	1 2 1
3	1 3 3 1
4	1 4 6 4 1
5	1 5 10 10 5 1
6	1 6 15 20 15 6 1
7	1 7 21 35 35 21 7 1
	C_7^0 C_7^1 C_7^2 C_7^3 C_7^4 C_7^5 C_7^6

Приклад 2.61. Знайти найбільший член розкладу $\left(\sqrt{5} + \sqrt{2}\right)^{20}$.

Розв'язок. За умовою маємо: $T_{m+1} > T_m$ і $T_{m+1} > T_{m+2}$, тобто

$$\begin{cases} C_{20}^{m}\cdot\left(\sqrt{5}\right)^{20-m}\cdot\left(\sqrt{2}\right)^{m} > C_{20}^{m-1}\cdot\left(\sqrt{5}\right)^{20-m+1}\cdot\left(\sqrt{2}\right)^{m-1} \\ C_{20}^{m}\cdot\left(\sqrt{5}\right)^{20-m}\cdot\left(\sqrt{2}\right)^{m} > C_{20}^{m+1}\cdot\left(\sqrt{5}\right)^{20-m-1}\cdot\left(\sqrt{2}\right)^{m+1} \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} \dfrac{20!}{m!\,(20-m)!}\cdot\dfrac{\left(\sqrt{5}\right)^{20}}{\left(\sqrt{5}\right)^{m}}\cdot\left(\sqrt{2}\right)^{m} > \dfrac{20!}{(m-1)!\,(20-m+1)!}\cdot\dfrac{\left(\sqrt{5}\right)^{21}}{\left(\sqrt{5}\right)^{m}}\cdot\dfrac{\left(\sqrt{2}\right)^{m}}{\sqrt{2}} \\ \dfrac{20!}{m!\,(20-m)!}\cdot\dfrac{\left(\sqrt{5}\right)^{20}}{\left(\sqrt{5}\right)^{m}}\cdot\left(\sqrt{2}\right)^{m} > \dfrac{20!}{(m+1)!\,(20-m-1)!}\cdot\dfrac{\left(\sqrt{5}\right)^{19}}{\left(\sqrt{5}\right)^{m}}\cdot\sqrt{2}\cdot\left(\sqrt{2}\right)^{m} \end{cases}$$

$\Leftrightarrow$

$\Leftrightarrow \dfrac{\mathbf{20\sqrt{2}-\sqrt{5}}}{\mathbf{\sqrt{5}+\sqrt{2}}} < m < \dfrac{\mathbf{21\sqrt{2}}}{\mathbf{\sqrt{5}+\sqrt{2}}}$, 7,1 < m < 8,1.

Відповідь: $m = 8$. Тоді $T_{m+1} = T_9 = C_{20}^{8}\cdot\left(\sqrt{5}\right)^{12}\cdot\left(\sqrt{2}\right)^{8} = \frac{20!}{8!2!}\cdot\left(\sqrt{5}\right)^{12}\cdot$ $\left(\sqrt{2}\right)^{8} = 314925\cdot 10^{5}$– 9-й член розкладу.

Приклад 2.62. За яких значень x найбільшим членом розкладу $(\mathbf{5+3x})^{\mathbf{10}}$є четвертий?

Розв'язок. За умовою маємо: $\begin{cases} T_4 > T_3 \\ T_4 > T_5 \end{cases} \Leftrightarrow$

$$\begin{cases} C_{10}^{3}\cdot 5^{7}\cdot(3x)^{3} > C_{10}^{2}\cdot 5^{8}\cdot\left(3x^{\square}\right)^{2} \\ C_{10}^{3}\cdot 5^{7}\cdot(3x)^{3} > C_{10}^{4}\cdot 5^{6}\cdot(3x)^{4} \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} \dfrac{10!}{3!\cdot 7!}\cdot 5^{7}\cdot(3x)^{3} > \dfrac{10!}{4!\cdot 6!}\cdot 5^{6}\cdot(3x)^{4} \\ \dfrac{10!}{3!\cdot 7!}\cdot 5^{7}\cdot(3x)^{3} > \dfrac{10!}{2!\cdot 8!}\cdot 5^{8}\cdot(3x)^{2} \end{cases} \Leftrightarrow \begin{cases} \dfrac{5}{7} > \dfrac{3}{4}x \\ x > \dfrac{5}{8} \end{cases} \Leftrightarrow \begin{cases} x < \dfrac{20}{21} \\ x > \dfrac{5}{8} \end{cases}.$$

Відповідь: $\frac{5}{8} < x < \frac{20}{21}$.

<u>**Обов'язкові завдання**</u>

1. Вирішити рівняння $\frac{P_{x+3}}{A_x^5 P_{x-5}} = 720.$

2. Вирішити систему рівнянь: $\begin{cases} A_x^y + 3C_x^y = 90 \\ A_x^y - 2C_x^y = 40 \end{cases}$.

3. Знайти x і y, якщо

а) $C_{x+1}^{y} : C_x^{y+1} : C_x^{y-1} = 6:5:2$;

б) $C_x^{y-1} : \left(C_{x-2}^{y} + C_{x-2}^{y-2} + 2C_{x-2}^{y-1}\right) : C_x^{y+1} = 3:5:5.$

4. Різниця між третім біноміальним членом розкладів $(\boldsymbol{a}+\boldsymbol{b})^{\boldsymbol{n}+\mathbf{1}}$ і $(\boldsymbol{a}+\boldsymbol{b})^{\boldsymbol{n}}$ дорівнює 225. Знайти кількість раціональних членів розкладу$\left(\sqrt[5]{\boldsymbol{x}}+\sqrt[9]{\boldsymbol{y}}\right)^{\boldsymbol{n}}$.

Розділ 3. Теорія графів

3.1. Основні положення теорії графів

Історія виникнення теорії графів

Історія теорії графів почалася із задачі про кенігсберзькі мости, придуманої і розв'язаної Ейлером у 1736 році. Ейлер поставив собі задачу, як обійти всі чотири частини суші, пройшовши по кожному з мостів рівно один раз, і повернутися у початкове місце. Мости, по яких ходив Ейлер, розташовувалися, як показано на рис. 3.1.

Для розв'язування цієї задачі Ейлер позначив ділянки суші точками, а мости, що їх з'єднують, лініями, одержавши таким чином перше представлення графа.

Один із варіантів графа задачі про кенігсберзькі мости має такий вигляд.

Рис. 3.1. Розташування кенігсберзьких мостів

Як видно з рис. 3.2, граф складається з точок і ліній. Точки називають вершинами. Лінії, що з'єднують вершини, називають ребрами.

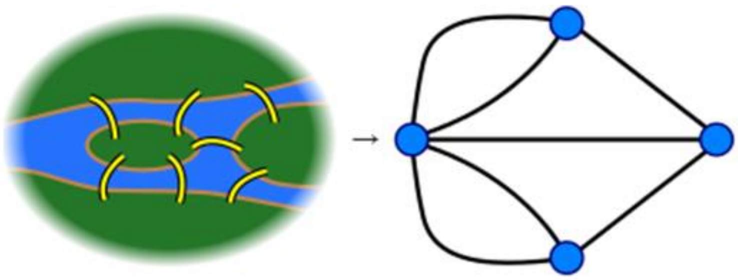


Рис. 3.2. Граф розташування кенігсберзьких мостів

Теорія графів у математиці займається вивченням особливого виду математичних структур – графів, що використовуються для моделювання парних відношень між об'єктами. Графи в цьому контексті складаються з вершин (точок), які з'єднані ребрами (лініями).

Враховуючи, що набір вершин можна використовувати для абстрагування будь-якого типу комп'ютерних даних, теорія графів глибоко вивчає взаємозв'язок між ними та дає відповіді на ряд питань розташування, налаштування мережі, оптимізації, узгодження та ін.

У такому ключі структури даних і їх використання вперше розглянув відомий математик Леонард Ейлер, який сформулював основні поняття теорії графів як розділу математики.

Теорія графів відіграє критичну роль у багатьох проблемах інформатики. Зокрема, теорія графів використовується для моделювання парних взаємозв'язків між об'єктами певної множини. Ряд комп'ютерних підходів було розроблено для полегшення використання графів, наприклад як SPANTREE або GTP для передачі даних.

Тобто графи – це метод візуальної ілюстрації даних і відношень між ними.

Мета графів – представити занадто численні або складні дані для їх адекватного опису в тексті або алгоритмі. Чіткість і коректність опису даних у графах забезпечують ефективність використання графів.

Основні визначення графів

Зображення графа містить точки та ребра. Однак точки та ребра не є графом, це лише елементи зображення графа, як, наприклад, показано на рис. 3.3.

Рис. 3.3. Загальний вигляд графа

Щоб перейти до визначення графа, розглянемо спочатку кілька прикладів.

Приклад 3.1. Розглянемо множину V студентів в аудиторії. Потужність множини |V| = 21. Якщо студенти сидять по двоє за партою і студентів позначимо точками, то студентів за однією партою можна позначити ребром. Маємо зображення, що показано на рис. 3.2. Із зображення зрозуміло, що парти стоять у два ряди, чотирнадцять студентів сидять по двоє за партою, п’ять студентів сидять по одному.

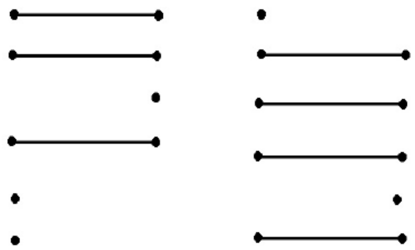


Рис. 3.4. Розміщення студентів за партами

На рис. 3.2. показано розміщення студентів за партами відповідно до прикладу 3.1. Із рис. 3.3 видно, що в аудиторії залишилося 7 студентів, один студент сидить сам за партою, а ще шість студентів сидять попарно, проте парти перекомбіновані, але це не означає, що вони розміщені одна на одній.

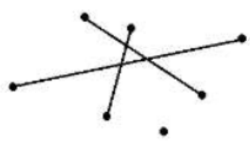


Рис. 3.3. Розміщення 7 студентів за партами

В обох випадках, описаних у прикладі 3.1 і показаних на рис. 3.2–3.3, маємо справу з графом. Тобто використання графа дає змогу підвищити інформативність передачі матеріалу і його засвоєння. Можна сказати, що граф – це не просто зображення на площині.

Граф – це абстрактний комбінаторний об'єкт, що складається з двох множин: V (vertex) – множини вершин і E (edge) – множини ребер. Саме тому в прикладі 3.1 кожне ребро має дві вершини – двох студентів, що сидять за однією партою.

Неорієнтований граф

Визначення. Неорієнтованим графом G (V, E) називають сукупність двох множин:

– непорожньої множини V, яку називають множиною **вершин**;

– множини E неупорядкованих пар елементів множини V, яку називають множиною **ребер.** Множина ребер у графі може бути порожньою.

Умови існування неорієнтованого графа G (V, E) задають такими правилами:

1. $V \neq \varnothing$. Множина вершин графа непорожня.

2. $E \subset V \times V$. Елементами множини ребер є відношення, елементами якого є двійки з вершин графа. Кількість двійок є підмножиною декартового добутку $V \times V$.

3. E = E^{-1}. Множина ребер неорієнтованого графа – це симетричне відношення.

Приклад неорієнтованого графа показано на рис. 3.4.

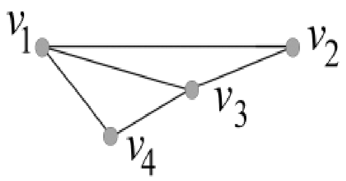


Рис. 3.4. Неорієнтований граф

Кількість вершин графа G позначимо через p, а кількість ребер графа G позначимо через q: $p = p$ (G) = V, $q = q$ (G) = E,

$p = p(G) = |V|$, $q = q(G) = |E|$, $V = \{v_1, v_2, v_3, v_4\}$, $E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3), (v_3, v_4)\}$,

$E^{-1} = \{(v_2, v_1), (v_3, v_1), (v_4, v_1), (v_3, v_2), (v_4, v_3)\}$.

Приклад 3.2. Розглянемо ту ж множину студентів в аудиторії V, що являють собою вершини графа. А як відношення між вершинами візьмемо збігання днів народження – це буде множина ребер E. Тобто V – множина студентів в аудиторії, E – множина людей з днем народження, що збігається.

Розглянемо приклад 3.3 зі складнішою конструкцією – зорієнтованими ребрами.

Приклад 3.3. Повернемося до множини студентів в аудиторії V, а як відношення, що характеризує множину ребер, розглядатимемо симпатію між ними.

У цьому разі симпатія є направленою. І може бути як односторонньою, так і взаємоспрямованою, як показано на рис. 3.5.

Як видно з верхнього графа на рис. 3.5, $E = (x, y)$, тобто x симпатизує y. Проте зворотне не правильне $(x, y) \neq (y, x)$. У цьому разі ребра зображають зі стрілкою і називають дугами, а сам граф називають орієнтований граф, або скорочено орграф. Із нижнього графа на рис. 3.5. можна зробити висновок, що $(a, b) = (b, a)$, тобто між a і b існує взаємна симпатія.

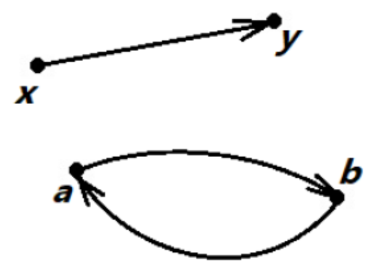


Рис. 3.5. Орієнтовані графи

Орієнтований граф

Визначення. Орієнтованим графом G (V, E) називають сукупність двох множин:

– непорожньої множини V, яку називають множиною **вузлів**;

– множини E упорядкованих пар елементів множини V, яку називають множиною **дуг**.

Умови існування орієнтованого графа G (V, E) задають такими правилами:

1. $V \neq \varnothing$. Множина вузлів графа непорожня.

2. $E \subset V \times V$. Елементами множини дуг є відношення, елементи якого – це двійки з вузлів графа. Кількість двійок є підмножиною декартового добутку $V \times V$.

3. $E = E^{-1}$. Множина дуг орієнтованого графа – це антисиметричне відношення.

Визначення. Якщо елементами множини є впорядковані пари, то граф називають **орієнтованим** (або **орграфом**). Дуги зображують лініями зі стрілками, що вказують напрямок. Приклад орієнтованого графа показано на рис. 3.6.

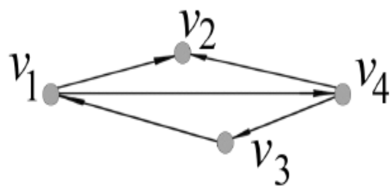


Рис. 3.6. Орієнтований граф

$V = \{v_1, v_2, v_3, v_4\}$, $E = \{(v_1, v_2), (v_1, v_4), (v_4, v_2), (v_4, v_3), (v_3, v_1)\}$.

Приклад 3.4. Дано орграф *G* (*V*, *E*), зображений на рис. 3.7:

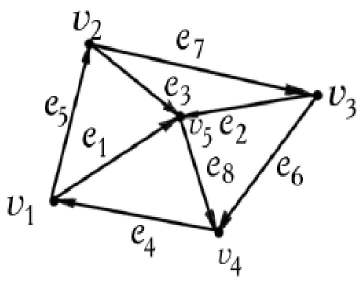


Рис. 3.7. Орграф *G* (*V*, *E*)

Визначити впорядковані пари множин $E = \{e_i | i = 1,2,3,\ldots,7\}$, що задає дуги даного графа.

Розв'язок. Запишемо відношення, яке відповідає множині впорядкованих пар вузлів *V*/

$E = \{(v_1, v_5), (v_3, v_5), (v_2, v_5), (v_4, v_1), (v_1, v_2), (v_3, v_4), (v_2, v_3)\}$, $e_1 = (v_1, v_5)$, $e_2 = (v_3, v_5)$,
$e_3 = (v_2, v_5)$.

Помічені графи. Граф із петлями

Визначення. Якщо задана функція *F*:*V*→*M* або *F*: *E*→*M*, то множину *M* називають множиною **міток**, а граф називають **поміченим графом**.

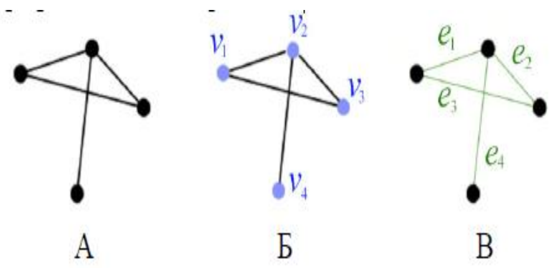


Рис. 3.8. Помічені (Б, В) і непомічені графи (А)

Такі графи показані на рис. 3.8: А – непомічений граф, Б – граф із поміченими вершинами $V = \{v_1, v_2, v_3, v_4\}$ (вершинно-помічений граф), В – граф із поміченими ребрами $E = \{e_1, e_2, e_3, e_4\}$ (реберно-помічений граф).

Якщо серед елементів множини E зустрічаються пари, які містять однакові вершини, то такий граф називають **графом із петлями.**

Приклад 3.5. На рис. 3.7 показано граф із петлями і орграф із петлями.

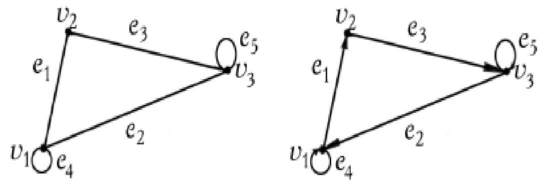


Рис. 3.9. Граф $G_1(V_1, E_1)$ та орграф $G_2(V_2, E_2)$ із петлями

Задати ці графи аналітично.

Розв'язок. $G_1(V_1, E_1)$; $V_1 = \{v_1, v_2, v_3\}$; $E_1 = \{(v_1, v_2), (v_2, v_1), (v_2, v_3), (v_3, v_2), (v_1, v_3), (v_3, v_1), (v_1, v_1), (v_3, v_3)\}$; $G_2(V_2, E_2)$; $V_2 = \{v_1, v_2, v_3\}$; $E_2 = \{(v_1, v_2), (v_2, v_3), (v_3, v_1), (v_1, v_1), (v_3, v_3)\}$.

Приклад 3.6. Повернемось до задачі про кенігсбергські мости. Ця задача, сформульована в XVIII ст. Леонардо Ейлером, поклала початок теорії графів. Суть задачі в така. Ейлер, живучи в Кенігсберзі, полюбляв прогулюватися вздовж річки і через мости. І от у нього виникало запитання, чи існує такий шлях, щоб вийти з дому, обійти всі мости й повернутися додому, але обійти мости так, щоб кожен міст пройти рівно один раз. Ейлер сам цю задачу сформулював і сам її розв'язав, тим самим поклавши початок використанню графів у математиці. До того ж у цьому разі суша – це вершини графа, мости – ребра. Тоді схема мостів у задачі Ейлера приводить нас до графа, що показаний на рис. 3.10. Як видно з рис. 3.10, з вершини до вершини є кілька ребер. Такі ребра називаються кратними, а сам граф носить назву мультиграфа.

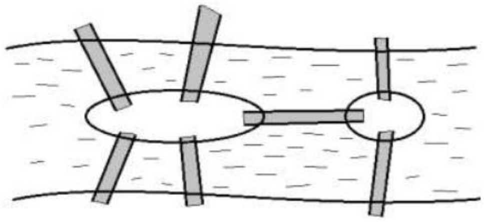


Рис. 3.10. Схема мостів у задачі Ейлера

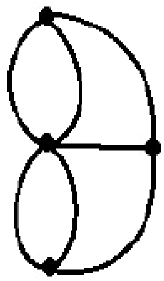


Рис. 3.11. Мультиграф до задачі Ейлера

Мультиграф

Визначення. Якщо множина *E* містить повторювані елементи, то відповідний граф *G* (*V*, *E*) включає кратні ребра. Тоді його називають **мультиграфом.**

Приклад 3.7. Мультиграф із петлями *G* (*V*, *E*) показаний на рис. 3.12.

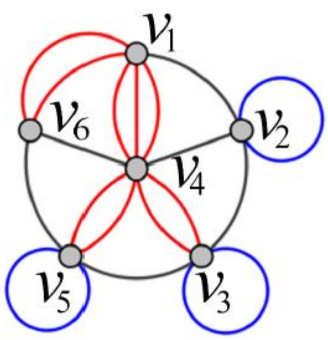


Рис. 3.12. Графічне представлення мультиграфа

Описати цей граф аналітично.

Розв'язок.

Множина вершин: $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$,

Множина ребер: $E =$
$$\begin{Bmatrix} (v_1, v_2), (v_1, v_6), (v_1, v_6), (v_1, v_4), (v_1, v_4)(v_1, v_4), (v_2, v_2), (v_2, v_3), (v_2, v_4), \\ (v_3, v_3), (v_3, v_5), (v_3, v_4), \\ (v_3, v_4), (v_4, v_5), (v_4, v_5), (v_4, v_6), (v_5, v_6), (v_5, v_5) \end{Bmatrix}$$

Загалом ми розглянули чотири види графів: неорієнтований, орієнтований, граф із петлями і мультиграф.

Приклад 3.8. Розглянемо структуру певного web-сайту де вершинами будуть сайти, а ребрами – гіперлінки. У цьому разі зображення графа може мати вигляд, наведений на рис. 3.13. Зліва на рис. показано гіперлінк сторінки на саму себе. Ребро такого типу називається петля. Як бачимо з рис. 3.13, петлі можуть бути кратні. Граф, що містить лише петлі (x, x), називається **псевдографом**.

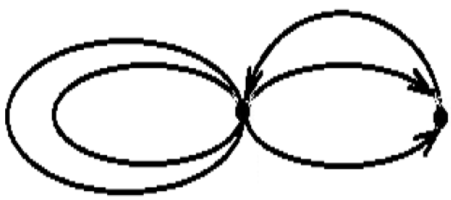


Рис. 3.13. Граф web-сайту

Загалом граф на рис. 3.13 є ор-, мульти-, псевдографом. Таким чином, підходимо до визначення. Граф G = <V, E> – це абстрактний комбінаторний об’єкт, що складається з двох множин. V – довільна множина вершин, або об’єктів. Множина може бути нескінченною. E – множина ребер, або пар об’єктів з V.

При цьому

1) якщо немає кратних ребер, то кожна пара об’єктів зустрічається в множині V не більше одного разу;

2) якщо це не орграф, то пари в E – невпорядковані;

3) якщо немає петель, то $(x, x) \notin$ E.

Якщо пункт 1 не виконується, маємо мультиграф. Якщо пункт 2 не виконується – орграф. Якщо пункт 3 не виконується, маємо псевдограф. Можлива також комбінація цих пунктів, наприклад псевдомультиграф і т. ін. Тобто графом можна оперувати, не використовуючи його зображення на площині.

Повний граф

Визначення. Якщо кожна пара вершин графа *G* (*V*, *E*) з'єднана ребром, то такий граф називають **повним**. Кількість ребер у повному графі дорівнює $m = \frac{n(n-1)}{2}$. Повний граф з *n* вершин позначають K_n.

Насиченість графа *D* визначається: $D = \frac{2m}{n(n-1)}$. Для повного графа *D* = 1.

Насичений граф – це граф, у якому кількість ребер наближається до максимально можливої: $|E| = O(|V^2|)$.

Розріджений граф – це граф, у якому кількість ребер наближається до кількості вершин: $|E| = O(|V|)$.

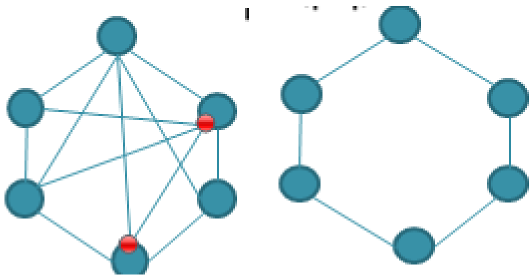


Рис. 3.14. Насичений і розріджений графи

Для насиченого графа $D = \frac{2 \cdot 12}{6 \cdot 5} = 0{,}8 > 0{,}5$.

Для розрідженого графа: $D = \frac{2 \cdot 6}{6 \cdot 5} = 0{,}4 < 0{,}5$

Приклад 3.9. На рис. 3.15 представлені повні графи:

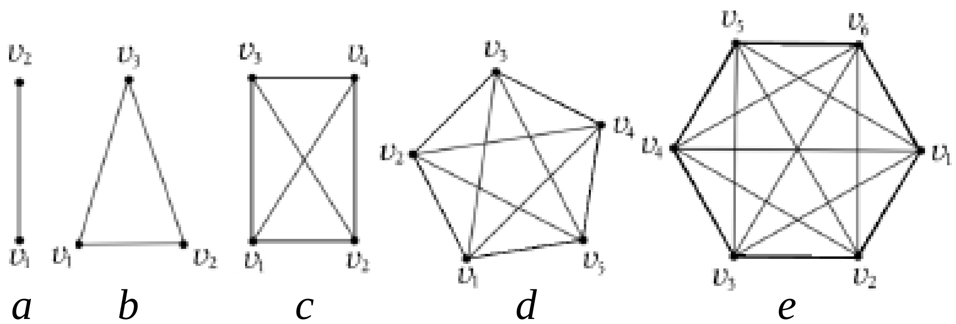


Рис. 3.15. Повні графи: *a* – K_2, *b* – K_3, *c* – K_4, *d* – K_5, *e* – K_6

Дводольний граф

Визначення. Граф G (V, E) називають **дводольним**, якщо множину його вершин V можна представити об'єднанням множин, які не перетинаються.

Нехай $V = A \cup B$, $A \neq \emptyset$, $B \neq \emptyset$, $A \cap B = \varnothing$, де $A = \{a_1, \dots, a_i, \dots, a_n\}$ і $B = \{b_1, \dots, b_i, \dots, b_m\}$. Тоді в дводольному графі існують тільки ребра(a_i, b_j) або (b_j, a_i), $1 \leq i \leq n$, $1 \leq j \leq m$.

Таким чином, кожне ребро зв'язує вершину, яка належить множині A, з вершиною, яка належить множині B, але жодні дві вершини з A або дві вершини з B не мають спільних ребер. Приклади графічного представлення дводольних графів показані на рис. 3.15.

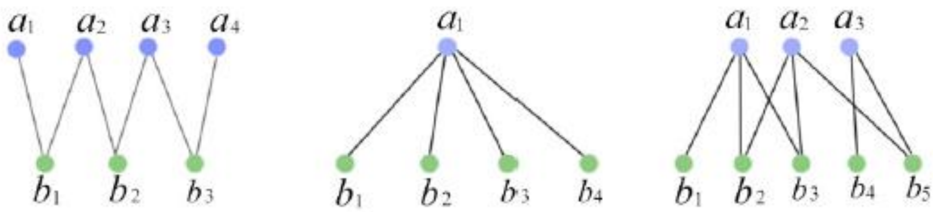


Рис. 3.16. Дводольні графи

Повний дводольний граф

Визначення. Дводольний граф називають **повним дводольним** графом $K_{m,n}$, якщо A містить m вершин, B містить n вершин і кожна вершина з множини A з'єднана ребром з кожною вершиною з множини B.

$K_{m,n} \to V = A \cup B$, $A \cap B = \emptyset$, для будь-яких $a \in A$, $b \in B$, якщо $(a, b) \in E$.

На рис. 3.17 представлені повні дводольні графи $K_{1,2}$, $K_{2,3}$, $K_{2,2}$, $K_{3,3}$.

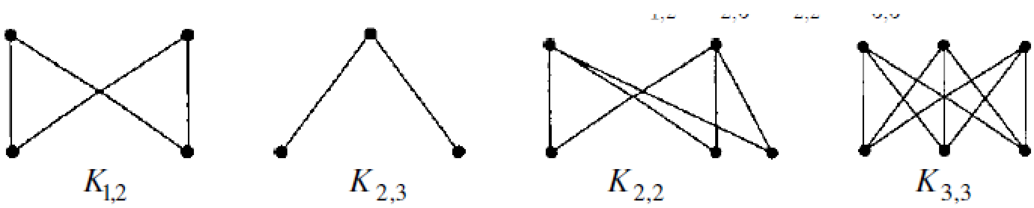


Рис. 3.17. Повні дводольні графи

Граф, який може бути зображено на площині (без перетину ребер), називається **планарним.**

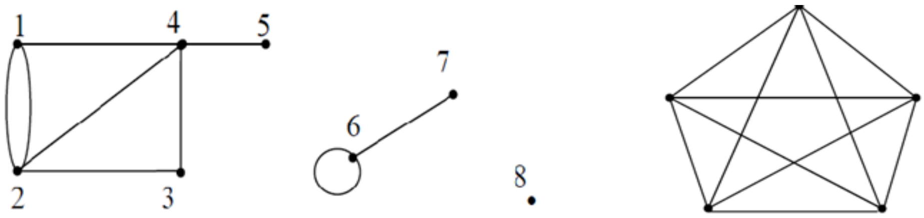


Рис. 3.18. Планарний і непланарний графи

Граф, вершини якого можна розбити на n непересічних підмножини так, що ніякі дві вершини, що належать одній підмножині, не суміжні, називається **n-дольним**.

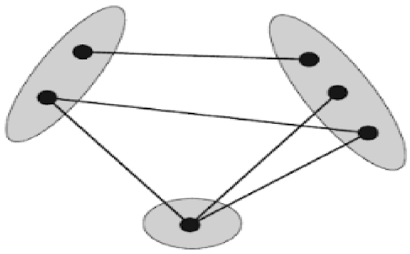


Рис. 3.19. Тридольний граф

Суміжність

Нехай $v_1 \in V$ і $v_2 \in V$ – вершини, $e = (v_1, v_2)$ – ребро, що з'єднує вершини v_1 та v_2, $e \in E$. Тоді вершина v_1 та ребро e інцидентні. Також вершина v_2 та ребро e інцидентні.

Два ребра, які інцидентні одній вершині, називають **суміжними ребрами** (рис. 3.20, а).

Дві вершини, які інцидентні одному ребру, називають **суміжними вершинами** (рис. 3.20, б).

Рис. 3.20. Суміжні ребра (а) та суміжні вершини (б)

Визначення. Множину вершин, суміжних із вершиною *v*, називають **множиною суміжності вершини,** або відображенням вершини *v*, і позначають *G*(*v*). На рис. 3.21 показано множину вершин $\{u_1, u_2, u_3, u_4, u_5\}$, суміжних із вершиною *v*.

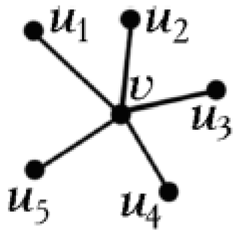


Рис. 3.21. Вершини $\{u_1, u_2, u_3, u_4, u_5\}$, суміжні з v

Вершини утворюють множину відображень $G(v)$ вершини v. Звідси $G(v) = \{u_i \in V | (u_i, v) \in E, 0 \leq i \leq p-1\}$, де $p = |V|$.

Приклад 3.10. Нехай дано граф G (V, E), показаний на рис. 3.22.

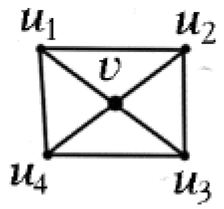


Рис. 3.22. Неорієнтований граф G (V, E)

Описати аналітично множини вершин і ребер цього графа та задати множину G(v) предикатом.

Розв'язок. Множина вершин: $V = \{u_1, u_2, u_3, u_4, u_5\}$, p = $|V| = 5$.

Множина ребер:
$E = \{(u_1, v), (u_2, v), (u_3, v), (u_4, v), (u_1, u_2), (u_1, u_4), (u_3, u_4), (u_3, u_2)\}$, $q = |E| = 8$..

Множина відображення вершини v: $G(v) = \{u_i \in V | (u_i, v) \in E, i = 1, \ldots, 4\}$.

Степінь вершини

Визначення. Степенем вершини v називають кількість ребер, інцидентних цій вершині. **Степінь** вершини позначають deg (v) або d (v), для будь-якого $v \in V$ $0 \leq deg(v) \leq p-1$, де $p = |V|$.

Зауваження. Наявність петлі не збільшує степінь вершини. **Степінь** вершини дорівнює потужності множини суміжності: $deg(v) = |G(v)|$.

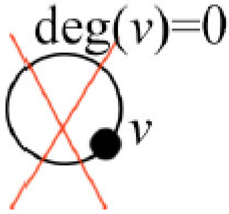


Рис. 3.23. Степінь вершини з петлею

Позначимо мінімальний степінь вершини графа G через $\delta(G)$, а максимальний – через $\Delta(G)$. Тоді $\delta(G) = \min\limits_{v \in V} \deg(v)$, $\Delta(G) = \max\limits_{v \in V} \deg(v)$.

Розглянемо характеристики неорієнтованого графа $G(V, E)$, показаного на рис. 3.24.

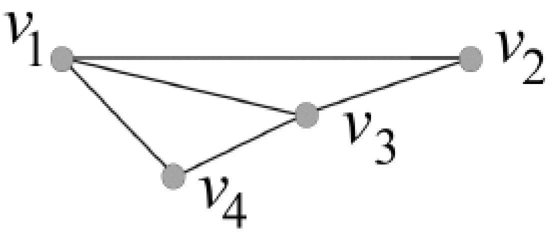


Рис. 3.24. Неорієнтований граф *G* (*V*, *E*)

$\boldsymbol{V = \{v_1, v_2, v_3, v_4\}, p = |V| = 4}$.
$\boldsymbol{E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3), (v_3, v_4)\}}$.
Вершина $\boldsymbol{v_1}$ характеризується відображенням $G(v_1)$ і потужністю відображення $|G(v)|$, яка дорівнює степеню цієї вершини $deg(v_1)$: $\boldsymbol{G(v_1) = \{v_2, v_3, v_4\}, |G(v_1)| = 3}$, $deg(v_1) = \mathbf{3}$.
Вершина v_2 : $G(v_2) = \{v_1, v_3\}$, $|\boldsymbol{G(v_2)}| = \mathbf{2}$, $deg(v_2) = \mathbf{2}$.
Вершина v_3: $G(v_3) = \{v_1, v_2, v_4\}$, $|\boldsymbol{G(v_3)}| = \mathbf{3}$, $deg(v_3) = \mathbf{3}$.
Вершина v_4: $G(v_4) = \{v_1, v_3\}$, $|\boldsymbol{G(v_4)}| = \mathbf{2}$, $deg(\boldsymbol{v_4}) = \mathbf{2}$.

Отже,
$\delta(G) = deg(v_2) = deg(v_4) = 2, \Delta(G) = deg(v_1) = deg(v_3) = 3$.

Регулярний граф

Визначення. Якщо степені всіх вершин дорівнюють *k*, то граф називають **регулярним графом** зі степенем *k*.

Приклад регулярного графа *G* (*V*, *E*), для якого $\delta(G) = \Delta(G) = 4$, показано на рис. 3.25.

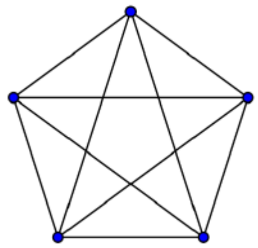


Рис. 3.25. Регулярний граф

Для регулярного k-графа справедливе співвідношення: $\delta(G) = \Delta(G) = k$.

Приклад 3.11. Визначити, чи є повним графом граф K_5.

Розв'язок. Повний граф K_5 є регулярним графом, оскільки $\delta(K_5) = 4$, $\Delta(K_5) = 4$, як показано на рис. 3.26.

Ізольована вершина

Визначення. Вершину v, для якої deg (v) = 0, називають **ізольованою.** На рис. 3.26 наведено приклад графа з ізольованою вершиною deg (4) = 0.

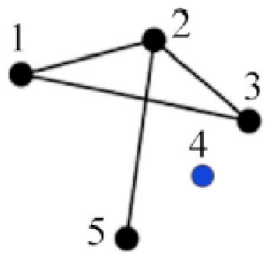


Рис. 3.26. Граф, який містить ізольовану вершину 4

Висяча вершина

Визначення. Вершину v, для якої deg (v) = 1, називають **кінцевою** або **висячою.** На рис. 3.27 показано граф, який містить **висячу вершину** deg (4) = 1.

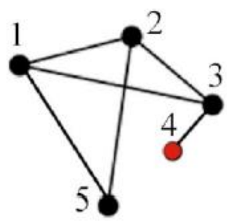


Рис. 3.27. Граф, який містить висячу вершину 4

Напівстепені в орграфі

Визначення. Напівстепенем виходу вузла або прямим відображенням вузла в орграфі називають кількість дуг, які виходять із цього вузла. Напівстепінь виходу для вузла *v* позначають як $deg^{+}(v) = |G^{+}(v)|$. 

Визначення. Напівстепенем входу вузла або зворотним відображенням вузла в орграфі називають кількість дуг, які входять у цей вузол. Напівстепінь входу для вузла *v* позначають як $deg^{-}(v) = |G^{-}(v)|$.

Розглянемо характеристики орієнтованого графа *G* (*V*, *E*), показаного на рис. 3.28.

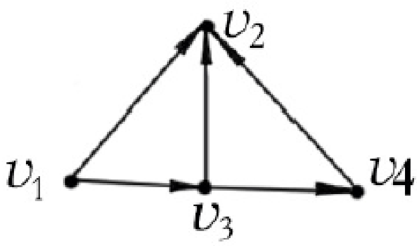


Рис. 3.28. Орієнтований граф *G* (*V*, *E)*

$V = \{v_1, v_2, v_3, v_4\},\ p = |V| = 4$,
$E = \{(v_1, v_2), (v_1, v_3), (v_3, v_2), (v_3, v_4), (v_4, v_2)\}$, $q = |E| = 5$,
$G^{+}(v_1) = \{v_2, v_3\}$, $deg^{+}(v_1) = |G^{+}(v_1)| = 2$,
$G^{-}(v_2) = \{v_1, v_3, v_4\}$, $deg^{-}(v_2) = |G^{-}(v_2)| = 3$,
$G^{+}(v_3) = \{v_2, v_4\}$, $deg^{+}(v_3) = |G^{+}(v_3)| = 2$,
$G^{-}(v_3) = \{v_1\}$, $deg^{-}(v_3) = |G^{-}(v_3)| = 1$,
$G^{+}(v_4) = \{v_2\}$, $deg^{+}(v_4) = |G^{+}(v_4)| = 1$,
$G^{-}(v_4) = \{v_3\}$, $deg^{-}(v_4) = |G^{+}(v_4)| = 1$.

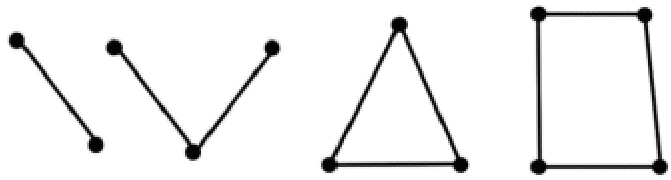
Теореми про степені вершин графа

Теорема 3.1. Сума степенів вершин графа завжди парна.

Доведення. Кожне ребро графа має два кінці. Тому кожне ребро збільшує степінь кожної з 2 інцидентних вершин на одиницю. Таким чином, кожне ребро збільшує суму степенів усіх вершин на 2. Отже, сума степенів усіх вершин завжди кратна 2, тобто парна.

Приклад 3.12. Зобразити 4 довільні графи й визначити суму їх степенів.

Розв'язок. Розглянемо 4 графи, показані на рис. 3.28, і визначимо суму їх степенів S_i, де i = 1, ..., 4.



$S_1 = 2,\ S_2 = 4,\ S_3 = 6,\ S_4 = 8.$

Рис. 3.29. Неорієнтовані графи для визначення суми степенів вершин

Теорема 3.2. У будь-якому графі кількість вершин непарного степеня парна.

Доведення. Доведення виконуватимемо методом від протилежного.

Припустимо, що теорема не вірна.

1. Якщо теорема не вірна, то існує непарна кількість вершин, степені яких непарні.

2. Якщо в графі немає вершин з парними степенями, як показано на рис. 3.30, то відразу виникає протиріччя з першою теоремою.

Рис. 3.30. Графи з вершинами, які мають тільки непарні степені

Протиріччя полягає в тому, що кількість вершин у цьому випадку повинна бути парною, оскільки сума степенів вершин графа завжди парна.

3. Якщо в графі є вершини з парними і непарними степенями, як показано на рис. 3.31, то очевидно, що сума степенів вершин з парними степенями парна.

Рис. 3.31. Графи з вершинами, які мають непарні й парні степені

4. Проте, оскільки сума всіх степенів графа парна, знову виникає протиріччя з початковим припущенням. Протиріччя полягає в такому: оскільки сума непарної кількості і парної кількості є число непарне, то в цьому випадку сума степенів усіх вершин повинна бути непарною. Але це суперечить теоремі, тому ми прийшли до протиріччя.

Отже, робимо висновок, що теорему доведено.

Теорема 3.3. Теорема Ейлера. Сума степенів вершин графа дорівнює подвоєній кількості ребер:

$\sum\limits_{v \in V} \deg(v) = 2$ – для неорієнтованого графа,

$\sum\limits_{v \in V} \deg^-(v) + \sum\limits_{v \in V} \deg^+(v) = 2q$ – для орграфа, де $q = |E|$ – потужність множини ребер.

Доведення. Під час підрахунку суми степенів вершин кожне ребро враховується двічі: для одного кінця ребра і для іншого. Звідси випливає, що кількість ребер у графі має дорівнювати половині суми степенів вершин. Теорему доведено.

Приклад 3.13. Перевірити задане теоремою Ейлера співвідношення вершин на графах, показаних на рис. 3.32.

Неорієнтований граф $G_1(V_1, E_1)$: $\sum\limits_{i=1}^{3} \deg(v_i) = 10$, $q = |E| = 5$,

Орієнтований граф $G_2(V_2, E_2)$ $\sum\limits_{i=1}^{5} \deg^-(v_i) = 8$, $\sum\limits_{i=1}^{5} \deg^+(v_i) = 8$, $q = |E| = 8$.

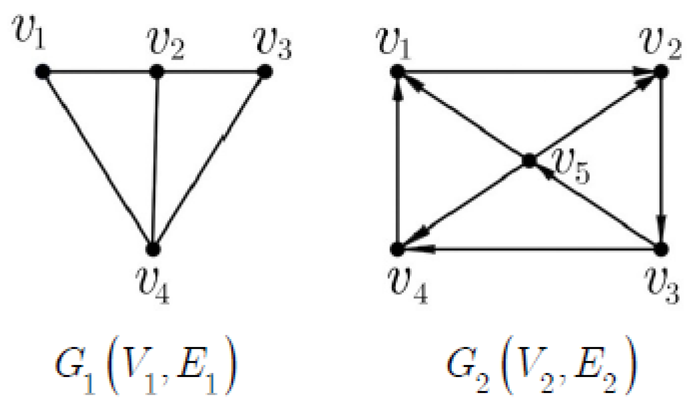


Рис. 3.32. Графи для визначення співвідношення вершин і ребер

Підграф графа

Визначення. Граф $G' = (V', E')$називають **під графом** графа G (V, E) $G'(V', E') < G(V, E)$, якщо $V' \subseteq V$і $E' \subseteq E$.Отже,

– кожна вершина в G' є одночасно вершиною в G,
– кожне ребро в G' є одночасно ребром в G.

Визначення. Граф $G'(V', E')$ називають **остовним підграфом** G або **суграфом** графа G, якщо $V' = V$ і $E' \subseteq E$. Отже,
– множини вершин графа G' та графа G збігаються,
– кожне ребро в G' є одночасно ребром в G.

Визначення. Граф $G'(V', E')$ називають **правильним підграфом** графа G, якщо $V' \subset V$ і G' містить усі можливі ребра G: для будь-яких $(u, v) \in G'(u, v)$ таких, що $(u, v) \in E \Rightarrow (u, v) \in E'$.

Правильний підграф утворюють шляхом виключення з графа певної кількості вершин та інцидентних до них ребер.

Приклад 3.14. На рисунку (а) показаний граф G (V, E). Визначити, якими є його підграфи (b), (c) і (d).

Розв'язок.

(**b**) На рисунку представлений **підграф** $\boldsymbol{G_1(V_1, E_1)}$ графа G (V, E), оскільки $V_1 \subset V$ і $E_1 \subset E$.

(**c**) Граф $\boldsymbol{G_2(V_2, E_2)}$є **остовним графом** або суграфом графа G (V, E), тому що $V_2 = V$ й $E_2 \subset E$.

(**d**) Граф $\boldsymbol{G_3(V_3, E_3)}$є **правильним підграфом** графа G (V, E), оскільки містить усі його можливі ребра.

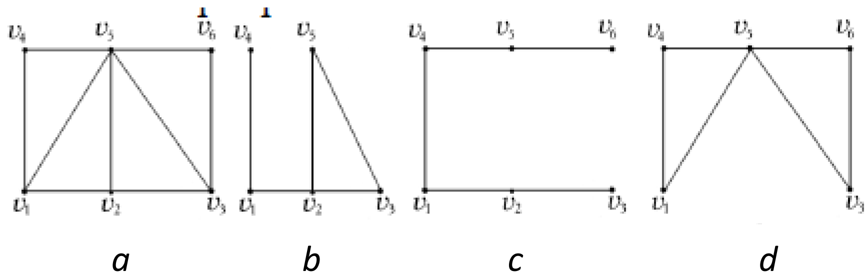


Рис. 3.33. Граф G (V, E) (a) та його підграфи (b), (c) і (d)

Структурні характеристики графів

Нехай $G = (V, E)$ – скінченний неорієнтований граф. Скінчена послідовність вершин і ребер графа $v_0 e_1 v_1 e_2 v_2 \ldots v_{k-1} e_k v_k$, у якій кожне ребро e_kє ребро, яке з'єднує вершини v_{k-1}та v_k, називається **маршрутом** на графі G.

Маршрут з'єднує вершини v_0та v_k.

Число k називають **довжиною маршруту**, тобто це кількість ребер, які входять до маршруту.

Маршрут називають **замкненим**, якщо $v_0 = v_k$.

Маршрут, у якому всі ребра є різні, називають **ланцюгом**.

Замкнений ланцюг називають **циклом.**

Цикл повністю визначають множиною його ребер. Тому часто під циклом ми будемо розуміти відповідну йому множину ребер.

Петля дає цикл довжини 1.

Пара кратних ребер утворює цикл довжини 2.

Цикли довжини 3 зазвичай називають *трикутниками.*

Ланцюг називають **простим**, якщо всі його вершини є різні.

Простий цикл – це цикл, у якому всі вершини, крім першої та останньої, є різними.

Граф без циклів називається **ациклічним**, в іншому разі граф називається **циклічним**.

Кожна вершина з'єднується сама з собою маршрутом довжини **0** і цей маршрут є простим циклом. Такий цикл називають **нульовим** (якщо сказано просто цикл, то мається на увазі, що він **не є нульовий**).

Лема. Якщо для деяких двох вершин u і v в графі існує (u, v)-маршрут, то існує і простий (u, v)-ланцюг.

Доведення. Розглянемо в графі (рис. 3.34) (u, v) – маршрут найменшої довжини.

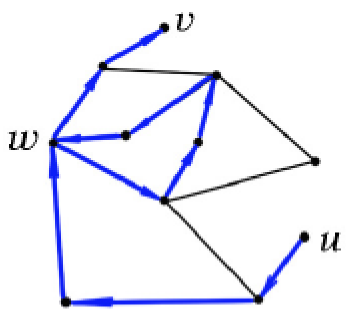


Рис. 3.34. Граф із маршрутом (u, v)

Покажемо, що цей маршрут є простим ланцюгом. Якщо в ньому є повторювана вершина w, то, заміняючи частину маршруту від першого входження вершини w до її другого входження на одну вершину w, ми одержимо більш короткий (u, v)-маршрут.

Приклад 3.15. Визначити структурні характеристики неорієнтованого графа.

Маршрут – $v_1 e_1 v_2 e_1 v_1 e_4 v_4 e_3 v_3 e_3 v_4 e_5 v_5$.

Ланцюг (всі ребра різні) – $v_1e_4v_4e_9v_2e_2v_3e_3v_4e_5v_5$.
Простий ланцюг (всі ребра і вершини різні) – $v_1e_4v_4e_9v_2e_2v_3e_7v_5$.
Цикл – $v_1e_1v_2e_9v_4e_3v_3e_7v_5e_5v_4e_4v_1$.
Простий цикл – $v_1e_4v_4e_9v_2e_1v_1$.

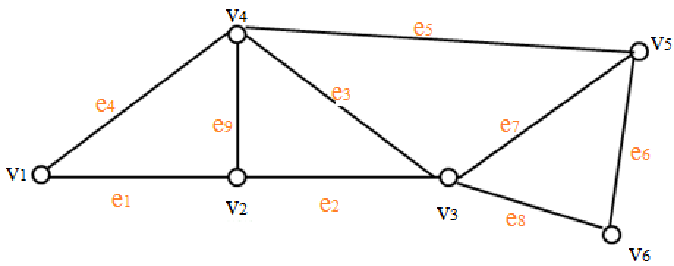


Рис. 3.35. Структурні характеристики неорієнтованого графа

Орієнтовані маршрути: в орграфі рух за маршрутом допускається лише в напрямках, зазначених стрілками (рис. 3.36).

Маршрут, який не містить повторних дуг, називається **шляхом**, а той, що не містить повторних вершин, – **простим шляхом**. Замкнений шлях називається **контуром**, а простий замкнений шлях – **простим контуром**.

Граф без циклів називається **безконтурним**, в іншому разі орграф називається **контурним**.

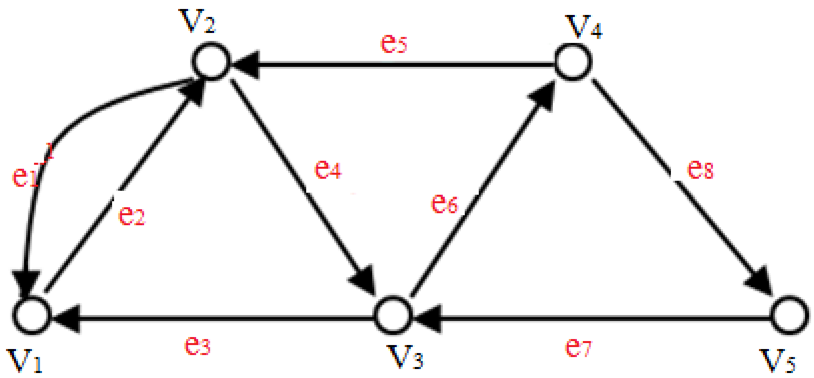


Рис. 3.36. Структурні характеристики орієнтованого графа

Маршрут – $v_1e_2v_2e_4v_3e_6v_4e_5v_2e_4v_3$.
Шлях (всі ребра різні) – $v_3e_6v_4e_8v_5e_7v_3e_3v_1$.
Простий шлях (всі ребра і вершини різні) – $v_5e_7v_3e_6v_4e_5v_2e_1v_1$.
Контур – $v_1e_2v_2e_4v_3e_6v_4e_5v_2e_1v_1$.
Простий контур – $v_1e_2v_2e_4v_3e_3v_1$.

3.2. Способи задавання та властивості графів

Представлення графа в пам'яті (формат збереження) визначає обчислювальну складність операцій над графом і об'єм необхідної пам'яті.

У дискретній математиці розглядають три способи задавання графів:

1. *Аналітичний спосіб*

Аналітичний спосіб припускає представлення графа G (V, E) у вигляді множин V і E. Для задавання цих множин можуть використовуватися всі три способи задавання множин. На рис. 3.37 показано граф G для математичного представлення.

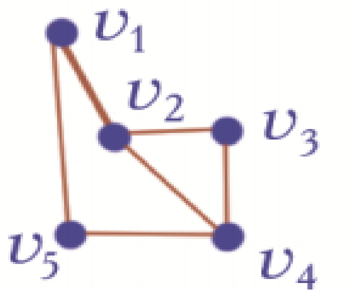


Рис. 3.37. Граф G

Явний спосіб у вигляді кортежу $G(V,E)$, де $V = \{v_1, v_2, v_3, v_4, v_5\}$ – множина вершин, $E = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_4), (v_4, v_5), (v_5, v_1)\}$ – множина ребер.

Задавання предикатом: $V = \{v_i | i = 1, \ldots, n\}$, $E = \{(v_i, v_j) | i = 2k + 1, j = 2k, k = 1 <\ldots, 2n - 1\}$.

Рекурсивною процедурою: $V = \{v_i | i = i + 1, i < m\}$,
$E = \{(v_i, v_j) | j = j + 1, i = i + 2, i, j < n\}$.

2. *Графічний спосіб*

Вершини представлені точками, а ребра – лініями, які з'єднують ці точки.

На рис. 3.38 показані способи графічного представлення різних графів.

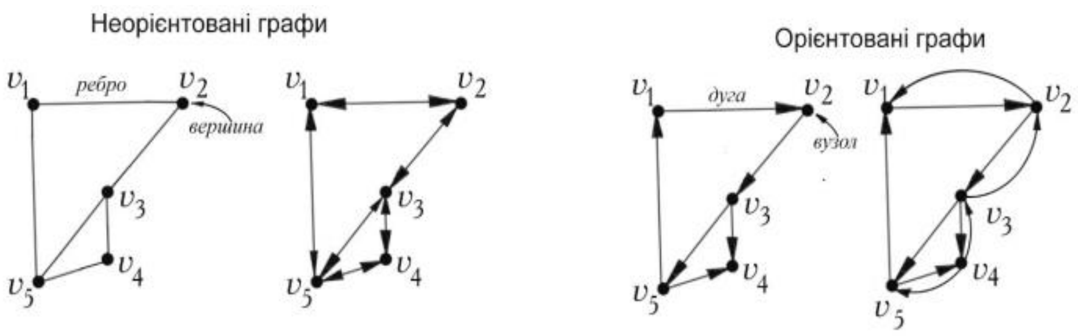


Рис. 3.38. Способи графічного представлення
орієнтованих і неорієнтованих графів

3. *Матричний спосіб*

Граф задають у вигляді матриці інцидентності або матриці суміжності.

Задавання неорієнтованого графа за допомогою матриці інцидентності

Нехай G – неорієнтований граф. Нехай B – матриця, кожний рядок якої відповідає вершині графа, а кожний стовпець відповідає ребру графа.

$$B = \begin{array}{cc} & \begin{array}{cccccc} e_1 & e_2 & \dots & e_j & \dots & e_m \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ \dots \\ v_i \\ \dots \\ v_n \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b_{ij} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

$$G\ (V, E),\ V = \{v_1, v_2, \dots, v_i, \dots, v_n\},\ E = \{e_1, e_2, \dots, e_j, \dots, e_m\}.$$

Елемент i-го рядка та j-го стовпця матриці B позначають b_{ij}. b_{ij}=1, якщо i-та вершина інцидентна j-му ребру й дорівнює 0 у протилежному випадку.

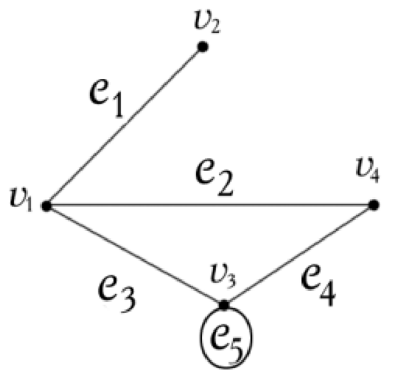


Рис. 3.39. Графічне представлення графа, заданого матрицею В

Матрицю В називають **матрицею інцидентності** неорієнтованого графа *G*.

Отже, елементи матриці інцидентності $B = (b_{ij})$ задаються формулою:

$$b_{ij} = \begin{cases} 1, \textit{якщо ребро } e_i \textit{ інцидентне вершині } v_j \\ 0 \textit{ у протилежному випадку}. \end{cases}$$

Приклад 3.16. Граф *G*=(*V*, *E*) задано аналітично множинами $V = \{v_1, v_2, v_3, v_4\}$, $E = \{e_1, e_2, e_3, e_4, e_5\} = \{(v_1, v_2), (v_1, v_4), (v_1, v_3), (v_3, v_4), (v_3, v_3)\}$.

Сформувати матрицю інцидентності та зобразити цей граф графічно.

Розв'язок.

Матриця інцидентності графа *G*= (*V*, *E*) має вигляд:

	e_1	e_2	e_3	e_4	e_5
v_1	1	1	1	0	0
v_2	1	0	0	0	0
v_3	0	0	1	1	1
v_4	0	1	0	1	0

або у більш формалізованому вигляді:

$$B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Графічне представлення графа, заданого матрицею В, відповідає зображенню графа на рис. 3.39.

Властивості матриці інцидентності неорієнтованого графа

1. Для вершин без петель степінь вершини дорівнює сумі одиничних елементів відповідного рядка матриці, оскільки кожна одиниця в цьому рядку представляє інцидентність цієї вершини ребру.

2. У кожному стовпці, який не представляє ребро петлі, будуть дві одиниці, тому що кожне таке ребро інцидентне двом вершинам.

3. У рядку матриці інцидентності, який відповідає вершині з петлею, сума одиниць на одну більше степеня цієї вершини.

4. Стовпець, що відповідає ребру петлі, містить тільки одну одиницю.

Задавання орієнтованого графа за допомогою матриці інцидентності

Нехай G – орієнтований граф. Тоді матриця інцидентності $B = (b_{ij})$ включає елементи, які дорівнюють 1, якщо вершина інцидентна з початком ребра, дорівнюють –1, якщо вершина інцидентна з кінцем ребра, дорівнюють 0, якщо вершина й ребро не інцидентні, дорівнюють 2 або будь-якому числу, якщо вершина містить петлю.

$$b_{ij} = \begin{cases} 1, \text{ якщо вершина } v_j \text{ є початком ребра } e_i, \\ -1, \text{ якщо вершина } v_j \text{ є кінцем ребра } e_i, \\ 2, \text{ якщо вершина } v_j \text{ є початком і кінцем ребра } e_i, \\ 0, \text{ в інших випадках.} \end{cases}$$

Приклад 3.17. Нехай задано орієнтований граф $G = (V, E)$, де $V = \{v_1, v_2, v_3, v_4\}$, $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Описати цей граф матрицею та зобразити графічно.

Розв'язок. Графічне представлення цього орграфа показано на рис. 3.39:

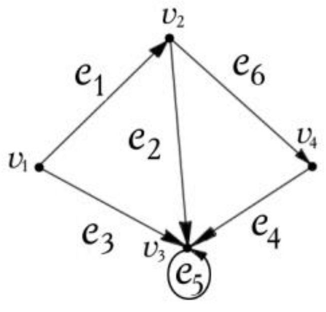


Рис. 3.40. Орграф $G = (V, E)$

Матриця інцидентності орграфа має вигляд:

$$\begin{array}{c|cccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \hline v_1 & 1 & 0 & 1 & 0 & 0 & 0 \\ v_2 & -1 & 1 & 0 & 0 & 0 & 1 \\ v_3 & 0 & -1 & -1 & -1 & 2 & 0 \\ v_4 & 0 & 0 & 0 & 1 & 0 & -1 \end{array}$$

або

$$B = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

Властивості матриці інцидентності орграфа

1. Для вершин без петель напівстепінь виходу дорівнює сумі додатних одиничних елементів відповідного рядка.

2. Для вершин без петель напівстепінь входу дорівнює сумі від'ємних одиничних елементів відповідного рядка.

3. У кожному стовпці, який не представляє ребро петлі, сума елементів дорівнює нулю.

4. Якщо дуга – це петля, то в стовпці один елемент, який дорівнює 2.

Задавання неорієнтованого графа за допомогою матриці суміжності

Нехай *G* – неорієнтований граф. Нехай C – матриця, рядки якої позначені вершинами графа і стовпці позначені тими ж вершинами в тому ж самому порядку.

Елемент *i*-го рядка та *j*-го стовпця матриці C позначається c_{ij}, і:

– дорівнює 1, якщо існує одне ребро з *i*-ї вершини в *j*-у вершину;

– дорівнює числу ребер з *i*-ї вершини в *j*-у вершину за наявності декількох ребер;

– дорівнює 0, якщо ребер між вершинами не існує.

$$C = \begin{array}{c} \\ v_1 \\ v_2 \\ v_i \\ v_j \\ v_n \end{array} \begin{pmatrix} v_1 & v_2 & v_i & v_j & v_n \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & c_{ij} & 1 \\ 0 & 1 & c_{ji} & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Матрицю С називають **матрицею суміжності** графа *G*.

Формула для визначення елементів матриці суміжності графа:

$$c_{ij} = \begin{cases} 1, \text{ якщо існує ребро } (v_i, v_j), \\ k, \text{ якщо існують ребра } \left\{ \overbrace{(v_i, v_j), (v_i, v_j), \ldots, (v_i, v_j)}^{k} \right\} \\ 0, \text{ в інших випадках.} \end{cases}$$

Приклад 3.18. Розглянемо неорієнтований граф *G* (*V*, *E*), показаний на рис. 3.58.

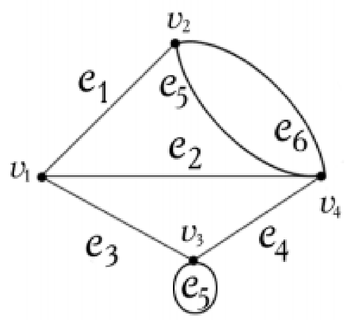


Рис. 3.41. Неорієнтований граф *G* (*V*, *E*)

Сформувати матрицю суміжності для цього графа.

Розв'язок.

Матриця суміжності цього графа має вигляд:

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 1 & 1 & 1 \\ v_2 & 1 & 0 & 0 & 2 \\ v_3 & 1 & 0 & 1 & 1 \\ v_4 & 1 & 2 & 1 & 0 \end{array} \text{ або } C = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 2 \\ 1 & 0 & 1 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}$$

Властивості матриці суміжності

1. Матриця суміжності неорієнтованого графа симетрична відносно головної діагоналі.

2. Якщо вершина має петлі, то їх число розміщається на головній діагоналі матриці суміжності.

3. Якщо між двома вершинами графа існує кілька ребер, то на перетині рядків і стовпців проставляють їх кількість.

Матриця суміжності орієнтованого графа

Нехай *G* – орієнтований граф. Нехай C – матриця, рядки якої позначені вершинами графа і стовпці позначені тими ж вершинами в тому ж самому порядку.

Елемент *i*-го рядка й *j*-го стовпця матриці C, позначається c_{ij}, і:

– дорівнює 1, якщо ребро виходить із вершиниv_i, представленої *i*-м рядком і входить у вершину v_j, представлену *j*-м стовпцем матриці;

– дорівнює числу ребер з *i*-ї вершини в *j*-у вершину за наявності декількох ребер;

– дорівнює 0, якщо ребер між вершинами не існує.

Матрицю C називають матрицею суміжності орграфа *G*.

Приклад 3.19. Розглянемо орієнтований граф *G* (*V*, *E*), що показаний на рис. 3.42:

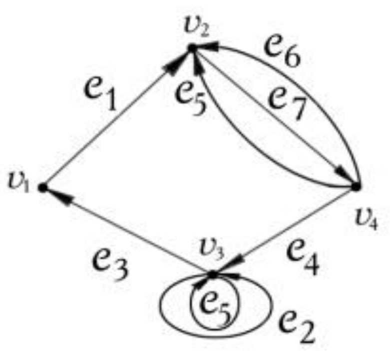


Рис. 3.42. Орієнтований граф *G* (*V*, *E*)

Сформувати матрицю суміжності для цього графа.

Розв'язок.

Його матриця суміжності має вигляд:

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 1 & 0 & 0 \\ v_2 & 0 & 0 & 0 & 1 \\ v_3 & 1 & 0 & 2 & 0 \\ v_4 & 0 & 2 & 1 & 0 \end{array} \text{ або } C = \begin{Vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 2 & 1 & 0 \end{Vmatrix}.$$

Властивості матриці суміжності орієнтованого графа

1. Матриця суміжності несиметрична відносно головної діагоналі.

2. Сума чисел у рядку матриці суміжності без урахування чисел на головній діагоналі дає змогу визначити потужність напівстепеня виходу для кожної вершини орграфа: $deg^{+}(v_i)$, де $1 \leq i \leq n$.

3. Сума чисел у стовпці матриці суміжності без урахування чисел на головній діагоналі дає змогу визначити потужність напівстепеня входу для кожної вершини орграфа: $deg^{-}(v_i)$, де $1 \leq i \leq n$.

Задавання графа за допомогою списку ребер

Список ребер графа (орграфа) представлено двома стовпцями. Перший стовпець містить ребра, а другий – інцидентні з ними вершини. Для неорієнтованого графа порядок проходження вершин довільний. Для орграфа першим стоїть номер вершини, з якої ребро виходить. Список ребер графа (орграфа) може також бути представлений послідовністю елементів, кожний з яких містить ребро та інцидентну йому пару вершин.

Приклад 3.20. Розглянемо граф *G* (*V*, *E*), показний на рис. 3.43.

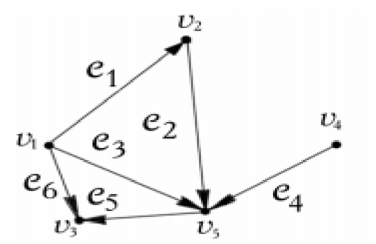


Рис. 3.43. Граф *G* (*V*, *E*)

Описати цей граф списком його ребер.

Розв’язок.

Орграф і список його ребер

$$e_1 \to (v_1, v_2), e_2 \to (v_2, v_3),\ e_3 \to (v_1, v_5), e_4 \to (v_4, v_5), e_5 \to (v_5, v_3), e_6 \to (v_1, v_3)$$

Ізоморфізм графів

З попереднього ми дізналися про різні способи задавання графів. Отже, довільний граф можна задати графічно (рисунком), матрицею інцидентності, матрицею суміжності і списком ребер.

У разі графічного задавання графа вигляд рисунка залежить від форми ліній і взаємного розташування вершин. Тому не завжди легко зрозуміти, чи однакові графи, зображені на різних рисунках. Наприклад, на рисунках 3.44, а та 3.44, б зображено той самий граф.

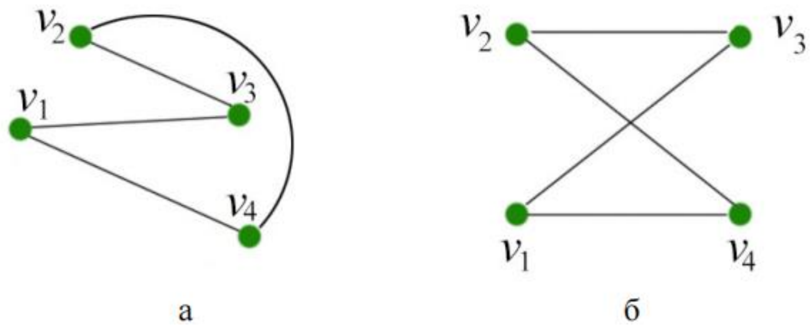


Рис. 3.44. Різні графічні представлення одного графа

У разі задавання графа матрицями або списком ребер їх вигляд залежить від нумерації вершин і ребер графа. Вважатимемо, що граф повністю заданий, якщо нумерація його вершин зафіксована.

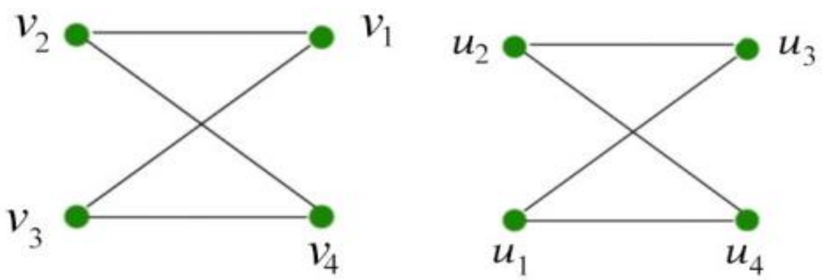
Визначення. Графи, що відрізняються тільки нумерацією вершин, називають ізоморфними графами.

Нехай $G = (V_1, E_1)$ і $H = (V_2, E_2)$– графи. Між вершинами цих графів існує взаємно однозначна відповідність (бієкція) $R: V_1 \to V_2$. Потужності множин V_1та V_2 збігаються: $|V_1| = |V_2|$.

Визначення. Відображення R називають ізоморфізмом графів G і H, якщо для будь-яких суміжних вершин $v_i, v_j \in G$ їх образи $(u_k, u_m) \in H$ також суміжні.

Визначення. Якщо відображення R існує на графах G і H, які відповідають попереднім умовам, то такі графи називають ізоморфними графами.

Приклад 3.21. На рис. 3.45 показані графи G (V, E) (а) та H (U, F) (б).



а *б*

Рис. 3.45. Графи G (V, E) (*а*) та H (U, F) (*б*).

Визначити, чи ці графи є ізоморфними.

Розв'язок.

Порівняємо множини відображень для вершини v_1у графі G (V, E) та u_1у графі H (U, F).

$$\begin{cases} \Gamma(v_1) = \{(v_1, v_2), (v_1, v_3)\} \\ \Gamma(u_3) = \{(u_3, u_2), (u_3, u_1)\} \end{cases}$$

Із цих відображень можна зробити висновок, що $R: v_1 \to u_3$.

Отже, відношення, яке задане перерахуванням, має вигляд:

$R = \{(v_1, u_3), (v_2, u_2), (v_3, u_1), (v_4, u_4)\}$.

Приклад 3.22. На рис. 3.46 показані графи $G(V_1, E_1)$ та $H(V_2, E_2)$.

Визначити, чи є ці графи ізоморфними.

Розв'язок.

1. Визначаємо потужність вершин графів: $|V_1| = 8$, $|V_2| = 8$, $|V_1| = |V_2|$

2. Знаходимо однозначне відображення вершин.

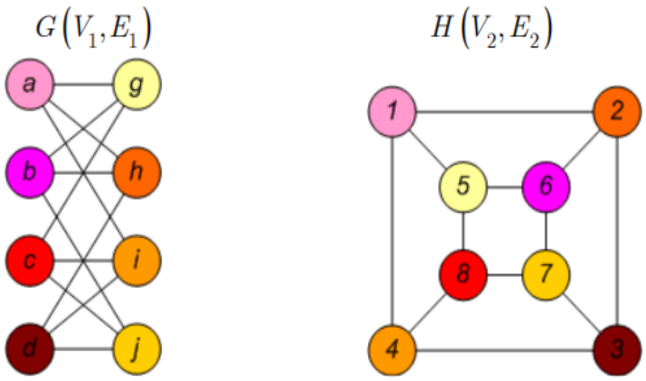


Рис. 3.46. Графи $G(V_1, E_1)$ та $H(V_2, E_2)$

$f(a)=1$		
$f(b)=6$	$(c,g)\to(8,5)$	$(a,g)\to(1,5)$
$f(c)=8$	$(c,i)\to(8,4)$	$(a,h)\to(1,2)$
$f(d)=3$	$(c,j)\to(8,7)$	$(a,i)\to(1,4)$
$f(g)=5$	$(d,h)\to(3,2)$	$(b,g)\to(6,5)$
$f(h)=2$	$(d,i)\to(3,4)$	$(b,h)\to(6,2)$
$f(i)=4$	$(d,j)\to(3,7)$	$(b,j)\to(6,7)$
$f(j)=7$		

3. Одержуємо відношення:
$R = \{(a,1),(b,6),(c,8),(d,3),(h,2),(g,5),(i,4),(j,7)\}$.

Приклад 3.23. На рис. 3.47 показані ізоморфні графи *G* й *H*.

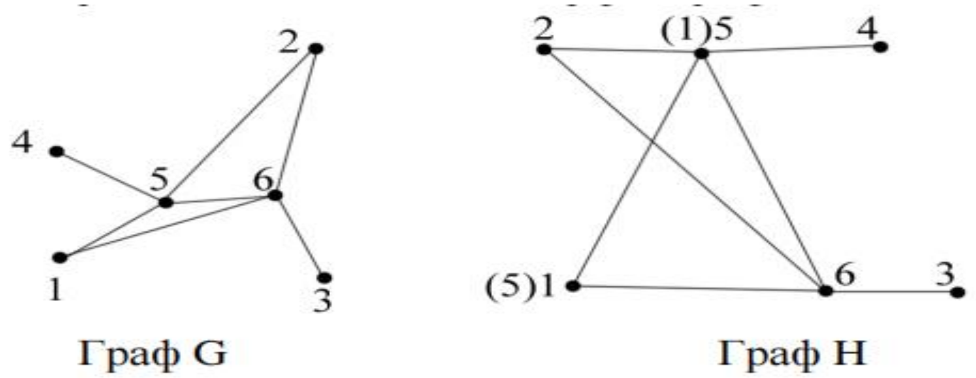


Рис. 3.47. Ізоморфні графи *G* і *H*

Побудувати матриці суміжності даних графів.

Розв'язок.

$G_{сум}$ – матриця суміжності графа *G* й $H_{сум}$ – матриця суміжності графа *H*.

$$G_{\text{сум}} = \begin{Vmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{Vmatrix} \quad H_{\text{сум}} = \begin{Vmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{Vmatrix}$$

Властивість матриць суміжності ізоморфних графів

Якщо графи G й H ізоморфні, то з матриці суміжності $G_{сум}$ графа можна одержати матрицю суміжності $H_{сум}$ шляхом послідовної перестановки рядків і стовпців. Для цього потрібно виконати максимально $n!$ перестановок, де n – число вершин графа.

Ізоморфізм орграфів

Визначення. Орграфи, що відрізняються нумерацією вершин і мають із точністю до нумерації однаково направлені ребра, називають **ізоморфними орграфами**.

Алгоритм розпізнавання ізоморфізму графів

1. Перевіряємо умову $|V| = |W| = n.$ Якщо кількість вершин графа $|V|$ не дорівнює кількості вершин графа $|W|$, то графи однозначно неізоморфні.

2. Сортуємо елементи множин $V = \{v_1, v_2, v_3, \ldots, v_n\}$ і $W = \{w_1, w_2, w_3, \ldots, w_n\}$за критерієм величини степенядля кожної вершини.

3. В отриманих упорядкованих послідовностях вершин шукаємо вершини з однаковими значеннями критерія упорядкування, тобто шукані вершини повинні мати однакові степені.

4. Якщо такі вершини знайдені, то з'єднуємо їх ребром із метою побудови графа взаємно однозначної відповідності. Якщо такої відповідності немає, тобто одна зі знайдених вершин уже включена в граф відповідності, то початкові графи G й H неізоморфні.

5. Якщо граф взаємно однозначної відповідності побудований, то розглянуті графи ізоморфні, а його ребра вказують на перестановки, які потрібно зробити для ізоморфного перетворення графа G в граф H.

На рис. 3.48 відповідні вершини ізоморфних графів з'єднані лініями.

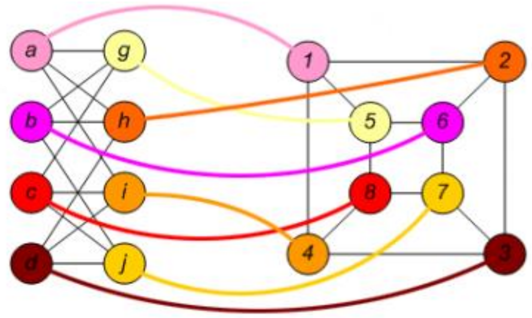
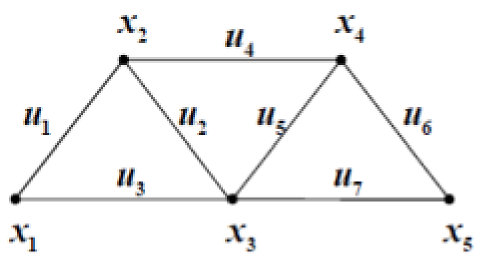


Рис. 3.48. Ізоморфні графи

<u>Обов'язкові завдання</u>

1. У графі, заданому на рисунку, визначити маршрут, ланцюг і простий ланцюг.



2. Вважаючи даний граф неорієнтованим, позначити його вершини і ребра різними символами і визначити:

1.1) локальні степені кожної вершини;

1.2) побудувати матриці суміжності й інцидентності;

1.3) показати підграф, що складається з трьох вершин;

1.4) навести приклади циклічного маршруту, ланцюга, простого ланцюга.

Вважаючи граф орієнтованим, визначити:

2.1) степені вершин;

2.2) матриці інцидентності й суміжності;

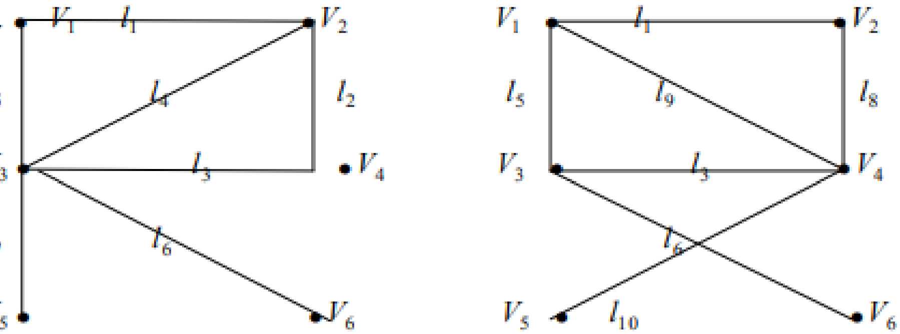
2.3) навести приклади шляху, орієнтованого ланцюга, простого ланцюга, контуру, циклу і простого циклу.

3. Побудуйте граф відношення $x + y \leq 7$ на множині $M = \{1, 2, 3, 4, 5, 6\}$. Визначте його властивості.

4. Нехай задано граф $G = (V, E)$; $V = \{1, 2, 3, 4\}$,

$E = \{(1,3),(2,3),(3,4),(4,1),(4,2)\}$. Побудувати діаграму, матриці суміжності й інцидентності.

3.



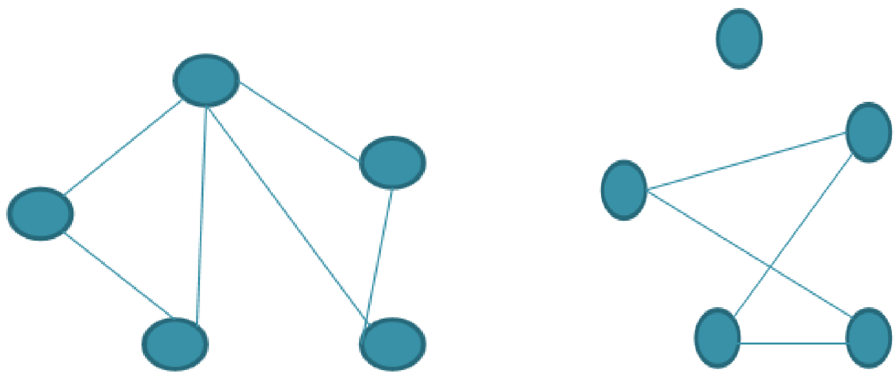
$G_1(V_1, E_1)$ $\qquad$ $G_2(V_2, E_2)$

Визначити степінь кожної вершини графів G_1, G_2. Побудувати по два підграфи для кожного графа.

3.3. Операції над графами

Унарні операції над графами

Доповненням графа $G = (V, E)$ називають граф $\overline{G} = (V, E')$, якщо його ребро (v_i, v_j) входить в V' тоді й тільки тоді, коли воно не входить в V. Інакше кажучи, дві вершини v_i і v_jсуміжні в $\overline{G}$, якщо вони не суміжні в G.

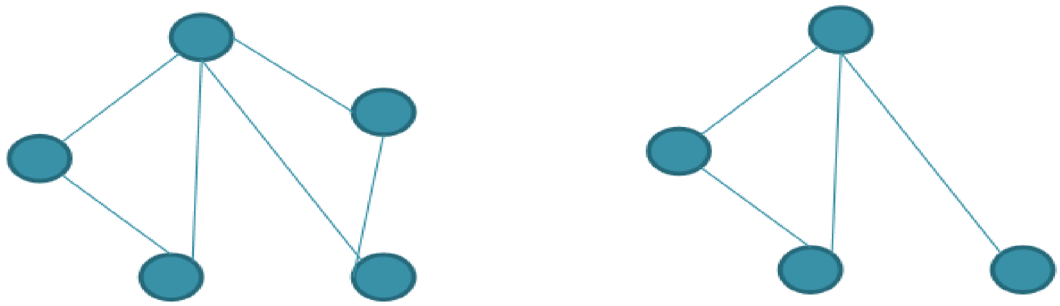


$G = (V, E)$ $\qquad$ $\overline{G} = (V, E')$

Рис. 3.49. Операція доповнення

Видалення вершини

Нехай v_i – вершина графа $G = (V, E)$. $G - v_i$ – граф, що отриманий видаленням з графа G вершини v_i та ребер, інцидентних цій вершині.

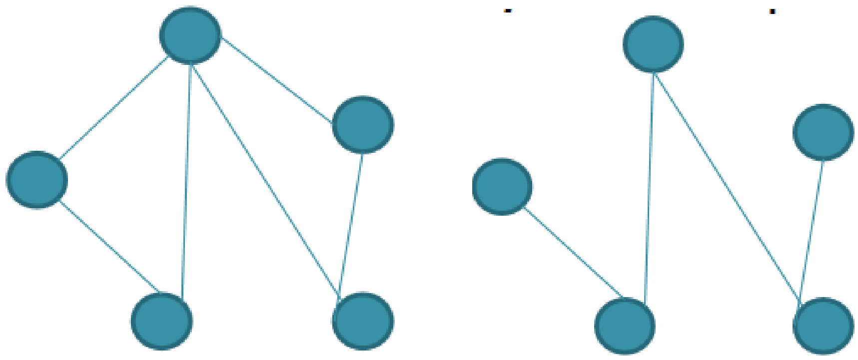


$G = (V, E)$ $G\text{-}v_i$

Рис. 3.50. Операція видалення вершини

Видалення ребер

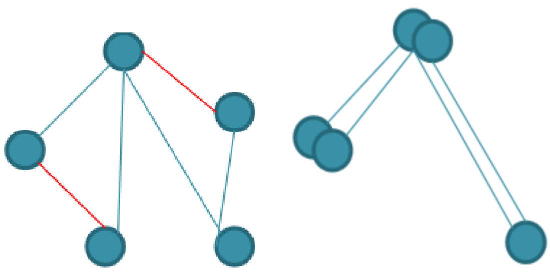
Нехай e_i – ребро графа $G = (V, E)$. Тоді $G – e_i$ – підграф графа G, який отримано після викидання ребра e_i.



$G = (V, E)$ $G\text{-}e_i$

Рис. 3.51. Операція видалення ребра

Стягування – операція видалення ребра e_i і ототожнювання його кінцевих вершин. Граф G називають стягненим до графа H, якщо H можна отримати з G послідовністю стягувань.



G H

Рис. 3.52. Операція стягування

Замкнення або ототожнювання

Кажуть, що пара вершин графа G v_i та v_j **замикаються (ототожнюються)**, якщо вони замінюються новою вершиною, усі ребра графа G, інцидентні v_i та v_j, стають інцидентними новій вершині.

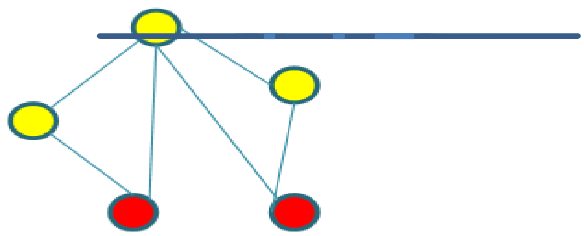


Рис. 3.53. Операція замкнення

Бінарні операції над графами

Об'єднання графів

Об'єднанням графів G_1 та G_2, позначається $G_1 \cup G_2$, є граф $G_3 = (V_1 \cup V_2, E_1 \cup E_2)$, множина його вершин є об'єднанням V_1 та V_2, а множина його ребер є об'єднанням E_1 та E_2.

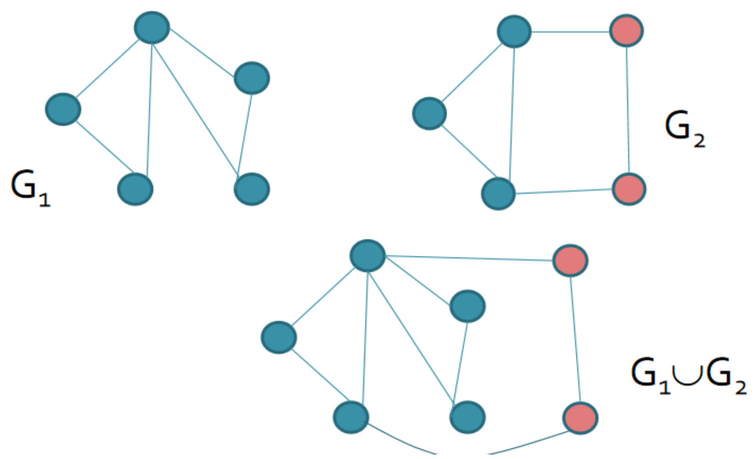
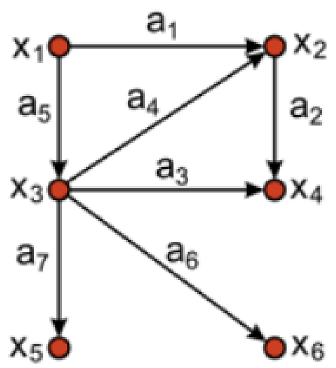


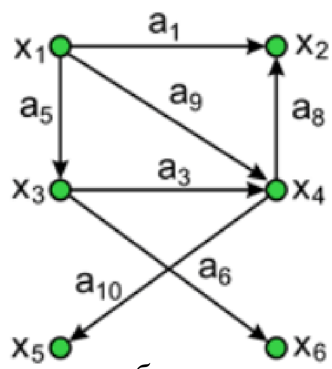
Рис. 3.54. Операція об'єднання

Приклад. 3.24. Об'єднання графів G_1і G_2, що позначається як $G_1 \cup G_2$, являє такий граф $G_3 = (X_1 \cup X_2, A_1 \cup A_2)$, що множина його вершин є об'єднанням X_1 і X_2, а множина ребер – об'єднанням A_1 і A_2. Граф G_3, що отриманий за допомогою операції об'єднання графів G_1 і G_2, показаний на рис. 3.55, *д*, а його матриця суміжності – на рис. 3.55, *е*.

Матрицю суміжності результуючого графа отримуємо за допомогою операції поелементного логічного додавання матриць суміжності вхідних графів G_1і G_2.



а



б

$A_1 =$

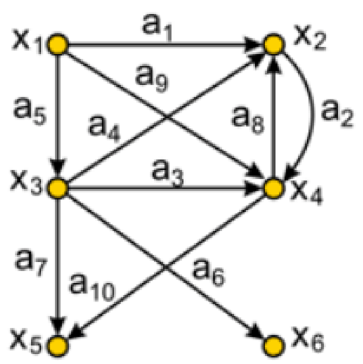
	x_1	x_2	x_3	x_4	x_5	x_6
x_1		1	1			
x_2				1		
x_3		1		1	1	1
x_4						
x_5						
x_6						

в

$A_2 =$

	x_1	x_2	x_3	x_4	x_5	x_6
x_1		1	1	1		
x_2						
x_3				1		1
x_4		1			1	
x_5						
x_6						

г



д

$A_3 =$

	x_1	x_2	x_3	x_4	x_5	x_6
x_1		1	1	1		
x_2				1		
x_3		1		1	1	1
x_4		1			1	
x_5						
x_6						

е

Рис. 3.55. Приклад об'єднання графів G_1 і G_2

Перетином графів G_1 та G_2 , позначається $G_1 \cap G_2$ є граф $G_3 = (V_1 \cap V_2, E_1 \cap E_2)$, тобто множина його вершин складається лише з тих вершин, які є одночасно в G_1 та G_2, а множина ребер G_3 складається з ребер, які одночасно присутні в G_1 та G_2.

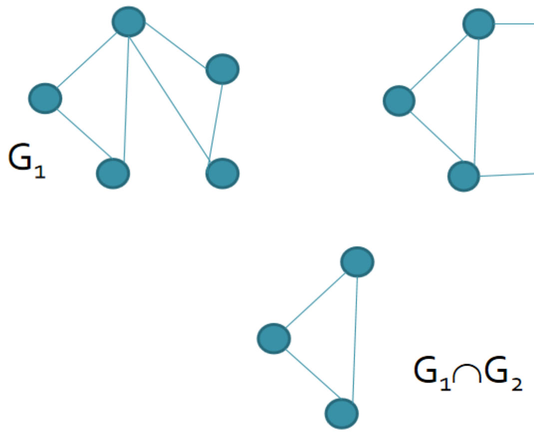
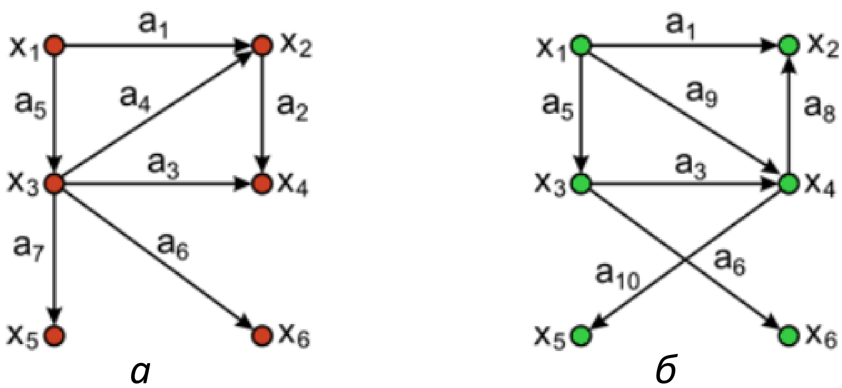


Рис. 3.56. Операція перетину

Приклад 3.25. Перетин графів G_1і G_2, що позначається як $G_1 \cap G_2$, являє собою граф $G_4 = (X_1 \cap X_2, A_1 \cap A_2)$. Таким чином, множина вершин графа G_4складається з вершин, що присутні одночасно в G_1і G_2. Операція перетину графів показана на рис. 3.57, *в*, а результуючу матрицю суміжності отримуємо за допомогою операції поелементного логічного добутку матриць суміжності вхідних графів G_1і G_2, що показано на рис. 3.57, *г*.



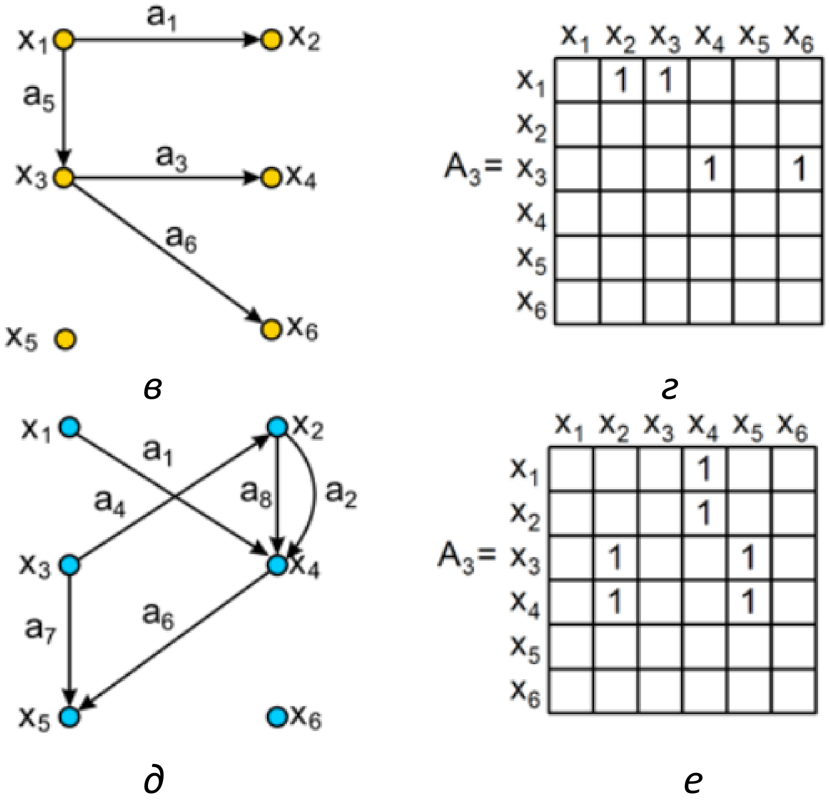


Рис. 3.57. Операція перетину і кільцевої суми:
а – граф G_1; *б* – граф G_2; в – граф $G_1 \cap G_2$; *г* – матриця суміжності графа $G_1 \cap G_2$; *д* – граф $G_1 \oplus G_2$; *е* – матриця суміжності графа $G_1 \oplus G_2$

Кільцева сума двох графів G_1 та G_2 позначається $G_1 \oplus G_2$, являє собою граф G_3, який складається з ребер, що присутні або в G_1, або в G_2, але не в обох одночасно.

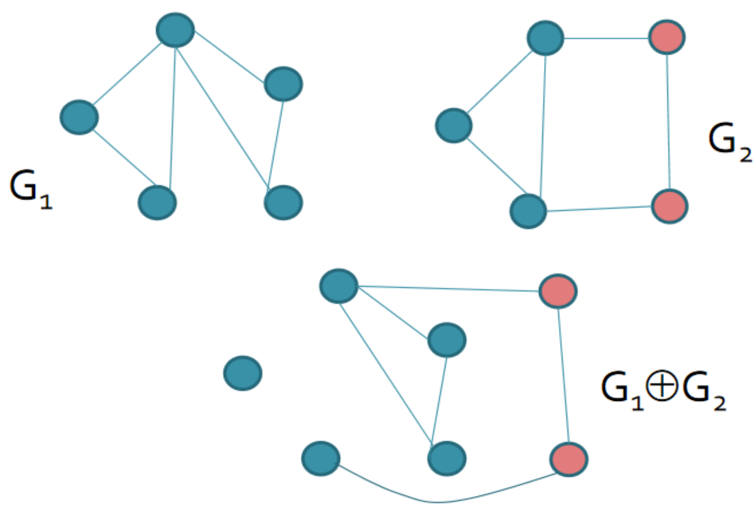
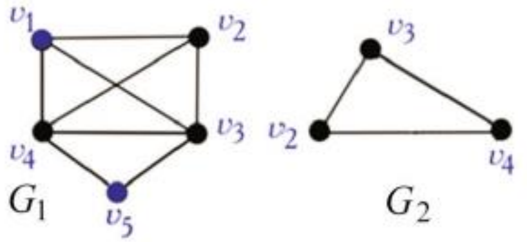


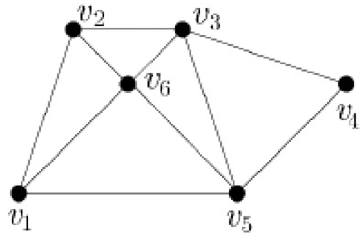
Рис. 3.58. Операція кільцевої суми

Обов'язкові завдання

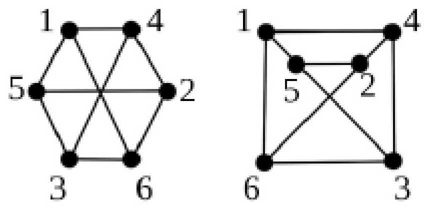
1. Визначте мінімальну кількість і послідовність операцій над графом G_1 для перетворення його у граф G_2.



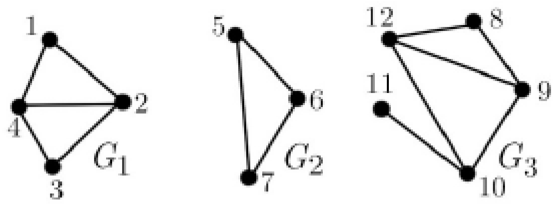
2. Для графа, зображеного на рис., побудувати матриці суміжності й інцидентності. Спираючись на властивості цих матриць, визначити основні характеристики графа.



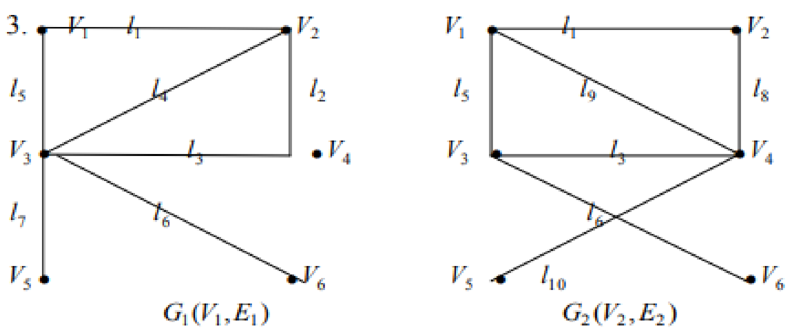
3. Довести ізоморфізм графів, показаних на рис.



4. Побудувати доповнення незв'язного графа, показаного на рис.



5. Визначити об'єднання, перетин і кільцеву суму двох графів G_1і G_2.



3.4. Графи та бінарні відношення

Відношенню R, заданому на множині V, взаємно однозначно відповідає орієнтований граф $G(R)$ без кратних ребер з множиною вершин V, у якому ребро (v_i, v_j) існує тільки тоді, коли виконано $v_i R v_j$. Представимо на графах деякі бінарні відношення.

Рефлексивність

Відношення Rна множині Vрефлексивне, якщо для кожного елемента $v \in V$справедливе $(v, v) \in R$. На графі це зображується петлею, а матриця суміжності графа з рефлексивними відношеннями містить одиниці на головній діагоналі. Інакше кажучи, якщо відношення R рефлексивне, то граф $G(R)$ без кратних ребер має петлі у всіх вершинах.

Приклад 3.26. Зобразити графічно та задати матрицею суміжності граф $G(R)$, заданий рефлексивним відношенням R.

Розв'язок.

На рис. 3.59 показано приклад графа рефлексивного відношення.

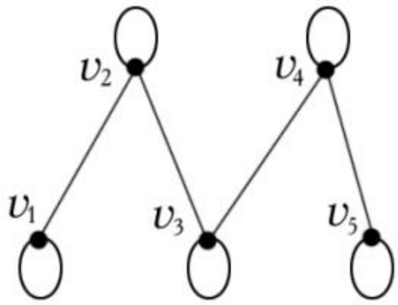


Рис. 3.59. Граф рефлексивного відношення

Головна діагональ матриці суміжності $G(R)$ складається з одиниць.

$$\mathbf{C} = \begin{Vmatrix} \mathbf{1} & 1 & 0 & 0 & 0 \\ 1 & \mathbf{1} & 1 & 0 & 0 \\ 0 & 1 & \mathbf{1} & 1 & 0 \\ 0 & 0 & 1 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 1 & \mathbf{1} \end{Vmatrix}$$

Антирефлексивність

Якщо відношення R на множині V антирефлексивне, то для всіх елементів v множини V справедливе $(v, v) \notin R$. Якщо R антирефлексивне, то граф $G(R)$ без кратних ребер не має петель.

Приклад 3.27. Зобразити графічно й задати матрицею суміжності граф G (R), заданий антирефлексивним відношенням R.

Розв'язок.

На рис. 3.60 показано приклад графа антирефлексивного відношення.

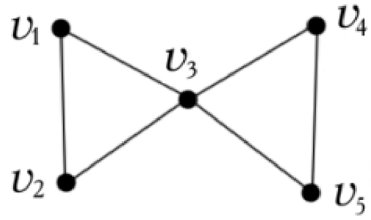


Рис. 3.60. Граф антирефлексивного відношення

Головна діагональ матриці суміжності $G(R)$ складається з нулів

$$\mathbf{C} = \begin{Vmatrix} \mathbf{0} & 1 & 1 & 0 & 0 \\ 1 & \mathbf{0} & 1 & 0 & 0 \\ 1 & 1 & \mathbf{0} & 1 & 1 \\ 0 & 0 & 1 & \mathbf{0} & 1 \\ 0 & 0 & 1 & 1 & \mathbf{0} \end{Vmatrix}$$

Обов'язкові завдання

1. Побудувати граф, який характеризується рефлексивністю, симетричністю і транзитивністю.

2. Зобразити графічно й задати матрицею суміжності граф G (R), заданий антитранзитивним та антисиметричним відношенням R.

3. Нехай дано множину вершин: $V = \{v_1, v_2, v_3, v_4, v_5\}$ та множину, на якій задано два відношення:
$R_1 = \{(v_1, v_2). (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_4, v_2), (v_3, v_5), (v_1, v_5)\}$,
$R_2 = \{(v_1, v_2). (v_2, v_3), (v_4, v_1), (v_4, v_3), (v_1, v_3), (v_2, v_5), (v_4, v_5)\}$.
Побудувати множину ребер графа, задану на об'єднанні відношень R_1 та R_2.
Представити графічно графи $G(R_1 \cup R_2)$, $G(R_1)$ та $G(R_2)$.
Розглянемо неорієнтований граф G (V, E), де
$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$.
Побудувати граф G (V, E).

3.5. Зв'язність графа

Зв'язність неорієнтованих графів

Нехай граф $\boldsymbol{G} = (\boldsymbol{V}, \mathbf{E})$ – неорієнтований. Вершина $\boldsymbol{a}$ називається зв'язаною з вершиною $\boldsymbol{b}$, якщо існує маршрут, який з'єднує ці вершини.

Стверджують, що вершина $\boldsymbol{b}$ **досяжна** з вершини $\boldsymbol{a}$.

Граф, будь-яка пара вершин якого є зв'язана, називається **зв'язним**.

Відношення зв'язності рефлексивне (вершина завжди зв'язана сама із собою), симетричне (із зв'язності вершини з вершиною випливає зв'язність вершини з вершиною) і транзитивне (якщо в довільному графі $\boldsymbol{G}$ вершина $\boldsymbol{a}$ зв'язана з $\boldsymbol{b}$, а вершина $\boldsymbol{b}$ зв'язана із $\boldsymbol{c}$, то, вочевидь, що $\boldsymbol{a}$ є зв'язана із $\boldsymbol{c}$). Таким чином, відношення зв'язності для вершин є відношенням еквівалентності. Тому існує таке розбиття множини вершин графа на підмножини (класи еквівалентності), які попарно не перетинаються, що всі вершини в кожній підмножині зв'язані, а вершини з різних підмножин не зв'язані. Кожна така підмножина вершин графа разом із ребрами, інцидентними цим вершинам, створює зв'язний підграф. Отже, неорієнтований граф можна представити єдиним чином у вигляді об'єднання зв'язних підграфів, що не перетинаються. Ці підграфи називають **зв'язними компонентами** графа, що розглядається. Зв'язний граф є своєю і єдиною компонентою зв'язності.

Відношення зв'язності для вершин є **відношенням еквівалентності**.

Отже, існує таке розкладання множини вершин $\boldsymbol{V}$:

1) $V = \boldsymbol{V_1} \cup \boldsymbol{V_2} \cup \ldots \cup V_p$, $V_i \cap V_j = \emptyset$ за $i \neq j$, на попарно неперерізувані підмножини, що всі вершини $\boldsymbol{b}$ однієї множини V_i є зв'язаними між собою, а вершини з різних множин V_i не є зв'язані.

2) $E = \boldsymbol{E_1} \cup E_2 \cup \ldots \cup E_p$, $E_i \cap E_j = \emptyset$ за $i \neq j$.

Тоді, відповідно до 1) та 2), матиме пряме розкладання.

3) $G = \boldsymbol{G_1} \cup G_2 \cup \ldots \cup \boldsymbol{G_p}$, де $\boldsymbol{G_1} = (\boldsymbol{V_1}, \boldsymbol{E_1})$, $\boldsymbol{G_2} = (\boldsymbol{V_2}, \boldsymbol{E_2})$, …, $\boldsymbol{G_n} = (\boldsymbol{V_p}, \boldsymbol{E_p})$ – неперерізувані зв'язні підграфи.

Ці підграфи називаються **зв'язними компонентами графа *G***, або **компонентами зв'язності графа *G***.

Число $\boldsymbol{p}$ – ще одна числова характеристика графа. Для зв'язного графа $\boldsymbol{p} = \mathbf{1}$; якщо граф є незв'язний, то $\boldsymbol{p} \geq \mathbf{2}$.

Теорема. Кожен неорієнтований граф розкладається в єдиний спосіб на пряму суму 3) власних зв'язних компонент.

Зауваження. Якщо певний граф не є зв'язним і розкладається на декілька компонентів, то вивчення цього незв'язного графа можна звести до досліджування окремих його компонентів, які є зв'язні.

Тому в переважній більшості випадків є сенс припускати, що заданий граф є зв'язний.

Граф, зображений на рисунку 3.61, має три компоненти зв'язності.

Оскільки кількість компонентів зв'язності дорівнює кількості зв'язних підграфів графа, то наведений граф – тризв'язний (число зв'язності $\boldsymbol{p}$ = **3**).

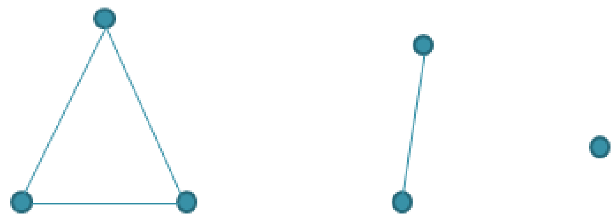


Рис. 3.61. Тризв'язний граф

Зв'язність для орієнтованих графів (орграфів) визначається так само, як і для неорієнтованих, тобто без урахування напрямків дуг.

Якщо в орграфі існує маршрут, що пов'язує вершини, то кажуть, що вершина **досяжна** з вершини. Будь-яка вершина вважається досяжною із себе самої. Вершина орграфа називається **джерелом**, якщо з неї досяжна будь-яка вершина орграфа.

Зв'язність орієнтованих графів визначається в принципі так само, як і неорієнтованих, тобто без урахування напрямку дуг. Специфічним для орграфа є поняття сильної зв'язності.

Орграф називається сильним (або **сильно зв'язним**), якщо будь-які дві його вершини досяжні одна з одної. Орграф називається **одностороннім** (або **одностороннє зв'язним**), якщо для будь-якої пари його вершин, меншою мірою, одна з них досяжна з іншої. Орграф називається сильно зв'язним, якщо для кожної пари його вершин $\boldsymbol{a}$ та $\boldsymbol{b}$ існує шлях з вершини $\boldsymbol{a}$ до вершини $\boldsymbol{b}$.

Рис. 3.62. Орграфи (зліва направо): сильний, односторонній і слабкий

Орграф називається **слабким** (або *слабо зв'язним*), якщо зв'язним графом є його неорієнтований дублікат. Оскільки вершина графа досяжна із себе самої, то одновершинний орграф буде одночасно і сильним, і одностороннім, і слабким. Кожний сильний орграф є одностороннім, а кожний односторонній – слабким. Вочевидь, дві будь-які вершини сильного орграфа, що не збігаються, належать певному контуру.

У деяких задачах істотною є вимога сильної зв'язності графа. Наприклад, граф, що представляє план міста з одностороннім рухом по деяких вулицях, повинен бути сильно зв'язаним, інакше знайдуться вершини (площі й перехрестя), між якими не можна було б проїхати містом без порушення правил руху.

Маршрут, що містить всі вершини орграфа, називається **остовним**.

Теорема. Орграф є сильним тоді й тільки тоді, коли в ньому є остовний контур, є одностороннім тоді й тільки тоді, коли в ньому є остовний шлях.

Відношення взаємної досяжності вершин орграфа рефлексивно, симетрично і транзитивно. Якщо це відношення еквівалентності, то воно розбиває множину вершин орграфа на класи еквівалентності, об'єднуючи в один клас всі вершини, що досяжні одна з одної. Вершини, що входять у такі класи, разом із дугами, їм інцидентними, обидві кінцеві вершини яких належать цьому ж класу, створюють підграфи, які називаються сильними (або сильно зв'язними) компонентами орграфа.

Орграф називається незв'язним, коли його неорієнтований дублікат не є зв'язним графом.

Орграф, зображений на рис. 3.38, має чотири сильні компоненти з множинами вершин $\{v_1, v_2, v_3, v_4\}$, $\{v_5, v_6, v_8\}$, $\{v_7\}$, $\{v_9\}$. В орграфі можуть бути дуги, що не входять в жодну з його сильних компонент, наприклад, дуги (v_3, v_5), (v_4, v_6), (v_9, v_2) і (v_9, v_8)у орграфа на рис. 3.63.

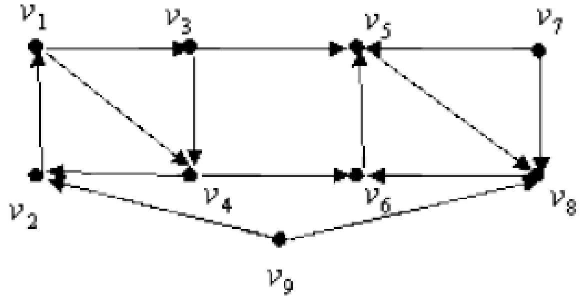


Рис. 3.63. Чотиризв'язний орграф

Вершинна зв'язність

Розглянемо граф, вершини якого відповідають деяким технологічним об'єктам, а ребра показують, які об'єкти можуть взаємодіяти або безпосередньо один з одним або опосередковано через інші об'єкти. Технологічна система, що представлена цим графом, вважається функціонуючою, якщо кожна пара її об'єктів зв'язана між собою. У цьому випадку система повинна мати зв'язний граф. Важливою характеристикою системи є її надійність, під якою зазвичай розуміють здібність системи функціонувати в разі виходу з ладу одного або декількох об'єктів і (або) порушення зв'язку між деякими з них. Вочевидь, що менш надійною слід вважати ту систему, яка перестає функціонувати з виходом із ладу меншої кількості її елементів. Оцінити степінь надійності такої системи можуть допомогти ті поняття, про які згадувалося вище і які зараз будуть визначені.

Визначення. Числом вершиною зв'язності (або просто **числом зв'язності**) $\chi(G)$ графа *G* називається число, що дорівнює найменшому числу вершин, видалення яких призведе до незв'язного або одновершинного графа.

Граф *G*, представлений на рис. 3.64, зв'язний, але він перестане бути зв'язним, якщо видалити вершину 4. Тому $\chi(G) = 1$.

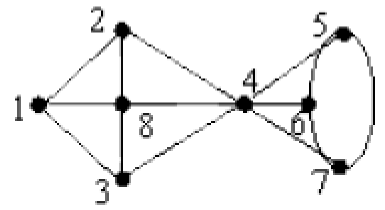


Рис. 3.64

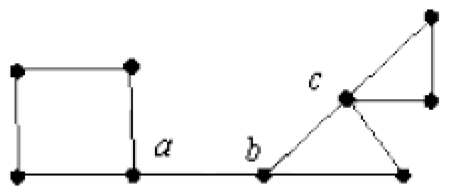


Рис. 3.65

Можна порушувати зв'язність графа, видаляючи деякі його ребра (дуги). У графа (рис. 3.64) для цього треба видалити не менше трьох ребер. Наприклад, граф розпадеться на дві компоненти після видалення ребер 4&5, 4&6, 4&7.

Реберна зв'язність

Визначення. Числом реберної зв'язності $\boldsymbol{\lambda(G)}$ графа *G* називається число, яке дорівнює найменшому числу ребер, видалення яких приведе до незв'язного графа. Число реберної зв'язності одновершинного графа дорівнює нулю.

Вище показано, що для графа *G* (рис. 3.61) $\boldsymbol{\lambda(G) = 3}$.

Вершина v графа G називається **точкою зчленування**, якщо граф G-v, отриманий після операції видалення у графа G вершини v, має більше компонент зв'язності, ніж сам граф G. Якщо G зв'язний і v точка зчленування, то G-v не зв'язний.

Ребро e графа G називається **мостом**, якщо його видалення збільшує число компонент зв'язності графа.

Таким чином, точки зчленування й мости – це свого роду «вузькі місця» графа. Граф на рис. 3.65 має три точки зчленування – це вершини a, b, c, і один міст – ребро a & b.

Повертаючись до технологічної системи, про яку йшлося спочатку, зазначимо, що число вершинної зв'язності й число реберної зв'язності її графа відображають чуттєвість системи до пошкоджень, а точки зчленування й мости графа системи вказують на найбільш вузькі місця системи.

Граф називається **нероздільним**, якщо він зв'язний і не має точок зчленування. Граф, що має хоча б одну точку зчленування, є роздільним і називається **сепарабельним**. Він розбивається на блоки, кожний з яких являє собою максимальний нероздільний підграф.

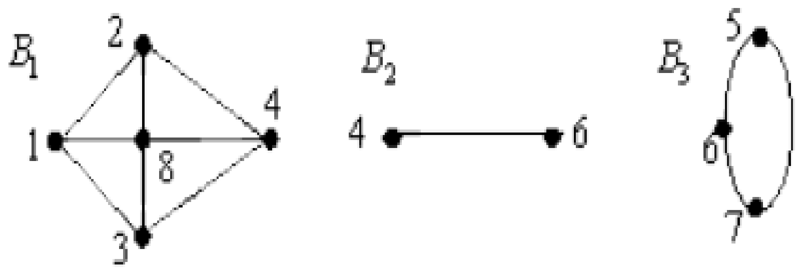


Рис. 3.66. Блоки B_1, B_2, B_3 графа з рис. 3.64

Якщо $\delta(G)$є мінімальна степінь вершин графа G, то вочевидь, що $\lambda(G) \leq \delta(G)$, оскільки видалення всіх ребер, інцидентних цій вершині, приведе до збільшення числа компонент зв'язності графа.

Теорема. *Для будь-якого графа G справедливі нерівності*: $\chi(G) \leq \lambda(G) \leq \delta(G)$.

Граф G називається k-*зв'язним*, якщо $\chi(G) \geq k$, *реберно* k-*зв'язним, якщо* $\lambda(G) \geq k$.

Граф, зображений на рис. 3.64, 1-зв'язний і реберно 3-зв'язний.

Отже, підсумуємо. Означення. Дві вершини v і w називаються зв'язаними, якщо існує маршрут з кінцями v та w. Граф називається зв'язним, якщо будь-яка пара його вершин є зв'язаною. Якщо граф не є зв'язним, то він називається незв'язним.

Визначення. Зв'язністю графа називається мінімальна кількість вершин, вилучення яких приводить до утворення незв'язного графа.

Приклад 3. 26. Граф, наведений на рис. 3.67, є зв'язним, тому що будь-яка пара його вершин зв'язана. Але яка його зв'язність? Тобто якою є мінімальна кількість вершин його графа, вилучення яких приводить до утворення незв'язного графа?

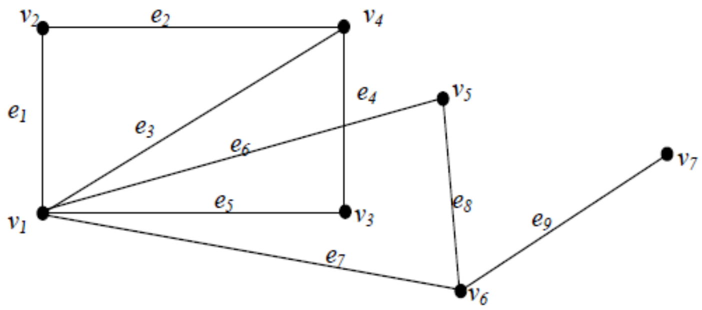


Рис. 3.67. Зв'язний граф

Очевидно, що зв'язність дорівнює одиниці, тому що вилучення однієї вершини v_1 приводить до утворення незв'язного графа.

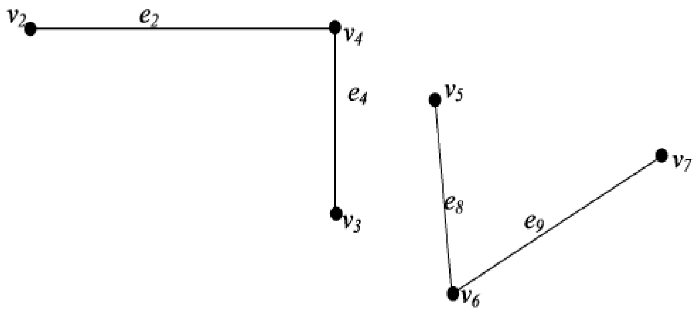
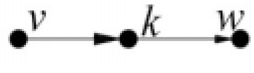


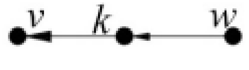
Рис. 3.68. Граф, після вилучення вершини v_1

3.6. *Досяжність і контрдосяжність вершини в графах*

Визначення. Вершину *w* графа *D* (або орграфа) називають досяжною з вершини *v*, якщо *w* = *v*, або існує шлях з *v* у *w* (маршрут від *v* у *w*).



Визначення. Вершину *w* графа *D* (або орграфа) називають контрдосяжною з вершини *v*, якщо існує шлях з *w* у *v* (маршрут від *w* у *v*).



Матриця відстаней графа

Нехай задано зв'язний граф G =(V, E). Відстанню поміж двома вершинами v та w графа G називається довжина найкоротшого ланцюга, який зв'язує ці вершини.

Відстань між вершинами v та w графа G позначається через G d (v, w).

Аксіоми метрики:

1) d (v, w)> 0; d (v, w) = 0 <=> $v = w$;

2) d (v, w) = d (v, w);

3) d (v, z) = d (v, w) + d (w, z).

Діаметром графа називається величина d (G), де максимум береться за всіма можливими парами вершин графа: $d(G) = \max\limits_{v,w} d(G)$

Визначимо для кожної вершини x графа G величину r (x), тобто відстань від вершини x до найвіддаленішої від x вершини графа:

$$r(x) = \max_{y} d(x, y).$$

Мінімум цієї величини за всіма вершинами графа називається **радіусом** графа G:

$$r\ (G) = \min_{x} r(x) = \min_{x} \max_{y} d(x, y)$$

Вершина x_0, якої досягається цей мінімум, називається **центральною**.

Приклад 3.27. Знайти діаметр і радіус для графа, зображеного на рис. 3.69.

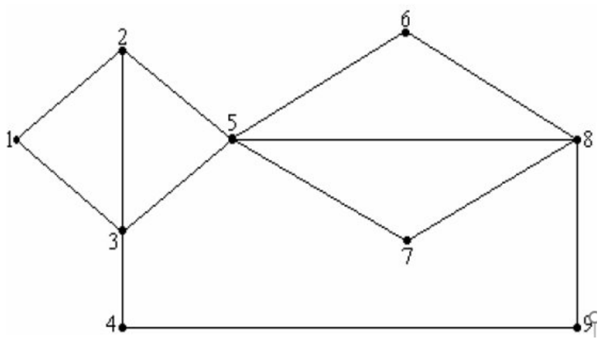


Рис. 3.69

Для розв'язання цієї задачі зручно попередньо обчислити так звану матрицю відстаней поміж вершинами графа. У цьому разі це буде матриця розміром 9 × 9, елемент якої, що стоїть на місці (i, j), дорівнює відстані від вершини і до вершини j.

$$\begin{pmatrix} 0 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 3 \\ 1 & 0 & 1 & 2 & 1 & 2 & 2 & 2 & 3 \\ 1 & 1 & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\ 2 & 2 & 1 & 0 & 2 & 3 & 3 & 2 & 1 \\ 2 & 1 & 1 & 2 & 0 & 1 & 1 & 1 & 2 \\ 3 & 2 & 2 & 3 & 1 & 0 & 2 & 1 & 2 \\ 3 & 2 & 2 & 3 & 1 & 2 & 0 & 1 & 2 \\ 3 & 2 & 2 & 2 & 1 & 1 & 1 & 0 & 1 \\ 3 & 3 & 2 & 1 & 2 & 2 & 2 & 1 & 0 \end{pmatrix} \begin{array}{c} 3 \\ 3 \\ 2 \\ 3 \\ 2 \\ 3 \\ 3 \\ 3 \\ 3 \end{array}$$

За визначенням, діаметр графа дорівнює найбільшому елементові матриці відстаней. Отже, $d = 3$.

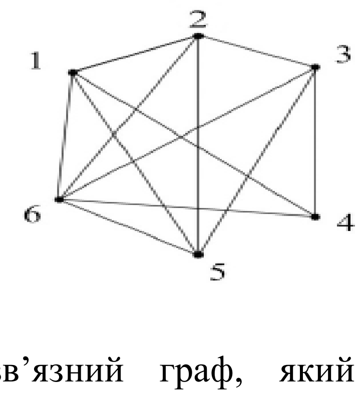
Для знаходження радіуса відшукаємо в кожному рядку найбільше число; ці числа виписано праворуч від матриці відстаней. Найменші з них дають значення радіуса $r = 2$. Вершини 3-тя та 5-та є центральними.

<u>Обов'язкові завдання</u>

1. Нехай дано зв'язний граф $G\ (V, E)$. Визначити величину загального степеня вершин графа за умови, що $|V| = 5$ і $|E| = 8$. Побудувати варіант графа з максимально можливим степенем вершини.

2. Позначимо через n – кількість будинків і m – кількість доріг. Будинки пронумеровані від 1 до n. Кожна дорога визначається двійкою чисел – двома номерами будинків – кінців дороги. У кожному будинку живе по одній людині. Знайти точку – місце зустрічі всіх людей, від якої сумарна відстань до всіх будинків буде мінімальною за умови, що довжина всіх доріг однакова.

3. Побудувати підграф, правильний підграф та остовний підграф графа, показаного на рис.



4. Побудувати незв'язний граф, який містить 12 вершин, 3 компоненти і мінімально можливу кількість вершин.

3.7. Цикломатика графів

Цикломатика – це вивчення циклів у графі.

З усієї сукупності циклів цього графа можна відокремити цілковито певну кількість незалежних (базисних) циклів, решту здобути з базисних циклів за допомогою спеціальної операції додавання.

Приклад такого додавання:

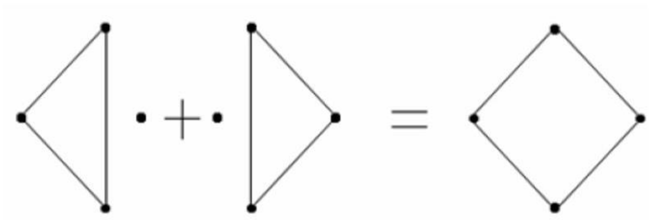


Рис. 3.70. Приклад додавання базисних циклів

Циклові ребра та перешийки

Нехай задано граф G = (V, E). Ребро графа, через яке проходить хоча б один цикл, назвемо цикловим ребром. Ребро, яке не входить до жодного циклу, називається **перешийком.**

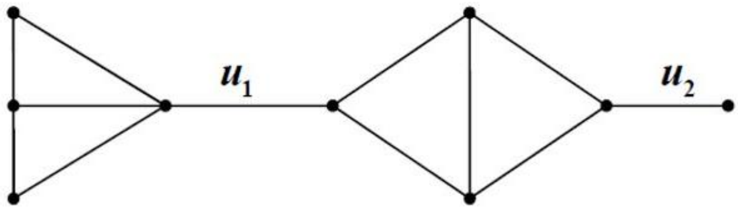


Рис. 3.71. Граф із перешийками та цикловими ребрами

Ребра u_1 та u_2 – перешийки, решта ребер − циклові.

Лема. У разі вилучення зі зв'язного графа циклового ребра він залишається зв'язним. Із вилученням зі зв'язного графа перешийка граф розпадається на дві компоненти зв'язності.

Доведення. Якщо ребро $\{a, b\}$, яке вилучається, є цикловим, то після його вилучення вершини a та b, як і раніше, можна з'єднати в ланцюг – залишком циклу, до якого входило ребро $\{a, b\}$. Звідси випливає, що і кожні дві вершини графа після вилучення ребра {a, b} залишаються зв'язаними ланцюгом. І навпаки, якщо $\{a, b\}$ – перешийок, то після його вилучення вершини a та b не можна зв'язати ланцюгом, інакше цей ланцюг разом із ребром $\{a, b\}$ утворить цикл у вихідному графі. З іншого боку, кожна вершина залишається зв'язаною чи то з вершиною a, чи то з вершиною b.

Наслідок. Із вилученням із графа циклового ребра кількість зв'язних компонентів графа не змінюється. Із вилученням перешийка кількість зв'язних компонентів графа збільшується на одиницю.

Цикломатичне число

Нехай задано граф $G = (V, E)$, $|V| = n$, $|E| = m$; p – кількість компонент зв'язності графа. Величина $\lambda = m - n + p$ називається **цикломатичним числом графа.**

Теорема. Для кожного графа цикломатичне число є невід'ємне, тобто $\lambda \geq 0$.

Доведення. Вилучаємо з графа по одному ребру й відстежуємо змінювання величини λ.

Параметри вихідного графа позначимо m, n і p. Ті самі величини після вилучення ребра позначимо через m', n' та p'.

У процесі вилучення ребер можливі два випадки:

а) ребро, яке вилучається – циклове. $m' = m - 1$, $n' = n$, $p' = p$; $\lambda' = m' - n' + p' = m - 1 - n + p = \lambda - 1$;

б) ребро, яке вилучається − перешийок. $m' = m - 1$, $n' = n$, $p' = p + 1$. $\lambda' = m' - n' + p' = m - 1 - n + p + 1 = m - n + p = \lambda$.

Отже, із вилученням ребра величина λ або не змінюється, або зменшується на одиницю.

Після вилучення всіх ребер дістанемо порожній граф, у якому $m_0 = 0, n_0 = n, p_0 = n$, тобто $\lambda_0 = m_0 - n_0 + p_0 = n - n = 0$.

Отже, у вхідного графа $\lambda \geq 0$.

Зауваження. Доведення теореми свідчить про зв'язок цикломатичного числа з наявністю циклів у графі.

Якщо $\lambda > 0$, то у графі є принаймні один цикл. Із вилученням циклового ребра деякі цикли розриваються, і це призводить до зменшення λ. Якщо продовжувати вилучення ребер, то зрештою розриваються всі цикли − і λ дорівнює 0.

Після цього λ вже не змінюється, тому що всі ребра стали перешийками. Цикломатичне число графа вказує кількість ребер, які потрібно видалити, щоб отримати дерево (для зв'язного графа) або ліс (для незв'язного графа), тобто досягти відсутності циклів.

Цикломатичне число графа дорівнює максимальній кількості незалежних циклів.

Знання цикломатичного числа корисне під час аналізу топології електросхем, а також для задач конструкторського проєктування на ЕОМ.

Приклад 3.28. Визначити цикломатичне число графа *G* (*V*, *E*), показаного на рис. 3.72.

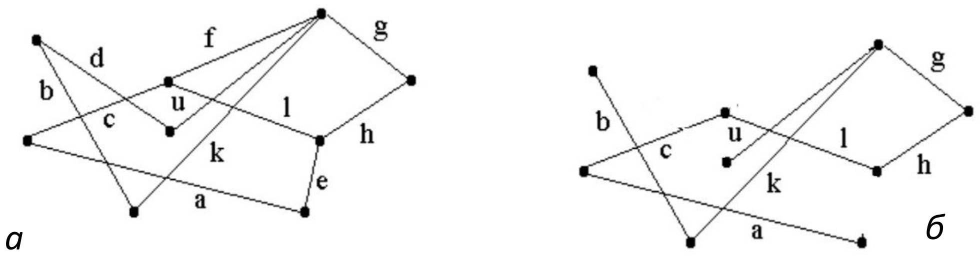


Рис. 3.72: *а* – граф *G* (*V*, *E*), *б* – граф *G* (*V*, *E*) після вилучення циклових ребер

Розв'язок: $m = 11$; $n = 9$; $p = 1$; $\lambda = 11 - 9 + 1 = 3$.

Приклад 3.29. Визначити цикломатичне число графа *G* (*V*, *E*), показаного на рис. 3.73.

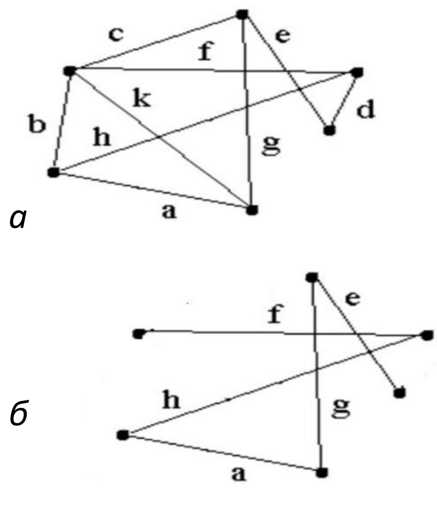


Рис. 3.73: *а* – граф *G* (*V*, *E*), *б* – граф *G* (*V*, *E*) після вилучення циклових ребер

Розв'язок: $m = 9$; $n = 6$; $p = 1$; $\lambda = 9 - 6 + 1 = 4$.

Обов'язкові завдання

1. Визначити цикломатичне число зв'язного графа G (V, E), якщо граф характеризується множиною вершин $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, а потужність множини ребер $|E| = 12$.

2. Нехай дано граф G (V, E), де $V = \{v_1, v_2, v_3, v_4, v_5\}$.
$E = \{(v_1, v_2), (v_5, v_2), (v_3, v_4), (v_1, v_4), (v_1, v_5), (v_3, v_2), (v_4, v_5), (v_4, v_2)\}$.

Скільки ребер потрібно видалити, щоб граф став деревом? Визначити підмножини ребер, які можуть бути видалені без втрати зв'язності графа.

3.8. Дерева та їхні властивості, ліс, цикли

Визначення дерева. Властивості дерев

Дерево – це сукупність елементів, що називаються вузлами (один з яких корінь), і відношень («батьківських»), що утворюють ієрархічну структуру вузлів. Вузли можуть бути елементами будь-якого типу (літерами, рядками, числами).

Піддерево, корінь якого перебуває в лівому (правому) нащадку вершини, називається лівим (правим) піддеревом цієї вершини.

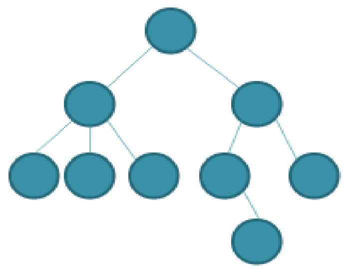


Рис. 3.74. Приклад дерева

Висота вузла дерева – це довжина самого довгого шляху із цього вузла до будь-якого листа.

Глибина вузла – це довжина шляху від кореня до цього вузла.

Степінь вузла – це кількість дуг, що з нього виходить.

Степінь дерева дорівнює максимальному степеню вузла, що входить у дерево.

Листя в дереві – це вузли, що мають степінь нуль.

Бінарне дерево – це дерево, степінь якого дорівнює два. Дерева, степінь яких більше двох, називаються **розгалуженими**.

Повне бінарне дерево – це дерево, для якого на всіх рівнях менше чим *n* вузли мають степінь 2, а на рівні *n* – степінь 0.

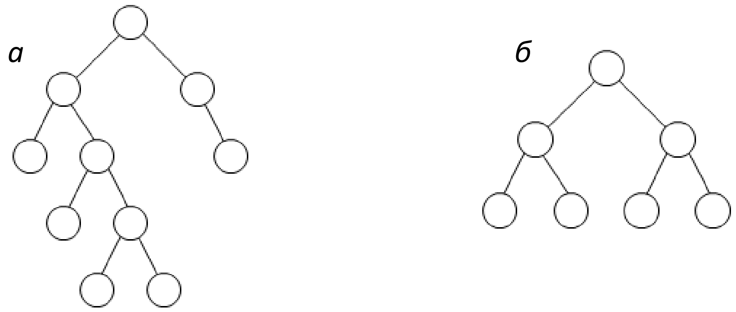


Рис. 3.75. Бінарні дерева: *а* – неповне бінарне дерево; *б* – повне бінарне дерево

Строго бінарне дерево складається тільки з вузлів, що мають степінь 2 або 0.

Не строго бінарне дерево містить вузли зі степенем 1.

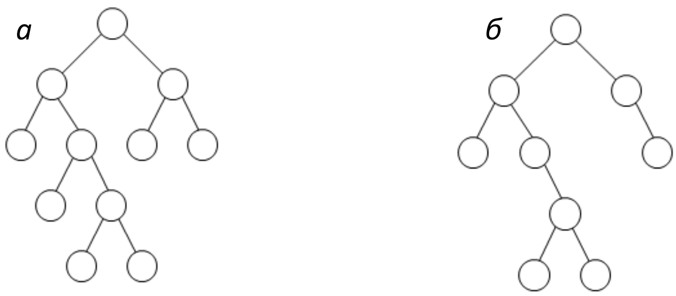


Рис. 3.76. Бінарні дерева: а – строго бінарне дерево; б – не строго бінарне дерево

Бінарне дерево пошуку (binary search tree – BST) – це бінарне дерево, у якому:

- кожен вузол має не більше двох нащадків;
- кожен вузол має ключ (key) і значення (value);
- ключі всіх вузлів лівого піддерева менші за значення ключа батьківського вузла;
- ключі всіх вузлів правого піддерева більші за значення ключа батьківського вузла.

Використовуються для реалізації словників і множин.

Алгоритм вставки елемента

1. Починаємо з кореня.
2. Якщо елемент < об’єкта у вершині, переходимо до лівого сина.

3. Якщо елемент > об’єкта у вершині, переходимо до правого сина.

4. Повторюємо кроки 2 і 3, доки не досягнемо вершини, яка не визначена.

5. Якщо досягнута вершина не визначена, то визначаємо вершину і вставляємо елемент.

Приклад 3.30. Побудувати дерево пошуку: 54, 37, 63, 21, 46, 73, 59, 12, 40, 60.

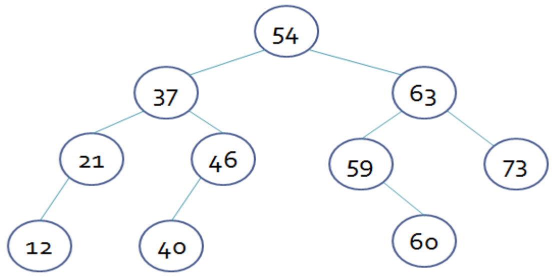


Рис. 3.77. Бінарне дерево пошуку

Алгоритм пошуку елемента

1. Починаємо з кореня.

2. Якщо елемент < об’єкта у вершині, переходимо до лівого сина.

3. Якщо елемент > об’єкта у вершині, переходимо до правого сина.

4. Якщо елемент = об’єкта у вершині, то елемент знайдено; виконуємо відповідні дії і виходимо.

5. Повторюємо кроки 2, 3 і 4, доки не досягнемо вершини, яка не визначена.

6. Якщо досягнута вершина не визначена і в дереві немає шуканого елемента, то виконуємо відповідні дії і виходимо.

Алгоритм видалення елемента

1. Якщо вершина v_0 не має синів, просто видаляємо її.

2. Якщо вершина v_0 має одного сина, видаляємо v_0 і заміняємо її сином.

3. Якщо v_0 має двох синів, знаходимо правого сина v_1 вершини v_0, а потім знаходимо лівого сина вершини v_1 (якщо він існує). Продовжуємо вибирати лівих синів кожної знайденої вершини, доки не знайдеться така вершина v, у якої не буде лівого сина. Замінимо v_0 на v і зробимо правого сина вершини v лівим сином батька вершини v.

Приклад 3.31.

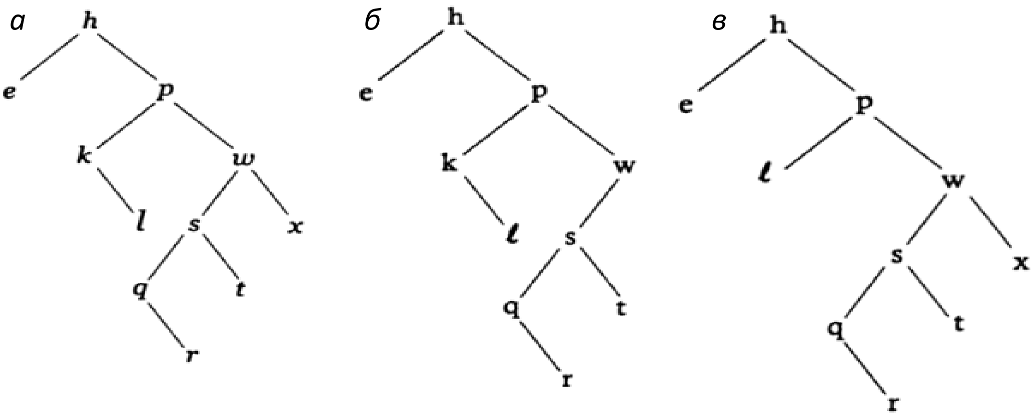


Рис. 3.78: *а* – задане дерево; *б* – дерево після видалення вершини *x*;
в – дерево, після видалення вершини *k*

Якщо v_0 має двох синів, знаходимо правого сина v_1 вершини v_0, а потім знаходимо лівого сина вершини v_1 (якщо він існує). Продовжуємо вибирати лівих синів кожної знайденої вершини, доки не знайдеться така вершина v, у якої не буде лівого сина. Замінимо v_0 на v і зробимо правого сина вершини v лівим сином батька вершини v.

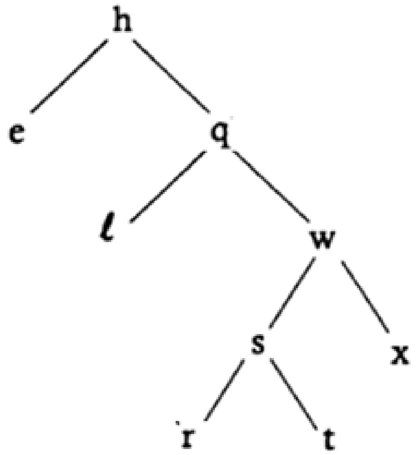


Рис. 3.79. Дерево після видалення вершини *q*

Збалансоване дерево пошуку – це дерево пошуку, у якому число вершин у його лівих і правих піддеревах відрізняються не більше ніж на 1.

Збалансовані дерева пошуку:

- червоно-чорні дерева
- АВЛ-дерева

- 2-3-дерева
- B-дерева
- …

Обхід дерев

Під **обходом** бінарного дерева розуміють визначений порядок проходження всіх вершин дерева. Розрізняють: прямий, зворотний і симетричний порядки обходу.

Прямий порядок обходу*:*

1) почати з кореня R;
2) пройти в прямому порядку ліве піддерево A;
3) пройти в прямому порядку праве піддерево B .

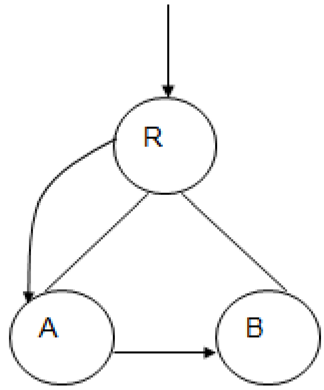


Рис. 3.80. Ілюстрація прямого порядок обходу дерева

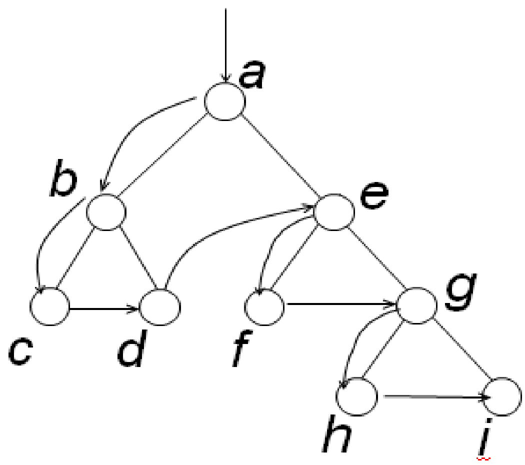


Рис. 3.81. Прямий порядок обходу бінарного дерева: *a b c d e f g h i*

Обхід дерева в зворотному порядку:

- пройти в зворотному порядку ліве піддерево A
- пройти в зворотному порядку праве піддерево B
- потрапити в корінь R

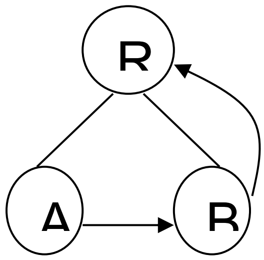


Рис. 3.82. Ілюстрація зворотного порядку обходу дерева

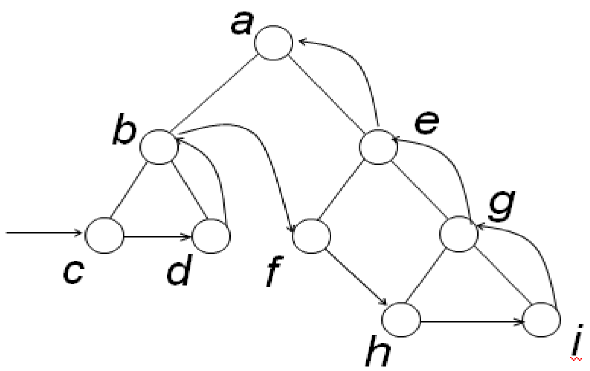


Рис. 3.83. Зворотний порядок обходу бінарного дерева: *c d b f h i g e a*

Симетричний порядок обходу бінарного дерева:

- пройти в симетричному порядку ліве піддерево A
- потрапити в корінь R
- пройти в симетричному порядку праве піддерево B

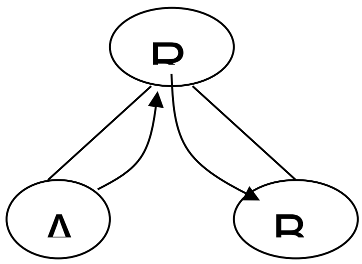


Рис. 3.84. Ілюстрація симетричного порядку обходу дерева

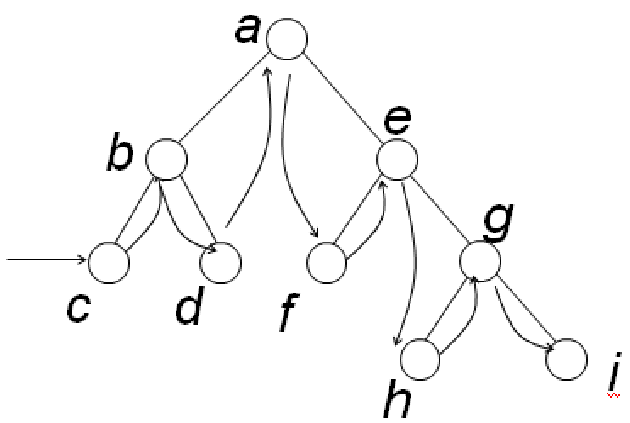


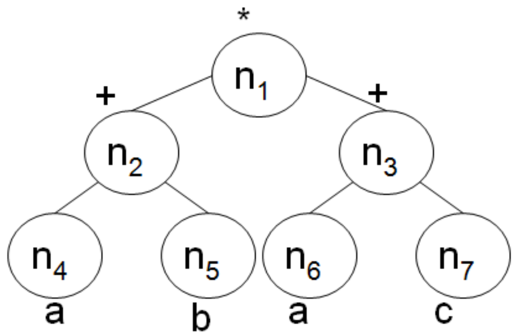
Рис. 3.85. Симетричний порядок обходу бінарного дерева: *c b d a f e h g i*

Дерева виразів

Часто під час обходу дерев складається список не імен вузлів, а їх міток (label) – значень, які зберігаються у вузлі. Такі дерева називаються **дерева з мітками**. Проводять таку аналогію: дерево – список, вузол – позиція, мітка – елемент.

За допомогою дерев можна представляти довільні арифметичні вирази. Кожному листові в такому дереві відповідає операнд, а кожному батьківському вузлу – операція. У загальному випадку дерево при цьому може виявитися не бінарним.

Приклад 3.32. На рисунку наведено дерево з мітками, що представляє арифметичний вираз $(a + b)*(a + c)$, де n_1, n_2… n_7 – імена вузлів, а мітки представлені поруч із відповідними вузлами.



- За прямого впорядкування міток отримуємо **префіксну** форму виразів, де оператор передує і лівому, і правому операндам. Префіксний вираз: $* + ab + ac$.
- Зворотне впорядкування міток дерева виразів дає **постфіксне** (або польське) представлення виразів, за якого оператор іде після лівого і правого операндів. Постфіксна форма: $ab + ac + *$.
- За симетричного обходу дерев виразів отримаємо так звану **інфіксну** форму виразу, яка збігається зі звичайною стандартною формою запису виразу, але не використовує дужок. Інфіксний вираз: $a + b*a + c$.

Орієнтованим деревом називають зв’язний орієнтований граф без циклів, у якому напівстепінь входу кожної вершини, за винятком кореневої, дорівнює одиниці, а напівстепінь входу кореневої вершини дорівнює 0 (рис. 3.77).

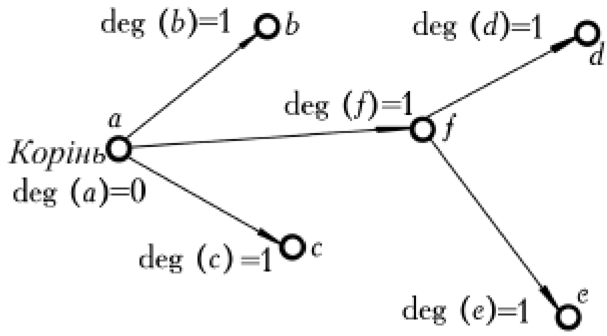


Рис. 3.86. Орієнтоване дерево

Остовним підграфом називають такий підграф, у якому множина його вершин збігається з множиною вершин самого графа.

Остовним деревом графа *G* називають **остовний підграф** графа *G*, який є деревом.

Початковий граф, остовний підграф та остовне дерево, які утворені шляхом послідовного видалення ребер, показані на рис. 3.78.

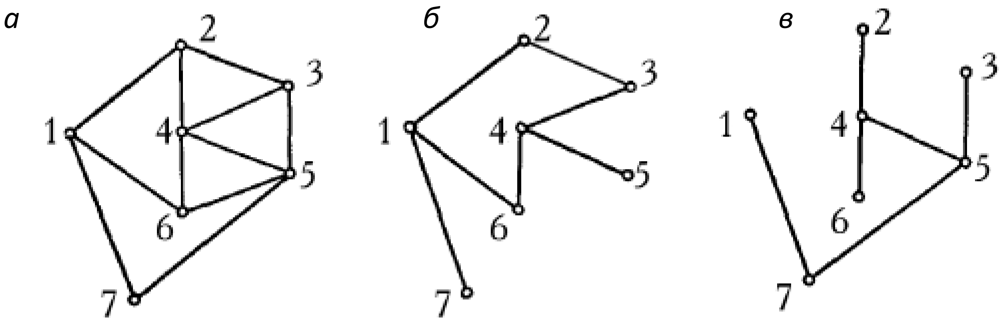


Рис. 3.87. Утворення остовного підграфа й остовного дерева шляхом видалення ребер: *а* – початковий граф; *б* – остовний підграф; *в* – остовне дерево

Теорема 3.1. Граф є деревом тоді й тільки тоді, коли будь-які дві його вершини зв’язані єдиним ланцюгом.

Доведення. Нехай граф є деревом. Якщо припустити існування більш ніж одного ланцюга, що зв’язує будь-які дві його вершини, то в такому графі існує цикл, тобто граф не може бути деревом (рис. 3.88).

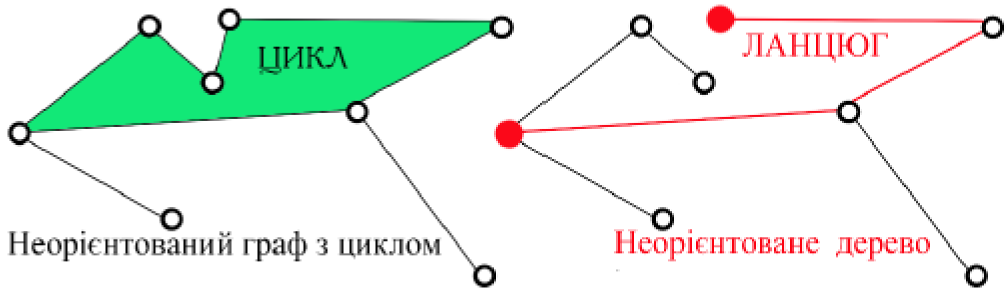


Рис. 3.88. Граф із циклом і неорієнтоване дерево

І навпаки, оскільки будь-які дві вершини графа з'єднані ланцюгом, то граф зв'язний, а в силу того, що цей ланцюг єдиний, він не має циклів. Отже, граф є деревом. Теорему доведено.

Наслідок 1. Якщо *T* – дерево і *u* – його кінцева вершина (листок), то граф *T* − *u* (*T мінус u*) – дерево. Дійсно, граф *T* −*u* – **правильний підграф** дерева *T*, для якого виконуються всі умови теореми (рис. 3.89).

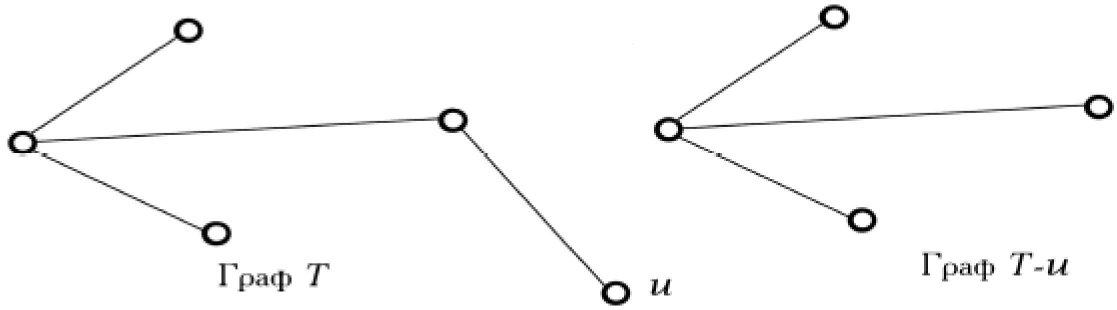


Рис. 3.89. Неорієнтовані дерева *T* і *T* − *u*

Наслідок 2. Будь-яке непусте дерево має принаймні дві висячі вершини й одне висяче ребро.

1. **Висяча вершина в неорієнтованому графі** – це вершина степеня 1.

2. **Висяча вершина в орграфі** – вершина з напівстепенем входу, що дорівнює 1, і напівстепенем виходу, що дорівнює 0.

3. **Висяче ребро** – це ребро, інцидентне вершині зі степенем 1.

Визначення. Ребро зв'язного графа називають **істотним**, якщо його видалення веде до порушення зв'язності цього графа.

Визначення. У неорієнтованому графі істотним ребром є міст.

Теорема 3.2. У дереві кожне ребро істотне.

Доведення. Доведення випливає з того, що видалення ребра *e* = (*u*, *v*) у дереві *T* через наявність єдиного ланцюга, який з'єднує вершини *u* і *v*, веде до появи двох компонентів зв'язності: один компонент містить вершину *u*, а інший – вершину v . Отже, граф $T - e$ не є зв'язним, як показано на рис. 3.90.

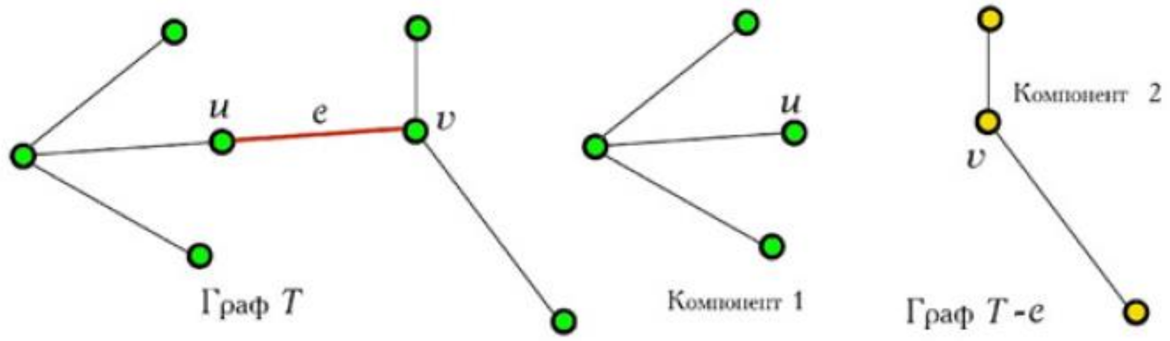


Рис. 3.90. Зв'язний граф T і незв'язний граф $T - e$

Теорема 3.3. Якщо $T = (V, E)$ – дерево і $v \notin V$ – вершина, u – довільна вершина $u \in V$, то граф $T' = (V \cup \{v\}, E \cup \{(u, v)\})$, теж є деревом.

Доведення. Оскільки T – дерево, то існує єдиний ланцюг, що з'єднує будь-яку вершину u' з вершиною u . Оскільки вершина $v \notin V$, то додавання одного кінцевого ребра (u, v) приводить до того, що з кожної вершини u' маємо лише єдиний ланцюг, який з'єднує вершини u' і v. Виходячи з теореми, граф T' є деревом.

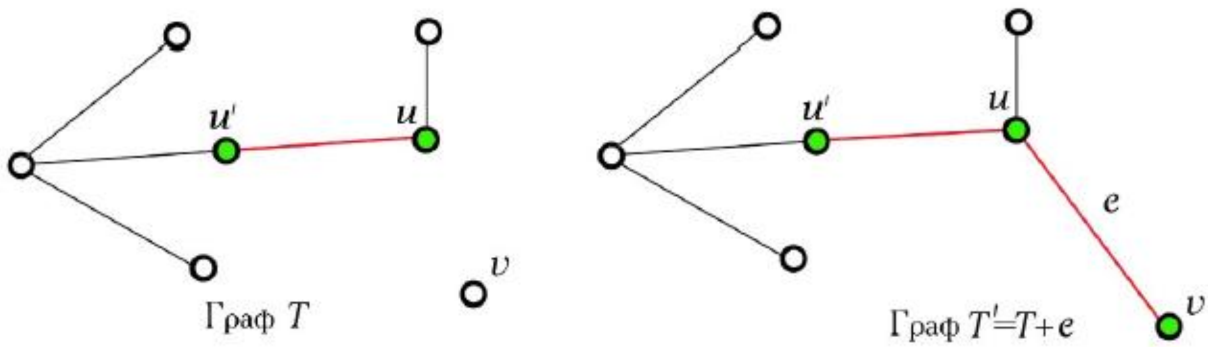


Рис. 3.91. Зв'язні графи-дерева T і $T + e$

Теорема 3.4. Нехай дерево T має n вершин. Тоді еквівалентними є такі твердження:

1. T не має циклів і має $n - 1$ ребро.
2. T – зв'язний граф і має $n - 1$ ребро.
3. T – зв'язний граф і кожне його ребро є мостом.
4. Будь-які дві вершини графа T з'єднані тільки одним простим ланцюгом.

5. T не має циклів, але додавання будь-якого нового ребра до T сприяє виникненню тільки одного циклу.

Визначення. Ліс – незв'язний n – граф без циклів.

1. Зв'язні компоненти лісу є деревами.

2. Будь-яка частина лісу або дерева також не має циклів.

3. Будь-яка частина лісу є лісом або деревом.

Приклад лісу показаний на рис. 3.92.

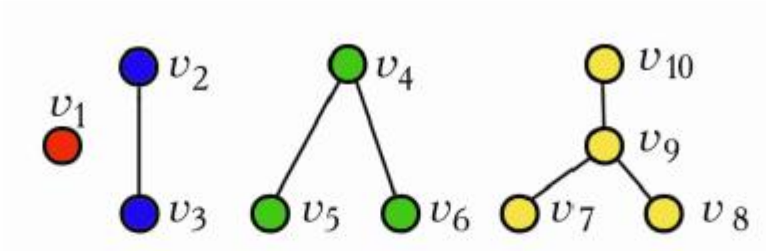


Рис. 3.92. Ліс із чотирьох дерев

Теорема 3.5. Нехай ліс G містить n вершин і k компонентів. Тоді ліс G має $n - k$ ребер.

Доведення. З умови 2 теореми 3.4 випливає, що кожний компонент G_i має $(n_i - 1)$ – ребро. Але тоді число ребер у G дорівнює $(n_1 - 1) + (n_2 - 1) + \ldots + (n_k - 1) = n_1 + n_2 + \ldots + n_k - k = n - k$, що і потрібно довести.

Теорема 3.6. Теорема Келі. Число різних дерев, які можна побудувати на n вершинах, дорівнює n^{n-2}.

Процедури побудови остовного дерева та лісу

Процедура побудови остовного дерева

1. Видалення із зв'язного графа G одного ребра, що належить деякому циклу, не порушує зв'язності графа G.

2. Застосуємо процедуру видалення ребра до одного із циклів у графі G.

3. Будемо повторювати видалення доти, доки в G не залишиться жодного циклу.

4. У підсумку одержимо дерево, що містить усі вершини графа G. Це дерево називають **остовним деревом графа** G.

Приклад побудови остовного дерева шляхом почергового видалення ребер показано на рис. 3.93.

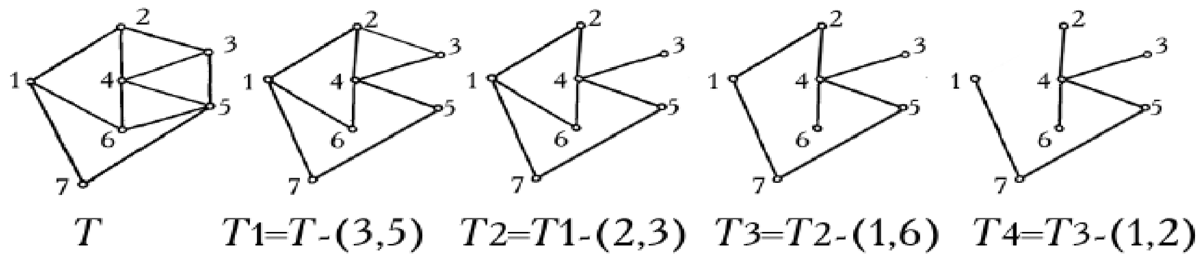


Рис. 3.93. Приклад побудови остовного дерева

Процедура побудови лісу

Нехай *G* є графом з *n* вершинами, *m* ребрами і *k* компонентами.

1. Застосуємо процедуру видалення ребра до одного із циклів у кожному компоненті зв'язності графа *G*.

2. Будемо повторювати видалення доти, доки в кожному компоненті *G* не залишиться жодного циклу.

3. У підсумку одержимо граф, який називають **остовним лісом**.

4. Число ребер, які при цьому видаляються, називають **цикломатичним числом**, або **циклічним рангом графа**, *G* і позначають *C*(*G*). Таким чином, цикломатичне число є мірою зв'язності графа.

Приклад побудови остовного лісу шляхом почергового видалення ребер показано на рис. 3.94.

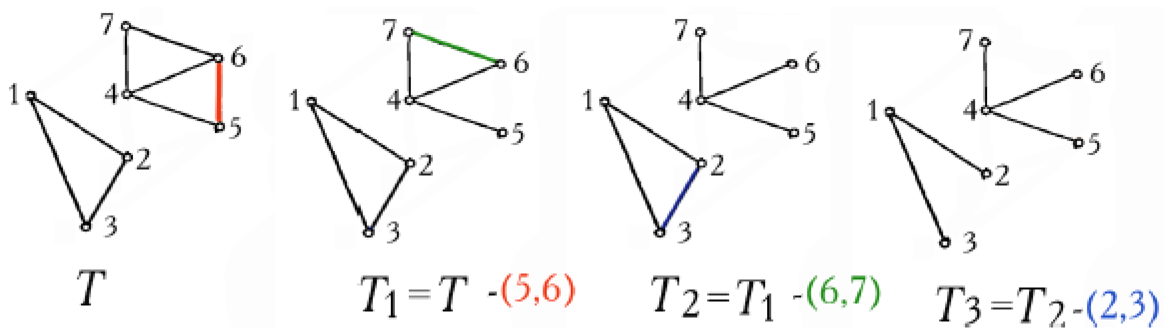


Рис. 3.94. Приклад побудови остовного лісу: *C* (*G*) = 3

Властивості циклічного рангу

1. **Циклічний ранг дерева** дорівнює нулю.

2. **Циклічний ранг циклічного графа** дорівнює одиниці.

Визначення. Циклічний граф – зв'язний регулярний граф степеня 2, єдиний компонент зв'язності циклічного графа є простим циклом (рис. 3.95).

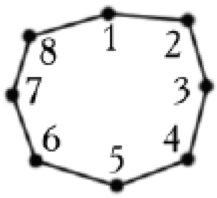


Рис. 3.95. Циклічний граф

Визначення. Остовне дерево графа – це дерево, що містить усі вершини графа.

Визначення. Остовним лісом називають незв'язний граф, що складається з компонентів, які є остовними деревами.

Теорема 3.7. Нехай *T* – остовний ліс графа *G*. Граф G', отриманий із графа *G* шляхом видалення всіх ребер графа *T*, називають **доповненням** остовного лісу *T* графа *G* (рис. 3.96).

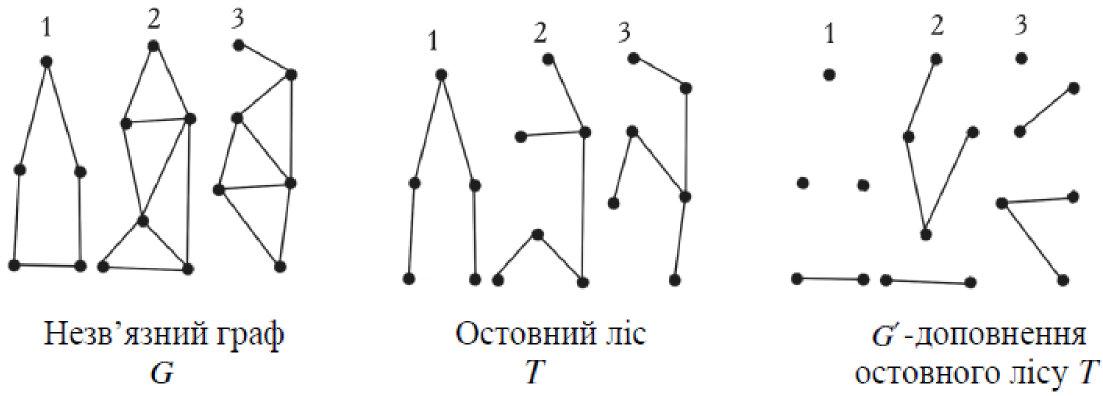


Рис. 3.96. Побудова доповнення остовного лісу G'

Фундаментальна система циклів графа

Нехай *T* – остовний ліс графа *G*. Якщо додати до *T* будь-яке ребро графа *G*, що не входить до нього, то за п. 5 теореми 3.4 одержимо єдиний цикл.

Визначення. Множину всіх циклів, які одержують шляхом додавання окремо кожного ребра з *G*, що не входить до *T*, називають фундаментальною системою циклів, асоційованою з *T*.

Цикли цієї фундаментальної системи будуть різними, але їх кількість дорівнює циклічному рангу графа *G*.

На рис. 3.97 графи показують фундаментальну систему циклів.

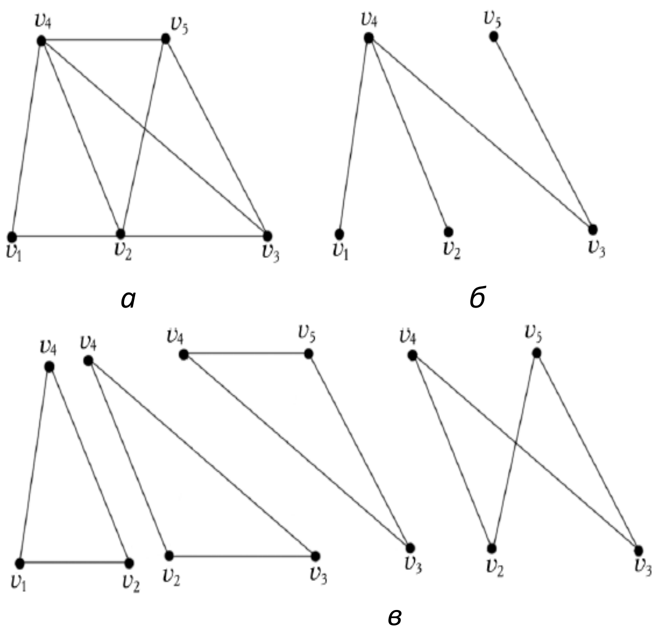


Рис. 3.97. Граф та його фундаментальні цикли: *а* – граф *G*; *б* – остовне дерево графа *G*; *в* – фундаментальна система циклів, асоційована з остовним деревом графа *G*

Теорема 3.8. Нехай $G = (V, E)$ – довільний скінченний граф. Число ребер графа *G*, які потрібно вилучити для одержання остовного лісу *T*, не залежить від порядку їх видалення і дорівнює $C(G) = E - V + k$, де *k* – число компонентів зв’язності графа *G*.

Наслідок 3.8.1. Граф *G* є остовним лісом тоді й тільки тоді, коли $C(G) = 0$.

Наслідок 3.8.2. Граф *G* має єдиний цикл тоді й тільки тоді, коли $C(G) = 1$.

Наслідок 3.8.3. Граф G, у якого число ребер перевищує число вершин, має цикл.

Наслідок 3.8.4. Будь-яке дерево порядку $n \geq 2$ має принаймні дві кінцеві вершини (рис. 3.98).

Рис. 3.98. Мінімальний граф із двома кінцевими вершинами

Остовні дерева. Пошук у глибину

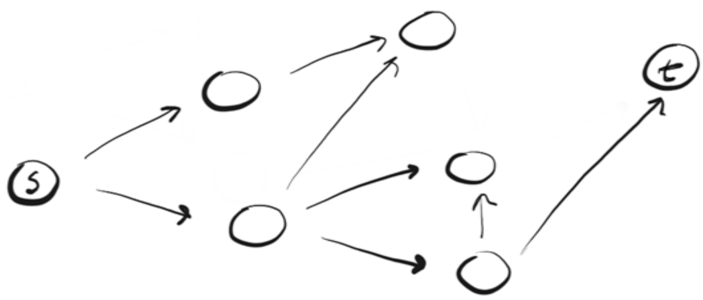
Методами пошуку остовного дерева називають алгоритми обходу вершин графа, за якого кожна вершина отримує унікальний порядковий номер.

Цей метод називають **пошуком у глибину**, або DFS-методом від англійського Depth First Search.

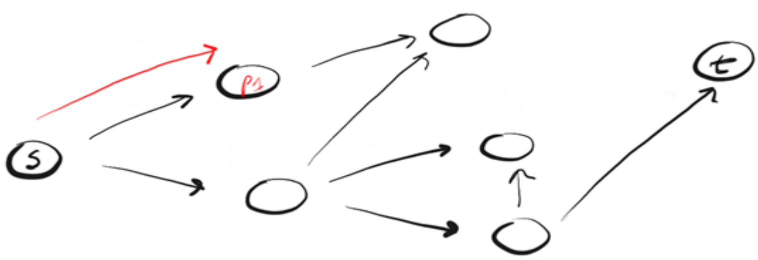
Нехай $G = (V, E)$ – простий зв'язний граф, усі вершини якого позначені попарно різними символами. У процесі пошуку вершинам надають номери (DFS-номери), які для вершини x позначають DFS(x). Пошук у глибину використовує для збереження множин структуру даних стек.

DFS слідує концепції «поринай глибше, головою вперед» (godeep, headfirst). Ідея полягає в тому, що ми рухаємося від початкової вершини (точки, місця) в заданому напрямку (за заданим шляхом) доти, доки не досягнемо кінця шляху або пункту призначення (шуканої вершини). Якщо ми досягли кінця шляху, але він не є пунктом призначення, то ми повертаємося назад (до точки розгалуження шляхів) і йдемо іншим маршрутом.

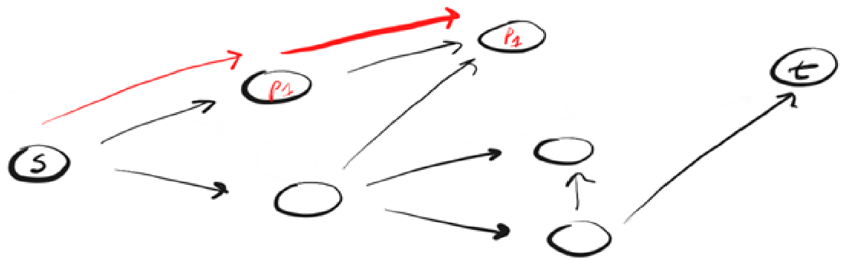
Давайте розглянемо приклад. Нехай у нас є орієнтований граф, який має такий вигляд:



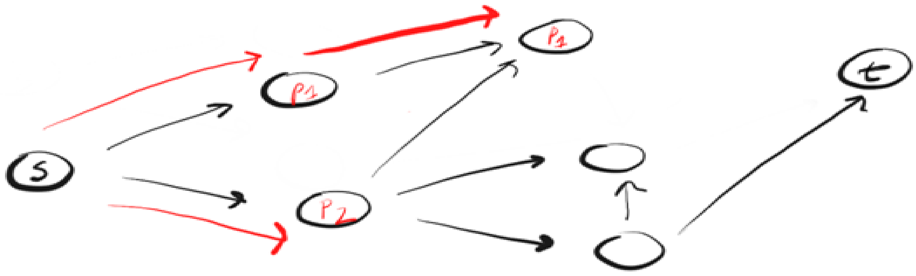
Ми перебуваємо в точці s, і нам потрібно знайти вершину t. Застосовуючи DFS, досліджуємо один із можливих шляхів, рухаємося по ньому до кінця і, якщо не знайшли t, повертаємося й досліджуємо інший шлях. Ось який вигляд має процес:



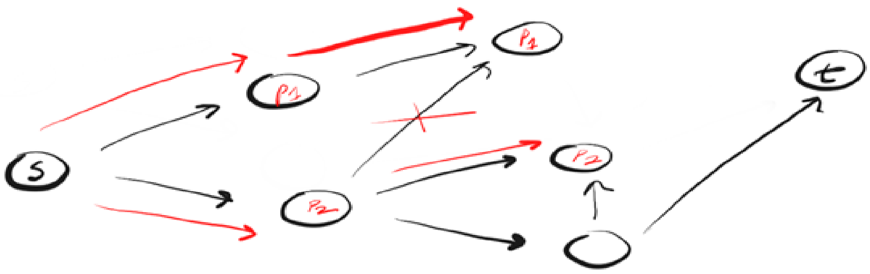
Тут ми рухаємося по маршруту (p1) до найближчої вершини й бачимо, що це не кінець шляху. Тому ми переходимо до наступної вершини.



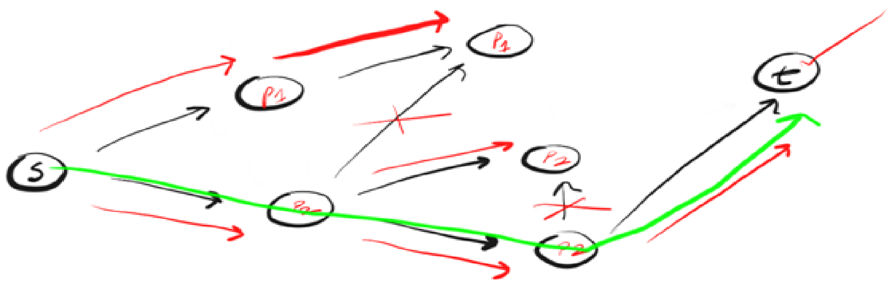
Ми досягли кінця p1, але не знайшли t, тому повертаємося в s і рухаємося іншим маршрутом.



Досягнувши найближчої доточки s вершини шляху p2, ми бачимо три можливі напрямки для подальшого руху. Оскільки вершину, де закінчувався перший напрямок, ми вже відвідували, то рухаємося другим.



Ми знову досягли кінця шляху, але не знайшли t, тому повертаємося назад. Слідуємо за третім шляхом і нарешті досягаємо шуканої вершини t.



Так працює DFS. Рухаємося за заданим шляхом до кінця. Якщо кінець шляху – це шукана вершина, ми закінчили. Якщо ні, повертаємося назад і рухаємося іншим шляхом доти, доки не дослідимо всі варіанти.

Ми слідуємо за цим алгоритмом, застосовуючи його до кожної відвідуваної вершини.

Потреба в багаторазовому повторенні процедури вказує на потребу в застосуванні рекурсії для реалізації алгоритму.

Примітка. Цей спеціальний DFS-алгоритм дає змогу перевірити, чи можливо дістатися з одного місця в інше. DFS може застосовуватися з різною метою. Від цієї мети залежить те, як вигляд матиме сам алгоритм. Проте загальна концепція має саме такий вигляд.

Ось JavaScript-код:

```
// за умови, що ми маємо справу з суміжним списком
// наприклад, таким: adj = {A: [B, C], B: [D, F], ...}
Function dfs (adj, v, t) {
    // adj – суміжний список
    // v – відвіданий вузол (вершина)
    // t – пункт призначення
    // це загальні випадки
    // або досягли пункту призначення, або вже відвідували вузол
    If (v === t) return true
    if (v. visited) return false
    // помічаємо вузол як відвіданий
    v.visited = true
    // досліджуємо всіх сусідів (найближчі сусідні вершини) v
    For (let neighbor of adj[v]) {
        // якщо сусід не відвідувався
        If (! neighbor. visited) {i
            // рухаємося шляхом і перевіряємо, чи не досягли ми пункту призначення
            let reached = dfs (adj, neighbor, t)
            // повертаємо true, якщо досягли
            if (reached) return true
        }
    }
    // якщо від v до t дістатися неможливо
    return false
}
```

Алгоритм пошуку остовного дерева в глибину

Крок 1. Почати з довільної вершини v_i. Покласти DFS $(v_i) = 1$. Включити вершину v_i *у* стек.

Крок 2. Розглянути вершину, яка розташована у вершині стека: нехай це буде вершині *x.* Якщо всі ребра, що інцидентні вершині *x*, позначені, то перейти до кроку 4, інакше – до кроку 3.

Крок 3. Нехай $\{x, y\}$ – непозначене ребро. Якщо DFS (y) вже визначений, то ребро $\{x, y\}$ позначити штриховою лінією і перейти до кроку 2. Якщо DFS (y) не визначений, то ребро позначити потовщеною лінією, визначити DFS (y) як черговий DFS-номер, включити цю вершину в стек і перейти до кроку 2.

Крок 4. Виключити вершину зі стека. Якщо стек порожній, то зупинитись, інакше – перейти до кроку 2.

- Для однозначності вибору номерів доцільно домовитися, що аналіз вершин, суміжних із вершиною, яка вже отримала DFS-номер, здійснюють за зростанням їх порядкового номера (або в алфавітному порядку).
- Динаміку роботи алгоритму зручно відобразити за допомогою таблиці з трьома стовпцями: вершина, DFS-номер, вміст стека. Цю таблицю називають **протоколом обходу графа** пошуком у глибину.

Аналіз DFS -алгоритму

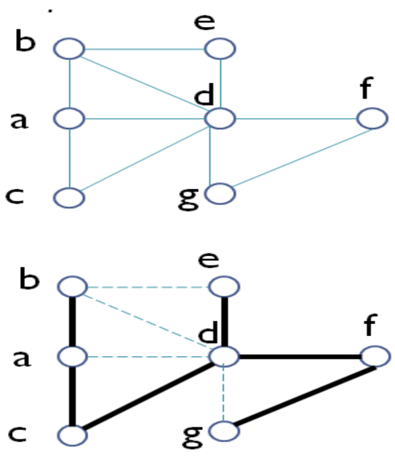
Давайте проаналізуємо цей алгоритм. Оскільки ми обходимо кожного «сусіда» кожного вузла, ігноруючи тих, яких відвідували раніше, ми маємо час виконання, що дорівнює O ($V + E$). Коротке пояснення того, що означає $V + E$:

V — загальна кількість вершин. E — загальна кількість граней (ребер). Може здатися, що правильніше застосувати $V*E$, однак давайте подумаємо, що означає $V*E$. $V*E$ означає, що стосовно кожної вершини ми повинні дослідити всі грані графа безвідносно приналежності цих граней конкретній вершині.

З іншого боку, $V + E$ означає, що для кожної вершини ми оцінюємо лише грані, що примикають до неї. Повертаючись до прикладу, зазначимо, що кожна вершина має визначену кількість граней, і в гіршому випадку ми обійдемо всі вершини (O(V)) і дослідимо всі грані (O(E)). Ми маємо V вершин і E граней, тому отримаємо $V + E$.

Далі, оскільки ми застосували рекурсію для обходу кожної вершини, це означає, що застосовується стек (безкінечна рекурсія приводить до помилки переповнення).

Приклад 3.33.



Протокол

Вершина	DFS-номер	Вміст стеку
b	1	b
a	2	ba
c	3	bac
d	4	bacd
e	5	bacde
-	-	bacd
f	6	bacdf
g	7	bacdfg
-	-	bacdf
-	-	bacd
-	-	bac
-	-	ba
-	-	b

Остовні дерева. Пошук у ширину

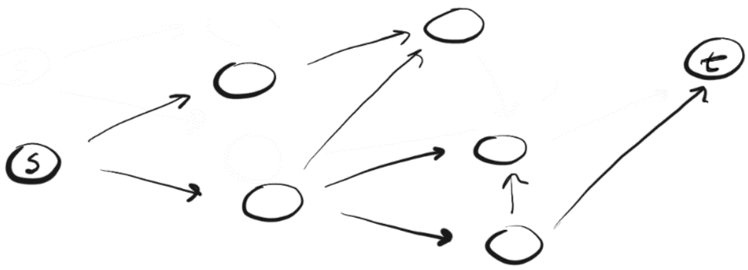
Цей метод називають **пошуком у ширину**, або BFS-методом – від англ. Breadth First Search.

У процесі пошуку вершинам надають номери (BFS-номери), які для вершини x позначають BFS (x). Пошук у ширину використовує для збереження множин структуру даних – черга.

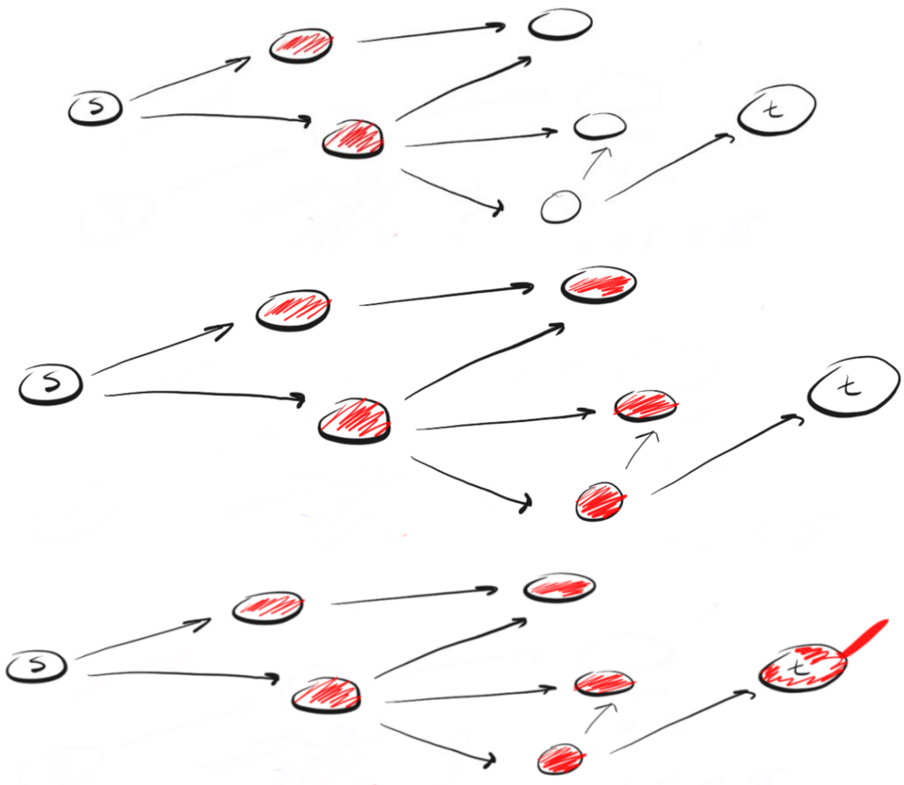
Для того щоб результат виконання алгоритму був однозначним, вершини суміжні з вершиною x, аналізуємо за зростанням їхніх порядкових номерів (або в алфавітному порядку).

Динаміку роботи алгоритму відображаємо **протоколом обходу графа пошуком у ширину**.

BFS слідує концепції «розширюйся, піднімаючись на висоту пташиного польоту» (gowide, bird'seye-view). Замість того щоб рухатися заданим шляхом до кінця, BFS передбачає рух вперед по одному сусіду за раз. Це означає таке:



Замість слідування шляхом BFS передбачає відвідування найближчих до s сусідів за один крок, потім відвідування сусідів і так далі доти, доки не буде досягнута t.



Чим DFS відрізняється від BFS? Мені здається що, DFS йде напролом, а BFS не поспішає, а досліджує все в межах одного кроку.

Далі виникає питання: як дізнатися, яких сусідів слід відвідати першими?

Для цього ми можемо використати концепцію «першим зайшов, першим вийшов» (first-in-first-out, FIFO) із черги (queue). Ми заносимо в чергу спочатку найближчу до нас вершину, потім її невідвіданих сусідів і продовжуємо цей процес, доки черги не залишиться або доки ми не знайдемо шукану вершину.

Ось JavaScript-код:

```
// за умовою, що ми маємо справу із суміжним списком
// наприклад, таким: adj = {A:[B, C, D], B:[E, F], ... }
functionbfs (adj, s, t) {
    // adj – суміжний список
    // s – початкова вершина
    // t – пункт призначення
    // ініціалізуємо чергу
```

```
letqueue = []
// додаємо s в чергу
queue. push(s)
// помічаємо s як відвідану вершину, щоб не повторити
додавання в чергу
s. visited = true
while (queue. length > 0) {
    // видаляємо перший (верхній) елемент із черги
    let v = queue. shift ()
    // abj[v] – сусіди v
    for (let neighbor of adj[v]) {
        // якщо сусід не відвідувався
        if (! neighbor. visited) {
            // додаємо його в чергу
            queue. push (neighbor)
            // помічаємо вершину як відвідану
            neighbor. visited = true
            // якщо сусід є пунктом призначення, ми
перемогли
            if (neighbor === t) return true
        }
    }
}
// якщо t не знайдена, значить пункт призначення досягти
неможливо
returnfalse
}
```

Алгоритм пошуку остовного дерева в ширину

- Крок 1. Почати з довільної вершини v_i. Покласти BFS(v_i) = 1. Включити вершину v_i в чергу.
- Крок 2. Розглянути вершину, яка розташована на початку черги: нехай це буде вершина *x*. Якщо для всіх вершин, суміжних із вершиною *x*, уже визначені BFS-номери, то перейти до кроку 4, інакше – до кроку 3.

- Крок 3. Нехай $\{x, y\}$ – ребро, у якому номер BFS (y) не визначений. Позначити це ребро потовщеною суцільною лінією, визначити BFS (y) як черговий BFS-номер, включити вершину до черги й перейти до кроку 2.
- Крок 4. Виключити вершину x із черги. Якщо черга порожня, то зупинитись, інакше – перейти до кроку 2.

Аналіз BFS

Може здатися, що BFS працює повільніше. Однак, якщо уважно придивитися до візуалізацій, можна побачити, що вони мають однаковий час виконання.

Черга передбачає обробку кожної вершини перед досягненням пункту призначення. Це означає, що в гіршому випадку BFS досліджує всі вершини та грані.

Незважаючи на те що BFS може здаватися повільнішим, але він скоріший, оскільки під час роботи з великими графами виявляється, що DFS витрачає багато часу на слідування шляхами, які в кінцевому підсумку виявляються хибними. BFS часто застосовують для знаходження найкоротшого шляху між двома вершинами.

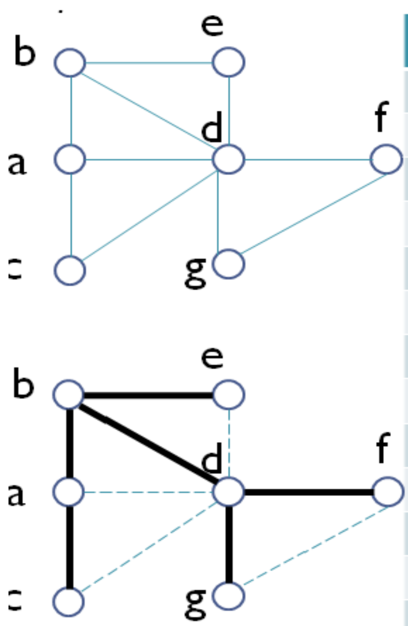
Таким чином, виконання BFS також складає O (V + E), а оскільки ми застосовуємо чергу, що вміщує всі вершини, його просторова складність становить O (V).

Приклад 3.34.

Обчислювальна складність обох алгоритмів обходу однакова й у випадку зображення графа списками суміжності становить O ($|V| + |E|$).

Аналоги з реального життя

Якщо наводити приклади з реального життя, то ось як я представляю собі роботу DFS і BFS. Коли я думаю про DFS, то представляю собі мишу в лабіринті в пошуку їжі. Для того щоб потрапити до їжі, миша змушена багато разів потрапляти в тупик, повертатися й рухатися в іншому напрямку, і так, доки вона не знайде вихід із лабіринту або їжу.



Протокол

Вершина	BFS-номер	Вміст черги
b	1	b
a	2	ba
d	3	bad
e	4	bade
-	-	ade
c	5	adec
-	-	dec
f	6	decf
g	7	decfg
-	-	ecfg
-	-	cfg
-	-	fg
-	-	g
-	-	∅

Рис. 3.99. Ілюстрація DFS-алгоритму

Спрощена версія має такий вигляд:

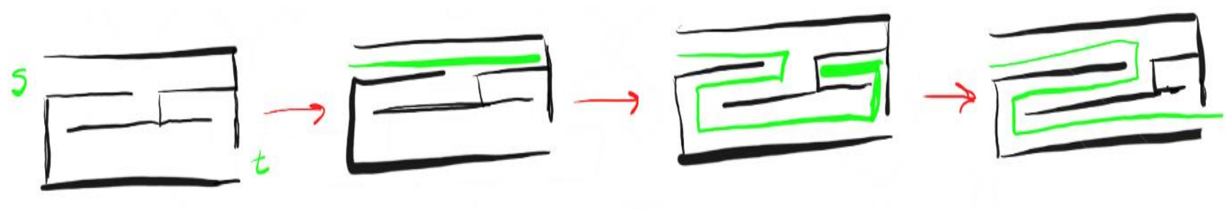


Рис. 3.100. Спрощена версія DFS-алгоритму

Зі свого боку, коли я думаю про BFS, то представляю собі кола на воді. Падіння каменя у воду приводить до утворення кіл по всіх напрямках від центра.

Рис. 3.101. Ілюстрація BFS-алгоритму

Спрощена версія має такий вигляд:

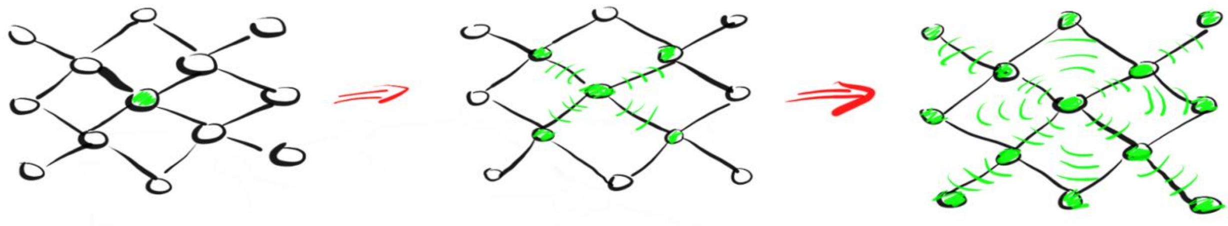


Рис. 3.102. Спрощена версія BFS-алгоритму

Висновки

- Пошук у глибину і пошук у ширину використовують для обходу графа.
- DFS рухається по гранях (ребрах) туди і назад, а BFS розповсюджується по сусідах у пошуку призначення.
- DFS використовує стек, а BFS — чергу.
- Час виконання обох становить O (V + E), а просторова складність – O (V).
- Ці алгоритми мають різну філософію, але однаково важливі для роботи з графами.

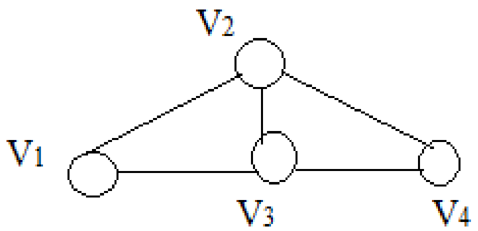
Алгоритм пошуку максимальної кількості остовних дерев

Цей алгоритм застосовується для будь-якого зв'язного графа *G* (*V*, *E*) з поміченими вершинами, у якому потрібно знайти всі можливі остовні дерева.

Кроки алгоритму пошуку максимальної кількості остовних дерев

- Знайти степені вершин цього зв'язного графа.
- Побудувати матрицю степенів вершин зв'язного графа.
- Побудувати матрицю суміжності зв'язного графа.
- Знайти різницю між матрицями степенів і суміжності зв'язного графа *G* (*V*, *E*).
- Знайти будь-яке з алгебраїчних доповнень отриманої матриці різниці на кроці 4 алгоритму, значення якого є кількісним значенням остовних дерев для графа *G* (*V*, *E*).

Приклад 3.35. Використовуючи розглянутий алгоритм, знайти максимальну кількість остовних дерев для зв'язного графа *G* (*V*, *E*), наведеного на рисунку:



$$A=\begin{pmatrix}0&1&1&0\\1&0&1&1\\1&1&0&1\\0&1&1&0\end{pmatrix}\qquad C=\begin{pmatrix}2&0&0&0\\0&3&0&0\\0&0&3&0\\0&0&0&2\end{pmatrix}$$

Розв'язок.

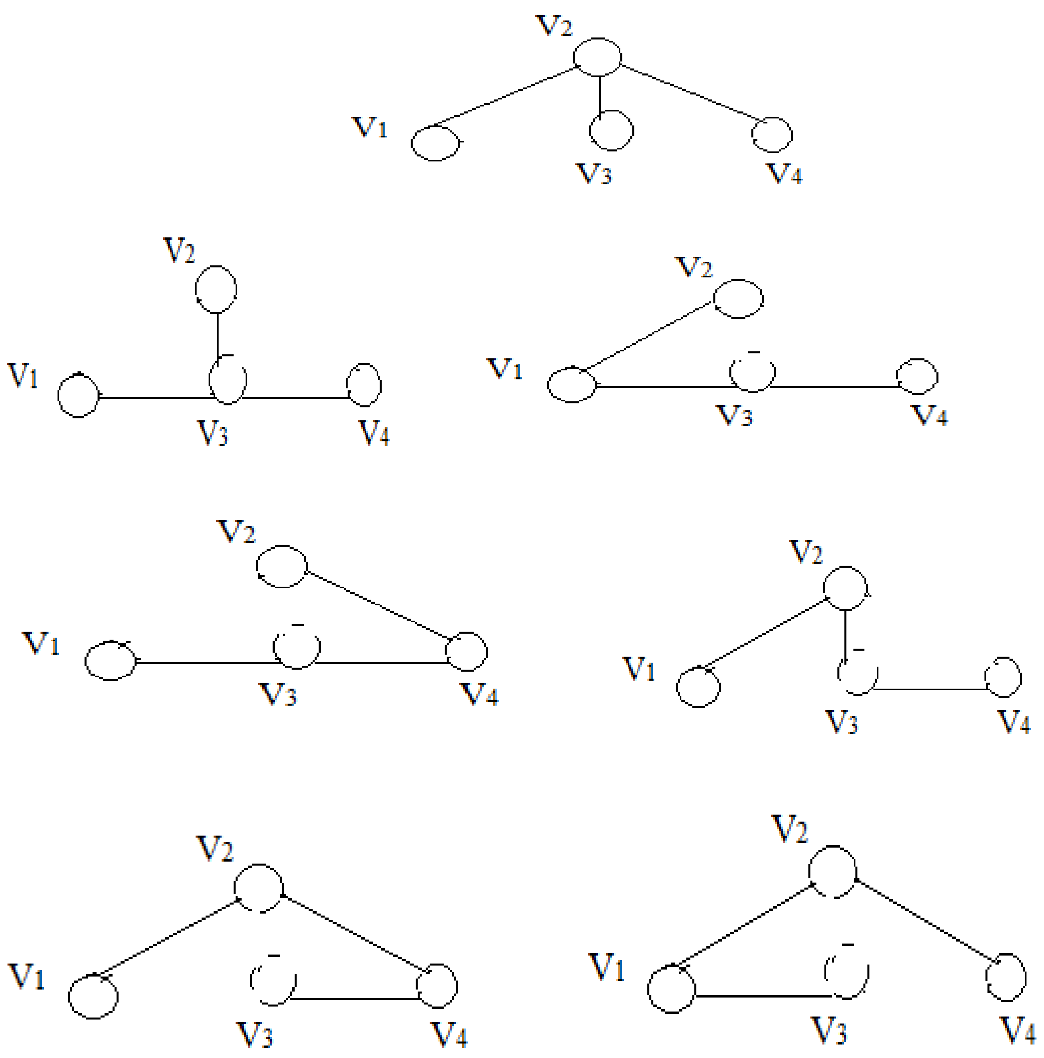
Відповідно до кроку 1 алгоритму, знайдемо ступені вершин графа *G* (*V*, *E*), які дорівнюють: deg(V_1) = deg(V_4) =2; deg(V_2) = deg(V_3). Згідно з кроком 2 алгоритму будуємо степенну вершинну матрицю С і кроком 3 – матрицю суміжності *A* графа *G* (*V*, *E*). Користуючись кроком 4 алгоритму, знаходимо різницю між двома матрицями *C* і *A*, унаслідок чого отримуємо третю матрицю *K*:

$$K_{11}=\det\begin{pmatrix}3&-1&-1\\-1&3&-1\\-1&-1&2\end{pmatrix}=3\cdot\det\begin{pmatrix}3&-1\\-1&2\end{pmatrix}-(-1)\begin{pmatrix}-1&-1\\-1&2\end{pmatrix}+(-1)\begin{pmatrix}-1&3\\-1&-1\end{pmatrix}=3\cdot5-3-4=8$$

Використовуючи крок 5 алгоритму, знаходимо алгебраїчне доповнення до матриці К, наприклад К11, яке дорівнює

$$K = C - A = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix}$$

Виходячи із значення цього доповнення, можна сказати, що граф G(V,E), який наведений на рисунку, має вісім остовних дерев:



<u>Обов'язкові завдання</u>

1. Сформулюйте задачу обходу графа та базовий алгоритм обходу. Яку вершину називають закритою вершиною?

2. Опишіть принципи, на яких ґрунтується алгоритм обходу графа в глибину. Яка умова зупинки алгоритму обходу графа в глибину?

3. Написати програму та вивести на друк проміжні результати обходу графа (рис. 3.103) в глибину.

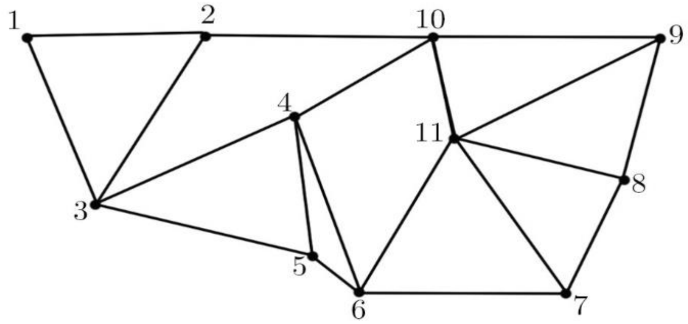


Рис. 3.103. Граф до задач 3 та 4

4. Створити програму обходу графа в ширину. Застосувати цю програму до графа, показаного на рис. 3.103.

Основне дерево найменшої ваги

Нехай кожному ребру графа $G = (V, E)$ поставлена у відповідність деяка вага $d_i = d(e_i)$, $e_i \in E$, $i = 1,2,\ldots,|E|$.

Потрібно в графі G знайти остовне дерево, сума ваг d_i у якому найменша.

$S = \min \sum_{e \in E_i} (d(e_i))$. Число d_i називають **вагою ребра** e_i , а сам граф G – таким, що має **вагу (зваженим)**. Отже, завдання полягає в тому, щоб знайти остовне дерево з найменшою вагою.

Практичні застосування задачі пошуку остовного дерева з мінімальною вагою.

Ця задача виникає під час проєктування доріг, ліній електропередач, трубопроводів та ін., коли потрібно з’єднати задані точки деякою системою зв’язку так, щоб загальна довжина ліній зв’язку була мінімальною. На рис. 3.91 показані об’єкти, у проєктуванні яких застосовують розв’язування задачі пошуку остовного дерева з мінімальною вагою.

Рис. 3.104. Приклади об’єктів, для яких потрібна мінімізація параметрів

Розглянемо кілька методів розв’язування цієї задачі.

Розв’язування, що випливає з теореми Келі

1. Розглянемо всі можливі остовні дерева повного графа G з n вершинами. Згідно з теоремою Келі число різних дерев, які можна побудувати на n вершинах, дорівнює n^{n-2}.

2. Під час вибору кожного дерева визначимо суму його ваг.

3. Застосувавши сортування за величиною суми ваг, на початку списку одержимо ті остовні дерева, які мають мінімальну суму ваг. Але оскільки число n^{n-2} навіть за невеликого n буде дуже великим, то такий шлях розв’язування цієї задачі досить трудомісткий. Тому актуальним є використання більш ефективних алгоритмів.

Попередні зауваження

1. **Порядок графа** – число, яке дорівнює кількості вершин графа.

2. **Порожній граф** – граф, що не містить ребер або регулярний граф степеня 0.

Постановка задачі. Задано зв’язний неорієнтований граф $G = (V, E)$, де V – множина вершин, а E – множина ребер і для кожного ребра $(u, v) \in E$ задано вагу $w(u, v)$. Потрібно знайти ациклічну підмножину $T \in E$, яка з’єднує всі вершини й загальна вага якої

$w(T) = \sum_{(u,v)\in T} w(u,v)$ мінімальна.

Задача пошуку дерева T називається задачею пошуку мінімального остовного дерева.

Ми розглянемо два алгоритми розв’язку цієї задачі – алгоритм Прима й алгоритм Крускала.

Алгоритм Прима

Ізольованою називається вершина, яка на деякому етапі побудови не зв'язана з іншими вершинами. *Фрагмент* – це підмножина вершин зв'язаних ребрами.

Ізольованим фрагментом називається фрагмент, який на цьому етапі побудови не зв'язаний з іншими вершинами або фрагментами.

Принципи побудови дерева мінімальної довжини:

- Довільна ізольована вершина з'єднується з найближчим сусідом – вершиною, яка розташована на найменшій відстані від цієї вершини.
- Довільний ізольований фрагмент з'єднується з найближчим сусідом найкоротшим ребром.

Алгоритм Прима

1. Побудова матриці суміжності ваг.

Якщо вершини u та v не з'єднані, то в матриці на перетині рядка u та стовпчика v ставиться нескінченість (∞). Діагональні елементи умовно приймаються рівними нескінчености (∞). Усі інші елементи матриці дорівнюють $w\,(u,\,v)$.

2. Визначення першого фрагмента.

За початкову вибирається довільна вершина. Згідно з принципу 1 для цієї вершини знаходимо найближчого сусіда. Для цього в матриці вибирається рядок відстаней від вибраної вершини до всіх інших і визначається вершина, до якої відстань найменша.

3. Розширення фрагмента.

Для розширення фрагмента порівнюються відстані від отриманого фрагменту до кожної ізольованої вершини. З усіх можливих з'єднань вибирається така ізольована вершина, відстань до якої найменша.

4. Закінчення.

Якщо всі ізольовані вершини приєднані, то мінімальне остовне дерево побудоване (роботу алгоритму завершено), якщо – ні, то перейти на крок 3.

Псевдокод алгоритму Прима:

// Вхідні дані: Зважений зв'язний граф $G = (V, E)$

//Вихідні дані: E_T, множина ребер, які утворюють мінімальне остовне дерево T

$V_T \leftarrow \{\, v_0 \,\}$ //Множина вершин остовного дерева T

$E_T \leftarrow \varnothing$ //Множина ребер остовного дерева T

for $i \leftarrow 1$ **to** $|V|-1$ **do**

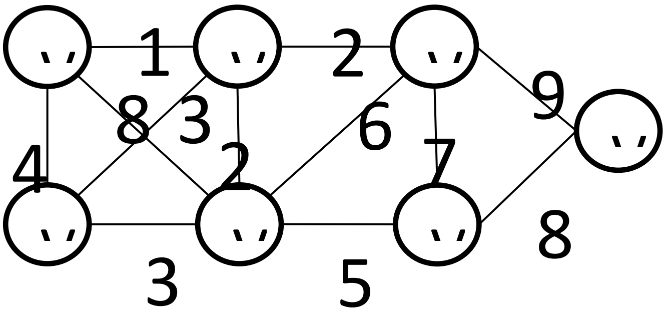
// Пошук ребра з мінімальною вагою $e^* = (u^*, v^*)$ серед всіх ребер (u, v) таких, що $v \in V_T$ та $u \in V - V_T$

$V_T \leftarrow V_T \cup \{u^*\}$

$E_T \leftarrow E_T \cup \{e^*\}$

return E_T

Приклад 3.35. Для заданого графа G побудувати остовне дерево мінімальної ваги, використовуючи алгоритм Прима.

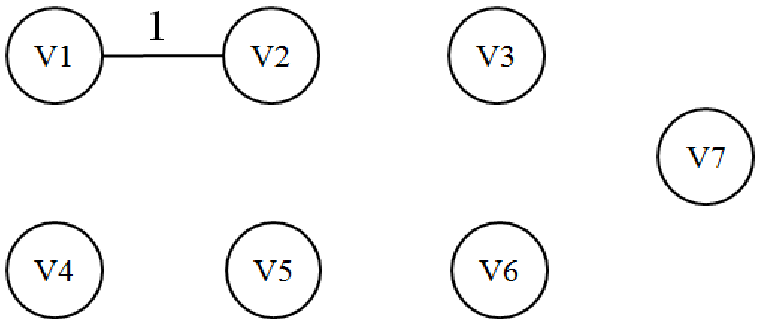


Побудуємо матрицю суміжності ваг W:

	V_1	V_2	V_3	V_4	V_5	V_6	V_7
V_1	∞	1	∞	4	8	∞	∞
V_2	1	∞	2	3	2	∞	∞
V_3	∞	2	∞	∞	6	7	9
V_4	4	3	∞	∞	3	∞	∞
V_5	8	2	6	3	∞	5	∞
V_6	∞	∞	7	∞	5	∞	8
V_7	∞	∞	9	∞	∞	8	∞

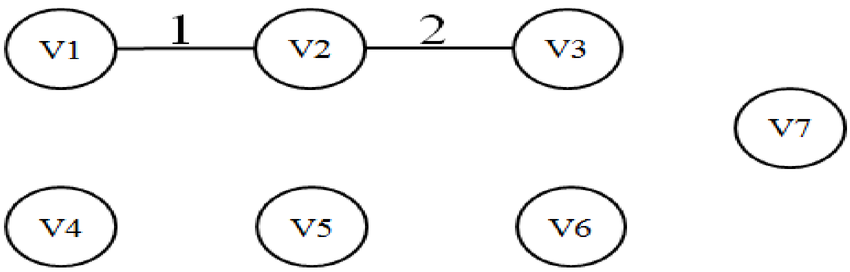
За початкову вибираємо довільну вершину, нехай це буде вершина V_1, і для неї шукаємо найближчого сусіда, тобто вершину відстань до якої найменша:

	v_2	v_3	v_4	v_5	v_6	v_7
v_1	1	∞	4	8	∞	∞



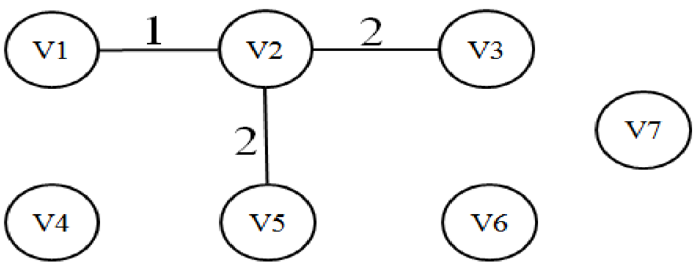
Розширюємо фрагмент

	v_3	v_4	v_5	v_6	v_7
v_1	∞	4	8	∞	∞
v_2	2	3	2	∞	∞



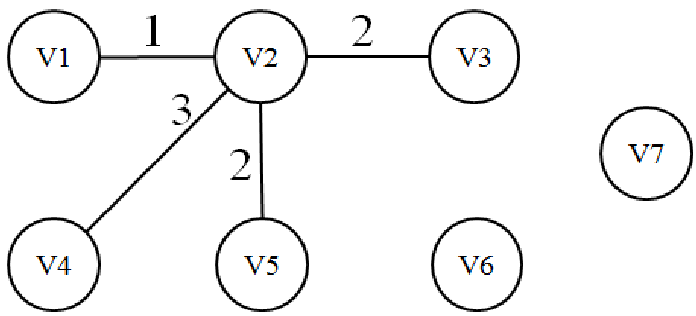
Розширюємо фрагмент

	v_4	v_5	v_6	v_7
v_1	4	8	∞	∞
v_2	3	2	∞	∞
v_3	∞	6	7	9



Розширюємо фрагмент

	v_4	v_6	v_7
v_1	4	∞	∞
v_2	3	∞	∞
v_3	∞	7	9
v_5	3	5	∞



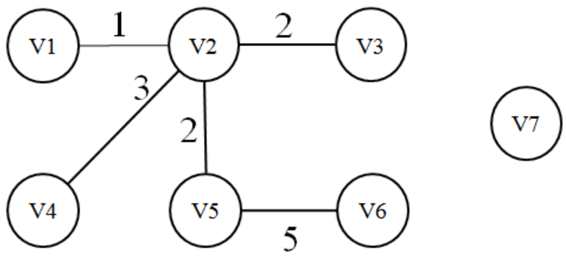
Властивості алгоритму Прима

1. Алгоритм використовується для зважених **неорієнтованих зв'язних** графів.

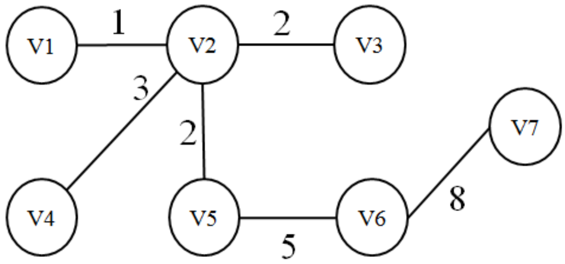
2. Обчислювальна складність алгоритму Прима $O(n^2)$, де n – кількість вершин графа. Якщо значення n достатньо **велике**, то використовувати цей алгоритм **не раціонально**.

3. Якщо кількість ребер e **значно меншою за** $\boldsymbol{n^2}$, то алгоритм Крускала кращий, але якщо e ***на*****ближена до** $\boldsymbol{n^2}$, то рекомендують застосовувати алгоритм Прима.

	V_6	V_7
V_1	∞	∞
V_2	∞	∞
V_3	7	9
V_4	∞	∞
V_5	5	∞



	V_7
V_1	∞
V_2	∞
V_3	9
V_4	∞
V_5	∞
V_6	8



$$w(T) = \sum_{(u,v)\in T} w(u,v) = 1+2+2+3+5+8 = 21$$

Алгоритм Крускала

Алгоритм Крускала будує мінімальне остовне дерево як послідовність підграфів, котрі завжди ациклічні, але на проміжних стадіях не завжди зв'язні.

Алгоритм Крускала

1. Відсортувати ребра графа G в зростаючому порядку.

2. Вибрати ребро e_1, яке має в графі G найменшу вагу.

3. На кожному кроці вибирати ребро (відмінне від попередніх) з найменшою вагою і таке, що не утворює простих циклів із попередніми ребрами. Отримане дерево T з множиною ребер

$E_T = \{e_1, e_2, e_3, \dots, e_{n-1}\}$ є мінімальним остовним підграфом графа G.

Псевдокод алгоритму Крускала:

// Вхідні дані: Зважений зв'язний граф $G = (V, E)$

//Вихідні дані: E_T, множина ребер, які утворюють мінімальне остовне дерево T

Сортування множини E за зростанням ваг ребер $w(e_1) \le \dots \le w(e_n)$, $n = |E|$

$E_T \leftarrow \varnothing$ //Множина ребер остовного дерева T

$ecounter \leftarrow 0$//розмір дерева T

$k \leftarrow 0$ //кількість оброблених ребер

while $ecounter < |V| - 1$ **do**

$k \leftarrow k + 1$

if $E_T \cup \{e_k\}$ – ациклічний граф **then**

$E_T \leftarrow E_T \cup \{ e_k \}$; $ecounter \leftarrow ecounter + 1$

return E_T

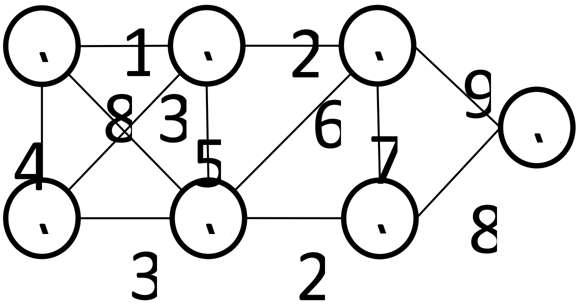
Одна з можливих реалізацій алгоритму Крускала

Початок. Упорядкувати множину ребер за зростанням ваг: e_1, e_2, e_3, …, e_n. Виконати розбиття множини вершин.

Ітерація. Вибрати таке чергове ребро з упорядкованої послідовності ребер, кінці якого містяться в різних множинах розбиття. Нехай вибрано ребро $e_i(u, v)$, тоді множини, що містять вершини u та v, об'єднуються в одну множину.

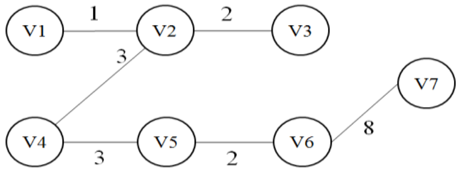
Закінчення. Роботу закінчити, коли буде вибране $(n - 1)$ ребро, при цьому всі підмножини розбиття об'єднуються в одну.

Приклад 3.36. Для заданого графа G побудувати остовне дерево мінімальної ваги, використовуючи алгоритм Крускала.



Ребра впорядковані за зростанням	Розбиття множини вершин	Вибір ребра у мінімальний остов T
	$\rho_0=\{\{v_1\},\{v_2\},\{v_3\},\{v_4\},\{v_5\},\{v_6\},\{v_7\}\}$	
$e_1(v_1, v_2) = 1$	$\rho_1=\{\{v_1,v_2\},\{v_3\},\{v_4\},\{v_5\},\{v_6\}, \{v_7\}\}$	$E_T = \{ e_1 \}$
$e_2(v_2, v_3) = 2$	$\rho_2=\{\{v_1,v_2,v_3\},\{v_4\},\{v_5\},\{v_6\}, \{v_7\}\}$	$E_T = \{ e_1, e_2\}$
$e_3(v_5, v_6) = 2$	$\rho_3=\{\{v_1,v_2,v_3\},\{v_4\},\{v_5,v_6\}, \{v_7\}\}$	$E_T = \{ e_1, e_2, e_3\}$
$e_4(v_2, v_4) = 3$	$\rho_4=\{\{v_1,v_2,v_3,v_4\},\{v_5,v_6\}, \{v_7\}\}$	$E_T = \{ e_1, e_2, e_3, e_4\}$
$e_5(v_4, v_5) = 3$	$\rho_5=\{\{v_1,v_2,v_3,v_4,v_5,v_6\}, \{v_7\}\}$	$E_T = \{ e_1, e_2, e_3, e_4, e_5\}$
$e_6(v_1, v_4) = 4$	ρ_5	$e_6 \notin E_T$
$e_7(v_2, v_5) = 5$	ρ_5	$e_7 \notin E_T$
$e_8(v_3, v_5) = 6$	ρ_5	$e_8 \notin E_T$
$e_9(v_3, v_6) = 7$	ρ_5	$e_9 \notin E_T$
$e_{10}(v_1, v_5) = 8$	ρ_5	$e_{10} \notin E_T$
$e_{11}(v_6, v_7) = 8$	$\rho_6=\{\{v_1,v_2,v_3,v_4,v_5,v_6,v_7\}\}$	$E_T = \{ e_1, e_2, e_3, e_4, e_5, e_{11}\}$
$e_{12}(v_3, v_7) = 9$	-	-

Із приєднанням ребра e_{11} робота алгоритму закінчується, оскільки вже приєднано n – 1 = 7–1 = 6 ребер і всі підмножини розбиття об'єдналися в одну $\rho_6 = \{\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}\}$. Остовне дерево мінімальної ваги T утворюють ребра e_1, e_2, e_3, e_4, e_5, e_{11}.



Властивості алгоритму Краскала

1. Алгоритм використовується для зважених неорієнтованих графів.

2. Алгоритм може бути використаний для побудови остовного лісу в багатокомпонентних незв'язних графах.

У цьому випадку **структури даних**, що описують кожну з компонент зв'язності, **повинні бути окремими**.

Обчислювальна складність алгоритму Краскала O (e × log$e$), де e – кількість ребер у цьому графі.

<u>Обов'язкові завдання</u>

1. Розглянемо ліс, який складається з трьох компонентів. Множина вершин лісу $V = \{v_1.v_2.v_3.v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$. Знайти потужність множини ребер лісу.

2. Дано множину вершин графа $V = \{v_1.v_2.v_3.v_4\}$. Обчислити кількість дерев, які можна побудувати на множині вершин V. Побудувати всі можливі дерева.

3. Побудувати остовний ліс і доповнення остовного лісу для незв'язного графа, показаного на рис. 3.105.

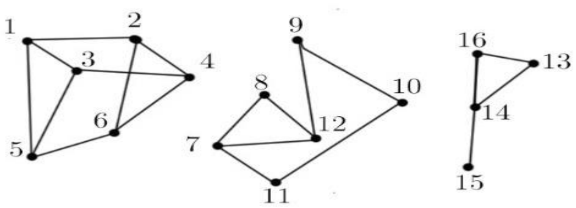


Рис. 3.105. Граф до задачі 3

4. Визначити кількість фундаментальних циклів у графі, показаному на рис. 3.106 і побудувати підграфи, кожен з яких містить тільки один фундаментальний цикл цього графа.

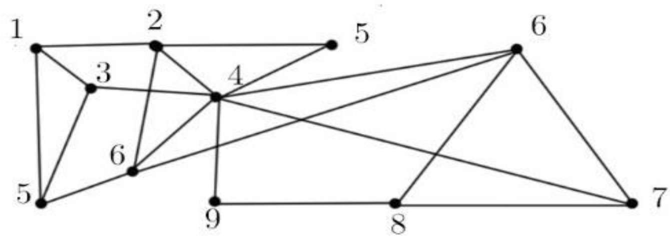


Рис. 3.106. Граф до задачі 4

5. Дано зв'язний неорієнтований зважений граф *G*, показаний на рис. 3.107. Побудувати послідовність створення остовного дерева мінімальної ваги за алгоритмом Крускала.

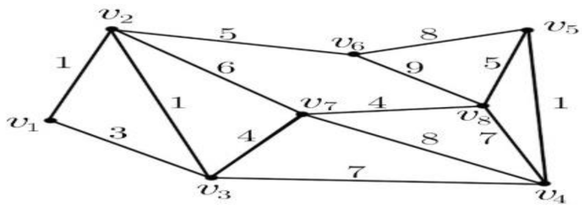


Рис. 3.107. Граф до задач 5 і 6

6. Побудувати остовне дерево мінімальної ваги за алгоритмом Прима, використовуючи граф, показаний на рис. 3.107.

3.9. Алгоритми пошуку найкоротших шляхів у графі

Задача про найкоротший шлях полягає у знаходженні найкоротшого шляху від заданої початкової вершини *a* до заданої вершини z.

Наступні дві задачі є безпосередніми узагальненнями сформульованої задачі про найкоротший шлях.

1. Для заданої початкової вершини знайти найкоротші шляхи від *a* до всіх інших вершин.

2. Знайти найкоротші шляхи між всіма парами вершин.

Розглянемо два алгоритми. Перший алгоритм розв'язує задачу номер один, другий – спеціально призначений для розв'язування задачі два.

Алгоритм Дейкстри

Найефективнішим алгоритмом знаходження довжини найкоротшого шляху від фіксованої вершини до будь-якої іншої є алгоритм, запропонований 1959 р. нідерландським математиком Е. Дейкстрою (E. Dijkstra).

Цей алгоритм застосовується лише у випадку, коли вага кожної дуги додатна.

Нехай G = (V, E) – орієнтований граф, $w(v_i, v_j)$ – вага дуги (v_i, v_j).

Пошук мінімального шляху здійснюється за допомогою присвоювання вершинам міток. Мітки є двох типів – тимчасові й постійні. Вершини з постійними мітками групують у множину М, яку називають **множиною позначених вершин**. Решта вершин має тимчасові мітки, і множину таких вершин позначають через T ($T = V\backslash M$).

Величина постійної мітки вершини $l(v)$ дорівнює довжині найкоротшого шляху від вершини a до вершини v. Якщо ж мітка тимчасова, то вона дорівнює довжині найкоротшого шляху, який проходить лише через вершини з постійними мітками.

Формальний опис алгоритму Дейкстри:

Крок 1. **Присвоювання початкових значень**. Виконати $l(a) = 0$ і вважати цю мітку постійною. Виконати $l(v) = \infty$ для всіх $v \neq a$ і вважати ці мітки тимчасовими. Виконати $x = a$, $M = \{a\}$.

Крок 2. **Оновлення міток.** Для кожної вершини $v \in V\backslash M$ замінити мітки:

$l(v) = \min\{l(v),\ l(x) + w(x, v)\}$, тобто оновлювати тимчасові мітки вершин, у які з вершини x іде дуга.

Крок 3. **Перетворення мітки у постійну.** Серед усіх вершин з тимчасовими мітками знайти вершину з мінімальною міткою, тобто знайти вершину v^* з умови

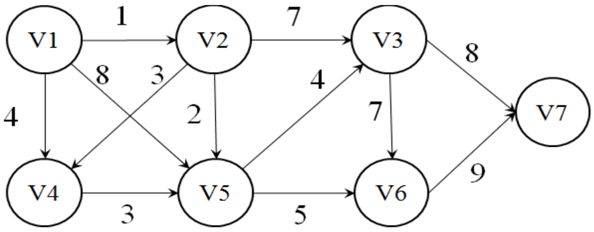
$l(v^*) = \min l(v),\ v \in T,\ T = V\backslash M.$

Крок 4. Вважати мітку вершини v^* постійною і покласти $M = M \cup v^*$, $x = v^*$.

Крок 5. (*а*) (Якщо потрібно знайти шлях від a до z.) Якщо $x = z$, то $l(z)$ – довжина найкоротшого шляху від a до z; зупинитись. Якщо $x \neq z$, то перейти до кроку 2.

(*б*) (Якщо потрібно знайти шлях від a до всіх інших вершин.) Якщо всі вершини отримали постійні мітки (включені у множину М), то ці мітки дають довжини найкоротших шляхів; зупинитись. Якщо деякі вершини мають тимчасові мітки, то перейти до кроку 2.

Приклад 3.37. Знайти найкоротший шлях від вершини v_1 до вершини v_7.



Початкові значення:

$M = \{V_1\}$, $T = \varnothing$, $l\,(V_1) = 0$, $l\,(v_2) = l\,(V_3) = \ldots\, l\,(V_7) = \infty$

Крок 1. $T_1 = \{v_2(1, V_1), v_4(4, V_1), v_5(8, V_1)\}$ – min V_2

$M_1 = \{V_1, V_2\ (1, V_1)\}$

Крок 2. $T_2 = \{v_4(4, v_1), v_5(8, v_1), v_4(4, v_2), v_5(3, v_2), v_3(8, v_2)\} = \{v_4(4, v_1),$ $V_5\ (3, V_2), V_3\ (8, V_2)\}$– min V_5

$M_2 = \{V_1, V_2\ (1, V_1), V_5\ (3, V_2)\}$

Крок 3. $T_3 = \{V_4\ (4, V_1), V_3\ (8, V_2), V_3\ (7, V_5), V_6\ (8, V_5)\} = \{V_4\ (4, V_1), V_3\ (7, V_5), V_6\ (8, V_5)\}$ – min V_4

$M_3 = \{V_1, V_2\ (1, V_1), V_5\ (3, V_2), V_2\ (4, V_1)\}$

Крок 4. $T_4 = \{V_3\ (7, V_5), V_6\ (8, V_5)\}$ – min v_3

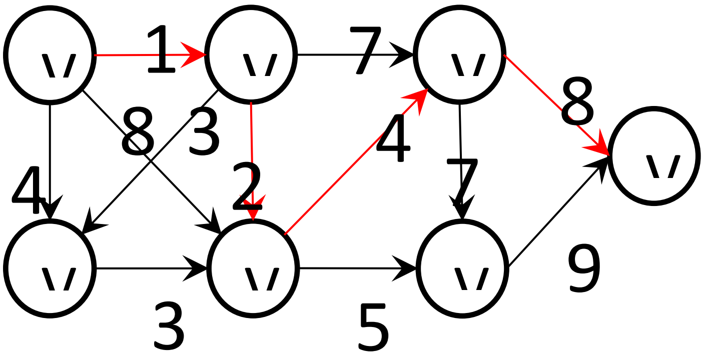
$M_4 = \{V_1, V_2\ (1, V_1), V_5\ (3, V_2), V_4\ (4, V_1), V_3\ (7, V_5)\}$

Крок 5. $T_5 = \{V_6\ (8, V_5), V_6\ (14, V_3), V_7\ (15, V_3)\} = \{V_6\ (8, V_5), V_7\ (15, V_3)\}$ – min V_6

$M_5 = \{V_1, V_2\ (1, V_1), V_5\ (3, V_2), V_4\ (4, V_1), V_3\ (7, V_5), V_6\ (8, V_5)\}$

Крок 6. $T_6 = \{V_7\ (15, V_3), V_7\ (17, V_6)\} = \{V_7\ (15, V_3)\}$

$M_6 = \{V_1, V_2\ (1, V_1), V_5\ (3, V_2), V_4\ (4, V_1), V_3\ (7, V_5), V_6\ (8, V_5), V_7\ (15, V_3)\}$



Машинний алгоритм Дейкстри

Структура даних

1. Логічний масив *visited*[*n*]: *False* – вершина не розглянута; *True* – вершина розглянута.

2. Масив *nlen*[*n*] містить поточні найкоротші відстані від початкової до відповідної вершини.

3. Масив *path*[*n*] містить номери вершин. *path*[*k*] містить номер передостанньої вершини на поточному найкоротшому шляху з початкової вершини у вершину *k*.

4. *matrix* [*i*, *j*] матриця відстаней; $1 \le i \le n$; $1 \le j \le n$.

Кроки алгоритму Дейкстри:

1. **Ініціалізація**

Очистимо масив *visited* для всіх вершин графа: **for** i **in range** (1, n+1): visited[i]=False.

Виберемо стартову вершину s.

Заповнимо масив шляху стартовою вершиною s.

for i **in range** (1, n+1): path[i]=s.

Перепишемо рядок стартової вершини з матриці відстаней у масив nlen

for i **in range** (1, n+1): nlen[i]=matrix [s, i].

Виконаємо початкову установку для стартової вершини s.

visited[d]=True; path[s]=0.

2. **Загальний крок**

Виконуємо перевірку на те, чи залишилися невідмічені вершини.

```
def possible ():
  r=True
  for i in range (1, n+1):
```

```
    if not visited[i]: exit
    r=False
  return=r
```

Знайдемо серед невідмічених ту вершину, що має мінімальну відстань від поточної вершини.

```
def min ():
   minvalue=infinity
   for i in range (1, n+1):
      if not visited[i]:
       if nlen[i]<minvalue:
currentmin=i
minvalue=nlen[i]
min=currentmin
```

Знайдену вершину з мінімальною відстанню від поточної позначаємо як позначену: visited[k]=True (Позначка стає постійною)

Потім модифікуємо списки nlen й path з метою визначення суміжних вершин і відстані до них від стартової вершини.

```
for i in range (1, n+1):
   if nlen[i]>nlen[k]+mattr [i, k]:
     nlen[i]=nlen[k]+mattr [i, k]
path[i]=k
```

3. **Завершальні дії**

Якщо всі елементи списку visited дорівнюють True, тобто всі вершини графа позначені, то довжина шляху від *i v* до *k v* дорівнює nlen[k].

Вивід вершин, що входять у шлях.

```
input (‘Кінцева вершина: ‘, finish)
print(finish)
k=finish
finish=path[pinish]
while finish! = start:
  print (‘ <- ‘, finish)
  finish=path[finish]
print (‘ <- ‘, start)
print (‘Довжина шляху = ‘, nlen[k])
```

Алгоритм Форда – Беллмана знаходження мінімального шляху

Передбачається, що орієнтований граф не містить контурів від'ємної довжини. Основними величинами, які потрібно обчислювати в цьому алгоритмі, є величини $\lambda_i(k)$, де i = 1,2, …, n (n – число вершин графа); k = 1, 2, …, n – 1. Для фіксованих i і k величина $\lambda_i(k)$ дорівнює довжині мінімального шляху, що веде із заданої початкової вершини v_1 у вершину v_i і складається з не більше ніж k дуг.

Крок 1. Установка початкових умов. Ввести кількість вершин графа n і матрицю ваг $C = |c_{ij}|$.

Крок 2. Встановити k = 0. Встановити $\lambda_i(0) = \infty$ для всіх вершин, крім v_1; встановити $\lambda_1(0) = 0$.

Крок 3. У циклі за k, k = 1, 2, …, n – 1, кожній вершині v_i на k-му кроці приписати індекс $\lambda_i(k)$ за таким правилом: $\lambda_i(k) = \min\limits_{1\leq j\leq n}\{\lambda_j(k-1) + c_{ji}\}$ для всіх вершин, крім v_1, встановити $\lambda_1(k) = 0$.

За результатами роботи алгоритму формується таблиця індексів $\lambda_i(k)$, i = 1, 2, …, n; k = 0, 1, 2, …, n – 1.

При цьому $\lambda_i(k)$ визначає довжину мінімального шляху з першої вершини у вершину i, що містить не більше ніж k дуг.

Крок 4. Відновлення мінімального шляху. Для будь-якої вершини v_i попередня до неї вершина v_r визначається зі співвідношення: $\lambda_r(n-2) + c_{rs} = \lambda_s(n-1)$. $v_r \in G^{-1}(v_s)$, де $G^{-1}(v_s)$ – прообраз вершини v_s.

Для знайденої вершини v_r попередня до неї вершина v_qвизначається із співвідношення $\lambda_q(n-3) + c_{qr} = \lambda_r(n-2)$, $v_q \in G^{-1}(v_r)$, де $G^{-1}(v_r)$- прообраз вершини v_rі т. д.

Послідовно застосовуючи це співвідношення, починаючи від останньої вершини v_i, знайдемо мінімальний шлях.

Макроструктура алгоритму

Алгоритм послідовно уточнює значення функції d (cur) .

1. Спочатку задаємо значення d (s) = 0; d (cur) = ∞ для будь-якого $cur \neq s$.

2. Виконуємо $n - 1$ ітерацій, під час яких виконуємо релаксацію всіх ребер графа.

Вхідні дані:

Граф: вершини V, ребра $(i, j) \in E$ з вагами $W(i, j)$. Початкова вершина s.

Вихідні дані: відстані $d(cur)$ до кожної вершини $cur \in V$ від вершини s.

Код алгоритму Форда – Беллмана

```
start=1
ML = 10 ** 9
d = [ML] * N
d[start] = 0
for k in range (1, N):
  for i in range(N):
    for j in range(N):
      if d[j] + W[j][i] < d[i]:
        d[i] = d[j] + W[j][i]
```

Алгоритм Флойда – Воршелла

Метод Флойда – Воршелла безпосередньо ґрунтується на тому факті, що в графі з додатними вагами ребер будь-який неелементарний (довжиною більше 1 ребра) найкоротший шлях складається з інших найкоротших шляхів.

Цей алгоритм більш загальний, на відміну від алгоритму Дейкстри, оскільки він знаходить найкоротші шляхи між будь-якими двома вузлами мережі. У цьому алгоритмі мережа представлена у вигляді квадратної матриці з n рядками і n стовпчиками. Елемент (i, j) дорівнює відстані d_{ij} від вузла i до вузла j, яке має кінцеве значення, якщо існує дуга (i, j), і дорівнює нескінченості в протилежному випадку.

Покажемо спочатку ідею методу Флойда. Нехай задані три вузли i, j і k і задані відстані між ними (рис. 3.108). Якщо виконується нерівність $d_{ij} + d_{jk} < d_{ik}$, то доцільно замінити шлях $i \to k$ на шлях $i \to j \to k$. Така заміна (далі її будемо умовно називати **трикутним оператором**) виконується систематично в процесі виконання алгоритму Флойда.

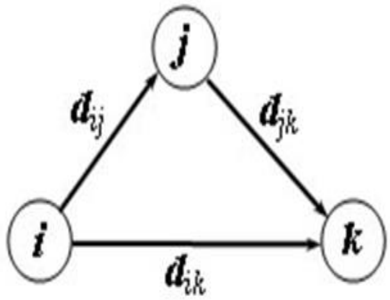


Рис. 3.108. Граф *G*

Ідея алгоритму Флойда:

Припустимо, що нам відомі:

1) найкоротший шлях з вершини i у вершину k, у якому як внутрішні допускають використання лише перших (k – 1) вершин;

2) найкоротший шлях з вершини k у вершину j , у якому як внутрішні допускають використання лише перших (k – 1) вершин;

3) найкоротший шлях з вершини i у вершину j, у якому як внутрішні допускають використання лише перших (k – 1) вершин.

Оскільки за припущенням граф *G* не містить циклів від'ємної довжини, то один з двох шляхів – шлях 3) або об'єднання шляхів 1) та 2) – є найкоротшим шляхом з вершини i у вершину j, у якому як внутрішні допускають використання лише перших k вершин.

$$w_{ij}^{(k)} = \min\left\{w_{ik}^{(k-1)} + w_{kj}^{(k-1)}, w_{ij}^{(k-1)}\right\}$$

Основний алгоритм методу Флойда

Крок 0. Визначаємо вхідну матрицю відстаней D_0 і матрицю послідовності вузлів S_0. Діагональні елементи обох матриць помічаються знаком «–», який показує, що ці елементи не беруть участі в обчисленнях. Покладаємо k = 1:

$D_0 =$

	1	2	...	j	...	n
1	—	d_{12}	...	d_{1j}	...	d_{1n}
2	d_{21}	—	...	d_{2j}	...	d_{2n}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
i	d_{i1}	d_{i2}	...	d_{ij}	...	d_{in}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	d_{n1}	d_{n2}	...	d_{nj}	...	—

$S_0 =$

	1	2	...	j	...	n
1	—	2	...	j	...	n
2	1	—	...	j	...	n
⋮	⋮	⋮	⋮	⋮	⋮	⋮
i	1	2	...	j	...	n
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	1	2	...	j	...	—

Основний крок k. Задаємо рядок k і стовпчик k як **головний рядок** і **головний стовпчик**. Розглядаємо можливість застосування трикутного оператора до всіх елементів d_{ij} матриці D_{k-1}. Якщо виконується нерівність $d_{ik} + d_{kj} < d_{ij}$, $(i <> k, j <> k, i <> j)$, то виконуємо такі дії:

- Створюємо матрицю D_k шляхом заміни в матриці D_{k-1} елемента d_{ij} на суму $d_{ik} + d_{kj}$.
- Створюємо матрицю S_k шляхом заміни в матриці S_{k-1} елемента s_{ij} на k. Покладаємо

$k = k + 1$ і повторюємо крок k.

- Пояснимо дії, що виконуються на k-му кроці алгоритму, представивши матрицю D_{k-1} так, як вона показана на рисунку 3.109. На цьому рисунку рядок k і стовпчик k є головними. Рядок i – будь-який рядок з номером від 1 до $k – 1$, а рядок p – довільний рядок з номером від $k + 1$ до n. Аналогічно стовпчик **j** представляє будь-який стовпчик з номером від 1 до $k – 1$, стовпчик q – довільний стовпчик з номером від k + 1 до n. Трикутний оператор виконується таким чином. Якщо сума елементів головних рядка і стовпчика (що показані в квадратах) менша за суму елементів, що розташовані на перетині стовпчика й рядка (показані в колах), що відповідають головним елементам, які розглядаються, то відстань (елемент в колі) замінюється на суму відстаней, що представлені головними елементами:

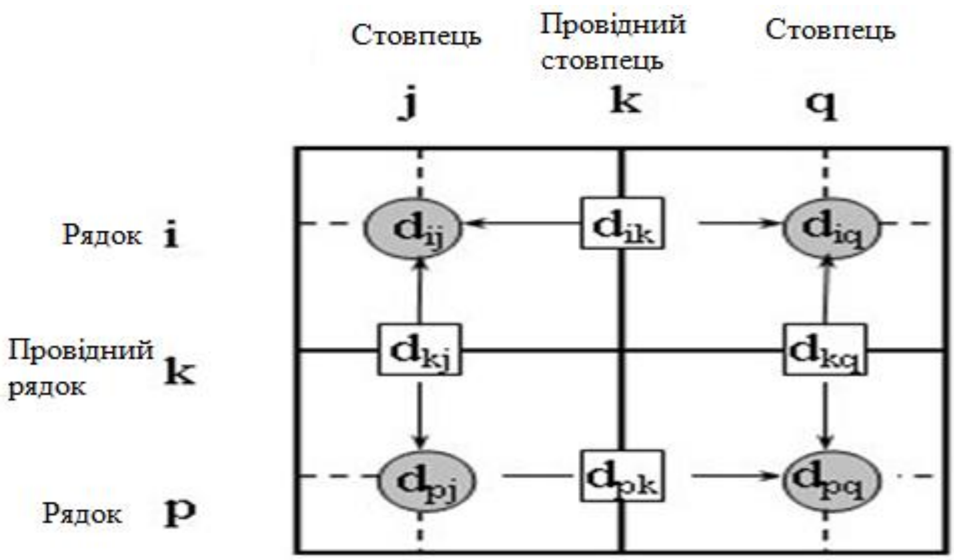


Рис. 3.109. Ілюстрація алгоритму Флойда

- Після реалізації n кроків алгоритму визначення за матрицями D_n і S_n найкоротшого шляху між вузлами i і j виконується за такими правилами:

1. Відстань між вузлами i і j дорівнює елементу d_{ij} в матриці D_n.
2. Проміжні вузли шляху від вузла i до вузла j визначаємо за матрицею S_n. Нехай

$s_{ij} = k$, тоді маємо шлях $i \to k \to j$. Якщо далі $s_{ik} = k$ і $s_{kj} = j$, то вважвємо, що весь шлях визначений, оскільки знайдені всі проміжні вузли. В іншому випадку повторюємо описану процедуру для шляхів від вузла i до вузла k і від вузла k до вузла j.

Приклад 3.38. Знайдемо для мережі, що показана на рисунку 3.110, найкоротші шляхи між будь-якими двома вузлами. Відстань між вузлами цієї мережі проставлені на рисунку біля відповідних ребер. Ребро (3, 5) орієнтоване, тому не допускається рух від вузла 5 до вузла 3. Усі інші ребра допускають рух в обох напрямках:

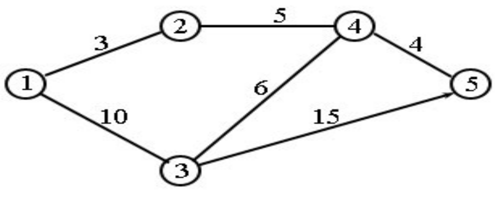


Рис. 3.110. Граф G

<u>Крок 0</u>**.** Початкові матриці D_0 і S_0 будуються безпосередньо за заданою схемою мережі. Матриця D_0 симетрична, за винятком пари елементів d_{35} і

d_{53}, де d_{53} дорівнює нескінечнності, оскільки неможливий перехід від вузла 5 до вузла 3:

D_0

	1	2	3	4	5
1	—	3	10	∞	∞
2	3	—	∞	5	∞
3	10	∞	—	6	15
4	∞	5	6	—	4
5	∞	∞	∞	4	—

S_0

	1	2	3	4	5
1	—	2	3	4	5
2	1	—	3	4	5
3	1	2	—	4	5
4	1	2	3	—	5
5	1	2	3	4	—

<u>Крок 1</u>. У матриці D_0 виділені головні рядок і стовпчик **(**k = 1**)**. Подвійною рамкою представлені елементи d_{23} і d_{32}, єдині серед елементів матриці D_0, значення яких можна покращати за допомогою трикутного оператора. Таким чином, щоб на основі матриць D_0 і S_0 отримати матриці D_1 і S_1, виконуємо такі дії.

1. Замінюємо d_{23} на $d_{21} + d_{13}$ = 3 + 10 = 13 і встановлюємо s_{23} = 1.
2. Замінюємо d_{32} на $d_{31} + d_{12}$ = 10 + 3 = 13 і встановлюємо s_{32} = 1.

- Матриці D_1 і S_1 мають такий вигляд:

D_1

	1	2	3	4	5
1	—	3	10	∞	∞
2	3	—	**13**	5	∞
3	10	**13**	—	6	15
4	∞	5	6	—	4
5	∞	∞	∞	4	—

S_1

	1	2	3	4	5
1	—	2	3	4	5
2	1	—	**1**	4	5
3	1	**1**	—	4	5
4	1	2	3	—	5
5	1	2	3	4	—

<u>Крок 2</u>. Покладаємо k = 2; у матриці D_1 виділені наступні рядок і стовпчик. Трикутний оператор застосовується до елементів матриць D_1 і S_1, що виділені подвійною рамкою. У підсумку отримуємо матриці D_2 і S_2:

D_2

	1	2	3	4	5
1	—	3	10	**8**	∞
2	3	—	13	5	∞
3	10	13	—	6	15
4	**8**	5	6	—	4
5	∞	∞	∞	4	—

S_2

	1	2	3	4	5
1	—	2	3	**2**	5
2	1	—	1	4	5
3	1	1	—	4	5
4	**2**	2	3	—	5
5	1	2	3	4	—

Крок 3. Покладаємо k = 3; у матриці D_2 виділені головні рядок і стовпчик. Трикутний оператор застосовується до елементів матриць D_2 і S_2, що виділені подвійною рамкою. У підсумку отримуємо матриці D_3 і S_3:

D_3

	1	2	3	4	5
1	—	3	10	8	25
2	3	—	13	5	28
3	10	13	—	6	15
4	8	5	6	—	4
5	∞	∞	∞	4	—

S_3

	1	2	3	4	5
1	—	2	3	2	3
2	1	—	1	4	3
3	1	1	—	4	5
4	2	2	3	—	5
5	1	2	3	4	—

Крок 4. Покладаємо k = 4, головні рядок і стовпчик у матриці D_3 виділені. Отримуємо нові матриці D_4 і S_4:

D_4

	1	2	3	4	5
1	—	3	10	8	**12**
2	3	—	**11**	5	**9**
3	10	**11**	—	6	**10**
4	8	5	6	—	4
5	**12**	**9**	**10**	4	—

S_4

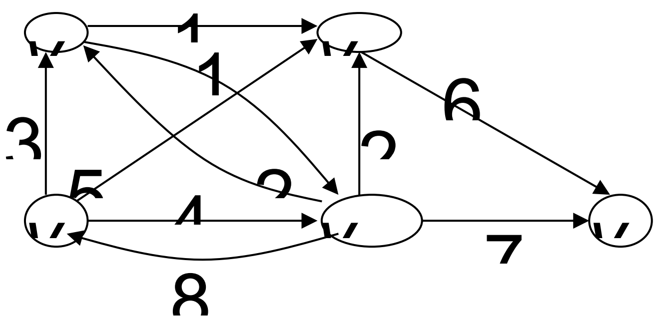
	1	2	3	4	5
1	—	2	3	2	**4**
2	1	—	**4**	4	**4**
3	1	**4**	—	4	**4**
4	2	2	3	—	5
5	**4**	**4**	**4**	4	—

Крок 5. Покладаємо k = 5, головні рядок і стовпчик у матриці D_4 виділені. Жодних дій на цьому кроці не виконуємо; обчислення закінчені.

- Кінцеві матриці D_4 і S_4 мають всю інформацію, необхідну для визначення найкоротших шляхів між будь-якими двома вузлами мережі. Наприклад, найкоротша відстань між вузлами 1 і 5 дорівнює d_{15} = 12.
- Для знаходження відповідних маршрутів нагадаємо, що сегмент маршруту (i, j) складається з ребра (i, j) тільки в тому випадку, коли $s_{ij} = j$. У протилежному випадку вузли i і j зв'язані меншою мірою, через один проміжний вузол. Наприклад, оскільки s_{15} = 4 і s_{45} = 5, спочатку найкоротший маршрут між вузлами 1 і 5 матиме вигляд $1 \rightarrow 4 \rightarrow 5$. Але оскільки s_{14} не дорівнює 4, вузли 1 і 4 у визначеному шляху не зв'язані одним ребром (але у вхідній мережі вони можуть бути зв'язані безпосередньо). Далі слід визначити проміжний вузол (вузли) між першим і четвертим вузлами. Маємо s_{14} = 2 і s_{24} = 4, тому маршрут $1 \rightarrow 4$ замінюємо $1 \rightarrow 2 \rightarrow 4$. Оскільки s_{12} = 2 і s_{24} = 4, інших проміжних вузлів немає. Комбінуючи

визначені сегменти маршруту, нарешті отримаємо наступний найкоротший шлях від вузла 1 до вузла 5: 1→2→4→5. Довжина цього шляху дорівнює 12 кілометрам.

Приклад 3.39. Знайти найкоротший шлях між всіма парами вершин.



Матриця суміжності ваг Матриця маршрутів

$$\begin{array}{c|c} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{pmatrix} 0 & 3 & 4 & 5 & \infty \\ \infty & 0 & 1 & 1 & \infty \\ 8 & 2 & 0 & 2 & 7 \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \end{array} \quad k=1 \qquad \begin{array}{c|c} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix} \end{array}$$

$$\begin{pmatrix} 0 & 3 & 4 & 5 & \infty \\ \infty & 0 & 1 & 1 & \infty \\ 8 & 2 & 0 & 2 & 7 \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \quad k=2 \qquad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 3 & 4 & (4) & \infty \\ \infty & 0 & 1 & 1 & \infty \\ 8 & 2 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \quad k=3 \qquad \begin{pmatrix} 1 & 1 & 1 & (2) & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 3 & 4 & 4 & \infty \\ (9) & 0 & 1 & 1 & \infty \\ 8 & 2 & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \quad k = 4 \quad \begin{pmatrix} 1 & 1 & 1 & 2 & 1 \\ (3) & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 3 & 4 & 4 & (10) \\ 9 & 0 & 1 & 1 & (7) \\ 8 & 2 & 0 & 2 & (8) \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \quad k = 5 \quad \begin{pmatrix} 1 & 1 & 1 & 2 & (4) \\ 3 & 2 & 2 & 2 & (4) \\ 3 & 3 & 3 & 3 & (4) \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

Відповідь:

$$\begin{pmatrix} 0 & 3 & 4 & 4 & 10 \\ 9 & 0 & 1 & 1 & 7 \\ 8 & 2 & 0 & 2 & 8 \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 2 & 4 \\ 3 & 2 & 2 & 2 & 4 \\ 3 & 3 & 3 & 3 & 4 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

Алгоритм Джонсона

Алгоритм Джонсона знаходить найкоротший шлях між всіма парами вершин у зваженому графі з від'ємними вагами без негативних контурів. Давайте розглянемо умови задачі частинами.

Граф, у якому кожне ребро має напрямок, називається орієнтованим (або коротко – орграфом), а його ребра називаються дугами. Наприклад, візьмемо орграф з 4 вершин і 8 дуг:

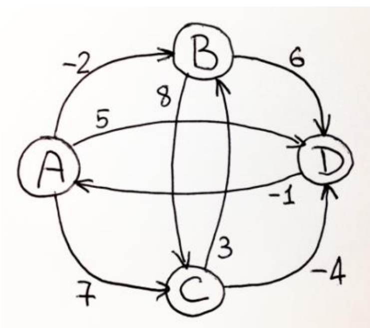


Рис. 3.111. Орграф G

Ми можемо переміщуватися з однієї вершини в іншу по дугах у вказаному напрямку. Переміщення по одній або декількох дугах називається «шлях» або «маршрут».

Для кожної дуги може бути вказане число, яке називається «вага» або «довжина» дуги. Ми будемо дивитися на кожне число як на довжину. Мета алгоритму – знайти найкоротший шлях між будь-якими двома вершинами.

Ви, напевно, вже помітили, що деякі дуги мають від'ємну вагу. Подорож по таких дугах приносить дохід. Мрією мандрівника було б знайти замкнений шлях з від'ємною довжиною і ходити по ньому вічно. Замкнений шлях називається «циклом» або «контуром», і якщо в орграфі є такий негативний контур, то рішення задачі відсутнє, оскільки мінімальна довжина шляху може досягти мінус безкінечності.

Отже, ми розібрали всі терміни умови, сформулюємо мету алгоритму ще раз, більш конкретно: алгоритм Джонсона знаходить найкоротший шлях між всіма парами вершин у зваженому орієнтованому графі з від'ємними вагами без негативних контурів.

Для вирішення цієї задачі можна застосувати алгоритм Флойда – Воршелла, який вирішує задачу «в лоб» повним перебором:

for k = 1 to n // n – кількість вершин у графі
for x = 1 to n
for y = 1 to n
$W[x][y] = \min(W[x][y], W[x][k] + W[k][y])$

Значення $W[x][y]$ елемента містить довжину найкоротшого шляху з вершини x у вершину y.

Початкові значення для W – це матриця суміжності вихідного графа. Замість нулів треба записати плюс безкінечність, оскільки шлях між такими вершинами ще не знайдений.

Суть алгоритму в постійному «релаксуванні» довжини маршруту. На кожній ітерації ми намагаємося дістатися з X в Y через проміжну вершину K і, якщо цей шлях коротший, запам'ятовуємо зменшену довжину.

Алгоритм Флойда – Воршелла перебирає всі можливі варіанти побудови шляхів, складність у нього – кубічна, і якщо в графі вершин багато – він буде працювати дуже довго.

Ще є ефективний алгоритм Дейкстри для пошуку найкоротшого шляху від однієї вершини до всіх інших. Якщо застосувати алгоритм

Дейкстри для кожної вершини, то ми отримаємо найкоротший шлях для всіх пар вершин, причому значно швидше, ніж в алгоритмі Флойда – Воршелла.

Однак алгоритм Дейкстри некоректно працює з графом з від'ємними вагами. Наприклад, під час пошуку від вершини A на першому етапі буде знайдений найкоротший шлях у вершину B (–2), а на другому етапі «найкоротшим» буде шлях з B у D (–2 + 6 = 4). На цьому алгоритм відрапортує про результати і закінчить роботу. Від'ємна дуга CD навіть не буде розглянута і правильна відповідь не буде знайдена. Ось так. Один алгоритм повільний, другий некоректний. Що ж робити?

Відповідь очевидна: застосувати алгоритм Джонсона! Ідея геніальна: на основі цього орграфа сформувати новий орграф таким чином, щоб у ньому не було від'ємних дуг, а всі найкоротші шляхи були такими самими. Пропустити отриманий орграф через алгоритм Дейкстри й на виході отримати правильну відповідь! Як же нам сформувати такий орграф?

Просте неправильне рішення – збільшити довжину кожної дуги на константне значення, щоб від'ємних значень не залишилося. Чому цей варіант некоректний? Тому що він дає фору більш коротким маршрутам, безпідставно збільшуючи сумарну довжину на ту саму константу з кожною новою дугою.

Наприклад, якщо в запропонованому графі довжину кожної дуги збільшити на 4, то найкоротшим шляхом з A в D стане прямий шлях: 5 + 1 * 4 = 9. Тоді як правильна відповідь з 3 дуг (A-B-C-D) отримає зайвих 12 грамів навантаження і випаде з конкуренції: –2 + 8 – 4 + 3 * 4 = 14.

Отже, наша задача – змінити довжину кожної дуги таким чином, щоб позбутися від'ємних дуг і щоб усі найкоротші відстані залишилися такими самими. Як це організувати? Давайте до довжини кожної дуги XY додамо $h(X)$ і віднімемо $h(Y)$, де $h(v)$ – «чиста» функція, яка визначає, де яке константне значення для кожної вершини. Отримаємо такі довжини дуг:

$$A \to D$$

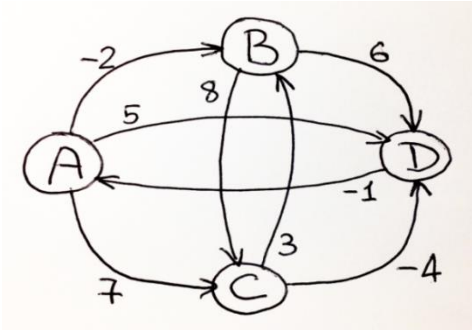
$$|AD| = 5$$

$$|ABD| = -2 + 6 = 4$$

$$|ACD| = 7 - 4 = 3$$

$$|ABCD| = -2 + 8 - 4 = 2$$

$$|ACBD| = 7 + 3 + 6 = 16$$



$$|AC| = 7 + h(A) - h(C)$$
$$|AB| = -2 + h(A) - h(B)$$
$$|AD| = 5 + h(A) - h(D)$$
$$|BC| = 8 + h(B) - h(C)$$
$$|BD| = 6 + h(B) - h(D)$$
$$|CB| = 3 + h(C) - h(B)$$
$$|CD| = -4 + h(C) - h(D)$$
$$|DA| = -1 + h(D) - h(A)$$

Подивимось, як у цьому випадку зміниться довжина кожного маршруту з вершини *A* в *D*:

$$|AD| = 5 + h(A) - h(D)$$
$$|ABD| = -2 + h(A) - h(B) + 6 + h(B) - h(D) = 4 + h(A) - h(D)$$
$$|ACD| = 7 + h(A) - h(C) - 4 + h(C) - h(D) = 3 + h(A) - h(D)$$
$$|ABCD| = -2 + h(A) - h(B) + 8 + h(B) - h(C) - 4 + h(C) - h(D)$$
$$= 2 + h(A) - h(D)$$
$$|ACBD| = 7 + h(A) - h(C) + 3 + h(C) - h(B) + 6 + h(B) - h(D)$$
$$= 16 + h(A) - h(D)$$

Як бачимо, будь-який шлях з вершини *A* в *D* змінюється на однакову величину, *h* (*A*) – *h* (*D*), а, отже, всі найкоротші шляхи залишаться такими ж самими! Саме те, що потрібно. Залишилося тепер знайти значення функції *h*, що підходить, щоб модифіковані довжини дуг стали невід’ємними.

Розглянемо, як можна розрахувати такі значення. Для цього додамо в орграф нову вершину-джерело *S*, з якої проведемо нульові дуги у всі вершини графа. Звернемо увагу, що додавання вершини *S* і дуг *S** не змінює найкоротші шляхи у вихідному графі, тому як всі додані дуги йдуть тільки з *S*, а «нульового» маршруту в зворотному напрямку немає.

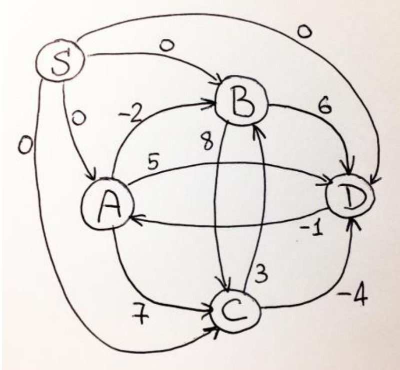
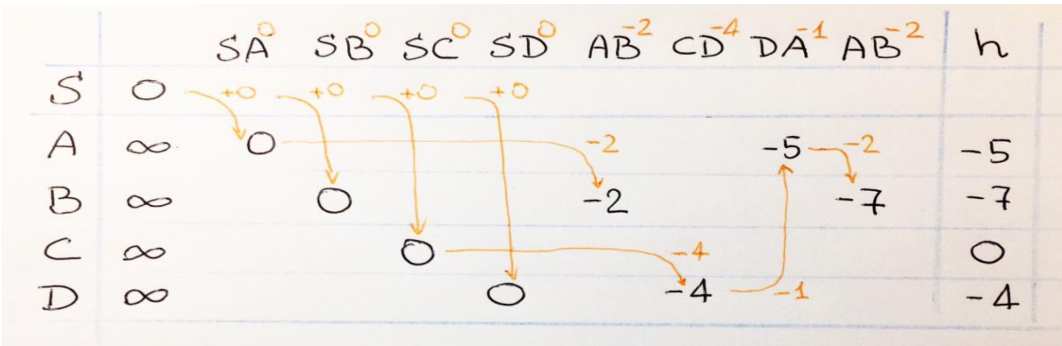


Рис. 3.112. Граф G з вершиною-джерелом S

Тепер запустимо алгоритм Беллмана – Форда, який знайде найкоротші шляхи від S до всіх інших вершин. Цей алгоритм N разів поспіль перебирає всі дуги на предмет «релаксації» найкоротших маршрутів з S у всі інші вершини. У таблиці перераховані «значимі» ітерації роботи цього алгоритму, у кожному стовпці вказана чергова дуга, яка скорочує маршрут від однієї з вершин:



Результат роботи цього алгоритму якраз і дасть нам шукане значення функції h для кожної вершини. Суть цих значень – найкоротший шлях із вершини S. Оскільки всі дуги з S нульові, то і значення функції h – недодатнє. Але нехай це нас не спантеличує, головне, w що отримані значення зафіксовані, не змінюються, а отже, ми можемо перерахувати довжини всіх дуг вихідного орграфа:

$$\hat{w}(x,y) = w(x,y) + h(x) - h(y)$$
$$|AC| = 7 + h(A) - h(C) = 7 + (-5) - (0) = 2$$
$$|AB| = -2 + h(A) - h(B) = -2 + (-5) - (-7) = 0$$
$$|AD| = 5 + h(A) - h(D) = 5 + (-5) - (-4) = 4$$
$$|BC| = 8 + h(B) - h(C) = 8 + (-7) - (0) = 1$$
$$|BD| = 6 + h(B) - h(D) = 6 + (-7) - (-4) = 3$$

$$|CB| = 3 + h(C) - h(B) = 3 + (0) - (-7) = 10$$
$$|CD| = -4 + h(C) - h(D) = -4 + (0) - (-4) = \otimes$$
$$|DA| = -1 + h(D) - h(A) = -1 + (-4) - (-5) = 0$$

За результатами отримаємо орграф без від’ємних дуг:

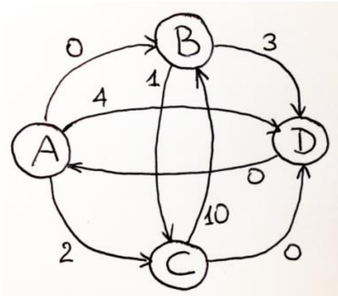
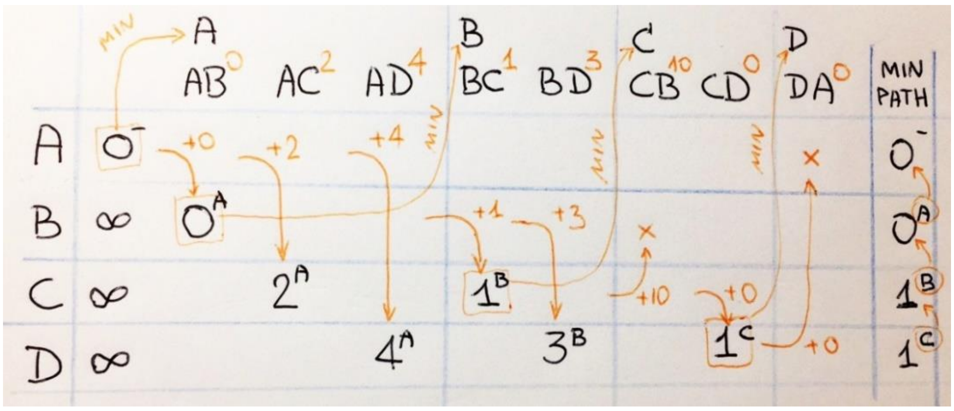


Рис. 3.113. Граф *G* без від’ємних дуг

Тут немає від’ємних дуг, а всі найкоротші маршрути такі самі, як і у вихідному орграфі, чудово! Тепер ми можемо зі спокійною душею запустити алгоритм Дейкстри на цьому орграфі й для кожної вершини отримати правильні відповіді.

Подивимось, як будуть знайдені найкоротші маршрути з вершини *A* до всіх інших:

У кожному стовпці, поруч із новим значенням найкоротшого маршруту, записана вершина, звідки ми до неї прийшли. За цими літерами ми можемо встановити найкоротший шлях. Наприклад, найкоротший маршрут з *A* в *D* встановлюється за останнім стовпчиком і записується справа наліво: $A \Leftarrow B \Leftarrow C \Leftarrow D$, що і треба було знайти.



Складність

Алгоритм Джонсона працює за *O* (*VE* + *VD*), де *O* (*D*) — час роботи алгоритму Дейкстри. Якщо в алгоритмі Дейкстри неспадаюча черга з пріоритетами реалізована у вигляді купи Фібоначчі, то час роботи алгоритму Джонсона – *O* (*V*2log*V* + *VE*). У випадку реалізації черги з пріоритетами у вигляді двоїчної купи час роботи дорівнює *O* (*VE*log*V*).

Обов'язкові завдання

1. Знайти найкоротший шлях з вершини v_1 до вершини v_5 з використанням алгоритму Дейкстри для орграфа *G*, який задано ваговою матрицею.

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}\begin{pmatrix} v_1 & v_2 & v_3 & v_4 & v_5 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2. Визначити мінімальний шлях з вершини v_1 до вершини v_6 за допомогою алгоритму Форда – Беллмана в орграфі *G*, який показаний на рис. 3.114.

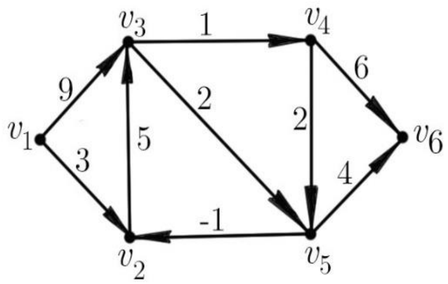


Рис. 3.114. Граф *G* до задачі 2

3. Визначити найкоротший шлях між усіма вершинами за допомогою алгоритму Флойда – Воршелла у графі, заданому на рис. 3.115.

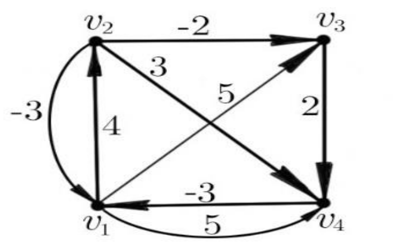


Рис. 3.115. Граф *G* до задачі 3

3.10. Розфарбування графа

Задачі розфарбування

Задачі розфарбування вершин або ребер графа посідають важливе місце в теорії графів.

До задачі розфарбування графа зводиться цілий ряд практичних задач. Одна із сфер – складання розкладів:

- розкладу для освітніх закладів;
- розкладу в спорті;
- планування зустрічей, зборів, інтерв'ю;
- розкладу транспорту, у тому числі авіатранспорту;
- розкладу для комунальних служб;
- інші.

Основні визначення

Нехай $G = (V, E)$ – скінченний граф, а k – деяке натуральне число.

Вершинне розфарбування

Довільну функцію виду $f : V \to N_k$ де $N_k = \{1, 2, \ldots k\}$, $|V| = 4$, N_2 називають вершинним k-розфарбуванням, або просто k-розфарбуванням графа G .

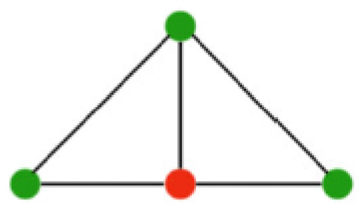


Рис. 3.116. Вершинне 2-розфарбування графа G

Правильне розфарбування. Розфарбування називають правильним, якщо кольори суміжних вершин не співпадають, тобто для будь-яких $(u, v) \in E \Rightarrow f(u) \neq f(v)$. $|V| = 4$, N_3.

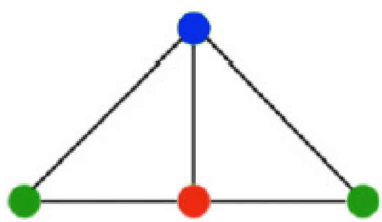


Рис. 3.117. Вершинне правильне 3-розфарбування графа G

Розфарбований граф. Граф, для якого існує правильне k-розфарбування, називають **розфарбованим графом.**

Базовий принцип оптимізації розфарбування

• Якщо функція f не взаємно однозначна, то за $V = k$ фактично може бути використано менше, ніж k кольорів.

• Правильне розфарбування – це розбиття множини вершин.

Правильне k-розфарбування можна розглядати як розбиття множини вершин V графа G на класи

o $V_1 \cup V_2 \cup \ldots \cup V_l = V$, де $l \leq k$,

o $V_i \neq \emptyset$, $i = 1, 2, \ldots, l$.

Кожний клас i V_i – це незалежна множина. Такі класи називають **кольоровими класами.**

Хроматичне число

Визначення. Мінімальне число k, за якого існує правильне k-розфарбування графа G, називають **хроматичним числом** цього графа й позначають $X_p(G)$.

Визначення. Якщо $X_p(G) = k$, то граф G називають **k-хроматичним**. Тобто його вершини можна розфарбувати k різними кольорами так, що в будь-якого ребра інцидентні вершини матимуть різний колір.

Визначення. Правильне k-розфарбування графа G за $k = X_p(G)$ називають **мінімальним.**

Визначення. Хроматичне число незв’язного графа дорівнює максимальному з хроматичних чисел його компонент зв’язності.

Приклад 3.40. Розглянемо граф G, зображений на рис. 3.118, на якому показано одне із правильних k-розфарбувань. Натуральними числами 1, 2, 3, 4 позначені кольори відповідних вершин.

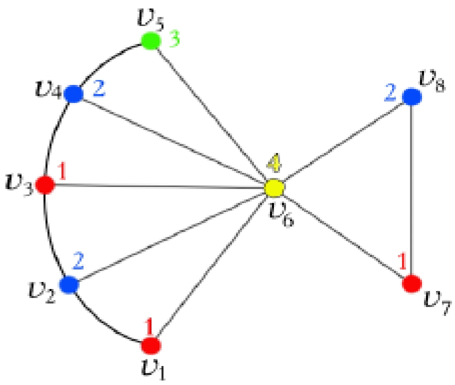


Рис. 3.118. Розфарбований граф G

Хроматичні числа деяких графів

Для деяких простих графів неважко знайти хроматичні числа.

1. Повний граф K_n, що складається з n вершин, має хроматичне число $X_p(K_n) = n$.

Приклад 3.41. Знайти хроматичні числа правильних графів K_2, K_3, K_4, K_5.

Розв'язок. Для знаходження хроматичних чисел потрібно виконати мінімальне правильне розфарбування відповідних графів, як показано на рис. 3.119.

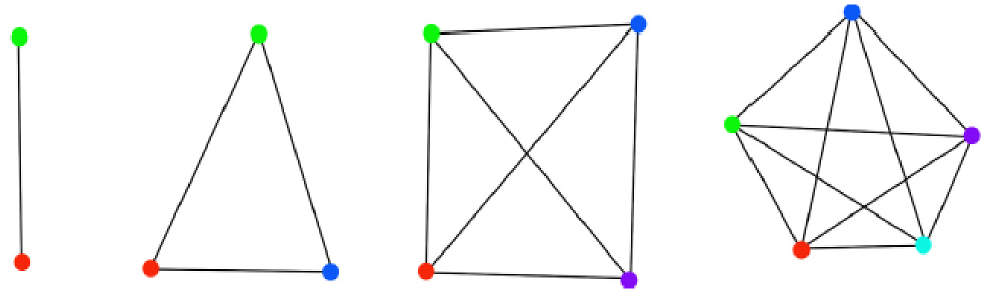


Рис. 3.119. Мінімальне правильне розфарбування повних графів

Повний граф $K_n - e$, який складається з n вершин з одним відсутнім ребром, має хроматичне число $X_p(K_n - e) = n - 1$.

Приклад 3.42. Знайти хроматичні числа графів $K_3 - 1, K_4 - 1, K_5 - 1$.

Розв'язок. Для знаходження хроматичних чисел потрібно виконати мінімальне правильне розфарбування графів, як показано на рис. 3.120.

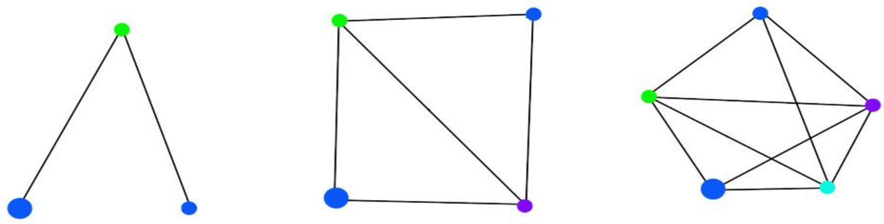


Рис. 3.120. Мінімальне правильне розфарбування графів 3

Повні дводольні графи $K_{m,n}$, що складаються з долей $|A| = m$ і $|B| = n$, мають хроматичне число $X_p(K_{m,n}) = 2$.

Приклад 3.43. Знайти хроматичні числа графів $K_{1,2}, K_{2,2}, K_{2,3}, K_{3,3}$.

Розв'язок. Для знаходження хроматичних чисел потрібно виконати мінімальне правильне розфарбування біхроматичних графів, як показано на рис. 3.121.

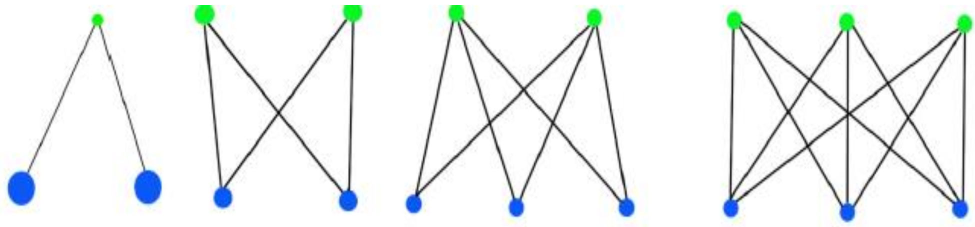


Рис. 3.121. Мінімальне правильне розфарбування біхроматичних графів

Теорема 3.17. Непорожній граф є **біхроматичним** тоді й тільки тоді, коли він не має циклів непарної довжини.

Приклад 3.44. Навести приклади 1-хроматичного, 2-хроматичного і 3-хроматичного графів.

Розв'язок. 1-*хроматичний граф* – порожній граф, показаний на рис. 3.122.

Рис. 3.122. 1-хроматичний граф

2-*хроматичний граф* – дводольний непорожній граф, показаний на рис. 3.123.

2-хроматичні графи зазвичай називають **біхроматичними**.

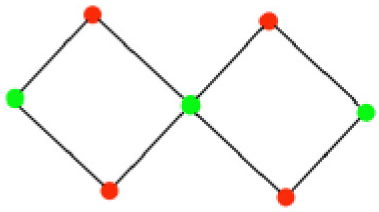


Рис. 3.123. 2-хроматичний граф

3-*хроматичний граф* – циклічний граф із непарним числом вершин у кожному з циклів, показаний на рис. 3.124.

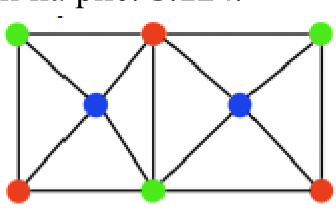


Рис. 3.124. 3-хроматичний граф

Визначення. Якщо граф має n вершин, то його хроматичне число не перевищує n.

Визначення. Якщо граф має підграф K_m, то його хроматичне число не менше, ніж m.

Хроматичне число та стандартні характеристики

У загальному випадку хроматичне число графа не можна обчислити, знаючи тільки його стандартні числові характеристики: число вершин, ребер, компонент зв'язності, розподіл степенів вершин.

Розглянемо графи G_1 й G_2, показані на рис. 3.125.

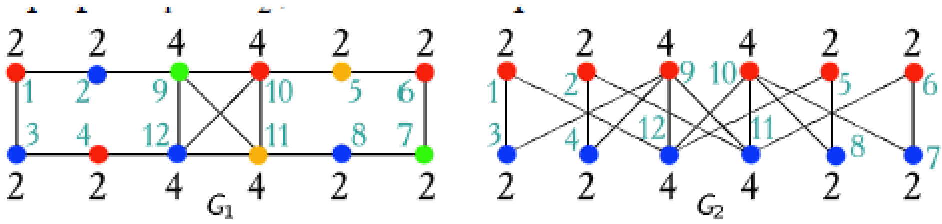


Рис. 3.125. Графи G_1 і G_2

Кожний з них має 12 вершин, у тому числі 4 вершини зі степенем 4 і 8 вершин зі степенем 2, 16 ребер, 1 компонент зв'язності. Але, як видно з рисунка, $X_p(G_1)=4$, а $X_p(G_2)=2$.

Оскільки G_1 містить K_4 підграф, то $X_p(G_1)=4$. Оскільки граф G_2 – дводольний, маємо $X_p(G_2)=2$.

Тому надалі вестимемо мову про оцінки, а не про точні значення хроматичного числа.

Хроматичне число та щільність графа. Три нижні оцінки хроматичного числа

Під **нижніми оцінками** хроматичного числа будемо розуміти нерівності виду $X\ (G) \geq c$, де c – деяка константа, що обчислюється на графі G.

Верхня оцінка хроматичного числа – це нерівності виду $X\ (G) \leq c$, де c – деяка константа, що обчислюється на графі G.

Визначення. Максимальне число вершин, що утворюють повний підграф у графі G, називають щільністю G і позначають через $\omega(G)$.

Повний підграф деякого графа G – це підграф, що складається з попарно суміжних вершин.

Перша нижня оцінка може застосовуватися у випадку, якщо підграфом деякого графа є повний підграф.

Перша нижня оцінка

Для довільного графа G справедлива нерівність $X(G) \geq \omega(G)$.

Визначення. Будь-яку множину попарно несуміжних вершин графа G називають **незалежною множиною.**

Визначення. Максимальне число вершин у незалежній множині називають **числом незалежності** графа G й позначають через $\beta(G)$.

Число незалежності графа – це поняття, протилежне за змістом поняттю щільності графа. Якщо G – звичайний граф, а $\overline{G}$ – його доповнення, то $\beta(G) = \omega(\overline{G})$.

Друга нижня оцінка

Для довільного графа G справедлива нерівність $X(G) \geq \dfrac{n(G)}{\beta(G)}$, де $n = n$ (G) − кількість вершин графа G,

β (G) − число незалежності, яке дорівнює числу внутрішньої стійкості графа G.

Третя нижня оцінка хроматичного числа

Існують **нижні оцінки** хроматичного числа, які використовують тільки ті характеристики графа, що легко обчислюються. Наведемо без доведення одну з них.

Якщо G – звичайний граф і $n = n$ (G) − кількість вершин графа G, $m = m(G)$ – кількість ребер графа G, то хроматичне число $X(G) \geq \dfrac{n^2}{n^2 - 2m}$.

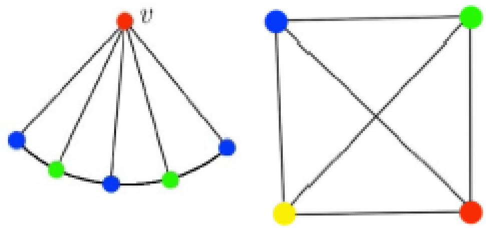
Легко зрозуміти, що в повному графі (як і в будь-якому звичайному графі) подвоєне число ребер менше квадрата числа вершин, і тому число, що стоїть в знаменнику в правій частині нерівності, завжди додатне.

Як видно з описаних вище результатів, **задачі визначення хроматичного числа** графа й побудови мінімального розфарбування довільного графа **досить складні**, а ефективні алгоритми їх розв’язування невідомі. Розглянемо простий алгоритм побудови правильного розфарбування, який у деяких випадках дає розфарбування, близькі до мінімальних.

Верхня оцінка хроматичного числа

Теорема. Для будь-якого графа G наявна нерівність $X_p(G) \le r+1$, де $r = \max\limits_{v \in V}(\deg(v))$.

Приклад 3.45. Перевірити правильність верхньої оцінки хроматичного числа для графів, зображених на рис. 3.126.



а *б*

Рис. 3.126. Графи для визначення верхньої оцінки хроматичного числа

Розв'язок. Розглянемо граф, що показаний на рис. 3.126, *а*.

1. Знаходимо вершину графа, яка має максимальний степінь. $r = \max\limits_{v \in V}(\deg(v)) = 5$.

2. Визначаємо верхню оцінку хроматичного числа $X_p(G) \le r+1$.

Оскільки r = 5, то $X_p(G) \le 6$. З рис. 3.126 а видно, що $X_p(G) = 3$. Оскільки 3 < 6, то оцінка правильна. Граф, показаний на рис. 3.126, *б*, є регулярним графом. Тому r = 3. Звідси $X_p(G) \le 4$. З рис. 3.126, *б* видно, що $X_p(G) = 4$.

Отже, оцінка правильна і у цьому випадку.

Наслідок. Будь-який кубічний граф розфарбовується за допомогою чотирьох фарб. На рис. 3.127 наведено приклад розфарбування кубічного графа.

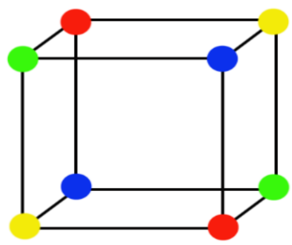


Рис. 3.127. Мінімальне правильне розфарбування куба

Теорема Брукса дає можливість більш точної верхньої оцінки хроматичного числа для спеціальних графів.

Теорема Брукса

Якщо G – зв'язний неповний граф і $r \geq 3$, де $r = \max\limits_{v \in V}(\deg(v))$, $X_p(G) \leq r$.

Приклад 3.46. Перевірити правильність верхньої оцінки хроматичного числа для графа, зображеного на рис. 3.128.

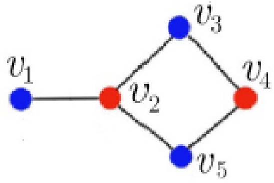


Рис. 3.128. Мінімальне правильне розфарбування графа

Розв'язок. Розглянемо умови застосування теореми Брукса.

1. Вершина v_2 має максимальну степінь deg (v_2) = 3.

2. Граф є зв'язним та неповним. Отже, $X_p(G) \leq 3$ за теоремою Брукса.

Звідси можна зробити висновок, що оцінка правильна. Хоча обидві теореми й дають певну інформацію про хроматичне число графа, але їх оцінки досить неточні. Дійсно, **зірковий граф** K_{1n}, який згідно з теоремою Брукса **розфарбовується** n **фарбами**, насправді є **біхроматичним** (рис. 3.129).

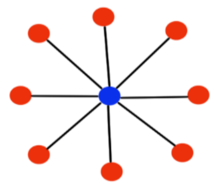


Рис. 3.129. Біхроматичний граф

Ця ситуація значно спрощується, якщо обмежитися ***планарними графами***. У цьому випадку легко довести такий досить загальний і важливий факт.

Теореми про шість, п'ять і чотири фарби

Теорема про шість фарб. Для будь-якого планарного (ізоморфного плоского, у якому ребра перетинаються лише у вершинах) графа *G* правильна нерівність $X_p(G) \le 6$.

Більш детальний аналіз шляхів зниження верхньої границі хроматичного числа приводить до так званої **теореми про п'ять фарб.**

Теорема про п'ять фарб. Для будь-якого планарного графа *G* правильна нерівність $X_p(G) \le 5$.

Теорема про чотири фарби. Кожний планарний граф без петель і кратних ребер є не більш ніж 4-хроматичним. Проблема чотирьох фарб залишалася невирішеною протягом багатьох років. Стверджується, що ця теорема була доведена за допомогою певних міркувань і комп'ютерної програми в 1976 році (Kenneth Appel and Wolfgang Haken. Every Planar Map is Four Colorable. Contemporary Mathematics 98, American Mathematical Society, 1980).

Задача про розподіл устаткування

На підприємстві планують виконати 8 робіт. Для виконання цих робіт потрібні механізми. Використання механізмів для кожної з робіт визначається таблицею 3.1.

Жоден із механізмів не може бути використаний одночасно на двох роботах. Виконання кожної роботи займає 1 годину. Як розподілити механізми, щоб сумарний час виконання всіх робіт був мінімальним і який цей час?

Таблиця 3.1

Таблиця використання механізмів

Меха-нізм	Робота						
	v_1	v_2	v_3	v_4	v_5	v_6	v_7
a_1	+		+				+
a_2		+		+			
a_3			+			+	+
a_4	+	+		+	+		
a_5			+		+		
a_6					+	+	

Розв'язок. Розглянемо граф *G*, показаний на рис. 3.130. Вершинами даного графа є плановані роботи $v_1, v_2, \dots, v_8$, а ребра з'єднують роботи, у яких бере участь хоча б один загальний механізм (і які, із цієї причини, не можна проводити одночасно).

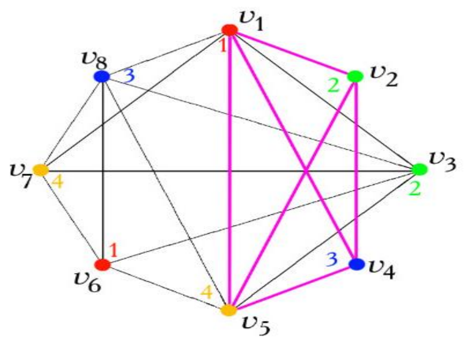


Рис. 3.130. Граф планування робіт

Вершини v_1, v_2, v_4, v_5 породжують підграф графа *G*, ізоморфний K_4. Отже, $X\{G\} \geq 4$. Отже, $X(G)$. Таким чином, усі роботи можна виконати за 4 години.

Для цього, відповідно до знайденого розфарбування графа *G*, потрібно за першу годину виконувати роботи v_1 і v_6, за другу годину – роботи v_2 і v_3, за третю годину – роботи v_4 і v_8, за четверту годину – роботи v_5 й v_7. У таблиці 3.2 кольорами показана послідовність виконання робіт, яка забезпечує максимальне завантаження механізмів.

Таблиця 3.2

Таблиця визначення послідовності робіт

Механізм	Робота							
	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
a_1	+		+				+	+
a_2		+		+				
a_3			+			+	+	
a_4	+	+		+	+			
a_5			+		+			+
a_6					+	+		+

Задача складання розкладу

Умова задачі. Потрібно прочитати лекції із чотирьох предметів двом групам студентів. Деякі з лекцій не можуть бути прочитані одночасно з ряду причин, а саме:

1. Лекцію із цього предмета в різних групах читає той самий лектор.
2. Дві лекції тій самій групі одночасно читатися не можуть.
3. Різні лекції повинні проходити в тому самому приміщенні.

Потрібно скласти розклад так, щоб читання всіх лекцій зайняло мінімально можливий час (за «одиницю часу» в цій задачі природно розглядати одну пару).

Конкретизуємо постановку задачі

1. Будемо розглядати дві групи студентів: 1 і 2.

2. Предмети:

– обчислювальні методи – читає викладач X,

– дискретна математика – читає викладач X,

– математичний аналіз – читає викладач Y,

– українська мова – читає викладач Z.

Знайти мінімальне число пар, у які можна «укласти» всі заняття, і скласти відповідний розклад. **Тобто створити максимально щільний розклад**

Розв'язок. Створимо граф, показаний на рис. 3.131. Вершини цього графа позначені такими символами: О1, О2, Д1, Д2, М1, М2, У1 і У2 (літера відповідає предмету, а цифра – номеру групи). З'єднаємо ребрами вершини, відповідні до пар, які не можна проводити одночасно.

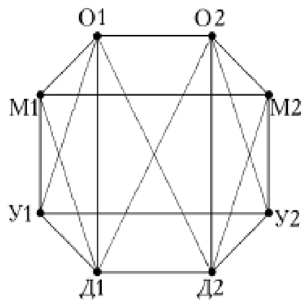


Рис. 3.131. Граф для складання розкладу

Вершини О1, О2, Д1 і Д2 цього графа породжують у ньому підграф, ізоморфний графу K_4. Отже, хроматичне число нашого графа не менше 4.

На рис. 3.132 зазначене правильне розфарбування нашого графа в 4 фарби.

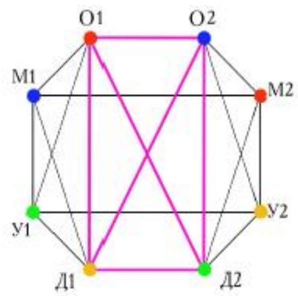


Рис. 3.132. Мінімальне правильне розфарбування графа розкладу

Отже, хроматичне число графа дорівнює 4, тобто всі заняття можна провести за 4 пари. Відповідний розклад зазначений у таблиці 3.3.

Таблиця 3.3

Оптимальний розклад

	Група 1	Група 2
1 пара	Обч. методи	Матем. аналіз
2 пара	Матем. аналіз	Обч. методи
3 пара	Українська мова	Дискр. матем
4 пара	Дискр. матем.	Українська мова

Обов'язкові завдання

1. Для яких графів відомі точні значення хроматичних чисел?

2. Чому дорівнює хроматичне число графа *G*, показаного на рис. 3.133?

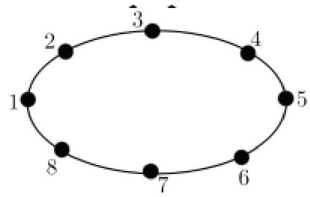


Рис. 3.133. Граф до задачі 2

3. Визначити першу нижню оцінку хроматичного числа для графа, показаного на рис. 3.134.

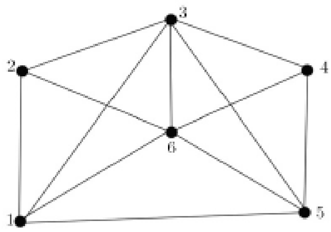


Рис. 3.134. Граф до задачі 3

4. Визначити другу та третю нижні оцінки хроматичного числа для графа, показаного на рис. 3.135.

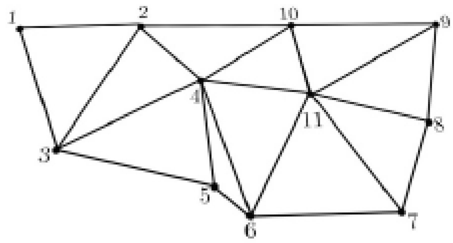


Рис. 3.135. Граф до задач 4 та 5

5. Застосувати теорему Брукса для визначення верхньої оцінки хроматичного числа для графа, зображеного на рис. 3.135.

6. Визначте точне значення хроматичного число повного графа з 15 вершин, після видалення одного ребра.

3.11. Основні алгоритми розфарбування графів

Базові відомості

Будемо розглядати алгоритми розфарбування на неорієнтованих графах без петель.

Граф називають **r-хроматичним**, якщо його вершини **можуть бути розфарбовані** з використанням *r* кольорів (фарб) так, що не знайдеться двох суміжних вершин одного кольору. Найменше число *r*, таке, що граф *G* є **r-хроматичним**, називають **хроматичним числом** графа і позначають *X* (*G*).

Задачу знаходження хроматичного числа графа називають **задачею про розфарбовування (або завданням розфарбовування**) графа. Відповідно до хроматичного числа розфарбування вершин розбиває множину вершин графа на *r* підмножин, кожна з яких містить вершини одного кольору. Ці множини є незалежними, оскільки в межах однієї множини немає двох суміжних вершин.

Завдання знаходження хроматичного числа довільного графа стало предметом багатьох досліджень наприкінці XIX – XX столітті. Із цього питання отримано багато цікавих результатів.

Хроматичне число графа не можна знайти, знаючи тільки кількість вершин і ребер графа. Недостатньо також знати степінь кожної вершини, щоб обчислити хроматичне число графа. За відомих величин *n* (кількість вершин), m (кількість ребер) і $\deg(v_1),...,\deg(v_n)$ (степені вершин графа) можна отримати тільки верхню і нижню оцінки для хроматичного числа графа.

Алгоритм неявного перебору

Алгоритм прямого неявного перебору є найпростішим алгоритмом вершинного розфарбування графів. Цей алгоритм дає змогу реалізувати правильне розфарбування графа з вибором мінімальної в рамках цього алгоритму кількості фарб.

Нехай множина вершин графа довільно упорядкована, тобто для довільної i-ї вершини графа існує певний індекс v_i. Тоді всі вершини графа G утворюють множину $V = \{v_1, v_2, ..., v_n\}$, де n – потужність множини вершин: $|V| = n$.

Тоді перше допустиме розфарбування може бути одержано так:

1. Розфарбуємо першу довільну вершину v_i кольором 1.

2. Кожну наступну вершину будемо розфарбовувати, застосовуючи таке правило:

3. Вибираємо вершину v_{i+1}:

– намагаємося використати колір 1;

– якщо колір 1 використати не вдається, то вводимо колір 2 і розфарбовуємо вершину кольором 2.

4. Вибираємо вершину v_{i+2} – намагаємося використати колір 1:

– якщо колір 1 використати не вдається, то намагаємося використати колір 2;

– якщо колір 2 використати не вдається, то вводимо колір 3 і розфарбовуємо вершину кольором 3.

5. Продовжуємо алгоритм, використовуючи такий порядок розфарбування до того моменту, коли всі вершини графа будуть розфарбовані.

Вибравши початковою іншу вершину або інакше впорядкувавши граф, можемо одержати інше розфарбування.

Кількість обчислень за цим алгоритмом може бути мінімальною, якщо вершини пронумерувати таким чином, щоб першими стояли ті вершини, які входять у **кліку** графа.

Кліка в неорієнтованому графі – це підмножина його вершин, яка утворює повний підграф.

Отже, якщо вершини $v_1, v_2, \ldots, v_l$ розфарбовані *l* кольорами 1, 2, ..., *l*; $l \le i$, то новій довільно взятій вершині v_{i+1} припишемо мінімальний колір, не використаний під час розфарбування суміжних зі нею вершин.

Розфарбування, до якого приводить описаний алгоритм, називають алгоритмом прямого неявного перебору або послідовним алгоритмом розфарбування.

Розглянемо кільцевий граф, вершини якого пронумеровані у різний спосіб (рис. 3.136). У першому випадку маємо граф G_1 (а), а у другому випадку – граф G_2 (б).

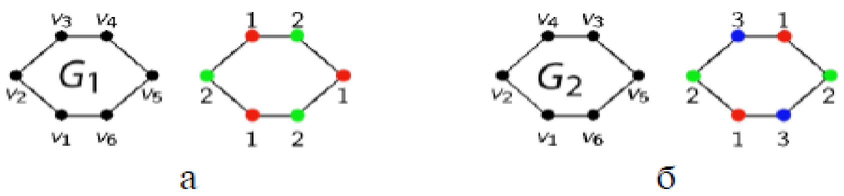


Рис. 3.136. Розфарбування кільцевих графів з різною нумерацією

Отримане розфарбування завжди правильне, але не завжди оптимальне навіть для простих графів. Починаючи з вершини v_1 та використовуючи алгоритм неявного перебору, одержимо розфарбування лише двома фарбами. У другому випадку під час розфарбування вершини v_4, виникає ситуація, коли потрібно вводити колір 3. Отже, основним недоліком алгоритму прямого неявного перебору є його залежність від нумерації вершин.

Програмний код алгоритму прямого неявного перебору

Програмний код написаний мовою програмування *Python*. Він забезпечує генерацію випадкової симетричної матриці суміжності *a*[*r*] та випадкового списку кольорів *colarr*[i].

```
from random import *
n = 5 # максимальна кількість вершин графа
colarr=[0 for i in range(n)]
# Генерація симетричної матриці
```

```
a = [ ]
for r in range (n): # n рядків
   a.append([ ]) # створили рядок
   for c in range(n): # в кожному рядку по n елементів
      if r > c:
         # додаємо однонаправлене ребро
         a[r].append(randrange(0,2))
      else:
        a[r].append(0) # протилежний напрям
# Формування протилежних напрямів
for r in range(n):
   for c in range(n):
       if a[r][c]!=0:
         a[c][r]=a[r][c]
for r in range(n): # 6 рядків
   print(a[r])
def color(i):
#Функція вибору фарби для розфарбування вершини з номером і
   w = {0};
   for j in range(i):
     if a[j][i]>0: w.add(colarr[j])
   curcol=0
   while True:
      curcol+=1
      if curcol not in w: break
   return curcol
 for i in range(n):
    colarr[i]=color(i)
 print(colarr)
```

Наведемо приклад розфарбування графа за цим алгоритмом.

Приклад 3.47. Розглянемо граф *G* (*V*, *E*), показаний на рис. 3.137.

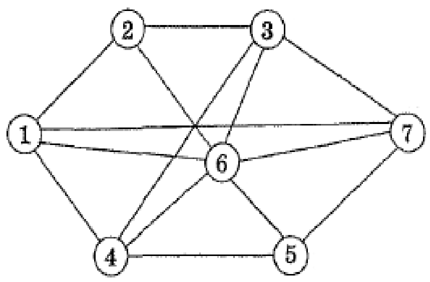


Рис. 3.137. Граф для розфарбування прямим неявним перебором

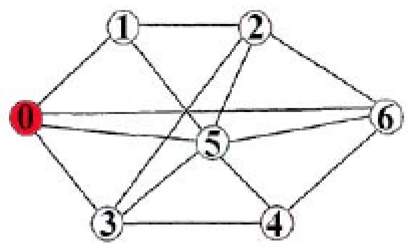
Множину вершин графа $V = \{1, 2, 3, 4, 5, 6, 7\}$ потрібно розфарбувати з використанням алгоритму послідовного розфарбування.

Розв'язок. Сформуємо матрицю суміжності A:

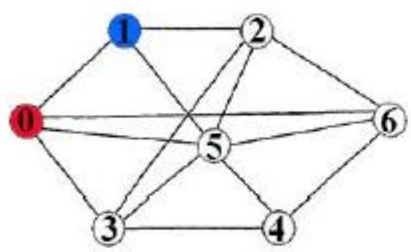
$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}.$$

Використовуючи цю матрицю, виконаємо алгоритм покроково.

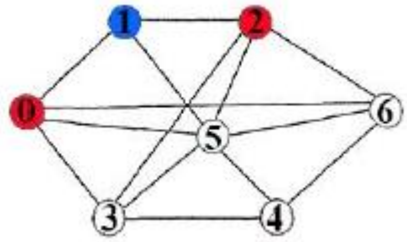
***Крок* 0**. Розглядаємо вершину 0. Множина розфарбованих суміжних вершин *w* містить колір 0. Тому функція color (0) повертає 1. Нехай колір 1 – червоний.



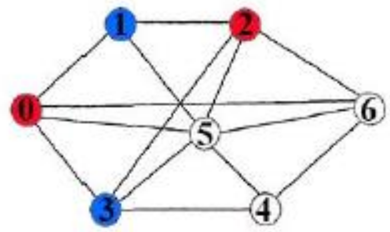
***Крок* 1**. Розглянемо вершину 1. Єдиною меншою за номером суміжною вершиною є вершина 0, яка вже червона. Тому множина w містить елементи {0, 1}. Функція color (1) повертає наступну за номером фарбу синього кольору: curcol=1.



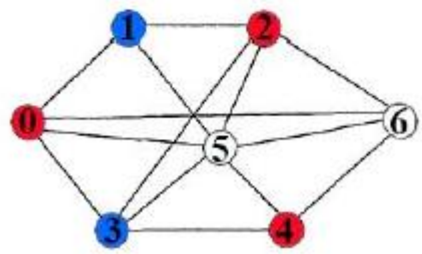
***Крок* 2**. Вершина 2 має єдину суміжну вершину 1 з меншим номером. Множина w містить елементи [0, 2]. Функція color (2) повертає фарбу з номером 1 червоного кольору.



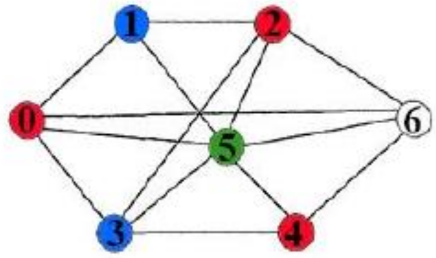
***Крок* 3**. Вершина 3 має дві суміжні вершини з меншими номерами: 0 і 2. Оскільки обидві вершини розфарбовані в колір 1, то множина w містить елементи {0, 1}. Тому функція color (3) повертає наступну за номером фарбу 2 синього кольору.



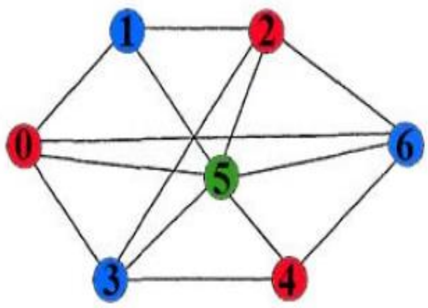
***Крок* 4**. Вершина 4 має єдину суміжну вершину з меншим номером. Це вершина 3. Множина w містить елементи {0, 2}. Тому функція color (4) повертає фарбу з номером 1 червоного кольору.



***Крок* 5**. Вершина 5 має такі суміжні вершини з меншими номерами: 0, 1, 2 і 4. Ці вершини розфарбовані в колір 1 і колір 2. Отже, множина містить елементи: {0, 1, 2}. Тому функція color (5) повертає наступну за номером фарбу 3 зеленого кольору.



***Крок* 6**. Вершина 6 має такі суміжні вершини з меншими номерами: 0, 2, 4 і 5. Ці вершини розфарбовані в колір 1 та колір 3. Отже, множина w містить два елементи: {0, 1, 2}. Тому функція color (6) повертає фарбу 2 синього кольору. За результатами роботи цього алгоритму одержуємо правильно розфарбований граф, що показаний на рисунку.



Евристичний алгоритм розфарбування

Точні методи розфарбовування графа складні для програмної реалізації. Однак існує багато евристичних процедур розфарбовування, які дають змогу знаходити хороші наближення для визначення хроматичного числа графа. Такі процедури також можуть з успіхом використовуватися під час розфарбовування графів із великим числом вершин, де застосування точних методів не виправдане з огляду на високу трудомісткість обчислень. З евристичних процедур розфарбовування слід зазначити послідовні методи, засновані на впорядкуванні множини вершин. В одному з найпростіших методів вершини спочатку розташовуються в порядку зменшення їх степенів. Перша вершина зафарбовується в колір 1, потім список вершин переглядається за зменшенням степенів, і в колір 1 зафарбовується кожна вершина, яка не є суміжною з вершинами, зафарбованими в той самий колір. Потім повертаємося до першої в списку незафарбованої вершини, фарбуємо її в колір 2 і знову переглядаємо список вершин зверху вниз, зафарбовуючи в колір 2 будь-яку незафарбовану вершину, яка не з’єднана ребром з іншою, уже пофарбованою в колір 2, вершиною. Аналогічно діємо із кольорами 3, 4 і т. д., допоки не будуть пофарбовані всі вершини. Кількість використаних кольорів буде тоді наближеним значенням хроматичного числа графа.

Послідовність дій в евристичному алгоритмі має вигляд:

1. Упорядкувати вершини за спаданням степеня.
2. Вибрати колір фарбування 1.

3. Розфарбувати першу вершину в колір 1.

4. Поки не пофарбовані всі вершини, повторювати п. 4.1–4.2:

4.1. Розфарбувати у вибраний колір кожну вершину, яка не суміжна з іншою вершиною, уже пофарбованою в цей колір.

4.2. Вибрати наступний колір.

5. Повернутися до першої в списку нерозфарбованої вершини. Число використаних кольорів буде тоді наближеним значенням хроматичного числа графа.

На рис. 3.138 показано приклад вибору кольору вершини v_{i+1} за евристичним алгоритмом.

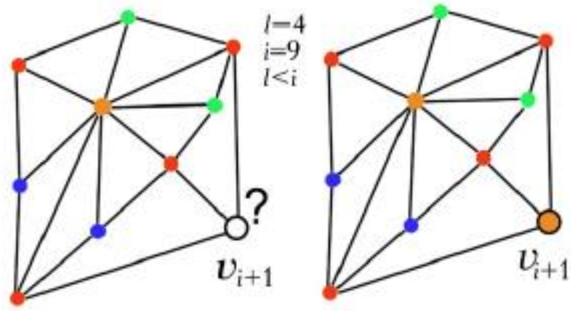


Рис. 3.138. Розфарбування графа за евристичним алгоритмом

Розглянемо кроки евристичного алгоритму детально:

***Крок* 1**. Сортувати вершини графа за степенями зменшення:
$\deg(x_i) \ge \deg(x_j)$, для будь-яких $x_i, x_j \in G$.

Встановити поточний колір $p = 1$ та $i = 1$.

***Крок* 2**. Вибрати чергову нерозфарбовану вершину зі списку і призначити їй новий колір $col(x_i) = p$, $X = \{x_i\}$.

***Крок* 3**. $i = i + 1$. Вибрати чергову не розфарбовану вершину x_i і перевірити умову суміжності: $x_i \cap \Gamma(X) = \varnothing$, де X – множина вершин, уже розфарбованих у колір p. Якщо вершина x_i не є суміжною з даними вершинами, то також присвоїти їй колір p: $col[x_i] = p$.

***Крок* 4**. Повторювати крок 3 до досягнення кінця списку ($i == n$).

***Крок* 5**. *Якщо* всі вершини графа розфарбовані, *то* – кінець алгоритму; *інакше*: $p = p + 1$; $i = 1$. Повторити крок 2.

Для роботи алгоритму можна використовувати довільну структуру даних, яка однозначно задає граф.

Розглянемо приклад програми реалізації евристичного алгоритму зі структурою графа, яка задана матрицею суміжності *a*.

Програмний код евристичного алгоритму

Як і у попередньому випадку, цей алгоритм передбачає початкову генерацію симетричної матриці суміжності та списку кольорів.

```
# Код евристичного алгоритму
from random import *
n = 5 # максимальна кількість вершин графа
# Початковий колір
curcol=1
# Список кольорів вершин
colarr=[0 for i in range(n)]
# Генерація симетричної матриці
a = [[0 for i in range(n)] for j in range(n)]
for r in range(n): # n рядків
  for c in range(n): # в кожному рядку по n елементів
    if r > c:
      a[r][c]= randrange(0,2) # додаємо ребро
          a[c][r]=a[r][c] # протилежний напрям
# Формування списку за степенями
def degforming():
  def getkey(item):
    return item[0]
  # Список кортежей (степінь, номер вершини)
  degarr=[[0 for i in range(2)] for j in range(n)]
  for i in range(n):
   for j in range(n):
      degarr[i][0] += a[i][j]
      degarr[i][1] = i
  # Сортуємо кортежі за спаданням степенів
 degarr.sort(key=getkey, reverse=True)
 return degarr
# Розфарбовка вершин
def dyer(curcol,node):
   for k in range(n):
```

```
        if a[node][k]==0:
         if colarr[k]==0:colarr[k]= curcol
# Основний код
sortarr=degforming()
for i in range(n):
  if not colarr[sortarr[i][1]]:
     colarr[sortarr[i][1]]=curcol
     dyer(curcol,sortarr[i][1])
     curcol+=1
       for r in range(n): print(a[r])
          print(sortarr)
           print(colarr)
```

Приклад евристичного алгоритму розфарбування

Розфарбуємо граф *G*, зображений на рис. 3.139.

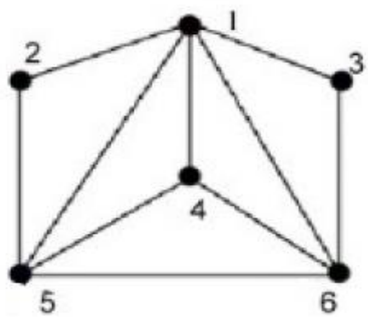


Рис. 3.139. Граф для розфарбування евристичним алгоритмом

Проміжні дані для вирішення завдання будемо записувати в таблицю 3.4.

Таблиця 3.4

Таблиця розфарбування

Номери вершин SortArr	1	5	6	4	2	3
Степені вершин DegArr	*5*	*4*	*4*	*3*	*2*	*2*
CurCol = 1	*1*	-	-	-	-	-
CurCol = 2	*1*	*2*	-	-	-	*2*
CurCol = 3	*1*	*2*	*3*	-	*3*	*2*
CurCol = 4	*1*	*2*	*3*	*4*	*3*	*2*

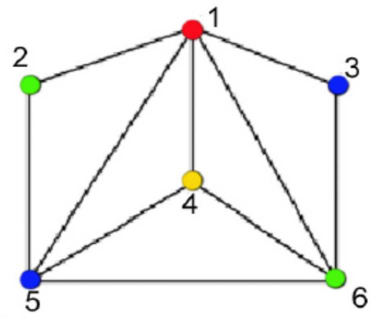
Крок 1. Першою SortArr стоїть вершина 1, яку фарбуємо червоним кольором 1. Несуміжних з 1 немає.

Крок 2. Другою в SortArr стоїть вершина 5, яку фарбуємо синім кольором 2. Несуміжна з 5 вершина 3, яку процедура dyer(curcol,5) фарбує також синім кольором 2.

Крок 3. Третьою в SortArr є вершина 6, яку фарбуємо зеленим кольором 3. Несуміжна з 6 вершина 2, яку процедура dyer(curcol,6) фарбує також зеленим кольором 3.

Крок 4. Четвертою в SortArr є вершина 4, яку фарбуємо жовтим кольором 4. Всі несуміжні вершини із 4 уже розфарбовані.

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{array}$$



Рекурсивна процедура послідовного розфарбування

1. Фіксуємо порядок обходу вершин.

2. Ідемо по суміжних вершинах, використовуючи такий найменший колір, який не створить конфліктів.

3. Якщо на черговому кроці колір вибрати не вдалося, то «відкочуємось» до попередньої вершини й вибираємо для неї наступний колір, який не створить конфліктів. У процедурі використовується рекурсивний виклик процедури фарбування наступної вершини у випадку успішного фарбування попередньої вершини.

Код алгоритму має вигляд:

```
#Код рекурсивного алгоритму
from random import *
# Максимально допустима кількість кольорів
cmax=10
n = 5 # максимальна кількість вершин графа
#Список кольорів вершин
color=[0 for i in range(n)]
# Генерація симетричної матриці
```

```
a = [[0 for i in range(n)] for j in range(n)]
for r in range(n): # n рядків
  for c in range(n): # в кожному рядку по n елементів
    if r > c:
      a[r][c]= randrange(0,2) # додаємо ребро
      a[c][r]=a[r][c] # протилежний напрям
def visit (i):
# Функція вибору фарби для розфарбування вершини
з номером i
  def nicecolor():
    w = {0}
    newcol=0
    for j in range(n):
      if a[i][j] > 0: w.add(color[j])
for cm in range(1,cmax):
      if cm not in w:
        newcol=cm
        break
return newcol
# код функції visit
  if i == n:
#Якщо всі вершини розфарбовані, те виводимо результат
    print("FINAL")
  else:
#Якщо поточна вершина не розфарбована
     if color[i]==0:
      curcol=nicecolor()
      if curcol >0:
#Якщо неконфліктний, то розфарбовуємо вершину і фарбою c
          color [i] = curcol
#Рекурсивно викликаємо для наступної вершини
          visit (i + 1)
#Основний код програми
visit (0)
for r in range(n):
  print(a[r])
```

```
print()
print(color)
```

Приклад роботи рекурсивної процедури

Розглянемо граф G та застосуємо рекурсивну процедуру для його розфарбування (рис. 3.140).

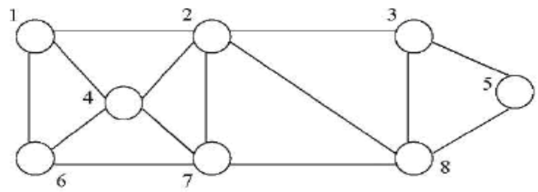


Рис. 3.140. Граф G для рекурсивного розфарбування

Матриця суміжності має такий вигляд

$$A = \begin{pmatrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 4 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 6 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 7 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 8 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Робота алгоритму зведена в таблицю 3.6.

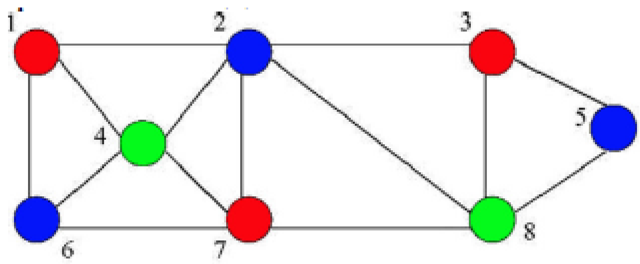
Перший стовпець містить виклики процедури Visit, а решта стовпців показує, яка фарба була прийнята, а яка – відхилена.

Таблиця 3.6

Послідовність розфарбувань рекурсивним алгоритмом

	Червоний	Синій	Зелений
Visit(1)	+		
Visit(2)	-	+	
Visit(3)	+		
Visit(4)	-	-	+
Visit(5)	-	+	
Visit(6)	-	+	
Visit(7)	+		
Visit(8)	-	-	+

Розфарбований граф має вигляд:



«Жадібний» алгоритм розфарбування

Нехай дано зв'язний граф *G* (*V*, *E*).

1. Задамо множину $monochrom = \varnothing$, куди будемо записувати всі вершини, які можна пофарбувати одним кольором.

2. Переглядаємо всі вершини й виконуємо наступний «жадібний» алгоритм:

Procedure Greedy
For (для кожної незафарбованої вершини $v \in V$) **do**
If *v не суміжна з вершинами з monochrom* **then**
begin
  color (*v*) =колір*;*
  $monochrom = monochrom \cup \{v\}$
end.

Розглянемо детальніше програмну реалізацію цього алгоритму за умови, що для представлення графа використовують матрицю суміжності.

```
# код жадібного алгоритму
from random import *
n=5
allcolored = False #Ознака того, що всі вершини не розфарбовані
color=0
#Список кольорів вершин
colarr=[0 for i in range(n)]
# Генерація симетричної матриці
a = [[0 for i in range(n)] for j in range(n)]
for r in range(n): # n рядків
  for c in range(n): # в кожному рядку по n елементів
    if r > c:
      a[r][c]= randrange(0,2) # додаємо ребро
      a[c][r]=a[r][c] # протилежний напрям
```

```
#Функція вибору фарби для розфарбування вершини з номером i
def avid (i,color):
   def check (i): #Перевірка кольору суміжних вершин
     ch = True
    for j in range(n):
#Якщо вершина j суміжна з тією, що підлягає перевірці
            if a [i][j] == 1:
             if (j in w): ch = False
       return ch
   w = set() # Очищаємо множину одноколірних вершин
#Розфарбовуємо першу вершину новою фарбою
   colarr [i]=color
#Доповнюємо множину одноколірних вершин вершиною
   w.add(i)
   #Перевіряємо інші вершини на можливість розфарбування цією фарбою
   for k in range(n):
      if colarr[k] == 0:
        if check(k):
           colarr [k] = color
           w.add(k)
# Головний код
while not allcolored: #Цикл по вершинах графа
    allcolored = True
    for i in range(n):
       if colarr [i] == 0: #Знайшли не розфарбовану вершину
         color += 1 # Встановлюємо новий колір
         allcolored = False
         avid (i,color) #процедура жадібного розфарбування
for r in range(n): # 6 рядків
print(a[r])
print()
print(colarr)
```

Можливі результати роботи «жадібного» алгоритму розфарбування.

Розглянемо варіанти розфарбування графа, показаного на рис. 3.141. Нехай зафарбуємо вершину 1 у синій колір, а потім, пропустивши вершину 2, зафарбуємо в синій колір вершини 3 і 4. Тоді можна одержати 2 фарби.

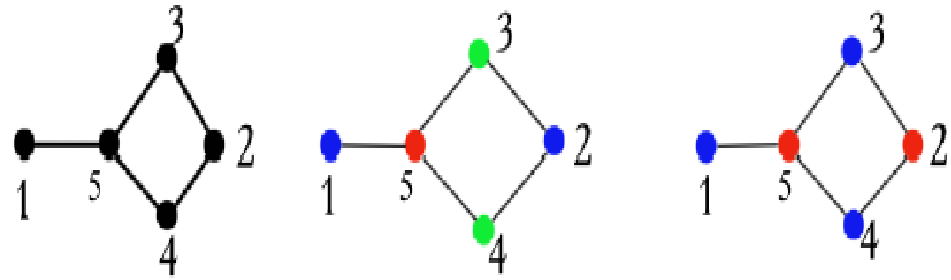


Рис. 3.141. Варіанти розфарбування графа

Але «жадібний» алгоритм, ґрунтуючись на нумерації вершин, зафарбує в синій колір вершини 1 і 2, для розфарбування графа тепер потрібно 3 фарби.

Приклад роботи «жадібного» алгоритму розфарбування

Розглянемо граф *G* (рис. 3.142) та застосуємо до нього «жадібний» алгоритм розфарбування.

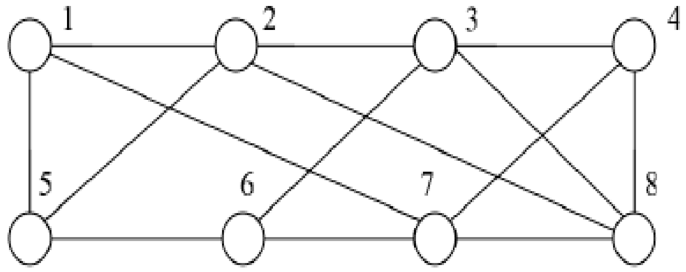


Рис. 3.142. Граф для розфарбування жадібним алгоритмом

Матриця суміжності *A* має такий вигляд:

$$A = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Спочатку розфарбування вибираємо вершину з номером 1 та розфарбовуємо її в колір 1 (червоний). Далі відбувається пошук несуміжної вершини з вершиною 1. Якщо така вершина знайдена, то вона також розфарбовується в колір 1 (червоний). Наступна знайдена для розфарбування кольором 1 вершина повинна бути не суміжною з двома попередніми. Процес продовжується до того часу, поки всі можливості розфарбувати вершини кольором 1 будуть вичерпані. Після цього вибираємо фарбу кольору 2 (синя) і розфарбовуємо нею вершину з мінімальним номером, яка є не розфарбованою до цього часу. Наступна придатна для розфарбування фарбою 2 вершина повинна бути не суміжною з вершиною, яка була розфарбована кольором 2 (синій) на попередньому кроці. Процес розфарбування фарбою 2 також продовжується до того часу, поки не будуть вичерпані всі можливості розфарбування вершин цією фарбою.

Перед вибором чергової фарби для розфарбування завжди перевіряємо, чи залишилися ще не розфарбовані вершини. Якщо такі вершини знайдено, то вибираємо чергову фарбу і продовжуємо процес розфарбування. Якщо ж всі вершини графа розфарбовано, то процес розфарбування жадібним алгоритмом закінчується. Результат розфарбування «жадібним» алгоритмом графа *G* показано на рисунку 3.143.

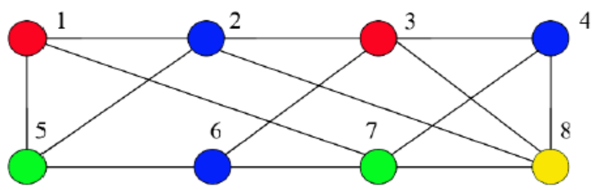


Рис. 3.143. Результат розфарбування графа *G* «жадібним» алгоритмом

<u>Обов’язкові завдання</u>

1. Які переваги і недоліки алгоритму прямого неявного перебору?
2. На яких базових принципах ґрунтується евристичний алгоритм розфарбування?
3. Опишіть у вигляді псевдокоду принцип роботи «жадібного» алгоритму розфарбування графа.
4. Які переваги і недоліки рекурсивного алгоритму розфарбування графів?

3.12. Шляхи і цикли Ейлера. Плоскі та планарні графи

Визначення. Нехай $G = (V, E)$ – граф. Цикл, який включає всі ребра і вершини графа G, називають *циклом Ейлера.* Якщо ця умова виконується, говорять, що граф G має цикл Ейлера.

Якщо тепер повернутися до задачі про кенігсберзькі мости, легко переконатися, що вона зводиться до спроби визначити, чи містить граф, що ілюструє задачу, цикл Ейлера. Для цього нам буде потрібна наведена нижче теорема. Ця теорема справедлива також для мультиграфів. До того ж доведення теореми залишається тим самим. Для ясності будемо використовувати термін граф, розуміючи, що кожне твердження справедливе для мультиграфів.

Теорема 3.12.1. Граф з більше ніж однією вершиною має цикл Ейлера тоді й тільки тоді, коли він зв'язний і кожна його вершина має парний степінь.

Приклад 3.48. Наведений на рис. 3.144 граф має цикл Ейлера, оскільки степінь кожної його вершини парна.

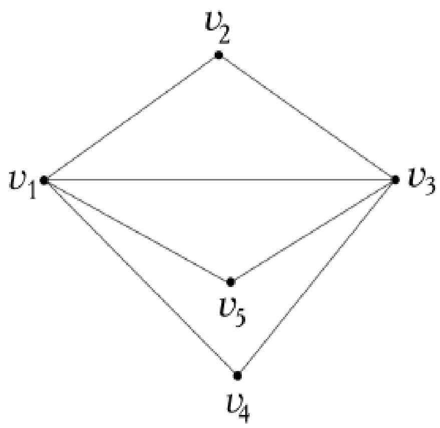


Рис. 3.144. Граф із циклом Ейлера

Приклади циклів Ейлера:

$(v_1, v_2, v_3, v_4.v_1, v_5, v_3, v_1)$, $(v_4, v_3, v_2, v_1, v_3, v_5, v_1, v_4)$ і т. д.

Повертаючись до задачі про кенігсберзькі мости, виявляємо, що мультиграф, побудований Ейлером для опису кенігсберзьких мостів, має непарні степені всіх його вершин.

Отже, цей мультиграф не має циклу Ейлера, тому неможливо пройти кожний міст по одному разу й повернутися в початкову точку шляху у графі, показаному на рис. 3.145.

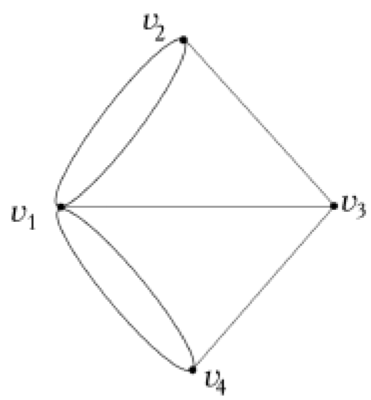


Рис. 3.145. Граф кенігсберзьких мостів

Як видно з рисунка, задача про кенігсберзькі мости була розв'язана з використанням мультиграфа. Однак цю задачу можна розв'язати, використовуючи простий граф. Для цього початковий мультиграф потрібно привести до простого графа шляхом введення додаткових вершин (рис. 3.146):

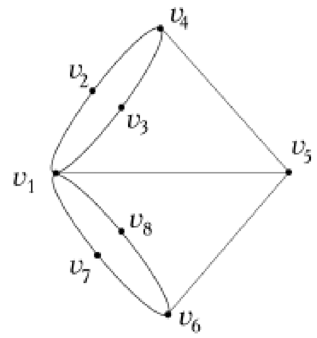


Рис. 3.146. Простий граф кенігсберзьких мостів

Якщо поставити задачу не повертатися в початкову точку, то попередня задача зводиться до пошуку шляху в графі.

Визначення. Нехай G (V, E) – граф. Шлях, який включає кожне ребро графа G тільки один раз, називають *шляхом Ейлера*. У цьому випадку говорять, що граф G має шлях Ейлера.

У випадку, коли шлях Ейлера не є циклом Ейлера, його називають власним шляхом Ейлера.

Теорема 3.12.2. Граф або мультиграф має власний шлях Ейлера тоді й тільки тоді, коли він зв'язний і рівно дві його вершини мають непарний степінь.

Оскільки граф для кенігсберзьких мостів має чотири вершини з непарними степенями, можна зробити висновок про неможливість пройти кожний міст по одному разу, навіть якщо не потрібно повертатися в початкову точку маршруту.

Приклад 3.49. На рис. 3.147 показано граф, який має шлях Ейлера, оскільки дві його вершини мають непарний степінь.

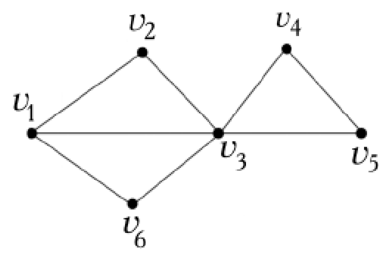


Рис. 3.147. Граф із двома непарними степенями

Приклад шляху Ейлера: $v_1, v_2, v_3, v_1, v_6, v_3, v_4, v_5, v_3$.

Цикл Ейлера в орієнтованому графі

Визначення. Нехай *G* (*V*, *E*) – орієнтований граф. **Орієнтованим циклом** називають орієнтований шлях ненульової довжини з вершини в ту саму вершину без повторення ребер.

Визначення. Нехай *G* (*V*, *E*) – орієнтований граф. Орієнтований цикл, який включає всі ребра й вершини графа *G*, називають **циклом Ейлера**. У цьому випадку говорять, що орієнтований граф *G* має цикл Ейлера.

Теорема 3.12.3. Орієнтований граф має цикл Ейлера тоді й тільки тоді, коли він зв'язний і напівстепінь входу кожної вершини дорівнює її напівстепені виходу.

Алгоритм побудови циклу Ейлера

1. Виходимо з довільно вибраної вершини *v* і кожне пройдене ребро закреслюємо.

2. Ніколи не йдемо по тому ребру *e*, яке в цей момент є мостом (тобто з видаленням якого граф, утворений незакресленими ребрами, розпадається на два компонента зв'язності, що мають хоча б по одному ребру).

3. Не вибираємо ребро, що веде в *v*, поки є інші можливості.

Цикли Гамільтона (основні визначення)

Визначення. Гамільтоновим ланцюгом графа називають його простий ланцюг, який проходить через кожну вершину графа точно один раз.

Визначення. Цикл графа, що проходить через кожну його вершину, називають **гамільтоновим циклом**.

Визначення. Граф називають **гамільтоновим**, якщо він має гамільтонів цикл.

Визначення. Граф, який містить простий шлях, що проходить через кожну його вершину, називають **напівгамільтоновим**.

Це визначення можна поширити на орієнтовані графи, якщо шлях вважати орієнтованим. Гамільтонів цикл не обов'язково містить усі ребра графа. Зрозуміло, що гамільтоновим може бути тільки зв'язний граф і що будь-який гамільтонів граф є напівгамільтоновим.

Помітимо, що гамільтонів цикл існує далеко не в кожному графі.

Зауваження. Будь-який граф *G* можна перетворити в гамільтонів граф, додавши достатню кількість вершин. Для цього, наприклад, достатньо до вершин $v_1,\ldots,v_p$ графа *G* додати вершини $u_1,\ldots,u_p$ і множини ребер $\{v_i,u_i\}$ та $\{u_i,v_{i+1}\}$.

Степенем вершини *v* називають число ребер deg(*v*), інцидентних їй, при цьому петля не враховується. У випадку орієнтованого графа розрізняють степінь $\deg^+(v)$ за вихідними дугами і $\deg^-(v)$ – за вхідними.

Теорема 3.13.1. Нехай *G* має $p \geq 3$ вершин. Якщо для будь-якого *n*, $1 \leq n \leq \frac{p-1}{2}$, кількість вершин зі степенями, що не перевищують *n*, менше ніж *n*, і для непарного *p* кількість вершин степеня $\frac{p-1}{2}$ не перевищує $\frac{p-1}{2}$, то *G* – гамільтонів граф.

Наслідок 1. Якщо $p \geq 3$ і deg(*u*) + deg (*v*) ≥ *p* для будь-якої пари *u* і *v* несуміжних вершин графа *G*, то *G* – гамільтонів граф.

Наслідок 2. Якщо *p* > 3 і $\deg(v) \geq \frac{p}{2}$ для будь-якої вершини *v* графа *G*, то *G* – гамільтонів граф.

Теорема 3.13.2. У повному графі *G* (*V*) завжди існує гамільтонів шлях.

Плоскі та планарні графи.

Загальні поняття про плоский граф

У визначенні графа як геометричної фігури до цього ми не накладали жодних обмежень на розташування фігури в просторі. Тепер же будемо говорити, що граф зображений на поверхні (площини, сфері тощо), якщо всі його **вершини й ребра належать цій поверхні**.

Визначення. Граф, зображений на площині, називають **плоским**, якщо його ребра не перетинаються в точках, відмінних від вершин графа.

Відзначимо, що властивість графа **бути** *або* **не бути плоским** – це властивість геометричного зображення, а не алгебраїчного об’єкта.

На рис. 3.148 показані графи G_1 і G_2, які є ізоморфними, але G_1 – плоский, а G_2 – неплоский.

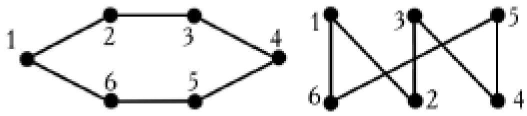


Рис. 3.148. Ізоморфні графи

Загальні поняття про планарний граф

Таким чином, термін «плоский граф» завжди належить до **конкретного** (одного з багатьох) геометричного зображення графа.

Той самий граф (як множина вершин + множина ребер) може мати **як плоскі, так і неплоскі зображення**.

Водночас принципове питання, на яке потрібно відповідати під час розв’язування задач типу прокладки комунікацій: «Чи має цей граф хоча б одне плоске зображення?». Визначимо клас графів, для яких відповідь на це питання позитивна.

Визначення. Граф називають **планарним**, якщо він ізоморфний плоскому графу. Інакше кажучи, **планарним графом** називають граф, який може бути зображений на площині так, що його ребра не перетинаються.

Укладання графа на поверхні

Про планарні графи говорять, що вони мають **плоске укладання** або що вони **укладаються на площині**.

Можна визначити укладання графів не тільки на площині, але й на **інших поверхнях** у просторі.

Властивість графа укладатися на поверхні, безумовно, **залежить від виду цієї поверхні**.

Однак багато поверхонь із огляду на укладання графів нічим **не відрізняються від площини**.

Жорданова крива

Жордановою кривою на площині називають неперервну криву лінію без самоперетинань.

Замкненою жордановою кривою називають жорданову криву, початок і кінець якої збігаються.

Теорема Жордана. Якщо S – замкнена жорданова крива на площині, а x та y – дві різні точки цієї кривої, то будь-яка жорданова крива, що з'єднує точки x і y, або повністю лежать всередині S, крім точок x і y, або поза кривою S, крім точок x і y, або перетинає криву S у деякій точці, відмінній від точок x і y.

На рис. 3.149 показано три варіанти взаємного розташування жорданових кривих.

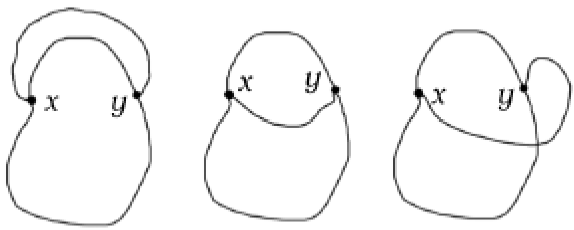


Рис. 3.149. Жорданові криві

Будемо говорити, що граф G укладається в просторі L, якщо існує взаємно однозначне відображення вершин графа G у точки і ребер у жорданові криві цього простору таке, що криві, які відповідають різним ребрам, перетинаються в інцидентних цим ребрам вершинах. Зображений у такий спосіб граф G у просторі L називають **укладанням графа** G.

Теорема про укладання графа в тривимірному просторі

Теорема 3.14.1. Будь-який граф укладається в тривимірному просторі.

Доведення. Розмістимо всі вершини графа $G = (V, E)$ на осі OX. З пучка площин, що проходять через цю вісь, виберемо $|E|$ різних площин. Далі кожне ребро $(u, v) \in E$ зобразимо у відповідній площині півколом, що проходить через вершини *u* і *v*, як показано на рис. 3.150. Ясно, що в підсумку одержимо укладання графа *G* у тривимірний простір, оскільки всі ребра лежать у різних площинах і тому не перетинаються в жодних точках, крім вершин.

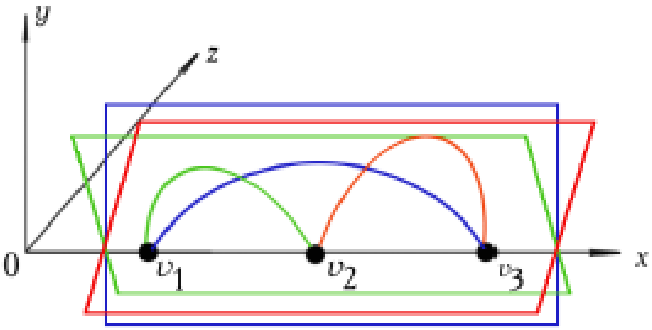


Рис. 3.150. Граф із ребрами в різних площинах

Непланарні графи

Теорема 3.15.1. Графи K_5 (рис. 3.151) і $K_{3,3}$ не є планарними.

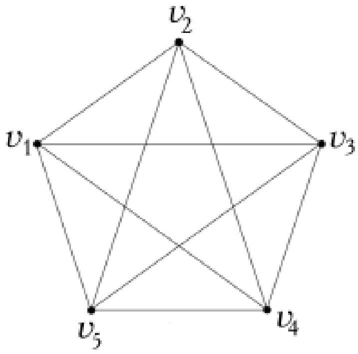


Рис. 3.151. Граф K_5

Доведення. Припустимо протилежне, що граф K_5 планарний. Оскільки він має цикл довжини 5, наприклад $(v_1, v_2, v_3, v_4, v_5)$, то можна вважати, що за будь-якого укладання цього графа на площину цей цикл зображується правильним п’ятикутником. За теоремою Жордана ребро (v_1, v_3) має лежати або цілком усередині цього п’ятикутника, або цілком поза ним. Третю можливість (коли ребро має загальну точку з п’ятикутником) ми не розглядаємо, оскільки маємо справу з укладанням на площину.

1. Розглянемо спочатку випадок, коли (v_1, v_3) лежить усередині п’ятикутника. Оскільки ребра (v_2, v_4)і (v_2, v_5) не мають перетинати ребро (v_1, v_3), то обидва ці ребра лежать поза п’ятикутником. Ця ситуація зображена на рис. 3.152.

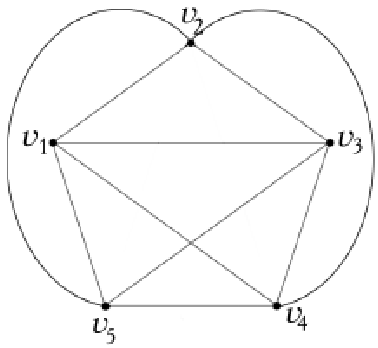


Рис. 3.152. Спроба уникнути перетину з ребром (v_1, v_3)

Ребро (v_1, v_4) не має перетинати ребро (v_2, v_5) і тому воно має лежати усередині п’ятикутника. Аналогічно ребро (v_3, v_5) має лежати усередині п’ятикутника, оскільки не має перетинатися з ребром (v_2, v_4). Однак ребра (v_1, v_4) і (v_3, v_5) обов’язково перетинаються, тому, згідно з теоремою Жордана, одне з них має лежати поза п’ятикутником.

Таким чином, ми приходимо до протиріччя з нашим припущенням. Наявність цього протиріччя доводить помилковість початкового припущення, тобто граф K_5 не є планарним.

Аналогічно доводять, що граф $K_{3,3}$ не є планарним, оскільки неможливо побудувати ребро (v_1, v_4) без перетину з іншим ребром графа $K_{3,3}$ (рис. 3.153).

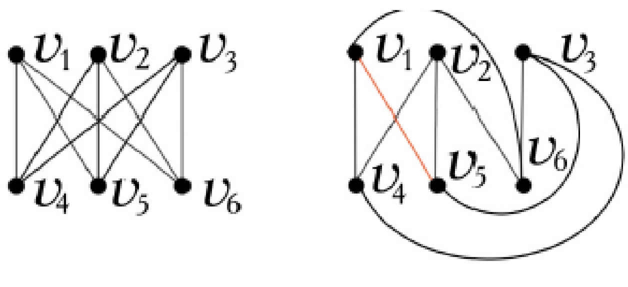


Рис. 3.152. Доведення того, що граф $K_{3,3}$ не є планарним

Грані плоского графа

Точку *x* площини *S*, на якій розміщено граф *G*, називають **диз'юнктною** з *G*, якщо ця точка не відповідає жодній вершині графа *G* і не лежить на жодному ребрі цього графа. Дві точки *x* і *y* площини *S* називають еквівалентними, якщо вони диз'юнктні з *G* і їх можна з'єднати такою жордановою кривою, усі точки якої диз'юнктні з *G*.

Введене відношення еквівалентності точок площини розбиває множину всіх точок площини *S* на класи еквівалентності, які називають гранями графа *G* . Наприклад, граф *G*, зображений на рис. 3.153, має 4 грані: f_1, f_2, f_3, f_4. Причому грань f_1 – нескінченна.

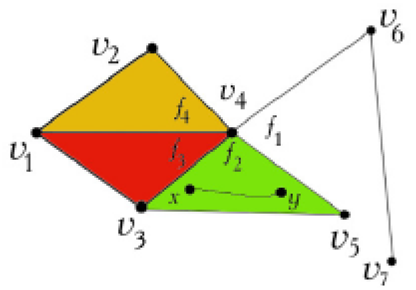


Рис. 3.153. Грані графа *G*

Теорема Ейлера

Теорема 3.15. 1. Нехай *G* – зв'язний плоский граф, у якому *n* – число вершин, *m* – число ребер і *f* – число граней. Тоді справедливе співвідношення: $n + f = m + 2$.

Доведення. План доведення ґрунтується на виконанні індукції за числом ребер у графі *G*.

1. Нехай $m = 0$, то $n = 1$ (оскільки за умовою теореми *G* – зв'язний) і $f = 1$ (нескінченна грань). У цьому випадку теорема вірна, оскільки $n + f = 1 + 1 = 2$ і $m + 2 = 0 + 2 = 2$. Приклад цього графа показаний на рис. 3.154.

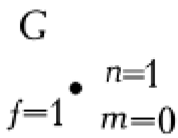


Рис. 3.154. Зв'язний граф *G* з параметрами $n = 1$, $m = 0$, $f = 1$

2. Нехай граф представлений такими параметрами:

Ребро e_1 є петлею, і в цьому випадку число граней збільшується на одну, а число вершин залишається незмінним. Граф з однією вершиною і двома гранями показаний на рис. 3.155.

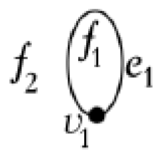


Рис. 3.155. Зв’язний граф *G* з параметрами *n* = 1, *m* = 1, *f* = 2

Для розглянутого графа *G* = (*V*, *E*) одержуємо $V = \{v_1\}$, $E = \{e_1\}$, $F = \{f_1, f_2\}$.

Тому $n = |V| = 1$, $m = |E| = 1$, $f = |F| = 2$.

Звідси $n + f = 1 + 2 = 3$ і $m + 2 = 1 + 2 = 3$.

3. Нехай граф *G* представлений такими параметрами:

ребро e_1 з’єднує дві різні вершини в *G*. Граф з двома вершинами і одним ребром показаний на рис. 3.156.

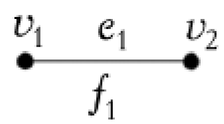


Рис. 3.156. Зв’язний граф *G* з параметрами *n* = 2, *m* = 1, *f* = 1

У цьому графі *G* = (*V*, *E*) одержуємо $V = \{v_1, v_2\}$, $E = \{e_1\}$,. $F = \{f_1\}$.

Тому $n = |V| = 1$, $m = |E| = 1$, $f = |F| = 1$.

Звідси $n + f = 2 + 1 = 3$ і $m + 2 = 1 + 2 = 3$.

4. Нехай зв’язний граф *G* містить кілька вершин і ребер. Граф з трьома вершинами і трьома ребрами показаний на рис. 3.157.

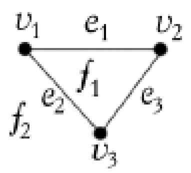


Рис. 3.157. Зв’язний граф *G* з параметрами *n* = 3, *m* = 3, *f* = 2

Граф $G = (V, E)$ характеризується параметрами $V = \{v_1.v_2, v_3\}$, $E = \{e_1, e_2, e_3\}$, $F = \{f_1, f_2\}$.

Тому $n = |V| = 3$, $m = |E| = 3$, $f = |F| = 2$.

Звідси $n + f = 3 + 2 = 5$ і $m + 2 = 3 + 2 = 5$.

5. До графа, показаного на рис. 3.157, додамо петлю. У підсумку одержимо граф, показаний на рис. 3.158.

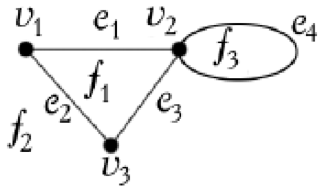


Рис. 3.158. Зв'язний граф G з параметрами $n = 3$, $m = 3$, $f = 3$

Граф $G = (V, E)$ характеризується параметрами $V = \{v_1.v_2, v_3\}$, $E = \{e_1, e_2, e_3, e_4\}$, $F = \{f_1, f_2, f_3\}$. Тому $n = |V| = 3$, $m = |E| = 4$, $f = |F| = 3$. Отже, $n + f = 3 + 3 = 6$ і $m + 2 = 4 + 2 = 6$.

6. Нехай зв'язний граф G_1 містить 4 вершини і 4 ребра (рис. 3.159, *а*).

Цей граф має такі параметри:

$$V = \{v_1.v_2, v_3, v_4\},\ E = \{e_1, e_2, e_3, e_4\},\ F = \{f_1, f_2\}.$$

Побудуємо граф G_2 (рис. 3.159 б), додавши ребро, що з'єднує дві його вершини.

$$V = \{v_1.v_2, v_3, v_4\},\ E = \{e_1, e_2, e_3, e_4, e_5\},\ F = \{f_1, f_2, f_3\}.$$

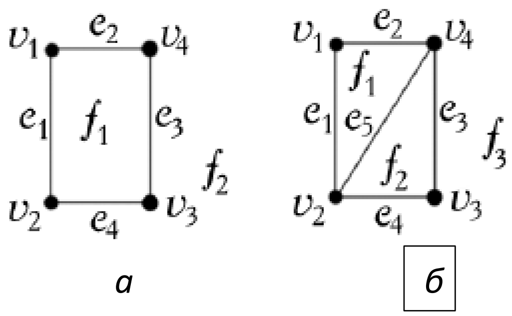


Рис. 3.159. Граф G_1 (*а*) і G_2 (*б*)

Параметри нового графа $n = 4$, $m = 5$, $f = 3$.

Отже, $n + f = 4 + 3 = 7$, $m + 2 = 5 + 2 = 7$.

7. Якщо до зв'язного графа G_2 (рис. 3.159 б), який має такі параметри $V = \{v_1.v_2, v_3, v_4\}$, $E = \{e_1, e_2, e_3, e_4, e_5\}$, $F = \{f_1, f_2, f_3\}$ додати ребро,

інцидентне тільки з однією з його вершин, то одержимо граф із параметрами (рис. 3.160):

$$V = \{v_1.v_2, v_3, v_4, v_5\},\ E = \{e_1, e_2, e_3, e_4, e_5, e_6\},\ F = \{f_1, f_2, f_3\}$$

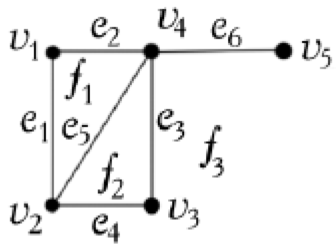


Рис. 3.160. Зв'язний граф G з параметрами $n = 5$, $m = 6$, $f = 3$

Одержуємо $n + f = 5 + 3 = 8$, $m + 2 = 6 + 2 = 8$.

Наслідок. Нехай – плоский граф з n вершинами, m ребрами, f гранями і k компонентами зв'язності; тоді $n + f = m + k + 1$.

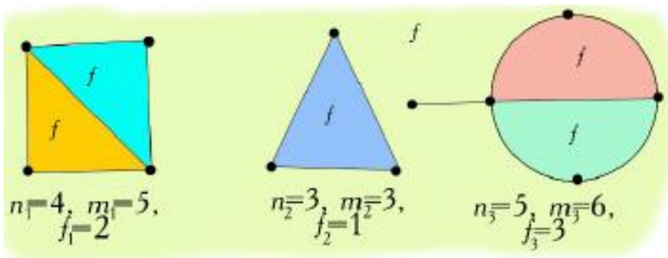


Рис. 3.161. Незв'язний граф G з параметрами $n = 12$, $m = 14$, $f = 6$

$$n = n_1 + n_2 + n_3 = 4 + 3 + 5 = 12$$

$$m = m_1 + m_2 + m_3 = 5 + 3 + 6 = 14$$

$$f = f_1 + f_2 + f_3 = 2 + 1 + 3 = 6$$

$$n + f = 12 + 6 = 18$$

$$m + k + 1 = 14 + 3 + 1 = 18$$

Наслідок. Якщо G – зв'язний планарний граф (рис. 3.162) з m ребрами і $n \geq 3$ вершинами, то $m \leq 3n - 6$.

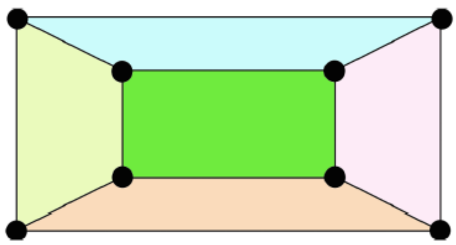


Рис. 3.162. Зв'язний граф з параметрами $n = 8$, $m = 12$

$$n = 8 > 3,\ m = 12$$
$$m < 3n - 6,\ m < 3 \times 8 - 6,\ m < 24 - 6{,}121 < 18$$

Наслідок. У будь-якому планарному графі існує вершина, степінь якої не більше п’яти (рис. 3.163).

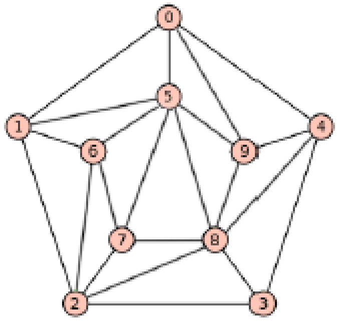


Рис. 3.163. Планарний граф

Твердження. Будь-який підграф планарного графа теж планарний.

Твердження. Якщо граф містить непланарний підграф, то і сам граф непланарний.

Наслідок. Будь-який підграф, що включає граф K_5 або $K_{3,3}$, не є планарним.

Виявляється, що графи K_5 (рис. 3.164) і $K_{3,3}$ (рис. 3.165) – **єдині непланарні** графи в тому розумінні, що будь-який непланарний граф містить один із них як підграф.

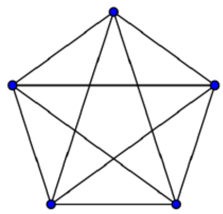


Рис. 3.164. Зв’язний граф K_5

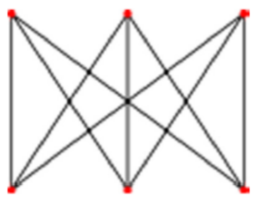


Рис. 3.165. Зв’язний граф $K_{3,3}$

Для формулювання цього результату введемо поняття **гомеоморфності графів.**

Гомеоморфні графи

Визначення 1. Два графа називають **гомеоморфними**, або **тотожними**, з точністю до вершин степеня 2, якщо вони можуть бути отримані з того самого графа за допомогою операції введення вершини степеня 2 у ребро. Наприклад, графи, зображені на рис. 3.166, гомеоморфні.

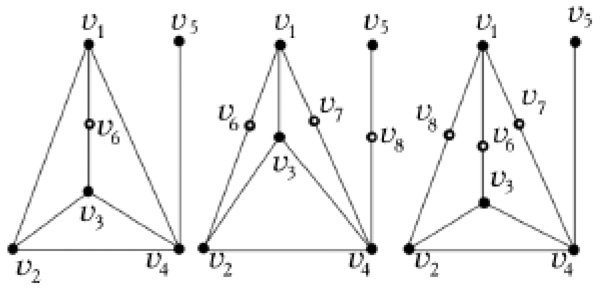


Рис. 3.166. Приклад гомеоморфних графів

Неважко переконатися, що гомеоморфізм є відношенням еквівалентності.

Визначення 2. Два графи називають гомеоморфними, якщо кожен із них може бути одержаний розбиттям певного графа.

Визначення 3. Два графа G_1 і G_2 є гомеоморфними, якщо можливо сформувати таке розбиття графа G_1 і розбиття графа G_2, які ізоморфні між собою.

Визначення 4. Розбиття графа – це граф, який отримуємо послідовним застосуванням до цього графа операції розбиття ребер. Під час розбиття ребро e , яке інцидентне вершинам $\{u, v\}$, замінюємо на вершину w і два ребра $\{u, w\}$ і $\{w, v\}$ (рис. 3.167).

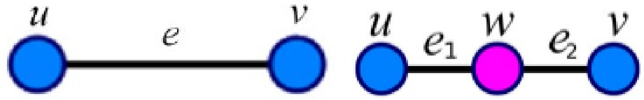


Рис. 3.167. Розбиття ребра e

Теорема Понтрягіна – Куратовського

Граф є планарним тоді й тільки тоді, коли він не має підграфів, гомеоморфних K_5 і $K_{3,3}$.

Доведення. З геометричної позиції додавання вершини степеня 2 – це додавання точки на ребрі, а стирання такої вершини поєднує два ребра із спільним кінцем в одне. Очевидно, що кожна із цих операцій, застосована до плоского графа, знову дасть плоский граф. Отже, за наслідками з теореми Ейлера, жодний плоский (а, отже, і планарний) граф не гомеоморфний графам K_5 і $K_{3,3}$. З урахуванням зауваження про непланарні підграфи теорему доведено.

Операція стягування

Нехай (*u*, *v*) – ребро графа *G*. Вилучимо із графа *G* вершини *u і v*. Після цього додамо в граф *G* нову вершину *w* і з'єднаємо її ребрами з усіма вершинами, з якими була суміжна хоча б одна з вершин *u* і *v* (рис. 3.168).

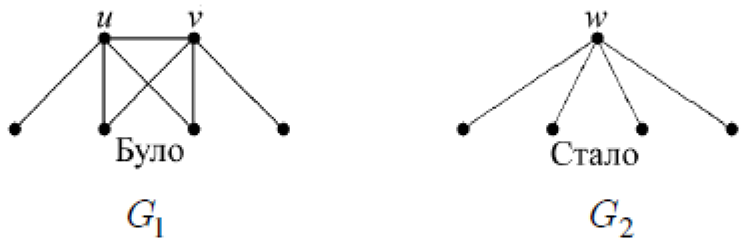


Рис. 3.168. Стягування графа G_1 до графа G_2

Позначимо отриманий граф через G_2. Говорять, що граф G_2 отримано з G_1 **стягуванням ребра** (*u*, *v*).

Якщо скінченним (можливо нульовим) числом операцій стягування ребра із графа G_1 можна одержати граф G_2, то говорять, що G_1 **стягується** до G_2.

Теорема Вагнера

Теорема. Граф планарний тоді й тільки тоді, коли він не має підграфів, які стягуються графом K_5 і $K_{3,3}$.

Доведення необхідності. Якщо стягти ребро в плоскому графі, він залишиться плоским. Отже, за наслідками з теореми Ейлера, жоден плоский (а отже, і планарний) граф не стягується ні до графа K_5, ні до графа $K_{3,3}$. З урахуванням зауваження про непланарні підграфи необхідність доведено.

Процес стягування до $K_{3,3}$ і K_5 (рис. 3.169).

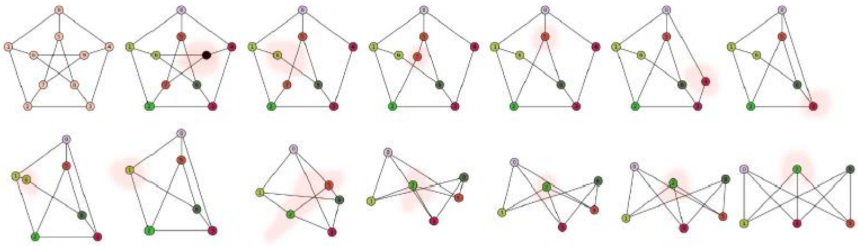


Рис. 3.169. Процес стягування

<u>Обов'язкові завдання</u>

1. Коли існує шлях Ейлера в графі? Поясніть, чому граф для кенігсберзьких мостів не має шляху Ейлера.

2. Дайте визначення гамільтонового графа. Чи існує можливість перетворення негамільтонового графа в гамільтоновий?

3. Чи є повний граф з множиною вершин $V = \{ v_1, v_2, v_3, v_4, v_5 \}$ планарним графом? Якщо так, то виконайте його укладку на площині.

4. Перевірте, чи є планарним граф, який задано такою матрицею суміжності:

$$S = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{array} \begin{array}{ccccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 \\ \end{array}$$

$$\left(\begin{array}{c|ccccccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 \\ \hline v_1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ v_2 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ v_3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ v_4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ v_5 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_6 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ v_7 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ v_8 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ v_9 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}\right)$$

Якщо даний граф планарний, то виконайте його укладку на площину.

5. Визначте число граней графа, заданого матрицею суміжності:

$$S = \left(\begin{array}{c|cccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ \hline v_1 & 0 & 1 & 1 & 1 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 1 & 1 & 1 \\ v_3 & 1 & 1 & 0 & 1 & 0 & 0 \\ v_4 & 1 & 1 & 1 & 0 & 1 & 1 \\ v_5 & 0 & 1 & 0 & 1 & 0 & 1 \\ v_6 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right)$$

6. Нехай дано граф, який містить 10 ребер і 6 граней. Визначте кількість вершин цього графа.

Розділ 4. Математична логіка

4.1. Алгебра висловлювань

Загальні поняття

Логіка – це наука про закони мислення та його форми. Вона як мистецтво суджень бере свій початок із далекої давнини. У логіку було впроваджено математичну символіку, і сьогодні вона використовує мову й методи математики. Звідси й назва – математична логіка.

Основи математичної логіки було закладено в середині XIX сторіччя ірландським математиком Дж. Булем. Останніми десятиліттями логіка набула широкого застосовування в техніці під час дослідження й розроблення електронних схем, обчислювальних машин, дискретних автоматів. Вона використовується також і в інших науках: економіці, біології, психології тощо.

Мова алгебри висловлювань

Мова алгебри висловлювань – це штучна мова, призначена для аналізу логічної структури складних висловлювань. Вона характеризується алфавітом (списком знакових засобів) і визначенням формули.

Під алфавітом будемо розуміти кожну не порожню множину символів:

пропозиційних змінних – $x, y, z, x_1, x_2, \ldots$; логічних зв'язок $\neg, \wedge, \vee, \rightarrow, \sim, \ldots$;

технічних символів – (,) тощо.

Словом у певному алфавіті називається довільна скінчена послідовність символів (можливо, порожня). Слово a називається підсловом b, якщо $b = b_1 a b_2$ для певних слів b_1 та b_2. Слово ab називається сполученням (конкатенацією) слів a та b.

Висловлюванням називають осмислений вираз звичайної мови, якому можна приписати значення істинності. Таким виразом може стати твердження або розповідне речення, про яке можна сказати, істинне воно або хибне. Значення істинності – це абстрактний об'єкт, що ставиться у відповідність висловлюванню: істина – коли висловлювання відповідає дійсності, хибність – коли висловлювання не відповідає дійсності. Позначають: «Істина» – I, T (True), або 1; «Хибність» – X, F (False) або 0.

Приклад 4.1. Визначити, які із цих речень є висловлюваннями: «Дніпро впадає в Чорне море»; «Дніпро впадає в Азовське море»; «Хто ви?»; «Відстань від Землі до Сонця дорівнює 150 млн км».

Розв'язок. Перші два речення є висловлюваннями, причому перше є істинним, а друге – хибним висловлюванням. Третє речення не є висловлюванням, оскільки воно не розповідне. Четверте речення також не є висловлюванням. Його істинність або хибність залежить від потрібної точності.

Атомом (елементарним висловлюванням) називається таке висловлювання, яке є простим розповідним реченням, тобто не має складових частин. Для позначення атомів використовують як символи букви латинського алфавіту з індексами або без них: A, B, p, q, r, s, p1, q2. Складні речення зазвичай складаються з простих речень, поєднаних сполучниками. Тобто прості речення, які представляють атоми та сполучники, є елементами словника, необхідного для формалізації природної мови за допомогою логіки висловлювання. Значення істинності складного висловлення визначається значеннями істинності його складових частин.

A, B, C, D, A1, B1... – позначають масиви висловлювань (схеми формул).

Поміж висловленнями використовуються різні логічні зв'язки: «якщо... то...», «... або ...», «... і ...» тощо. За їхньої допомоги будуються інші нові висловлення.

Логічною операцією називається операція, у якій операндами є висловлення, а операторами – логічні зв'язки.

За допомогою алгебри логіки можна, наприклад, описувати роботу релейно-контактних схем. Для конкретики обмежимося розгляданням двополюсних схем, у яких поміж полюсами можуть існувати релейні контакти, з'єднані послідовно чи паралельно. При цьому стан контакту – 1 (0) означає, що він замкнений (розімкнений), тобто сигнал 1 (0) переводить електронний елемент у відкритий (закритий) стан.

Розглянемо спочатку схеми з одним контактом (рис. 2.1), на яких сам контакт та його стан позначено через x, а стан двополюсника позначатимемо літерою y.

Рис. 4.1. Схеми з одним контактом

Вочевидь, змінна x є незалежною, а змінна y – залежною булевою змінною.

У разі першої схеми коло буде замкнене, якщо буде змінено стан контакту (замкнено), тобто змінна y набуде істинного значення (y = 1) тоді й тільки тоді, коли змінна x також набуде істинного значення (x = 1). У разі другої схеми, навпаки, змінна y набуде істинного значення (y = 1), коли змінна x збереже хибне значення, тобто стан контакту не зміниться (x = 0).

Перейдемо тепер до розглядання схеми «або» і схеми «і» (рис. 4.2).

Рис. 4.2. Схеми з двома контактами

Якщо контакти x_1 та x_2 з’єднані паралельно, то коло буде замкнене, тобто змінна y набуде істинного значення (y = 1), коли хоча б один з контактів x_1 та x_2 є замкненим, і розімкненим, тобто y набуватиме хибного значення, коли обидва контакти x_1 та x_2 є розімкненими.

За послідовного з’єднання контактів x_1 та x_2 коло буде замкнене (y = 1), коли обидві змінні x_1 та x_2 набуватимуть істинного значення (тобто x_1 = 1, x_2 = 1), і розімкнене (y = 0), коли хоча б одна зі змінних x_1 та x_2 набуде хибного значення.

Опис більш складних релейно-контактних схем здійснюється за допомогою булевих функцій (див. п. 4.2).

Для логічних зв’язок застосовуються символи: $\neg$ – для «не» (заперечення», $\wedge$ – для «і» (кон’юнкції), $\vee$ – для «або» (диз’юнкції), $\rightarrow$ – для «якщо… то…» (імплікації), $\sim$ – для «тоді й лише тоді, коли» (еквіваленції). Наприклад, якщо A та B – твердження, то $\overline{A}$, $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \sim B$ будуть запереченням твердження A, кон’юнкцією тверджень A та B тощо.

Задамо значення істинності висловлень A та B таблицею істинності, тоді для розглянутих зв'язок таблиця істинності має такий вигляд:

A	B	$\overline{A}$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \sim B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Отже, якщо значення істинності простих висловлень відомі, то значення істинності складних висловлень може бути визначено за допомогою цих таблиць.

Формули алгебри висловлень. Семантика. Класифікація та рівносильність формул

Алгебра висловлювань – це алгебраїчна структура [{І, Х}, $\vee$, $\wedge$, $\neg$, $\rightarrow$, ~, Х, І] з носієм –двійковою множиною {Х: «Хибність», І: «Істина»}, операціями – логічними зв'язками: кон'юнкція, диз'юнкція, заперечення, імплікація, еквівалентність і константами: Х – хибність і І – істина. Операції $\vee$, $\wedge$, $\rightarrow$, ~ є бінарними логічними зв'язками, а операція $\neg$ – унарною. Складні висловлювання, які побудовані з простих висловлювань, називають **формулами** або **молекулами**.

Логічні зв'язки можна розглянути як формальні позначення зв'язок, що їм відповідають.

Алгебра логіки (алгебра висловлень) являє собою науку про сукупність висловлень, над якими визначені логічні операції.

Формулою алгебри висловлень (ФАВ) називається слово, яке задовольняє такому визначенню:

1) кожна пропозиційна змінна – формула;

2) якщо A та B – формули, то ($\overline{A}$), ($A \wedge B$), ($A \vee B$), ($A \rightarrow B$), ($A \sim B$) – формули;

3) слово є формулою, якщо воно побудоване лише з використанням п. 1 та п. 2.

Наприклад, вирази (слова)

$((A \rightarrow B) \vee A)$, $((A \sim B) \rightarrow (C))$

є формулами, а слова ($A \sim B$), ($\overline{C}$), A, B, C – підформули останньої формули.

Назва	Позначення	Аналог природної мови
заперечення	¬	“ ні”, “ неправильно, що”
диз’юнкція	∨	“або”
кон’юнкція	∧	“і”
імплікація	→	“ якщо ...,то ”
еквівалентність	~ , ↔, ⇔	“ еквівалентно”, “рівносильно”, “... тоді і тільки тоді ”, “ якщо і тільки якщо ”

З метою економії дужок операції виконуються в такому порядку (пріоритет операцій): ¬, ∧, ∨, →, ~.

Формули логіки висловлювань, які відповідають складним висловлюванням, приймають значення І і Х залежно від значень елементарних висловлювань і логічних зв’язків, з яких вони побудовані.

Формули логіки висловлювань зручно представляти таблицями істинності, які надають значення істинності формули залежно від послідовного перебору всіх можливих значень істинності простих висловлень, що складають формулу:

A	B	¬A	A∧B	A∨B	A→B	A~B
X	X	I	X	X	I	I
X	I	I	X	I	I	X
I	X	X	X	I	X	X
I	I	X	I	I	I	I

Для позначення істинного висловлення використовують літеру *i* (істина, чи цифру 1, чи T – true), а для хибного – літеру *x* **(**хибність, чи цифру 0, чи F –false).

Висловлювання A ∧ B називають **кон’юнкцію** висловлювання A і B, яке істинне тоді й тільки тоді, коли істинні обидва висловлювання A і B. Це логічна операція відповідає у природній мові зв’язці «і», що з’єднує два речення.

Приклад 4.2. Записати у вигляді формули логіки висловлювань і визначити істинне значення таких висловлювань:

1. «8 ділиться на 4 і 8 більше 6»; 2. «8 ділиться на 4 і 7 більше 8».

Розв’язок. У цих висловлюваннях виділимо атоми. Їх три: A – «8 ділиться на 4», B – «8 більше 6», C– «7 більше 8».

1) $A \wedge B = I \wedge I = I$;

2)$A \wedge C = I \wedge X = X$.

Висловлювання $A \vee B$ називають **диз’юнкцією** висловлювань A або B, яке дійсне тоді й тільки тоді, коли істинне хоча би одне логічне висловлювання A або B. Ця логічна операція відповідає у природній мові зв’язці «або», що з’єднує два речення.

Приклад 4.3. Записати у вигляді формули логіки висловлювань і визначити істинні значення таких висловлювань:

1. «$3 + 2 = 6$ або $3 \times 2 = 6$»;

2. «$4 – 2 = 3$ або $4 \times 2 = 7$».

Розв’язок. У цих висловлюваннях виділимо атоми: A – «$3 + 2 = 6$»; B – «$3 \times 2 = 6$»; C – «$4 – 2 = 3$»; D – «$4 \times 2 = 7$».

1) $A \vee B = X \vee I = I$;

2) $C \vee D = X \vee X = X$.

Слабка/нестрога диз’юнкція – це логічний сполучник, який є хибним лише в одному випадку, коли логічні значення його складників хибні. В усіх інших випадках цей логічний сполучник істинний.

Приклад 4.4. Розглянемо таке висловлювання: «Ця людина – викладач або письменник». Це висловлювання складене з двох простих:

A = «Ця людина – викладач»,

B = «Ця людина – письменник». Висловлювання $A \vee B$ може бути оцінене як істинне лише в тих випадках, коли:

– людина є і викладачем, і письменником;

– людина є викладачем, але не є письменником;

– людина не є викладачем, але є письменником.

Хибним це складне висловлювання буде лише тоді, якщо ця людина не є ні викладачем, ні письменником.

Приклад 4.5. Нехай задані висловлювання: А – «Микола полюбляє грати у футбол»; В – «Микола полюбляє грати у волейбол»; С – «Микола полюбляє грати у теніс». Потрібно записати висловлювання «Микола полюбляє грати у футбол, і неправильно, що він полюбляє грати у волейбол або теніс» у вигляді формули логіки висловлювань і побудувати відповідно їй таблицю істинності.

Розв'язок. Висловлювання «Микола полюбляє грати у волейбол або теніс» можна записати у вигляді формули логіки висловлювань як $B \vee C$. Висловлювання «Неправильно, що він полюбляє грати у волейбол або теніс» може бути записане у вигляді формули логіки висловлювань $\neg (B \vee C)$, оскільки заперечення застосовне до всього висловлювання, яке слідує після зв'язки «що...». Виходячи із цього, вихідна форма складного логічного висловлювання матиме вигляд $A \wedge \neg (B \vee C)$.

A	B	C	B∨C	¬(B∨C)	A∧¬(B∨C)
Х	Х	Х	Х	І	Х
Х	Х	І	І	Х	Х
Х	І	Х	І	Х	Х
Х	І	І	І	Х	Х
І	Х	Х	Х	І	І
І	Х	І	І	Х	Х
І	І	Х	І	Х	Х
І	І	І	І	Х	Х

Із таблиці істинності складного висловлювання слідує, що тільки в одному випадку дане логічне висловлювання буде істинне, коли висловлювання А – істинне, а висловлювання В і С – хибні.

На відміну від слабкої (нестрогої) диз'юнкції, **сильна (строга) диз'юнкція** передбачає використання сполучника «або» в строго розмежувальному сенсі – «А або В, але не обидва разом». Для неї в логіці висловлювань вводиться новий символ – V.

Сильна (строга) диз'юнкція – це логічний сполучник, який буде істинним лише в тих випадках, коли логічні значення його складників не збігаються. Цей логічний сполучник буде хибним, коли логічні значення його складників збігаються.

Приклад 4.6. Розглянемо таке висловлювання: «Ця людина народилася в Києві або у Львові». Це висловлювання складене з двох простих:

A = «Ця людина народилася в Києві».

B = «Ця людина народилася у Львові».

Це висловлювання $A \vee B$ може бути оцінене як істинне тільки у тому випадку, якщо одне з простих висловлювань буде істинним, а друге – хибним.

Якщо ж обидва простих висловлювання виявляться одночасно істинними, тоді їхня диз'юнкція з необхідністю буде оцінена як хибна, оскільки неможливо одночасно народитися у двох містах.

Хибною строга диз'юнкція буде й у тому випадку, коли всі прості висловлювання, які її складають, виявляться хибними, тобто якщо людина, про яку йдеться в складному висловлюванні, не народилася ні в Києві, ні у Львові. Через нестрогу диз'юнкцію можна записати: $(A \vee B) \setminus (A \wedge B) = (A \vee B) \neg \wedge (A \wedge B)$.

Приклад 4.7. E = «Сонце світить – Петрик грає у футбол, вітер дме – Петрик читає книжку». Атоми:

A = «Сонце світить»;

B = «Петрик грає у футбол»;

C = «Вітер дме»;

D = «Петрик читає книжку».

Форма складного висловлювання: $E = (A \rightarrow B) \vee (C \rightarrow D)$.

Приклад 4.8. E = «Ваш приїзд не є ні необхідним, ні бажаним».

A = «Ваш приїзд необхідний»;

B = «Ваш приїзд бажаний».

Форма складного висловлювання: $E = \neg A \neg \wedge B$.

Приклад 4.9. E = «Пошуки ворога тривали вже 3 години, проте результату не було, притаївшись, ворог нічим себе не видавав».

A = «Пошуки ворога тривали 3 години»;

B = «Ворога знайшли (був результат)»;

C = «Ворог себе видав».

Форма складного висловлювання: $E = (A \neg \wedge B) \neg \rightarrow C$.

Приклад 4.10. Побудувати фразу:

A = «Дехто є лікарем»;

B = «Хворий поговорив із лікарем»; C = «Хворому стало краще».

Вислів Бехтерєва: «Якщо хворому після розмови з лікарем не стало краще, то це не лікар: E = (B ¬ ∧ C) ¬ → A.

Раніше зазначалося, що висловлення може бути чи то істинне, чи хибне.

Інтерпретувати формулу – означає приписати їй одне з двох значень істинності. Набір правил інтерпретації формул (семантика числення висловлень) має бути композиційним, тобто значення формули має бути функцією значень її складових. Для інтерпретації можна використовувати таблиці істинності.

Наприклад, покажемо істинність формули $A \to B \sim \overline{A} \vee B$ за будь-яких інтерпретацій:

A	B	$A \to B$	$\overline{A}$	$\overline{A} \vee B$	$A \to B \sim \overline{A} \vee B$
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	1	1	0	1	1

Якщо існує хоча б одна інтерпретація, за якої формула набуває істинного значення, вона називається **здійсненною**, а якщо існує хоча б одна інтерпретація, за якої формула набуває хибного значення, вона називається ***спростовною.***

Серед формул алгебри висловлювань особливе місце мають ті формули A ($p_1, p_2, ..., p_n$), які на всіх наборах ($a_1, a_2, ..., a_n$) значень своїх змінних набувають значення 1.

Формула називається **тавтологією** (чи тотожно істинною, чи загальнозначущою), якщо за будь-яких інтерпретацій її складових (змінних) вона набуває істинного значення. Позначення тавтології є | = *A*.

Наведемо приклади деяких важливих тавтологій:

$(p \vee (\neg p))$ (*закон виключення третього*),

$(\neg(p \vee (\neg p)))$ (*закон виключення суперечності*),

$(p \to p)$ (*закон тотожності*).

Формула називається **протиріччям** (тотожно хибною), якщо за будь-яких інтерпретацій вона набуває хибного значення.

Формулу, яка не є ні тавтологією, ні суперечністю, називають **нейтральною**.

Множина всіх формул алгебри висловлювань розбивається на тавтології, суперечності й нейтральні формули. Формула, яка не є суперечністю, називається **виконуваною**.

Областю істинності (областю хибності) формули називається множина наборів значень змінних, за яких формула набуває істинного (хибного) значення.

Результати цих означень:

формула A є тавтологією тоді й лише тоді, коли A не є спростовною;

формула A є тотожно-хибна тоді й лише тоді, коли A не є здійсненною;

формула A є тавтологією тоді й лише тоді, коли $\overline{A}$ є тотожно хибною;

формула A є тотожно-хибна тоді й лише тоді, коли $\overline{A}$ тавтологія;

формула $A \sim B$ – тавтологія тоді й лише тоді, коли A та B набувають однакових значень (є рівносильні) за всіма наборами значень змінних.

Вочевидь, що дві формули є рівносильні тоді й лише тоді, коли за будь-яких інтерпретацій їхніх змінних вони набувають однакових значень. Позначення: $A = B$.

Теорема 4.1. Дві формули A та B є рівносильні тоді й лише тоді, коли $\models A \sim B$.

Доведення. Нехай $P_1, P_2, ..., P_m$ є сукупність усіх простих компонентів в A та B. Якщо цим компонентам надано певні істинні значення, то перша частина обчислення значень формули $A \sim B$ полягає в обчисленні значень A та B, після чого обчислення завершується застосуванням таблиці істинності для еквівалентності, за якою значення $A \sim B$ буде істинним тоді й лише тоді, коли значення для A та B будуть однакові, тобто A та B є рівносильні.

Наслідок. Нехай с − формула, у якій є певні входження формули A, і нехай F_2 – результат заміни цього входження формули A на формулу B.

Тоді:

якщо $\models A \sim B$, то $\models F_1 \sim F2$;

якщо $\models A \sim B$ і $\models F_1$, то $\models F_2$.

Теорема 4.2. Якщо $\models A$ і $\models A \to B$, то $\models B$.

Доведення. Нехай $P_1, P_2, ..., P_m$ є сукупність усіх простих компонентів в A та B. Якщо цим компонентам надано певних значень істинності, то перша частина обчислення значень формули $A \to B$ полягає в обчисленні значень A та B, після чого обчислення завершуються застосуванням таблиці істинності для імплікації.

З припущення |= A та |= $A \rightarrow B$ випливає, що значення A та $A \rightarrow B$ будуть істинними.

З таблиці для $A \rightarrow B$ випливає, що B теж матиме значення «істина». Через то що це має місце для кожних значень компонентів $P_1, P_2, ..., P_m$, формула B є загальнозначуща.

Ці теореми дають здійснювати еквівалентні перетворювання формул і здобувати нові загальнозначущі формули.

Наприклад, тавтології можна здобути з рівносильності заміною знака = на знак ~. Скажімо, з рівносильності $A \vee AB = A$ здобуваємо тавтологію |= $A \vee AB \sim A$.

Доведення тавтології, наприклад, |= $(A \rightarrow B)(A \rightarrow C) \sim (A \rightarrow BC)$, можна виконати за допомогою перетворень:
$(A \rightarrow B)(A \rightarrow C) = (\overline{A} \vee B)(\overline{A} \vee C) = \overline{A} \vee \overline{A}B \vee \overline{A}C \vee BC = \overline{A} \vee BC = A \rightarrow BC$.

Числення висловлювань

У найзагальнішому вигляді **формальну теорію** T (інший термін – **числення**) будують таким чином:

1. Визначають набір основних символів – **алфавіт** теорії.
2. Конструктивно визначають **множину формул**, або правильно побудованих виразів, яка утворює мову теорії.
3. Виокремлюють підмножину формул, які називають **аксіомами** теорії.
4. Задають **правила виводу (виведення**) теорії.

Правило виводу $R(F_1, F_2, ..., F_m, G)$ – це відношення (або операція) на множині формул.

Якщо формули $F_1, F_2, ..., F_m, G$ перебувають у відношенні R, то формула G називається **безпосередньо вивідною** з формул $F_1, F_2, ..., F_m$ за правилом R.

Часто правило виводу $R(F_1, F_2, ..., F_m, G)$ записують у вигляді $\dfrac{F_1, F_2, ..., F_m}{G}$.

Формули $F_1, F_2, ..., F_m$ називають **припущеннями, посилками** або **гіпотезами** правила R, а формулу G – **висновком** або **наслідком**.

Виведенням (виводом) формули *B* з формул $A_1, A_2, ..., A_n$ називають послідовність формул $F_1, F_2, ..., F_m$ таку, що F_m =*B*, а будь-яка формула F_i, i = 1, 2, ..., m є:

1) або аксіомою;

2) або однією з початкових формул $A_1, A_2, ..., A_n$;

3) або безпосередньо вивідною з формул $F_1, F_2, ..., F_{i-1}$ (або будь-якої їх підмножини) за одним із правил виведення.

Якщо існує виведення формули *B* з формул $A_1, A_2, ..., A_n$, то кажуть, що *B* є **вивідною** з $A_1, A_2, ..., A_n$ і позначають цей факт так: $A_1, A_2, ..., A_n \vdash B$. Формули $A_1, A_2, ..., A_n$ називають **посилками**, або **гіпотезами**, виведення. Перехід у виведенні від формули F_{i-1} до F_i називають *i*-м кроком виведення.

Доведенням формули *B* у теорії *T* називають виведення *B* з порожньої множини формул, тобто виведення, у якому як початкові формули використовують тільки аксіоми теорії.

Формула *B*, для якщо існує доведення, називається формулою **довідною(вивідною)** у теорії *T*, або **теоремою** теорії *T*; факт довідності формули *B* позначають $\vdash B$.

$\vdash$ – знак виведення.

Довести, що формули є тавтологіями, можна за допомогою відповідних таблиць істинності. Той факт, що формула *A* алгебри висловлювань є тавтологією позначають так$\vDash A$. Символ $\vDash$, як і *A*, належить метамові.

У вивченні формальних теорій існує два типи тверджень:

1) твердження самої теорії або її теореми;

2) твердження про теорію (про властивості її теорем, властивості доведень тощо).

Перші є елементами (словами, виразами, формулами) внутрішньої мови теорії, а другі – зовнішніми й формулюються у термінах мови, зовнішньої стосовно теорії і званої **метамовою** теорії; самі ці твердження називають **метатеоремами**.

Для розпізнавання мови логіки висловлювань і метамови дослідника між посилками й наслідком вводимо такі заміни:

Об'єктні символи мови логіки висловлювань	Суб'єктні символи метамови
Імплікація $\rightarrow$	Метаімплікація $\Rightarrow$
Кон'юнкція $\wedge$	Метакон'юнкція **,**
Диз'юнкція $\vee$	Метадиз'юнкції **;**

Тоді твердження, яке потрібно довести, у логіці висловлювань можна оформити у вигляді такого причинно-наслідкового відношення:

$$P_1, P_2, \ldots, P_{n+1}, P_n \Rightarrow \mathrm{C}(1)$$

де P_i – посилка (причина); С – висновок (наслідок).

Читається: «*Якщо посилки* $P_1, P_2, \ldots, P_{n+1}, P_n$ *істинні, то і висновок С також істинний*» або «*Якщо причини* $P_1, P_2, \ldots, P_{n+1}, P_n$ *мали місце, то матиме місце і наслідок*».

Щоб не переплутати об'єктивне висловлювання із суб'єктивним висловлюванням, справедливість якого потрібно встановити, домовимося речення (1) називати клаузою (clause).

Клаузою називають **метапропозицію**, у якій виконується відношення порядку, оформлене через символ метаімплікації «⇒».

Твердження, справедливість яких очевидна:

1. Заперечення тавтології є суперечністю і навпаки.
2. Кожна тавтологія є виконуваною формулою (навпаки, взагалі кажучи, ні).
3. Кожна нейтральна формула є виконуваною, але не навпаки.
4. Заперечення виконуваної формули може бути як виконуваною формулою, так і невиконуваною формулою.

Дві формули *A* і *B* алгебри висловлювань називаються **рівносильними**, якщо їм відповідає та сама функція істинності. Рівносильність формул *A* і *B* позначають за допомогою знака = ($\equiv$або$\leftrightarrow$): записують *A* = *B* ($A \equiv B$ або $A \leftrightarrow B$).

Очевидно, що відношення рівносильності на множині формул є **відношенням еквівалентності**, тому часто це відношення називають **еквівалентністю**.

Теорема 1. Формули алгебри висловлювань A і B рівносильні тоді й тільки тоді, коли формула $((A \to B) \wedge (B \to A))$ є тавтологією.

З метою скорочення запису формул, подібних до формули з наведеної теореми, до сигнатури алгебри висловлень вводять додаткову операцію, що позначається ~ і позначається так: $(A \sim B)$ є скороченим записом формули $((A \to B) \wedge (B \to A))$. Отже, теорему можна сформулювати так.

Формули A і B рівносильні тоді й тільки тоді, коли формула $(A \sim B)$ є тотожно істинною.

Нехай $A(p_1, p_2, ..., p_n)$ і $B(p_1, p_2, ..., p_n)$ – дві формули алгебри висловлювань. Будемо говорити, що формула $B(p_1, p_2, ..., p_n)$ є ***логічним слідуванням*** формули $A(p_1, p_2, ..., p_n)$, якщо B приймає значення 1 для всіх тих наборів значень пропозиційних змінних, для яких формула A істинна (тобто приймає значення 1); позначатимемо це $A \Rightarrow B$.

Це означає ,що множина наборів значень змінних, для яких істинна формула A, є підмножиною множини наборів значень змінних, для яких істинна формула B, що є логічним слідуванням формули A.

Теорема 2. Формула $B(p_1, p_2, ..., p_n)$ є логічним слідуванням формули

$A(p_1, p_2, ..., p_n)$ тоді й тільки тоді, коли тотожно істинною є формула $(A \to B)$.

Приклад 4.11. Формула $B(x, y, z) = (x \vee z)$ є логічним слідуванням формули $A(x, y, z) = ((x \vee y) \wedge z)$, що випливає з таблиці істинності:

$x\ y\ z$	$A\ \ B$
0 0 0	0 0
0 0 1	0 1
0 1 0	0 0
0 1 1	1 1
1 0 0	0 1
1 0 1	1 1
1 1 0	0 1
1 1 1	1 1

Очевидно, що дві формули A і B є рівносильними тоді й тільки тоді, коли кожна з них є логічним слідуванням іншої, тобто $A = B$ тоді й тільки тоді, коли $A \Rightarrow B$ і $B \Rightarrow A$.

Аксіоми числення висловлювань

A1: $a \to (b \to a)$.

A2: $(a \to b) \to ((a \to (b \to c)) \to (a \to c\ ))$.

A3: $(a \wedge b) \to a$.

A4: $(a \wedge b) \to b$.

A5: $(a \to b) \to ((a \to c) \to (a \to (b \wedge c)))$.

A6: $a \to (a \vee b)$.

A7: $b \to (a \vee b)$.

A8: $(a \to c) \to ((b \to c) \to ((a \vee b) \to c))$.

A9: $(a \to \neg b) \to (b \to \neg a)$.

A10: $\neg\neg a \to a$.

Теорема дедукції

Теорема дедукції дає нове правило виведення у логіці висловлювань, і тому вона є важливою для доведення теорем і різних висловлювань.

Теорема дедукції 1. Якщо *Г* – множина формул, *A* і *B* – формули і *Г*, A $\vdash B$, то Г $\vdash A \to B$. Зокрема, якщо A $\vdash B$, то $\vdash A \to B$.

Теорема 2. $A \to B$, $B \to C \vdash A \to C$.

Теорема 3. $A \to (B \to C), B \vdash A \to C$.

Теорема 4. Для будь-яких формул *A* і *B* такі формули є теоремами формальної теорії *L*:

а) $\neg\neg B \to B$;

б) $B \to \neg\neg B$;

в) $\neg A \to (A \to B)$;

г) $(\neg B \to \neg A) \to (A \to B)$;

д) $(A \to B) \to (\neg B \to \neg A)$;

є) $A \to (\neg B \to (A \to B))$;

ж) $(A \to B) \to ((\neg A \to B) \to B)$.

а) $\vdash \neg\neg B \to B$

Правила виведення

1. **Правило підстановки**. Якщо *A* – вивідна формула, яка містить літеру *p* (позначимо цей факт *A*(*p*)), то вивідною є і формула *A*(*B*), що здобувається з *A* заміною всіх входжень літери *p* на довільну формулу *B*; записується

$$\frac{A(p)}{A(B)} \text{ або } S_B^p A = A(B)$$

S – формула, отримана з *A* шляхом підстановки замість *p* змінної *B*.

2. **Правило висновку (modus ponens)**. Якщо A і $A = \rightarrow B$ – вивідні формули, то вивідною є й формула *B*; записується

$$\frac{A, A \rightarrow B}{B} \text{ або } MP(A, A \rightarrow B) = B\,.$$

Якщо відомо, що висловлювання А породжує висловлювання В та А істина, то і В істина.

Доведення логіки висловлювань

Доведення за допомогою таблиці істинності

Цей простий метод може бути застосований для перевірки рівносильності або нерівносильності будь-яких формул *A* і *B* довільної складності. Отже, на перший погляд може здатися, що проблема встановлення рівносильності або нерівносильності формул алгебри висловлень є розв'язаною і до того ж найпростішим чином, тож усі подальші дослідження в цьому напрямі є непотрібними.

Наведемо лише два міркування, які демонструють, що перше враження є оманливим.

Перше міркування пов'язане з тим, що коли кількість пропозиційних змінних у досліджуваних формулах є значною, то застосування зазначеного простого методу може стати практично нездійсненним. Адже вже для 30 змінних потрібно випробувати понад 10^9 наборів значень змінних для кожної формули. Це тільки кількість кроків загальної процедури, крім того, слід врахувати трудомісткість обчислення значень функцій істинності даних формул на кожному з наборів.

По-друге, в алгебрі висловлювань у більшості випадків цікавляться не рівносильністю двох будь-яких заданих формул, а рівносильністю нескінченної множини пар формул. Потрібні твердження, згідно з якими всі формули певного типу є рівносильними відповідно до формул певного іншого типу. Якщо множини формул обох цих типів є нескінченними, то подібні твердження, очевидно, не можуть бути встановлені жодним методом, що спирається на побудові таблиць істинності, а потребують загальних міркувань.

Приклад 4.12. Довести за допомогою таблиці істинності істинність виведення:

$$A \rightarrow B, \neg A \rightarrow B \vdash B.$$

A	B	¬A	A→B	¬A→B	A→B, ¬A→B
х	х	і	і	х	х
х	і	і	і	і	і
і	х	х	х	і	х
і	і	х	і	і	і

Приклад 4.13. Довести істинність логічного висловлювання за допомогою таблиці:

$$A \to B, \neg C \to \neg B, \neg C \Rightarrow \neg A.$$

A	B	C	¬A	¬B	¬C	A→B	¬C→¬B	P
0	0	0	1	1	1	1	1	1
0	0	1	1	1	0	1	1	0
0	1	0	1	0	1	1	0	0
0	1	1	1	0	0	1	1	0
1	0	0	0	1	1	0	1	0
1	0	1	0	1	0	0	1	0
1	1	0	0	0	1	1	0	0
1	1	1	0	0	0	1	1	0

Клауза вважається істиною, якщо одиниці наслідку ($\neg A$) накривають усі одиниці узагальненої причини P.

Аксіоматичний метод доведення

Аксіоматичний метод перевірки тотожної істинності логічних висловлювань ґрунтується на тому, щоб серед нескінченного числа істинних клауз знайти незалежну систему аксіом, за допомогою якої можна було встановити справедливість будь-яких клауз.

Оскільки доведення в логіці висловлювань будуються на відношенні порядку, то логіка висловлювань є **розширенням логіки Буля**. Тому всі істинні тотожності логіки Буля автоматично стають справедливими кляузами логіки висловлювань.

Наприклад, **закон склеювання** логіки Буля

$$(A \vee B) \wedge (A \vee \neg B) = A$$

можна подати такими клаузами:

$$(A \vee B), (A \vee \neg B) \Rightarrow A$$

$$A \Rightarrow (A \vee B) \wedge (A \vee \neg B),$$
$$1 \Rightarrow ((A \vee B) \wedge (A \vee \neg B)) \sim A,$$
$$A \vee B \Rightarrow (A \vee \neg B) \to A$$

Таким чином, незалежна система аксіом логіки Буля, яка складається з чотирьох законів – комутативності, асоціативності, дистрибутивності, нуля і одиниці, – автоматично стає системою аксіом і логіки висловлювань.

Приклад 4.14. Довести істинність логічного висловлювання аксіоматичним методом:

$$A \to B, \neg C \to \neg B, \neg C \Rightarrow \neg A.$$
$$F1: MP\ (\neg C \to \neg B,\ \neg C) \Rightarrow \neg B.$$
F2: Закон контрапозиції $A \to B \Leftrightarrow \neg B \to \neg A$.
$$F3: MP\ (\neg B \to \neg A,\ \neg B) \Rightarrow \neg A.$$

Приклад 4.15. Доведемо, що формула *a*→*a* є теоремою алгебри висловлювань алгебри висловлювань.

Підставляючи в аксіому *A*2 змінну *a* замість змінної *c* і *b*→*a* замість *b*, матимемо вивідну формулу.

$$\text{F1: } S^{b,c}_{b\to a,a}A2 = (a \to (b \to a)) \to ((a \to ((b \to a) \to a)) \to (a \to a)).$$

Підформула *a*→(*b*→*a*) є аксіомою *A*1. Тоді з вивідних формул за правилом висновку виводимо формулу

$$\text{F2: } MP(A1, F1) = ((a \to ((b \to a) \to a)) \to (a \to a)).$$

Замінимо в аксіомі *A*1 *b* на (*b*→*a*):

$$\text{F3: } S^{b}_{b\to a}A1 = (a \to ((b \to a) \to a)).$$

Знову, застосовуючи правило висновку до двох останніх формул, матимемо, що формула *a*→*a* є вивідною.

$$\text{F4: } MP(F3, F2) = (a \to a).$$

Метод резолюцій

У цьому методі для доведення істинності висловлювань використовують аксіоми порядку. Суть методу зводиться до того, що дві посилки диз'юнкти з протилежними термами завжди можливо склеїти в один остаточний диз'юнкт, у якому вже не буде протилежних термів, наприклад $A \vee B,\ C \vee \neg B \Rightarrow A \vee C$, де *A* і *C* – задовільні терми або цілі диз'юнкти з будь-яким набором термів, включно з хибністю *B* і $\neg B$ – задовільні терми. Якщо послідовно застосувати метод резолюцій до досліджуваної клаузи, то внаслідок цього отримаємо зменшення числа букв, у тому числі до їх повного зникнення.

Приклад 4.16. Довести істинність клаузи методом резолюцій:

$$D \to F,\ A \to (E \to D),\ C \to (B \to A)\ \Rightarrow E \to (\neg C \to F) \to \neg B.$$

Запишемо клаузу в КНФ:

$$\neg D \vee F,\ \neg A \neg E \vee D,\ \neg C \vee \neg B \vee A \Rightarrow \neg E \vee \neg C \vee F \vee \neg B.$$

Перенесемо всі посилки в ліву частину клаузи

$$\neg D \vee F,\ \neg A \vee \neg E \vee D,\ \neg C \vee \neg B \vee A,\ E,\ C,\ \neg F,\ B \Rightarrow 0.$$

Випишемо посилки клаузи у стовпчик і склеїмо їх по черзі:

1. $\neg D \vee F$	8. $\neg A \vee \neg E \vee F$ (1,2)
2. $\neg A \vee \neg E \vee D$	9. $\neg C \vee A$ (3,7)
3. $\neg C \vee \neg B \vee A$	10. $\neg C \vee \neg E \vee F$ (8,9)
4. E	11. $\neg C \vee F$(4,10)
5. C	12. F(5,11)
6. $\neg F$	13. 0(6,12)
7. B	

Із черги склеювання посилок випливає, що ця клауза істинна.

Приклад 4.17. Довести істинність клаузи методом резолюцій:

$$A \to B,\ C \to D,\ A \vee C,\ \neg A \vee \neg D,\ C \to \neg B \Rightarrow (A \vee B) \to (A \wedge B).$$

Запишемо клаузу в КНФ:

$$\neg A \vee B,\ \neg C \vee D,\ A \vee C,\ \neg A \vee \neg D,\ \neg C \vee \neg B \Rightarrow \neg (A \vee B) \vee (A \wedge B).$$

Перенесемо всі посилки в ліву частину клаузи

$$\neg A \vee B,\ \neg C \vee D,\ A \vee C,\ \neg A \vee \neg D,\ \neg C \vee \neg B,\ A \vee B,\ \neg A \vee \neg B \Rightarrow 0.$$

Випишемо посилки клаузи у стовпчикі склеїмо їх по черзі:

1. $\neg A \vee B$	8. $A \vee D$ (2,3)
2. $\neg C \vee D$	9. $B \vee D$ (1,8)
3. $A \vee C$	10. $B \vee \neg D$ (4,6)
4. $\neg A \vee \neg D$	11. B (9,10)
5. $\neg C \vee \neg B$	12. $\neg C$ (5,11)
6. $A \vee B$	13. A (3,12)
7. $\neg A \vee \neg B$	14. $\neg B$ (7,13)
	15. 0 (11,14)

Із черги склеювання посилок випливає, що ця клауза істинна.

Основні закони алгебри висловлень

Тотожно-істинні формули та формули рівносильності називаються законами алгебри висловлень (властивостями, правилами, теоремами). Існує нескінчена множина тавтологій та рівносильностей, а отже, і законів алгебри висловлень. Нижче наведено закони (з використанням пропозиційних змінних), які найчастіше зустрічаються на практиці:

$x \wedge 0 = 0$, $x \wedge 1 = x$, $x \vee 0 = x$, $x \vee 1 = 1$ – закони сталих (констант);

$x = x$ – закон тотожності;

$\bar{\bar{x}} = x$ – закон подвійного заперечення;

$x \wedge \bar{x} = 0$ – закон протиріччя;

$x \vee \bar{x} = 1$ – закон вилученого третього;

$x \vee y = y \vee x$, $x \wedge y = y \wedge x$ – комутативність $\vee$ та $\wedge$;

$x \vee (y \vee z) = (x \vee y) \vee z$, $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ – асоціативність $\vee$ та $\wedge$;

$x \wedge (y \vee z) = x \wedge y \vee x \wedge z$ – перший дистрибутивний закон;

$x \vee y \wedge z = (x \vee y) \wedge (x \vee z)$ – другий дистрибутивний закон;

$x \wedge (x \vee y) = x$ – перший закон поглинання;

$x \vee x \wedge y = x$ – другий закон поглинання;

$x \vee x = x$, $x \wedge x = x$ – ідемпотентність;

$x \wedge y \vee x \wedge \bar{y} = x$ – перший закон склеювання;

$(x \vee y) \wedge (x \vee \bar{y}) = x$ – другий закон склеювання;

$\overline{x \vee y} = \bar{x} \wedge \bar{y}$; $\overline{x \wedge y} = \bar{x} \vee \bar{y}$ – правила де Моргана;

$\models (x \to y) \wedge x \to y$ – правило твердження, modus ponens;

$\models (x \to y) \wedge \bar{y} \to \bar{x}$ – правило спростування, modus tollens.

Логічний наслідок

Формула алгебри висловлень B є логічним наслідком з формули A (позначається $A \models B$), якщо B є істинне на всіх наборах значень змінних, для яких A є істинна. Наприклад, формула $B = x \vee x$ є логічним наслідком формули $A = x \wedge (y \vee \bar{y})$, тобто $x \wedge (y \vee \bar{y}) \models x \vee x$.

Теорема 5. Формула алгебри висловлень B є логічним наслідком з формули A тоді й лише тоді, коли формула $A \to B$ є загальнозначущою, тобто $A \models B \sim \models A \to B$.

Доведення. Відповідно до визначення імплікації, вираз $A \to B$ є хибний лише за істинного A та хибного B а отже, якщо $A \to B$ – тавтологія, то з істинності A завжди випливає істинність B, тобто $A \models B$.

І, навпаки, якщо $A \models B$, то виключається випадок, коли A є істинне та B − хибне, а отже, $A \to B$ є істинне на всіх наборах значень змінних, тобто $\models A \to B$.

Логічний наслідок можна узагальнити на сукупності формул: формула алгебри висловлень B є логічним наслідком формул $A_1, A_2, \dots, A_n$ та позначається як $A_1, A_2, \dots, A_n \models B$, якщо для довільного набору значень з істинності всіх A_i , $i = 1, n$, на цьому наборі випливає істинність B. Наприклад, розглядаючи таблицю істинності, здобудемо три ілюстрації до наведеного визначення:

$$x, z, x \wedge y \to \overline{z} \models \overline{y} \text{ − 6-й рядок;}$$

$$x, x \to z, z \models x \vee y \to z \text{ – 6-й і 8-й рядки;}$$

$$x \wedge y \to \overline{z}, x \to z \models \overline{x \wedge y} \text{ − 1-й, 2-й, 6-й рядки.}$$

№ пп	x	y	z	$x \wedge y \to \overline{z}$	$\overline{y}$	$x \to z$	$x \vee y \to z$	$\overline{x \wedge y}$
1	0	0	0	1	1	1	1	1
2	0	0	1	1	1	1	1	1
3	0	1	0	1	0	1	0	1
4	0	1	1	1	0	1	1	1
5	1	0	0	1	1	0	0	1
6	1	0	1	1	1	1	1	1
7	1	1	0	1	0	0	0	0
8	1	1	1	0	0	1	1	0

Теорема 6. Формула алгебри висловлень B є логічним наслідком формул $A_1, A_2, \dots, A_n$ тоді й лише тоді, коли формула $A_1 \wedge A_2 \wedge \dots, \wedge A_n \to B$ є загальнозначущою, тобто $A_1, A_2, \dots, A_n \models B \sim \models A_1 \wedge A_2 \wedge \dots \wedge A_n \to B$.

Доведення. Доведення виконується аналогічно до доведення теореми 4.3.

Формули алгебри висловлень можна застосовувати для перевірки правильності логічних суджень, незважаючи на конкретний зміст висловлень. Що ж стосується здорового глузду, то він має виявлятися під час застосування законів логіки висловлень у її конкретних додатках.

Наприклад, висловлення «A = 100 < 10» − хибне. Однак воно стає істинним, якщо вважати, що число 100 записане у двійковій системі числення, а 10 – у десятковій.

Приклад 4.18. Перевірити правильність наступного судження. Якщо замінити мікросхему (A), телевізор працюватиме (B) за умови, що напругу ввімкнено (C). Мікросхему замінили, а напругу не ввімкнули. Отже, телевізор не працюватиме.

Розв'язок. Це судження можна записати у вигляді

$$A \to B \wedge \overline{C},\ A,\ \overline{C} \models \overline{B}.$$

Оскільки формули A, B, C не містять підформул, то можна перейти до відповідних пропозиційних змінних x, y, z.

Тоді цей логічний висновок набуде більш зручного вигляду:

$$x \to y \wedge z,\ x,\ \overline{z} \models \overline{y}\ .$$

Судження буде слушним, якщо формула

$$(x \to y \wedge z) \wedge x \wedge \overline{z} \to \overline{y}$$

є загальнозначуща. Для викладок скористаємося основними законами алгебри висловлень:

$$(x \to y \wedge z) \wedge x \wedge \overline{z} \to \overline{y} = \overline{(\overline{x} \vee y \wedge z) \wedge x \wedge \overline{z}} \vee \overline{y} = \overline{0} \vee \overline{y} = 1 \vee \overline{y} = 1.$$

Формула є загальнозначущою, отже, судження є правильне.

Відповідь: судження є правильне.

Приклад 4.19. Я піду на лекцію (x) або залишуся в барі й вип'ю кави (y). Я не піду на лекцію. Отже, я залишуся й вип'ю кави.

Запишемо логічне слідування:

$$x \vee y,\ \overline{x} \models y.$$

Перевіримо загальнозначимість:

$$(x \vee y) \wedge \overline{x} \to y = \overline{(x \vee y) \wedge \overline{x}} \vee y = \overline{x}\ \overline{y} \vee x \vee y = 1 \vee x = 1.$$

Судження є правильне.

4.2 Функції алгебри логіки. Булева алгебра

Способи задання булевих функцій

Для відображення інформації в комп'ютерах використовується двійкова система числення, тобто всі операції, які виконує комп'ютер, проводяться на множині $\{0;1\}$.

Джорджем Булем у середині XIX ст. було створено апарат двійкової логіки, алгебри, яку називають **булевою**.

Ця алгебра використовується у проєктуванні інтелектуальних систем, роботі з базами даних тощо.

Джордж Буль (Boole) – англійський математик і логік, один із засновників математичної логіки. Розробив алгебру логіки (булеву алгебру), основу функціонування цифрових комп'ютерів («Дослідження законів мислення», 1854 р.).

Булева алгебра – це алгебраїчна структура ($A,\wedge,\vee,\ \bar{}\ ,0,1$) з бінарними операціями $\wedge,\vee$: $A^2 \to A$, унарною операцією «$\bar{}$» : $A \to A$ і виділеними елементами 0,1 в носії *A,* які задовольняють властивості комутативності, асоціативності і дистрибутивності.

Якщо носій алгебраїчної структури $B = \{0,1\}$ складається з двох елементів, то така структура $\left(B,\wedge,\vee,\ \bar{}\ \right)$ називається **двохелементною булевою алгеброю.**

Алгеброю логіки називається двохелементна булева алгебра ($B,\wedge,\vee,\ \bar{}\ ,\ \to,\ \sim$). $B = \{0,1\}$, у якій множину операцій доповнено двома бінарними операціями: імплікацією і еквівалентністю.

Відношення поміж булевими змінними подаються булевими функціями, які, подібно до числових функцій, можуть залежати від однієї, двох чи більшої кількості змінних (аргументів).

Визначення. Булевою функцією *n* незалежних змінних називається функція $y = f(x_1, x_2, \ldots, x_n)$, $n \geq 1$, у якій кожна змінна і сама функція набувають власних значень з множини {0; l}, тобто $x_k \in \{0; 1\}$, $k = 1, \ldots, n$, $y = \{0, 1\}$.

Визначення. Кортеж ($x_1, x_2, \ldots, x_n$) конкретних значень булевих змінних називається **набором**, або **булевим вектором.**

Якщо незалежні змінні розміщено в прямому порядку, тобто у вигляді $x = (x_1, x_2, \ldots, x_n)$, то набір називається **прямим**, а якщо їх розміщено у зворотному порядку, тобто у вигляді $x = (x_n, x_{n-1}, \ldots, x_1)$, то набір називається **зворотним**.

Областю визначення булевої функції n **аргументів** є сукупність 2^n булевих кортежів. Число різних булевих функцій є скінченне і дорівнює 2^{2^n}. За $n = 1$ число булевих функцій дорівнює 4, а за $n = 2$ – 16.

Існують такі способи задання булевих функцій.

1. Табличний. Функція задається у вигляді таблиці істинності. Наприклад, така таблиця визначає функцію y.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

2. Графічний. Функція задається у вигляді n-вимірного одиничного куба, у вершинах якого записано значення функції (у кружечках) та набори значень аргументів. Наприклад, функції, задані на рис. 4.3 та 4.4.

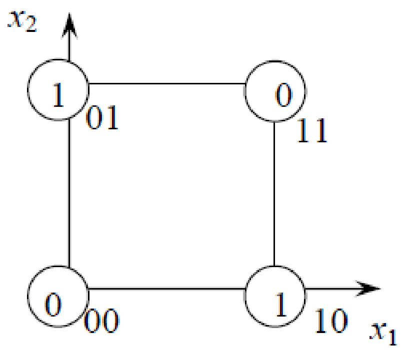


Рис. 4.3.
Двовимірний одиничний квадрат

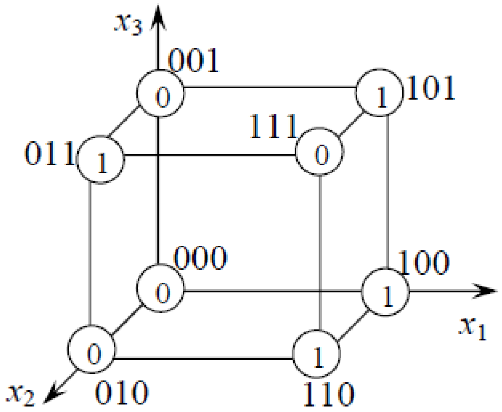
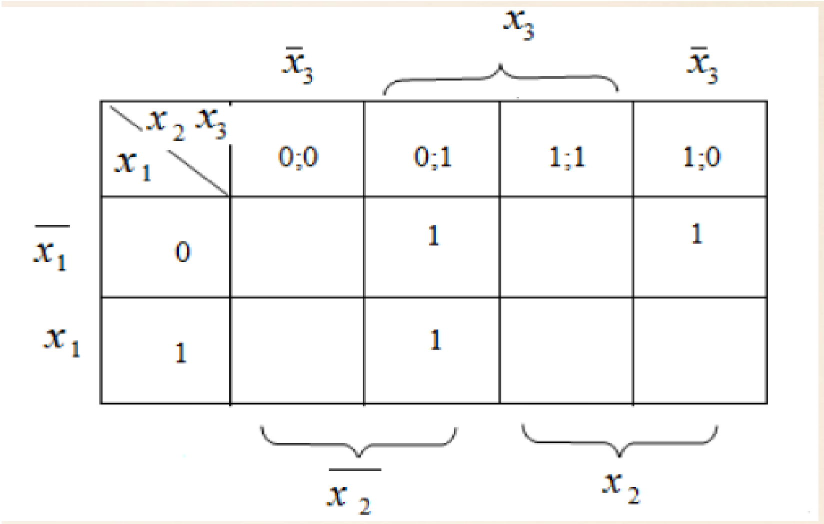


Рис. 4.4.
Тривимірний одиничний куб

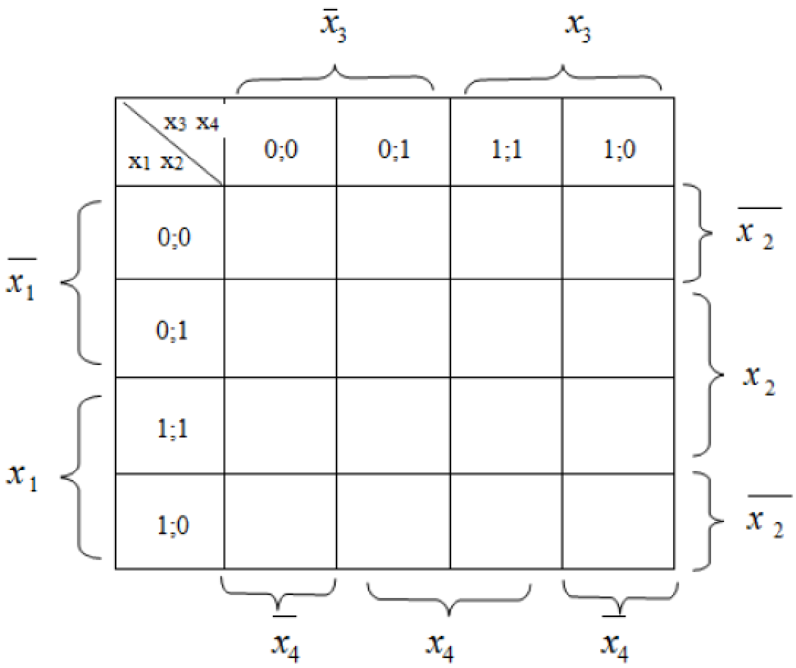
3. Координатний(картою Карно). У клітинках карти записуються значення функції (нулі зазвичай не вписують, їм відповідають порожні клітини).



Карта Карно для трьох змінних

Функція $f(x_1, x_2, x_3)$ = 1 на наборах (0, 0, 1), (1, 0, 1), (0, 1, 0) або ($(\bar{x}_1, \bar{x}_2, x_3)$, $(x_1, \bar{x}_2, x_3)$, $(\bar{x}_1, x_2, \bar{x}_3)$).

Карта Карно для чотирьох змінних



4. Числовий. Функція задається у вигляді цілих десяткових (вісімкових, шістнадцяткових) чисел, які є еквівалентами тих наборів значень аргументів, на яких функція набуває значення 1. Наприклад, f= {2; 4; 5; 7} для трьох змінних x_1, x_2, x_3.

Десяткова система числення	x_1	x_2	x_3	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

5. Аналітичний. Функція задається у вигляді формули. Наприклад:

$$f = x_1 + x_2 \cdot x_3$$

$$f(x_1, x_2, x_3, x_4) = \left(\left(\left(\overline{x_1 \mid x_2 \mid \overline{x_3} \mid x_4}\right) \to \left(\overline{\overline{x_1 \vee \overline{x_2} \vee x_3}}\right)\right)\right) \to$$

Елементарні функції алгебри логіки

Булеві функції однієї та двох незалежних змінних прийнято називати **елементарними** булевими функціями. Вони використовуються як логічні операції над булевими змінними в побудові булевих функцій багатьох незалежних змінних. Алгебра з такими логічними операціями називається **алгеброю логіки**, а булеві функції називаються ще **функціями алгебри логіки**.

Загальне число різних елементарних функцій (логічних операцій) дорівнює загальному числу функцій двох змінних, тобто 2^{2^2} = 16 (функції однієї змінної є окремим випадком функцій двох змінних). Основними в алгебрі логіки є три логічні операції.

Заперечення (інверсія) − функція $y = f(x)$, яка набуває значення 1, коли $x = 0$, і значення 0 − за $x = 1$. Позначення: $y = \overline{x}$. Читається: «не *x*».

Диз’юнкція (логічне додавання) – функція $y = f(x_1, x_2)$ яка набуває значення 0 тоді й лише тоді, коли обидва аргументи дорівнюють нулю.

Позначення: $y = x_1 + x_2$ або $y = x_1 \vee x_2$. Читається: «x_1 плюс x_2» або «x_1 або x_2».

Кон'юнкція (логічне множення) – функція $y = f(x_1, x_2)$, яка набуває значення 1 тоді й лише тоді, коли обидва аргументи дорівнюють одиниці. Позначається: $y = x_1 \cdot x_2$ або $y = x_1 \wedge x_2$, Читається: «x_1 помножити на x_2» або «x_1 та x_2».

Таблиці істинності наведених логічних операцій мають відповідно вигляд:

x	$y = \overline{x}$
0	1
1	0

x_1	x_2	$y = x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	$y = x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Якщо операція містить один операнд, вона називається **одномісною**, або **унарною**, а якщо два, то – **двомісною**, або **бінарною**. Заперечення – це одномісна операція, а диз'юнкція та кон'юнкція − двомісні. При цьому вирази $\overline{x}$, $x_1 + x_2$, $x_1 \cdot x_2$ є прикладами логічних формул. Більш складні формули дістаємо за рахунок суперпозиції логічних формул, які зазвичай беруться в круглі дужки.

Наприклад: $y = x_1 + x_2(\overline{x_1} + x_2)(x_1 + x_1\overline{x_2})$.

Основні логічні операції над висловлюваннями. Унарні

Константа нуля

Логічні операції можна задавати за допомогою таблиць істинності, що показують відповідність значень істинності висловлювань.

x	$F(x)$
0	0
1	0

Константа одиниці

x	$F(x)$
0	1
1	1

Повторення x

x	$F(x)$
0	0
1	1

Логічним запереченням висловлювання x називається висловлювання, яке є істинним тоді й тільки тоді, коли висловлювання x є хибним. Заперечення позначається $\overline{x}$ або $\neg x$ (читається: «не x»).

x	$\overline{x}$
1	0
0	1

Бінарні

Константа нуля

x	y	$F(x, y)$
1	1	0
1	0	0
0	1	0
0	0	0

Кон'юнкцією двох висловлювань x і y називається висловлювання **і**, яке є істинним тоді й тільки тоді, коли істинними є обидва висловлювання x і y. Кон'юнкція позначається: $x \wedge y$, або x & y (читається: «x **і** y»). Таблиця істинності для $x \wedge y$ має вигляд:

x	y	$x \wedge y$
1	1	1
1	0	0
0	1	0
0	0	0

Заборона за y*:* $f(x, y) = x \Delta y$ або $x \wedge \overline{y}$

x	y	$f(x, y)$
1	1	0
1	0	1
0	1	0
0	0	0

Змінна x*:* $f(x, y) = x$

x	y	$f(x, y)$
1	1	1
1	0	1
0	1	0
0	0	0

Заборона за x*:* $f(x, y) = y \Delta x$ або $y \wedge \overline{x}$

x	y	$f(x, y)$
1	1	0
1	0	0
0	1	1
0	0	0

Змінна y*:* $f(x,y) = y$

x	y	$f(x, y)$
1	1	1
1	0	0
0	1	1
0	0	0

Диз'юнкцією двох висловлювань x і y називається висловлювання, хибне тоді й тільки тоді, коли обидва висловлювання x і y хибні. Диз'юнкція позначається $x \vee y$ або $x + y$ (читається: «x **або** y»). Таблиця істинності для $x \vee y$ має вигляд:

x	y	$x \vee y$
1	1	1
1	0	1
0	1	1
0	0	0

Сума за модулем 2 (альтернативною диз'юнкцією, логічною сумою, виключаючим «АБО», строгою диз'юнкцією) двох висловлювань x і y називається висловлювання, що є істинним тоді й тільки тоді, коли обидва висловлювання x і y *приймають різні значення*. Диз'юнкція позначається $x \oplus y$ (читається: «або x, або y»). Таблиця істинності для $x \oplus y$ має вигляд заперечення еквівалентності, виключаючи «АБО»

$$f(x, y) = x \oplus y \text{ або } (\bar{x} \wedge y) \vee (x \wedge \bar{y})$$

x	y	$f(x, y)$
1	1	0
1	0	1
0	1	1
0	0	0

Стрілка Пірса – це заперечення диз'юнкції. Стрілка Пірса позначається X ↓ Y. Читається «ані X, ані Y».

Введена до розгляду Чарльзом Пірсом (CharlesPeirce) в 1880–1881 рр. Таблиця істинності для стрілки Пірса має вигляд: $f(x, y) = x \downarrow y$ або $\overline{x \vee y}$ або $\bar{x} \wedge \bar{y}$

x	y	$f(x, y)$
1	1	0
1	0	0
0	1	0
0	0	1

Заперечення х*:* $f(x, y) = \overline{y}$

x	y	$f(x, y)$
1	1	0
1	0	1
0	1	0
0	0	1

Імплікацією двох висловлювань x і y називається висловлювання, хибне тоді й тільки тоді, коли висловлювання x є істиною, а y – хибне. Імплікація позначається: $x \to y$ (читається: «x тягне за собою y» або «з x слідує y»). Права імплікація: $f(x, y) = x \vee \overline{y}$ або $y \to x$:

x	y	$f(x, y)$
1	1	1
1	0	1
0	1	0
0	0	1

Ліва імплікація: $f(x, y) = \overline{x} \vee y$ або $x \to y$

x	y	$f(x, y)$
1	1	1
1	0	0
0	1	1
0	0	1

Еквіваленцією (еквівалентністю) двох висловлювань x і y називається висловлювання, що є істинним тоді й тільки тоді, коли істинності висловлювань збігаються. Еквіваленція позначається: $x \leftrightarrow y$, або $x \sim y$,або $(\overline{x} \wedge \overline{y}) \vee (x \wedge y)$, або $\overline{x \oplus y}$ (читається: «x еквівалентний y» або «x тоді й тільки тоді, коли y»). Таблиця істинності для $x \leftrightarrow y$ має вигляд:

x	y	$x \leftrightarrow y$
1	1	1
1	0	0
0	1	0
0	0	1

Заперечення у: $f(x, y) = \bar{x}$

x	*y*	*f (x, y)*
1	1	0
1	0	0
0	1	1
0	0	1

Штрих Шеффера – це заперечення кон’юнкції.

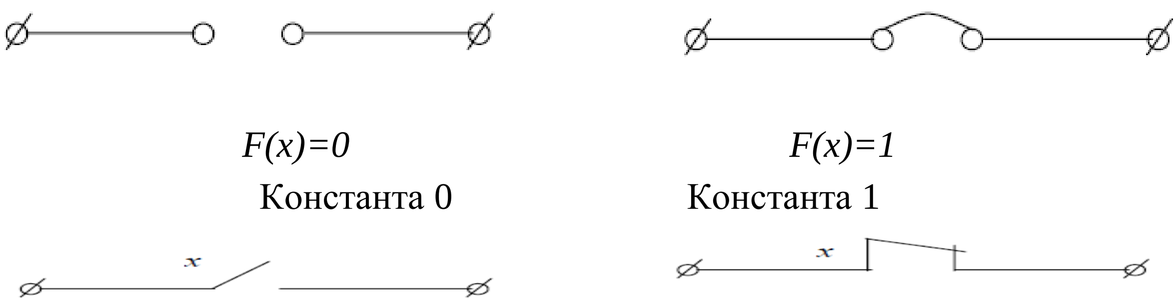
Введений до розгляду Генрі Шеффером в 1913 р. Штрих Шеффера позначається $x|y$ або $\overline{x \wedge y}$ або $\bar{x} \vee \bar{y}$, задається такою таблицею істинності:

x	*y*	x|y
1	1	0
1	0	1
0	1	1
0	0	1
0	0	1

Константа одиниці

x	*y*	*f (x, y)*
1	1	1
1	0	1
0	1	1
0	0	1

Технічну реалізацію функцій однієї змінної наведено на рис. 4.6.



F(x)=0 Константа 0

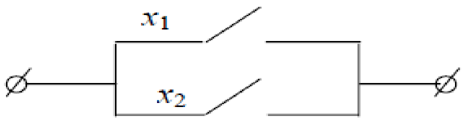
F(x)=1 Константа 1

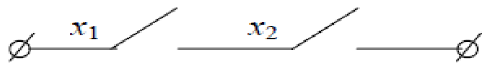
F(x)=x Повторення

F(x)= $\bar{x}$ Заперечення

Рис. 4.6. Технічна реалізація функцій однієї змінної

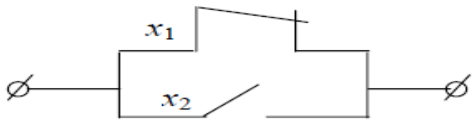
Технічну реалізацію деяких функцій двох змінних показано на рис. 4.7.

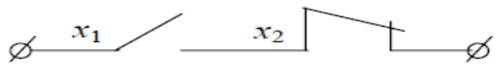




$F(x_1, x_2) = x_1 + x_2$ Диз'юнкція

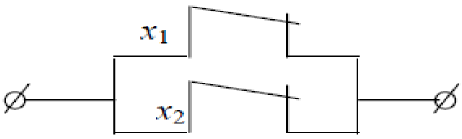
$F(x_1, x_2) = x_1 \cdot x_2$ Кон'юнкція

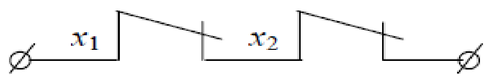




$F(x_1, x_2) = x_1 \to x_2$ Ліва імплікація

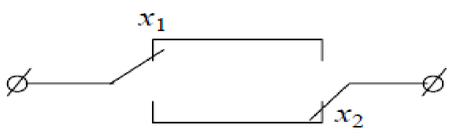
$F(x_1, x_2) = x_1 \leftarrow x_2$ Права імплікація

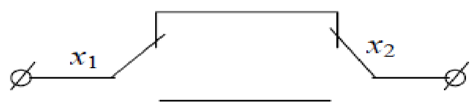




$F(x_1, x_2) = x_1 \mid x_2$ Штрих Шеффера

$F(x_1, x_2) = x_1 \downarrow x_2$ Стрілка Пірса





$F(x_1, x_2) = x_1 \oplus x_2$ Сума за модулем 2

$F(x_1, x_2) = x_1 \sim x_2$ Еквівалентність

Рис. 4.7. Технічна реалізація деяких функцій двох змінних

Основні властивості функцій алгебри логіки

Формулою алгебри логіки, або **логічним виразом**, називається скінчена послідовність булевих змінних, функцій та їх суперпозицій, пов'язаних знаками логічних операцій і круглими дужками.

Суперпозицією називається спосіб одержання нових функцій шляхом підстановки значень одних функцій замість значень аргументів інших функцій, при цьому деякі з функцій можуть тотожно збігатися з однією зі змінних.

Принцип суперпозиції

Функцію, що відповідає формулі *F*, називають суперпозицією функцій з множини функцій, а процес здобуття функції з множини функцій – операцією суперпозиції.

Приклад 4.21. Функція $F_1 = (((x_1 x_2) + x_1) + x_2)$ будується за три кроки (таблиця 1):

- $(x_1 x_2)$
- $((x_1 x_2) + x_1)$
- $(((x_1 x_2) + x_1) + x_2) = F_1$

Таблиця 1

x_1	x_2	$x_1 x_2$	$(x_1 x_2)+x_1$	$F_1 = (((x_1 x_2)+x_1)+x_2)$
0	0	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	1	1

Приклад 4.22. Функція $F_2 = \overline{x_1}(x_2 + x_3)$ будується також за три кроки (таблиця 2):

- $(x_2 + x_3)$
- $(\overline{x_1})$
- $((x_2 + x_3)\overline{x_1}) = F_2$

Для складання логічного висловлення із простих використовується принцип суперпозиції, тобто підстановка у функцію замість її аргументу інших функцій. Замість будь-якої змінної використовується як власне незалежна змінна, аргумент, так і змінна, що є функцією інших змінних. Цей принцип є правильним також у звичайній алгебрі. За принципом суперпозиції з двомісних булевих функцій можна побудувати будь-яку булеву функцію.

Принцип суперпозиції дає змогу на основі трьох основних елементарних функцій (заперечення, кон’юнкція, диз’юнкція) здобути складене логічне висловлення, що описує функціонування цифрових систем і автоматів.

Якщо F_1 і F_2 – формули, то $\overline{F_1}, (F_1 \wedge F_2)(F_1 \vee F_2), (F_1 \rightarrow F_2), (F_1 \approx F_2)$ – також формули.

На відміну від табличного задання, зображення функції формулою не єдине.

Формули, що зображають ту саму функцію називають **еквівалентними** або **рівносильними.**

Таблиця 2

x_1	x_2	x_3	$x_2 + x_3$	$\overline{x_1}$	$F_2 = \overline{x_1}\ (x_2 + x_3)$
0	0	0	0	1	0
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

Приклад 4.23. Функцію штрих Шеффера можна зобразити за допомогою основних операцій булевої алгебри формулами:

$$F = x_1 | x_2 = \overline{x_1 x_2} = \overline{x_1} \vee \overline{x_2} \ ,$$

а функцію стрілка Пірса:

$$F = x_1 \downarrow x_2 = \overline{x_1 \vee x_2} = \overline{x_1} \wedge \overline{x_2}$$

Функція алгебри логіки – це рівність, у лівій частині якої стоїть булева змінна, а у правій − логічний вираз. Отже, функція алгебри логіки визначається формулою. Наприклад, $x_1(x_2 \rightarrow x_3)$ − логічний вираз, $y = x_1 + x_2\ x_3$ – булева функція.

Для обчислення логічних виразів дотримуються такого пріоритету операцій: насамперед обчислюються функції, потім заперечення, після чого логічне множення і, врешті, логічне додавання. Вирази, які стоять у дужках, обчислюються в першу чергу. Інші операції мають найменший пріоритет. Порядок їхнього виконання визначається круглими дужками.

Якщо у формулі відсутні дужки, то операції виконуються в такій послідовності:

заперечення;

кон’юнкція;

диз’юнкція;

імплікація;

еквівалентність;

Для стрілки Пірса і штриха Шеффера пріоритет не визначений.

Функції, які зводяться до залежності від меншого числа змінних, називаються **виродженими**, а функції, які суттєво залежать від усіх змінних, є **невиродженими**. Наприклад, серед функцій однієї змінної є дві вироджені функції. Це *F(x) = 0*, *F(x)* = 1, які можна розглядати як функції від нуля змінних.

Функція багатьох змінних $f(x_1, x_2, ..., x_n)$ називається функцією, яка **зберігає константу 0**, якщо $f(0, 0, ..., 0) = 0$.

Функція багатьох змінних $f(x_1, x_2, ..., x_n)$ називається функцією, яка **зберігає константу 1**, якщо $f(1, 1, ..., 1) = 1$.

Логічна функція $f^*(x_1, x_2, ..., x_n)$ називається **двоїстою** до функції $f(x_1, x_2, ..., x_n)$, якщо має місце рівність $f^*(x_1, x_2, ..., x_n) = \overline{f}\left(\overline{x_1}, \overline{x_2}, ..., \overline{x_n}\right)$. Наприклад, функція $f_1 = x_1 \cdot x_2$ має властивість двоїстості до функції $f_2 = x_1 + x_2$, тому що $x_1 \cdot x_2 = \overline{\overline{x_1} + \overline{x_2}}$.

$x1$	$x2$	$f1 = x1 \cdot x2$
0	0	0
0	1	0
1	0	0
1	1	1

$\overline{x_1}$	$\overline{x_2}$	$f2 = \overline{x_1} + \overline{x_2}$	$\overline{f_2}$
1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

Логічна функція $f(x_1, x_2, ..., x_n)$ називається **самодвоїстою**, якщо має місце рівність

$$f(x_1, x_2, ..., x_n) = \overline{f}\left(\overline{x_1}, \overline{x_2}, ..., \overline{x_n}\right).$$

Наприклад, функція $f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_2 \cdot x_3 + x_1 \cdot x_3$ є самодвоїстою, тому що $x_1 \cdot x_2 + x_2 \cdot x_3 + x_1 \cdot x_3 = \overline{\overline{x_1} \cdot \overline{x_2} + \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_3}}$.

Принцип двоїстості

Для того щоб одержати двоїсту формулу булевої алгебри, потрібно замінити в ній всі кон’юнкції на диз’юнкції, диз’юнкції на кон’юнкції, 0 на

1, 1 на 0 і використовувати дужки, де потрібно, щоб порядок операцій залишався попереднім.

Приклад 4.24. Знайти функцію двоїсту до функції $f = x_1 \vee \overline{x_2}\ x_3 \vee 1$.

Розв'язок. Скористаємося правилом отримання двоїстих функцій $f* = x_1 \wedge x_2 \vee x_3 \wedge 0$.

Доведемо за допомогою таблиці істинності.

x_1	x_2	x_3	$\overline{x_2}$	$\overline{x_2}x_3$	$x_1 \vee \overline{x_2}x_3$	$x_1 \vee \overline{x_2}x_3 \vee 1$
0	0	0	1	0	0	1
0	0	1	1	1	1	1
0	1	0	0	0	0	1
0	1	1	0	0	0	1
1	0	0	1	0	1	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	1	0	0	1	1

x_1	x_2	x_3	$\overline{x_1}$	$\overline{x_3}$	$x_2 \vee \overline{x_3}$	$\overline{x_1} \wedge (x_2 \vee \overline{x_3})$	$f^* = x_1 \wedge (\overline{x_2} \vee x_3) \wedge 0$	$\overline{f^*}$
0	0	0	1	1	1	1	0	1
0	0	1	1	0	0	0	0	1
0	1	0	1	1	1	1	0	1
0	1	1	1	0	1	1	0	1
1	0	0	0	1	1	0	0	1
1	0	1	0	0	0	0	0	1
1	1	0	0	1	1	0	0	1
1	1	1	0	0	1	0	0	1

Булева функція називається **монотонною**, якщо для будь-яких пар наборів значень змінних $(x_1, x_2, ..., x_n)$ та $(y_1, y_2, ..., y_n)$ для яких виконується відношення $(x_1.x_2, ..., x_n) \le (y_1, y_2, ..., y_n)$ правильна і нерівність $f(x_1, x_2, ..., x_n) \le f(y_1, y_2, ..., y_n)$.

Приклад 4.25. Дослідити на монотонність функцію $f(x,y) = x \sim y$.

Розв'язок. Запишемо всі набори значень змінних, для яких виконується відношення порядку, визначимо значення функцій на цих наборах і порівняємо їх.

$(0;0) \le (0;1) \quad f(0;0) = 1 \quad f(0;1) = 0 \quad f(0;0) \not\le f(0;1);$

$(0;0) \le (1;0) \quad f(0;0) = 1 \quad f(1;0) = 0 \quad f(0;0) \not\le f(1;0);$

$(0;0) \le (1;1) \quad f(0;0) = 1 \quad f(1;1) = 1 \quad f(0;0) \le f(1;1);$

$(0;1) \le (1;0) \quad f(0;1) = 0 \quad f(1;0) = 0 \quad f(1;0) \le f(0;1);$

$(0;1) \le (1;1) \quad f(0;1) = 0 \quad f(1;1) = 1 \quad f(0;1) \le f(1;1);$

$(1;0) \le (1;1) \quad f(1;0) = 0 \quad f(1;1) = 1 \quad f(1;0) \le f(1;1)$

$$f(x,y) = x \sim y$$

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	1

Функція $f(x,y) = x \sim y$ не є монотонною.

Функція багатьох змінних називається **лінійною**, якщо її можна подати у вигляді многочлена

$f(x_1, x_2, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n$, що не містить кон'юнкцій, де $a_i \in \{0, 1\}$, $i = 0, n$. Наприклад, функція $x \sim y = x \oplus y \oplus 1$ є лінійною.

Повні системи функцій. Базис

Визначення. Система функцій алгебри логіки $\{f_1, f_2, \dots, f_n\}$ називається **повною**, якщо кожна інша функція алгебри логіки може бути виражена за допомогою суперпозицій цих функцій. При цьому стверджують, що повна система функцій утворює базис у логічному просторі.

Визначення. Мінімальним базисом є такий базис, вилучення з якого будь-якої функції порушує його повноту.

Класами Поста називають такі п’ять множин булевих функцій:

T_0 – клас функцій, що зберігає 0;

$T_0 = \{f \in P | f(0,0,\dots,0) = 0\}$.

T_1 – клас функцій, що зберігає 1;

$T_1 = \{f \in P | f(1,1,\dots,1) = 1\}$.

S – клас самодвоїстих функцій;

$S = \{f \in P | \forall (a_1, a_2,\dots, a_n) f(a_1, a_2,\dots, a_n) = \overline{f}(\overline{a_1}, \overline{a_2},\dots, \overline{a_n})\}$.

M – клас монотонних функцій;

$M = \{f \in P | \forall \alpha \forall \beta \alpha \le \beta \to f(\alpha) \le f(\beta)\}$.

L – клас лінійних функцій;

$L = \{f \in P | f(x_1, x_2,\dots, x_n) = a_0 \oplus a_1 x_1 \oplus \dots \oplus a_n x_n\}$.

Булева функція $f(x_1, x_2,\dots, x_n)$ може належати одному або декільком класам Поста й не належати до жодного.

Теорема Поста – Яблонського. Для того щоб система функцій була повною, необхідно і достатньо, щоб вона містила в собі хоча б одну функцію: ту, що не зберігає константу 0, ту, що не зберігає константу 1, несамодвоїсту, немонотонну й нелінійну.

З теореми випливає, що таких функцій має бути п’ять. Але, через те що деякі функції мають одразу кілька потрібних властивостей, базис може складатися з меншого числа функцій.

Властивості функцій	**Функції**									
	0	**1**	–	+	•	/	↓	→	⊕	←
Незберігаюча 0		*	*			*	*	*		
Незберігаюча 1	*		*			*	*		*	*
Несамодвоїста	*	*		*	*	*	*	*	*	*
Немонотонна			*			*	*	*	*	*
Нелінійна				*	*	*	*	*		*

З таблиці видно, що повними системами функцій будуть: {¬, +, •},
{ ¬, +},
{¬, •}, {/}, {↓}, {0, →} тощо.

Приклад 4.26. Визначити за допомогою теореми Поста, чи є система $\{x \vee y, \overline{x}\}$ функціонально повною.

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1
$\overline{x}$	$\overline{y}$	$\overline{\overline{x} \vee \overline{y}}$
1	1	0
1	0	0
0	1	0
0	0	1

1) $f(x) = x \vee y = xy \oplus x \oplus y$ – функція нелінійна, бо має кон'юнкцію;
2) зберігає 0 $f(0,0) = 0$;
3) зберігає 1 $f(1,1) = 1$;
4) (0,0)$\le$(0,1); f (0, 0) =0; f (0,1) =1; f (0,0)$\le$ f (0,1);
   (0,0)$\le$(1,0); f (0, 0) =0; f (1, 0) =1; f (0,0)$\le$ f (1,0);
   (0,0)$\le$(1,1); f (0, 0) =0; f (1, 1) =1; f (0,0)$\le$ f (1,1);
   (0,1)$\le$(1,0); f (0, 1) =1; f (1, 0) =1; f (0,1)$\le$ f (1,0);
   (0,1)$\le$(1,1); f (0, 1) =1; f (1, 1) =1; f (0,1)$\le$ f (1,1);
   (1,0)$\le$(1,1); f (1, 0) =1; f (1, 1) =1; f (1,0)$\le$ f (1,1);
5) несамодвоїста $f(x, y) \ne \overline{f}(\overline{x}, \overline{y})$.

x	$\overline{x}$
0	1
1	0
$\overline{x}$	$\overline{\overline{x}}$
1	1
0	0

1) $f(x) = \overline{x} = 1 \oplus x$ – лінійна;
2) не зберігає 0 $f(0) \ne 0$;

3) не зберігає 1 $f(1) \neq 1$;

4) самодвоїста $f(x) = \overline{f}(\overline{x})$.

Зведемо всі дані в таблицю:

Назва властивості	$\overline{x}$	$x \vee y$
Лінійність	–	+
Монотонність	–	+
Зберігає 0	–	+
Зберігає 1	–	+
Самодвоїстість	+	–

У кожному рядку таблиці присутній знак «мінус». Отже, для кожного класу є хоча б одна функція, що не зберігає 0, що не зберігає 1, нелінійна, немонотонна і несамодвоїста. За теоремою Поста така система функцій є функціонально повною.

Приклад 4.27. Довести повноту або неповноту приведеної системи булевих функцій $\{f_1 = x_1 \wedge x_2,\ f_2 = 0,\ f_3 = x_1 \approx x_2\}$.

Застосуємо теорему Поста про повноту. Для того щоб система булевих функцій була повною необхідно і достатньо, щоб вона містила хоча б одну функцію, що не зберігає 0, хоча б одну, що не зберігає 1, хоча б одну нелінійну функцію, хоча б одну несамодвоїсту і хоча б одну немонотонну функції. Складемо таблицю істинності функцій.

x_1	x_2	f_1	f_2	f_3
0	0	0	0	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Функція f_3 не зберігає 0, оскільки f_3 (0,0) = 1. Функція f_2 не зберігає 1, так як f_2 (1,1) = 0. Функція f_1 – несамодвоїста, оскільки $f_1(0,1) \neq \overline{f_1}(1,0) =$ 1. Функція f_3 – немонотонна, так як для впорядкованих наборів $(0,0) \leq (0,1)$

вона приймає значення f_3 (0,0) = 1 ≥ f_3 (0,1). Функція f_1 – нелінійна. Покажемо це. Нехай

$f_1 = a_0 \oplus a_1 x_1 \oplus a_2 x_2$

$f_1(0,0) = a_0 \oplus 0 \oplus 0 = 0 \Rightarrow a_0 = 0$;

$f_1(0,1) = a_0 \oplus 0 \oplus a_2 = 0 \oplus a_2 = 0 \Rightarrow a_2 = 0$; $f = 0$, що не є правильним.

$f_1(1,0) = a_0 \oplus a_1 = 0 \oplus a_1 = 0 \Rightarrow a_1 = 0$.

Таким чином, за теоремою Поста система $\{f_1 = x_1 \wedge x_2,\ f_2 = 0, x_3 = x_1 \approx x_2\}$ повна, бо містить функцію, що не зберігає 0, функцію, що не зберігає 1, несамодвоїсту функцію, нелінійну функцію і немонотонну функцію.

Приклад 4.28. Визначити, до яких класів Поста належить функція $F = \overline{x_1} x_3 \vee x_1 \overline{x_3}$, додати, якщо це потрібно, до F елементарні функції, щоб отримана множина була повною.

Складемо таблицю істинності для функції $F = \overline{x_1} x_3 \vee x_1 \overline{x_3}$:

x_1	x_2	x_3	$\overline{x_1}$	$\overline{x_3}$	$\overline{x_1} x_3$	$x_1 \overline{x_3}$	F		
0	0	0	1	1	0	0	0		0 1 0 1 1 0 1 0
0	0	1	1	0	1	0	1	$x_3 \Rightarrow$	1 1 1 0 1 1 1
0	1	0	1	1	0	0	0		0 0 1 1 0 0
0	1	1	1	0	1	0	1		0 1 1 1 0
1	0	0	0	1	0	1	1	$x_1 \Rightarrow$	1 0 0 1
1	0	1	0	0	0	0	0		1 0 1
1	1	0	0	1	0	1	1		1 1
1	1	1	0	0	0	0	0		0

Функція F зберігає 0 ($F \in T_0$), оскільки F (0, 0, 0) = 0.

Функція Fне зберігає 1 ($F \notin T_1$), оскільки F (1, 1, 1) = 0.

Функція Fне є монотонною ($F \notin M$), оскільки на порівнюваних наборах (0,0,1)≤(1,0,1), отримуємо F(0,0,1)=1 і F(1,0,1)=0 F(0,0,1) ≥F(1,0,1).

Функція F не є самодвоїстою ($F \notin S$), оскільки на протилежних наборах приймає однакові значення F(0,0,0)=F(1,1,1)=0

Функція F є лінійною ($F \in L$), оскільки її поліном має вигляд $F = x_1 \oplus x_3$.

Щоб доповнити функцію до повної системи, потрібно ввести функцію, що не зберігає 0 і є нелінійною, як таку функцію можна вибрати $G = x \to y$ (імплікація). Множина $\{F, G\}$ – повна.

Булева алгебра та її основні закони

Визначення. Булевою алгеброю називається множина логічних функцій з операціями диз'юнкція, кон'юнкція та заперечення, тобто алгебра, базисом якої є система функцій { ¬, +, •}.

Операції булевої алгебри звичайно називають булевими операціями, а функції − булевими функціями.

Розглянемо тепер основні закони булевих операцій:

1) комутативний (для диз'юнкції та кон'юнкції):

$$x_1 \vee x_2 = x_2 \vee x_1; \quad x_1 \wedge x_2 = x_2 \wedge x_1;$$

2) асоціативний:

$$x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3; \quad x_1 \wedge (x_2 \wedge x_3) = (x_1 \wedge x_2) \wedge x_3;$$

3) дистрибутивний:

$x_1 \wedge (x_2 \vee x_3) = x_1 \wedge x_2 \vee x_1 \wedge x_3$ – перший дистрибутивний закон;

$x_1 \vee (x_2 \wedge x_3) = (x_1 \vee x_2) \wedge (x_1 \vee x_3)$ – другий дистрибутивний закон;

4) ідемпотентний:

$$x \vee x = x; x \wedge x = x;$$

5) інверсний (формули де Моргана):

$$\overline{x_1 \vee x_2} = \overline{x_1} \wedge \overline{x_2}; \quad \overline{x_1 \wedge x_2} = \overline{x_1} \vee x_2;$$

6) закон вилученого третього (для диз'юнкції) і закон суперечності (для кон'юнкції):

$$x \vee \overline{x} = 1; x \wedge \overline{x} = 0;$$

7) закони елімінації:

$$x_1 \wedge (x_1 \vee x_2) = x_1; \quad x_1 \vee (x_1 \wedge x_2) = x_1$$

У булевій алгебрі мають місце такі властивості:

$\bar{0} = 1$; $\bar{1} = 0$; $x \vee 0 = x$; $x \vee 1 = 1$; $x \wedge 0 = 0$; $x \wedge 1 = x$; $\bar{\bar{x}} = x$.

Решта функцій двох змінних логіки виражаються через базис булевої алгебри в такий спосіб:

$$x_1 \rightarrow x_2 = \overline{x_1} \vee x_2; \; x_1 \leftarrow x_2 = x_1 \wedge \overline{x_2};$$

$$x_1 \mid x_2 = \overline{x_1} \vee \overline{x_2}; \; x_1 \downarrow x_2 = \overline{x_1} \wedge \overline{x_2};$$

$$x_1 \oplus x_2 = x_1 \wedge \overline{x_2} \vee \overline{x_1} \wedge x_2; \;\; x_1 \sim x_2 = x_1 \wedge x_2 \vee \overline{x_1} \wedge \overline{x_2}.$$

У справедливості цих формул легко переконатися за допомогою таблиці істинності.

Закони булевої алгебри та її властивості надають можливість виконувати перетворювання логічних виразів з метою побудови найбільш простих (компактних) формул.

Приклад 4.29. Довести справедливість формул поглинання:

$$x_1 \vee x_1 \wedge x_2 = x_1; \quad x_1 \wedge (x_1 + x_2) = x_1.$$

Доведення

$$x_1 \vee x_1 \wedge x_2 = x_1 \wedge 1 \vee x_1 \wedge x_2 = x_1 \wedge (1 \vee x_2) = x_2 \wedge 1 = x_1;$$

$$x_1 \wedge (x_1 \vee x_2) = x_1 \wedge x_1 \vee x_1 \wedge x_2 = x_1 \vee x_1 \wedge x_2 = x_1.$$

Приклад 4.30. Спростити: $((x_1 \downarrow x_2) \mid x_3) \rightarrow (x_1 \rightarrow x_3)$.

Розв'язок:

$$\left(\left(\overline{x_1} \wedge \overline{x_2}\right) \middle| x_3\right) \rightarrow \left(\overline{x_1} \vee x_3\right) = \left(\overline{\overline{x_1} \wedge \overline{x_2}} \vee \overline{x_3}\right) \rightarrow \left(\overline{x_1} \vee x_3\right) = \overline{\overline{\overline{x_1}} \wedge \overline{\overline{x_2}} \vee \overline{x_3}} \vee \overline{x_1} \vee x_3 =$$

$$= \overline{x_1} \vee x_3 \wedge \left(1 \vee \overline{x_1} \wedge \overline{x_2}\right) = \overline{x_1} \vee x_3$$

Нормальні форми булевих функцій

Визначення. **Елементарною диз'юнкцією (кон'юнкцією)** називається диз'юнкція (кон'юнкція) скінченного числа булевих змінних, у якій кожна змінна зустрічається не більше одного разу в прямому чи інверсному вигляді.

Наприклад:

$x_1 \vee \overline{x_2}$, $\overline{x_1} \vee x_2 \vee \overline{x_4}$ – елементарні диз'юнкції;

$x_1 \wedge x_2 \wedge \overline{x_3}$, $\overline{x_5} \wedge x_7 \wedge \overline{x_9} \wedge x_{10}$ – елементарні кон'юнкції.

Визначення. Диз'юнктивною нормальною формою (кон'юнктивною нормальною формою) називається формула, яка містить диз'юнкцію (кон'юнкцію) скінченного числа різних елементарних кон'юнкцій (диз'юнкцій).

Позначення: ДНФ, КНФ.

Наприклад: $x_1 \wedge \overline{x_2} \vee x_1 \wedge x_3$, $x_1 \wedge x_5 \vee \overline{x_6}$ – ДНФ;

$(x_1 \vee x_2) \wedge \overline{x_3}$, $(\overline{x_1} \vee x_3) \wedge (x_1 \vee \overline{x_4}) \wedge (x_2 \vee x_5)$ – КНФ.

Визначення. ДНФ (КНФ) називається **досконалою** і позначається ДДНФ. (ДКНФ), якщо в кожній її елементарній кон'юнкції (диз'юнкції) подано всі змінні.

Наприклад: $x_1 \wedge \overline{x_2} \wedge x_2 \vee x_1 \wedge x_2 \wedge \overline{x_3}$ – ДДНФ;

$(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee x_3 \vee x_4)$ – ДКНФ.

Для того щоб привести формулу до ДДНФ, потрібно:

– за допомогою законів і властивостей булевої алгебри привести її до ДНФ;

– якщо в елементарній кон'юнкції не міститься змінної x_i із загальної кількості змінних, які входять до цієї формули, додати до цієї кон'юнкції співмножник $x_i \vee \overline{x_i}$ і розкрити дужки;

– з однакових елементарних кон'юнкцій вилучити всі, крім однієї.

Приклад 4.31. $x_1 \wedge x_2 \vee \overline{x_3} = x_1 \wedge x_2 (x_3 \vee \overline{x_3}) \vee (x_1 \vee \overline{x_1}) \wedge (x_2 \vee \overline{x_2}) \wedge$ $\wedge \overline{x_3} = x_1 \wedge x_2 \wedge x_3 \vee x_1 \wedge x_2 \wedge \overline{x_3} \vee x_1 \wedge x_2 \wedge \overline{x_3} \vee x_1 \wedge \overline{x_2} \wedge \overline{x_3} \vee \overline{x_1} \wedge x_2 \wedge$ $\overline{x_3} \vee \overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3} = x_1 \wedge x_2 \wedge x_3 \vee x_1 \wedge x_2 \wedge \overline{x_3} \vee x_1 \wedge \overline{x_2} \wedge \overline{x_3} \vee \overline{x_1} \wedge x_2 \wedge \overline{x_3}$ $\vee \vee \overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}$ – ДДНФ.

Для того щоб привести формулу до ДКНФ, доцільно спочатку привести її до ДНФ, а потім від ДНФ перейти до КНФ в такий спосіб.

Нехай ДНФ має вигляд $f = c_1 \vee c_2 \vee \ldots \vee c_m$, де c_i – елементарні кон'юнкції, $i = \overline{1, m}$.

Формулу $\overline{c_1 \vee c_2 \vee \ldots \vee c_m}$ приведемо до ДНФ $k_1 \vee k_2 \vee \ldots \vee k_l$, де k_i – елементарні кон'юнкції.

Тоді $f = \overline{\overline{c_1 \vee c_2 \vee \ldots \vee c_m}} = \overline{k_1 \vee k_2 \vee \ldots \vee k_l} = \overline{k_1} \wedge \overline{k_2} \wedge \ldots \wedge \overline{k_l}$.

Застосовуючи правило де Моргана, перетворимо елементарні кон'юнкції k_i на елементарні диз'юнкції D_i, де $i = \overline{1,k}$. Отже, дістанемо КНФ.

$$f = D_1 \wedge D_2 \wedge \ldots \wedge D_l .$$

І, врешті, використовуючи закон суперечності та другий дистрибутивний закон, зробимо перехід від КНФ до ДКНФ.

Приклад 4.32.

1) $\overline{x_1x_2x_3} \wedge (x_1x_2 \to x_1\overline{x_2x_3}) = (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1x_2} \vee x_1\overline{x_2x_3}) == (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_1) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge$
$(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) = (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) =$
$=(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3\overline{x_3}) = (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge\wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) = (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$ - ДКНФ.

2) $x_1\overline{x_2} \vee \overline{x_1x_2} = \overline{\overline{x_1\overline{x_2} \vee \overline{x_1}x_2}} = \overline{\overline{x_1x_2} \wedge \overline{\overline{x_1}x_2}} =$
$\overline{(\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2})} = \overline{\overline{x_1x_2} \vee x_1x_2} = \overline{\overline{x_1x_2}} \wedge \overline{x_1x_2}$ - ДКНФ

Визначення. Елементарна диз'юнкція (кон'юнкція), яка містить усі змінні, називається **конституентою нуля** (**одиниці**). Наприклад, якщо загальна кількість змінних $n = 3$, то $\overline{x_1} \vee x_2 \vee x_3$– конституента нуля, а $x_1 \wedge \overline{x_2} \wedge x_3$– конституента одиниці.

Вочевидь, що конституента нуля перетворюється на нуль лише за одного набору значень змінних. У нашому прикладі конституенті нуля відповідає набір (1, 0, 0). Аналогічно, конституента одиниці перетворюється на одиницю також лише за одного набору. Наприклад, конституенті одиниці $x_1 \wedge \overline{x_2} \wedge x_3$ відповідає набір (1, 0, 1).

Оскільки для заданої булевої функції її ДДНФ являє собою диз'юнкцію конституент одиниці, а її ДКНФ − це кон'юнкція конституент нуля, то ця функція перетворюється на одиницю чи нуль лише за відповідних цим конституентам наборів значень змінних. Справедливе є і зворотне твердження.

Це дає змогу за заданою таблицею істинності булевої функції відразу записати її досконалі нормальні форми і, навпаки, за заданою ДНФ – скласти таблицю істинності.

Досконалі форми для функції $f(x_1, x_2, \ldots, x_n)$ позначають:

для ДДНФ – $f(x_1, x_2, \ldots, x_n) = \bigvee_{1} (x_1, x_2, \ldots, x_n)$;

для ДКНФ – $f\,(x_1, x_2, \ldots, x_n) = \underset{0}{\wedge}\,(x_1, x_2, \ldots, x_n)$, де символ $\underset{1}{\vee}$або $\underset{0}{\wedge}$ позначає, що диз'юнкція або кон'юнкція виконуються за відповідними конституентами.

Приклад 4.33. Побудувати таблицю істинності для висловлювання: $(x\ |\bar{y}) \to (y \oplus z)$, побудувати ДДНФ, ДКНФ.

Розв'язок.

Будуємо таблицю істинності – таблицю, за допомогою якої встановлюється істинне значення складного висловлювання за цих значень простих висловлювань, що до нього входять.

За таблицею складаємо диз'юнктивну нормальну форму (ДНФ). ДНФ в булевій логіці — нормальна форма, у якій булева формула має вигляд диз'юнкції декількох кон'юнктів.

Алгоритм отримання ДДНФ за таблицею істинності:

1. Відзначаємо ті рядки, в останньому стовпчику яких стоять 1.

2. Виписати для кожного відзначеного рядка кон'юнкцію всіх змінних таким чином: якщо значення деякої змінної в цьому рядку = 1, то в кон'юнкцію включають саму цю змінну, якщо = 0, то її заперечення.

3. Усі отримані кон'юнкції зв'язати в диз'юнкцію:

Вибираємо в таблиці рядки, у яких булева функція приймає значення 1. У цьому випадку – це 2, 3, 4, 6 і 7-й рядки.

Для кожного рядка складаємо кон'юнкцію: якщо значення змінної дорівнює 0, то беремо її заперечення, а якщо 1, то беремо саму змінну. Потім складаємо диз'юнкцію отриманих кон'юнкцій: $f(x, y, z) = (x \wedge \bar{y} \wedge z) \vee (x \wedge y \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{x} \wedge \bar{y} \wedge z) \vee (x \wedge \bar{y} \wedge z)$.

x	y	z	$\bar{y}$	$x\ \vert\bar{y}$	$y \oplus z$	
1	1	1	0	1	0	0
1	0	1	1	0	1	1
1	1	0	0	1	1	1
1	0	0	1	0	0	1
0	1	1	0	1	0	0
0	0	1	1	1	1	1
0	1	0	0	1	1	1
0	0	0	1	1	0	0

Вибираємо в таблиці рядки, у яких булева функція набуває значення 0. У цьому випадку – це 1-й, 5-й і 8-й рядки:

$f(x, y, z) = (\overline{x} \vee \overline{y} \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (x \vee y \vee z)$.

Приклад 4.34. Перетворити функцію $\{0,3,5\}_{x_1 x_2 x_3}$ на ДДНФ.

Розв'язок. y = {(0,0,0), (011), (101)}$x_1 x_2 x_3$ = $\overline{x_1 x_2 x_3} \vee \overline{x_1} x_2 x_3 \vee x_1 \overline{x_2} x_3$ – ДДНФ.

Приклад. 4.35. Для функції, заданої власною ДКНФ

$y = (x_1 \vee x_2 \vee \overline{x_3}\ ) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$, скласти таблицю істинності.

Розв'язок

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Обов'язкові завдання

Для заданої логічної функції $F = (\overline{\overline{A} \vee B \cdot \overline{C}}) \cdot ((\overline{\overline{B \downarrow C}) \cdot D})$

1) знайти ДНФ;

2) скласти таблицю істинності і діаграму Карно (для самостійного вивчення, будуть питання, пишіть);

3) отримати мінімальну ДНФ;

4) від мінімальної ДНФ перейти до КНФ.

4.3. Алгебра Жегалкіна та її основні закони

Визначення. Алгеброю Жегалкіна називається множина логічних функцій з операціями кон'юнкція, додавання за модулем 2 і константа 1, тобто алгебра, базисом якої є система функцій $\{1, \wedge, \oplus\}$.

Наведемо основні закони цієї алгебри:

1) комутативний:

$$x_1 \wedge x_2 = x_2 \wedge x_1; \quad x_1 \oplus x_2 = x_2 \oplus x_1;$$

2) асоціативний:

$$x_1 \wedge (x_2 \wedge x_3) = (x_1 \wedge x_2) \wedge x_3; \; x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3;$$

3) дистрибутивний:

$$x_1 (x_2 \oplus x_3) = x_1 x_2 \oplus x_1 x_3;$$

4) ідемпотентний:

$$x \wedge x = x;$$

5) закон приведення подібних членів:

$$x \oplus x = 0.$$

В алгебрі Жегалкіна мають місце такі властивості:

$$x \oplus 0 = x; \quad x \wedge 0 = 0; \quad x \wedge 1 = x.$$

Решта операцій алгебри логіки виражаються через базис цієї алгебри в такий спосіб:

$$\overline{x} = x \oplus 1; \quad x_1 \vee x_2 = x_1 \oplus x_2 \oplus x_1 \wedge x_2; \quad x_1 \sim x_2 = 1 \oplus x_1 \oplus x_2;$$

$$x_1 \to x_2 = 1 \oplus x_1 \oplus x_1 \wedge x_2; \qquad x_1 \leftarrow x_2 = x_1 \oplus x_1 \wedge x_2;$$

$$x_1 \downarrow x_2 = 1 \oplus x_1 \oplus x_2 \oplus x_1 \wedge x_2; \qquad x_1 \mid x_2 = 1 \oplus x_1 \wedge x_2.$$

Визначення. Функція алгебри Жегалкіна, подана у вигляді суми за модулем 2 добутків незалежних змінних, називається канонічним многочленом, або **поліномом Жегалкіна**.

Поліномом Жегалкіна називається скінченна сума за модулем 2 попарно різних елементарних кон'юнкцій над множиною змінних $\{x_1, x_2, ..., x_n\}$. У зарубіжній літературі представлення полінома Жегалкіна зазвичай називається алгебраїчною нормальною формою (АНФ).

Теорема Жегалкіна – стверджує існування й унікальність будь-якої булевої функції у вигляді полінома Жегалкіна. Формально поліном Жегалкіна можна представити у вигляді:

$$P(x_1,...x_n) = a \oplus a_1 \wedge x_1 \oplus a_2 \wedge x_2 \oplus ... \oplus a_n x_n \oplus a_{12} \wedge x_1 \wedge x_2 \oplus a_{13} \wedge x_1 \wedge x_3 \oplus .... \oplus a_{1..n} \wedge x_1 ... \wedge x_n$$

Кількість змінних, що входять до елементарної кон'юнкції називається **рангом елементарної кон'юнкції**. Кількість попарно різних елементарних кон'юнкцій у поліномі називається **довжиною полінома**.

Функція $f(x_1, x_2, x_3)$ від трьох змінних буде мати такий поліном Жегалкіна:

$$f(x_1, x_2, x_3) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus a_{23} x_2 x_3 \oplus a_{123} x_1 x_2 x_3.$$

Способи побудови полінома Жегалкіна

Способи побудови полінома Жегалкіна

- Метод еквівалентних перетворень
- Метод трикутника Паскаля
- Метод невизначених коефіцієнтів

Метод еквівалентних перетворень

Для функції $f(x_1, x_2, x_3)$ записують ДДНФ, потім виражають диз'юнкцію та заперечення через операції кон'юнкції та суми за модулем 2. Використовуючи тотожності алгебри Жегалкіна, отримують поліном.

Приклад 4.36. Зобразити поліном Жегалкіна для функції $x \sim y$.

Розв'язок. $x \sim y = xy \vee \overline{x}\,\overline{y} = xy\overline{x}\,\overline{y} \oplus xy \oplus \overline{x}\,\overline{y} = xy(\overline{x}\,\overline{y} \oplus 1) \oplus \overline{x}\,\overline{y} = xy \oplus (x \oplus 1)(y \oplus 1) =$
$= xy \oplus x \oplus x \oplus y \oplus 1 = x \oplus y \oplus 1.$

Приклад 4.37. Зобразити поліном Жегалкіна для функції $f(x_1, x_2, x_3) = (x_1 x_2 \vee x_3) \to \overline{x_2}$ **.**

Застосовуючи основні закони алгебри логіки, приведемо спочатку цю функцію до ДНФ: $f(x_1, x_2, x_3) = (x_1 x_2 \vee x_3) \to \overline{x_2}$ = (застосуємо рівнозначність $x \to y = \overline{x} \vee y) = \overline{(x_1 x_2 \vee x_3)} \vee \overline{x_2}$ = (застосуємо закон де Моргана) = $\overline{x_1 x_2} \wedge \overline{x_3} \vee \overline{x_2}$ = (застосуємо ще раз закон де Моргана) = $(\overline{x_1} \vee \overline{x_2}) \wedge \overline{x_3} \vee \overline{x_2}$ = (застосуємо закон дистрибутивності) = $\overline{x_1} \wedge \overline{x_3} \vee \overline{x_2} \wedge \overline{x_3} \vee \overline{x_2}$ = (застосуємо закон поглинання $\overline{x_2} \wedge \overline{x_3} \vee \overline{x_2} = \overline{x_2}$) = $\overline{x_1}\,\overline{x_3} \vee \overline{x_2}$ – ДНФ.

Далі в отриманій ДНФ потрібно позбутися диз'юнкцій, використовуючи закони де Моргана:

$$\overline{x_1}\,\overline{x_3} \vee \overline{x_2} = \overline{x_1} \wedge \overline{x_3} \vee \overline{x_2} = \overline{x_1 \vee x_3} \vee \overline{x_2} = \overline{\overline{\overline{x_1 \vee x_3}} \vee \overline{\overline{x_2}}} = \overline{\overline{\overline{x_1} \wedge \overline{x_3}} \wedge \overline{\overline{x_2}}} = \overline{\overline{x_1}\,\overline{x_3}\,x_2} \,.$$

Замінюємо кожне заперечення $\overline{x} = 1 \oplus x$ і застосовуємо згадані вище закони, отримаємо:

$$\overline{\overline{x_1}\,\overline{x_3}\,x_2} = 1 \oplus \overline{x_1}\,\overline{x_3}\,x_2 = 1 \oplus (1 \oplus (1 \oplus x_1)(1 \oplus x_3)x_2 = 1 \oplus (1 \oplus 1 \oplus x_3 \oplus x_1 \oplus x_{13})x_2 = 1 \oplus$$
$$\oplus (x_3 \oplus x_1 \oplus x_{13})x_2 = 1 \oplus x_2 x_3 \oplus x_1 x_2 \oplus x_1 x_2 x_3$$

Для побудови ДДНФ за таблицею істинності вибираємо набори, на яких функція f приймає значення, що дорівнює 1. Якщо значення змінної в цьому наборі дорівнює 0, то вона береться із запереченням, якщо значення змінної дорівнює 1, то змінна береться без заперечення. З'єднавши знаком кон'юнкції всі змінні відповідного набору, отримаємо елементарну кон'юнкцію. Тоді диз'юнкція всіх таких елементарних кон'юнкцій і є ДДНФ.

Перетворення ДДНФ

x_1	x_2	x_3	$x_1 x_2$	$x_1 x_2 \vee x_3$	$\overline{x_2}$	f
0	0	0	0	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	0	1
0	1	1	0	1	0	0
1	0	0	0	0	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	1	1	0	0

$$f(x_1, x_2, x_3) = \overline{x_1 x_2 x_3} \vee \overline{x_1 x_2} x_3 \vee \overline{x_1} x_2 \overline{x_3} \vee x_1 \overline{x_2 x_3} \vee x_1 \overline{x_2} x_3 .$$

Щоб побудувати поліном Жегалкіна через ДДНФ, потрібно виключити операції диз'юнкції і заперечення, потім розкрити дужки.

$$\begin{aligned}&f(x_1,x_2,x_3) = \overline{x_1 x_2 x_3} \vee \overline{x_1 x_2} x_3 \vee \overline{x_1} x_2 \overline{x_3} \vee x_1 \overline{x_2 x_3} \vee x_1 \overline{x_2} x_3 = (1\oplus x_1)(1\oplus x_2)(1\oplus x_3) \oplus (1\oplus x_1)\\
&(1\oplus x_2)x_3 \oplus (1\oplus x_1)(1\oplus x_3)x_2 \oplus x_1 x_3 (1\oplus x_2) = (1\oplus x_2 \oplus x_1 \oplus x_1 x_2)(1\oplus x_3) \oplus (1\oplus x_2 \oplus x_1 \oplus\\
&\oplus x_1 x_2)x_3 \oplus (1\oplus x_3 \oplus x_1 \oplus x_1 x_3)x_2 \oplus (1\oplus x_3 \oplus x_2 \oplus x_2 x_3)x_1 \oplus x_1 x_3 \oplus x_1 x_2 x_3 = (1\oplus x_3 \oplus x_2 \oplus\\
&\oplus x_2 x_3 \oplus x_1 \oplus x_1 x_3 \oplus x_1 x_2 \oplus x_1 x_2 x_3) \oplus (x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3) \oplus (x_2 \oplus x_2 x_3 \oplus x_1 x_2 \oplus x_1 x_2 x_3) \oplus\\
&\oplus (x_1 \oplus x_1 x_3 \oplus x_1 x_2 \oplus x_1 x_2 x_3) \oplus x_1 x_3 \oplus x_1 x_2 x_3 = 1 \oplus x_1 x_2 \oplus x_2 x_3 \oplus x_1 x_2 x_3\end{aligned}$$

– поліном Жегалкіна.

Метод трикутника Паскаля

Розглянемо метод на прикладі.

Приклад 4.38. Побудувати поліном Жегалкіна для функції *f*. Візьмемо функцію голосування $f(x_1, x_2, x_3) = x_1 x_2 \vee x_2 x_3 \vee x_1 x_3 = (00010111)$.

Крок 1. Будуємо таблицю значень функції (рядки таблиці йдуть в порядку збільшення двоїстих кодів). Таблицю краще розташовувати в лівій частині сторінки.

Крок 2. Побудова трикутника. Для цього беремо вектор значень функції і виписуємо його навпроти першого рядка таблиці. Далі заповнюємо трикутник, складаючи попарно сусідні значення за модулем 2. Результат додавання записуємо нижче.

Крок 3. Побудова полінома Жегалкіна. Нас цікавить ліва сторона трикутника (значення виділені жирним). Числа на лівій стороні (виділені жирним шрифтом) трикутника є коефіцієнтами полінома за монотонних кон'юнкцій, які відповідають наборам значень змінних. Тепер випишемо для наочності ці кон'юнкції. Кон'юнкції виписуємо за двоїстими наборами в лівій частині таблиці за таким принципом: якщо навпроти змінної x_i стоїть 1, то змінна входить в кон'юнкцію, в протилежному випадку – змінна відсутня в кон'юнкції. Набору (0, 0, 0) відповідає 1.

Якщо принцип отримання кон'юнкцій зрозумілий, то стовпчик із ним (краще) не виписувати, а відразу переходити до побудови полінома. Для побудови полінома потрібні кон'юнкції тільки з рядків з одиницями на лівій стороні трикутника. Це і є кон'юнкції, що входять до складу полінома Жегалкіна. Залишилося виписати сам поліном:

$$f(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_1x_3 .$$

x_1	x_2	x_3	f		
0	0	0	0	1	**0** 0 0 1 0 1 1 1
0	0	1	0	x_3	**0** 0 1 1 1 0 0
0	1	0	0	x_2	**0** 1 0 0 1 0
0	1	1	1	$x_2x_3 \Rightarrow$	**1** 1 0 1 1
1	0	0	0	x_1	**0** 1 1 0
1	0	1	1	$x_1x_3 \Rightarrow$	**1** 0 1
1	1	0	1	$x_1x_2 \Rightarrow$	**1** 1
1	1	1	1	$x_1x_2x_3 \Rightarrow$	**0**

Якщо змінних в функції не 3, а 4 і більше, то метод працює без змін, тільки змінюється розмір таблиці. Проте, на відміну від методу невизначених коефіцієнтів, розрахунки можна виконувати без зайвих зусиль.

Приклад 4.39. Знайдемо поліном Жегалкіна функції $f(x_1, x_2, x_3) = (x_1 x_2 \vee x_3) \to \overline{x_2}$, застосовуючи трикутник Паскаля.

Поясню ще раз, як заповнюється трикутник Паскаля. Верхній рядок трикутника задає вектор значень мулевої функції $f(x_1, x_2, x_3) = x_1 x_2 \vee x_2 x_3 \vee x_1 x_3 = (00010111)$. У кожному рядку, починаючи з другого, будь-який елемент такого трикутника обчислюється як сума за модулем 2 двох сусідніх елементів попереднього рядка.

Лівій стороні трикутника Паскаля відповідають набори значень змінних вхідної функції $f(x_1, x_2, x_3) = x_1 x_2 \vee x_2 x_3 \vee x_1 x_3 = (00010111)$. З'єднуючи знаком кон'юнкції змінні, значення яких в наборі дорівнюють 1, ми отримаємо доданок у поліномі Жегалкіна. Набору (0, 0, 0) відповідає x_3 і так далі. Оскільки одиницям лівої сторони трикутника відповідають доданки 1, $x_1 x_2; x_2 x_3; x_1 x_2 x_3$, то поліном Жегалкіна: $f(x_1, x_2, x_3) = 1 \oplus x_1 x_2 \oplus x_2 x_3 \oplus x_1 x_2 x_3$.

Метод невизначених коефіцієнтів

Нехай *P(x)* поліном Жегалкіна, який реалізує задану функцію.

Для двох змінних:

$$P(x_1, x_2) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_{12} x_1 x_2.$$

Для трьох змінних:

$$P(x_1, x_2, x_3) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus a_{23} x_2 x_3 \oplus a_{123} x_1 x_2 x_3$$

Коефіцієнти $a_0, a_1, a_2, a_3 \ldots$ – це невизначені коефіцієнти, які потрібно знайти.

Кожну змінну $x_1, x_2, x_3 \ldots$ розглянемо двійковим набором $\sigma_1, \sigma_2, \sigma_3 \ldots$ (наприклад, $x_1 = 001,\ x_2 = 101$).

Для кожного двійкового набору значень змінних записують 2^n рівнянь $f(\sigma_1, \ldots, \sigma_n) = P(\sigma_1, \ldots, \sigma_n)$. Розв'язавши їх, отримують коефіцієнти полінома.

x_1	x_2	x_3	x_1x_2	$x_1x_2 \vee x_3$	$\overline{x_2}$	f		
0	0	0	0	0	1	1	1	1 1 1 0 1 1 0 0
0	0	1	0	1	1	1	x_3	0 0 1 1 0 1 0
0	1	0	0	0	0	1	x_2	0 1 0 1 1 1
0	1	1	0	1	0	0	$x_2x_3 \Rightarrow$	1 1 1 0 0
1	0	0	0	0	1	1	x_1	0 0 1 0
1	0	1	0	1	1	1	x_1x_3	0 1 1
1	1	0	1	1	0	0	$x_1x_2 \Rightarrow$	1 0
1	1	1	1	1	0	0	$x_1x_2x_3 \Rightarrow$	1

Приклад 4.40. Знайдемо поліном Жегалкіна для функції $f(x_1, x_2, x_3) = (x_1x_2 \vee x_3) \rightarrow \overline{x_2}$, використовуючи метод невизначених коефіцієнтів. Для цього спочатку потрібно побудувати таблицю істинності для булевої функції $f(x_1, x_2, x_3) = (x_1x_2 \vee x_3) \rightarrow \overline{x_2}$.

Загальний вигляд полінома Жегалкіна для функції трьох змінних:

$$f(x_1, x_2, x_3) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus a_{23}x_2x_3 \oplus a_{123}x_1x_2x_3.$$

Послідовно представимо набори значень змінних і знаходимо коефіцієнти $a_0, a_1, \ldots a_{123}$.

$f(0,0,0) = a_0 = 1;$

$f(0,0,1) = a_0 \oplus a_3 = 1 \oplus a_3 = 1 \Rightarrow a_3 = 0;$

$f(0,1,0) = a_0 \oplus a_2 = 1 \oplus a_2 = 1 \Rightarrow a_2 = 0;$

$f(0,1,1) = a_0 \oplus a_2 \oplus a_3 \oplus a_{23} = 1 \oplus 0 \oplus 0 \oplus a_{23} = 1 \oplus a_{23} = 0 \Rightarrow a_{23} = 1;$

$f(1,0,0) = a_0 \oplus a_1 = 1 \oplus a_1 = 1 \Rightarrow a_1 = 0;$

$f(1,0,1) = a_0 \oplus a_1 \oplus a_3 \oplus a_{13} = 1 \oplus 0 \oplus 0 \oplus a_{13} = 1 \oplus a_{13} = 1 \Rightarrow a_{13} = 0;$

$f(1,1,0) = a_0 \oplus a_1 \oplus a_2 \oplus a_{12} = 1 \oplus 0 \oplus 0 \oplus a_{12} = 1 \oplus a_{12} = 0 \Rightarrow a_{12} = 1;$

$f(1,1,1) = a_0 + a_1 \oplus a_2 \oplus a_3 \oplus a_{12} \oplus a_{13} \oplus a_{23} \oplus a_{123} = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus a_{123} = 1 \oplus a_{123} = 0$
$\Rightarrow a_{123} = 1$

Підставляючи отримані коефіцієнти, отримаємо поліном Жегалкіна:

$$f(x_1, x_2, x_3) = 1 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_2x_3.$$

x_1	x_2	x_3	x_1x_2	$x_1x_2 \vee x_3$	$\overline{x_2}$	f
0	0	0	0	0	1	1

0	0	1	0	1	1	1
0	1	0	0	0	0	1
0	1	1	0	1	0	0
1	0	0	0	0	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	1	1	0	0

Обов'язкові завдання

1. Перевірити на повноту систему булевих функцій $\{x_1 \vee x_2, x_1 \to x_2\}$.

2. Перевірити, чи є повною система $J = \{x \to \overline{y}, \overline{x} \wedge y\}$.

3. Скласти поліном Жегалкіна методом невизначених коефіцієнтів і за допомогою трикутника Паскаля для $F = \overline{A}(A \wedge B) \to (A \wedge B \to B)$.

4.4. Мінімізація булевих функцій

Булеві функції, як відомо, можуть бути реалізовані різними формулами, проте для практики найбільше значення мають так звані мінімальні нормальні форми, у яких число входжень символів змінних найменше.

Для довільних функцій методів знаходження таких форм не існує, мінімізацію проводять лише для диз'юнктивних нормальних форм.

Задачі знаходження (побудови) мінімальних ДНФ називають задачами мінімізації.

Змінні $x_i\left(i = \overline{1,n}\right)$ та $\overline{x_i}\left(i = \overline{1,n}\right)$ досить часто називають **термами**. Повний набір із термів утворює **конституенту**. У процесі мінімізації деякі терми з конституант зникають. Ту частину, яка залишилася, називають **імплікантою**.

Булева функція $g\left(x_1, x_2, ..., x_n\right)$ називається **імплікантою** функції $f\left(x_1, x_2, ..., x_n\right)$, якщо вона перетворюється в одиницю під час набору змінних, на якому сама функція також дорівнює одиниці, тобто, якщо $g = 1$, то й $f = 1$.

Кожна конституанта одиниці, яка входить до складу ДДНФ, або їхня диз'юнкція є імплікантою певної булевої функції.

Елементарна кон’юнкція $k = x\bar{y}\bar{z}$ – імпліканта функції $f = x\bar{y}\bar{z} \vee \bar{x}\bar{y}z$, бо в разі k = 1 значення функції f дорівнює 1.

Імпліканта g називається простою, якщо жодна її частина не може бути імплікантою функції f.

Приклад 4.41. Для функції $f(x_1, x_2, x_3) = x_1x_2x_3 \vee x_1x_2\overline{x_3} \vee \overline{x_1}x_2x_3$ знайти всі імпліканти.

$$g_1 = x_1x_2x_3;$$

$$g_2 = x_1x_2\overline{x_3};$$

$$g_3 = \overline{x_1}x_2x_3;$$

$$g_4 = x_1x_2\overline{x_3} \vee x_1x_2x_3 = x_1x_2;$$

$$g_5 = \overline{x_1}x_2x_3 \vee x_1x_2x_3 = x_2x_3;$$

$$g_6 = x_1x_2\overline{x_3} \vee \overline{x_1}x_2x_3;$$

$$g_7 = x_1x_2x_3 \vee x_1x_2\overline{x_3} \vee \overline{x_1}x_2x_3 = f$$

x1	x2	x3	f	g1	g2	g3	g4	g5	g6	g7
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	1	0	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	1	0	1	1
1	1	1	1	1	0	0	1	1	0	1

ДНФ, що складається з усіх простих імплікант булевої функції, називають її **скороченою диз’юнктивною нормальною формою** (СДНФ).

Скорочена ДНФ (англ. *reduced disjunctive normal form*) – форма запису функції, що має такі властивості:

1) будь-які два доданки розрізняються мінімум як у двох позиціях,

2) жоден кон’юнкт не міститься в іншому.

За основу розв’язання задач мінімізації довільних логічних функцій приймаємо схему, що містить три етапи:

1) на першому етапі складається таблиця істинності функції $f(x_1, x_2, \ldots, x_n)$;

2) на другому етапі виконується пошук простих імплікант *f*, тобто будується **скорочена** досконала нормальна форма.

Наприклад, (*x*∧*y*) міститься в (*x*∧*y*∧*z*). Функцію можна записати за допомогою скороченої ДНФ не єдиним способом. Запишемо функцію (x, y, z) у вигляді доскональної ДНФ: (*x*∧*y*∧*z*) ∨ (*x*∧*y*∧¬z) ∨ (*x*∧¬*y*∧*z*) ∨ (¬*x*∧*y*∧*z*). Відомо, що цей вираз еквівалентний такому: ((*x*∧*y*∧*z*) ∨ (*x*∧*y*∧¬z)) ∨ ((*x*∧¬*y*∧*z*) ∨ (*x*∧*y*∧*z*)) ∨ ((¬*x*∧*y*∧*z*) ∨ (*x*∧*y*∧*z*)). Винесемо в кожній дужці спільний кон’юнкт (наприклад, у першій (*x*∧*y*∧*z*) ∨ (*x*∧*y*∧¬z) = (*x*∧*y*) ∨ (*z*∧¬z). Оскільки *z*∧¬z=0, то такий кон’юнкт не впливає на значення виразу і його можна опустити.

Отримаємо в підсумку формулу (*x*∧*y*) ∨ (*y*∧*z*) ∨ (*x*∧*z*);

3) на третьому етапі проводиться побудова **тупикових** (без зайвих імплікант) досконалих нормальних форм, із числа яких вибирають **мінімальні** досконалі нормальні форми.

ДНФ називають **тупиковою** ДНФ функції, якщо:

а) кожна елементарна кон’юнкція з ТДНФ – проста імпліканта *f*;

б) вилучення з ТДНФ довільного диз’юнктивного члена призводить до ДНФ, яка не задає функцію *f*.

Мінімальна ДНФ булевої функції являє собою її тупикову ДНФ.

Існують тупикові, але не мінімальні ДНФ, та сама булева функція *f* може мати декілька різних мінімальних ДНФ.

Приклад 4.42. Нехай скорочена ДНФ функції має вигляд: $f(x, y, z) = xy \vee \overline{y}z \vee xz$.

Тоді її одинична множина може бути представлена у вигляді: $M_f = M_{xy} \vee M_{\overline{y}z} \vee M_{xz} = \{110,111\} \vee \{001,101\} \vee \{101,111\}$. Зазначимо, що набори, що входять до останньої підмножини, перебувають також і в першому і в другому. Так, $101 \in M_{\overline{y}z}$ (говорять, ***набір 101 покривається множиною*** $M_{\overline{y}z}$), а $111 \in M_{xy}$. Отже, якщо прибрати складову M_{xz}, функція від цього не зміниться. Говорять, що ***множина покривається об’єднанням*** $M_{\overline{y}z}$ ***і*** M_{xy}. Отже, ***імпліканта xz – зайва.***

Для розв’язання задач мінімізації на другому етапі застосовують методи законів і тотожностей алгебри логіки, метод Куайна, метод Мак-Класкі, метод Карно – Вейча, метод Куайна – Мак-Класкі й інші. Основним методом для проведення третього етапу є метод імплікантної таблиці логічних функцій (метод Куайна).

Розглянемо ці методи.

Метод послідовного застосування законів і тотожностей алгебри логіки

В основі цього методу лежить пошук виразів, які можна записати у більш простому вигляді. При цьому зазвичай використовуються такі операції та закони логіки:

$x_1\overline{x_2} \vee x_1x_2 = x_1(\overline{x_2} \vee x_2) = x_1$ – операція склеювання;

$x_1 \vee x_x x_2 = x_1(1 \vee x_2) = x_1$ – операція поглинання;

$x_1 \vee \overline{x_1}x_2 = (x_1 \vee \overline{x_1})(x_1 \vee x_2) = x_1 \vee x_2$ – дистрибутивний закон;

$x \vee x = x$ – ідемпотентність, із цього закону випливає, що кожний доданок в ДНФ можна групувати з іншими неодноразово.

Методом користуються лише в досить простих випадках, він містить елементи довільних розв’язків і є дуже громіздким. Розглянемо приклад.

Приклад 4.43. Мінімізувати булеву функцію

$$\boldsymbol{f(x_1, x_2, x_3, x_4) = \overline{x_1}\overline{(x_2x_3 \vee x_4)(x_1x_2x_3 \vee x_4)}(\overline{x_1}x_2x_3x_4 \vee \overline{x_1}x_2x_3 \vee \overline{x_1}).}$$

Розв’язок. Скористаємося законом де Моргана та дистрибутивним законом.

$$f = \overline{x_1}(\overline{(x_2x_1 \vee x_4)} \vee \overline{(x_1x_2x_3 \vee x_4)})(\overline{x_1}x_2x_3x_4 \vee \overline{x_1}x_2x_3 \vee \overline{x_1}) =$$
$$= \overline{x_1}\big((x_2x_1 \vee x_4) \vee (x_1x_2x_3 \vee x_4)\big) \cdot$$
$$\cdot (x_2x_3x_4 \vee x_2x_3 \vee 1) = \overline{x_1}(x_2x_1 \vee x_4 \vee x_1x_2x_3 \vee x_4)(x_2x_3x_4 \vee x_2x_3 \vee 1)$$

Застосуємо дистрибутивний закон та закон поглинання.

$$f = \overline{x_1}(x_2x_3(1 \vee x_1) \vee x_4) = \overline{x_1}(x_2x_3 \vee x_4).$$

Метод Карно – Вейча

Якщо число змінних логічних функцій мале $(n \le 4)$, знаходження мінімальних форм можна проводити за допомогою спеціальних таблиць, які називаються діаграмами Вейча або картами Карно. Нехай n = 4, тобто

$f = f(x_1, x_2, x_3, x_4)$. Карта Карно для чотирьох змінних являє собою квадрат, що розбитий на 16 малих квадратів (4 x 4). Складемо карту Карно. Для змінних x_1 і x_2 відведемо вертикальну сторону карти, а для x_3 і x_4 – горизонтальну.

Сенс карти в тому, що функція задається таблицею, але не в стовпчиках, як завжди, а на площині у вигляді 16 квадратів. Набори змінних використовуються в порядку, так званого коду Грея (00), (01), (11), (10). На карті Карно сусідні набори відмінні лише однією координатою від сусіднього за розміщенням. Значення функції записують у малих квадратах. За таблицею Карно з'ясовують, які квадрати карти можна закріпити тією чи іншою імплікантою, якщо два сусідні рядки чи два сусідні стовпчики заповнені одиницями, то їх можна покрити однотермовою імплікантою, тобто якби в перших двох рядках всюди були одиниці, то: $f = \overline{x_1} \vee \phi(x_1, x_2, x_3, x_4)$.

Зауважимо, що карту Карно треба уявляти так, що вона відтворена не на площині, а на поверхні, що має форму Тора, у якому сусідніми будуть перший та останній рядки, перший та останній стовпчики. Тобто можна стверджувати, що однобуквеним імплікантам відповідають або два сусідні рядки, або два сусідні стовпчики.

Двотермові імпліканти розглядають так: $x_1 x_2 = 1$ відповідає третій рядок. Аналогічно знаходяться три термові й чотири термові імпліканти. Тобто за допомогою карти Карно графічно виконуються операції склеювання, поглинання, об'єднання одиниць і груп одиниць між собою. Продемонструємо метод Карно – Вейча на прикладі.

Приклад 4.44. Нехай функція $f(x_1, x_2, x_3, x_4)$ задана таблицею.

Розв'язок. Проаналізуємо цю таблицю істинності й позначимо квадрати, що вміщують одиниці літерами.

З таблиці видно, що однотермові імпліканти відсутні, бо немає двох сусідніх стовпчиків або рядків, що вміщують лише одиниці.

Будемо шукати двотермові імпліканти, тобто чотири квадрати з одиницями, що витягнуті в одну лінію або складені у великий квадрат. Бачимо, що перший рядок (квадрати *e, a, b, f*) – утворює лінію, що покривається $\overline{x_1}$ та $\overline{x_2}$, отже отримано імпліканту $\overline{x_1 x_2}$.

$\overline{x_3}$ x_3

x_3,x_4 / x_1,x_2	00	01	11	10
00	1 e	1 a	1 b	1 f
01	0	1 c	1 d	0
11	0	0	1 i	1 j
10	1 g	1 k	0	0

$\overline{x_1}$ x_1

$\overline{x_4}$ x_4 $\overline{x_4}$

Квадрат *abcd*, що складається з одиниць, повністю покривається імплікантою $\overline{x_1}x_4$.

Квадрат *eagk* покривається імплікантою $\overline{x_2x_3}$ (квадрат утворено замкненням карти до утворення тору). Більше квадратів утворити не можна, тобто

$$f(x_1,x_2,x_3,x_4) = \overline{x_1x_2} \vee \overline{x_2x_3} \vee \overline{x_1}x_4 \vee \phi(x_1,x_2,x_3,x_4).$$

У таблиці непокритими залишилися дві одиниці (квадрати *i* та *j*). Покрити їх неможливо, тобто потрібно витратити на них тритермову імпліканту $x_1x_2x_3$.

Усі одиниці покриті імплікантами, за методом Карно – Вейча МДНФ має вигляд:

$$f(x_1,x_2,x_3,x_4) = \overline{x_1x_2} \vee \overline{x_2x_3} \vee \overline{x_1}x_4 \vee x_1x_2x_3.$$

Приклад 4.45. Знайти мінімальну ДНФ для функції:

$$\boldsymbol{f(x_1,x_2,x_3) = x_1x_2\overline{x_3} \vee x_1\overline{x_2x_3} \vee \overline{x_1}x_2x_3 \vee \overline{x_1x_2x_3}}.$$

Розв'язок.

x2x3 / x1	00	01	11	10
0	1		1	
1	1			1

Мінімальна ДНФ: $f_{\min}(x_1, x_2, x_3) = x_1\overline{x_3} \vee \overline{x_2}\,\overline{x_3} \vee \overline{x_1}x_2x_3$.

Приклад 4.46. Знайти мінімальну ДНФ для функції:

$f(x_1, x_2, x_3) = x_1\overline{x_2x_3} \vee x_1\overline{x_2}x_3 \vee \overline{x_1}x_2x_3 \vee \overline{x_1x_2}x_3 \vee \overline{x_1x_2x_3}$

x2x3 x1	00	01	11	10
0	1	1	1	
1	1	1		

Мінімальна ДНФ: $f(x_1, x_2, x_3) = \overline{x_2} \vee \overline{x_1}x_3$.

Приклад 4.47. Знайти мінімальну ДНФ для функції: $f(x_1, x_2, x_3) =$
$= x_1x_2x_3 \vee x_1x_2\overline{x_3} \vee x_1\overline{x_2}x_3 \vee x_1\overline{x_2x_3} \vee \overline{x_1}x_2x_3 \vee \overline{x_1x_2}x_3 \vee \overline{x_1x_2x_3}$

Мінімальна ДНФ: $f(x_1, x_2, x_3) = \overline{x_2} \vee x_3 \vee x_1$.

x2x3 x1	00	01	11	10
0	1	1	1	
1	1	1	1	1

Приклад 4.48. Знайти мінімальну ДНФ для функції:

$f(x_1, x_2, x_3, x_4) = x_1x_2x_3\overline{x_4} \vee x_1x_2\overline{x_3x_4} \vee x_1\overline{x_2}x_3x_4 \vee x_1\overline{x_2}x_3\overline{x_4} \vee x_1\overline{x_2x_3x_4} \vee$
$\vee \overline{x_1}x_2x_3x_4 \vee \overline{x_1}x_2x_3\overline{x_4} \vee \overline{x_1}x_2\overline{x_3x_4} \vee \overline{x_1}x_2\overline{x_3}x_4 \vee \overline{x_1x_2}x_3\overline{x_4} \vee \overline{x_1x_2x_3x_4}$.

x3x4 x1x2	00	01	11	10
00	1			1
01	1	1	1	1
11	1			1
10	1		1	1

Мінімальна ДНФ: $f(x_1, x_2, x_3) = \overline{x_4} \vee \overline{x_1} x_2 \vee x_1 \overline{x_2} x_3$.

Метод Куайна

За методом Куайна прості імпліканти знаходяться за доскональною диз'юнктивною нормальною формою (ДДНФ) булевої функції внаслідок застосування до неї закону неповного склеювання й операції поглинання. Продемонструємо дію методу на прикладі.

Приклад 4. 49. Мінімізувати булеву функцію, що задана таблицею.

Склеюємо попарно ті конституенти, що відрізняються одним термом. У цьому випадку: 1 і 2 конституенти, потім 1 і 3, потім 2 і 4 і т. д. Після першого етапу склеювання повинні залишитися імпліканти, що містять вже 3 терми замість 4. На другому етапі знову проводимо попарно операцію склеювання так, щоб в імплікантах залишилося по 2 терми.

На четвертому етапі застосуємо операцію поглинання: $BC \vee C = C$. Тобто викреслюємо всі імпліканти, що містять $\overline{a}\overline{d}$. Записавши імпліканти, що залишилися невикресленими, отримаємо скорочену диз'юнктивну нормальну форму. СДНФ не є останнім степенем спрощення.

Складаємо таблицю Куайна, у якій помістимо отримані спрощенням імпліканти та вихідні конституенти. У стовпчиках записуємо конституенти, у рядках – отримані імпліканти.

Одиницю ставимо там, де імпліканта «покриває» конституенту, це тому, що конституента може бути замінена імплікантою за законом поглинання.

a	b	c	d	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Запишемо ДДНФ:

$\cancel{\bar{a}\bar{b}\bar{c}\bar{d}}$
$\cancel{\bar{a}\bar{b}c\bar{d}}$
$\cancel{\bar{a}b\bar{c}\bar{d}}$
$\cancel{\bar{a}bc\bar{d}}$
$\cancel{\bar{a}bcd}$
$a\bar{b}\bar{c}d$
$\cancel{ab\bar{c}\bar{d}}$
$\cancel{abcd}$

$Ax \cdot A\bar{x} = A$

Застосуємо операцію склеюваня конституент:

$\cancel{\bar{a}\bar{b}\bar{d}}$
$\cancel{\bar{a}\bar{c}\bar{d}}$
$\cancel{ac\bar{d}}$
$\cancel{\bar{a}b\bar{d}}$
$b\bar{c}\bar{d}$
$\bar{a}bc$
bcd

$\bar{a}\bar{d}$
$\cancel{\bar{a}\bar{d}}$

$\bar{a}\bar{d} \vee b\bar{c}\bar{d} \vee \bar{a}bc \vee bcd \vee a\bar{b}\bar{c}d$ - СДНФ

Шукаємо ті стовпчики, де є тільки одна одиниця. Це стовпчики 1, 2, 6, 7 і 8. Ці стовпчики відповідають імплікантам $a\bar{b}\bar{c}d$, $b\bar{c}\bar{d}$, bcd і $\bar{a}\bar{d}$. Ці імліканти складають **ядро**, і їх вилучати в жодному разі не можна. Тепер подивимось, чи можемо ми виключити імпліканту. Одиниця, що стоїть у четвертому стовпчику навпроти імпліканти $\bar{a}bc$, покривається одиницею, що стоїть в тому ж стовпчику навпроти імпліканти $\bar{a}\bar{d}$, а одиницю, що стоїть в п'ятому стовпчику навпроти $\bar{a}bc$, покриває одиниця, що стоїть у тому самому стовпчику навпроти bcd.

	$\bar{a}\bar{b}\bar{c}\bar{d}$	$\bar{a}\bar{b}c\bar{d}$	$\bar{a}b\bar{c}\bar{d}$	$\bar{a}bc\bar{d}$	$\bar{a}bcd$	$a\bar{b}\bar{c}d$	$ab\bar{c}\bar{d}$	$abcd$
$a\bar{b}\bar{c}d$						1		
$b\bar{c}\bar{d}$			1				1	
$\cancel{\bar{a}bc}$				~~1~~	~~1~~			
bcd					1			1
$\bar{a}\bar{d}$	1	1	1	1				

Отже обидві ці одиниці покриваються іншими одиницями. Робимо висновок, імпліканту $\bar{a}bc$ можна виключити. Отже, мінімальна ДНФ має такий вигляд: МДНФ $= a\bar{b}\bar{c}d \vee b\bar{c}\bar{d} \vee bcd \vee \bar{a}\bar{d}$. Ця МДНФ складається лише з імплікант ядра, але це не завжди так. Бувають випадки, коли спочатку з таблиці Куайна отримуємо **тупикові** ДНФ, з яких потім вибираємо мінімальну ДНФ. Тупикова ДНФ складається з ядра і деяких імплікант.

Приклад 4.50. Мінімізувати булеву функцію

$$f(x_1,x_2,x_3,x_4)=x_1x_2x_3\overline{x_4}\vee x_1x_2\overline{x_3}\,\overline{x_4}\vee x_1\overline{x_2}x_3x_4\vee x_1\overline{x_2}x_3\overline{x_4}\vee x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}\vee \overline{x_1}x_2x_3x_4\vee$$
$$\vee\overline{x_1}x_2x_3\overline{x_4}\vee\overline{x_1}x_2\overline{x_4}\,\overline{x_4}\vee\overline{x_1}x_2\overline{x_3}x_4\vee\overline{x_1}\,\overline{x_2}x_3\overline{x_4}\vee\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,.$$

Застосуємо закон склеювання конституент

1) $x_1x_2x_3\overline{x_4}$ - 1110 1) та 2) 1110 $\vee$ 1100 = 11_0

2) $x_1x_2\overline{x_3}\,\overline{x_4}$ - 1100 1) та 4) 1110 $\vee$ 1010 = 1_101

1) та 7) 1110 $\vee$ 0110 = _110

3) $x_1\overline{x_2}x_3x_4$ - 1011 2) та 8) 1100 $\vee$ 0100 = _100

4) $x_1\overline{x_2}x_3\overline{x_4}$ - 1010 2) та 5) 1100 $\vee$ 1000 =1_00

3) та 4) 1011 $\vee$ 1010 = 101_

5) $x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}$ - 1000 4) та 5) 1010 $\vee$ 1000 =10_0

6) $\overline{x_1}x_2x_3x_4$ - 0111 4) та 10) 1010 $\vee$ 0010 = _010

7) $\overline{x_1}x_2x_3\overline{x_4}$ - 0110 5) та 11) 1000 $\vee$ 0000 =_000

6) та 7) 0111 $\vee$ 0110 = 011_

8) $\overline{x_1}x_2\overline{x_3}\,\overline{x_4}$ - 0100 6) та 9) 0111 $\vee$ 0101 =01_1

9) $\overline{x_1}x_2\overline{x_3}x_4$ - 0101 7) та 8) 0110 $\vee$ 0100 = 01_0

7) та 10) 0110 $\vee$ 0010 =0_10

10) $\overline{x_1}x_2\overline{x_3}\,\overline{x_4}$ - 0100 8) та 9) 0100 $\vee$ 0101 = 010_

11) $\overline{x_1}x_2\overline{x_3}\,\overline{x_4}$ - 0100 8) та 11) 0100 $\vee$ 0000 =0_00

10) та 11)0010 $\vee$ 0000 = 00_0

12) 11_0 Застосуємо закон склеювання імплікант, враховуючи
13) 1_10 позицію прочерку
14) _110 12) та 18) 11_0 $\vee$ 10_0 = 1__0
15) 1_00 13) та 16) 1_10 $\vee$ 1_00 = 1__0
16) _100 12) та 23) 11_0 $\vee$ 01_0 = _1_0
17) 101_ 13) та 24) 1_10 $\vee$ 0_10 = __10
18) 10_0 14) та 15) _110 $\vee$ _100 = _1_0
19) _010 14) та 19) _110 $\vee$ _010 = __10
20) _000 15) та 20) _100 $\vee$ _000 = __00

21) 011_ 16) та 26) 1_00 ∨ 0_00 = __00
22) 01_1 18) та 27) 10_0 ∨ 00_0 = _0_0
23) 01_0 19) та 20) _010 ∨ _000 = _0_0
24) 0_10 21) та 25) 011_ ∨ 010_ = 01__
25) 010_ 22) та 23) 01_1 ∨ 01_0 = 01__
26) 0_00 23) та 27) 01_0 ∨ 00_0 = 0__0
27) 00_0 24) та 26) 0_10 ∨ 0_00 = 0__0

1__0		
_1_0		Застосуємо закон склеювання
1__0	Видаляємо імпліканти, що повторюються	імплікант, враховуючи позицію прочерку
__10		
_1_0		
__10		
__00	1__0	1__0 ∨ 0__0 = ___0
__00	_1_0	_1_0 ∨ _0_0 = ___0
_0_0	__10	__10 ∨ __00 = ___0
_0_0	__00	
01__	_0_0	Видаляємо імпліканти, що повторюються
01__	01__	
0__0	0__0	___0
0__0		

Мінімальна ДНФ формується з імплікант, які жодного разу не склеїлися (виділені червоним кольором):

101_ 01__ ___0

$$f_{\min} = x_1 \overline{x_2} x_3 \vee \overline{x_1} x_2 \vee \overline{x_4}$$

Метод Мак-Класкі

Метод застосовують тоді, коли булева функція задана нормальною формулою.

Алгоритм методу містить такі етапи:

1. Кожній конституанті присвоюється індекс – число одиниць термів і номер – відповідне число в десятковій системі числення.

2. Отримані результати заносяться в таблицю, у першому рядку якої записують індекси, а в другому – номери конституент.

Індекс (число одиниць термів)	0	1	2	3	4
Номер (відповідне число в двійковій системі)					

3. Виконується склеювання за правилом: нехай i – індекс, j – індекс, $j > i$.

Склеюються ті конституанти, різниця між m_i та n_j є степінь двійки:

$n_j - m_i = 2^n$; n=0; 1; 2; 4; …

За склеювання справа вказується величина різниці. Склеювання відбувається доти, доки воно можливе (різниці степені двійки). Склеювати можна лише сусідні індекси.

X1	X2	X3	X4	Десяткове число
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Приклад 4.51. Мінімізувати булеву функцію $f(x_1, x_2, x_3, x_4) = x_1x_2x_3\overline{x_4} \vee x_1x_2\overline{x_3}\,\overline{x_4} \vee x_1\overline{x_2}x_3x_4 \vee x_1\overline{x_2}x_3\overline{x_4} \vee x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4} \vee \overline{x_1}x_2x_3x_4 \vee$
$\vee \overline{x_1}x_2x_3\overline{x_4} \vee \overline{x_1}x_2\overline{x_4}\,\overline{x_4} \vee \overline{x_1}x_2\overline{x_3}x_4 \vee \overline{x_1}\,\overline{x_2}x_3\overline{x_4} \vee \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}$.

1) $x_1x_2x_3\overline{x_4} - (1110)_3 - 14$

2) $x_1 x_2 \overline{x_3}\,\overline{x_4} - (1100)_2 - 12$

3) $x_1 \overline{x_2} x_3 x_4 - (1011)_3 - 11$

4) $x_1 \overline{x_2} x_3 \overline{x_4} - (1010)_2 - 10$

5) $x_1 \overline{x_2}\,\overline{x_3}\,\overline{x_4} - (1000)_1 - 8$

6) $\overline{x_1} x_2 x_3 x_4 - (0111)_3 - 7$

7) $\overline{x_1} x_2 x_3 \overline{x_4} - (0110)_2 - 6$

8) $\overline{x_1} x_2 \overline{x_4}\,\overline{x_4} - (0100)_1 - 4$

9) $\overline{x_1} x_2 \overline{x_3} x_4 - (0101)_2 - 5$

10) $\overline{x_1}\,\overline{x_2} x_3 \overline{x_4} - (0010)_1 - 2$

11) $\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4} - (0000)_0 - 0$

12)

Індекс (число одиниць термів)	0	1	2	3	4
Номер (відповідне число в двійковій системі)	0	2 4 8	5 6 10 12	7 11 14	

I

(індекси 1 та 0)

(0, 2) (2)

(0, 4) (4)

(0, 8) (8)

II

(індекси 2 та 1)

(2, 6) (4)

(4, 6) (2)

(2, 10) (8)

(4, 12) (8)

(8, 12) (4)

(4, 5) (1)

(8, 10) (2)

III

(індекси 3 та 2)

(5, 7) (2)

(10, 11) (1)

(10, 14) (4)

(6, 14) (8)

(12, 14) (2)

(6, 7) (1)

IV

(етапи I та II)

(0, 2, 4, 6) (2, 4)

(0, 2, 8, 10) (2, 8)

~~(0, 4, 2, 6) (2, 4)~~

(0, 4, 8, 12) (4, 8)

V

(етапи III та II)

(2, 6, 10, 14) (4, 8)

(8, 10, 12, 14) (2, 4)

(4, 5, 6, 7) (1, 2)

(4, 6, 12, 14) (2, 8)

~~(0, 4, 8, 12) (4, 8)~~ ~~(2, 6, 10, 14) (4, 8)~~
~~(0, 2, 8, 10) (2, 8)~~ ~~(4, 6, 12, 14) (2, 8)~~
~~(4, 5, 6, 7) (1, 2)~~

VI

(етапи IV та V)

(0, 2, 4, 6, 7, 10, 12, 14) (2, 4, 8)
~~(0, 4, 8, 12, 2, 6, 10, 14) (2, 4, 8)~~
~~(0, 4, 8, 12, 2, 6, 10, 14) (2, 4, 8)~~

(10, 11) (1)

$x_1\overline{x_2}x_3x_4$ - $(1011)_3$ - 11 $\qquad$ $x_1\overline{x_2}x_3$ (101_)

$x_1\overline{x_2}x_3\overline{x_4}$ - $(1010)_2$ - 10

(4, 5, 6, 7) (1, 2)

$\overline{x_1}x_2x_3x_4$ - $(0111)_3$ - 7 $\qquad$ $\overline{x_1}x_2$ (01_ _)

$\overline{x_1}x_2x_3\overline{x_4}$ - $(0110)_2$ - 6

$\overline{x_1}x_2\overline{x_3x_4}$- $(0100)_1$ - 4

$\overline{x_1}x_2\overline{x_3}x_4$- $(0101)_2$ - 5

(0, 2, 4, 6, 8, 10, 12, 14) (2, 4, 8)

$x_1x_2x_3\overline{x_4}$ - $(1110)_3$- 14 $\qquad$ $\overline{x_4}$ (_ _ _0)

$x_1x_2\overline{x_3x_4}$ - $(1100)_2$- 12

$x_1\overline{x_2}x_3\overline{x_4}$ - $(1010)_2$ - 10

$x_1\overline{x_2x_3x_4}$ - $(1000)_1$ - 8

$\overline{x_1}x_2x_3\overline{x_4}$ -$(0110)_2$ - 6

$\overline{x_1}x_2\overline{x_3x_4}$ - $(0100)_1$ - 4

$\overline{x_1x_2}x_3\overline{x_4}$ - $(0010)_1$ - 2

$\overline{x_1x_2x_3x_4}$- $(0000)_0$ - 0

Після склеювання отримуємо скорочену диз'юнктивну нормальну форму (СДНФ)

$$f_{скор} = x_1\overline{x_2}x_3 \vee \overline{x_1}x_2 \vee \overline{x_4}.$$

Для отримання мінімальної ДНФ складемо таблицю Куайна, у якій помістимо отримані спрощенням імпліканти та вихідні конституанти. Одиницю ставимо там, де одиниця «покриває» конституанту, це тому, що конституанта може бути замінена імплікантою за законом поглинання.

x1	x2	x3	x4	1110	1100	1011	1010	1000	0111	0110	0100	0101	0010	0000
_	_	_	0	1	1		1	1		1	1		1	1
1	0	1	_			1	1							
0	1	_	_						1	1	1	1		

Отже, мінімальна ДНФ:

$$f1\overline{x_2}_3\overline{x_1}_2\overline{x_4}_{min}.$$

Приклад 4.52. Методом Мак-Класкі мінімізувати булеву функцію, нормальна форма якої: $f = x_4x_3x_2\overline{x_1} \vee x_4x_3\overline{x_2}x_1 \vee x_4\overline{x_3}x_2x_1 \vee \overline{x_4}x_3x_2x_1 \vee x_4x_3\overline{x_2x_1} \vee x_4\overline{x_3x_2}x_1 \vee \overline{x_4x_3}x_2x_1 \vee x_4x_3x_2x_1 \vee$

$$\vee \overline{x_1x_2x_3x_4}$$

Розв'язок. Запишемо всі конституенти через індекси та присвоєні номери (десяткове числення).

$x_4x_3x_2\overline{x_1}$- $(1110)_3$ – 14

$x_4x_3\overline{x_2}x_1$ - $(1101)_3$ – 13

$x_4\overline{x_3}x_2x_1$ - $(1011)_3$ – 11

$\overline{x_4}x_3x_2x_1$ - $(0111)_3$ – 7

$x_4x_3\overline{x_2x_1}$- $(1100)_2$ – 12

$x_4\overline{x_3x_2}x_1$ - $(1001)_2$ – 9

$\overline{x_4x_3}x_2x_1$ - $(0011)_2$ – 3

$x_4x_3x_2x_1$ - $(1111)_4$ – 15

$\overline{x_4x_3x_2x_1}$- $(0000)_0$ – 0

Отримані результати занесемо в таблицю:

Індекс	0	1	2	3	4
Номер	0^*	-	3;9;12	7;11;13;14	15

Враховуючи, що склеювати можна лише сусідні індекси, бачимо:

1) конституенту з індексом 0 склеїти з жодною не можна, бо немає конституент з індексом 1;

2) процес склеювання ($j > i$; $n_j - m_i = 2^n$; n = 0; 1; 2; 4; ... n_{ij}) 1 група: (3; 7) – (4); (3; 11) – (8); (9; 11) – (2); (9; 13) – (4); (12; 13) – (1); (12; 14) – (2) (склеювання конституент з індексом 2 і 3),

2 група: (7; 15) – (8); (11; 15) – (4); (13; 15) – (2); (14; 15) – (1) (склеювання конституент з індексом 3 і 4). Знову проведемо склеювання між 1 і 2 групами, у яких різниці однакові;

$j > i$: (12; 13; 14; 15) – (1; 2); (9; 11; 13; 15) – (2; 4); (12; 13; 14; 15) – (1; 2); (3; 7; 11; 15) – (4; 8); (9; 11; 13; 15) – (2; 4); (3; 7; 11; 15) – (4; 8); (12; 13; 14; 15) – (1; 2).

Процес склеювання закінчено. Отримані результати запишемо у вигляді конституент (ліва дужка), права дужка вказує на терми, які треба виключити з конституент, що залишилися.

(12; 13; 14; 15) – (1; 2)

$$\left.\begin{array}{l} 12 = 1100 - x_4x_3\overline{x_2x_1} = x_4x_3 \\ 13 = 1101 - x_4x_3\overline{x_2}x_1 = x_4x_3 \\ 14 = 1110 - x_4x_3x_2\overline{x_1} = x_4x_3 \\ 15 = 1111 - x_4x_3x_2x_1 = x_4x_3 \end{array}\right\} x_4x_3$$

(9; 11; 13; 15) – (2; 4)

$$\left.\begin{array}{l} 9 = 1001 - x_4\overline{x_3x_2}x_1 = x_4x_1 \\ 11 = 1011 - x_4\overline{x_3}x_2x_1 = x_4x_1 \\ 13 = 1101 - x_4x_3\overline{x_2}x_1 = x_4x_1 \\ 15 = 1111 - x_4x_3x_2x_1 = x_4x_1 \end{array}\right\} x_4x_1$$

(3; 7; 11; 15) – (4; 8)

$$\left.\begin{array}{l} 3 = 0011 - \overline{x_4x_3}x_2x_1 = x_2x_1 \\ 7 = 1011 - \overline{x_4}x_3x_2x_1 = x_2x_1 \\ 11 = 1011 - x_4\overline{x_3}x_2x_1 = x_2x_1 \\ 15 = 1111 - x_4x_3x_2x_1 = x_2x_1 \end{array}\right\} x_2x_1$$

Таким чином, мінімальна нормальна форма, отримана за методом Мак-Класкі для функції має вигляд: $f_{\min} = \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4} \vee x_4x_3 \vee x_4x_1 \vee x_2x_1$.

Обов'язкові завдання

1. За наведеною таблицею істинності знайти логічну функцію та спростити її.

x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

2. Мінімізувати булеву функцію:

$$f(x_1, x_2, x_3, x_4) = \overline{x_4 x_3} x_2 \overline{x_1} \vee \overline{x_4} x_3 \overline{x_2} x_1 \vee \overline{x_4} x_3 x_2 \overline{x_1} \vee \overline{x_4} x_3 x_2 x_1 \vee$$
$$\vee x_4 \overline{x_3} x_2 \overline{x_1} \vee x_4 x_3 \overline{x_2 x_1} \vee x_4 x_3 \overline{x_2} x_1 \vee$$
$$\vee x_4 x_3 x_2 \overline{x_1} = f(0010; 0101; 0110; 0111; 1010; 1100; 1101; 1110)3$$

3. Мінімізувати булеву функцію: $f = x\overline{y}z\overline{w} \vee x\overline{y}\overline{zw} \vee xy\overline{zw} \vee \overline{x}y\overline{zw} \vee \overline{xy}zw \vee \overline{x}yzw \vee xyz\overline{w}$ методом Карно – Вейча.

4. Мінімізувати функцію, що задана таблицею, методом Мак-Класкі:

x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Список літератури

1. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Дискретна математика : підручник. – Львів : Магнолія, 2006, 2010. – 431 с.
2. Міхайленко В. М., Федоренко Н. Д., Демченко В. В. Дискретна математика : підручник. Київ : Вид-во Європ. ун-ту, 2003, 318 с.
3. Бардачов Ю. М., Соколова Н. А., Ходаков В. Є. Дискретна математика : підручник для студ. вищ. техн. закл. / Ю. М. Бардачов [та ін.]; за ред. В. Є. Ходакова. 2-ге вид., перероб. і доп. – Київ : Вища школа, 2007. – 383 с.
4. Бондаренко М. Ф., Білоус Н. В., Руткас А. Г. Комп'ютерна дискретна математика : підручник для вищ. навч. закл. / М. Ф. Бондаренко [та інш.]. – Харків : Компанія СМІТ, 2004. – 479 с.
5. Федоренко Н. Д., Білощицька С.В., Демченко В.В., Баліна О.І. Задачі з теорії множин, теорії графів та комбінаторики : навч. посібник для студ. вищ. навч. закл. / Київськ. нац. ун-т буд-ва і архіт. – Київ : КНУБА, 2004. – 103 с.
6. Федоренко Н. Д., Білощицька С. В., Білощицький А. О., Баліна І. О., Безклубенко І. С., Буценко Ю. П. Дискретна математика : навч. посібник для студ. інженерно-техніч. спец. вищ. навч. закладів: у 2 ч. / Київ. нац. ун-т буд-ва і архітектури. Ч. 1. – Київ : КНУБА, 2014. – 103 с.
7. Трохимчук Р. М. Дискретна математика : навч. посібник для студ. вищ. навч. закл. – Київ : ДП «Видавничий дім «Персонал», 2010. 528 с.
8. Матвієнко М. П. Дискретна математика XXI століття : навч. посібник для студ. вищ. навч. закл. / М. П. Матвієнко; Мін-во освіти і науки, молоді та спорту України. – Київ : Ліра-К, 2013. – 347 с.

Навчальне видання

ТЕРЕНТЬЄВ Олександр Олександрович,
СЕРПІНСЬКА Ольга Ігорівна,
БАЛІНА Олена Іванівна та ін.

ДИСКРЕТНА МАТЕМАТИКА

Навчальний посібник

Редагування та коректура Т. В. Івченко
Комп'ютерне верстання Л. В. Лабунець

Ум. друк. арк. 20,11. Обл.-вид. акр. 21,63.
Електронний документ. Вид. № 36/V-24

Видавець і виготовлювач
Київський національний університет будівництва і архітектури

Повітрофлотський проспект, 31, Київ, 03037

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002