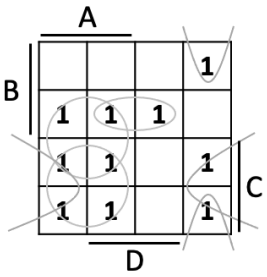
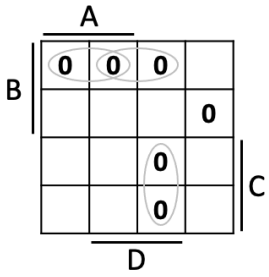
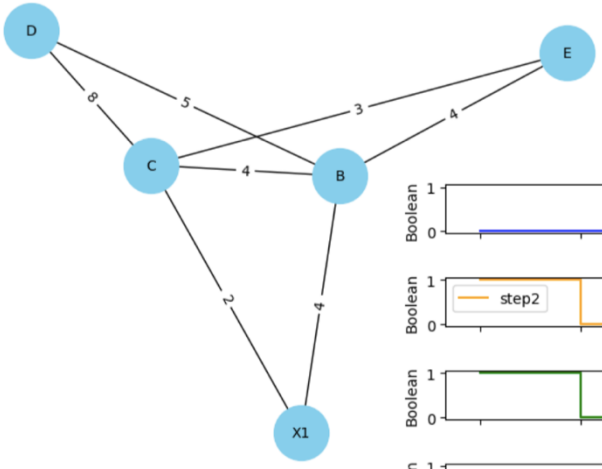
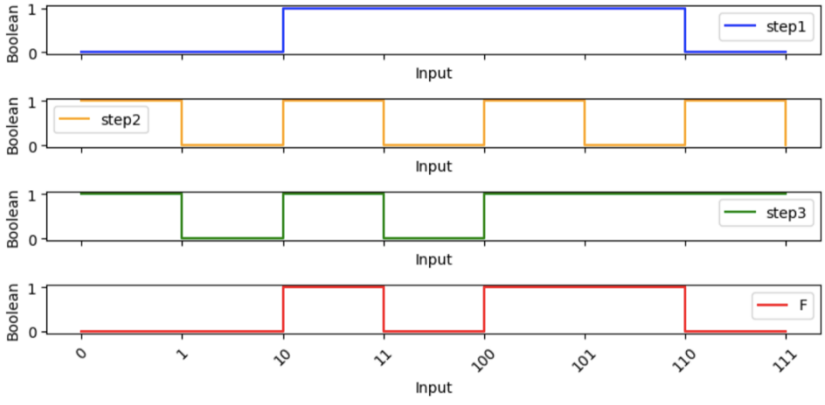
О. К. КОЛЕСНИЦЬКИЙ, О. Ф. ШЕВЧУК, Т. Г. КИРИЛАЩУК

ДИСКРЕТНА МАТЕМАТИКА
Частина 1









F = (x1 ⊕ x2) ∧ (x1 ∨ ¬x3) = [0 0 1 0 1 1 0 0]

Міністерство освіти і науки України
Вінницький національний технічний університет

О. К. КОЛЕСНИЦЬКИЙ, О. Ф. ШЕВЧУК, Т. Г. КИРИЛАЩУК

ДИСКРЕТНА МАТЕМАТИКА
Частина 1

Електронний навчальний посібник

Вінниця
ВНТУ
2025

УДК 519.1
К60

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 8 від 30.01.2025 р.)

Рецензенти:
Н. І. Заболотна, доктор технічних наук, професор
Т. Б. Мартинюк, доктор технічних наук, професор
О. В. Зелінська, кандидат технічних наук, доцент

Колесницький, О. К.

К60 Дискретна математика. Частина 1 : навчальний посібник [Електронний ресурс] / Колесницький О. К., Шевчук О. Ф., Кирилащук Т. Г. − Вінниця : ВНТУ, 2025. − (PDF, 120 с.)

ISBN 978-617-8163-37-2 (PDF)

Навчальний посібник створено на основі матеріалів першої частини курсу «Дискретна математика» для здобувачів першого (освітньо-наукового) рівня вищої освіти за спеціальністю 122 «Комп'ютерні науки». У посібнику викладено основні теоретичні положення з теорії множин, комбінаторики та алгебри висловлень. До кожної теми додаються індивідуальні практичні завдання з прикладами їхнього розв'язання.

Окремий розділ присвячено лабораторному практикуму, який містить методичні рекомендації щодо виконання завдань із використанням мови програмування Python.

УДК 519.1

ISBN 978-617-8163-37-2 (PDF)

ЗМІСТ

ВСТУП ДО КУРСУ ДИСКРЕТНОЇ МАТЕМАТИКИ

Ключовим в назві курсу «Дискретна математика» є термін «дискретність», що походить від англійського *discrete* – роздільний (окремий). Дискретність – антипод неперервності. Нас оточують як дискретні, так і неперервні величини. Наприклад, кількість здобувачів вищої освіти у групі – це дискретна величина, а освітленість на робочому столі – неперервна величина, причому як за амплітудою, так і за часом (аналогова величина). Але коли ми хочемо передати інформацію про неперервну величину іншим людям, або ввести в комп'ютер, то ми маємо її виміряти і подати якимось числом, тобто замінити неперервну величину дискретною.

Дискретна математика – це наукова дисципліна, яка вивчає та досліджує математичні моделі об'єктів навколишнього світу, які мають властивості скінченних множин і описуються дискретними величинами.

Основні розділи дискретної математики – теорія множин, алгебраїчні структури, комбінаторика, математична логіка, теорія графів, теорія автоматів, теорія алгоритмів.

Особливого розвитку дискретна математика набула за останні 60–70 років в зв'язку з бурхливими дослідженнями в галузі комп'ютерної техніки

На сьогодні актуальність дискретної математики визначається такими дуже важливими факторами:

1) розвитком комп'ютерної техніки і комп'ютерних наук, що базуються, а, власне кажучи, є продовженням дискретної математики (вся інформація в комп'ютерах подається та обробляється в дискретному вигляді);

2) запитами різних прикладних наук – теорії керування, економіки, оптимізації і багатьох, багатьох інших;

3) логікою внутрішнього розвитку дискретної математики, тобто появою нових розділів, глибоких цікавих проблем, розвитком методів їхнього розв'язання.

Тема 1 ТЕОРІЯ МНОЖИН

1.1 Множина. Способи задання множин

Теорія скінченних множин – зручний засіб формування та запису основних властивостей та об'єктів дискретної математики.

Означення:	***Множиною** називається сукупність об'єктів, які розглядаються спільно.*

Означення:	***Об'єкти множини*** називаються елементами.

У процесі задання множини елементи не повторюються, порядок перерахування елементів значення не має.

Означення:	***Потужністю множини** називають кількість елементів, які вона містить.*

Позначають множини великими літерами латинського алфавіту *A, B, C, ..., Z*. Елементи множин позначають маленькими латинськими літерами, часто з індексами $a_i, b_i, c_i, \dots, z_i$. Наприклад, множина *X* потужністю *n* складається з елементів $x_1, x_2, \dots, x_n$. Записується це так: $X = \{x_1, x_2, \dots, x_n\}$.

Індекси дозволяють розрізняти елементи множини. Для позначення належності елемента x_i до множини *X* використовують символ належності $\in$: $x_i \in X$ (читають: елемент x_i належить множині *X*). В протилежному випадку позначають $x_j \notin X$ (читають: елемент x_j не належить множині *X*).

Далі будемо використовувати такі загальноприйняті позначення основних числових множин.

N – множина натуральних чисел, $N = \{1, 2, 3, \dots\}$.

Z – множина цілих чисел, $N = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$.

Q – множина раціональних чисел. Будь-яке раціональне число можна зобразити у вигляді дробу: *a/b,* де *a, b* $\in$ *Z, b* $\neq$ *0*.

R – множина дійсних чисел. Будь-яке дійсне число можна зобразити у вигляді нескінченного десяткового дробу $a, b_1 b_2 b_n \dots$ із цілою частиною

$a \in Z$ *і* $b_k \in \{0, ..., 9\}$. Множині дійсних чисел відповідає множина точок на числовій прямій.

Елементами множин можуть бути інші множини, тоді ці елементи позначатимуться великими літерами.

Приклад. $A = \{D, C\}$, $D = \{a, b\}$, $C = \{c, d, e\}$. При цьому $D \in A$, $C \in A$, але $a \notin A$ і $c \notin A$.

Означення:	Множина називається ***скінченною***, якщо вона містить скінченне число елементів, і ***нескінченною***, якщо вона містить необмежене число елементів.

Приклад. Множина $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$ цифр в десятковій системі числення скінченна, а множина точок кола нескінченна.

Означення:	***Упорядкованою*** вважається така множина, в якій важливі не тільки її елементи, але і порядок їх наступності у множині. Наприклад, упорядкованою є множина, в якій кожний елемент має свій порядковий номер. Позначають упорядковану множину, як правило, або круглими, або трикутними дужками.

Наприклад, $A = < 1, 2, 3 >$

$A = < a_1, a_2, \dots, a_n > \ n \in N;$

$B = (a, b, c)$.

Способи задання множин:

1) Перерахуванням

Перераховуються всі елементи множини і вони розташовуються у фігурних дужках. Наприклад: $A = \{a, b, c\}$, $B = \{1, 2, 3, 4\}$, $E = \{0, 1\}$.

Спосіб задання множини переліком її елементів не придатний для задання нескінченних множин, а у випадку скінченних множин його часто практично не можна реалізувати. Так, не можна перелічити множину риб у Тихому океані, хоча їхнє число скінченне.

2) Правилом $P(x)$ (або ***визначеною властивістю***)*:*

$X = \{x | P(x)\}$, де $P(x)$ означає, що елемент x має властивість $P(x)$.

Приклад. Множину N_{10} всіх натуральних чисел, менших за 10, можна задати так: $N_{10} = \{x | x \in N, x < 10\}$.

Властивості елементів можуть бути задані не формально, а за допомогою опису природною мовою.

Приклад. Множина *S* студентів групи 1КН, які одержують стипендію.

3) Рекурсією, тобто вказанням способу послідовного породження елементів множини.

Приклад. Множина значень рекурсивної функції є рекурсивно заданою множиною $F = \{\varphi_1, \varphi_2, \varphi_3, \dots, \}$, де $\varphi_i \in N$, $i = 1, 2, 3, \dots$.

Нехай $\varphi_1 = 1$, $\varphi_2 = 2$, а кожне наступне число залежить від двох попередніх так:

$$\varphi_n = 3\varphi_{n-2} + \varphi_{n-1}, \qquad n = 3, 4, \dots,$$

Тоді

$$\varphi_3 = 3\varphi_1 + \varphi_2 = 3 \cdot 1 + 2 = 5,$$
$$\varphi_4 = 3\varphi_2 + \varphi_3 = 3 \cdot 2 + 5 = 11 \text{ і т. д.}$$

У процесі задання множин можуть виникати помилки та протиріччя. Множина задана коректно, якщо для будь-якого елемента можна визначити, належить він множині чи ні.

Приклад. Визначення множини *A* як множини, що містить будь-які п’ять натуральних чисел, не є коректним, оскільки неможливо визначити точно елементи *A*. Множина всіх простих чисел визначена коректно. Для будь-якого натурального числа можна перевірити, чи є воно простим, хоча практично на це може знадобитися дуже багато часу.

Некоректність задання множини часто пов’язана з ***протиріччям*** що виникає у разі перевірки належності деякого елемента множині. Наведемо один із класичних прикладів.

Приклад. Єдиний перукар у місті *N* визначає множину *K* мешканців, яких він має голити, як сукупність всіх тих мешканців *N,* які не голяться самі. Але тоді для самого перукаря виходить протиріччя і, вносячи його до множини *K,* і відносячи його до мешканців, які голяться самі.

Такі протиріччя називаються ***логічними парадоксами*** і вивчаються вони в математичній логіці.

Ще деякі позначення:
$\Leftrightarrow$ – тоді і тільки тоді,
$\exists\, x$ – існує x такий, що,
$\forall\, x$ – для будь-якого x,
$\Rightarrow$ – слідує (випливає).

1.2 Основні поняття теорії множин

Означення:	Дві множини ***рівні***, якщо вони складаються з одних і тих самих елементів (з точністю до порядку). Позначається $A = B$. Якщо множини не рівні, це позначається $A \neq B$. Число елементів скінченної множини A позначимо через $\lvert A \rvert$.

Для множин A і B з нескінченним або великим числом елементів перевірка збігу наборів всіх елементів може бути важкою. Більш ефективною виявляється логічна перевірка ***двостороннього включення***. А саме: $A = B$ тоді і тільки тоді, коли з $x \in A$ випливає що $x \in B$ *і з* $y \in B$ випливає, що $y \in A$.

Означення:	Множина A, всі елементи якої є одночасно елементами множини B, називається ***підмножиною*** множини B.

Позначення. ***Нестроге включення*** позначається $A \subseteq B$, означає, що A – ***підмножина*** множини B, що, можливо, збігається з B. ***Строге включення*** позначається $A \subset B$ і означає, що A – підмножина множини B, що не збігається з B. Символьний вираз $A \subset B$ читають *«A* включено до B».

Виконання співвідношень $A \subseteq B$ і $B \subseteq A$ можливе тільки коли $A = B$. І навпаки, $A = B$, якщо $A \subseteq B$ *і* $B \subseteq A$ одночасно. Зауважимо, що іноді в літературі символом $\subset$ позначають «нестроге» включення, що допускає і рівність множин. У цьому випадку символ $\subseteq$ не використовується, а строге включення записують двома співвідношеннями $A \subset B$, $A = B$.

Означення:	***Універсальною*** називається множина, яка містить всі можливі елементи, що зустрічаються в цій задачі. Універсальна множина позначається символом U.

Зауважимо, що універсальна множина U може бути індивідуальною для кожної окремої задачі та визначатися в її умові.

Приклад. Розглянемо деяку групу здобувачів вищої освіти. Нехай A – множина юнаків групи, B – множина відмінників. У цій задачі універсальною є множина здобувачів вищої освіти групи, а множини A і B є її підмножинами: $A \subseteq U$, $B \subseteq U$.

Означення:	***Порожньою*** називається така множина, яка не містить ніяких елементів. Така множина позначається символом $\emptyset$.

Роль порожньої множини $\emptyset$ аналогічна ролі числа нуль. Це поняття можна використовувати для визначення насправді неіснуючої сукупності елементів (наприклад, множини зелених котів). Порожня множина $\emptyset$ є підмножиною будь-якої множини A, $\emptyset \subseteq A$. Слід пам'ятати, що порожня множина є множиною, тому якщо деяка множина A не містить жодного елемента, то $A = \emptyset$; $|A| = 0$. Запис $A = \{\emptyset\}$ означає, що A містить один елемент – $\emptyset$, $|A = 1|$.

Таким чином, будь-яка непорожня множина A обов'язково має, як мінімум, дві підмножини – порожню множину і саму цю множину.

Означення:	Множину всіх підмножин множини X називають ***множиною-степенем***, або ***булеаном*** множини X, і позначають 2^X.

Приклад. Нехай задана множина $A = \{a, b, c\}$. Система всіх її підмножин є

$$2^A = \big\{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\big\},$$

так що 2^A містить 8 елементів.

Порожня множина має тільки одну підмножину – саму порожню множину, тому $2^\emptyset = \{\emptyset\}$. Для довільної множини X з n елементів кількість всіх її підмножин (тобто $|2^X|$) дорівнює 2^n:

$$|2^X| = 2^{|x|} = 2^n.$$

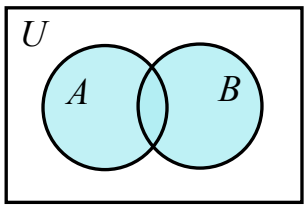
1.3 Операції над множинами

Для наочного зображення операцій будемо використовувати діаграми Венна, в яких круги зображують множини, що беруть участь в операції, а заштрихована частина – результат операції.

1. Об'єднання множин.

Означення:	***Об'єднанням (сумою)*** $\boldsymbol{A \cup B}$ двох множин A і B називається множина C, що складається з тих і тільки тих елементів, які входять або до A, або до B, або до A і B одночасно (рис. 1.1). $$C = A \cup B = \{c \mid c \in A \text{ або } c \in B\}$$

Приклад. Нехай дано множини $A = \{a, b, m\}$; $B = \{m, c, p\}$, тоді їхнє об'єднання $A \cup B = \{a, b, c, m, p\}$.



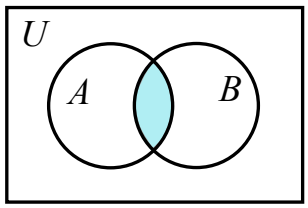
$A \cup B = \{c | c \in A \text{ або } c \in B\}$

Рисунок 1.1 – Діаграма Венна для $A \cup B$
(зафарбовано множину $C = A \cup B$)

2. Перетин множин

Означення:	***Перетином (добутком)*** $\boldsymbol{A \cap B}$ двох множин A та B називається множина C, що містить тільки елементи, які належать до A і B одночасно (рис. 1.2). $$C = A \cap B = \{c \mid c \in A \ i \ c \in B\}$$

Приклад. Для множин A та B з попереднього прикладу $A \cap B = \{m\}$. Якщо $A \cap B = \emptyset$, то множини A та B не перетинаються.



$A \cap B = \{c | c \in A \ i \ c \in B\}$

Рисунок 1.2 – Діаграма Венна для $A \cap B$
(зафарбовано множину $C = A \cap B$)

3. Віднімання множин

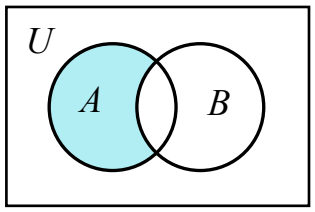
Означення:	***Різницею*** $\boldsymbol{A \backslash B}$ двох множин A та B називається множина C, що складається з усіх елементів A, які не належать до B (рис. 1.3, а). $$C = A \backslash B = \{c \mid c \in A \ i \ c \notin B\}$$

Приклад. Для множин $A = \{a, b, m\}$ та $B = \{m, c, p\}$, їхня різниця $A \backslash B = \{a, b\}$.

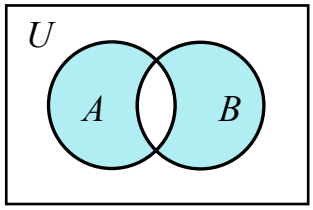
Означення:	***Симетричною різницею*** $\boldsymbol{A \Delta B}$ двох множин A та B називається множина C, що складається з усіх елементів A, які не належать до B, а також з усіх елементів B, які не належать до A (рис. 1.3, б). $$C = A \Delta B = \{c \mid (c \in A \ i \ c \notin B) \text{ або } (c \in B \ i \ c \notin A)\}$$

Симетричною різницею множин A і B *(позначається* $A\Delta B$*) є множина* $(A\backslash B) \cup (B\backslash A)$.

Приклад. Для множин $A = \{a, b, m\}$ та $B = \{m, c, p\}$, їхня симетрична різниця $A\Delta B = \{a, b, c, p\}$.



$A\backslash B = \{c | c \in A \ i \ c \notin B\}$



$A\Delta B = \{c | (c \in A \ i \ c \notin B)$ або $(c \in B \ i \ c \notin A)\}$

а) *б*)

Рисунок 1.3 – Діаграми Венна для $A\backslash B$ та $A\Delta B$
(зафарбовано множину $C = A\backslash B$ (*а*) та $C = A\Delta B$ (*б*))

4. Доповнення (заперечення)

Означення:	***Доповненням*** *множини A* ***(запереченням A)*** є множина $\overline{A}$ (читається *«не A»*) така, що $\overline{A} = U\backslash A$ (рис. 1.4), де *U* – універсальна множина.

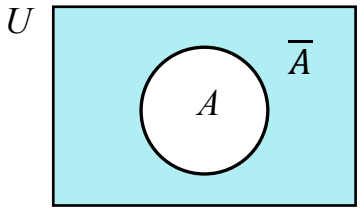


Рисунок 1.4 – Діаграма Венна для $\overline{A}$

5. Розбиття множини A

Означення:	***Розбиттям*** множини *A* називається множина $C = \{A_1, A_2, \ldots, A_k\}$ така, що елементи $A_1, A_2, \ldots, A_k$ – підмножини множини *A*, які взаємно не перетинаються (рис. 1.5). Підмножини A_i називають класами.

C = розбиття $A = \{A_1, A_2, \ldots A_k\}$, причому

$$\bigcup_{i=1}^{k} A_i = A\,, \ \ A_i \cap A_j = \emptyset \ \ (i \neq j \ \text{та} \ \ i, j \in \{1, 2, \ldots, k\})$$

Очевидно, що розбиття множини A на класи можна виконати неоднозначно.

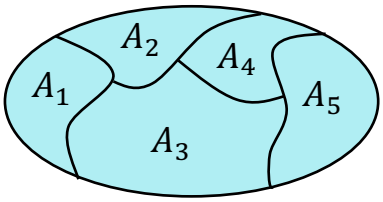


Рисунок 1.5 – Діаграма Венна для операції розбиття

Зауваження: операції 1–4 називають елементарними операціями над множинами.

1.4 Алгебра множин

Множина 2^U всіх підмножин універсальної множини U з заданими на ній чотирма елементарними операціями складає ***алгебру множин***.

У загальному випадку алгебру може складати будь-який клас $\varpi \subset 2^U$ підмножин універсальної множини U, замкнений відносно всіх чотирьох елементарних операцій. Означення алгебри, що не містить надмірних (точніше, залежних) обмежень, виглядає таким чином.

Означення:	Клас множин ϖ називається ***алгеброю*** (множин), якщо: 1. $U \in \varpi$. 2. З $A, B \in \varpi$ випливає $A \cup B \in \varpi$. 3. З $A, B \in \varpi$ випливає $A \setminus B \in \varpi$.

Алгебра множин широко застосовується у програмуванні, зокрема, під час роботи з різноманітними базами даних, і є основою для побудови багатьох математичних структур. Разом з тим, що алгебра множин має основоположне значення в математиці, вона дуже проста і близька до реального життя. Ми щодня застосовуємо операції та закони алгебри множин, не замислюючись над цим. Ми відраховуємо з множин задач, які потрібно розв'язати, множину вже розв'язаних та беремося до розв'язання решти. Із них ми, вірогідно, в першу чергу виберемо ті, які належать до множини легких. Ми готуємо сніданок, визначаючи перетин множини наявних продуктів з множиною продуктів, які нам подобаються. Все наше життя проходить серед множин, які якось взаємопов'язані.

Ми маємо достатньо операцій, щоб створювати складні алгебраїчні вирази. Для цього потрібно визначити, який пріоритет мають операції відносно одна одної.

Пріоритет операцій в алгебрі множин такий:

1. $\overline{A}$.
2. $A \cap B$.
3. $A \cup B$.
4. $A \backslash B$.

Приклад. Нехай треба розташувати дужки (визначити послідовність виконання операцій) у формулі

$$E = \left(A \backslash B \cup \overline{A} \cap D\right) \backslash B.$$

З урахуванням наведених вище пріоритетів це потрібно зробити так

$$E = \left(A \backslash \left(B \cup \left(\overline{A} \cap D\right)\right)\right) \backslash B.$$

В алгебрі множин ϖ автоматично виконуються такі тотожності, які дозволяють віднести ϖ до класу так званих ***булевих алгебр***:

1. Комутативні закони

1.1. $A \cup B = B \cup A$.
1.2. 1.2. $A \cap B = B \cap A$.

2. Асоціативні закони

2.1. $A \cup (B \cup C) = (A \cup B) \cup C$.
2.2. $A \cap (B \cap C) = (A \cap B) \cap C$.

3. Дистрибутивні закони

3.1. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.
3.2. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

4. Властивості порожньої та універсальної множин

4.1. $A \cup \emptyset = A$.
4.2. $A \cap U = A$.
4.3. $A \cup U = U$.
4.4. $A \cap \emptyset = \emptyset$.

5. Закони ідемпотентності

5.1. $A \cup A = A$.
5.2. $A \cap A = A$.

6. Закон інволюції

$\overline{\overline{A}} = A$.

7. ***Закон протиріччя***

$A \cap \overline{A} = \emptyset$.

8. ***Закон виключеного третього***

$A \cup \overline{A} = U$.

9. ***Закон елімінації***

9.1. $A \cap (A \cup B) = A$.
9.2. $A \cup (A \cap B) = A$.

10. ***Закони де Моргана***

10.1. $\overline{A \cup B} = \overline{A} \cap \overline{B}$.
10.2. $\overline{A \cap B} = \overline{A} \cup \overline{B}$.

Усі наведені тотожності можна наочно зобразити і довести, використовуючи діаграми Венна.

Приклад. Спростити вираз

$\left(\overline{\overline{A} \cup B \cup \mathrm{C}}\right) \cap \left(A \cap \left(B \cup \overline{C}\right)\right) \cap \overline{B} =$	(застосуємо закон де Моргана)
$= \left(A \cap \overline{B} \cap \overline{\mathrm{C}}\right) \cap \left(A \cap \left(B \cup \overline{C}\right)\right) \cap \overline{B} =$	(асоціативність і комутативність)
$= A \cap \overline{B} \cap \overline{C} \cap A \cap \overline{B} \cap \left(B \cup \overline{C}\right) =$	(застосуємо закон ідемпотентності)
$= A \cap \overline{B} \cap \overline{C} \cap \left(B \cup \overline{C}\right) =$	(застосуємо дистрибутивний закон
$= A \cap \overline{B} \cap \overline{C} \cap B \cup A \cap \overline{B} \cap \overline{C} \cap \overline{C} =$	
$= \emptyset \cup A \cap \overline{B} \cap \overline{C} = A \cap \overline{B} \cap \overline{C}$	(згідно з властивостями порожньої множини)

Відповідь: $A \cap \overline{B} \cap \overline{C}$.

1.5 Нескінченні множини

У дискретній математиці, як і у всій математиці, в природознавстві, дуже важлива роль натурального ряду чисел N = {1, 2, ..., }. Хоча практично обчислення завжди виконуються зі скінченним відрізком {1, 2, ..., n} цієї нескінченної множини, немає можливості вказати найбільше число *n,* яке не буде перевершене в усіх випадках життя. Тому доводиться досліджувати властивості всієї нескінченної множини чисел *N.* На дійсній числовій осі *R* натуральні та цілі числа утворюють доволі «розріджені» підмножини *N, Z* (рис. 1.6), які інтуїтивно можна назвати «дискретні». Множина раціональних чисел *Q*, що утворюються при діленні цілих чисел *m/n,* щільно розташовується на дійсній осі *R*: інтервал (*a, b*) як завгодно

малої довжини $E = b - a$ ($a \neq b$) містить нескінченну множину різних раціональних точок. Чи можна множину Q вважати «дискретною»? Виявляється так, хоча інтуїція відмовляється це визнати.

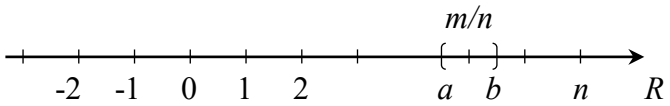


Рисунок 1.6 – Натуральні, цілі та раціональні числа на осі R

Нарешті, вся дійсна вісь R – явно неперервна множина, і тут інтуїтивний висновок нас не підводить. Для строгого визначення дискретних і неперервних нескінченних множин використовуються зіставлення дискретних множин натуральному ряду чисел, а неперервних множин – відрізку [0, 1] дійсної осі. Потрібно тільки уточнити термін «зіставити». Мається на увазі встановити взаємно однозначну відповідність.

Поняття взаємно однозначної відповідності є первинним у свідомості людини при диференційованому сприйнятті предметів зовнішнього світу. Якщо зал для глядачів заповнений і немає вільних місць, нам не обов'язково рахувати кількість присутніх глядачів, вона дорівнює числу крісел у залі. Щоб зробити цей висновок, ми інтуїтивно використовуємо наявність взаємно однозначної відповідності між глядачами і кріслами. Відзначимо одну особливість: фактично у цьому випадку реалізована конкретна взаємно однозначна відповідність глядачі – крісла (кожному глядачеві відповідає одне і тільки одне визначене крісло і навпаки). Після перерви деякі глядачі можуть помінятися місцями, і конкретна відповідність стане іншою, але висновок залишиться попереднім: число глядачів дорівнює числу місць.

Означення:	***Взаємно однозначною*** називається така ***відповідність*** між множинами A і B, при якій кожному елементу $a \in A$ відповідає один і тільки один елемент $b \in B$, і кожному елементу $b \in B$ відповідає один і тільки один елемент $a \in A$. Функція, що визначає взаємно однозначну відповідність, називається ***бієктивною функцією*** *або* ***бієкцією.***

Означення:	Множини A *і* B називаються ***еквівалентними*** або ***рівнопотужними*** ($A \sim B$), якщо між ними можна встановити взаємно однозначну відповідність.

У прикладі з заповненим глядачами залом множина глядачів еквівалентна множині крісел. Таким чином, еквівалентними одна одній

виявляються всі скінченні множини з однаковим числом елементів *n,* і число *n* вважається ***потужністю*** цих множин.

Для нескінченної множини строге поняття потужності не вводиться, але сам термін ***«потужність»*** використовується для позначення властивості, загальної для всіх еквівалентних множин. Якщо дві нескінченні множини A і B еквівалентні $(A \sim B)$, то рівність їхніх потужностей формально записується як $|A| = |B|$.

Означення:	Множина A називається ***зчисленною,*** якщо вона еквівалентна натуральному ряду N $(A \sim N)$. Термін ***«зчисленність»*** є точним замінником інтуїтивного поняття ***«дискретність».***

За допомогою бієкції $\varphi(n) = N \to A$ можна «перелічити» всі елементи a з A, присвоївши їм індекси за правилом $\varphi(n) = a_n$. Можна записати, що $A = \{a_n,\ n = 1, 2, \dots\}$. Множини парних натуральних чисел $N_e = \{2, 4, \dots, m, \dots\}$, всіх натуральних чисел $N = \{1, 2, \dots, n, \dots\}$, цілих чисел Z і раціональних чисел Q послідовно вкладені: $N_e \subset N \subset Z \subset Q$. Хоча для будь-яких двох з цих множин немає рівності, вони ***еквівалентні одна одній,*** тобто мають однакову потужність і є зчисленними: $|N_e| = |N| = |Z| = |Q|$. Тому, відповідно до наведених міркувань, множини $N_e,\ N,\ Z,\ Q$ **є *дискретними***.

Звичайно, існують нескінченні ***незчисленні*** множини, їх потужність природно вважати більшою за $|N|$.

Так, множина точок відрізка $[0, 1] = \{x \in R;\ 0 \le x \le 1\}$ *не є зчисленною* (теорема Г. Кантора). Її потужність називається ***континуум*** і позначається малою буквою *c:* $|[0, 1]| = c$. Сама множина $[0, 1]$ і будь-яка еквівалентна їй множина називаються ***континуальними.***

Виявляється, що на дійсній осі R континуальними (тобто еквівалентними одна одній і відрізку $[0, 1]$) є, наприклад, множини $[a, b]$, (a, b), при будь-якому $a < b$; $(0, +\infty)$; множина $(-\infty, +\infty)$, що дорівнює R.

Континуальні також множини точок будь-якого квадрата та круга на площині R^2, паралелепіпеда та кулі у просторі R^3 і самого простору R^3: $|R| = |R^2| = |R^3| = c$.

Питання для самоконтролю

1) Що таке множина?
2) Які множини називаються скінченними, а які – нескінченними?
3) Наведіть означення впорядкованої множини
4) Що таке потужність множини?
5) Які Ви знаєте способи задання множин?
6) Які множини називаються рівними?

7) Що таке підмножина?
8) Наведіть означення універсальної множини
9) Наведіть означення порожньої множини
10) Що таке булеан?
11) Які Ви знаєте операції над множинами?
12) Що таке алгебра множин?
13) Які Ви знаєте закони алгебри множин?
14) Що таке бієкція?
15) Коли відповідність є взаємно однозначною?
16) Які множини називаються еквівалентними?
17) Наведіть означення дискретності.

Індивідуальні практичні завдання № 1

Задача 1. Обчислення множин. Потрібно обчислити наведену в таблиці 1.1 множину, якщо:

U = {1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14},
A = {1; 2; 3; 4; 7; 9},
B = {3; 4; 5; 6; 11; 12; 13},
C = {2; 3; 4; 7; 8; 12; 13; 14},
D = {1; 7; 14}.

Таблиця 1.1 – Варіанти завдань

1. $(A \cup B) \cap (D \backslash C)$;	31. $(\bar{A} \cup \bar{B}) \cup (\bar{C} \cap D)$;
2. $(A \backslash B \backslash \bar{C}) \cup D$;	32. $\big((A \cap \bar{C}) \cap D\big) \cup (B \cup \bar{A})$;
3. $(\overline{A \cup D}) \backslash (B \backslash C)$;	33. $\Big(\big(\overline{A \cap \bar{D}}\big) \cup (B \cup C)\Big) \cap \bar{A}$;
4. $(A \backslash D) \cup (B \cap C)$;	34. $(\overline{B \cup C}) \cap (\overline{A \cup D}) \cap \bar{A}$;
5. $(A \backslash C)\ \Delta\ (B \cup D)$;	35. $\big((A \backslash B) \cup D\big) \backslash (C \cup D)$;
6. $(A \cup C) \cup (\bar{B} \backslash C)$;	36. $\big((A \cap \bar{C}) \cap D\big) \cup \bar{B}$;
7. $\big((A \cup B) \cup C\big) \cap (B \cup \bar{A})$;	37. $(A \cap B) \cup \big(\overline{\bar{C} \cap D}\big)$;
8. $\big((A \cap B) \cup C\big) \cup D$;	38. $\big((A \backslash B) \cup (\overline{B \cap D})\big) \backslash C$;
9. $(\bar{A} \cup B) \cup (B \cup D)$;	39. $(\bar{A} \cup B)\ \Delta\ (B \cap C)$;
10. $(A \cup \bar{B}) \cap (C \backslash D)$;	40. $(A \cup \bar{D}) \cap (\overline{B \cup C})$;
11. $(A \cap B) \backslash (\overline{C \cup D})$;	41. $\overline{(A \cup B) \cup (\overline{B \cap C})} \cup (\bar{A} \cup D)$;
12. $\big((A \backslash B) \cup (\overline{C \cap A})\big) \cup (D \cup \bar{B})$;	42. $(A \cap B) \cup (\bar{D} \backslash C)$;

Продовження таблиці 1.1

13. $\big((A\backslash C)\backslash(B\cap D)\big)\cup A$;	43. $(A\cup B)\cup(D\backslash\bar{C})$;
14. $(\bar{B}\cup C)\cup(D\backslash A)$;	44. $\big((\overline{B\cap D})\backslash(C\cup D)\big)\cup\bar{A}$;
15. $(A\cap D)\cap(B\cap\bar{C})$;	45. $(\overline{A\cup D})\cap(B\cup C)$;
16. $\big(B\cap(\overline{D\cap C})\big)\cup A$;	46. $\big(\overline{\bar{A}\cup B}\big)\cup(\bar{C}\cap\bar{D})$;
17. $(A\cap B)\cup(C\cap\bar{D})$;	47. $\big(A\cup(B\cup C)\big)\cap\bar{D}$;
18. $(A\cap B)\cap\big(A\backslash(\overline{C\cap D})\big)$;	48. $\big((A\cap B)\cup(\overline{C\cap D})\big)\cap\bar{A}$;
19. $\big((B\cup D)\cap A\big)\cup\bar{C}$;	49. $\big((A\cup D)\cup C\big)\cap B$;
20. $\big((A\cup C)\backslash(B\cap D)\big)\cup(\overline{A\cap D})$;	50. $(B\cup C)\cup\big(C\cap(A\backslash\bar{D})\big)$;
21. $(\overline{A\cap B})\cup\big(\overline{\bar{C}\cap\bar{D}}\big)$;	51. $\overline{(B\cap C)\cup\bar{D}}\cap(A\cap D)$;
22. $\big((A\backslash\bar{C})\cap(B\cap D)\big)\cup D$;	52. $(A\cap B)\backslash\big(C\cup(D\cup\bar{A})\big)$;
23. $(A\backslash D)\cup(\bar{B}\cup C)$;	53. $(\overline{A\cup B})\cup(A\cap D)\cup\big(\overline{\bar{A}\cup\bar{C}}\big)$;
24. $\big((A\cup B)\cap(C\cup D)\big)\cup(\bar{A}\cap B)$;	54. $\big(C\cup(\bar{A}\cup B)\big)\backslash(A\cap D)$;
25. $\big((A\cap B)\cup C\big)\cup(\overline{D\cap A})$;	55. $\big(\overline{\bar{C}\cap A}\big)\cap(B\cup D)$;
26. $\big((A\backslash B)\cup C\big)\cap\bar{D}$;	56. $\Big(A\cup\big(\overline{\bar{B}\cup A}\big)\cup(\bar{C}\cap D)\Big)$;
27. $A\cap\Big(D\cup\big(\bar{\bar{B}}\cap C\big)\Big)$;	57. $(A\backslash D)\cap\big(B\cap(C\cup\bar{A})\big)$;
28. $\big((\overline{A\cup B})\cup(C\cap D)\big)\cup(A\cap B)$;	58. $\big((B\cup D)\,\Delta\,A\big)\cup\bar{C}$;
29. $(A\cup B)\cup\big(\overline{C\cap\bar{D}}\big)$;	59. $(A\cup D)\,\Delta\,(\bar{B}\backslash C)$;
30. $(A\,\Delta\,B)\cup\big(\overline{C\backslash\bar{D}}\big)$;	60. $(D\cup C)\cap(\mathrm{A}\,\Delta\,\mathrm{B})\cup\bar{D}$.

Приклад. Нехай U = {1, 2, 3, 4, 5, 6, 7, 8}, A = {1, 2, 3, 4, 5},
B = {2, 4, 6, 8}, C = {1, 3, 5, 7}, D = {1, 2, 4, 5, 7, 8}.

Знайти $(D\backslash A)\cap(B\cup C)\cup(C\backslash D)$.

Розв'язання. Виконаємо операції покроково:

1) $(D\backslash A)=\{7,8\}$;
2) $(B\cup C)=\{1,2,3,4,5,6,7,8\}$;
3) $(D\backslash A)\cap(B\cup C)=\{7,8\}\cap\{1,2,3,4,5,6,7,8\}=\{7,8\}$;
4) $(C\backslash D)=\{3\}$;
5) $(D\backslash A)\cap(B\cup C)\cup(C\backslash D)=\{7,8\}\cup\{3\}=\{3,7,8\}$.

Задача 2. Визначення множин. Необхідно визначити множину, подану в таблиці 1.2 через відомі множини *A*, *B*, *C*, *D*, або довести, що це зробити неможливо, якщо:

$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$;
$A = \{1, 2, 3, 4, 5, 9\}$;
$B = \{2, 4, 6, 8\}$;
$C = \{1, 3, 5, 7\}$;
$D = \{1, 4, 5, 7, 8, 9\}$.

Таблиця 1.2

1.{5; 6; 3; 4; 7; 1; 8};	16.{1; 4; 8; 5; 2; 9; 3; 6; 7};
2.{7; 1; 4; 3; 8; 5; 9; 6};	17.{8; 4; 2; 7; 1; 3};
3.{9; 8; 6; 4};	18.{6; 7; 9; 3; 5; 1; 2; 4};
4.{5; 7; 1; 6};	19.{1; 9; 6; 4; 7; 5; 8; 2};
5.{3; 1; 6; 2; 7};	20.{3; 4; 1; 7; 8; 9; 5};
6.{8; 6; 1; 2; 3; 7; 9; 5};	21.{5; 8; 1; 2; 7; 4};
7.{2; 9; 7; 1; 6};	22.{7; 9; 4; 5; 6; 1; 2};
8.{1; 9; 2; 7};	23.{1; 6; 8; 4; 5; 3; 2};
9.{3; 1; 2; 6; 7; 9; 4};	24.{5; 6; 2; 3; 7; 8; 4};
10.{1; 2; 6};	25.{4; 1; 8; 5; 7};
11.{4; 7; 3; 6};	26.{3; 6; 5; 7; 8};
12.{2; 6};	27.{3; 4; 1; 5; 2};
13.{5; 8; 4; 6; 3};	28.{6; 9; 3; 7; 1};
14.{3; 8; 5; 9; 7; 6; 4; 1};	29.{2; 3; 7; 6; 8};
15.{7; 9; 8; 1; 5; 6};	30.{4; 1; 2; 3; 5; 9}.

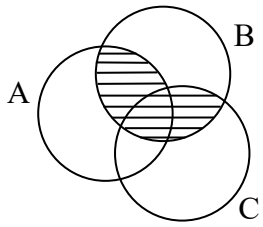
Приклад. Нехай $U = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $A = \{1, 2, 3, 4, 5\}$,
$B = \{2, 4, 6, 8\}$, $C = \{1, 3, 5, 7\}$, $D = \{1, 2, 4, 5, 7, 8\}$.
Визначити через відомі множини *A*, *B*, *C*, *D* множину $\{1, 5\}$.

Розв′язання.

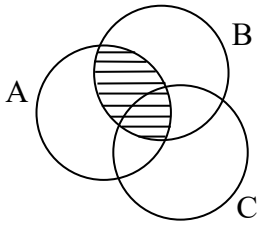
1) $A \cap C = \{1, 3, 5\}$;
2) $C \backslash D = \{3\}$;
3) $(A \cap C) \backslash (C \backslash D) = \{1, 3, 5\} \backslash \{3\} = \{1, 5\}$.

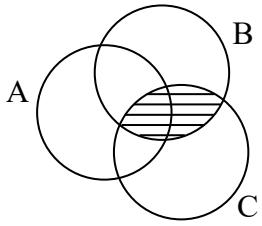
Задача 3. Кола Ейлера (діаграми Венна). Для завдань, що подані у додатку А, визначити зафарбовану область через позначені множини.

Приклад. Виразити через множини *A, B* і *C* множину *E*, якій відповідає заштрихована область.



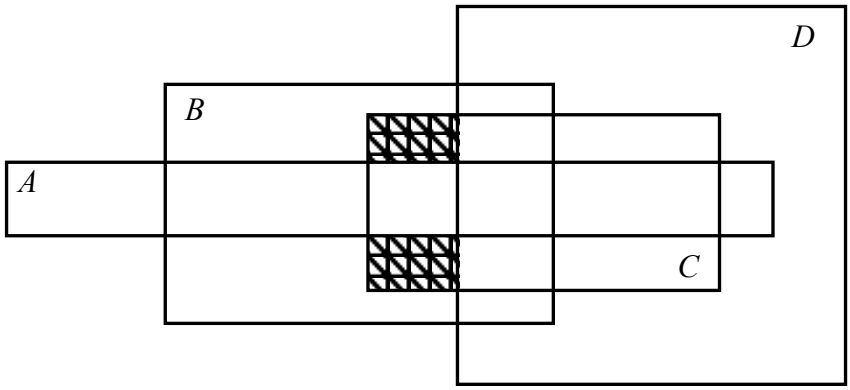
Розв'язання. Оскільки заштрихована множина на лівому рисунку відповідає множині $A \cap B$, а множина на правому рисунку відповідає множині $B \cap C$, то множина E є об'єднанням цих множин.





Тобто, $E = (A \cap B) \cup (B \cap C)$.

Приклад. Визначити через множини *A, B, C, D* множину *E*, якій відповідає заштрихована область.



Розв'язання.

$$E = (C \backslash A) \backslash D.$$

Тема 2. ВІДНОШЕННЯ

2.1 Поняття відношення. Способи задання відношень

Відношення реалізують у математичних термінах на абстрактних множинах реальні зв'язки між реальними об'єктами. Відношення застосовуються при побудові комп'ютерних баз даних, які організовані у вигляді таблиць даних. Зв'язки між групами даних у таблицях описуються мовою відношень. Самі дані обробляються і перетворюються за допомогою операцій, математично точно визначених для відношень. Такі бази даних називаються реляційними і широко застосовуються для зберігання та обробки найрізноманітнішої інформації: виробничої, комерційної, статистичної тощо. Відношення також часто використовуються в програмуванні. Такі складові структури даних, як списки, дерева тощо звичайно використовуються, щоб описувати деяку множину даних разом з відношенням між елементами цієї множини.

Щоб формально описати будь-які комбінації з елементів множин, що входять до відношення, використовується поняття декартового добутку множин.

Означення:	***Декартовим добутком*** $D = X \times Y$ називається множина впорядкованих пар (x_i, y_j) елементів $x_i \in X = \{x_1, \dots, x_n\}$ та $y_j \in Y = \{y_1, \dots, y_m\}$.

Тобто, $D = \{(x_1, y_1), \dots, (x_1, y_m), (x_2, y_1), \dots, (x_n, y_1), \dots, (x_n, y_m)\}$.

Наприклад, якщо $X = \{a, b\}$, $Y = \{1, 2, 3\}$, то

$$D = X \times Y = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}.$$

Потужність множини D: $|D| = n \times m$. За множину Y може бути прийнята множина X. Тоді отримаємо декартів квадрат множини X.

$$D = X \times X = \{(a, a), (a, b), (b, a), (b, b)\}.$$

Означення:	***Бінарним відношенням*** називається підмножина R множини D $(R \subseteq D)$. Символ R використовується для позначення підмножини і для позначення того, що елементи x_i та y_j знаходяться у відношенні R. $(x_i, y_j) \in R \leftrightarrow x_i \, R \, y_j$

Приклади: діагональне відношення $E = \{(x_i, x_i)/x_i \in X\}$;
повне відношення $U = D = X \times Y$;
відношення суворого порядку $R: =, >$;
відношення несуворого порядку $R: =, \geq$;
функціональне відношення (визначає функцію $y = R(x)$), у разі якого для кожного елемента $x_i \in X$ існує один і тільки один елемент $y_j \in Y$ такий, що $(x_i, y_j) \in R$.

Останній приклад показує, що поняття відношення є більш загальним ніж поняття функції.

Способи задання бінарних відношень

Якщо R – бінарне відношення на множинах X, Y, то факт $(x_i, y_j) \in R$ часто записується у вигляді $x_i \, R \, y_j$, і говорять, що елемент $x_i \in X$ знаходиться у відношенні R з елементом $y_j \in Y$.

Спосіб 1. У вигляді списку, елементами якого є пари, з яких складається відношення.

Приклад. На множинах чисел A = {1, 2, 3, 4, 5, 6, 7, 8, 9}; B = {24, 25, 26} побудуємо відношення «дільник», яке складається з упорядкованих пар вигляду (x_i, y_j), де x_i – дільник y_j, $x_i \in A$, $y_j \in B$. Позначимо це відношення через R

R = {(1, 24), (1, 25), (1, 26), (2, 24), (2, 26), (3, 24), (4, 24), (5, 25), (6, 24), (8, 24)}.

Спосіб 2. За допомогою матриці $(W = W(R))$, рядки якої відповідають елементам множини X, стовпці – елементам множини Y. Якщо $n = |X|$, $m = |Y|$ – кількості елементів множин X та Y відповідно, то довільна матриця W має розмір $n \times m$. Елемент w_{ij} матриці відповідає парі $(x_i, y_j) \in A \times B$ декартового добутку множин, причому $w_{ij} = 1$, якщо $(x_i, y_j) \in R$ та $w_{ij} = 0$, якщо $(x_i, y_j) \notin R$, тобто коли відношення R не містить пару (x_i, y_j).

Приклад. Наведена нижче матриця W задає відношення R «дільник» для числових множин A і B з попереднього прикладу:

W =

A \ *B*	**24**	**25**	**26**
1	1	1	1
2	1	0	1
3	1	0	0
4	1	0	0
5	0	1	0
6	1	0	0
7	0	0	0
8	1	0	0
9	0	0	0

Спосіб 3. Графічно. На площині зобразимо точками x_i та y_j, елементи множин X і Y. Якщо пара (x_i, y_j) належить відношенню R, з'єднаємо точки x_i та y_j лінією, яка спрямована від першого елемента пари до другого. Позначивши таким чином всі пари, що належать відношенню R, одержимо фігуру, яка називається графом відношення. Спрямовані лінії, що з'єднують пари точок, називаються дугами, а точки, що зображують елементи множин, – вершинами графу. Якщо бінарне відношення R задане на одній множині $X(R \subseteq X^2)$, то вершинами графу будуть елементи множини X.

Приклад. Граф на рис. 2.1 задає відношення R «дільник» для числових множин A і B з попереднього прикладу.

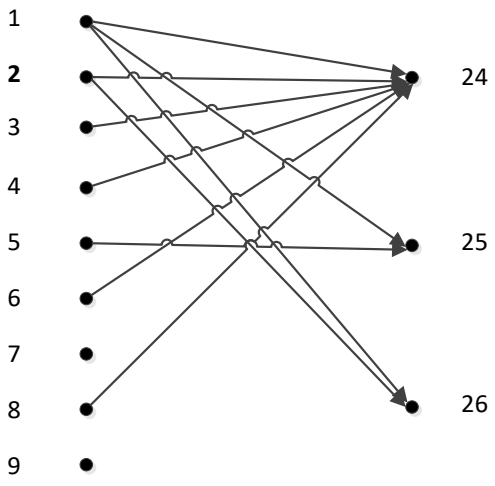


Рисунок 2.1 – Граф, що задає відношення R «дільник» для числових множин A = {1, 2, 3, 4, 5, 6, 7, 8, 9} і B = {24, 25, 26}

Ми розглянули декартів добуток двох множин та задані на ньому відношення. Такі відношення є бінарними (від «бі-», що означає 2). У

загальному випадку відношення можуть бути *n*-арними, тобто складатися з довільної кількості елементів. Тоді вони визначаються на декартовому добутку *n* множин.

Означення:	***Декартовим добутком*** множин $X_1 \times X_2 \times ... \times X_n$ називається множина всіх можливих упорядкованих наборів $(x_1, x_2, ..., x_n)$ з n елементів (які називають ***кортежами*** довжини *n*), в яких перший елемент належить множині X_1, другий – множині X_2, *…, n-й* – множині X_n. Декартів добуток $X \times X \times ... \times X$, в якому одна й та ж множина X помножується n раз сама на себе, називають ***декартовим степенем*** множини X і позначають X^n. Водночас $X^1 = X$. Множину X^2 називають ***декартовим квадратом*** множини X, множину X^3 – ***декартовим кубом*** множини X.

Приклад. Нехай $A = \{a_1,\ a_2,\ a_3\},\ \ B = \{b_1,\ b_2\},\ C = \{c_1,\ c_2\}$.

Тоді,

$A \times B = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2), (a_3, b_1), (a_3, b_2)\}$.

$B \times A = \{(b_1, a_1), (b_1, a_2), (b_1, a_3), (b_2, a_1), (b_2, a_2), (b_2, a_3)\}$.

$A \times B \times C = \{(a_1, b_1, c_1), (a_1, b_1, c_2), (a_1, b_2, c_1), (a_1, b_2, c_2), (a_2, b_1, c_1),$
$(a_2, b_1, c_2), (a_2, b_2, c_1), (a_2, b_2, c_2), (a_3, b_1, c_1),$
$(a_3, b_1, c_2), (a_3, b_2, c_1), (a_3, b_2, c_2)\}$.

$B^2 = \{(b_1, b_1), (b_1, b_2), (b_2, b_1), (b_2, b_2)\}$.

Порядок проходження пар може бути довільним, але розміщення елементів у кожній парі визначається порядком проходження множин, що перемножуються, тобто $A \times B \neq B \times A$, якщо $A \neq B$.

Означення:	***n-арне відношення R*** на множинах $X_1 \times X_2 \times ... \times X_n$ – це підмножина декартового добутку цих *n* множин: $R \subseteq X_1 \times X_2 \times ... \times X_n$.

Якщо набір елементів $(x_1, x_2, ..., x_n)$ належить відношенню R, то стверджують, що елементи $x_1, x_2, ..., x_n$ знаходяться у відношенні R. Під ***n-арним відношенням R на множині X*** розуміється підмножина *n*-го степеня цієї множини: $R \subset X^n$. Якщо $n = 1$, то відношення називається ***унарним***, якщо $n = 2$ – ***бінарним***. Зауважимо, що унарне відношення R на множині X – це підмножина в самому X: $R \subset X$.

Приклад. Відношенням на множинах A, B, C з попереднього прикладу є будь-яка підмножина множини $A \times B \times C$, зокрема

$R_1 = \{(a_1, b_1, c_1), (a_2, b_1, c_1), (a_2, b_1, c_2), (a_3, b_1, c_1), (a_3, b_1, c_2), (a_3, b_2, c_2)\}$;
$R_2 = \{(a_2, b_2, c_1), (a_2, b_2, c_2), (a_3, b_1, c_1)\}$.

Розглянемо окремо бінарні відношення, які є «базисними» у тому розумінні, що будь-яке n-арне відношення можна зобразити у вигляді ланцюжка бінарних відношень, що послідовно конструюються. Цей очевидний факт є наслідком асоціативності декартового добутку множин.

2.2 Операції над відношеннями

Оскільки відношення є множинами, то для них можна застосовувати всі відомі теоретико-множинні операції: *об'єднання, перетин, доповнення, різниця, симетрична різниця, розбиття* тощо.

Введемо ще дві операції, які стосуються відношень: *обернення відношення* та *композиція відношень*.

Означення:	Нехай R – бінарне відношення. ***Обернене*** відношення до R позначається R^{-1}. ***Упорядкована пара*** (y, x) належить R^{-1} тоді і тільки тоді, коли (x, y) належить R.

Якщо $(x_i, y_j) \in R$, то $(y_j, x_i) \in R^{-1}$.

Приклад. Для відношення «*більше або дорівнює*» оберненим є відношення «*менше або дорівнює*».

$(\geq)$	**1**	**2**	**3**	**4**
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

$(\geq)^{-1}$	**1**	**2**	**3**
1	1	1	1
2	0	1	1
3	0	0	1
4	0	0	0

Для обернених відношень справедливі дві нижченаведені властивості:
1) $(R^{-1})^{-1} = R$.
2) Якщо $R_1 \subseteq R_2$, то ${R_1}^{-1} \subseteq {R_2}^{-1}$.

Означення:	Якщо $R_1 \subseteq X \times Y$, а $R_2 \subseteq Y \times Z$, то ***композицією*** (добутком) відношень буде відношення $R_2 \circ R_1 = \{(x_i, z_k) \mid \exists y_j \in Y$ таке, що $(x_i, y_j) \in R_1$, а $(y_j, z_k) \in R_2\}$.

Приклад. Знайти композицію $R_2 \circ R_1$, якщо $R_1 = X \subseteq Y$, $R_2 = Y \subseteq Z$, $X = \{\{1\}, \{1, 2\}, \{2, 3\}\}$, $Y = \{\{1, 4\}, \{1, 2, 3\}\}$, $Z = \{\{1, 3, 4\}, \{1, 2, 4\}\}$.

R_1	$\{1,4\}$	$\{1,2,3\}$
$\{1\}$	1	1
$\{1,2\}$	0	1
$\{2,3\}$	0	1

R_2	$\{1,3,4\}$	$\{1,2,4\}$
$\{1,4\}$	1	1
$\{1,2,3\}$	0	0

$R_2 \circ R_1$	$\{1,3,4\}$	$\{1,2,4\}$
$\{1\}$	1	1
$\{1,2\}$	0	0
$\{2,3\}$	0	0

2.3 Властивості бінарних відношень

Наведемо список важливих властивостей, за якими класифікують відношення.

***1. Рефлексивність**:*

а) Відношення $R \subseteq X \times X$ називається ***рефлексивним***, якщо для будь-якого $x_i \in X$ пара $(x_i, x_i) \in R$.

Наприклад, відношення $R = \geq$ – рефлексивне.

б) Відношення $R \subseteq X \times X$ називається ***антирефлексивним*** (ірефлексивним), якщо для всіх $x_i \in X$ пара $(x_i, x_i) \notin R$.

Наприклад, відношення «*більше*», «*менше*», «*є сином*» – антирефлексивні.

в) Відношення $R \subseteq X \times X$ називається ***нерефлексивним***, якщо хоча б для одного $x_i \in X$ пара $(x_i, x_i) \notin R$, а для решти $(x_i, x_i) \in R$.

Всі елементи головної діагоналі матриці рефлексивного відношення дорівнюють 1, а для антирефлексивного відношення дорівнюють 0.

***2. Симетричність**:*

а) Відношення $R \subseteq X \times X$ називається ***симетричним***, якщо з $(x_i, x_j) \in R$ слідує, що $(x_j, x_i) \in R$.

б) Відношення $R \subseteq X \times X$ називається ***антисиметричним***, якщо з $(x_i, x_j) \in R$ слідує, що $(x_j, x_i) \in R$ лише коли $x_i = x_j$.

в) Відношення $R \subseteq X \times X$ називається ***асиметричним***, якщо з $(x_i, x_j) \in R$ слідує, що $(x_j, x_i) \notin R$.

Наприклад, відношення $R = \neq$ – симетричне, відношення $R = \geq$ – антисиметричне, а відношення $R = >$ – асиметричне

Для симетричних відношень $R = R^{-1}$.

***3. Транзитивність**:*

а) Відношення $R \subseteq X \times X$ називається ***транзитивним***, якщо зі співвідношень $(x_i, x_k) \in R$ та $(x_k, x_j) \in R$ слідує, що $(x_i, x_j) \in R$ (іншими словами, якщо з aRb і bRc випливає aRc).

Наприклад, відношення $R = \geq$ та відношення $R = \parallel$ (паралельність на множині прямих ліній) – транзитивні.

Неважко переконатись, що відношення R транзитивне тоді і тільки тоді, коли $R_2 \circ R_1 \subseteq R$.

2.4 Відношення еквівалентності, толерантності, порядку

R називається відношення еквівалентності, якщо воно має властивості рефлексивності, транзитивності і симетричності.

R називається відношенням толерантності, якщо воно має властивості рефлексивності і симетричності, але не є транзитивним. Наприклад, відношення «*бути схожим*» (x_i схоже на x_k, x_k схоже на x_j, але x_i може не бути схожим на x_j).

R називається відношенням несуворого порядку, якщо воно має властивості рефлексивності, антисиметричності і транзитивності. Наприклад, $R = \geq$ (більше або дорівнює).

R називається відношенням суворого порядку, якщо воно має властивості антирефлексивності, асиметричності і транзитивності. Наприклад, $R = >$ (більше).

Питання для самоконтролю

1) Що є декартовим добутком?
2) Наведіть способи задання бінарних матриць.
3) Що таке рефлексивність?
4) Скільки є властивостей бінарних відношень? Наведіть їх.
5) Що таке транзитивність?

Індивідуальні практичні завдання № 2

Задача. Задано множину $D = \{1, 2, 3, 4, 5, 6\}$ та відношення $R \subseteq D \times D$ у вигляді предиката (табл. 2.1). Визначити впорядковані пари відношення, що відповідають предикату, подати їх графічно та у матричному вигляді. Класифікувати отримане відношення.

Таблиця 2.1 – Варіанти завдань

Номер варіанта	Предикат
1	2
1.	$(m, n) \in R \Leftrightarrow$, де m є парним числом, а n – кратне трьом.
2.	$(m, n) \in R \Leftrightarrow$, де $\frac{m \cdot n}{2}$ є простим числом.

Продовження таблиці 2.1

1	2
3.	$(m,n) \in R \Leftrightarrow$, де $\lvert m-n \rvert$ є простим числом.
4.	$(m,n) \in R \Leftrightarrow$, де $m^2+n^2 \geq 20$.
5.	$(m,n) \in R \Leftrightarrow$, де $(m+n)$ є парним числом.
6.	$(m,n) \in R \Leftrightarrow$, де $\lvert m-n \rvert \leq 2$.
7.	$(m,n) \in R$, де m^2+n є простим числом.
8.	$(m,n) \in R \Leftrightarrow$, де $min(m,n)=3$.
9.	$(m,n) \in R \Leftrightarrow$, де $(3m+1-2n)$ є парним числом.
10.	$(m,n) \in R \Leftrightarrow$, де m є дільником $(n+2)$.
11.	$(m,n) \in R \Leftrightarrow$, де $(4n-m+3)$ є непарним числом.
12.	$(m,n) \in R \Leftrightarrow$, де m ділиться на n без остачі.
13.	$(m,n) \in R \Leftrightarrow$, де $m+n=8$.
14.	$(m,n) \in R \Leftrightarrow$, де $(m+n-1)$ є складним числом.
15.	$(m,n) \in R \Leftrightarrow$, де (m^2+n) – кратне трьом.
16.	$(m,n) \in R \Leftrightarrow$, де $4 < m \cdot n < 30$.
17.	$(m,n) \in R \Leftrightarrow$, де $\frac{m}{n} > 1$.
18.	$(m,n) \in R \Leftrightarrow$. де $(m-n)$ ділиться на 2 без остачі.
19.	$(m,n) \in R \Leftrightarrow$, де $m^2+n^2 \leq 40$.
20.	$(m,n) \in R \Leftrightarrow$, де m є простим числом, а n – складним.
21.	$(m,n) \in R \Leftrightarrow$, де $\frac{m}{n} < 3$.
22.	$(m,n) \in R \Leftrightarrow$, де $(m+2n)$ є складним числом.
23.	$(m,n) \in R \Leftrightarrow$, де $(m+n)$ ділиться на 3 без остачі.
24.	$(m,n) \in R \Leftrightarrow$, де (m^2-n) – кратне двом.
25	$(m,n) \in R \Leftrightarrow$, де m ділиться на n без остачі.
26.	$(m,n) \in R \Leftrightarrow$, де $m-n \leq 8$.
27.	$(m,n) \in R \Leftrightarrow$, де $m \cdot n \geq 10$.
28.	$(m,n) \in R \Leftrightarrow$, де $\lvert m-n \rvert = 3$.
29.	$(m,n) \in R \Leftrightarrow$, де $m+n > 6$.
30.	$(m,n) \in R \Leftrightarrow$, де $3 < m+n \leq 9$.

Приклад. Задано множину $D = \{1, 2, 3, 4, 5\}$ та відношення $R \subseteq D \times D$, що визначено так: $(m, n) \in R \Leftrightarrow 0 < m^2 - n < 9$. Визначити упорядковані пари відношення та подати їх у матричному вигляді а також графічно у вигляді графу. Класифікувати отримане відношення.

Розв'язання. Для кожної впорядкованої пари (m, n) з множини D обчислимо $(m^2 - n)$ та перевіримо виконання умови $0 < m^2 - n < 9$.

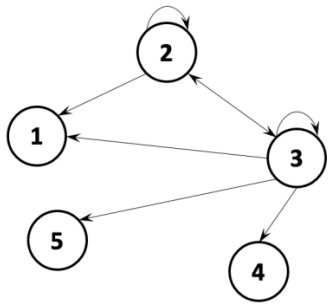
1) Отримуємо відношення R у вигляді списку:

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

2) Подамо отримане відношення R у матричному вигляді

R	**1**	**2**	**3**	**4**	**5**
1	0	0	0	0	0
2	1	1	1	0	0
3	1	1	1	1	1
4	0	0	0	0	0
5	0	0	0	0	0

3) Зобразимо отримане відношення R у вигляді графу



4) Класифікуємо отримане відношення R:

- відношення не рефлексивне, бо не всі пари (m, m) входять до R, зокрема $(1, 1) \notin$ R;
- відношення не симетричне, оскільки не для всіх $(m, n) \in R$ виконується умова $(n, m) \in R$. Наприклад, $(3, 1) \in R$ але $(1, 3) \notin R$.
- відношення не є антисиметричним, оскільки $(3, 2) \in R$ та $(2, 3) \in R$.
- відношення не транзитивне, оскільки не виконується умова, що для всіх $(m, n) \in R$ і $(n, p) \in R$ має також $(m, p) \in R$. Зокрема, у заданому відношенні $(2, 3) \in R$ та $(3, 4) \in R$, проте $(2, 4) \notin R$.

Тема 3. ЗАДАЧА ПРО ПОКРИТТЯ

3.1 Постановка задачі про покриття

Задача про покриття є моделлю великої кількості оптимізаційних задач дискретної математики.

Приклад. Задача «про повну збірку творів». Деякий письменник видав множину $A = \{A_1, \dots, A_m\}$ збірок, які містять у сукупності всю множину його творів $B = \{B_1, \dots, B_n\}$. Кожна збірка A_i, містить деяку підмножину творів з B $(A_i \subseteq B)$ і має ціну a_i. Потрібно знайти таку множину $P = \{A_{i_1}, \dots, A_{i_l}\}, (i \in \{1, \dots, m\},\ l \leq m)$ збірок, щоб в їхній сукупності містились всі твори автора

$$\bigcup_{j=1}^{l} A_{i_j} = B \tag{3.1}$$

і при цьому ціна

$$S = \sum_{j=1}^{l} a_{i_j} \tag{3.2}$$

такої повної збірки творів була найменшою, або кількість збірок l була мінімальною.

Приклад.

$$B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9\}$$
$$A_1 = \{b_1, \ b_3, \ b_6, b_7\} \qquad a_1 = 1;$$
$$A_2 = \{b_1, b_2, \ b_4, \ b_8\} \qquad a_2 = 2{,}5;$$
$$A_3 = \{\ b_2, \ b_5, \ b_7\} \qquad a_3 = 2;$$
$$A_4 = \{\ b_3, \ b_8, b_9\} \qquad a_4 = 1{,}5;$$
$$A_5 = \{\ b_4, b_5, \ b_8, b_9\} \qquad a_5 = 3;$$
$$A_6 = \{\ b_3, \ b_5, b_6, b_7, \ b_9\} \qquad a_6 = 7;$$
$$A_7 = \{\ b_2, \ b_4, \ b_6\} \qquad a_7 = 1.$$

Множина $P_i = \{A_{i_1}, \dots, A_{i_l}\}$, що задовольняє умову (3.1), є можливим розв'язком задачі. Розглянемо деякі з таких розв'язків:

1. $P_1 = \{A_1, A_2, A_5\}$, $A_1 \cup A_2 \cup A_5 = B$, $S_1 = 1 + 2{,}5 + 3 = 6{,}5$, $l_1 = 3$.

2. $P_2 = \{A_2, A_6\}$, $A_2 \cup A_6 = B$, $S_2 = 2{,}5 + 7 = 9{,}5$, $l_2 = 2$.

3. $P_3 = \{A_1, A_3, A_4, A_7\}$, $A_1 \cup A_3 \cup A_4 \cup A_7 = B$, $S_3 = 1 + 2 + 1{,}5 + 1 = 5{,}5$, $l_3 = 4$.

Можна побудувати багато розв'язків (індекс *i* показує, що не всі збірки $A = \{A_1, \dots, A_m\}$ потрапляють в множину P, але ті, що потрапили, містять всі твори з B).

Але не кожна підмножина з A є розв'язком. Наприклад, підмножина $\{A_1, A_2\} \subseteq A$ збірок не є повним зібранням творів, оскільки не містить творів b_5 та b_9.

Умова (3.1) є необхідною умовою для побудови розв'язку, до того ж умови $S \to min$ або $l \to min$ виступають критеріями, за якими з можливих розв'язків $P_1, P_2, \dots$ вибирають найкращий.

У цьому прикладі за критерієм $S \to min$ обираємо P_3, а за критерієм $l \to min$ – P_2. Оскільки не було розглянуто інших можливих розв'язків, то неможливо гарантувати, що знайдені розв'язки є найкращими (оптимальними).

Формулювання задачі про покриття мовою теорії множин

Нехай $B = \{B_1, \dots, B_n\}$ – опорна множина. Існує множина $A = \{A_1, \dots, A_m\}$ підмножин множини B

$$\left(A_i \subseteq B \quad \text{та} \quad \bigcup_{i=1}^{m} A_i = B \right).$$

Кожній підмножині A_i зіставлено число a_i, що як і раніше називаємо ціною. Множина $P = \{A_{i_1}, \dots, A_{i_l}\}, (l \le m)$ називається розв'язком задачі про покриття або просто покриттям, якщо використана умова (3.1)

$$\bigcup_{j=1}^{l} A_{i_j} = B$$

Поясненням терміна «покриття» є те, що сукупність множин $A_{i_1}, \dots, A_{i_l}$ містить всі елементи множини B, тобто покриває множину B.

Теорема 3.1. Якщо P – покриття, то і $P' \supseteq P$ – також покриття, оскільки множина всіх можливих покриттів увігнута.

Означення:	Покриття P називається ***безнадлишковим***, якщо у разі видалення з нього хоча б одного елемента воно перестає бути покриттям. Інакше покриття – ***надлишкове***.

Означення:	Покриття P називається ***мінімальним***, якщо його ціна $S = \sum_{j=1}^{l} a_{i_j}$ – найменша серед всіх покриттів даної задачі.

Означення:	Покриття P називається ***найкоротшим***, якщо l – найменше серед всіх покриттів даної задачі.

Теорема 3.2. Мінімальне і найкоротше покриття – безнадлишкові.

Приклади технічного застосування задачі про покриття.

До задачі про покриття зводиться багато технічних задач. Наприклад, задача «про мінімальний перевірний набір тестів».

Приклад. Нехай деякий пристрій має множину $B = \{b_1, \dots, b_n\}$ елементів, кожний з яких може знаходитись у працездатному і непрацездатному стані. Нехай тест T_i – деяка дія на пристрій. Вона дозволяє визначити працездатність підмножини $A_i \subseteq B$ елементів за правильною чи неправильною реакцією на дію T_i.

Задана множина $T = \{T_1, \dots, T_m\}$ допустимих тестів і відповідна їй множина $A = \{A_1, \dots, A_n\}$ підмножин елементів, що перевіряються,

$$\left(\bigcup_{i=1}^{n} A_i = B \right).$$

Потрібно знайти набір тестів $\{T_{i_1}, \dots, T_{i_l}\}$, які перевіряють працездатність всіх елементів $b_j \in B$ пристрою.

$$\left(\bigcup_{j=1}^{l} T_{i_j} = B \right).$$

Розв'язання задачі. Можна знаходити найкоротші $(l \to min)$ або мінімальні набори тестів. Ціна a_i теста T_i може мати, наприклад, зміст часу дії теста на пристрій, тоді мінімальний набір тестів

$$\left(\sum_{j=1}^{l} a_{i_j} \to min \right)$$

має найменший час перевірки працездатності всіх елементів пристрою.

Іншими прикладами є побудова таблиці у задачі мінімізації булевих функцій та задача теорії графів про мінімальне розфарбування.

3.2 Таблиця покриттів

Таблиця покриттів T дуже зручна для подання початкових даних у задачах про покриття. Ця таблиця подається матрицею T відношення залежності елементів множини $A_i \in A$ опорній множині B. Стовпці – елементи B, а рядки – A. $T_{i_j} = 1$ якщо $b_j \in B$ та $b_j \in A_i$. $T_{i_j} = 0$ якщо $b_j \in B$ та $b_j \notin A_i$. Нулі в матрицю не записуються.

Для прикладу (С. 32) таблиця покриттів має такий вигляд:

B	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	a_i
A_1	1		1			1	1			1
A_2	1	1		1				1		2.5
A_3		1			1		1			2
A_4			1					1	1	1.5
A_5				1	1			1	1	3
A_6			1		1	1	1		1	7
A_7		1		1		1				1

Рисунок 3.1

Мовою таблиці покриттів задача про покриття формулюється так: покриття – це така підмножина рядків, у якій в усіх стовпцях є одиниці (покрити одиницями всі стовпці матриці). Така підмножина рядків називається покриттям.

В цьому прикладі $P_1 = \{A_1, A_2, A_5\}$ – покриття.

B	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	a_i
A_1	1		1			1	1			1
A_2	1	1		1				1		2.5
A_5				1	1			1	1	3

Рисунок 3.2

Принаймні одна одиниця є у кожному стовпці.

3.3 Методи розв'язання задачі про покриття

3.3.1 Метод повного перебору

Очевидним способом побудови всіх можливих покриттів є метод перебору всіх підмножин рядків таблиці покриттів: порожня множина рядків, підмножини з одного рядка, з двох і т. д., … з усіх рядків. Потрібно розглядати 2^m підмножин. Отже, їх можна подати у вигляді такого переліку підмножин: $\{\emptyset,\ \{A_1\},\ \{A_2\}, \dots, \{A_m\},\ \{A_1, A_2\}, \dots, \{A_1, A_m\}, \dots, \{A_2, A_3\}, \dots$,

$$\{A_2, A_m\}, \{A_3, A_4\}, \{A_{m-1}, A_m\}, \{A_1, A_2, A_3\}, \dots, \{A_1, \dots, A_m\}\} = D$$

У прикладі (С. 32) доведеться розглянути $2^7 = 128$ підмножин.
Розглянемо більш просту таблицю покриттів.

Приклад.

B	b_1	b_2	b_3	b_4	b_5
A_1	1		1		1
A_2	1	1		1	
A_3	1			1	1
A_4		1	1		

Рисунок 3.3

Тут D – це множина всіх підмножин, які потрібно розглядати:

$$D = \{\{\emptyset\}, \{A_1\}, \{A_2\}, \{A_3\}, \{A_4\}, \{A_1, A_2\}, \{A_1, A_3\}, \{A_1, A_4\}, \{A_2, A_3\}, \{A_2, A_4\},$$
$$\{A_3, A_4\}, \{A_1, A_2, A_3\}, \{A_1, A_2, A_4\}, \{A_1, A_3, A_4\}, \{A_2, A_3, A_4\}, \{A_1, A_2, A_3, A_4\}\}.$$

А саме: підмножина ∅, очевидно, не є покриттям, підмножина $\{A_1\}$ є покриттям? Тобто, чи є у рядку A_1 всі одиниці? Ні, тоді розглядаємо $\{A_2\}$ – не є покриттям, $\{A_3\}$ – не є покриттям, $\{A_4\}$ – не є покриттям, $\{A_1, A_2\}$ – покриття, сукупність цих двох рядків містить одиниці в усіх стовпцях таблиці покриття; $\{A_1, A_3\}$ – немає одиниці в стовпці b_2; $\{A_1, A_4\}$ – немає одиниці в b_4; $\{A_2, A_3\}$ – немає одиниці в b_3; $\{A_2, A_4\}$ – немає одиниці в b_5; $\{A_3, A_4\}$ – покриття; $\{A_1, A_2, A_3\}$ – покриття, причому надлишкове, оскільки поглинає покриття $\{A_1, A_2\}$; $\{A_1, A_2, A_4\}$ – надлишкове покриття; $\{A_1, A_3, A_4\}$ – надлишкове покриття; $\{A_2, A_3, A_4\}$ – надлишкове покриття; $\{A_1, A_2, A_3, A_4\}$ – надлишкове покриття.

Очевидно, що для більш складних випадків подати множину всіх підмножин буде важко. Для спрощення процесу розв'язання пропонується поданий нижче алгоритм перебору.

Будемо вважати, що підмножини A_i пронумеровані $A_1, \dots, A_m$ і упорядковані. Основна ідея алгоритму полягає у тому, що спочатку будуються всі підмножини D_i, які містять A_1, далі – ті, що містять A_2, але не містять A_1, якщо побудована підмножина D_1, то за нею будуються підмножини D_{i+j}, які містять повністю D_i $\left(D_i \subset D_{i+j}\right)$. Сформулюємо такий алгоритм:

0. Поточна множина $D = \{A_1\}$, $i = 0$.

1. $i = i + 1$. Запам'ятовуємо множину D як чергову побудовану множину D_i.

2. Знаходимо найбільший номер j елемента $A_j \in D$. Якщо $j \neq n$, то переходимо до пункту 3 алгоритму. Якщо $j = n$, то видаляємо A_n з D і, якщо $D = \emptyset$, то закінчуємо побудову, якщо $D \neq \emptyset$, то знаходимо

найбільший номер j елемента $A_j \in D$ і видаляємо A_j із D.

3. $j = j + 1$. Вводимо в D елемент A_j. Переходимо до пункту 1.

Виконання алгоритму розглянемо для попереднього прикладу (див. рис. 3.3).

$D_1 = \{A_1\};\ D_2 = \{A_1, A_2\};\ D_3 = \{A_1, A_2, A_3\};\ D_4 = \{A_1, A_2, A_3, A_4\};$
$D_5 = \{A_1, A_2, A_4\};\ D_6 = \{A_1, A_3\};\ D_7 = \{A_1, A_3, A_4\};\ D_8 = \{A_1, A_4\};$
$D_9 = \{A_2\};\ D_{10} = \{A_2, A_3\};\ D_{11} = \{A_2, A_3, A_4\};\ D_{12} = \{A_2, A_4\};\ D_{13} = \{A_3\};$
$D_{14} = \{A_3, A_4\};\ D_{15} = \{A_4\}.$

3.3.2 Метод граничного перебору

У багатьох випадках достатньо знаходити лише безнадлишкові покриття (теорема 3.2). Для цього потрібно змінити алгоритм так, щоб не будувати надлишкові покриття. Однак не можна знайти такий простий і ефективний алгоритм, який не потребував би побудови усіх надлишкових покриттів. У кращому випадку вдається лише зменшити їхню кількість. Основа зміни алгоритму полягає у тому, що коли поточна підмножина є покриттям, то немає потреби вводити у неї нові елементи. Сформулюємо *алгоритм граничного перебору* формування підмножин D_i.

0. Поточна множина $D = \{A_i\}$, $i = 0$. Переходимо до пункту 4.

1. Знаходимо найбільший номер j елемента $A_j \in D$. Аналізуємо ситуації:

а) якщо $j \neq m$ і D не покриття, то переходимо до пункту 3 алгоритму.

б) якщо $j \neq m$ і D є покриття, то переходимо до пункту 2 алгоритму.

в) якщо $j = m$, то видаляємо A_m із D і якщо $D = \emptyset$, то переходимо до пункту 5 алгоритму. Інакше – знову знаходимо найбільший номер j елемента в D.

2. Видаляємо A_j із D.

3. $j = j + 1$. Вводимо елемент A_j у D.

4. Визначаємо, чи є поточна побудова покриттям. Якщо ні, то переходимо до пункту 1 алгоритму. Інакше – по-перше, видаляємо з раніше побудованих покриттів ті, які поглинають D (надлишкові покриття), відповідно зменшуючи значення i. По-друге, запам'ятовуємо D як покриття P_i. $(i = i + 1;\ P_i = D)$. Переходимо до пункту 1 алгоритму.

5. Закінчуємо побудову всіх безнадлишкових покриттів.

Для попереднього прикладу (див. рис. 3.3) за даним алгоритмом реалізується така послідовність визначення підмножин D (підмножини D, які є покриттям, підкреслені однією рискою, а безнадлишкові покриття – двома рисками):

$D = \{A_1\};\ \underline{D_1 = \{A_1, A_2\}};\ D = \{A_1, A_3\};\ \underline{D_2 = \{A_1, A_3, A_4\}};\ D = \{A_1, A_4\};$
$D = \{A_2\};\ D = \{A_2, A_3\};\ \underline{D_3 = \{A_2, A_3, A_4\}};\ D = \{A_2, A_4\};$
$D = \{A_3\};\ \underline{D_4 = \{A_3, A_4\}};\ D = \{A_4\}.$

Таким чином, на відміну від попереднього алгоритму, за методом граничного перебору з п’яти можливих надлишкових покриттів отримано лише два.

Розглянутий алгоритм граничного перебору простий та універсальний. Він може бути застосований для будь-яких задач, де можливі рішення складають увігнуту множину. Однак скорочення варіантів перебору за цим алгоритмом є недостатнім. Інші методи дозволяють ще більше скоротити час вирішення задачі.

3.3.3 Метод мінімального стовпця – максимального рядка

Пошук покриття, що є наближеним до найкоротшого, дає такий простий алгоритм перетворення таблиці покриття.

1. Вихідна таблиця вважається поточною перетвореною таблицею покриттів, множина рядків покриття – порожня.

2. У поточної таблиці виділяється стовпець з найменшим числом одиниць. Серед рядків, що містять одиниці в цьому стовпці, виділяється один з найбільшим числом одиниць. Цей рядок вноситься в покриття, розглядувана таблиця скорочується викреслюванням усіх стовпців, у яких обраний рядок має одиниці. Якщо в таблиці є не викреслені стовпці, то виконується пункт 2, інакше – покриття побудоване.

Відзначимо, що, підраховуючи числа одиниць у рядку, враховуються лише одиниці в не викреслених стовпцях.

Приклади наближеного розв’язання задачі. Розв’язання прикладу (див. рис. 3.3) зводиться до вибору мінімального стовпця b_2 та максимального рядка A_2, що має три одиниці. Після викреслювання покритих рядком A_2 стовпців, залишаються лише стовпці b_3 та b_5, які покриваються рядком A_1. Отже, отримано покриття – $\{A_1, A_2\}$.

Для прикладу, наведеному на рис. 3.1, процес розв’язання зображено на рис. 3.4. Цифрами відзначена послідовність вибору стовпців і рядків. Цифри біля одиниць вказують кількість одиниць у рядку, скороченому після розгляду таблиці.

		1)	2)			3)				
	B	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
1)	A_1	1_4!		1			1	1		
2)	A_2	1_4	1_3!		1				1	
	A_3		1_2			1_1		1		
	A_4			1					1	1
	A_5				1	1_2!			1	1
	A_6			1		1_2	1	1		1
	A_7		1_2		1		1			

Рисунок 3.4

Отриманий розв'язок $\{A_1, A_2, A_5\}$ виявився на один рядок більшим за найкоротше покриття. Зокрема відзначимо, що якби за першим розглянутим стовпцем b_1, був обраний рядок A_2, що має таку ж саму кількість одиниць як A_1, то покриття було б найкоротшим.

3.3.4 Метод ядерних рядків

Розглянемо теореми про скорочення таблиці покриттів. Зрозуміло, що обсяг перебору суттєво залежить від розмірів таблиці покриттів, а тому способи викреслювання деяких рядків і стовпців таблиці покриттів мають істотне значення для скорочення перебору.

Теорема 3.3 (про ядро). Якщо в деякому стовпці таблиці покриттів є лише одна одиниця, то рядок, що містить цю одиницю, входить у всі покриття. Цей рядок називається ***ядерним***.

Множина ядерних рядків видаляється и запам'ятовується для введення в усі покриття. Ядерні рядки з таблиці видаляються й викреслюються всі покриті ними стовпці, тобто викреслюються всі стовпці, в яких є одиниці в ядерних рядках.

Теорема 3.4 (про антиядро). Якщо після видалення ядерних рядків і покритих ними стовпців в таблиці покриттів в деякому рядку не залишається одиниці, то цей рядок не входить в жодне ненадлишкове покриття. Такі рядки називаються ***антиядерними*** і вони викреслюються з таблиці без запам'ятовування.

Перед тим як сформулювати подальші теореми, розглянемо поняття поглинання вектора. Вектор $\boldsymbol{E}$ поглинає вектор $\boldsymbol{F}$ ($\boldsymbol{E} \geq \boldsymbol{F}$), якщо для всіх компонент цих векторів e_i, f_i одночасно виконується умова: $e_i \geq f_i$.

Приклад. Нехай $\boldsymbol{E}$ = 1101, $\boldsymbol{F}$ = 0101, $\boldsymbol{S}$ = 1011, $\boldsymbol{T}$ = 1111.

Легко побачити, що $\boldsymbol{E} \geq \boldsymbol{F}$, $\boldsymbol{E} \leq \boldsymbol{T}$, а вектори $\boldsymbol{E}$ і $\boldsymbol{S}$ та $\boldsymbol{F}$ і $\boldsymbol{S}$ – непорівнювані.

Отже, $\boldsymbol{E}$ – вектор, що поглинає вектор $\boldsymbol{F}$.

$\boldsymbol{E}$ – вектор, що поглинається вектором $\boldsymbol{T}$.

Теорема 3.5 (про поглинальні стовпці). У таблиці покриттів можуть бути викреслені усі стовпці (розглядаються як вектори), що поглинають інші стовпці без втрат для побудови усіх ненадлишкових покриттів.

Теорема 3.6 (про рядки, поглинуті у процесі побудови одного найкоротшого покриття). Якщо у процесі розв'язання задачі про покриття достатньо гарантувати побудову хоча б одного найкоротшого покриття, то можна видалити всі рядки, що поглинаються.

Теорема 3.7 (про рядки, які поглинаються, у процесі побудови мінімальних покриттів). Якщо, розв'язуючи задачу про покриття,

достатньо гарантувати побудову всіх (або хоча б одного) мінімальних покриттів, то можна викреслювати рядки, які поглинаються, якщо їхня ціна більше (або дорівнює) ціні рядків, що їх поглинають.

Розглянемо використання згаданих теорем на прикладах. Таблиця покриттів, що наведена на рис. 3.1 не спрощується: в ній немає ні ядерних, ні антиядерних рядків, немає стовпців що поглинають, та рядків, які поглинаються. Таблиця покриттів (див. рис. 3.3) спрощується дуже незначно: в ній лише стовпець b_1 можна викреслити як такий, що поглинає, оскільки виконуються нерівності $b_1 \geq b_4$ та $b_1 \geq b_5$.

Розглянемо інформативніший приклад, що наведений на рис. 3.5. Насамперед відзначаємо, що у стовпцях b_3 та b_4 міститься лише по одній одиниці, а тому рядки A_3 та A_5, що містять ці одиниці, є ядерними. Ці рядки запам'ятовуємо для введення в усі ненадлишкові покриття цієї таблиці. Скорочуємо таблицю покриттів за рахунок видалення ядерних рядків (A_3, A_5) та всіх покритих ними стовпців b_3, b_4, b_5, b_8 (рис. 3.6). Зауважимо, що в такому разі таблицю не обов'язково перекреслювати, а достатньо лише викреслити відповідні рядки і стовпці безпосередньо на рис. 3.5.

Після викреслювання відзначених вище рядків та стовпців (рис. 3.6) помічаємо, що рядок A_6 не містить одиниць. Отже, він є антиядерним, і його потрібно викреслити з таблиці. Також бачимо, що стовпець b_7 поглинає b_2 ($b_7 \geq b_2$), а тому стовпець b_7 теж викреслюємо.

Перед подальшими спрощеннями перепишемо таблицю, що залишилася, ще раз (рис. 3.7). На рис. 3.7 відзначаємо, що рядок A_2 поглинається рядком A_1 ($A_2 \leq A_1$). Можливі декілька розгалужень процесу розв'язання:

1. Якщо потрібно побудувати всі ненадлишкові покриття, то подальші скорочення рядків не виконуються, й таблиця (рис. 3.7) більше не спрощується;

2. Якщо потрібно побудувати мінімальні покриття, то, згідно з теоремою 3.7 потрібно звернути увагу на ціну a_i рядків. Оскільки $A_2 \leq A_1$ але водночас $a_2 < a_1$, то рядок A_2 не можна викреслювати, і тому таблиця (рис. 3.7) також не спрощується;

3. Якщо потрібно побудувати лише одне найкоротше покриття, то рядок A_2 як такий, що поглинається, можна викреслити. Тоді рядок A_1 стає ядерним, тому що в стовпці b_2 залишається єдина одиниця. Рядок A_1 додається до множини ядерних рядків – $\{A_1, A_3, A_5\}$. Таблиця покриттів спрощується: викреслюється рядок A_1 та покриті ним стовпці b_2 та b_6 Таблиця, що залишилася (рис. 3.8), далі не спрощується.

B	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	a_i	*Познач.*
A_1		1				1	1				3	
A_2		1			1		1				2	
A_3			1		1			1			2	ядерний
A_4	1									1	3	
A_5				1				1			3	ядерний
A_6					1			1			3	
A_7	1				1		1		1		2	
A_8						1			1	1	1	

Рисунок 3.5

B	b_1	b_2	b_6	b_7	b_9	b_{10}	a_i	*Познач.*
A_1		1	1	1			3	
A_2		1		1			2	
A_4	1					1	3	$\{A_3, A_5\}$
A_6							3	
A_7	1			1	1		2	
A_8			1		1	1	1	

Рисунок 3.6

B	b_1	b_2	b_6	b_9	b_{10}	a_i	*Познач.*
A_1		1	1			3	
A_2		1				2	
A_4	1				1	3	$\{A_3, A_5\}$
A_7	1			1		2	
A_8			1	1	1	1	

Рисунок 3.7

B	b_1	b_9	b_{10}	a_i	*Познач.*
A_4	1		1	3	
A_7	1	1		2	$\{A_1, A_3, A_5\}$
A_8		1	1	1	

Рисунок 3.8

Розглянемо узагальнений алгоритм скорочення таблиці покриттів. Використовуючи теореми 3.3–3.7, можна спростити таблицю покриттів, запам’ятавши ядерні рядки. В такому випадку можливі два варіанти процесу розв’язання:

1. Таблиця покриттів після спрощень стає порожньою – викреслені всі стовпці. Отже, множина ядерних рядків є шуканим покриття;

2. Після застосування теорем 3.3–3.7 отриманий залишок таблиці покриттів більше не спрощується у разі повторного застосування окреслених вище прийомів. Тобто, залишок таблиці покриттів є циклічним. Покриття для циклічного залишку таблиці можна будувати лише з використанням методів перебору покриттів: граничним перебором або розкладанням за стовпцем. Після того як до ядерних рядків додадуться рядки покриття циклічного залишку, буде отримано покриття початкової таблиці.

Сформулюємо алгоритм побудови циклічного залишку таблиці покриттів і множини ядерних рядків для випадку побудови одного найкоротшого покриття.

0. Приймаємо початкову таблицю покриттів як поточну таблицю покриттів, а множину ядерних рядків – як порожню.

1. Визначаємо ядерні рядки та запам’ятовуємо їхню множину. Поточну таблицю покриттів скорочуємо: викреслюємо ядерні рядки та всі стовпці, покриті ними.

2. Викреслюємо антиядерні рядки.

3. Викреслюємо стовпці, що поглинають інші стовпці.

4. Викреслюємо рядки, що поглинаються іншими рядками.

5. Якщо в результаті виконання пунктів 1–4 поточна таблиця покриттів змінилася, повторно виконуємо пункт 1, в протилежному випадку – перетворення таблиці закінчуємо.

Зверніть увагу, що в результаті виконання алгоритму буде отримано два результати – множину ядерних рядків та циклічний залишок таблиці покриттів. Таблиця покриттів прикладу (див. рис. 3.1) – циклічна, множина ядерних рядків – порожня. Для прикладу (див. рис. 3.5) множина ядерних рядків містить три елементи – $\{A_1, A_3, A_5\}$; циклічний залишок у разі побудови одного найкоротшого покриття наведений на рис. 3.8.

Скорочуючи таблицю покриттів до циклічної у разі побудови одного мінімального покриття зміниться лише п. 4 алгоритму, який формулюється так: викреслюються всі рядки, що поглинаються, за умови, що їхні ціни не менше за ціну відповідних рядків, що їх поглинають.

У такому випадку для прикладу (див. рис. 3.5) множиною ядерних рядків є $\{A_3, A_5\}$, а циклічний залишок таблиці покриттів наведений на рис. 3.7.

Скорочуючи таблицю покриттів до циклічної, за умови побудови всіх ненадлишкових покриттів, п. 4 алгоритму не виконується: в цьому випадку не можна викреслювати жодного рядка, що поглинається. Відповідь буде такою ж, що й у разі побудови одного мінімального покриття.

Питання для самоконтролю

1) Сформулюйте задачу про покриття мовою теорії множин.
2) Яке покриття називається ненадлишковим?
3) Дайте означення мінімального покриття.
4) Дайте означення найкоротшого покриття.
5) Що є умовою ненадлишкового покриття?
6) Наведіть приклади технічного застосування задачі про покриття.
7) Скільки підмножин аналізується в методі повного перебору?
8) В чому зміст алгоритмів перебору?

Індивідуальні практичні завдання №3

Задача. Для заданої таблиці покриттів побудувати мінімальне та найкоротше покриття.

B1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A	1						1			1
B			1		1				1	1
C		1		1					1	2
D		1					1	1		1
E	1		1		1			1		2
F	1		1			1	1			3
G		1		1	1			1	1	3
H	1			1		1				2

B2	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A		1	1				1	1		3
B		1	1	1	1					2
C						1		1	1	2
D	1			1	1	1			1	3
E	1		1	1						2
F	1	1				1				1
G					1		1		1	2
H				1		1				1

B3	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A						1	1		1	2
B		1	1		1	1	1			3
C					1	1		1		1
D	1			1				1	1	3
E	1	1	1	1						2
F			1	1			1			1
G	1	1			1					1
H		1				1				1

B4	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A	1	1				1				1
B		1	1	1	1					2
C						1	1	1		1
D		1	1				1		1	3
E	1		1	1						2
F	1			1	1	1		1		3
G					1			1	1	1
H	1					1				1

B5	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A		1			1			1		1
B		1				1	1		1	3
C	1		1	1	1			1		3
D			1	1			1			2
E	1		1			1	1			2
F	1			1		1				1
G					1			1	1	1
H			1					1		1

B6	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A	1				1				1	1
B						1	1	1	1	2
C			1			1	1			1
D		1	1					1		2
E		1	1		1		1		1	3
F	1			1		1				3
G		1		1	1					2
H		1						1		1

B7	1	2	3	4	5	6	7	8	9	*a*
A			1	1			1			1
B	1				1		1			2
C		1	1	1					1	2
D		1		1	1		1	1		4
E		1				1		1		2
F	1		1			1			1	3
G					1			1	1	1
H		1		1					1	4

B8	1	2	3	4	5	6	7	8	9	*a*
A				1	1		1			1
B	1					1			1	2
C	1			1	1			1	1	3
D			1					1	1	1
E		1	1			1	1			3
F	1	1		1		1	1			2
G		1			1			1		2
H				1			1			1

B9	1	2	3	4	5	6	7	8	9	*a*
A		1	1		1			1		3
B	1		1	1				1		2
C	1			1		1	1		1	4
D		1				1			1	1
E			1	1					1	2
F	1						1	1		2
G					1	1	1			1
H		1			1					3

B10	1	2	3	4	5	6	7	8	9	*a*
A		1		1				1		2
B			1			1			1	1
C		1			1		1		1	3
D	1			1			1		1	2
E	1		1		1					1
F	1		1	1		1		1		3
G		1						1		2
H							1	1	1	2

B11	1	2	3	4	5	6	7	8	9	*a*
A		1					1	1		1
B						1		1	1	2
C		1	1		1	1				3
D			1	1					1	2
E		1						1		1
F	1				1		1			1
G	1		1	1		1				2
H	1			1			1	1	1	2

B12	1	2	3	4	5	6	7	8	9	*a*
A	1			1		1	1		1	3
B	1				1		1		1	1
C		1				1	1			2
D		1	1		1			1		3
E	1		1	1	1					2
F		1					1			2
G	1					1		1		2
H			1	1					1	1

B13	1	2	3	4	5	6	7	8	9	*a*
A			1					1		1
B		1		1		1				2
C	1	1		1	1			1		3
D	1			1		1			1	2
E	1				1		1			1
F		1			1				1	2
G		1				1				2
H			1			1	1		1	3

B14	1	2	3	4	5	6	7	8	9	*a*
A		1				1		1		1
B		1		1			1		1	3
C				1	1			1	1	2
D			1			1			1	2
E		1		1						1
F					1		1			1
G	1		1	1						2
H	1		1		1	1		1		3

B15	1	2	3	4	5	6	7	8	9	*a*
A	1			1					1	2
B	1						1		1	1
C	1		1			1		1		3
D					1		1	1		2
E		1		1	1		1		1	4
F		1	1	1						1
G	1				1	1				1
H					1		1			2

B16	1	2	3	4	5	6	7	8	9	*a*
A	1	1						1		1
B		1		1	1		1	1		4
C		1		1		1				2
D					1	1	1			1
E			1				1	1		1
F	1		1			1			1	3
G				1	1				1	2
H		1					1	1		3

B17	1	2	3	4	5	6	7	8	9	*a*
A			1		1				1	1
B		1	1			1	1			3
C		1	1	1					1	2
D	1			1	1			1	1	3
E					1	1		1		1
F	1	1		1		1				2
G	1						1	1		2
H	1								1	1

B18	1	2	3	4	5	6	7	8	9	*a*
A					1		1	1		2
B	1			1		1				1
C	1	1	1							1
D		1			1	1			1	3
E			1		1		1		1	2
F	1		1	1	1			1		4
G				1				1	1	1
H	1		1							1

B19	1	2	3	4	5	6	7	8	9	*a*
A			1		1				1	1
B		1	1	1				1		3
C			1	1			1		1	3
D	1				1	1	1		1	4
E					1	1		1		2
F	1	1					1			1
G	1			1		1				1
H			1						1	1

B20	1	2	3	4	5	6	7	8	9	*a*
A	1	1					1			2
B				1	1				1	2
C					1	1	1	1		2
D		1				1			1	1
E	1		1		1	1				3
F			1	1				1		1
G	1			1			1	1	1	4
H			1				1			1

B21	1	2	3	4	5	6	7	8	9	*a*
A	1			1	1		1		1	4
B	1		1			1	1			2
C	1			1				1		1
D			1				1		1	1
E				1	1	1				2
F		1	1			1		1		3
G	1			1			1	1		3
H		1			1				1	1

B22	1	2	3	4	5	6	7	8	9	*a*
A		1			1	1				2
B	1	1		1						1
C	1	1			1			1	1	4
D			1					1	1	2
E	1					1	1	1		2
F					1		1		1	1
G	1			1						1
H			1	1		1	1			3

B23	1	2	3	4	5	6	7	8	9	*a*
A		1	1	1			1	1		3
B			1	1	1					1
C			1				1		1	2
D	1	1		1						1
E		1				1		1		1
F					1	1	1	1		2
G	1				1	1			1	3
H			1				1			1

B24	1	2	3	4	5	6	7	8	9	*a*
A		1			1	1	1			3
B		1						1	1	1
C	1		1				1			2
D			1		1	1			1	2
E	1		1	1				1	1	3
F	1			1		1				1
G				1	1			1		1
H			1				1			1

B25	1	2	3	4	5	6	7	8	9	*a*
A				1			1			1
B	1			1					1	2
C			1			1		1		1
D			1	1	1	1			1	3
E		1		1	1		1			2
F	1	1					1	1		3
G			1				1		1	1
H		1				1	1			1

B26	1	2	3	4	5	6	7	8	9	*a*
A			1	1			1			1
B		1	1		1				1	2
C					1	1		1		2
D	1			1	1				1	3
E							1	1	1	1
F	1	1				1				1
G		1	1			1	1	1		3
H		1			1					1

B27	1	2	3	4	5	6	7	8	9	*a*
A	1			1				1		1
B		1	1			1	1			2
C	1				1				1	1
D			1		1	1				2
E	1			1	1	1	1			3
F		1	1					1	1	3
G			1			1				1
H		1		1			1			1

B28	1	2	3	4	5	6	7	8	9	*a*
A	1				1					1
B				1			1	1	1	3
C		1	1						1	2
D	1	1	1		1	1				3
E			1	1	1			1		2
F	1					1	1			1
G	1			1	1					1
H		1				1		1		2

B29	1	2	3	4	5	6	7	8	9	*a*
A					1		1		1	1
B	1		1			1			1	3
C			1	1		1				2
D	1	1					1			1
E		1	1					1		2
F		1		1	1		1	1		3
G					1	1		1		1
H		1					1			1

B30	1	2	3	4	5	6	7	8	9	*a*
A			1			1			1	2
B	1				1		1	1		2
C		1	1	1						1
D	1			1	1				1	3
E					1	1		1		2
F	1	1					1			1
G		1	1			1	1	1		3
H	1							1		1

Тема 4. КОМБІНАТОРИКА

4.1 Основні властивості комбінаторних наборів
4.2 Основні типи наборів комбінаторики (розміщення, сполучення, перестановки)
4.3 Прийоми розв'язання комбінаторних задач

4.1 Основні властивості комбінаторних наборів

Означення:	***Комбінаторика*** – розділ дискретної математики, пов'язаний з побудовою, вивченням властивостей і розрахунком кількості наборів елементів кінцевих множин, що використовуються у процесі розв'язання задач дискретної математики.

Основні властивості наборів

Для кожної задачі ці властивості специфічні, однак виділяють три основні властивості наборів, які потрібно враховувати у всіх задачах:

- кількість елементів,
- можливість упорядкування елементів,
- можливість повторення елементів.

Залежно від комбінації цих основних властивостей виділяють шість основних типів наборів комбінаторики (рис. 4.1):

1) Розміщення (***A****rrangement*) без повторень,
2) Розміщення з повтореннями,
3) Сполучення (***C****ombination*) без повторень,
4) Сполучення з повтореннями,
5) Перестановки (***P****ermutation*) без повторень,
6) Перестановки з повтореннями,

Проводячи комбінаторні розрахунки використовують два «золотих» правила комбінаторики: 1) правило добутку та 2) правило суми.

Правило добутку (перше основне правило комбінаторики).

Приклад. З гуртожитка № 3 ВНТУ до центрального автовокзалу можна доїхати трьома способами: трамвай, тролейбус, маршрутка. А з автовокзалу до зупинки «Меморіал визволення» можна доїхати двома способами: тролейбус, маршрутка. Скількома способами можна доїхати з гуртожитка № 3 ВНТУ до зупинки «Меморіал визволення»?

Розв'язання задачі, очевидно, зводиться до підрахунку числа елементів декартового добутку двох множин:

{Трам, Трол, М} × {Трол, М } = {(Трам, Трол), (Трол, Трол), (М, Трол), (Трам, М), (Трол, М), (М, М)}.

Кількість варіантів $3 \times 2 = 6$.

За цією простою задачею стоїть перше основне правило комбінаторики: *правило добутку*.

Правило добутку:	Нехай потрібно послідовно виконати k дій. Якщо 1-у дію можна виконати n_1 способами, 2-у – n_2 способами, і т. д. до k-ї дії, яку можна виконати n_k способами, то всі n дій можна виконати $n_1 \cdot n_2 \cdot \ldots \cdot n_k$ способами.

Альтернативне формулювання:	Якщо деякий вибір A можна виконати n способами, а для кожного з них деякий другий вибір B можна виконати m способами, то вибір A і B (в заданому порядку) можна виконати $n \cdot m$ способами.

ОСНОВНІ ТИПИ НАБОРІВ КОМБІНАТОРИКИ	Обираються не всі Елементи	Враховується порядок	З повтором	⇨	Розміщення з повтореннями: $\overline{A_n^k} = n^k$
			Без повтору	⇨	Розміщення: $A_n^k = \frac{n!}{(n-k)!}$
		Не враховується порядок	З повтором	⇨	Комбінації з повтореннями: $\bar{C}_n^k = C_{n+k-1}^k$
			Без повтору	⇨	Комбінації: $C_n^k = \frac{n!}{k! \cdot (n-k)!}$.
	Обираються всі елементи	Враховується порядок	З повтором	⇨	Перестановки з повтореннями: $\bar{P}_{m=\sum_{i=1}^{n} n_i} = \frac{m!}{n_1! \cdot n_2! \cdot \ldots \cdot n_n!}$
			Без повтору	⇨	Перестановки: $P_n = n!$

Рисунок 4.1 – Основні типи наборів комбінаторики

Приклад. Скількома способами на першості світу з футболу можуть розподілитися медалі, якщо у фінальній частині грають 24 команди?

Розв'язання. Золоту медаль може одержати будь-яка з 24 команд. Срібну медаль може одержати вже одна з 23 команд, що залишились. Бронзову медаль може одержати вже одна з 22 команд, що залишились.

За правилом добутку загальне число способів $24 \cdot 23 \cdot 22 = 12144$.

Правило суми (друге основне правило комбінаторики)

Правило суми:	Нехай $N(A)$ – потужність множини A, тоді для визначення $N(A \cup B \cup C)$ використовується правило включення та виключення: $$N(A \cup B \cup C) = N(A) + N(B) + N(C) - $$ $$-\big(N(A \cap B) + N(A \cap C) + N(B \cap C)\big) + N(A \cap B \cap C)$$

Якщо множини A, B, C не перетинаються, то, очевидно, формула включення і виключення значно спрощується

$$N(A \cup B \cup C) = N(A) + N(B) + N(C), \text{ якщо } (A \cap B = A \cap C = B \cap C = \emptyset)$$

Приклад. З пункту *А* в пункт *Б* курсують 5 потягів, 3 автобуси та 1 літак. Скількома способами можна добратися з пункту *А* в пункт *Б*?

Розв'язання. Оскільки події незалежні, то кількість варіантів $5 + 3 + 1 = 9$.

Приклад. Кожний здобувач освіти групи або брюнет, або цікавиться математикою, або займається спортом. У групі 20 брюнетів, з них 10 цікавляться математикою; брюнет, який цікавиться математикою та займається спортом, всього один. З 16 спортсменів групи – 10 брюнетів. Всього цікавляться математикою 15 здобувачів освіти, з них тільки 6 – спортсмени. Скільки здобувачів освіти в групі?

Розв'язання. Нехай *Б* – множина брюнетів, *М* – множина математиків, *С* – множина спортсменів.

Тоді,

$$|\text{Б} \cup \text{М} \cup \text{С}| = |\text{Б}| + |\text{М}| + |\text{С}| - |\text{Б} \cap \text{М}| - |\text{Б} \cap \text{С}| - |\text{М} \cap \text{С}| + |\text{Б} \cap \text{М} \cap \text{С}| =$$
$$= 20 + 15 + 16 - 10 - 10 - 6 + 1 = 26.$$

4.2 Основні типи наборів комбінаторики

На рис. 4.1 зображена класифікація та наведені формули розрахунку числа наборів для кожного з основних типів наборів комбінаторики.

Означення:	Набір, що містить k різних елементів, вибраних із n-елементної множини, з урахуванням порядку розташування цих елементів, називається ***розміщенням*** без повторень.

Перший елемент може бути вибраний n способами, оскільки повторення не дозволені, другий елемент може бути вибраний $(n - 1)$ способами, 3-й елемент – $(n - 2)$ способами, k-й елемент – $(n - k - 1)$ способами. За правилом добутку

$$A_n^k = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-k+1) = \frac{n!}{(n-k)!}.$$

Розміщення

$$A_n^k = \frac{n!}{(n-k)!}.$$

Приклад. У фінальній частині змагань беруть участь 10 команд. Скільки існує варіантів трійки призерів?

Розв'язання Перше місце займає одна команда з 10, друге місце – одна команда з 9, третє місце – одна команда з 8.

Отже, всього є $10 \cdot 9 \cdot 8 = 720$ варіантів.

$$A_{10}^3 = \frac{10!}{(10-3)!} = \frac{10!}{7!} = 10 \cdot 9 \cdot 8 = 720.$$

Означення:	Набір, що містить k елементів n-елементної множини, побудований з урахуванням порядку розташування елементів і з дозволеним повторенням елементів, називається ***розміщенням з повторенням***.

Перший елемент може бути вибраний n способами, інші, оскільки повторення дозволені, теж n варіантами:

$$\overline{A_n^k} = n \cdot n \cdot \ldots \cdot n = n^k.$$

Розміщення з повторенням

$$\overline{A_n^k} = n^k.$$

Приклад. Скільки існує варіантів чотиризначного пін-коду для пластикової картки?

Розв'язання. Перша цифра – 10 способів, друга цифра – 10 способів, третя цифра – 10 способів, четверта цифра – 10 способів.

Отже, всього є $10 \cdot 10 \cdot 10 \cdot 10 = 10^4$ способів.

Означення:	Набір, що містить всі n елементів n-елементної множини, побудований з урахуванням порядку розташування елементів і без повторення елементів, називається ***перестановкою***.

Очевидно, що

$$P_n = A_n^n = \frac{n!}{(n-n)!} = \frac{n!}{0!} = n!$$

Зауваження. 0! = 1 за означенням.

Перестановка

$$P_n = n!$$

Приклад. У фінальній частині змагань беруть участь 4 команди: А, Б, В і Г. Скільки існує варіантів розподілу місць між ними?

Розв'язання. АБВГ, АБГВ, АВБГ, АВГБ, АГБВ, АГВБ, БАВГ, БАГВ, БВАГ, БВГА, БГАВ, БГВА, ВАБГ, ВАГБ, ВБАГ, ВБГА, ВГАБ, ВГБА, ГАБВ, ГАВБ, ГБАВ, ГБВА, ГВАБ, ГВБА. Всього 24 = 4! варіантів.

Означення:	Набір, що складається з k будь-яких елементів n-елементної множини без урахування порядку їх розташування, називається ***сполученням (поєднанням, комбінацією)***. Інакше, поєднання – це k-елементна підмножина n-елементної множини.

З кожного сполучення можна побудувати $k!$ розміщень перестановкою елементів, тому

$$C_n^k = \frac{A_n^k}{P_k} = \frac{n!}{k! \cdot (n-k)!}.$$

Сполучення

$$C_n^k = \frac{n!}{k! \cdot (n-k)!}.$$

Приклад. Скільки існує варіантів вибору трьох літер із множини {А, Б, В, Г} ?

Розв'язання. {АБВ}, {АБГ}, {АВГ}, {БВГ}.

$$C_4^3 = \frac{4!}{3! \cdot (4-3)!} = 4.$$

Означення:	Набір, в якому кожен елемент x_i опорної n-елементної множини повторюється n_i разів (i може змінюватися від 1 до n) і у процесі його побудови враховується порядок розташування різних елементів, називається ***перестановкою з повторенням***.

Формула отримана, виходячи з такого міркування. Спочатку всі m елементів вважаємо різними, тоді $P_m = m!$. Потім відмінність для першого елемента прибираємо, а тому перестановки, що відрізняються тільки порядком розташування першого елемента, очевидно, стають непомітними. Перестановок першого елемента $P_{n_1} = n_1!$, тобто,

$$\bar{P}_{m = n_1 + (m - n_1)} = \frac{P_m}{P_{n_1}} = \frac{m!}{n_1!}.$$

Міркуючи аналогічним чином для 2-го, 3-го та решти елементів, отримуємо формулу для перестановок з повтореннями.

Перестановка з повтореннями

$$\bar{P}_{m = \sum_{i=1}^{n} n_i} = \frac{P_m}{P_{n_1}} = \frac{m!}{n_1! \cdot n_2! \cdot \ldots \cdot n_n!}.$$

Приклад. Перестановки, отримані різним розміщенням букв зі слова **МАМА** ($n_{\text{М}} = 2$, $n_{\text{А}} = 2$):

ММАА, МАМА, МААМ, АММА, АМАМ, ААММ.

$$\bar{P}_{4=2+2} = \frac{4!}{2! \cdot 2!} = 6.$$

Приклад. Перестановки з 0 та 1 при $n_0 = 2$, $n_1 = 3$:

00111, 01011, 01101, 01110, 10011, 10101, 10110, 11001, 11010, 11100.

$$\bar{P}_{5=2+3} = \frac{5!}{2! \cdot 3!} = 10.$$

Означення:	Набір, що складається з k елементів n-елементної множини без урахування порядку і з дозволеним повторенням елементів, називається ***сполученням з повтореннями***.

Кожне сполучення з повтореннями повністю визначається, якщо вказано скільки елементів x_i $(1 \le i \le n)$ в нього входить. Поставимо у відповідність набору послідовність нулів і одиниць за таким правилом: послідовно для кожного елемента запишемо стільки одиниць, скільки елементів x_i входить в сполучення, між цими групами одиниць вставляється один нуль.

11101101111

Таким чином в послідовності буде k одиниць і $(n-1)$ нулів. Різні перестановки $n_1 = k$ одиниць і $n_0 = n-1$ нулів визначають різні сполучення з повтореннями

$$\bar{C}_n^k = \bar{P}_{k+n-1} = \frac{(k+n-1)!}{k!\,(n-1)!} = \frac{(n+k-1)!}{k!\,\big((n+k-1)-k\big)!} = C_{n+k-1}^k .$$

Сполучення з повтореннями

$$\bar{C}_n^k = C_{n+k-1}^k .$$

Приклад. Скільки існує варіантів чотирибуквених сполучень з повтореннями на множині {А, М}.

Розв'язання. АААА, АААМ, АМММ, ММММ, ААММ; $n = 2$, $k = 4$.

$$\bar{C}_2^4 = C_{2+4-1}^4 = C_5^4 = 5 .$$

Для обчислення формул комбінаторики можна використовувати таблицю факторіалів.

n	$n!$	n	$n!$	n	$n!$
0	1	6	720	11	39916800
1	1	7	5040	12	479001600
2	2	8	40320	13	6227020800
3	6	9	362880	14	87178291200
4	24	10	36228800	15	1307674367000
5	120				

При великих значеннях n для наближеного обчислення факторіалів можна використовувати формулу Стірлінга:

$$n! \approx \frac{n^n}{e^n}\sqrt{2\pi n}, \quad e \approx 2{,}718\ldots.$$

4.3 Прийоми розв'язання комбінаторних задач

Розв'язання комбінаторних задач – процес творчий, проте існують декілька загальних прийомів, які потрібно вивчити для творчого використання у разі розв'язання конкретних задач. Ці прийоми надалі використовуватимуться при виведенні різних формул, розрахунку числа графів, числа булевих функцій тощо. Розглянемо основні прийоми на класичних прикладах.

Зведення початкового завдання до відомого

Розглянемо цей прийом на прикладі завдання про ящики. Є n ящиків, що вміщують кожен n_i предметів $(1 \le i \le n)$. Всі $m = \sum n_i$ предметів

різні, але мають однаковий об'єм. Знайти число розподілів предметів по ящиках. Для розв'язання завдання розподілимо номери ящиків 1, 2, ... , *n* по предметах: очевидно, кожен номер *i* повторюватиметься n_i разів. Тому отримуємо перестановки $m = \sum n_i$ номерів *i* з повтореннями n_i, тоді їхнє число

$$\bar{P}_{m=\sum_{i=1}^{n} n_i} = \frac{m!}{\prod_{i=1}^{n}(n_i\ !)}.$$

Отже, розв'язуючи задачі про ящики, ми отримуємо відповідь одразу ж, як тільки початкова постановка завдання зведена до відомої. Цей самий прийом фактично був застосований при виведенні формул числа перестановок без повторень, числа сполучень з повтореннями. Проте, переформовування кожного початкового завдання – процес творчий.

Розбиття початкового завдання на декілька відомих взаємодійних. Розглянемо цей прийом на прикладі розрахунку числа *X* можливих варіантів вибору президії на зборах. Присутні *n* осіб, з них потрібно вибрати *k* осіб в президію, зокрема голову і секретаря зборів. Завдання розбиваємо на дві частини – вибір президії (сполучення C_n^k) і вибір серед президії голови і секретаря (розміщення A_k^2). Застосовуючи правило добутку, отримуємо $X = C_n^k \cdot A_k^2$.

Такий самий прийом використовувався й при виведенні формули кількості сполучень і числа перестановок з повтореннями.

Використання формули включення і виключення.

Формула включень і виключень в загальному випадку має такий вигляд:

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i=1}^{n}|A_i| - \sum_{1\le i_1<i_2\le n}^{n} \left|A_{i_1} \cap A_{i_2}\right| + \ ...$$

$$+(-1)^k \sum_{1\le i_1< \ ... <i_k\le n}^{n} \left|A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_k}\right| + \ ... + (-1)^{n-1}|A_1 \cap A_1 \cap ... \cap A_n|$$

Проілюструємо цей прийом на прикладі розрахунку числа *Y* натуральних чисел на відрізку (1, *N*), що не діляться ні на одне з заданих чисел a_1, ..., a_k, де a_i – взаємно прості числа. Нехай A_i – множина натуральних чисел, що не перевищують *N*, і що діляться на *Q*.

Тоді $|A_i| = \left[\frac{N}{a_i}\right]$, де $[x]$ – найбільше ціле, що не перевершує x.

Очевидно, що $A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_5}$ – множина чисел, що діляться на $a_{i_1}, ..., a_{i_5}$ і $\left|A_{i_1} \cap ... \cap A_{i_5}\right| = \left[\frac{N}{a_{i_1}, ..., a_{i_5}}\right]$. За формулою включення і

виключення число X, що ділиться принаймні на одне з чисел a_1, ..., a_k і не більших за N знаходиться за формулою

$$X = \sum_{i=1}^{k} \left[\frac{N}{a_i}\right] + \sum_{1 \le i_1 < i_2 \le k}^{n} \left[\frac{N}{a_{i_1} a_{i_2}}\right] + \sum_{1 \le i_1 < i_2 < i_3 \le k}^{n} \left[\frac{N}{a_{i_1} a_{i_2} a_{i_3}}\right] + \ \ldots$$

$$+ (-1)^{k-1} \left[\frac{N}{a_1 a_2 \ldots a_k}\right].$$

Число $Y = N - X$.

Застосування функцій, що приводять, до виведення формул комбінаторики. Похідна функція – аналітична функція, в якій використовуються сполучення, розміщення тощо. Характерним прикладом є відомий біном Ньютона

$$(1 + x)^n = \sum_{i=0}^{n} C_n^i \, x^i.$$

Проілюструємо використання бінома Ньютона для виведення декількох цікавих формул і одночасно покажемо прийом використання похідних функцій.

1. $x = 1$.

$$(1 + x)^n = \sum_{i=0}^{n} C_n^i \,, \qquad 2^n = \sum_{i=0}^{n} C_n^i.$$

2. $x = -1$.

$$(1 + x)^n = \sum_{i=0}^{n} C_n^i (-1)^i \,,$$

$$0 = \sum_{i \in \text{парн}} C_n^i - \sum_{i \in \text{непарн}} C_n^i, \qquad \sum_{i \in \text{парн}} C_n^i = \sum_{i \in \text{непарн}} C_n^i \ = 2^{n-1}.$$

3. $(1 + x)^n = (1 + x)^m \, (1 + x)^{n-m}$,

$$\sum_{i=0}^{n} C_n^i \, x^i = \left(\sum_{i=0}^{m} C_m^i \, x^i\right)\left(\sum_{i=0}^{n-m} C_{n-m}^i \, x^i\right)$$

Для x^k

$$C_n^k = \sum_{i=0}^{k} \left(C_m^i \cdot C_{n-m}^{k-i}\right).$$

Зокрема, у разі $m = 1$

$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$ – ця формула – відомий трикутник Паскаля.

Ще один окремий випадок маємо, коли $n = 2k,\ m = k$:

$$C_{2k}^k = \sum_{i=0}^{k} C_k^i \cdot C_k^{k-i}, \quad \text{оскільки} \quad C_k^i = C_k^{k-i}, \quad \text{то} \quad C_{2k}^k = \sum_{i=0}^{k} \left(C_k^i\right)^2.$$

Для сполучень з повтореннями використовується функція

$$(1 + x + x^2 + \ \ldots)^n = \sum_{i=0}^{n} C_{n+i-1}^i \, x^i,$$

застосовуючи яку можна отримати формули, де буде з'являтися C_{n+i-1}^i – формула для сполучень з повтореннями. Для виведення формул для похідних функцій виконують різні операції – інтегрування, диференціювання, прирівнювання змінної до констант тощо.

Питання для самоконтролю

1) Наведіть перше правило комбінаторики.
2) Наведіть друге правило комбінаторики.
3) Дайте означення розміщення.
4) Чим відрізняється сполучення від сполучення з повтором?

Індивідуальні практичні завдання № 4

1. З Києва до Чернігова можна дістатися пароплавом, поїздом, автобусом, літаком; із Чернігова до Новгород-Сіверського – пароплавом або автобусом. Скількома способами можна здійснити подорож за маршрутом Київ – Чернігів – Новгород-Сіверський?

2. Скільки чотиризначних чисел можна скласти з цифр 1, 2, 3, 4, 5, якщо:

а) жодна цифра не повторюється більше одного разу;

б) цифри можуть повторюватись;

в) числа мають бути непарними (цифри можуть повторюватися)?

3. В кошику 12 яблук і 10 апельсинів. Брат вибирає яблуко або апельсин, після чого сестра вибирає з фруктів, які залишилися, і яблуко, і апельсин. Скільки можливостей таких виборів? За якого вибору брата сестра має більше можливостей вибору?

4. Є 5 видів конвертів без марок і 4 види марок. Скількома способами можна вибрати конверт і марку для листа?

5. Скількома способами можна вибрати голосну і приголосну у слові «паркет»?

6. Скількома способами можна вказати на шаховій дошці два квадрати – білий та чорний? Розв'яжіть цю задачу, якщо немає обмежень на колір квадрата. Розв'яжіть її, якщо треба вибрати 2 білих квадрати.

7. Скількома способами можна вибрати на шаховій дошці білий та чорний квадрати, які не лежать на одній горизонталі або на одній вертикалі?

8. Із 3-х екземплярів підручників алгебри, 7 геометрії і 6 фізики треба вибрати комплект, який містить по одному підручнику кожного предмета. Скількома способами це можна зробити?

9. У колоді 36 карт. Скількома способами можна здати 6 карт так, щоб серед них було 2 дами?

10. Скільки треба взяти елементів, щоб число перестановок, утворених з них, дорівнювало 5040?

11. В одинадцятому класі 35 учнів. Вони обмінялися фотографіями. Скільки всього фотографій було роздано?

12. Скільки різних прямих можна провести через 10 точок площини, з яких ніякі 3 не лежать на одній прямій?

13. Скільки слів можна одержати, переставляючи букви у слові «комбінаторика»?

14. На площині взяті 9 точок, розміщених у вигляді квадрата 3×3. Скільки існує трикутників, у яких одна вершина знаходиться у фіксованій точці А, а дві інші – серед інших 8 точок?

15. Віра, Галя, Діна, Зоя, Іра, Катя, Наташа, Юля відправилися на прогулянку на двох човнах. Менший човен може вмістити не більше 4 дівчат, а більший – не більше 6. Скількома різними способами вони можуть розміститися у човнах ?

16. На музичному факультеті навчається 340 здобувачів освіти. З них грає на баяні 105, на піаніно – 150, на гітарі 110, на баяні та піаніно 60, на баяні та гітарі 40, на піаніно й гітарі 50, на 3 інструментах грає 19 здобувачів освіти. Скільки з них; а) грає хоча б на 1 з указаних інструментів, б) грає лише на 1 з указаних інструментів, в) не грає на названих інструментах?

17. Автомобільний номер складається з 3 букв та 4 цифр. Знайти кількість всіх можливих номерів, якщо використовувати тільки 29 букв українського алфавіту?

18. Скількома способами можна порівну роздати чотирьом гравцям 28 кісток доміно?

19. У 9-му класі 12 навчальних предметів і 6 різних уроків щодня. Скількома способами можна скласти розклад на понеділок ?

20. Яких чисел більше серед першого мільйона: тих, в запису яких зустрічається 1, або тих, в запису яких її немає?

21. За пересилання бандеролі слід сплатити 18 грн. Скількома способами можна сплатити її марками вартістю у 4, 6, 10 грн, якщо два способи, які відрізняються порядком марок, вважаються різними?

22. За круглий стіл сідає n ($n > 2$) людей. Два розміщення за столом будемо вважати такими, що збігаються, якщо кожна людина має одних і тих же сусідів в обох випадках. Скільки існує способів сісти за стіл?

23. Скількома способами можна посадити за круглий стіл n чоловіків і n жінок таким чином, щоб ніякі дві особи однієї статі не сиділи поряд?

24. Скількома способами можна покласти 30 різних фотографій у 5 однакових конвертів так, щоб у кожному конверті було по 6 фотографій?

25. Маємо набір із 16 карток. На чотирьох написана буква А, на чотирьох – Б, на чотирьох – В, на чотирьох – Г. Скільки різних слів можна одержати, вибираючи з набору 4 картки і розміщуючи їх у деякому порядку?

26. Скільки семицифрових чисел можна скласти з цифр 1, 2, 3, якщо цифра 1 повторюється 3 рази, цифра 2 – 2 рази, цифра 3 –2 рази?

27. Скількома способами можуть розміститися у турнірній таблиці 10 футбольних команд, якщо відомо, що ніякі 2 не набрали однакової кількості очок?

28. Скільки чотиризначних чисел можна утворити з цифр 0, 1, 2, 3, не повторюючи їх?

29. На зборах мають виступити 5 людей: А, Б, В, Г, Д. Скількома способами можна їх розмістити у списку ораторів, якщо: 1) Б не має виступати перед А; 2) Б має виступити одразу за А?

30. Скількома способами можна скласти триколірний смугастий прапор, якщо є тканини п’яти різних кольорів? Розв’яжіть ту ж саму задачу за умови, коли одна смуга має бути червоною.

31. Є 8 токарів. Скількома способами можна доручити 3 з них виготовлення трьох різних деталей (за одним видом на кожного)?

32. В профактив факультету обрано 9 людей. З них треба обрати голову, його заступника, секретаря і культпрацівника. Скількома способами це можна зробити?

33. Скількома способами можна опустити 5 листів у 11 поштових скриньок, якщо у кожну з них кидається не більше одного листа?

34. Скільки різних натуральних чисел можна утворити з цифр 0, 1, 2, 3, 4, якщо кожне число містить будь-яку з даних цифр не більше одного разу?

35. Із колоди у 52 карти витягнули 10 карт. У скількох випадках серед цих карт виявиться: а) хоча б один туз; б) рівно один туз; в) не менше двох тузів; г) рівно два тузи?

36. Будівельна фірма формує бригаду з 5 робітників. В організації 20 робітників, в тому числі з них 5 малярів, 4 теслі і 2 штукатури. Скількома способами можна укомплектувати бригаду, щоб вона складалася з робітників усіх спеціальностей по одному?

37. Знайдіть суму чотиризначних чисел, які одержуються за допомогою різних перестановок цифр 1, 1, 4, 4. Те ж саме для 0, 0, 4, 4.

38. Скількома способами можна скласти три пари з n шахістів?

39. Скількома способами можна вибрати 6 карт із колоди у 52 карти таким чином, щоб серед них були карти кожної масті?

40. Скількома способами у грі «Спортлото» можна: а) вибрати п’ять куль з 36; б) вгадати 5, 4, 3 номери.

41. Скількома способами можна розмістити 12 різних деталей у 3-х ящиках?

42. Скільки існує автомобільних п’ятизначних номерів, а) складених з цифр 2, 3, 5, 7; б) що не містять цифру 8; в) що не містять цифр 0 і 8?

43. Знайти кількість способів розподілу n однакових куль за: а) двома; б) трьома; в) і n різними урнами.

44. Скількома способами можна утворити дозор із 3 солдатів і 1 офіцера, коли у підрозділі 80 солдатів і 3 офіцери?

45. В лабораторії науково-дослідного інституту працює кілька людей, причому кожна з них знає хоча б одну іноземну мову, 6 – англійську, 6 – німецьку, 7 – французьку, 4 – англійську і німецьку, 3 – німецьку і французьку, 2 – французьку і англійську, одна особа знає всі три мови.

Скільки людей працює в лабораторії? Скільки з них знає лише англійську мову? Скільки осіб знає лише одну мову?

46. Скільки чисел серед першої тисячі натуральних чисел не діляться ані на 2, ані на 3, ані на 5, ані на 7?

47. Скільки шестизначних чисел можна скласти з цифр 1, 2, 3, ... , 9, якщо будь-яке число має складатися з трьох парних і трьох непарних цифр, причому ніякі 2 цифри в числі не повторюються?

48. Кидають гральний кубик двічі. Скільки може бути результатів, за яких цифра, що випаде у разі першого кидання, відрізняється від цифри, що випаде при другому киданні?

49. Скільки слів можна одержати, переставляючи букви слів: а) «парабола»; б) «метаморфоза»?

50. Як премії на математичній олімпіаді виділено 3 екземпляри однієї книги, 4 – другої і 8 – третьої. Скількома способами можна розділити ці премії між 30 учасниками, якщо будь-кому вручають не більше однієї книги?

51. Абітурієнт має скласти 4 іспити. Він вважає, щоб бути зарахованим, достатньо набрати 17 очок. Скількома способами він зможе скласти іспити, набравши не менше 17 очок і не одержавши жодної двійки?

52. Скільки слів, можна скласти з двох букв А, п’яти букв Б і дев’яти букв В?

53. Скільки слів з п’яти букв, будь-яке з яких складається з трьох приголосних і двох голосних, можна скласти з букв слова «рівняння»?

54. Скількома способами можна 15 здобувачів освіти розділити на 2 групи так, щоб в одній групі було 11, а у другій групі – 4 здобувачі освіти?

55. Скількома способами можна переставити букви слова «обороноздатність», щоб дві букви «о» не йшли підряд?

56. Скількома способами можна 15 шахістів поділити на 3 команди по 5 осіб?

57. Скільки існує трикутників, довжини сторін яких набувають одне з таких значень: 4, 5, 6, 7 см?

58. З 10 різних квіток треба скласти букет так, щоб у ньому було не менше 2 квіток. Скількома способами можна скласти такий букет?

59. Скількома способами 9 пасажирів можна розмістити у трьох вагонах? Для скількох розміщень у перший вагон сядуть 3 з них? Для

скількох розміщень у кожний вагон сядуть 3 з них? Для скількох розміщень в один вагон сяде 4, у другий – 3, а у третій – 2 пасажири?

60. Скількома способами можна відібрати кілька фруктів із 7 яблук, 4 лимонів, 9 апельсинів (вважаємо, що фрукти одного виду не відрізняються один від одного)?

Приклад. *Інтернет-адреси.* Інтернет – це мережа взаємопов'язаних комп'ютерів. Кожен комп'ютер або його інтерфейс в інтернеті має свою інтернет-адресу. У схемі адресації IPv4 (Internet Protocol, Version 4) адреси поділяються на п'ять класів – від класу A до класу E. Лише класи A, B і C використовуються для ідентифікації комп'ютерів в інтернеті. Адреси класу A призначені для великих мереж, класу B – для мереж середнього розміру, а класу C – для малих мереж.

Адреса класу A – це бітовий рядок довжиною 32 біти. Перший біт – 0 (щоб ідентифікувати його як адресу класу A). Наступні 7 бітів, які називаються *netid*, ідентифікують мережу. Решта 24 біти, які називаються *hostid*, ідентифікують комп'ютерний інтерфейс. *Netid* не може складатися з усіх одиниць. *Hostid* не може складатися з усіх нулів або всіх одиниць. Скільки існує унікальних адрес класу A?

Розв'язання. Загальна кількість 7-бітових рядків становить 2^7. Але рядок 1111111 не дозволено використовувати як *netid*, а тому можливих значень для *netid* є $(2^7 - 1)$. Аналогічно визначаємо, що для *hostid* є $(2^{24} - 2)$ можливих значення, оскільки це 24-бітовий рядок, для якого два рядки, що складаються з усіх нулів або усіх одиниць, не дозволяються.

За *правилом добутку*, кількість адрес класу A дорівнює:

$$(2^7 - 1)\,(2^{24} - 2) = 127 \times 16\,777\,214 = 2\,130\,706\,178.$$

Таким чином, існує **2 130 706 178** адрес класу A.

Тема 5. ЧИСЛЕННЯ ВИСЛОВЛЕНЬ

В алгебрі логіки (численні висловлень) об'єктом досліджень є ***висловлення.***

Означення:	Будь-яке твердження, яке може бути істинним або хибним, називається ***висловленням***.

Істинним висловленням приписується значення 1, ***хибним*** – 0. З кількох простих висловлень можна скласти складні висловлення. Окремі висловлення позначаються латинськими літерами A, B, C,…

Приклад. Прості висловлення:

A = «Вісім – парне число»;
B = «Вісім ділиться на два»;
C = «Вісім ділиться на 3»;
D = «Київ – столиця України».

A = 1, B = 1, C = 0, D = 1.

Приклад. Складне висловлення:

«Якщо вісім – парне число, то вісім ділиться на 2» = 1 (істинне).

Означення:	Для об'єднання простих висловлень у складні використовуються ***логічні операції*** над висловленнями.

Основні логічні операції над висловленнями:

1. «*Константа нуль»*. Передає висловлення, яке завжди є хибним $F = 0$.

2. «*Константа одиниця*». Передає висловлення, яке завжди є істинним $F = 1$.

3. Логічною операцією «*Ні»* називають висловлення $F = \bar{A}$, яке є істинним тоді, коли A хибне, і хибним, коли A – істинне. Читається «не A»,

4. Логічною операцією «*І»* (кон'юнкція, логічне множення, логічний добуток) називають складне висловлення $F = A \wedge B$ $(A \cdot B, AB, A\&B)$, яке є істинним тоді і тільки тоді, коли обидва прості висловлення істинні, і хибним, коли хоча б одне просте висловлення хибне.

5. Логічною операцією «*Або»* (диз'юнкцією, логічною сумою, логічним додаванням) називають складне висловлення $F = A \vee B$ $(A + B)$, яке є хибним тоді і тільки тоді, коли обидва прості висловлення хибні, і істинним, коли хоча б одне просте висловлення істинне.

6. Логічною операцією «*Якщо – то»* (імплікація) називають складне висловлення $F = A \to B$, яке є хибним тоді і лише тоді, коли A істинне, а B хибне. Читається «Якщо A, то B».

7. Логічною операцією «*Заборона на B*» (заперечення імплікації) називають складне висловлення $F = A\Delta B = \overline{A \to B}$, яке є істинним тоді і лише тоді, коли A істинне, а B хибне. Читається «Хибно, що якщо A, то B».

8. Логічною операцією «*Заборона на A*» (заперечення імплікації) називають складне висловлення $F = B\Delta A = \overline{B \to A}$, яке є істинним тоді і тільки тоді, коли B істинне, а A хибне. Читається «Хибно, що якщо B, то A».

9. Логічною операцією «*Рівнозначність*» (еквівалентність) називають складне висловлення $F = \text{A}\sim\text{B}$ $(A \equiv B)$, яке є істинним тоді і тільки тоді, коли обидва прості висловлення A і B істинні або хибні одночасно. Читається «A рівнозначне B».

10. Логічною операцією «*Нерівнозначність»* (сума за модулем 2) називають складне висловлення $F = \text{A}\oplus\text{B}$ $(A \neq B)$, яке є істинним тоді і тільки тоді, коли одне висловлення є істинним, а друге – хибним. Читається «A нерівнозначне B» або «Сума за модулем 2».

11. Логічною операцією «*Стрілка Пірса*» (функція Вебба, операція Пірса) називають складне висловлення $F = A \downarrow B = \overline{A \vee B}$, яке є істинним тоді і лише тоді, коли обидва висловлення хибні одночасно. Читається «Ні A, ні B».

12. Логічною операцією «*Операція Шефера*» (Штрих Шефера) називають складне висловлення $F = A|B$, яке є хибним тоді і тільки тоді, коли обидва прості висловлення є істинними одночасно. Читається «Хибно, що A і B»

13. Логічною операцією «*Змінна A*» називають висловлення $F = A$, яке дорівнює 0 тоді і лише тоді, коли $A = 0$ і одиниці, коли $A = 1$. Читається «Висловлення залежить лише від A».

В алгебрі логіки існують логічні закони, логічні суперечності і твердження, що логічно виконуються.

Означення:	Складне висловлення, яке є істинним за всіх можливих комбінацій значень простих висловлень, називається ***логічним законом***.

Означення:	Складне висловлення, яке є хибним за всіх можливих комбінацій значень простих висловлень, називається ***логічною суперечністю***.

Означення:	Складне висловлення, яке є істинним для одних значень простих висловлень і хибним для решти, називається ***твердженням, що логічно виконується.***

Основні закони алгебри логіки

Ч.ч.	**Назва закону**	**Логічний запис**
1.	***Тотожності***	$A = A$
2.	***Суперечності***	$\overline{A \cdot \bar{A}} = 1$
3.	***Виключення третього***	$A + \bar{A} = 1$
4.	***Ідемпотентності***	$A \cdot A = A,$ $A + A = A$
5.	***Комутативний***	$AB = BA,$ $A + B = B + A$
6.	***Асоціативний***	$(AB)C = A(BC)$ $(A + B) + C = A + (B + C)$
7.	***Дистрибутивний***	$A(B + C) = AB + AC$ $A + BC = (A + B)(A + C)$
8.	***Поглинання***	$A(B + A) = A$ $A + AB = A$
9.	***Подвійності (правила де Моргана)***	$\overline{AB} = \bar{A} + \bar{B}$ $\overline{A + B} = \bar{A} \cdot \bar{B}$
10.	***Подвійного заперечення***	$A = \bar{\bar{A}}$
11.	***Властивість одиниці***	$A \cdot 1 = A$ $A + 1 = A$
12.	***Властивість нуля***	$A \cdot 0 = 0$ $A + 0 = A$

Поняття логічної функції

Означення:	Функція F від n змінних (аргументів) $x_1, x_2, \dots, x_n$, яка так само, як і її аргументи, може набувати лише два значення 0 і 1, називається ***логічною*** (двійковою, булевою, перемикальною).

Означення:	Сукупність значень $a_1, a_2, \dots, a_n$ n змінних $x_1, x_2, \dots, x_n$ називається ***набором*** і позначається $a_1, a_2, \dots, a_n$, де a_i є $\{0,1\}$, $i = 1, 2, \dots, n$.

Теорема 5.1 Кількість наборів для аргументів $x_1, x_2, \dots, x_n$ логічної функції $N = 2^n$.

№ набору	x_1	x_2	x_3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Теорема 5.2 Кількість різних логічних функцій від n аргументів $M = 2^{2^n}$. Коли $n = 1$, існує 4 логічні функції.

№ набору	A	F_0	F_1	F_2	F_3
0	0	0	0	1	1
1	1	0	1	0	1
		const 0	*A*	$\bar{A}$	*const* 1

Коли $n = 2$, існує $2^{2^2} = 16$ логічних функцій.

Ч,ч.	Змінна		Функції															
	A	B	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	**0**	**0**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	**0**	**1**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	**1**	**0**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	**1**	**1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

$F_0 = 0$ – «*константа нуль*»,
$F_1 = A \wedge B$ – «*кон'юнкція*»,
$F_2 = A \Delta B$ – «*заборона на B*»,
$F_3 = A$ – «*змінна A*»,
$F_4 = B \Delta A$ – «*заборона на A*»,
$F_5 = B$ – «*змінна B*»,
$F_6 = A \oplus B$ – «*нерівнозначність*»,
$F_7 = A \vee B$ – «*диз'юнкція*»,
$F_8 = A \downarrow B$ – «*стрілка Пірса*»,
$F_9 = A \sim B$ – «*рівнозначність*»,
$F_{10} = \bar{B}$ – «*не B*»,
$F_{11} = B \rightarrow A$ – «*імплікація від B до A*»,
$F_{12} = \bar{A}$ – «*не A*»,
$F_{13} = A \rightarrow B$ – «*імплікація від A до B*»,
$F_{14} = A|B$ – «*операція Шефера*»,
$F_{15} = 1$ – «*константа одиниця*».

Між логічними функціями і множинами існує тісний зв'язок. Логічні функції визначають операції над двома видами множин: універсальною і порожньою. Тому всі правила і закони теорії множин мають місце і для логічних функцій з урахуванням того, що таких множин лише 2, а звідси випливає, що теорія логічних функцій є окремим випадком теорії множин.

Приклад. $A + B \Leftrightarrow A \cup B$.

Перетворення логічних функцій

Розглянуті раніше 16 логічних функцій 2-х змінних називають елементарними. На основі них будується алгебра логіки та її численні застосування в науці і техніці. Серед цих функцій виділимо базисні. Вони так називаються тому, що через них можна виразити будь-які інші логічні функції. У той же час базисні логічні функції не можуть бути отримані з більш простих. До них належать:

1. Константа 0.
2. Константа 1.
3. Змінна.
4. Інверсія.
5. Диз'юнкція.
6. Кон'юнкція.

Означення:	Логічна функція називається ***інверсною*** відносно інших, якщо вона може бути отримана з останньої способом інверсії всіх її значень. Кожна з 16-ти функцій має інверсну $(F_0 - F_{15},\ F_3 - F_{12}, \dots,)$.

Теорема 5.3. $A \wedge B = \overline{\bar{A} \vee \bar{B}}$ – закон де Моргана.

A	B	$A \wedge B$	$\bar{A}$	$\bar{B}$	$\bar{A} \vee \bar{B}$	$\overline{\bar{A} \vee \bar{B}}$
0	0	0	1	1	1	0
0	1	0	1	0	1	0
1	0	0	0	1	1	0
1	1	1	0	0	0	1

Теорема 5.4. $A \vee B = \overline{\bar{A} \wedge \bar{B}}$ – закон де Моргана.

A	B	$A \vee B$	$\bar{A}$	$\bar{B}$	$\bar{A} \wedge \bar{B}$	$\overline{\bar{A} \wedge \bar{B}}$
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	0	1	0	1
1	1	1	0	0	0	1

Теорема 5.5. $A \to B = \bar{A} \vee B$.

A	B	$A \to B$	$\bar{A}$	B	$\bar{A} \vee B$
0	0	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	1	1	0	1	1

Теорема 5.6. $A{\sim}B = (\bar{A} \vee B) \wedge (A \vee \bar{B})$.

A	B	$A{\sim}B$	$\bar{A}$	B	$\bar{A} \vee B$	A	$\bar{B}$	$A \vee \bar{B}$	$(\bar{A} \vee B) \wedge (A \vee \bar{B})$
0	0	1	1	0	1	0	1	1	1
0	1	0	1	1	1	0	0	0	0
1	0	0	0	0	0	1	1	1	0
1	1	1	0	1	1	1	0	1	1

Аналогічно доводять і такі співвідношення:

1. $A = \bar{\bar{A}}$.

2. $A{\sim}B = AB \vee \bar{A}\bar{B}$.

3. $A\Delta B = \overline{\bar{A} \vee B} = A \wedge \bar{B}$.

4. $B\Delta A = \overline{\bar{B} \vee A} = B \wedge \bar{A}$.

5. $A \oplus B = \overline{A{\sim}B} = \overline{(\bar{A} \vee B) \cdot (A \vee \bar{B})} = (A \wedge \bar{B}) \vee (\bar{A} \wedge B)$.

6. $A \downarrow B = \overline{A \vee B} = \bar{A} \wedge \bar{B}$.

7. $A|B = \overline{A \wedge B} = \bar{A} \vee \bar{B}$.

8. $B \to A = \bar{B} \wedge A$.

Співвідношення для x та 0 або x та 1.

Ч.ч.	***x* та 1**	***x* та 0**	***x* та *x***	***x* та $\bar{x}$**
1.	$x \vee 1 = 1$	$x \vee 0 = x$	$x \vee x = x$	$x \vee \bar{x} = 1$
2.	$x \wedge 1 = x$	$x \wedge 0 = 0$	$x \wedge x = x$	$x \wedge \bar{x} = 0$
3.	$x{\sim}1 = x$	$x{\sim}0 = \bar{x}$	$x{\sim}x = 1$	$x{\sim}\bar{x} = 0$
4.	$x \oplus 1 = \bar{x}$	$x \oplus 0 = x$	$x \oplus x = 0$	$x \oplus \bar{x} = 1$
5.	$x\|1 = \bar{x}$	$x\|0 = 1$	$x\|x = \bar{x}$	$x\|\bar{x} = 1$
6.	$x \downarrow 1 = 0$	$x \downarrow 0 = \bar{x}$	$x \downarrow x = \bar{x}$	$x \downarrow \bar{x} = 0$
7.	$x \to 1 = 1$	$x \to 0 = \bar{x}$	$x \to x = 1$	$x \to \bar{x} = x$
8.	$1 \to x = x$	$0 \to x = 1$		

Означення:	Функція $F = F(F_1, F_2, \dots, F_n)$, яка отримана з функцій $F_1, F_2, \dots, F_n$ шляхом їхньої підстановки замість аргументів $x_1, x_2, \dots, x_n$ функції $F(x_1, x_2, \dots, x_n)$, називається ***суперпозицією функцій*** $F_1, F_2, \dots, F_n$.

Приклад. Дано $F = F(x_1, x_2, x_3, x_4) = \Big(\big((x_1 \vee x_2) \wedge x_3\big) \to x_4\Big)$ і функції $A = (\bar{x}_1 \sim x_3)$, $B = x_1 \downarrow x_2$, $C = x_1 \oplus x_2$, $D = x_3$. Потрібно отримати суперпозицію функцій A, B, C, D, $F(A, B, C, D)$, а з неї – функцію $F(x_1, x_2, x_3)$.

Розв'язання. $F(A, B, C, D) = \Big(\big((A \vee B) \wedge C\big) \to D\Big)$;

$$F = \Big(\Big(\big((\bar{x}_1 \sim x_3) \vee (x_1 \downarrow x_2)\big) \wedge (x_1 \oplus x_2)\Big) \to x_3\Big).$$

***Спеціальні формули*:**

1. Операції поглинання: $x \vee xy = x$, $x(x \vee y) = x$.
2. Операції склеювання: $xy \vee x\bar{y} = x$, $(x \vee y)(x \vee \bar{y}) = x$.
3. Операції дії з дужками: $xy \vee xz = x(y \vee z)$.
4. Формули де Моргана: $\overline{x \vee y} = \bar{x} \wedge \bar{y}$, $\overline{x \wedge y} = \bar{x} \vee \bar{y}$.
5. $x_1 \wedge F(x_1, x_2, \dots, x_n) = x_1 \wedge F(0, x_2, \dots, x_n)$.
6. $x_1 \vee (x_1, x_2, \dots, x_n) = x_1 \vee F(0, x_2, \dots, x_n)$.
7. $\bar{x}_1 \wedge F(x_1, x_2, \dots, x_n) = \bar{x}_1 \wedge F(0, x_2, \dots, x_n)$.
8. $\bar{x}_1 \vee F(x_1, x_2, \dots, x_n) = \bar{x}_1 \vee F(1, x_2, \dots, x_n)$.
9. $F(x_1, x_2, \dots, x_n, \vee, \wedge) = F(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n, \wedge, \vee)$.

Питання для самоконтролю

1) Що таке висловлення?
2) Дайте означення логічної операції «*І*».
3) Що таке логічний закон?
4) Що таке логічна суперечність?
5) Поясніть закони де Моргана. Як вони застосовуються в численні висловлень?
6) Які є методи доведення правильності формул у численні висловлень?

Індивідуальні практичні завдання № 5

Скласти таблицю істинності для логічної функції, заданої у таблиці 5.1.

Таблиця 5.1 – Варіанти завдань

Номер варіанта	Функція	Номер варіанта	Функція
1.	$f = (A \to B) \sim (C \mid B) \oplus D;$	16.	$f = A \sim C \mid B \to A \oplus D;$
2.	$f = A \sim B \wedge C \mid A \to D;$	17.	$f = A \sim (C \mid B \vee \overline{C}) \to D;$
3.	$f = A \to B \sim C \vee \overline{A} \oplus D;$	18.	$f = (A \mid B) \oplus (A \mid C) \sim D;$
4.	$f = (\overline{A} \sim B) \wedge (A \sim \overline{C}) \oplus D;$	19.	$f = (A \sim B) \vee C \to (A \wedge D);$
5.	$f = (C \vee \overline{A}) \to (A \sim B) \oplus D;$	20.	$f = A \sim B \vee C \to B \wedge D;$
6.	$f = A \sim B \to C \vee \overline{A} \oplus D;$	21.	$f = A \sim B \to C \vee A \oplus D;$
7.	$f = A \sim C \vee (B \mid A) \oplus D;$	22.	$f = (A \sim B) \to C \vee B \downarrow D;$
8.	$f = (A \sim B) \vee \overline{A} \to (C \oplus \overline{D});$	23.	$f = A \sim (B \mid C) \to B \oplus D;$
9.	$f = A \to B \wedge C \sim (A \oplus \overline{D});$	24.	$f = (A \to B) \wedge C \sim A \mid D;$
10.	$f = A \to B \mid (C \oplus A) \sim D;$	25.	$f = (A \sim B) \mid C \vee A \oplus D;$
11.	$f = (A \sim B) \wedge (\overline{A} \to C) \oplus D;$	26.	$f = (A \to B) \to C \vee A \sim D;$
12.	$f = (A \sim B) \oplus C \vee (\overline{A} \to D);$	27.	$f = (A \to B) \sim C \wedge (\overline{B} \oplus D);$
13.	$f = C \vee B \sim A \vee (\overline{B} \to D);$	28.	$f = C \vee \overline{A} \to B \sim (C \oplus D);$
14.	$f = A \to B \sim C \wedge (\overline{B} \oplus D);$	29.	$f = A \to B \sim (C \mid A) \oplus \overline{D};$
15.	$f = B \to C \sim A \vee (\overline{B} \oplus D);$	30.	$f = (A \sim B) \wedge C \to \overline{B} \oplus D.$

Приклад. Складемо таблицю істинності для функції $f = B \to C \equiv A \vee (\overline{B} \oplus D)$

A	B	C	D	$\overline{B}$	$\overline{B} \oplus D$	$A \vee (\overline{B} \oplus D)$	$B \to C$	$B \to C \equiv A \vee (\overline{B} \oplus D)$
0	0	0	0	1	1	1	1	1
0	0	0	1	1	0	0	1	0
0	0	1	0	1	1	1	1	1
0	0	1	1	1	0	0	1	0
0	1	0	0	0	0	0	0	1
0	1	0	1	0	1	1	0	0
0	1	1	0	0	0	0	1	0
0	1	1	1	0	1	1	1	1
1	0	0	0	1	1	1	1	1
1	0	0	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1
1	0	1	1	1	0	1	1	1
1	1	0	0	0	0	1	0	0
1	1	0	1	0	1	1	0	0
1	1	1	0	0	0	1	1	1
1	1	1	1	0	1	1	1	1

Тема 6. ДИЗ'ЮНКТИВНІ ТА КОН'ЮНКТИВНІ НОРМАЛЬНІ ФОРМИ

6.1 Диз'юнктивна нормальна форма подання логічної функції
6.2 Мінімізація логічних функцій в ДДНФ методом Квайна
6.3 Кон'юнктивна нормальна форма подання логічної функції
6.4 Застосування карт Карно для мінімізації логічних функцій

6.1 Диз'юнктивна нормальна форма подання логічної функції

Логічна функція може бути подана в різних формах. Найбільш поширеними формами є диз'юнктивна нормальна форма (ДНФ) та кон'юнктивна нормальна форма (КНФ) подання функцій.

Означення:	Логічний добуток кількох змінних, взятих із запереченням або без нього, називається ***елементарним добутком,*** або ***кон'юнкцією.***

Приклад. Елементарні добутки: $x_1\bar{x}_2$, $x_1\bar{x}_2x_3$, $\bar{x}z$, xyz. Не є елементарними добутками: $\overline{xy}$, $xy \vee x\bar{z}$.

Означення:	Логічна функція, що подається диз'юнкцією елементарних добутків (кон'юнкцій) називається ***диз'юнктивною нормальною формою (ДНФ)***

Теорема 6.1. Елементарний добуток дорівнює одиниці лише на одному наборі, коли змінні з запереченням приймають значення нуля, а змінні без заперечення приймають значення одиниці.

Приклад. $\bar{x}_1x_2x_3 = 1$ на наборі 011. $\bar{0} \cdot 1 \cdot 1 = 1 \cdot 1 \cdot 1 = 1$.

Теорема 6.2. Для кожного набору значень змінних існує один і лише один їх елементарний добуток, що дорівнює одиниці.

Означення:	Логічна функція *n* змінних, що набуває значення, яке рівне одиниці, лише на одному їх наборі, називається ***конституентою одиниці.***

Теорема 6.3. Елементарний добуток усіх *n* змінних, що входять до функції *F*, є конституентою одиниці.

Теорема 6.4. Число конституент одиниці *n* змінних дорівнює 2^n.

Приклад. Знайти конституенту одиниці 17-го набору.

Розв'язання. $(17)_{10} = \left(\overset{16}{\underset{\cup}{1}}\ \overset{8}{\underset{\cup}{0}}\ \overset{4}{\underset{\cup}{0}}\ \overset{2}{\underset{\cup}{0}}\ \overset{1}{\underset{\cup}{1}}\right)_2$, тому конституента $x_1\bar{x}_2\bar{x}_3\bar{x}_4x_5$.

Означення:	Диз'юнкція конституент одиниці, яка дорівнює одиниці на тих самих наборах, що й задана функція, називається ***досконалою ДНФ (ДДНФ)*** логічної функції.

Теорема 6.5. Будь-яка логічна функція F, крім константи нуля (c*onst* 0), може бути поданою в ДДНФ єдиним способом.

Алгоритм заповнення ДДНФ за таблицею істинності функції:

1. Виписати добутки всіх аргументів від першого до n-го у кількості, рівній кількості одиниць у таблиці істинності логічної функції і з'єднати їх знаком ∨ (диз'юнкції).

2. Записати під кожним аргументом його значення у відповідному до даного добутку наборі, що дорівнює 1 або 0 і над аргументами, що дорівнюють 0, поставити знаки заперечення.

Цей алгоритм називається алгоритмом запису логічної функції за одиницями.

Приклад. Подати у ДДНФ логічну функцію п'яти аргументів, що дорівнює одиниці на наборах з номерами 5, 14, 16, 31 і дорівнює нулю на решті наборів.

Розв'язання. $(5)_{10} = \left(\overset{16}{0}\ \overset{8}{0}\ \overset{4}{1}\ \overset{2}{0}\ \overset{1}{1}\right)_2 \Rightarrow \bar{x}_1\bar{x}_2x_3\bar{x}_4x_5;$

$$(14)_{10} = \left(\overset{16}{0}\ \overset{8}{1}\ \overset{4}{1}\ \overset{2}{1}\ \overset{1}{0}\right)_2 \Rightarrow \bar{x}_1x_2x_3x_4\bar{x}_5;$$

$$(16)_{10} = \left(\overset{16}{1}\ \overset{8}{0}\ \overset{4}{0}\ \overset{2}{0}\ \overset{1}{0}\right)_2 \Rightarrow x_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5;$$

$$(31)_{10} = \left(\overset{16}{1}\ \overset{8}{1}\ \overset{4}{1}\ \overset{2}{1}\ \overset{1}{1}\right)_2 \Rightarrow x_1x_2x_3x_4x_5.$$

$$F = \bar{x}_1\bar{x}_2x_3\bar{x}_4x_5 \vee \bar{x}_1x_2x_3x_4\bar{x}_5 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5 \vee x_1x_2x_3x_4x_5.$$

ДДНФ є найбільш загальною формою подання логічних функцій в ДНФ і тому містить найбільше можливе число літер. На практиці та, відповідно, в теорії постає проблема скорочення числа літер в ДДНФ.

Означення:	Якщо деяка логічна функція φ (в окремому випадку елементарний добуток) дорівнює нулю на тих наборах, на яких дорівнює нулю інша функція F, то функція φ називається ***імплікантою*** функції F. При цьому функція φ може дорівнювати нулю на тих наборах, на яких F дорівнює одиниці, але не навпаки.

У разі, коли функція φ є імплікантою функції F, то кажуть, що функція φ входить до функції F. Це записується як $\varphi \subset F$. Термін імпліканта виник через те, що функція імплікації $\varphi \to F$ дорівнює 1 лише тоді, коли $\varphi \subset F$, тобто, на наборах, де значення для φ *і* F, відповідно, мають вигляд 0-0, 0-1, 1-1.

Приклад. В таблиці задано функції $F = F(x_1, x_2, x_3)$, $\varphi = \varphi(x_1, x_2, x_3)$ та $Z = Z(x_1, x_2, x_3)$. Яка з функцій φ та Z є імплікантою функції F?

№ набору	x_1	x_2	x_3	F	φ	Z
0	0	0	0	1	1	0
1	0	0	1	0	0	0
2	0	1	0	1	0	0
3	0	1	1	1	1	0
4	1	0	0	***0***	0	***1***
5	1	0	1	0	0	0
6	1	1	0	1	0	0
7	1	1	1	1	1	1

Розв'язання. Функція Z не є імплікантою F, а функція φ є.

Теорема 6.6. Будь-яка конституента одиниці, що входить до ДДНФ логічної функції F, є її імплікантою.

Теорема 6.7. Константа нуля (*const* 0) є імплікантою будь-якої логічної функції.

Теорема 6.8. Будь-яка логічна функція є імплікантою константи одиниці (*const* 1).

Означення:	Дві логічні функції φ_1 і φ_2 називаються ***зрівнюваними***, якщо для них виконуються умови $\varphi_1 \subset \varphi_2$ і $\varphi_2 \subset \varphi_1$.

Означення:	Елементарний добуток, отриманий шляхом вилучення з початкового добутку однієї або кількох змінних, називається ***власною частиною*** останнього.

Приклад. Дано елементарний добуток $x\bar{y}z$. Тоді його власними частинами будуть добутки $x\bar{y}$, $\bar{y}z$, $xz, x, \bar{y}, z$.

Означення:	Елементарні добутки, які входять до заданої функції в ДДНФ, але ніяка їх власна частина самостійно, як добуток, не входить, називаються ***простими імплікантами.***

Приклад. Припустимо, що добутки $\varphi = x_1x_2x_3\bar{x}_4$ і $x_2\bar{x}_4$ входять до деякої логічної функції $F = F(x_1, x_2, x_3, x_4)$, а змінні x_2 і $\bar{x}_4$ самостійно не

входять як добутки, тоді добуток $x_2\bar{x}_4$ буде простою імплікантою функції F, оскільки $x_2\bar{x}_4 \subset \varphi \subset F$, а $x_2 \not\subset F$ і $\bar{x}_4 \not\subset F$. Разом з тим добуток φ не буде простою імплікантою, оскільки $x_2\bar{x}_4 \subset F$.

Означення:	Диз'юнкція простих імплікант називається ***скороченою ДНФ.***

Теорема 6.9. Будь-яку логічну функцію F можна подати у вигляді скороченої ДНФ.

Для подання логічної функції F у вигляді скороченої ДНФ використовують операції повного та неповного склеювання, а також поглинання і розгортання в ДНФ.

Операція повного склеювання: $xy \vee x\bar{y} = x$.

Операція неповного склеювання: $xy \vee x\bar{y} = x \vee xy \vee x\bar{y}$.

Операція поглинання: $x \vee xy = x$ і $x \vee x\bar{y} = x$.

Іноді потрібно навпаки, зі скороченої ДНФ отримати ДДНФ, тоді застосовується операція розгортання. Вона перетворює будь-яку просту імпліканту в диз'юнкцію конституент одиниці.

Приклад. $x\bar{y}$ – проста імпліканта логічної функції 4-ох аргументів x, y, z, u. Тоді, застосовуючи операцію розгортання, отримаємо

$$x\bar{y} = x\bar{y}(z \vee \bar{z}) = x\bar{y}z \vee x\bar{y}\bar{z} = x\bar{y}z(u \vee \bar{u}) \vee x\bar{y}\bar{z}(u \vee \bar{u}) = \\ = x\bar{y}zu \vee x\bar{y}z\bar{u} \vee x\bar{y}\bar{z}u \vee x\bar{y}\bar{z}\bar{u}$$

У процесі розгортання різні імпліканти можуть утворювати одну й ту ж саму конституенту одиниці. У цьому разі на основі тотожності $x \vee x = x$ потрібно залишити одну конституенту одиниці. У результаті буде отримано ДДНФ початкової логічної функції.

6.2 Мінімізація логічних функцій в ДДНФ методом Квайна

Розглянемо метод мінімізації логічних функцій на основі **теореми Квайна.** Якщо в ДДНФ логічної функції F провести всі можливі операції неповного склеювання, потім всі операції поглинання, то буде одержана скорочена ДНФ цієї функції, тобто диз'юнкція всіх її простих імплікант.

Якщо функція задана в довільній ДНФ, то перед мінімізацією за методом Квайна її потрібно розгорнути в ДДНФ.

Після цього потрібно виконати такі кроки.

1. У ДДНФ $F = F(x_1, x_2, \ldots . x_n)$ провести всі операції неповного склеювання конституент одиниці. Кожна конституента одиниці може водночас склеюватись з кількома іншими. Тоді, після першого склеювання, її не поглинають, а використовують для інших, у порядку черги операцій

склеювання. В результаті отримують імпліканти, що мають по $(n-1)$ змінних. При цьому можливі отримання і простих імплікант.

2. Далі відбуваються поглинання імплікантами всіх конституент одиниці, які беруть участь у неповному склеюванні. Конституенти одиниці, які не були задіяні в операціях склеювання, не можуть поглинатися, оскільки вони є простими імплікантами з n-змінними.

3. Здійснюють операції неповного склеювання і поглинання імплікант з $(n-1)$ змінною, одержаних на першому кроці склеювання за аналогією з п. 1 і п. 2.

Ця процедура повторюється доти, доки операції неповного склеювання залишаться можливими. Отримана в результаті ДНФ буде скороченою ДНФ (СДНФ).

Приклад. Знайти скорочену ДНФ логічної функції

$$F = F(xyz) = xy \vee \bar{x}z \vee \bar{y}\bar{z}.$$

Розв'язання.

1. Отримаємо ДДНФ: $F = xy(z \vee \bar{z}) \vee \bar{x}z(y \vee \bar{y}) \vee \bar{y}\bar{z}(x \vee \bar{x}) = xyz \vee xy\bar{z} \vee \bar{x}yz \vee \bar{x}\bar{y}z \vee x\bar{y}\bar{z} \vee \bar{x}\bar{y}\bar{z}$.

2. Виконаємо всі можливі склеювання. Для цього всі конституенти одиниці пронумеруємо і виконаємо всі операції неповного склеювання членів функції, починаючи з першого з рештою.

3. Результати склеювання занесемо до таблиці

Ч.ч.	№ склеюваних конституєнт	Результат склеювання (імпліканти)	Склеєна змінна
1	1-2	xy	z
2	1-3	yz	x
3	2-5	$x\bar{z}$	y
4	3-4	$\bar{x}z$	y
5	4-6	$\bar{x}\bar{y}$	z
6	5-6	$\bar{y}\bar{z}$	x

Очевидно, що подальше склеювання неможливе, і наведені в таблиці імпліканти будуть простими.

Одержимо СДНФ: $F = \underline{xy} \vee \underline{\underline{yz}} \vee \underline{\underline{x\bar{z}}} \vee \underline{\bar{x}z} \vee \underline{\underline{\bar{x}\bar{y}}} \vee \underline{\bar{y}\bar{z}}$,

$\underline{xy}, \underline{\bar{x}z}, \underline{\bar{y}\bar{z}}$ – старі; $\underline{\underline{yz}}, \underline{\underline{x\bar{z}}}, \underline{\underline{\bar{x}\bar{y}}}$ – зайві.

Висновок. СДНФ – це далеко не завжди мінімальна ДНФ, оскільки вона хоч і містить прості імпліканти, проте серед них можуть бути і надлишкові.

Означення:	Диз'юнкція простих імплікант, жодна з яких не є зайвою, називається ***тупиковою ДНФ*** логічної функції.

Тупикових ДНФ у загальному випадку може бути декілька. Виникає задача пошуку такої тупикової ДНФ, яка мала б найменшу кількість літер. Така ДНФ називається мінімальною ДНФ.

Деякі логічні функції мають кілька мінімальних ДНФ.

З метою визначення мінімальних ДНФ скористаємось імплікантною матрицею

Ч.ч.	**Проста імпліканта**	**Конституєнта**					
		1	**2**	**3**	**4**	**5**	**6**
		xyz	$xy\bar{z}$	$\bar{x}yz$	$\bar{x}\bar{y}z$	$x\bar{y}\bar{z}$	$\bar{x}\bar{y}\bar{z}$
1	xy	**x**	**x**				
2	yz	**x**		**x**			
3	$x\bar{z}$		**x**			**x**	
4	$\bar{x}z$			**x**	**x**		
5	$\bar{x}\bar{y}$				**x**		**x**
6	$\bar{y}\bar{z}$					**x**	**x**

По вертикалі: конституенти одиниці, що входять в зазначену функцію, по горизонталі: прості імпліканти, одержані з СДНФ.

Якщо імпліканта є власною частиною деякої конституенти одиниці, то відповідна клітинка відмічається «**x**». Щоб одержати мінімальну ДНФ заданої функції достатньо знайти мінімальну кількість імплікант, які разом накривають «**x**»-ми всі стовпці імплікантної матриці (задача про покриття).

Отже, дана логічна функція має дві тупикових ДНФ з кількістю літер, що дорівнює 6-ти:

$$1)\ F = xy \vee \bar{x}z \vee \bar{y}\bar{z}; \qquad \text{та} \qquad 2)\ F = yz \vee x\bar{z} \vee \bar{x}\bar{y}.$$

Є й інші тупикові ДНФ з кількістю літер більшою за 6, наприклад,

$$F = xy \vee yz \vee x\bar{z} \vee \bar{x}\bar{y}$$

Решта кроків мінімізації логічної функції:

4. За отриманою СДНФ будується імплікантна матриця.

5. В імплікантній матриці знаходиться набори простих імплікант, які накривають всі конституенти одиниці логічної функції.

6. Серед цих наборів знаходять один або кілька таких, які в сумі містять мінімальну кількість літер.

7. Об’єднують прості імпліканти цих наборів знаком диз’юнкції «∨», і одержують одну або декілька мінімальних ДНФ.

6.3 Кон'юнктивна нормальна форма подання логічної функції

Означення:	Логічна сума кількох змінних, взятих із запереченням або без нього, називається ***елементарною сумою*****,** або ***диз'юнкцією***.

Приклад. Елементарні суми: $x_1 \vee \bar{x}_2$, $x_1 \vee \bar{x}_2 \vee x_3$, $\bar{x} + z$, $x + y + z$; не є елементарними сумами: $\overline{x \vee y}$, $(x + y)(x + \bar{z})$.

Означення:	Логічна функція, що подається кон'юнкцією елементарних диз'юнкцій (сум), називається ***кон'юнктивною нормальною формою (КНФ).***

Теорема 6.10. Елементарна сума дорівнює нулю лише на одному наборі (в єдиному випадку), коли змінні з запереченням приймають значення одиниці, а змінні без заперечення – значення нуля

Приклад. $\bar{x}_1 + x_2 + x_3 = 0$ на наборі 100. $\bar{1} + 0 + 0 = 0 + 0 + 0 = 0$.

Теорема 6.11. Для кожного набору значень змінних існує одна і лише одна їхня елементарна сума, що дорівнює нулю.

Означення:	Логічна функція *n* змінних, що набуває значення, яке дорівнює нулю, лише на одному їхньому наборі, називається ***конституентою нуля.***

Теорема 6.12. Елементарна сума усіх *n* змінних, що входять до функції *F*, є конституентою нуля.

Теорема 6.13. Число конституент нуля *n* змінних дорівнює 2^n.

Приклад. Знайти конституенту нуля 17-го набору.

Розв'язання. $(17)_{10} = \left(\overset{16}{\overset{\frown}{1}}\ \overset{8}{\overset{\frown}{0}}\ \overset{4}{\overset{\frown}{0}}\ \overset{2}{\overset{\frown}{0}}\ \overset{1}{\overset{\frown}{1}}\right)_2$, тому конституента 0:

$$\bar{x}_1 + x_2 + x_3 + x_4 + \bar{x}_5.$$

Означення:	Кон'юнкція конституент нуля, яка дорівнює нулю на тих самих наборах, що й задана функція, називається ***досконалою КНФ (ДКНФ)*** логічної функції.

Теорема 6.14. Будь-яка логічна функція *F*, крім константи одиниці (*const* 1), може бути поданою в ДКНФ єдиним способом.

Алгоритм подання логічної функції у ДКНФ за таблицею істинності функції

1. Виписати суми всіх аргументів від першого до *n*-го у кількості, рівній кількості нулів у таблиці істинності логічної функції. та з'єднати їх знаком ∧ (кон'юнкції).

2. Записати під кожним аргументом його значення у відповідному для даного добутку наборі, що дорівнює 1 або 0, і над аргументами, що дорівнюють 1, поставити знаки заперечення.

Цей алгоритм також називається алгоритмом запису логічної функції за нулями.

Приклад. Подати у ДКНФ логічну функцію п'яти аргументів, що дорівнює нулю на наборах з номерами 5, 14, 16, 31, і дорівнює одиниці на решті наборів.

Розв'язання. $(5)_{10} = \left(\overset{16}{0}\ \overset{8}{0}\ \overset{4}{1}\ \overset{2}{0}\ \overset{1}{1}\right)_2 \quad \Rightarrow \quad x_1 + x_2 + \bar{x}_3 + x_4 + \bar{x}_5;$

$(14)_{10} = \left(\overset{16}{0}\ \overset{8}{1}\ \overset{4}{1}\ \overset{2}{1}\ \overset{1}{0}\right)_2 \quad \Rightarrow \quad x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 + x_5;$

$(16)_{10} = \left(\overset{16}{1}\ \overset{8}{0}\ \overset{4}{0}\ \overset{2}{0}\ \overset{1}{0}\right)_2 \quad \Rightarrow \quad \bar{x}_1 + x_2 + x_3 + x_4 + x_5;$

$(31)_{10} = \left(\overset{16}{1}\ \overset{8}{1}\ \overset{4}{1}\ \overset{2}{1}\ \overset{1}{1}\right)_2 \quad \Rightarrow \quad \bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 + \bar{x}_5.$

$$F = (x_1 + x_2 + \bar{x}_3 + x_4 + \bar{x}_5) \wedge (x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 + x_5) \wedge$$
$$\wedge (\bar{x}_1 + x_2 + x_3 + x_4 + x_5) \wedge (\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 + \bar{x}_5).$$

ДКНФ є найбільш загальною формою подання логічних функцій в КНФ і тому містить найбільше можливе число літер. На практиці і, відповідно, в теорії постає проблема скорочення числа літер в ДКНФ.

Означення:	Якщо деяка логічна функція φ (в окремому випадку елементарна сума) дорівнює одиниці на тих наборах, на яких дорівнює одиниці інша функція F, то функція φ називається функції F. При цьому функція φ може дорівнювати одиниці на тих наборах, на яких F дорівнює нулю, але не навпаки.

Приклад. В таблиці задано функції $F = F(x_1, x_2, x_3)$, $\varphi = \varphi(x_1, x_2, x_3)$ та $Z = Z(x_1, x_2, x_3)$. Яка з функцій φ та Z є імпліцентою функції F?

№ набору	x_1	x_2	x_3	F	φ	Z
0	0	0	0	1	1	***0***
1	0	0	1	0	0	0
2	0	1	0	1	1	1
3	0	1	1	1	1	1
4	1	0	0	0	1	1
5	1	0	1	0	0	0
6	1	1	0	1	1	***0***
7	1	1	1	1	1	1

Розв'язання. Функція Z не є імпліцентою F, а функція φ є.

Теорема 6.15. Будь-яка конституєнта нуля, що входить до ДКНФ логічної функції *F,* є її імпліцентою.

Теорема 6.16. Константа одиниці (*const* 1) є імпліцентою будь-якої логічної функції.

Теорема 6.17. Будь-яка логічна функція є імпліцентою константи нуля (*const* 0).

Означення:	Елементарна сума, що отримана шляхом вилучення з початкової суми однієї або кількох змінних, називається ***власною частиною*** останньої.

Приклад. Дано елементарний добуток $x + \bar{y} + z$. Тоді його власними частинами будуть суми $x + \bar{y},\ \bar{y} + z,\ x + z,\ x,\ \bar{y},\ z$.

Означення:	Елементарні суми, які входять до даної функції в ДКНФ, але ніяка їхня власна частина самостійно як сума не входить, називаються ***простими імпліцентами***.

Приклад. Припустимо, що суми $\varphi = x_1 + x_2 + x_3 + \bar{x}_4$ та $x_2 + \bar{x}_4$ входять до деякої логічної функції $F = F(x_1, x_2, x_3, x_4)$, а змінні x_2 та $\bar{x}_4$ самостійно не входять, як суми, тоді сума $x_2 + \bar{x}_4$ буде простою імпліцентою функції F, оскільки $F \subset \varphi \subset x_2 + \bar{x}_4$, а $F \not\subset x_2$ і $F \not\subset \bar{x}_4$. Разом з тим, сума φ не буде простою імпліцентою, оскільки $x_2 + \bar{x}_4 \subset \varphi$.

Означення:	Кон'юнкція простих імпліцент називається ***скороченою КНФ***.

Для подання логічної функції F у вигляді скороченої КНФ використовують операції повного та неповного склеювання, а також поглинання і розгортання в КНФ.

Операція повного склеювання: $(x + y)(x + \bar{y}) = x$.

Операція неповного склеювання: $(x+y)(x+\bar{y}) = x(x+y)(x+\bar{y})$.

Операція поглинання: $x(x+y) = x \quad і \quad x(x+\bar{y}) = x$.

Іноді потрібно навпаки, зі скороченої КНФ отримати ДКНФ, тоді застосовується операція розгортання. Вона перетворює будь-яку просту імпліценту в кон'юнкцію конституент нуля.

Приклад. $x+\bar{y}-$ проста імпліцента логічної функції 4-ох аргументів x, y, z, u. Тоді, застосовуючи операцію розгортання, отримаємо

$$x+\bar{y} = (x+\bar{y}) + z\bar{z} = (x+\bar{y}+z)(x+\bar{y}+\bar{z}) =$$
$$= \big((x+\bar{y}+z) + u\bar{u}\big)\big((x+\bar{y}+\bar{z}) + u\bar{u}\big) =$$
$$= (x+\bar{y}+z+u)(x+\bar{y}+z+\bar{u})(x+\bar{y}+\bar{z}+u)(x+\bar{y}+\bar{z}+\bar{u}).$$

У процесі розгортання різні імпліценти можуть утворювати одну й ту ж саму конституенту нуля. У цьому разі на основі тотожності $xx = x$ треба залишити одну конституенту нуля. У результаті отримаємо ДКНФ початкової логічної функції.

Теореми Квайна для КНФ

1. Якщо в ДКНФ логічної функції *F* провести всі можливі операції неповного склеювання, потім всі операції поглинання, то буде одержана скорочена КНФ цієї функції, тобто, кон'юнкція всіх її простих імпліцент.

2. Якщо функція задана в довільній КНФ, то перед мінімізацією за методом Квайна її потрібно розгорнути в ДКНФ.

Загальний алгоритм мінімізації логічних функцій в ДКНФ можна отримати аналогічно алгоритму, побудованому для ДДНФ.

6.4 Застосування карт Карно для мінімізації логічних функцій

Карта Карно є одним із графічних методів подання логічних функцій.

Для функції, що залежить від *n* змінних, карта складається з 2^n клітинок, кожна з яких відповідає певному набору значень змінних. Приклад карт Карно для функцій з двома, трьома, чотирма, п'ятьма та шістьма змінними зображено на рис. 6.1.

Вхідні змінні розташовуються вздовж зовнішніх сторін карти – проти її рядків і стовпців. Значення кожної змінної поширюється на всі клітинки відповідного рядка або стовпця. Якщо проти рядка чи стовпця наведено риску з позначенням змінної, її значення в цих клітинках дорівнює одиниці. Для решти клітинок відповідне значення змінної дорівнює нулеві.

Позначення для вхідних змінних мають бути розташовані так, щоб кожна клітинка карти відповідала унікальному набору значень змінних. Важливо, щоб кожна змінна ділила площину карти на дві рівні частини: одна половина клітинок відображає значення змінної, яке дорівнює одиниці, інша – значення, що дорівнює нулю.

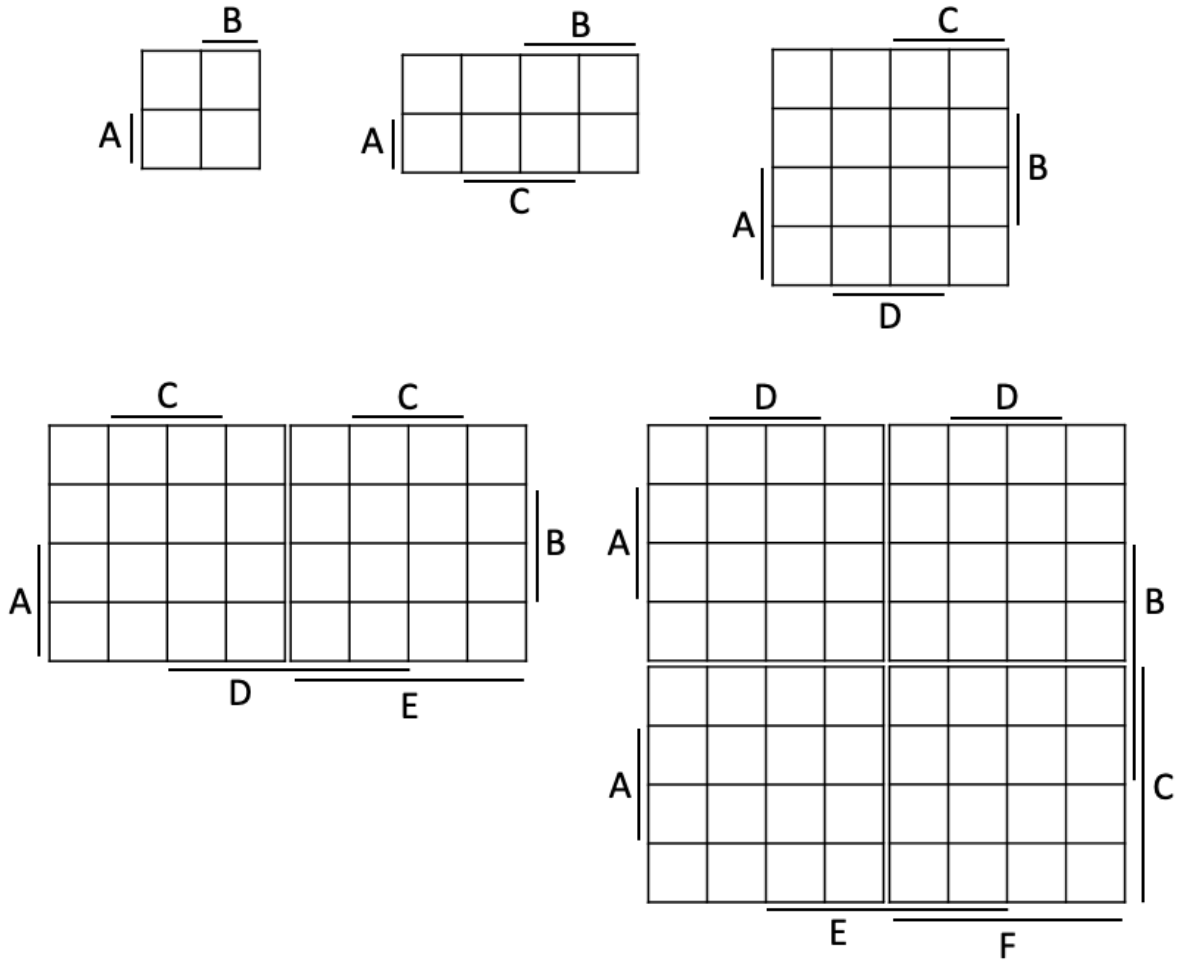


Рисунок 6.1 – Карти Карно для функцій двох, трьох, чотирьох, п’яти та шести змінних

У клітинках карти Карно записується значення логічної функції, яке вона набуває за відповідного набору вхідних змінних. Для ілюстрації процесу заповнення карти Карно розглянемо функцію трьох змінних, задану таблицею істинності (табл. 6.1). Ця функція є функцією трьох змінних.

Таблиця 6.1 – Таблиця істинності функції $F(a, b, c)$

Номер набору	Аргументи			Функція $F(a, b, c)$
	a	*b*	*c*	
1	0	0	0	1
2	0	0	1	1
3	0	1	0	0
4	0	1	1	0
5	1	0	0	0
6	1	0	1	1
7	1	1	0	1
8	1	1	1	1

Відповідна карта Карно складається з $2^n = 8$ клітинок (рис. 6.2, *а*). Клітинки карти для зручності пояснення процесу заповнення уявно пронумеруємо зліва направо у порядку зростання.

Функція *F* дорівнює одиниці для першого, другого, шостого, сьомого та восьмого наборів вхідних змінних. Першому набору змінних ($a = 0$, $b = 0$, $c = 0$) відповідає перша клітинка карти Карно, другому набору ($a = 0$, $b = 0$, $c = 1$) – друга, шостому ($a = 1$, $b = 0$, $c = 1$) – шоста, сьомому ($a = 1$, $b = 1$, $c = 0$) – восьма, восьмому ($a = 1$, $b = 1$, $c = 1$) – сьома. У першій, другій, шостій, сьомій і восьмій клітинках карти Карно ставимо одиниці, у решті клітинок – нулі. Знизу під картою записуємо позначення функції *F*. На цьому заповнення карти закінчується.

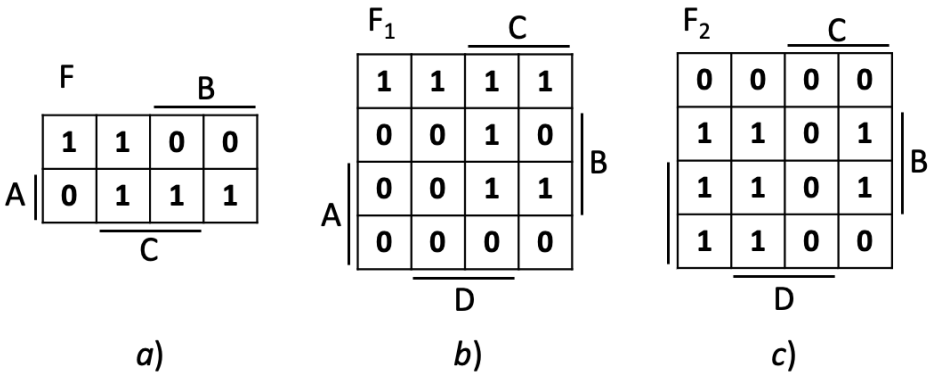


Рисунок 6.2 – Карти Карно: *а*) – для функції, що задана таблицею істинності 6.1; *b*) – для функції $F_1 = abc\ + \bar{a}\,\bar{b} + bcd$; *c*) – для функції $F_2 = (a + b)(b + \bar{c}\ + d)(c + d)$

Карту Карно можна скласти не тільки за таблицею істинності, але й за алгебраїчним виразом функції, заданої у ДНФ або КНФ. Якщо функцію задано у ДНФ, то її можна розгорнути у ДДНФ та заповнити одиницями клітинки карти, що відповідають конституентам одиниці. Проте, для заповнення карти Карно не обов'язково розгортати функцію у ДДНФ, можна заповнити одиницями зразу групи клітинок, що відповідають елементарним кон'юнкціям. Нехай, наприклад, для функції

$$F_1 = abc\ + \bar{a}\,\bar{b} + bcd$$

треба заповнити карту Карно.

Функція F_1 є функцією чотирьох змінних. Відповідну карту Карно показано на рис. 6.2, б. Елементарній кон'юнкції *abc* відповідають дві клітинки – 11-та і 12-та, кон'юнкції $\bar{a}\,\bar{b}$ відповідає група клітинок – перша, друга, третя, четверта, кон'юнкції *bcd* – клітинки сьома і 11-та. Заповнивши ці клітинки одиницями, отримаємо карту Карно, що зображує функцію F_1.

Якщо функцію задано у КНФ, то для заповнення карти Карно можна розгорнути функцію у ДКНФ і в клітинках карти, що відповідають

конституентам нуля, записати 0, а в решті клітинок – 1. Оскільки складаючи конституенти нуля, змінні, що дорівнюють нулеві у цьому наборі, записують без знаку інверсії, а змінні, що дорівнюють одиниці, – зі знаком інверсії, то, наприклад, конституенті нуля $(a + \bar{b} + c)$ відповідає клітинка, для якої a = 0, b = 1, c = 0, конституенті нуля $(\bar{a} + b + \bar{c})$ – клітинка, для якої a = 1, b = 0, c = 1 тощо.

Для заповнення карти Карно можна не розгортати функцію у ДКНФ, а заповнювати нулями зразу групи клітинок, що відповідають елементарним диз'юнкціям. Як приклад розглянемо заповнення карти Карно для функції

$$F_2 = (a + b)(b + \bar{c} + d)(\bar{c} + \bar{d}),$$

заданій у КНФ.

Елементарній диз'юнкції $(a + b)$ відповідає група клітинок, для яких $a = 0$, $b = 0$. Це перша, друга, третя та четверта клітинки карти Карно на рис. 6.2, *b)*. Диз'юнкції $(b + \bar{c} + d)$ відповідають клітинки, для яких $b = 0$, $c = 1$, $d = 0$, тобто четверта і 16-та. Диз'юнкції $(\bar{c} + \bar{d})$ відповідає група клітинок, для яких $c = 1$, $d = 1$. Це третя, сьома, 11- і 15-та клітинки. Заповнивши ці клітинки нулями, а решту – одиницями, отримаємо карту Карно, що зображує функцію F_2.

Ознайомившись з принципом заповнення карт Карно для логічних функцій, перейдемо до розгляду їх властивостей. Виявляється, що за допомогою карт Карно досить просто відшукувати кон'юнкції або диз'юнкції, до яких можна застосувати операцію склеювання і таким чином мінімізувати логічну функцію.

Розглянемо карту Карно чотирьох змінних (див. рис. 6.1) і запишемо конституенти одиниці для всіх клітинок одного рядка карти, наприклад, другого знизу, і одного стовпця, наприклад другого зліва. Тоді отримаємо:

Для клітинок у рядку:	Для клітинок у стовпці:
$a\ b\ \bar{c}\ \bar{d}$	$\bar{a}\ \bar{b}\ \bar{c}\ d$
$a\ b\ \bar{c}\ d$	$\bar{a}\ b\ \bar{c}\ d$
$a\ b\ c\ d$	$a\ b\ \bar{c}\ d$
$a\ b\ c\ \bar{d}$	$a\ \bar{b}\ \bar{c}\ d$

Аналізуючи ці конституенти, легко дійти висновку, що конституенти для клітинок, що розміщуються поряд, відрізняються значенням тільки однієї змінної, тобто для них можна застосувати операцію склеювання.

Будь-які клітинки, яким відповідають склеюванні конституенти одиниці, називаються сусідніми. Сусідніми є не тільки клітинки, що розміщені поряд в одному рядку або стовпці, але й клітинки на протилежних кінцях одного рядка або стовпця. У цьому легко переконатися, порівнявши перші й останні конституенти для клітинок у рядку або стовпці.

Для карти чотирьох змінних знаходити сусідні клітинки досить просто. Дещо складніше їх знаходити для карт п’яти, шести й більшої кількості змінних. Карту п’яти змінних можна подати у вигляді двох карт чотирьох змінних, карту шести змінних – у вигляді чотирьох карт чотирьох змінних тощо. Карти чотирьох змінних, що відрізняються значенням тільки однієї змінної, також можна назвати сусідніми. Тоді сусідні карти будуть розміщені поряд або на протилежних кінцях одного рядка або стовпця з карт чотирьох змінних і, отже, сусідніми будуть клітинки, що є сусідніми у тій же самій карті чотирьох змінних, а також клітинки у сусідніх картах, які розміщуються симетрично відносно ліній, що ділять карту великої кількості змінних на карти чотирьох змінних.

Як приклад на рис. 6.3 показано карту п’яти змінних. Лінії, які з’єднують точки, показують те, що пари стовпців є сусідніми. Наприклад, лінії у рядку $a = 0$, $b = 0$ (верхній рядок карти) показують, що кожний стовпець є сусіднім зі стовпцями, які розміщуються поряд з ним праворуч та ліворуч. У рядку $a = 0$, $b = 1$ показано, що сусідніми є крайні стовпці карт чотирьох та п’яти змінних. У рядку $a = 1$, $b = 1$ позначено як сусідні третій і шостий, другий та восьмий стовпці, тобто стовпці, які розміщуються симетрично відносно лінії, що ділить карту п’яти змінних на дві карти чотирьох змінних.

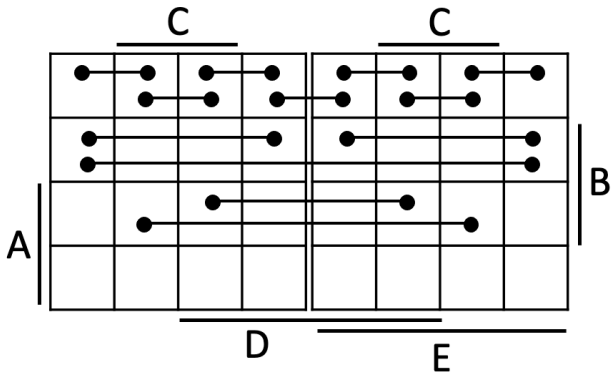


Рисунок 6.3 – До визначення сусідніх клітинок
для карти Карно п’ятьох змінних

Властивості карт Карно, які дозволяють досить просто відшуковувати сусідні клітинки, використовуються для мінімізації логічних функцій. Розглянемо, наприклад, логічну функцію, подану картою Карно (рис. 6.4, *а*). Функція F_1 у ДДНФ має вигляд

$$F_1 = \bar{a}\, b\, \bar{c}\, d\ + \bar{a}\, b\, c\, d.$$

Застосувавши теорему склеювання, отримаємо $F_1 = \bar{a}\, b\, d$. Цей результат можна отримати, не записуючи функцію у ДДНФ, таким способом. Одиниці стоять у сусідніх клітинках. Ці клітинки можна об’єднати в контур. Якщо об’єднуються дві сусідні клітинки, то

конституенти одиниці для них розрізняються тільки однією змінною: у першу конституенту одиниці входить змінна *c* з інверсією, а в другу – без неї. У загальному виразі для контуру з двох клітинок змінної *c* не має бути, оскільки заміна двох сусідніх клітинок одним контуром аналогічна застосуванню теореми склеювання.

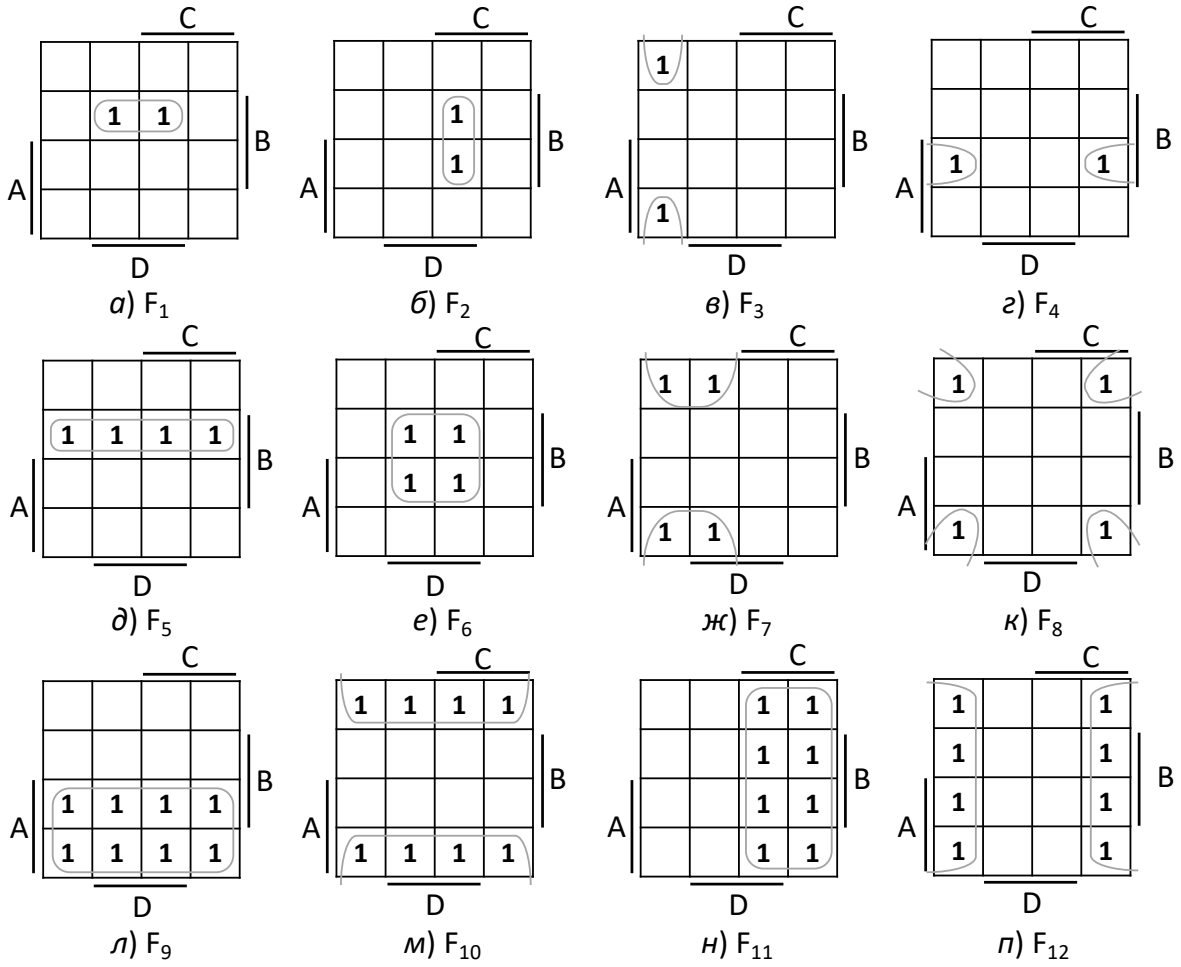


Рисунок 6.4 – Варіанти об’єднання клітинок для карт Карно чотирьох змінних

Так само для функцій, поданих картами Карно на рис. 6.4 *б*, *в*, *г*, можна записати

$$F_2 = b\ c\ d\ , \quad F_3 = \bar{b}\ \bar{c}\ \bar{d}\ , \quad F_4 = a\ b\ \bar{d}\ .$$

При цьому слід звернути увагу на те, що клітинки з одиницями у картах Карно на рис. 6.4, *в*, *г* є сусідніми, хоч і розділені просторово, і, отже, їх також можна об’єднати в один контур.

Розглянемо тепер карти Карно на рис. 6.4, *д-к*. Досконала диз’юнктивна нормальна форма функції F_5 має вигляд:

$$F_5 = \bar{a}\ b\ \bar{c}\ \bar{d}\ + \bar{a}\ b\ \bar{c}\ d + \bar{a}\ b\ c\ d\ + \bar{a}\ b\ c\ \bar{d}.$$

Застосувавши теорему склеювання, отримаємо

$$F_5 = \bar{a}\, b\, \bar{c} + \bar{a}\, b\, c = \bar{a}\, b.$$

Такий самий вираз отримаємо, якщо об'єднати чотири клітинки в контур. Для решти карт $F_6 = b\, d$, $F_7 = \bar{b}\, \bar{c}$, $F_8 = \bar{b}\, \bar{d}$.

Варто звернути увагу на те, що у вираз для контуру не входять ті змінні, межі яких перетинаються контуром. Наприклад, контур на карті, наведений на рис. 6.4, *е*, перетинає межі змінних *а* та *с*, і саме ці змінні не входять в остаточний вираз функції F_6.

У контур можна об'єднати й вісім клітинок. Приклади таких об'єднань показано на рис. 6.4, *л, м, н, п*. Функції, подані цими картами, мають вигляд:

$$F_9 = a\,,\ F_{10} = \bar{b}\,,\ F_{11} = \text{c},\ F_{12} = \bar{d}\,.$$

Розглянувши карти Карно (див. рис. 6.4), можна дійти таких висновків.

Якщо в контур об'єднано дві клітинки, то вираз для контуру містить на одну змінну менше порівняно з конституентою одиниці, якщо чотири клітинки – на дві змінні менше, вісім – на три. Взагалі, якщо контур містить 2^n клітинок, то у вираз для контуру входить на n змінних менше. Щоб використати карту Карно для мінімізації логічних функцій, потрібно побудувати карту для відповідної кількості змінних і нанести на неї задану функцію. Потім слід об'єднати сусідні клітинки з одиницями у контури, записати вирази для контурів і скласти їх диз'юнкцію.

Сусідні клітинки спочатку об'єднують у пари, потім у четвірки з сусідніх пар, тобто пар, що відрізняються тільки однією змінною, після цього сусідні четвірки об'єднують у вісімки тощо. Чим більше клітинок об'єднано у контур, тим простіший вираз, що відповідає контуру, тому слід прагнути того, щоб кожний контур мав якомога більше сусідніх клітинок. У такому разі деякі контури можуть частково перекриватися, тобто ті ж самі клітинки можуть одночасно входити у кілька контурів. У контур можна об'єднувати не будь-яку парну кількість клітинок, а тільки 2^n клітинок, тобто 2, 4, 8, 16 тощо.

Крім того, потрібно уникати створення зайвих контурів, тобто контурів, усі клітинки яких уже належать до інших контурів. Для цього об'єднання слід починати з тих одиниць, які можуть увійти тільки в один контур. Це положення ілюструється картою Карно (рис. 6.5, *а*) Контур з чотирьох одиниць тут зайвий через те, що всі клітинки цього контуру вже увійшли до інших контурів.

У контури можна об'єднувати клітинки не тільки з одиницями, але й з нулями. При цьому всі правила об'єднування залишаються попередніми, а функція записується у вигляді кон'юнкції елементарних диз'юнкцій, що відповідають контурам з нулями.

Вираз для контуру з нулями записують у вигляді диз'юнкції інверсій координат контуру. Наприклад, об'єднавши клітинки з нулями у карті Карно на рис. 6.5, *б*, отримаємо для першого контуру $\bar{b} + \bar{d}$, для другого $\bar{a} + \bar{c} + \bar{d}$, для третього $b + \bar{c} + d$. Функція у КНФ матиме вигляд:

$$F = (\bar{b} + \bar{d})(\bar{a} + \bar{c} + \bar{d})(b + \bar{c} + d).$$

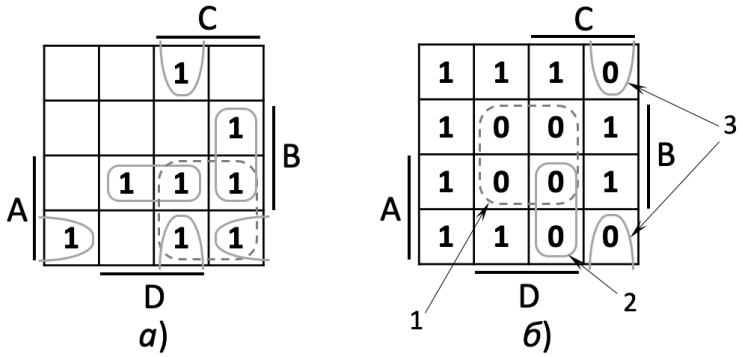


Рисунок 6.5 – Приклади об'єднання клітинок карти Карно в контури:
а) – з одиницями; *б*) – з нулями

Приклади мінімізації логічних функцій за допомогою карт Карно ілюструє рис. 6.6, на якому показано способи об'єднання клітинок з одиницями у контури. У результаті виконаних об'єднань отримано такі мінімізовані вирази функцій:

$$F_1 = \bar{a}\,d + c\,d + \bar{a}\,c + \bar{a}\,b\,\bar{c}$$
$$F_2 = b\,c + \bar{b}\,\bar{d} + a\,d\,\bar{e}$$
$$F_3 = c\,d + \bar{b}\,\bar{d} + \bar{a}\,\bar{d}$$
$$F_4 = b\,d + b\,e + \bar{c}\,e$$
$$F_5 = \bar{c}\,\bar{d}\,\bar{e} + \bar{a}\,\bar{b}\,\bar{e} + b\,\bar{c}\,\bar{e} + a\,c\,\bar{d}$$

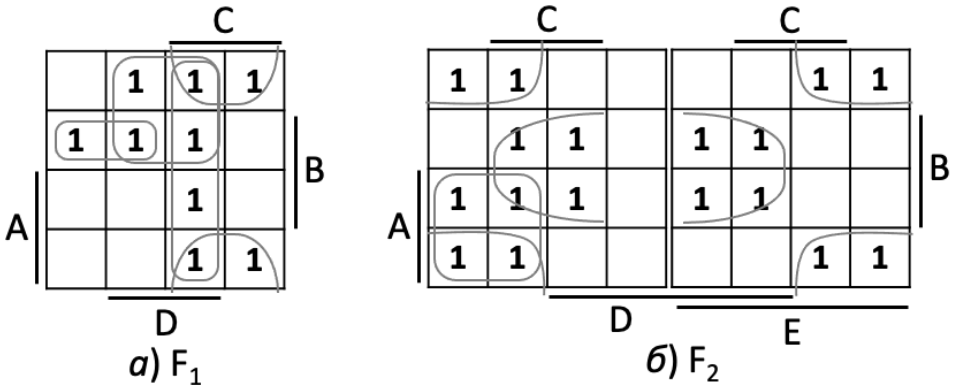


Рисунок 6.6 – Приклади мінімізації логічних функцій
за допомогою карт Карно

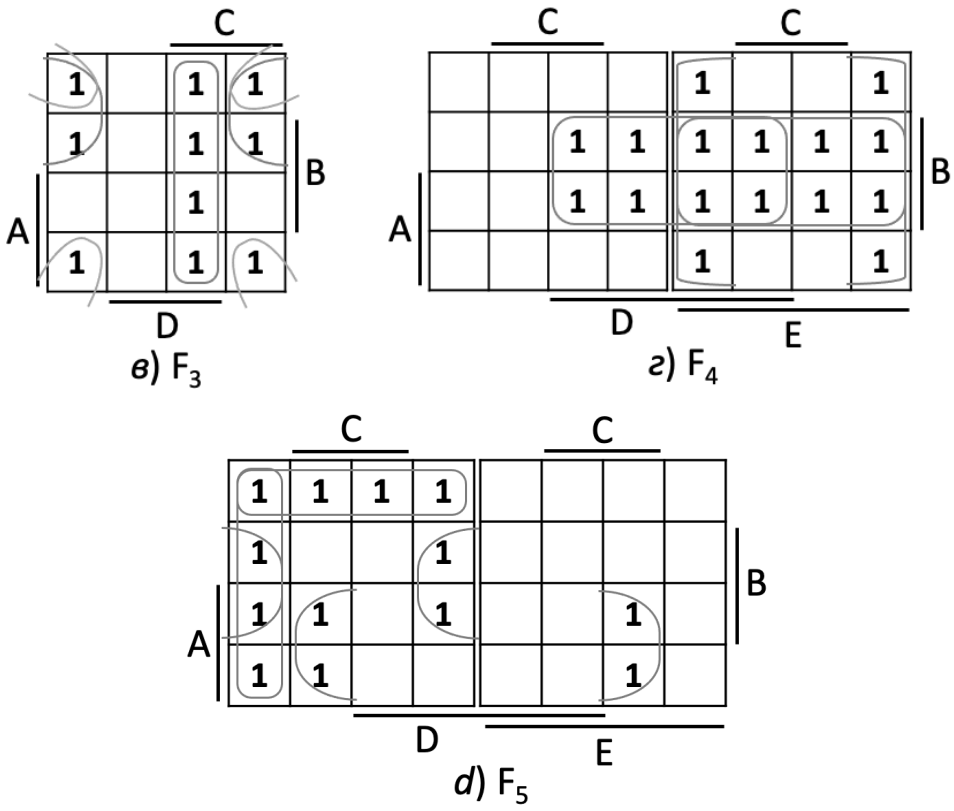


Рисунок 6.6, аркуш 2

У розглянутих прикладах функції були повністю визначеними, тобто кожному наборові вхідних змінних відповідало цілком визначене значення функції: 0 або 1. Проте, під час роботи багатьох дискретних систем керування можуть бути не всі набори вхідних змінних. Наприклад, одночасно не може бути сигналів про верхній та нижній рівні рідини, про наявність механізму у крайньому лівому і крайньому правому положеннях тощо. Набори вхідних змінних, яких за заданих умов роботи ніколи не може бути, називаються невикористаними станами входів. У клітинках карти Карно, що відповідають таким станам, проставляють риски. Функції, що описують схеми з невикористаними станами входів, називають *неповністю визначеними.*

Наявність невикористаних станів дозволяє спрощувати логічну функцію. Нехай, наприклад, функцію задано картою Карно (рис. 6.7, *а*), причому відомо, що сигнали a і c одночасно не можуть дорівнювати одиниці. У клітинках карти, що відповідають значенням $a = 1$, $c = 1$, проставлено риски. Якщо скласти вираз функції F_1, об'єднавши в контури тільки клітинки з одиницями, то отримаємо

$$F_1 = a\,\bar{d}\,\bar{c} + a\,b\,\bar{c} + \bar{a}\,b\,c. \qquad (6.1)$$

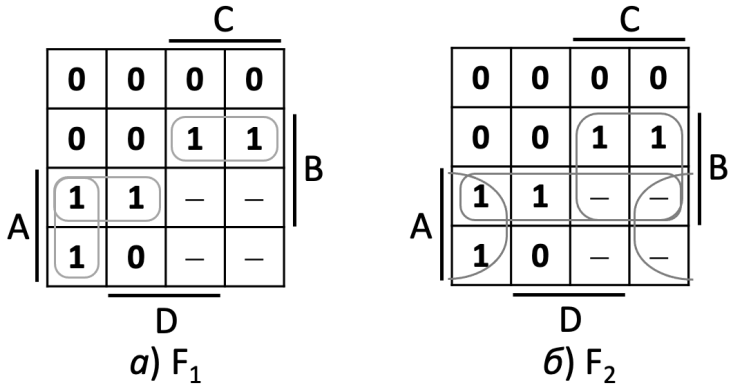


Рисунок 6.7 – Приклади об’єднання клітинок з одиницями в картах Карно: *а)* – без урахування невикористаних станів; *б)* – з їхнім урахуванням

Для спрощення функції у контури можна об’єднувати клітинки не тільки з одиницями, а й з рисками (рис. 6.7, *б*). Тоді отримаємо

$$F_2 = a\,b + a\,\bar{d} + b\,c. \qquad (6.2)$$

Функції F_1 і F_2, строго кажучи, не є рівносильними. Наприклад, якщо $a = 1$, $b = 0$, $c = 1$, $d = 0$, функція $F_1 = 0$, а $F_2 = 1$. Проте, якщо вважати, що a і c одночасно не дорівнюватимуть одиниці, то схеми, побудовані за формулами (6.1) та (6.2), працюватимуть однаково.

Відомі також інші методи мінімізації логічних функцій. Однак карти Карно, завдяки наочності та простоті, мають незаперечні переваги порівняно з іншими методами. Їх найчастіше застосовують для мінімізації логічних функцій з порівняно невеликою кількістю змінних (до шести).

Питання для самоконтролю

1) Що таке диз’юнктивна нормальна форма (ДНФ) та кон’юнктивна нормальна форма (КНФ)?
2) Які переваги та недоліки використання ДНФ та КНФ у поданні логічних функцій?
3) Наведіть означення досконалій та скороченій ДНФ.
4) Наведіть означення досконалій та скороченій КНФ.
5) Що таке імпліканта функції та проста імпліканта функції?
6) Як знаходяться первинна та істотна імпліканти функції?
7) Що таке імпліцента функції?
8) Що таке первинна та істотна імпліценти функції?
9) У чому суть методу Квайна для мінімізації логічних функцій?
10) Як визначити істотні імпліканти серед усіх первинних?
11) Як викреслюються зайві первинні імпліканти під час мінімізації?
12) Що таке карта Карно? Для чого вона використовується?
13) Як заповнюється карта Карно для заданої логічної функції?

14) Що таке групування сусідніх клітин у карті Карно?

15) Як карта Карно допомагає знаходити імпліканти логічної функції?

16) Як мінімізувати логічну функцію за допомогою карти Карно?

17) Які переваги використання карт Карно для мінімізації логічних функцій? Які обмеження мають карти Карно?

18) Яка різниця між мінімізацією функції методом Квайна та за допомогою карт Карно?

19) Як змінюється розмір карти Карно залежно від кількості змінних?

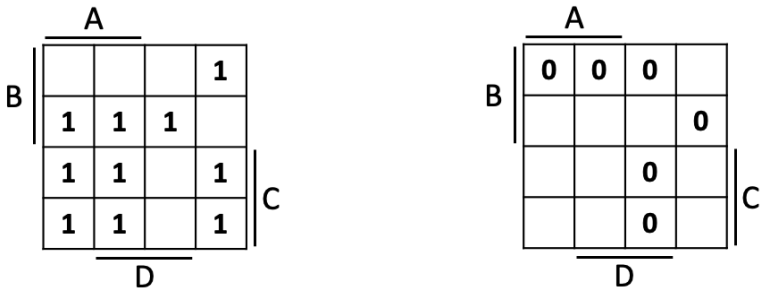
20) Чи можуть логічні функції мати більше ніж одну мінімальну форму? Чому?

Індивідуальні практичні завдання № 6

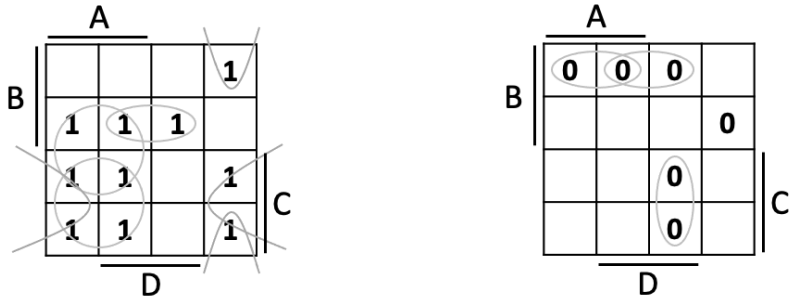
Задача 1. Мінімізувати логічну функцію, задану у таблиці 5.1 (С. 69) за допомогою карт Карно у ДНФ та КНФ та порівняти між собою отримані мінДНФ і мінКНФ за складністю.

Приклад. Мінімізуємо за допомогою карт Карно логічну функцію $f = B \to C \equiv A \vee (\overline{B} \oplus D)$, для якої отримано таблицю істинності в прикладі індивідуального практичного заняття № 5 (С. 69).

На початковому етапі побудуємо карту Карно для чотирьох змінних та нанесемо на неї задану логічну функцію *f*:



Після цього об’єднаємо заповнені сусідні клітинки у контури, намагаючись охопити якомога більшу кількість клітинок. Зазначимо, що деякі контури в такому разі можуть частково перекриватися, а об’єднувати можна лише 2^n клітинок, тобто 2, 4, 8, 16 тощо:



Визначимо мінДНФ: $f = ac + a\bar{b} + \bar{b}\,\bar{d} + \bar{a}\bar{c}\bar{d} + bcd$;

мінКНФ: $f = (\bar{a} + \bar{b} + c)(\bar{b} + c + \bar{d})(a + b + \bar{d})(a + \bar{b} + \bar{c} + d)$.

Як бачимо, мінДНФ на 1 літеру є раціональнішою за КНФ

Задача 2. Мінімізувати логічну функцію, задану у табл. 6.2 методом Квайна.

Таблиця 6.2 – Варіанти завдань

Варіант	Функція	Варіант	Функція
1	2	3	4
1.	F = (0, 1, 2, 3, 5, 7, 10, 11, 14)	31.	F = (0, 2, 4, 5, 6, 8, 11, 13)
2.	F = (1, 2, 3, 5, 10, 14, 15)	32.	F = (1, 2, 3, 4, 5, 10, 11, 12, 13)
3.	F = (0, 1, 7, 11, 12, 13, 15)	33.	F = (0, 1, 2, 7, 8, 9, 10, 11)
4.	F = (1, 3, 5, 6, 8, 9, 11, 15)	34.	F = (0, 1, 2, 4, 5, 7, 8, 15)
5.	F = (2, 4, 6, 8, 10, 12, 14, 15)	35.	F = (1, 3, 4, 7, 8, 10, 13, 15)
6.	F = (0, 1, 2, 4, 7, 10, 12, 13, 14)	36.	F = (1, 2, 4, 5, 10, 11, 14)
7.	F = (1, 2, 3, 4, 7, 10, 12, 15)	37.	F = (0, 1, 3, 6, 7, 9, 10, 13, 15)
8.	F = (3, 4, 5, 7, 10, 11, 14, 15)	38.	F = (1, 2, 5, 6, 9, 10, 11, 12)
9.	F = (5, 7, 8, 10, 11, 12, 14, 15)	39.	F = (0, 1, 4, 5, 7, 9, 11, 13)
10.	F = (2, 5, 6, 10, 12, 13, 14)	40.	F = (1, 5, 6, 10, 12, 13, 14, 15)
11.	F = (1, 2, 3, 4, 6, 13, 14, 15)	41.	F = (2, 5, 6, 10, 12, 14, 15)
12.	F = (1, 3, 7, 9, 10, 12, 14, 15)	42.	F = (0, 1, 3, 9, 10, 11, 12, 15)
13.	F = (1, 6, 7, 8, 9, 11, 13, 15)	43.	F = (1, 2, 3, 6, 7, 9, 11, 13, 15)
14.	F = (3, 4, 5, 6, 12, 13, 14, 15)	44.	F = (0, 1, 2, 3, 6, 7, 12, 14, 15)
15.	F = (3, 6, 9, 10, 11, 12, 13, 14)	45.	F = (3, 6, 8, 9, 11, 13, 14, 15)
16.	F = (2, 5, 6, 9, 11, 12, 14)	46.	F = (1, 2, 3, 4, 5, 10, 11, 12)
17.	F = (0, 3, 6, 7, 8, 9, 12, 13, 15)	47.	F = (2, 5, 7, 9, 11, 13, 14, 15)
18.	F = (2, 4, 5, 8, 9, 11, 12, 13)	48.	F = (2, 5, 6, 10, 12, 13, 14, 15)
19.	F = (0, 3, 4, 7, 8, 13, 14, 15)	49.	F = (1, 2, 4, 6, 8, 10, 13, 14)
20.	F = (2, 3, 4, 6, 8, 12, 13, 14, 15)	50.	F = (0, 1, 3, 9, 10, 11, 12, 14)
21.	F = (0, 1, 3, 4, 7, 8, 9, 12, 13)	51.	F = (3, 5, 6, 7, 8, 9, 11, 15)
22.	F = (0, 1, 3, 4, 5, 7, 8, 9, 13)	52.	F = (0, 5, 6, 8, 9, 10, 11, 13, 14)
23.	F = (0, 1, 2, 4, 5, 6, 7, 8, 14)	53.	F = (0, 1, 2, 6, 8, 9, 10, 13, 15)
24.	F = (0, 4, 5, 8, 9, 10, 12, 13, 14)	54.	F = (0, 1, 3, 4, 6, 8, 9, 13)
25.	F = (4, 5, 8, 9, 10, 11, 14, 15)	55.	F = (1, 3, 4, 7, 8, 9, 11, 13, 15)
26.	F = (0, 1, 2, 3, 4, 5, 7, 10, 12)	56.	F = (0, 5, 7, 8, 9, 10, 11, 12)
27.	F = (0, 6, 7, 8, 9, 11, 13, 14, 15)	57.	F = (2, 4, 5, 7, 9, 11, 13, 15)
28.	F = (0, 1, 3, 4, 6, 10, 11, 12, 14)	58.	F = (2, 5, 6, 10, 12, 14, 15)
29.	F = (0, 1, 3, 4, 7, 8, 9, 11)	59.	F = (2, 4, 5, 6, 7, 9, 11, 13, 15)
30.	F = (0, 1, 3, 5, 7, 9, 10, 11, 12)	60.	F = (1, 2, 3, 5, 7, 13, 14, 15)

Приклад. Метод Квайна виконується у декілька етапів, розглянемо його застосування на конкретному прикладі.

Нехай потрібно мінімізувати логічну функцію, задану у вигляді

$$f(x_1, x_2, x_3, x_4) = V(3, 4, 5, 7, 9, 11, 12, 13) = \bar{x}_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee x_1\bar{x}_2x_3x_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_3x_4$$

Етап 1. Знаходження первинних імплікант. Складаємо табл. 6.3 та знаходимо імпліканти четвертого та третього рангу, тобто знижуємо ранг членів, що входять в СНДФ.

Складаємо табл. 6.4, яка містить всі терми, що не піддалися поглинанню, а також первинні імпліканти третього рангу. Складання таблиць продовжується доти, доки не вдасться застосувати правило поглинання: $Fx_i \vee F\bar{x}_i = F$. У цьому випадку можемо дійти до первинної імпліканти другого рангу – $x_2\bar{x}_3$ (табл. 6.4). В табл. 6.4 також виділені інші первинні імпліканти найменшого рангу.

Етап 2. Розставляння міток. Складаємо додаткову таблицю, число рядків якої дорівнює числу одержаних первинних імплікант, а число стовпців збігається з числом мінтермів СНДФ. Якщо в один з мінтермів СНДФ входить деяка з первинних імплікант, то на перетині відповідного стовпця і рядка виставляється мітка (табл. 6.5).

Етап 3. Знаходження істотних імплікант. Якщо в деякому стовпці табл. 6.5 знаходиться лише одна мітка, то первинна імпліканта у відповідному рядку є суттєвою, оскільки без неї не може бути отримана вся множина заданих мінтермів. У табл. 6.5 істотною імплікантою є терм $x_2\bar{x}_3$. Стовпці, відповідні істотним імплікантам, з таблиці вилучаються.

Етап 4. Викреслювання зайвих стовпців. Після третього етапу в результаті викреслювання стовпців 2, 3, 7 і 8 отримуємо табл. 6.6. Якщо в таблиці є два стовпці, в яких є мітки в однакових рядках, то один з них викреслюється. Покриття стовпця, який залишився, здійснюватиме відкинутий мінтерм. У нашому прикладі такого випадку немає.

Етап 5. Викреслювання зайвих первинних імплікант. Якщо після викреслювання деяких стовпців на етапі 4 в табл. 6.6 з'являються рядки, в яких немає жодної мітки, то первинні імпліканти, відповідні цим рядкам, в подальшому не розглядаються, оскільки вони не покривають мінтерми, шо залишилися в розгляді.

Етап 6. Вибір мінімального покриття. У табл. 6.6 вибирається така сукупність первинних імплікант, яка містить принаймні по одній мітці в кожному стовпці. У декількох можливих варіантах такого вибору віддається перевага варіанту покриття з мінімальним сумарним числом букв в імплікантах, що входять у покриття. Цю вимогу задовольняють первинні імпліканти $\bar{x}_1x_3x_4$ та $x_1\bar{x}_2x_4$.

Отже, мінімальна форма заданої функції складається з суми істотних імплікант (етап 3) і первинних імплікант, які покривають мінтерми, що залишилися (етап 6):

$$f(x_1, x_2, x_3, x_4) = V(3, 4, 5, 7, 9, 11, 12, 13) = x_2\bar{x}_3 \vee \bar{x}_1x_3x_4 \vee x_1\bar{x}_2x_4.$$

Таблиця 6.3

Початкові терми	1	2	3	4	5	6	7	8
	$\bar{x}_1\bar{x}_2x_3x_4$	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\bar{x}_1x_2\bar{x}_3x_4$	$\bar{x}_1x_2x_3x_4$	$x_1\bar{x}_2\bar{x}_3x_4$	$x_1\bar{x}_2x_3x_4$	$x_1x_2\bar{x}_3\bar{x}_4$	$x_1x_2\bar{x}_3x_4$
$\bar{x}_1\bar{x}_2x_3x_4$	1			$\bar{x}_1x_3x_4$		$\bar{x}_2x_3x_4$		
$\bar{x}_1x_2\bar{x}_3\bar{x}_4$		1	$\bar{x}_1x_2\bar{x}_3$				$x_2\bar{x}_3\bar{x}_4$	
$\bar{x}_1x_2\bar{x}_3x_4$		$\bar{x}_1x_2\bar{x}_3$	1	$\bar{x}_1x_2x_4$				$x_2\bar{x}_3x_4$
$\bar{x}_1x_2x_3x_4$	$\bar{x}_1x_3x_4$		$\bar{x}_1x_2x_4$	1				
$x_1\bar{x}_2\bar{x}_3x_4$					1	$x_1\bar{x}_2x_4$		$x_1\bar{x}_3x_4$
$x_1\bar{x}_2x_3x_4$	$\bar{x}_2x_3x_4$				$x_1\bar{x}_2x_4$	1		
$x_1x_2\bar{x}_3\bar{x}_4$		$x_2\bar{x}_3\bar{x}_4$					1	$x_1x_2\bar{x}_3$
$x_1x_2\bar{x}_3x_4$			$x_2\bar{x}_3x_4$		$x_1\bar{x}_3x_4$		$x_1x_2\bar{x}_3$	1

Таблиця 6.4

Первинні імпліканти рангу 3	$\bar{x}_1x_3x_4$	$\bar{x}_2x_3x_4$	$\bar{x}_1x_2\bar{x}_3$	$x_2\bar{x}_3\bar{x}_4$	$\bar{x}_1x_2x_4$	$x_2\bar{x}_3x_4$	$x_1\bar{x}_2x_4$	$x_1\bar{x}_3x_4$	$x_1x_2\bar{x}_3$
$\bar{x}_1x_3x_4$	1								
$\bar{x}_2x_3x_4$		1							
$\bar{x}_1x_2\bar{x}_3$			1						$x_2\bar{x}_3$
$x_2\bar{x}_3\bar{x}_4$				1		$x_2\bar{x}_3$			
$\bar{x}_1x_2x_4$					1				
$x_2\bar{x}_3x_4$				$x_2\bar{x}_3$		1			
$x_1\bar{x}_2x_4$							1		
$x_1\bar{x}_3x_4$								1	
$x_1x_2\bar{x}_3$			$x_2\bar{x}_3$						1

Таблиця 6.5

Первинні імпліканти	1	2	3	4	5	6	7	8
	Початкові терми							
	$\bar{x}_1\bar{x}_2x_3x_4$	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\bar{x}_1x_2\bar{x}_3x_4$	$\bar{x}_1x_2x_3x_4$	$x_1\bar{x}_2\bar{x}_3x_4$	$x_1\bar{x}_2x_3x_4$	$x_1x_2\bar{x}_3\bar{x}_4$	$x_1x_2\bar{x}_3x_4$
$\bar{x}_1x_3x_4$	∨			∨				
$\bar{x}_2x_3x_4$	∨					∨		
$\bar{x}_1x_2x_4$			∨	∨				
$x_1\bar{x}_2x_4$					∨	∨		
$x_1\bar{x}_3x_4$					∨		∨	∨
$x_2\bar{x}_3$		∨	∨				∨	∨

Таблиця 6.6

Первинні імпліканти	1	4	5	6
	Початкові терми			
	$\bar{x}_1\bar{x}_2x_3x_4$	$\bar{x}_1x_2x_3x_4$	$x_1\bar{x}_2\bar{x}_3x_4$	$x_1\bar{x}_2x_3x_4$
$\bar{x}_1x_3x_4$	∨	∨		
$\bar{x}_2x_3x_4$	∨			∨
$\bar{x}_1x_2x_4$		∨		
$x_1\bar{x}_2x_4$			∨	∨
$x_1\bar{x}_3x_4$			∨	

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Практичне засвоєння дискретної математики є ключовим етапом навчання майбутніх фахівців у галузі інформаційних технологій. Теоретичні основи, які розглядаються в межах цього посібника, закладають фундамент для розуміння основних понять і методів. Лабораторний практикум дозволить поглибити отримані знання, закріпити їх через практичні завдання та набути навичок застосування інструментів програмування для розв'язання задач дискретної математики.

Лабораторні роботи пропонується виконувати з використанням мови програмування Python та інтерактивного середовища Jupyter Notebook. Такий підхід обрано через його зручність, інтерактивність і широкі можливості для реалізації математичних концепцій та алгоритмів.

Мова програмування Python є популярним інструментом у сфері комп'ютерних наук і програмної інженерії завдяки її зрозумілому синтаксису та потужному функціоналу. У контексті дискретної математики Python є ефективним засобом для виконання складних обчислень, візуалізації даних та автоматизації процесів.

Python надає багатий набір бібліотек для математичних обчислень і роботи з множинами, графами, послідовностями тощо. Використання цієї мови у практикумі допоможе не лише виконувати лабораторні завдання, а й підготуватися до застосування програмування в більш складних наукових і прикладних задачах.

Jupyter Notebook є інтерактивним середовищем для виконання програмного коду, яке поєднує текстові пояснення, результати виконання та графічну візуалізацію в одному документі. Ця особливість робить Jupyter Notebook ідеальним інструментом для навчання та досліджень.

У рамках лабораторного практикуму Jupyter Notebook використовується як основне середовище для розробки та тестування програмного коду. Воно дозволяє:

- інтерактивно виконувати окремі частини коду;
- переглядати результати обчислень у реальному часі;
- створювати чітко структуровані документи з теоретичними поясненнями та практичними розв'язаннями.

Однак студенти можуть обирати й інші мови програмування чи середовища для виконання пропонованих завдань. При цьому реалізація завдань має відповідати поставленим цілям і забезпечувати коректне виконання усіх математичних обчислень.

Звіт про виконання лабораторної роботи потрібно оформлювати на аркушах формату А4 (210×297 мм), дотримуючись таких розмірів берегів: лівий – 30 мм, верхній і нижній – 20 мм, правий – 10 мм. Шрифт документа має бути Times New Roman розміром 14 пт, з міжрядковим

інтервалом 1,5. Абзацний відступ – 1,25 см, текст вирівняний по ширині. Нумерація сторінок починається з першої сторінки (титульна сторінка не нумерується, але вважається першою).

Усі графіки, діаграми та схеми мають бути підписані та розміщені після їхнього згадування у тексті. Наприклад, «Рисунок 1 – Графік функції f(x)». Перед рисунком та після назви рисунка потрібно залишити порожній рядок.

Цифровий матеріал, який подається у вигляді таблиці, підписується, наприклад, «Таблиця 1 – Результати тестування програми» з абзацу, а другий рядок вирівнюється по лівому краю. Перед назвою таблиці та після таблиці потрібно залишити порожній рядок.

Звіт з виконання лабораторної роботи має містити такі основні складові:

1. Титульний аркуш.
2. Мета роботи.
3. Варіант та завдання для виконання.
4. Скріншот ручного виконання завдання.
5. Блок-схема алгоритму програми.
6. Опис програми.
7. Результати тестування програми.
8. Висновки.
9. Додатки (інструкція користувача, лістинг програми, тощо)

Лабораторна робота № 1

Тема: Множини. Основні поняття та операції над множинами.

Мета роботи: Набути навиків реалізації основних операцій над множинами мовою програмування Python.

Методичні рекомендації

Множини – це одна з базових структур даних у Python, яка призначена для зберігання лише унікальних елементів. Вони забезпечують високу швидкодію при виконанні таких операцій, як об'єднання, перетин, різниця та симетрична різниця. Крім того, множини особливо зручні для швидкої перевірки наявності елемента в колекції.

Множини в Python створюються за допомогою конструктора `set()` або використанням фігурних дужок `{}` (табл. Л1.1).

Таблиця Л1.1 – Основні операції над множинами

Операція	Синтаксис
Створення множини *A*	`A = {1, 2, 3}` `B = set{1, 2, 3}`
Додавання елемента до множини *A*	`A.add(4)`
Видалення елемента з множини *A*	`A.remove(2)`
Об'єднання множин	`C = A | B` `C = A.union(B)`
Перетин множин	`C = A & B` `C = A.intersection(B)`
Різниця множин	`C = A - B` `C = A.difference(B)`
Симетрична різниця	`A ^ B` `A.symmetric_difference(B)`
Перевірка чи є *A* підмножиною *B*	`A <= B` `A.issubset(B)`
Перевірка чи є *B* є надмножиною *A*	`B >= A` `B.issuperset(A)`

Елементи множини мають бути незмінними (тобто типами, які можна хешувати, наприклад числа, рядки, кортежі), однак сама множина є змінною, тобто її можна доповнювати чи змінювати.

Python підтримує такі основні операції над множинами (табл. Л1.1):

1. Об’єднання множин (`|` або `union`) створює нову множину, яка містить усі унікальні елементи з обох множин.
2. Перетин множин (`&` або `intersection`) повертає елементи, спільні для двох множин.
3. Різниця множин (`-` або `difference`) містить лише ті елементи, які є в одній множині, але відсутні в іншій.
4. Симетрична різниця множин (`^` або `symmetric_difference`) створює множину, яка містить всі елементи, що є унікальними для кожної множини.

Python також підтримує функції для перевірки таких підмножин та надмножин, як `issubset` і `issuperset`. Зокрема,

- Оператор <= означає «підмножина або рівна» (аналог `issubset`).
- Оператор < означає «строго підмножина» (тобто підмножина, але не рівна).
- Оператор >= означає «надмножина або рівна» (аналог `issuperset`).
- Оператор > означає «строго надмножина» (надмножина, але не рівна).

Операції над множинами дозволяють виконувати багато таких практичних завдань, як порівняння даних; фільтрація унікальних елементів; знаходження спільних або відмінних характеристик у наборах даних тощо.

Завдання до лабораторної роботи

Завдання 1. Задайте універсальну множину U, яка містить 10 довільних елементів. Для парних варіантів елементами множини U є літери алфавіту, а для непарних – натуральні числа. Використовуючи універсальний набір елементів U створити довільно множини *A, B, C* та *D* за умови, що $|A| = n$, $|B| = m$, $|C| = (n - k)$, $|D| = 5$ (n – кількість літер імені здобувача вищої освіти, m – номер групи, k – номер підгрупи). Для заданих множин *A, B, C* та *D* виконайте наведені у табл. 1.1 (С. 19) (індивідуальні практичні задання № 1) операції та перевірте результат ручним розрахунком.

Завдання 2. За допомогою мови програмування Python перевірте виконання індивідуальних практичних завдань № 1.

Завдання 3. Використовуючи діаграми Венна (кола Ейлера), розв’яжіть задачу про порівняння трьох мобільних додатків.

Опитано n респондентів щодо використання трьох мобільних додатків: A – для навчання мов, B – для фізичних вправ, C – для фінансового планування. За результатами анкетування встановлено, що: n_1 респондентів використовують додаток A, n_2 респондентів використовують додаток B та n_3 респондентів використовують додаток C. Крім того, відомо що n_{12} респондентів використовують одночасно додатки A та B, n_{13} респондентів – додатки A та C, n_{23} респондентів – B та C, n_{123} респондентів

використовують одночасно всі три додатки. Яка загальна кількість із опитаних респондентів використовують принаймні один з цих додатків? Скільки респондентів використовують лише один конкретний додаток? Скільки респондентів не користуються вказаними додатками? Результати опитування наведено у табл. Л1.2

Таблиця Л1.2 – Варіанти завдань

Варіант	n_1	n_2	n_3	n_{12}	n_{13}	n_{23}	n_{123}	n
1.	10	15	20	4	3	5	2	50
2.	12	18	22	5	4	6	3	50
3.	15	20	25	6	5	7	4	50
4.	8	12	16	3	2	4	1	40
5.	20	25	30	7	6	8	5	70
6.	18	22	28	6	5	7	4	70
7.	14	19	23	5	4	6	3	70
8.	9	14	18	4	3	5	2	50
9.	16	21	26	6	5	7	4	50
10.	10	13	17	3	2	4	1	40
11.	11	16	20	4	3	5	2	40
12.	13	17	21	5	4	6	3	40
13.	14	18	22	5	4	6	3	50
14.	19	23	27	7	6	8	5	100
15.	22	27	32	8	7	9	6	100
16.	17	21	25	6	5	7	4	55
17.	15	19	24	5	4	6	3	50
18.	20	26	31	8	7	9	6	80
19.	11	15	19	4	3	5	2	35
20.	18	23	28	7	6	8	5	60
21.	21	26	30	8	7	9	6	60
22.	12	17	22	5	4	6	3	50
23.	19	24	29	7	6	8	5	60
24.	13	18	23	5	4	6	3	42
25.	10	16	21	4	3	5	2	37
26.	14	19	24	6	5	7	4	50
27.	16	22	27	7	6	8	5	50
28.	17	23	28	7	6	8	5	60
29.	22	27	33	9	8	10	7	100
30.	16	21	26	6	5	7	4	50

Порядок виконання лабораторної роботи

Завдання 1.

1. У робочому вікні Jupyter.Notebook задаємо універсальну множину та вводимо вхідні параметри для визначення потужностей множин *A, B, C* та *D* (рис. Л1.1). Результат виконання програми наведено на рис. Л1.2.

```
# Задання універсальної множини U
U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

# Задання параметрів множин
name = input("Введіть ваше ім'я: ")
group = int(input("Введіть номер групи: "))
group_h = int(input("Введіть номер підгрупи: "))

# Обчислення розмірів множин
n = len(name)  # Потужність множини A
m = group  # Потужність множини B
k = len(name) - group_h # Потужність множини C

# Виведення результатів
print(f"Потужність множини A: |A| = {n}")
print(f"Потужність множини B: |B| = {m}")
print(f"Потужність множини C: |C| = {m}")
print(f"Потужність множини D: |D| = 5")
```

Рисунок Л1.1

```
Введіть ваше ім'я:  Сергій
Введіть номер групи:  3
Введіть номер підгрупи:  2
Потужність множини A: |A| = 6
Потужність множини B: |B| = 3
Потужність множини C: |C| = 3
Потужність множини D: |D| = 5
```

Рисунок Л1.2

2. Задаємо множини *A, B, C, D* визначеної потужності та обчислюємо, згідно з заданим варіантом невідому множину *X* (рис. Л1.3). Результат виконання програми наведено на рис. Л1.4.

```
# Задаємо множини, що відповідають умові
A = {1, 2, 3, 6, 7, 9}
B = {3, 8, 9}
C = {1, 9, 10}
D = {2, 4, 6, 7, 8}

# Виведення результатів
print(f"Універсальна множина U: {U}")
print(f"Множина A (|A| = {n}): {A}")
print(f"Множина B (|B| = {m}): {B}")
print(f"Множина C (|C| = {m}): {C}")
print(f"Множина D (|D| = 5): {D}")

# Виконуємо дії над множинами
X = (A | B)&(C - D)
print(f"X = (A ∪ B) ∩ (C \ D) = {X}")
```

Рисунок Л1.3

```
Множина A (|A| = 6): {1, 2, 3, 6, 7, 9}
Множина B (|B| = 3): {8, 9, 3}
Множина C (|C| = 3): {1, 10, 9}
Множина D (|D| = 5): {2, 4, 6, 7, 8}
X = (A ∪ B) ∩ (C \ D) = {1, 9}
```

Рисунок Л1.4

3. Перевіряємо результат виконання програми, виконавши ручний розрахунок множини X.

Завдання 2. Аналогічно виконуємо програмну перевірку виконання індивідуальних практичних завдань № 1 для власного варіанта.

Завдання 3. Використаємо діаграми Венна (кола Ейлера), щоб розв'язати наведену далі задачу: серед 100 випускників школи, що брали участь у ЗНО, 19 складали біологію, 21 – хімію, 23 – географію, 7 – біологію і хімію, 9 – біологію і географію, 8 – хімію і географію. Скільки учнів не вибрали жодне з вказаних ЗНО, якщо всі вказані ЗНО вибрало 3 учні?

1. У робочому вікні Jupyter.Notebook імпортуємо бібліотеку Matplotlib та інструмент для побудови діаграм Венна (venn3):

```
from matplotlib_venn import venn3
import matplotlib.pyplot as plt
```

2. Вводимо вхідні дані задачі та позначаємо відповідні області на діаграмах Венна для візуалізації взаємозв'язків між множинами (рис. Л1.5). Загальний розподіл учасників ЗНО наведено на рис. Л1.6.

```
# Вихідні дані
students_total = 100
biology = 19
chemistry = 21
geography = 23
biology_and_chemistry = 7
biology_and_geography = 9
chemistry_and_geography = 8
all_three = 3

# Побудова діаграми Венна
venn = venn3(
    subsets={
        '100': biology - biology_and_chemistry - biology_and_geography + all_three,
        '010': chemistry - biology_and_chemistry - chemistry_and_geography + all_three,
        '001': geography - biology_and_geography - chemistry_and_geography + all_three,
        '110': biology_and_chemistry - all_three,
        '101': biology_and_geography - all_three,
        '011': chemistry_and_geography - all_three,
        '111': all_three
    },
    set_labels=('Біологія', 'Хімія', 'Географія')
)

plt.title("Розподіл учнів за вибраними предметами ЗНО")
plt.show()
```

Рисунок Л1.5

3. Аналіз діаграм на рис. Л1.6 дозволяє зробити висновок, що лише один предмет ЗНО складали 24 учні (лише біологію – 6 учнів, лише хімію

та лише географію – по 9 учнів). Загалом 42 учасники обрали принаймні один предмет, тоді як жодної дисципліни не обрали 58 осіб.

Рисунок Л1.6 – Діаграми Венна

Контрольні питання

1) Як створити множину в Python? Наведіть приклади.
2) Чим відрізняються методи add() і update() у роботі з множинами?
3) Як перевірити, чи є один набір елементів підмножиною або надмножиною іншого?
4) Які функції або методи використовуються для видалення елементів із множини?
5) Що станеться, якщо спробувати додати до множини елемент, який уже існує?
6) Чи можна використовувати списки або інші змінні типи даних як елементи множини?
7) Як знайти кількість елементів у множині? Яка функція для цього використовується?
8) Як реалізувати перевірку наявності елемента в множині?
9) Що таке порожня множина, і як її створити в Python?
10) Як можна використати множини для виявлення унікальних елементів у списку?
11) Які переваги множин у Python порівняно з іншими структурами даних?
12) Чим множини відрізняються від списків і словників у Python?
13) Як множини можна застосовувати для розв’язання реальних завдань?

Лабораторна робота № 2

Тема: Задача про покриття. Основні методи розв'язання задачі про покриття на множинах.

Мета роботи: Набути практичних навичок реалізації основних методів розв'язання задачі про покриття на множинах за допомогою мови програмування Python.

Методичні рекомендації

Задача про покриття є моделлю великої кількості оптимізаційних задач дискретної математики. Теоретичні та практичні підходи, що використовуються для її розв'язання, детально розглянуто у третій темі цього посібника.

Метод повного перебору дозволяє отримати всі можливі покриття, зокрема надлишкові, для будь-якої задачі про покриття. Однак цей метод не є раціональним, оскільки потребує значних обчислювальних ресурсів.

Програмно реалізувати даний метод можна за допомогою функції `combinations`, яка дозволяє поступово перебрати всі можливі комбінації підмножин. Приклад коду з використанням цієї функції наведено на рис. Л2.1.

```python
from itertools import combinations

# Універсальна множина
U = {1, 2, 3, 4, 5, 6, 7, 8}

# Підмножини
A = [
    {1, 2, 4, 6, 5},    # A1
    {1, 2, 3, 4},       # A2
    {2, 4, 5, 6, 7, 8}, # A3
    {1, 2, 4, 7, 8}     # A4
]

covered_sets = []

# Знайдемо всі комбінації підмножин, які покривають універсальну множину
for r in range(1, len(A) + 1):
    for subset_combination in combinations(A, r):
        union_result = set().union(*subset_combination)
        if union_result == U:
            covered_sets.append(subset_combination)

# Форматуємо результат для виводу у вигляді об'єднання підмножин
formatted_results = []
for combination in covered_sets:
    subsets_indices = [f"A{A.index(subset) + 1}" for subset in combination]
    formatted_results.append(" ∪ ".join(subsets_indices))

print("Покриттям є: ")
for result in formatted_results:
    print(result)
```

Рисунок Л2.1 – Метод повного перебору

На початку коду імпортується функція `combinations`, після чого задається універсальна множина U та всі підмножини A_i. Далі, за допомогою циклу, для кожної комбінації підмножин обчислюється їхнє об'єднання з подальшою перевіркою рівності U. Якщо об'єднання дорівнює U, то ця комбінація додається до списку покриттів. У результаті виконання коду подається список у вигляді об'єднання підмножин, які є покриттям універсальної множини U (рис. Л2.2).

```
Покриттям U є:
A2 ∪ A3
A1 ∪ A2 ∪ A3
A1 ∪ A2 ∪ A4
A2 ∪ A3 ∪ A4
A1 ∪ A2 ∪ A3 ∪ A4
```

Рисунок Л2.2 – Результат виконання методу повного перебору

У наведеному переліку (рис. Л2.2) є всі можливі покриття, зокрема надлишкові, для універсальної множини U.

Програмний код, зображений на рис. Л2.1, можна вдосконалити для визначення всіх безнадлишкових покриттів або побудови одного найкоротшого покриття.

Завдання до лабораторної роботи

Завдання 1. Розробити схему алгоритму та програму побудови всіх безнадлишкових покриттів методом повного та граничного перебору. Розрахувати ціну кожного безнадлишкового покриття.

Завдання 2. Розробити схему алгоритму та програму побудови найкоротшого покриття методом мінімального стовпця – максимального рядка та методом ядерних рядків.

Порядок виконання лабораторної роботи

1. Для виконання лабораторної роботи потрібно вибрати варіант, який відповідає індивідуальному практичному завданню № 2.
2. Для вибраного варіанта протестувати виконання коду, наведеного на рис. Л2.1.
3. Внести зміни у код (рис. Л2.1) для побудови найкоротшого покриття.
4. Внести зміни у код (рис. Л2.1) для побудови безнадлишкових покриттів та розрахунку їхньої ціни.
5. Для парного варіанта розробити алгоритм та програму для побудови безнадлишкових покриттів методом граничного перебору.
6. Для непарного варіанта розробити алгоритм та програму для побудови найкоротшого покриття методом мінімального стовпця – максимального рядка та методом ядерних рядків.

7. Навести результати тестування програмного коду та порівняти їх із результатами ручного розрахунку для індивідуального практичного завдання № 2.

8. Зробити висновки щодо результатів застосування методів побудови покриття, відзначивши їхні ключові особливості та відмінності.

Контрольні питання

1) Що таке задача про покриття, та які її основні приклади технічного застосування?
2) Яке покриття називається безнадлишковим, та яка умова використовується для перевірки безнадлишковості?
3) Наведіть означення мінімального та найкоротшого покриттів.
4) Розкрийте зміст алгоритму граничного перебору.
5) Який ваш критерій завершення перебору?
6) Проведіть порівняльний аналіз методів повного та граничного переборів.
7) Скільки підмножин аналізується в методі повного перебору?
8) Як у вашому алгоритмі вирішується задача повторів (тобто, не будуються множини, які вже були побудовані)?
9) Розкрийте суть методу мінімального стовпця – максимального рядка
10) В чому зміст методу ядерних рядків?
11) Який рядок називають антиядерним?

Лабораторна робота № 3

Тема: Булеві функції. Реалізація функцій алгебри логіки в середовищі програмування.

Мета роботи: Набуття практичних навичок побудови функцій алгебри логіки мовою програмування Python.

Методичні рекомендації

Булеві функції є основою для опису алгоритмів роботи дискретних пристроїв, зокрема цифрових обчислювальних машин і логічних схем. Такі пристрої оперують фізичними сигналами, квантованими на два рівні: 0 та 1, що дозволяє моделювати їхню роботу через набір логічних операцій. Булеві функції широко застосовуються для проєктування комбінаційних схем, мультиплексорів, суматорів та інших компонентів цифрових систем.

Операції над булевими функціями у Python можна ефективно виконувати за допомогою бібліотеки NumPy. Ця бібліотека пропонує широкий спектр інструментів (табл. Л3.1), забезпечуючи високу продуктивність і зручність у роботі. Приклад використання основних логічних операцій наведено у програмному коді (рис. Л3.1), а результат його виконання – на рис. Л3.2

Таблиця Л3.1 – Основні логічні операції над булевими функціями у Python з використанням бібліотеки NumPy

Операція	**Позначення**	**Опис**	**Синтаксис**
1	2	3	4
Логічне «І» (AND)	$A \wedge B$	Повертає True, якщо обидва елементи є True	`np.logical_and(A,B)`
Логічне «АБО» (OR)	$A \vee B$	Повертає True, якщо хоча б один елемент є True	`np.logical_or(A,B)`
Логічне «НЕ» (NOT)	$\bar{B}$	Повертає протилежне значення	`np.logical_not(B)`
Сума за модулем (XOR)	$A \oplus B$	Повертає True, якщо елементи різні	`np.logical_xor(A,B)` `np.not_equal(A,B)`
Рівнозначність	$A \sim B$	Повертає True, якщо елементи однакові	`np.equal(A,B)`
Імплікація (від А до В)	$A \rightarrow B$	Повертає True, якщо A <= B	`np.greater_equal(B,A)`

Продовження таблиці Л3.1

1	2	3	4
Імплікація (від B до A)	$B \to A$	Повертає True, якщо B <= A	`np.greater_equal(A,B)`
Штріх Шефера «NAND» (NOT AND)	$A\|B$	Повертає False, якщо обидва елементи є True	`np.logical_not(np.logical_and(A,B))`
Стрілка Пірса «NOR» (NOT OR)	$A \downarrow B$	Повертає True, якщо обидва елементи є False	`np.logical_not(np.logical_or(A,B))`
Заборона по A	$B\Delta A$	Повертає True, якщо B більше A	`np.greater(B,A)`
Заборона по B	$A\Delta B$	Повертає True, якщо A більше B	`np.greater(A,B)`

```python
import numpy as np

# Задаємо набір значень A та B
B = np.array([0, 1, 0, 1])
A = np.array([0, 0, 1, 1])

result_and = np.logical_and(A, B)       # Логічне І (AND)
result_or = np.logical_or(A, B)         # Логічне АБО (OR)
result_not = np.logical_not(B)          # Логічне НЕ (NOT)
result_equiv = np.equal(A, B)           # Лог. рівнозначність
result_not_equiv = np.not_equal(A, B)   # Лог. нерівнозначність
result_logical_xor = np.logical_xor(A, B)       # Сума за модулем
result_1 = np.greater_equal(B,A)        # Імплікація від A до B
result_4 = np.greater_equal(A,B)        # Імплікація від B до A
result_2 = np.logical_not(np.logical_and(A, B)) # Штріх Шефера
result_3 = np.logical_not(np.logical_or(A, B))  # Стрілка Пірса
result_5 = np.greater(B,A)              # Заборона по A
result_6 = np.greater(A,B)              # Заборона по B

print("A =", A)
print("B =", B)
print("Кон'юнкція AB =", result_and.astype(int))
print("Диз'юнкція A+B =", result_or.astype(int))
print("Логічне не B =", result_not.astype(int))
print("Лог. рівнозначність A~B =", result_equiv.astype(int))
print("Лог. нерівнозначність XOR =", result_not_equiv.astype(int))
print("Сума за модулем 2 XOR =", result_logical_xor.astype(int))
print("Штріх (операція) Шефера", result_2.astype(int))
print("Стрілка Пірса", result_3.astype(int))
print("Імплікація від A до B", result_1.astype(int))
print("Імплікація від B до A", result_4.astype(int))
print("Заборона по A", result_5.astype(int))
print("Заборона по B", result_6.astype(int))
```

Рисунок Л3.1 – Основні логічні операції над булевими функціями у Python з використанням бібліотеки NumPy

```
A = [0 0 1 1]
B = [0 1 0 1]
Кон'юнкція AB = [0 0 0 1]
Диз'юнкція A+B = [0 1 1 1]
Логічне не B = [1 0 1 0]
Лог. рівнозначність A~B = [1 0 0 1]
Лог. нерівнозначність XOR = [0 1 1 0]
Сума за модулем 2 XOR = [0 1 1 0]
Штріх (операція) Шефера [1 1 1 0]
Стрілка Пірса [1 0 0 0]
Імплікація від A до B [1 1 0 1]
Імплікація від B до A [1 0 1 1]
Заборона по A [0 1 0 0]
Заборона по B [0 0 1 0]
```

Рисунок Л3.2 – Результат виконання коду (рис. Л3.1)

Завдання до лабораторної роботи

Завдання 1. Використовуючи бібліотеку NumPy, виконати покроково логічні операції над трьома булевими змінними x_1, x_2, x_3 для знаходження функції f (табл. Л3.2). Побудувати відповідні часові діаграми для кожної з операцій та для кінцевої функції f. Результат знаходження функції f перевірити ручним виконанням.

Завдання 2. Перевірити ручний розв'язок індивідуального практичного завдання № 5 (табл. 3.1) щодо знаходження булевої функції чотирьох змінних A, B, C та D за допомогою бібліотеки NumPy.

Таблиця Л3.2. – Варіанти завдань

Номер варіанта	Функція	Номер варіанта	Функція
1.	$f = x_1 \cdot \bar{x}_2 \mid (x_3 + \bar{x}_2)$.	16.	$f = x_1 \cdot (x_3 \oplus \bar{x}_2)$.
2.	$f = x_1 \cdot \bar{x}_2 \to x_2 \cdot \bar{x}_3$.	17.	$f = (x_1 \oplus \bar{x}_2) \to (x_2 \Delta \bar{x}_3)$.
3.	$f = x_1 \to (x_3 + \bar{x}_2)$.	18.	$f = x_1 \Delta (x_3 \oplus \bar{x}_2)$.
4.	$f = (x_1 \oplus \bar{x}_2) \to x_3$.	19.	$f = (x_1 \mid \bar{x}_2) \sim (x_2 \Delta \bar{x}_3)$.
5.	$f = (x_1 \Delta \bar{x}_2) \cdot (x_3 \downarrow x_2)$.	20.	$f = (x_1 \oplus \bar{x}_2) \sim (x_3 \Delta x_2)$.
6.	$f = x_1 \cdot \bar{x}_2 \oplus x_3$.	21.	$f = x_1 \mid \bar{x}_2 \Delta x_3$.
7.	$f = (x_1 \oplus \bar{x}_3) \cdot (x_1 \sim \bar{x}_2)$.	22.	$f = (x_1 \mid \bar{x}_3) \sim (x_1 \Delta \bar{x}_2)$.
8.	$f = (x_1 \mid \bar{x}_2) \to x_3 + \bar{x}_2$.	23.	$f = x_1 \cdot \bar{x}_2 \sim (x_3 \oplus \bar{x}_2)$.
9.	$f = x_2 \cdot \bar{x}_1 \oplus (x_3 \to \bar{x}_2)$.	24.	$f = x_2 \cdot \bar{x}_1 \to (x_3 \downarrow x_2)$.
10.	$f = (x_1 \to x_2) \oplus x_3$.	25.	$f = (x_1 \sim x_2) + x_3 \downarrow \bar{x}_2$.
11.	$f = (x_1 \downarrow \bar{x}_2) + (x_3 \downarrow x_2)$.	26.	$f = (x_1 \Delta \bar{x}_2) \sim (x_3 \to x_2)$.
12.	$f = (x_1 \cdot \bar{x}_3 \to x_2) \oplus x_3$.	27.	$f = (x_1 \cdot \bar{x}_3 \Delta x_2) \sim x_3$.
13.	$f = (x_1 \downarrow \bar{x}_2 \downarrow x_3) \sim \bar{x}_2$.	28.	$f = (x_1 \Delta \bar{x}_2 \downarrow x_3) \oplus \bar{x}_2$.
14.	$f = (x_1 \Delta \bar{x}_2) \cdot (x_3 \sim x_2)$.	29.	$f = (x_1 \oplus \bar{x}_2) \cdot (\bar{x}_3 \downarrow x_2)$.
15.	$f = (x_1 \to \bar{x}_2) \downarrow (x_3 \oplus x_2)$.	30.	$f = (x_1 \oplus \bar{x}_2) \downarrow (x_3 \Delta \bar{x}_2)$.

Порядок виконання лабораторної роботи

Завдання 1. Для прикладу розглянемо функцію $F = (x_1 \oplus x_2)(x_1 + \bar{x}_3)$.

1. У робочому вікні Jupyter.Notebook імпортуємо бібліотеки Numpy та Matplotlib:

```
import numpy as np
import matplotlib.pyplot as plt
```

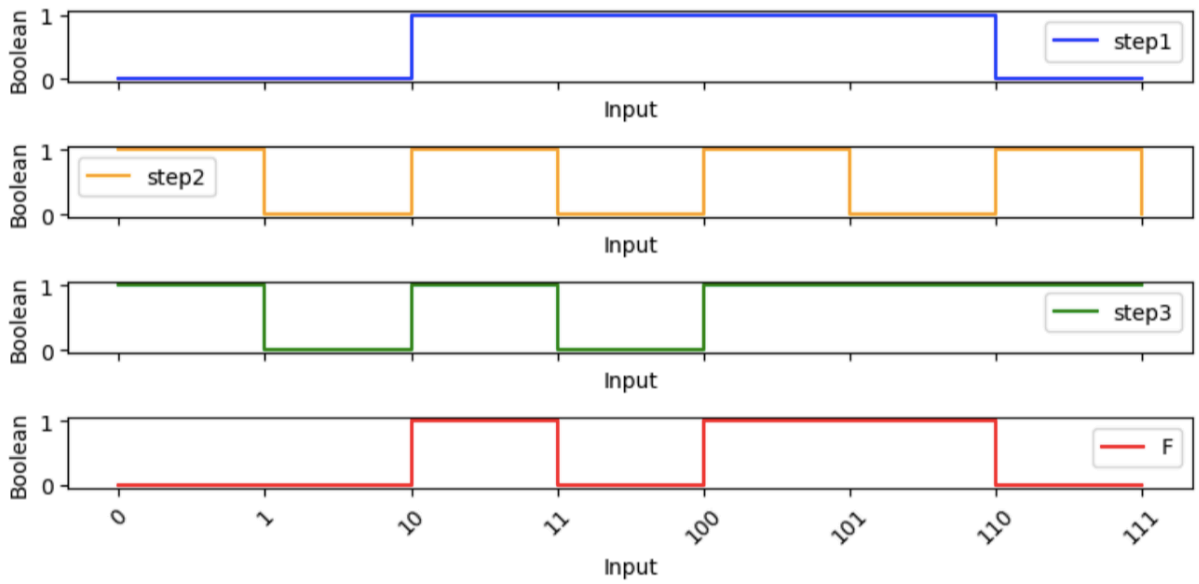
2. Задаємо загальну функцію для побудови графіків часових діаграм:

```
def plot_boolean_function(ax, F, label=None, color='green'):
    x = np.arange(len(F))
    ax.step(x, F, where='post', label=label, color=color)
    ax.set_xticks(x)
    ax.set_xticklabels([f'{i:b}' for i in range(len(F))], rotation=45)
    ax.set_xlabel('Input')
    ax.set_ylabel('Boolean')
    ax.legend()
```

3. Задаємо аргументи булевої функції та послідовно виконуємо потрібні логічні операції:

```
# Задаємо набір значень аргументів функції F
x3 = np.array([0, 1, 0, 1, 0, 1, 0, 1])
x2 = np.array([0, 0, 1, 1, 0, 0, 1, 1])
x1 = np.array([0, 0, 0, 0, 1, 1, 1, 1])

# Задаємо розміри загального графіка та визначаємо кількість підграфіків
fig, axs = plt.subplots(4, 1, figsize=(8, 4), sharex=True)

# Покроково виконуємо операції для знаходження F
step1 = np.logical_xor(x1, x2)
step2 = np.logical_not(x3)
step3 = np.logical_or(x1, step2)
F = np.logical_and(step1, step3).astype(int)

# Визначаємо підграфіки
plot_boolean_function(axs[0], step1, label='step1', color='blue')
plot_boolean_function(axs[1], step2, label='step2', color='orange')
plot_boolean_function(axs[2], step3, label='step3', color='green')
plot_boolean_function(axs[3], F, label='F', color='red')

plt.tight_layout()
plt.show()

print("F = (x1 ⊕ x2) ∧ (x1 ∨ ¬x3) =", F)
```

4. У результаті виконання коду отримуємо набір часових графіків для кожного кроку та заданої функції *F* (рис. Л3.3).



F = (x1 ⊕ x2) ∧ (x1 ∨ ¬x3) = [0 0 1 0 1 1 0 0]

Рисунок Л3.3

Завдання 2. Для виконання завдання 2 потрібно модифікувати попередній код таким чином, щоб підсумкова функція f відповідала булевій функції чотирьох змінних A, B, C, D.

Контрольні питання

1) Що таке перемикальна функція?
2) На скількох наборах визначена перемикальна функція?
3) Які способи задання перемикальної функції існують?
4) Яке призначення функції `plot_boolean_function` у коді? Як її використовувати для побудови графіків?
5) Що означає логічна операція XOR (виключне АБО), і як вона реалізується у Python?
6) Як змінюється результат булевої функції у випадку застосування операції NOT до одного з її аргументів?
7) Які особливості потрібно враховувати при розширенні булевої функції до чотирьох аргументів?
8) Який результат роботи функції `np.logical_and`, і як вона використовується для булевих функцій?
9) Чим відрізняються логічні операції OR і XOR? Наведіть приклади.

Лабораторна робота № 4

Тема: Мінімізація булевих функцій алгебри логіки за допомогою методів спрощення логічних виразів.

Мета роботи: Набуття практичних навичок мінімізації булевих функцій алгебри логіки з використанням мови програмування Python.

Методичні рекомендації

Теоретичні відомості та основні прийоми, що використовуються для мінімізації логічних функцій, були детально розглянуті в шостій темі нашого посібника. У цій лабораторній роботі основну увагу приділено практичній реалізації згаданих методів за допомогою мови програмування Python із використанням функцій `simplify_logic`, `SOPform` і `POSform` бібліотеки SymPy (табл. Л4.1).

Таблиця Л4.1 – Функції спрощення булевих виразів

Функція	Дія	Умова використання
`simplify_logic`	Мінімізує булевий вираз у найкоротшій формі	Для автоматичного спрощення булевої функції без жорстких вимог до форми (ДНФ чи КНФ).
`SOPform`	Побудова мінДНФ	Коли потрібно отримати функцію у вигляді суми добутків, використовуючи мінтерми.
`POSform`	Побудова мінКНФ	Коли потрібно отримати функцію у вигляді добутку сум, використовуючи макстерми.

Функція `simplify_logic` із бібліотеки SymPy використовується для мінімізації булевих функцій. Вона приймає як вхідний параметр булевий вираз у символьному вигляді та повертає його спрощену форму, зводячи кількість логічних операцій до мінімуму.

Ця функція підтримує як стандартну форму подання логічних виразів, так і роботу з істинними таблицями. Її можна застосовувати для отримання мінімізованих форм у вигляді диз’юнктивної нормальної форми (ДНФ) або кон’юнктивної нормальної форми (КНФ) залежно від завдання. Для цього використовується параметр `form`, який дозволяє обирати між `'dnf'` (диз’юнктивна форма) і `'cnf'` (кон’юнктивна форма). На рис. Л4.1 наведено приклад використання цієї функції.

`Simplify_logic` найкраще використовувати для задач, у яких потрібно отримати компактне подання таких булевих функцій, як оптимізація електронних схем чи зменшення складності логічних виразів у програмному коді.

```python
from sympy import symbols, simplify_logic

# Оголошення змінних
A, B, C = symbols('A B C')

# Початковий булевий вираз
F = (A & ~B & ~C) | (A & B & ~C) | (~A & B & C)

# Отримання мінімальної ДНФ
F_dnf_min = simplify_logic(F, form='dnf')

# Отримання мінімальної КНФ (кон'юнктивної нормальної форми)
F_cnf_min = simplify_logic(F, form='cnf')

print("Початкова функція: F =", F)
print("Мінімальна диз'юнктивна нормальна форма (мінДНФ):", F_dnf_min)
print("Мінімальна кон'юнктивна нормальна форма (мінКНФ):", F_cnf_min)
```

```
Початкова функція: F = (A & B & ~C) | (B & C & ~A) | (A & ~B & ~C)
Мінімальна диз'юнктивна нормальна форма (мінДНФ): (A & ~C) | (B & C & ~A)
Мінімальна кон'юнктивна нормальна форма (мінКНФ): (A | B) & (A | C) & (~A | ~C)
```

Рисунок Л4.1 – Приклад використання функції `Simplify_logic`

Функції `SOPform` (Sum of Products) і `POSform` (Product of Sums) із бібліотеки SymPy використовуються для отримання мінімізованих форм булевих функцій у вигляді диз'юнктивної нормальної форми (мінДНФ) та кон'юнктивної нормальної форми (мінКНФ).

Обидві функції приймають три основні параметри: список змінних, набір мінтермів або макстермів (для яких функція, відповідно, приймає істинність або хибність), а також, за потреби, набір неважливих значень. Результатом є логічний вираз у відповідній мінімальній формі – компактний і зручний для аналізу або реалізації.

Мінтерми (minterms) – це окремі логічні вирази, які відповідають наборам значень змінних, за яких булева функція набуває значення 1 (істина).

У канонічній диз'юнктивній нормальній формі (ДНФ) булева функція записується як сума (логічне «або») добутків (логічне «І») змінних, де кожен добуток подає один мінтерм.

Макстерми (maxterms) – це логічні вирази, які відповідають тим комбінаціям змінних, за яких булева функція набуває хибного значення (0). Макстерми є основою для формування кон'юнктивної нормальної форми (КНФ), оскільки вони є набором умов, які мають бути виконані одночасно, щоб функція була хибною.

Неважливі терми або терми «байдужості» (dontcares) – це значення булевої функції, для яких її результат не впливає на кінцеву поведінку системи чи результат роботи. Тобто, для цих термів можна вибрати значення функції як 0 або 1, залежно від того, що забезпечує найпростішу форму функції.

`SOPform` використовується для отримання булевої функції у формі мінДНФ, що складається з диз'юнкції кількох кон'юнкцій. Це корисно для задач, де потрібно описати функцію через істинні значення, наприклад, у процесі побудови комбінаційних схем. Програмний код з прикладом використання цієї функції наведено на рис. Л4.2.

```
from sympy import symbols
from sympy.logic import SOPform, POSform

x1, x2, x3, x4 = symbols('x1 x2 x3 x4') # Оголошення змінних

minterms = [1, 3, 7, 11, 15]  # Список мінтермів
dontcares = [0, 2, 5]         # Список неважливих (невизначених) термів

# Отримання мінімальної ДНФ без урахування неважливих термів
minDNF_1 = SOPform([x1, x2, x3, x4], minterms)

# Отримання мінімальної ДНФ з урахуванням неважливих термів
minDNF_2 = SOPform([x1, x2, x3, x4], minterms, dontcares)

print("мінДНФ без урахування неважливих термів F1 =", minDNF_1)
print("мінДНФ з урахуванням неважливих термів F2 =", minDNF_2)
```

```
мінДНФ без урахування неважливих термів F1 = (x3 & x4) | (x4 & ~x1 & ~x2)
мінДНФ з урахуванням неважливих термів F2 = (x3 & x4) | (~x1 & ~x2)
```

Рисунок Л4.2 – Приклад використання функції `SOPform`

`POSform` генерує булеву функцію у формі КНФ, подану кон'юнкцією кількох диз'юнкцій. Такий підхід ідеально підходить для випадків, коли функцію потрібно описати через хибні значення, наприклад, у разі побудови схем на основі логічних елементів «OR-AND». Програмний код з прикладом використання `POSform` наведено на рис. Л4.3.

```
maxterms = [1, 3, 7, 11, 15]  # Список макстермів
dontcares = [0, 2, 5]         # Список неважливих (невизначених) термів

# Отримання мінімальної КНФ без урахування неважливих термів
minKNF_1 = POSform([x1, x2, x3, x4], minterms)

# Отримання мінімальної КНФ з урахуванням неважливих термів
minKNF_2 = POSform([x1, x2, x3, x4], minterms, dontcares)

print("мінКНФ без урахування неважливих термів F1 =", minKNF_1)
print("мінКНФ з урахуванням неважливих термів F2 =", minKNF_2)
```

```
мінКНФ без урахування неважливих термів F1 = x4 & (x3 | ~x1) & (x3 | ~x2)
мінКНФ з урахуванням неважливих термів F2 = x4 & (x3 | ~x1)
```

Рисунок Л4.3 – Приклад використання функції `POSform`

Обидві функції ефективні для мінімізації булевих функцій і широко використовуються в цифровій логіці, оптимізації схем і програмному моделюванні. Вибір між ними залежить від форми подання, яка найбільше підходить для конкретного завдання.

Завдання до лабораторної роботи

Завдання 1. Мінімізувати логічну функцію, задану у табл. 5.1. використовуючи `simplify_logic` бібліотеки NumPy. Отриманий результат порівняти з результатом ручного виконання індивідуального практичного заняття № 6, задача 1.

Завдання 2. Мінімізувати логічну функцію, задану у табл. 6.2. використовуючи `SOPform` і `POSform` бібліотеки NumPy. Отриманий результат порівняти з результатом ручного виконання індивідуального практичного заняття № 6, задача 2.

Порядок виконання лабораторної роботи

Завдання 1.

1. Для виконання завдання 1 потрібно вибрати логічну функцію, яка відповідає варіанту виконання індивідуального практичного завдання № 6, задача 1 (див. табл. 5.1).

2. Використовуючи функцію `simplify_logic` та приклад, наведений на рис. Л4.1, розробити програму для знаходження мінімальної диз'юнктивної нормальної форми (мінДНФ) та мінімальної кон'юнктивної нормальної форми (мінКНФ) для вибраної логічної функції. Порівняти отриманий результат з ручним виконанням.

Завдання 2.

1. Для виконання завдання 2 потрібно вибрати логічну функцію, яка відповідає варіанту виконання індивідуального практичного завдання № 6, задача 2 (див. табл. 6.2).

2. Використовуючи функції `SOPform` і `POSform` та приклади, наведені на рис. Л4.2 та рис. Л4.3, розробити програму для знаходження мінімальної диз'юнктивної нормальної форми (мінДНФ) та мінімальної кон'юнктивної нормальної форми (мінКНФ) для вибраної логічної функції. Порівняти отриманий результат з ручним виконанням.

3. Довільно вибрати декілька неважливих термів та повторно розрахувати мінДНФ та мінКНФ для заданої логічної функції. Порівняти, як додавання неважливих термів вплинуло на отримані вирази.

Контрольні питання

1) Що таке булевий вираз? Наведіть приклади.

2) У чому полягає різниця між ДНФ (СДНФ) і КНФ (СКНФ)?

3) Що таке мінтерми та макстерми? Як вони використовуються у процесі побудови ДНФ і КНФ?

4) Яка мета мінімізації булевих виразів? Наведіть приклади її практичного застосування.

5) Чим відрізняється функція `SOPform` від `simplify_logic`?

6) Як враховувати неважливі значення булевої функції за мінімізації булевих функцій?

7) Які функції SymPy можна використати для автоматичного спрощення складного булевого виразу? Чим вони відрізняються?

8) Які переваги має спрощення булевих виразів у ДНФ або КНФ перед їх використанням у цифровій логіці?

9) Як неважливі значення булевої функції впливають на результат мінімізації?

10) Порівняйте результати мінімізації булевого виразу за допомогою функцій `SOPform`, `POSform` і `simplify_logic`.

ЛІТЕРАТУРА

1. *Discrete Mathematics and Its Applications* / Kenneth H. Rosen. – 7th ed. – New York: McGraw-Hill Education, 2012. – Доступно за посиланням: https://elearn.daffodilvarsity.edu.bd/pluginfile.php/783865/mod_resource/intro/Discrete%20Mathematics%20and%20Its%20Applications%2C%207%20edition%20-%20Rosen.pdf

2. *Discrete Mathematics* / Richard Johnsonbaugh. – 7th ed. – Boston: Pearson, 2014. – Доступно за посиланням: https://api.pageplace.de/preview/DT0400.9781292035819_A24583516/preview-9781292035819_A24583516.pdf

3. Новотарський М. А. Дискретна математика : навчальний посібник для студентів спеціальності 123 «Комп'ютерна інженерія», спеціалізації «Комп'ютерні системи та мережі» [Електронний ресурс] Київ : КПІ ім. Ігоря Сікорського, 2020. – Доступно за посиланням: https://ela.kpi.ua/handle/123456789/37806

4. Темнікова О. Л. Дискретна математика: конспект лекцій (Частина 1): навч. посіб. для студ. спеціальності 113 «Прикладна математика», освітньої програми «Наука про дані та математичне моделювання» [Електронний ресурс] Київ : КПІ ім. Ігоря Сікорського, 2021. 154 с. – Доступно за посиланням: https://ela.kpi.ua/server/api/core/bitstreams/990893b6-f853-408a-8476-d3dd7c89d2a1/content

5. *Дискретна математика : навч. посіб.* / уклад. С. І. Балога; рец. О. О. Погоріляк. Ужгород : ПП «АУТДОР-ШАРК», 2021. 124 с. – Доступно за посиланням: https://dspace.uzhnu.edu.ua/jspui/handle/lib/36740

6. Бондаренко М. Ф., Білоус Н. В., Руткас А. Г. Комп'ютерна дискретна математика : підручник. Харків : «Компанія СМІТ», 2004. 480 с. – Доступно за посиланням: https://pz.vntu.edu.ua/media/uploads/metod/Дискретня%20математика.pdf

7. Боднарчук Ю. В., Олійник Б. В. Основи дискретної математики (для студентів-інформатиків). Київ : Київський національний університет імені Тараса Шевченка, 2007. 138 с. – Доступно за посиланням: https://www.ukma.edu.ua/~bogd/Discrete%20Mathematics/PosibnykNew.pdf

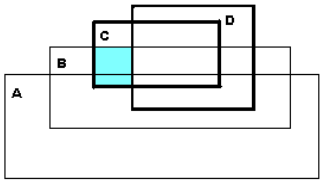
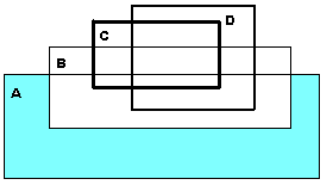
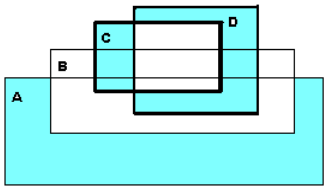
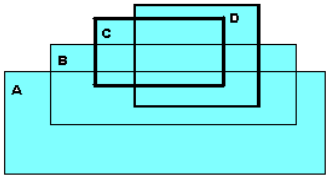
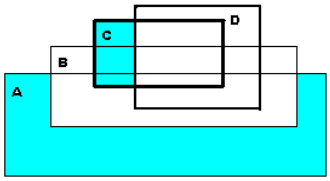
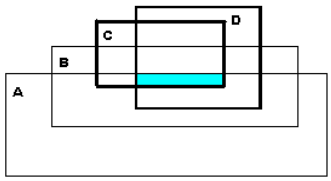
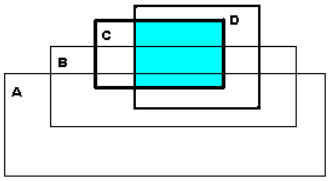
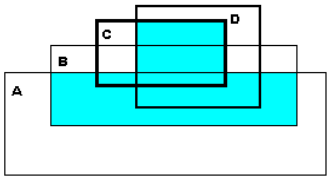
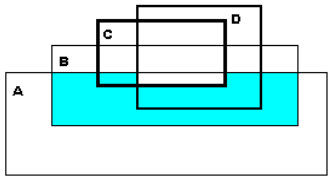
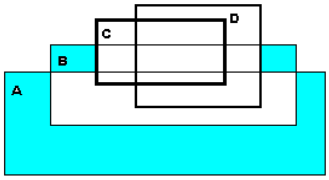
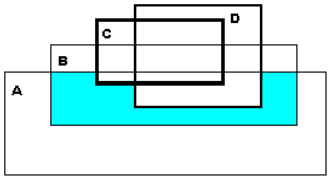
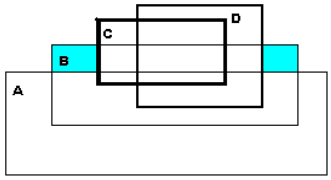
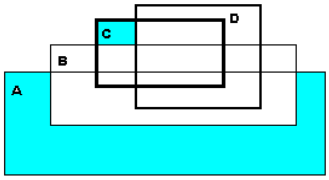
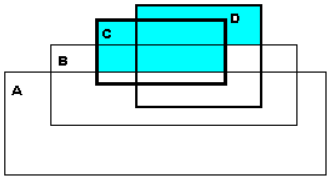
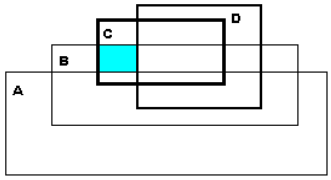
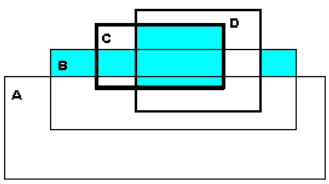
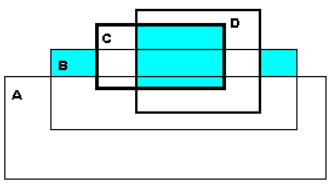
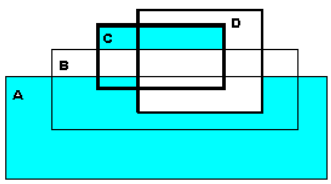
8. Олійник Л. О. Дискретна математика : навч. посібник. Дніпродзержинськ : ДДТУ, 2015. 256 с. – Доступно за посиланням: https://www.dstu.dp.ua/Portal/Data/3/17/3-17-b2.pdf?utm_source=chatgpt.com

9. Харченко В. М. Практикум з дискретної математики. Ніжин : НДУ ім. М. Гоголя, 2022. 148 с. – Доступно за посиланням: http://lib.ndu.edu.ua/dspace/bitstream/123456789/2622/1/Практикум%20з%20дискретної%20математики.pdf

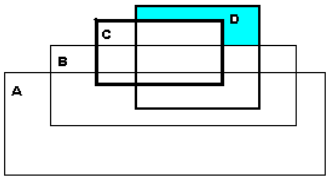
10. *Discrete Mathematics* [Електронний ресурс]. – Доступно за посиланням: http://mathworld.wolfram.com/topics/DiscreteMathematics.html
11. *Python Programming Language*. – Доступно за посиланням: https://www.python.org/
12. *Jupyter Notebooks*. – Доступно за посиланням: https://jupyter.org/
13. *NumPy: The fundamental package for scientific computing with Python*. – Доступно за посиланням: https://numpy.org/
14. *SciPy: Open source scientific tools for Python*. – Доступно за посиланням: https://scipy.org/
15. *Matplotlib: Python plotting library*. – Доступно за посиланням: https://matplotlib.org/

Додаток А

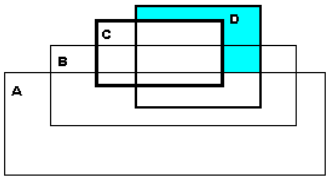
Діаграми Венна. Варіанти індивідуальних завдань

1. 	2. 	3. 
4. 	5. 	6. 
7. 	8. 	9. 
10. 	11. 	12. 
13. 	14. 	15. 
16. 	17. 	18. 

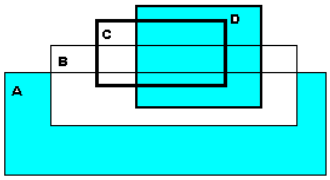
19.



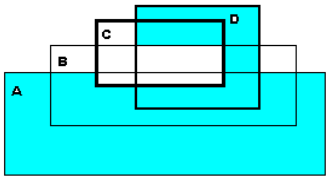
20.



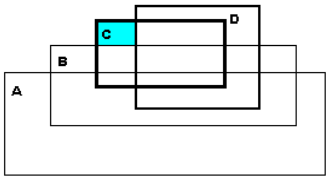
21.



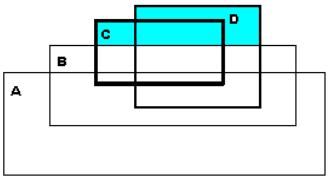
22.



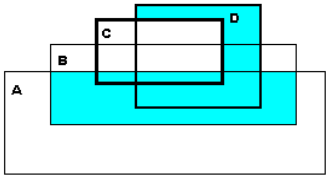
23.



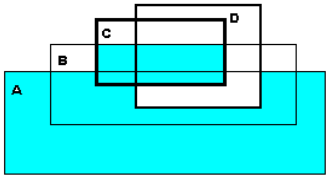
24.



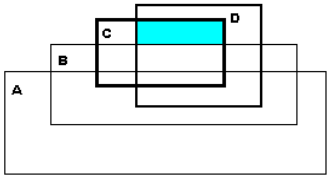
25.



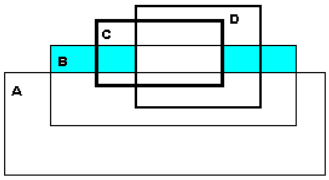
26.



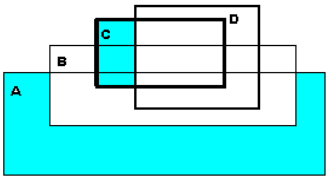
27.



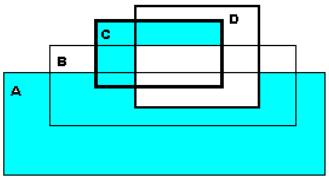
28.



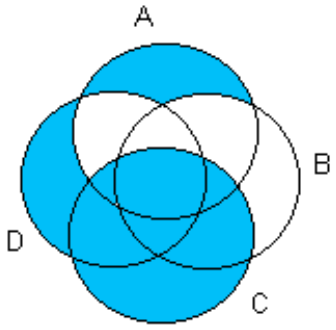
29.



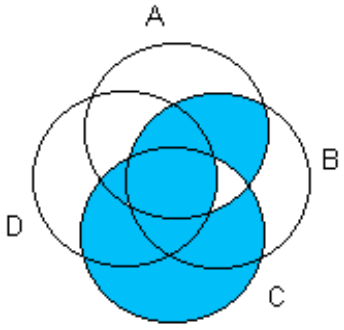
30.



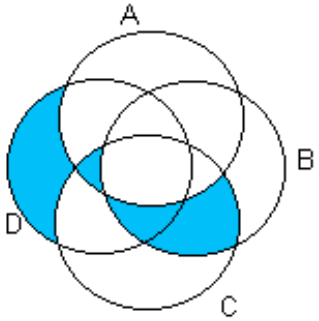
31.



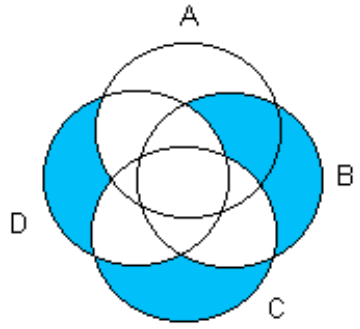
32.



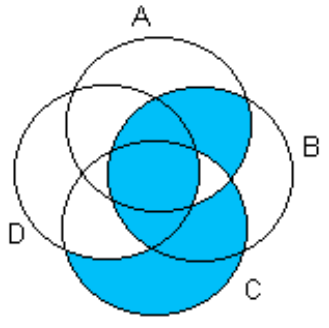
33.



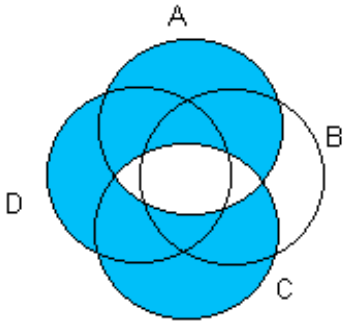
34.

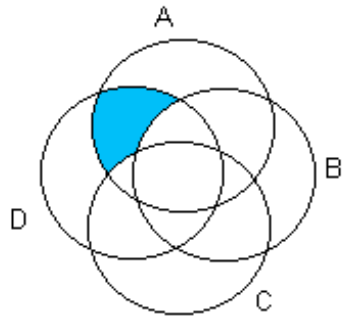
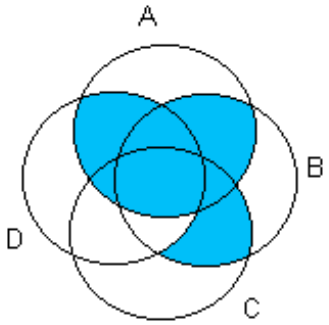
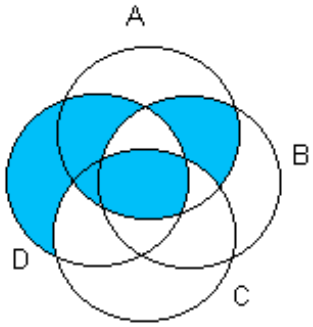
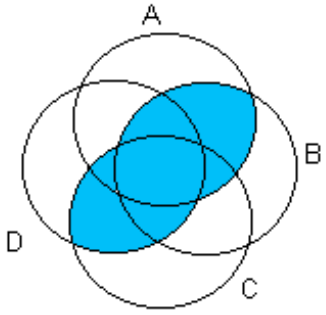
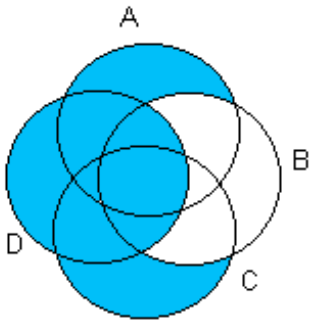
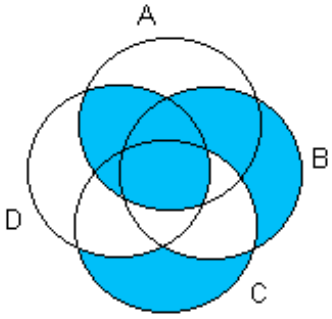
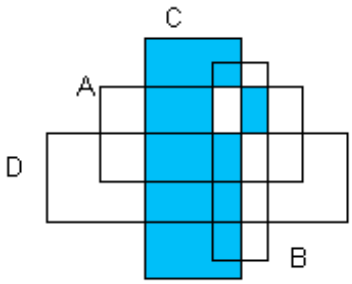
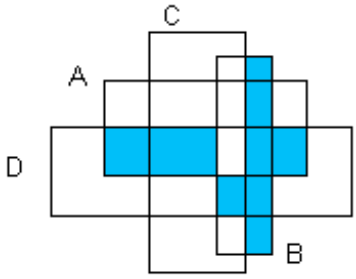
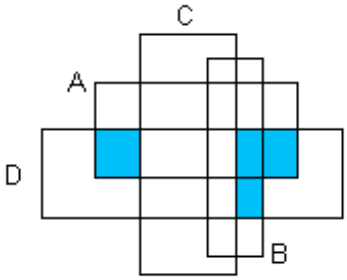
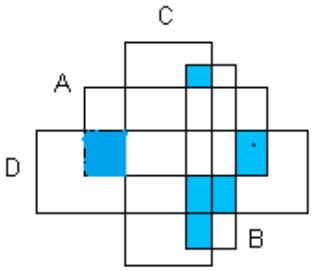
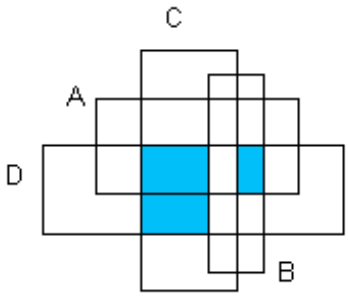
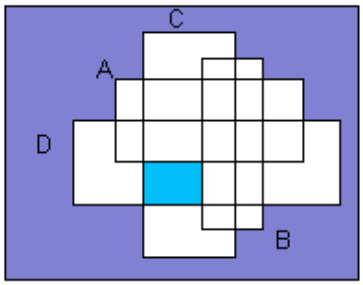


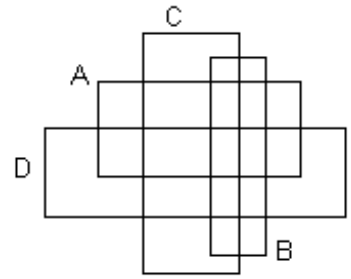
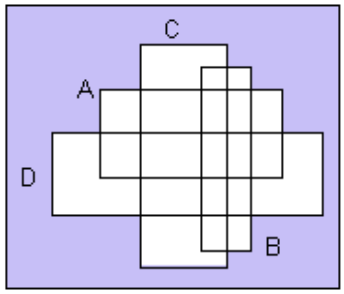
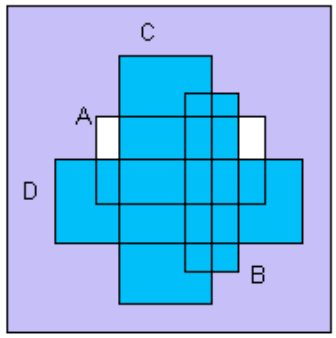
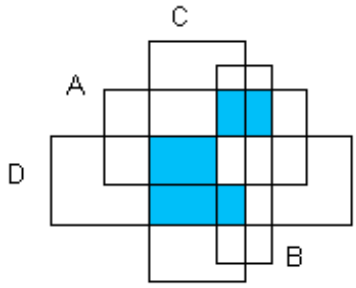
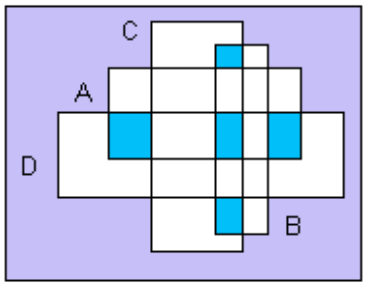
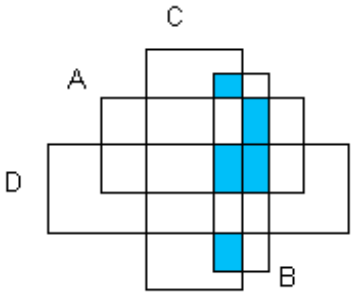
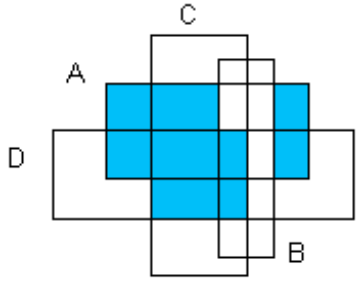
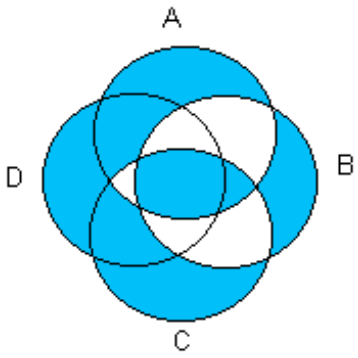
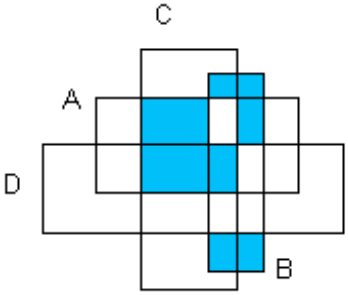
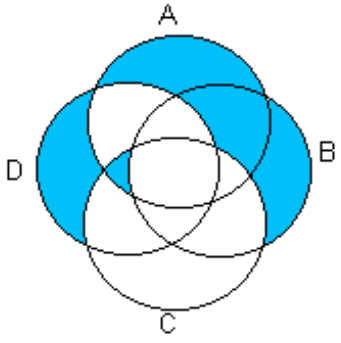
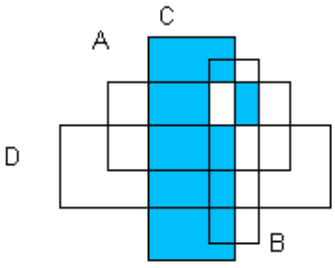
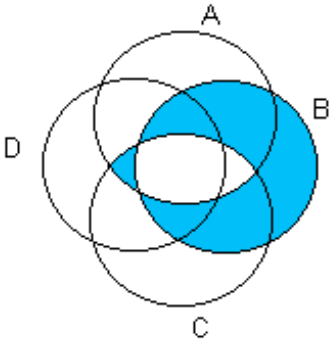
35.



36.



37. 	38. 	39. 
40. 	41. 	42. 
43. 	44. 	45. 
46. 	47. 	48. 

49. 	50. 	51. 
52. 	53. 	54. 
55. 	56. 	57. 
58. 	59. 	60. 

Електронне навчальне видання

Олег Костянтинович Колесницький
Олександр Федорович Шевчук
Тетяна Геннадіївна Кирилащук

ДИСКРЕТНА МАТЕМАТИКА
Частина 1

Навчальний посібник

Рукопис оформив *О. Шевчук*

Редактор *В. Дружиніна*

Оригінал-макет виготовила *Т. Старічек*

Підписано до видання 18.03.2025 р.
Гарнітура Times New Roman.
Зам. № Р2025-054

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
E-mail: irvc.ed.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.