МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

В.В. Слєсарєв І.В. Новицький
С.А. Ус

ДИСКРЕТНА МАТЕМАТИКА

НАВЧАЛЬНИЙ ПОСІБНИК

Дніпро
НТУ «ДП»
2023

УДК 519.17:519.45(075.8)
С47

Затверджено до видання вченою радою університету як навчальний посібник для здобувачів ступеня бакалавра спеціальності 124 Системний аналіз (протокол № 7 від 29 червня 2023).

Рецензенти:
А. В. Бакурова – д-р екон. наук, професор (Національний університет «Запорізька політехніка»);
Л. І. Лозовська – канд. фіз.-мат. наук, доцент (Український державний університет науки і технологій)

Слєсарєв В.В.
С47 Дискретна математика: навч. посібник / В.В. Слєсарєв, І.В. Новицький, С.А. Ус. – М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2023. – 183 с.

Навчальний посібник охоплює матеріал, передбачений програмою дисципліни «Дискретна математика» для студентів спеціальності 124 Системний аналіз. Розглянуто базові теоретичні засади дискретної математики, методи й алгоритми, необхідні для розв'язування практичних задач, показано їх застосування на конкретних прикладах.

Навчальний посібник має на меті допомогти студентам у вивченні обов'язкової дисципліни «Дискретна математика» під час виконання контрольних робіт і підготовки до практичних і лабораторних занять.

Книгу розраховано на осіб, які знають математику в межах університетського курсу, рекомендовано для студентів технічних спеціальностей і тих, хто використовує методи дискретної математики при розв'язуванні практичних задач.

УДК 519.17:519.45(075.8)

ЗМІСТ

ПЕРЕДМОВА

При вивченні певних об’єктів виробничої, економічної та інших сфер діяльності людини з’ясовується, що їх точний опис потребує виміру великої кількості змінних або параметрів. У деяких випадках змінні, котрі беруть участь у системі контролю або керування, є такими, що неперервний вимір їх значень просто неможливий через величезну кількість даних, які обробляються.

Ситуацію можна вирішити, якщо перейти до дискретних уявлень про параметри. Це дозволяє виконувати виміри тільки в певні фіксовані моменти часу, що неминуче знижує точність подання об’єкта дослідження, але й одночасно спрощує його модель. Правомірність переходу до дискретного подання має бути ретельно обґрунтована, чим і займається дискретна математика.

Дискретна математика, або *дискретний аналіз* – це самостійний розділ сучасної математики, що вивчає властивості різних структур, які мають скінченний характер. Вони можуть виникати як у самій математиці, так і в її застосуваннях. До них відносять об’єкти, що мають переривчастий (дискретний) характер, на відміну від досліджуваних класичною математикою неперервних об’єктів.

Основними об’єктами дискретної математики є множини, графи, функції логіки й автомати, формальні системи і комбінаторні кофігурації.

Стимулом до розвитку багатьох напрямів цієї науки стали потреби теоретичної кібернетики, безпосередньо пов’язаної з розвитком інформаційних технологій. Застосування ЕОМ з метою комплексної автоматизації інформаційної діяльності принципово змінило характер контакту людини з машиною. Якщо раніше комп’ютер опановували тільки ті, хто безпосередньо його обслуговував, то в сучасному світі без комп’ютерної обробки інформації не обходиться жодна галузь діяльності.

У цьому посібнику викладено основи дискретної математики та висвітлено їх практичне застосування в системах керування та прийняття рішень.

Матеріал, розглянутий у книзі, буде сприяти досягненню таких компетенцій:

– здатність до абстрактного мислення, аналізу та синтезу;

– спроможність застосовувати знання у практичних ситуаціях;

– знання та розуміння предметної галузі та розуміння професійної діяльності;

– здатність використовувати системний аналіз як сучасну міждисциплінарну методологію, що базується на прикладних математичних методах та сучасних інформаційних технологіях і орієнтована на вирішення задач аналізу і синтезу технічних, економічних, соціальних, екологічних та інших складних систем;

– спроможність формалізувати проблеми, описані природною мовою, у тому числі за допомогою математичних методів, застосовувати загальні підходи до математичного моделювання конкретних процесів.

Перелічені компетентності мають забезпечити такі результати навчання:

– знати і вміти застосовувати на практиці дискретну математику в обсязі, необхідному для вирішення типових завдань системного аналізу;

– вміти розпізнавати стандартні схеми для розв’язання комбінаторних та логічних задач, що сформульовані природною мовою; застосовувати класичні алгоритми для перевірки властивостей та класифікації об’єктів, множин, відношень, графів, булевих функцій тощо;

– використовувати на практиці розрахунки на графах, аналітичні перетворення множин, методи визначення істинності висловлювань та синтез автомата Мілі.

– знати положення формальних систем та методи застосування теорії граматик і математичної логіки числення предикатів першого порядку

Посібник поділено на п’ять розділів.

Перший присвячено теорії множин. Тут подано основні поняття, які є основою дискретної математики. Розглянуто базові операції з множинами, бінарні відношення і їх властивості, а також відображення і функції.

Другий розділ вивчає елементи теорії графів. Завдяки наочності й універсальності мова графів стала однією з найбільш поширених для задач керування. Саме дослідження в цій галузі привели до появи поняття об’єктивно складних задач, тобто тих, складність яких не можна усунути підвищенням потужності обчислювальних систем. Більшість понять, які вивчають у теорії графів, мають самостійне теоретичне і прикладне значення. Тут наведено основні визначення теорії графів, розглянуто задачі пошуку найкоротшого шляху, максимального потоку і алгоритми їх розв’язування, приклад розрахунку мережевого графіка.

Третій розділ має на меті вивчення елементів математичної логіки і теорії автоматів, які є традиційними для курсу дискретної математики. Тут розглянуто базові закони математичної логіки і методики роботи з логічними виразами. Викладено основи теорії автоматів для класу кінцевих автоматів. Докладно описано питання синтезу і мінімізації автоматів.

У четвертому розділі мова йде про принципи побудови і організації формальних систем, знання яких стає важливим елементом математичної культури будь-якого дослідника, котрий має відношення до алгоритмізації процесів керування та обробки інформації. Тобто вони дають розуміння про те, що можна і чого не можна зробити за допомогою обчислювальних машин.

П’ятий розділ включає елементи комбінаторики, які є необхідними для систем, пов’язаних із генеруванням варіантів прийняття рішень.

Усі розділи книги містять контрольні питання та задачі для самостійного розв'язування, що сприятиме засвоєнню навчального матеріалу.

Посібник написано з використанням матеріалу лекцій, читаних в НТУ «Дніпровська політехніка». Автори сподіваються, що ця книга буде для студентів, котрі вперше починають вивчати викладену тут теорію, своєрідним вступом до засвоєння широкого класу нових задач і методів їх розв'язування.

РОЗДІЛ 1

ОСНОВИ ТЕОРІЇ МНОЖИН

Мета розділу: вивчення базових понять теорії множин та набуття навичок їх практичного застосування при описі об’єктів

1.1. Основні визначення

У математиці зазвичай мають справу з найрізноманітнішими множинами. Наприклад, можна говорити про множину граней багатогранника, точок на прямій, цілих або раціональних чисел, неперервних функцій та ін. Поняття множини настільки загальне, що йому важко дати скільки-небудь зрозуміле визначення, яке не зводилося б до заміни слова "множина" синонімами: сукупність, набір елементів і т. ін.

Множина являє собою деякий набір об’єктів, які не повторюються і називаються *елементами*.

Елементами множини називають об’єкти, з яких вона складається.

Множини позначають великими літерами латинського алфавіту.

Елементи множини позначають малими літерами.

Отже, записати множину можна у такий спосіб:

$$A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}, \tag{1.1}$$

тут A – множина, $\alpha_1, \alpha_2, \ldots, \alpha_n$ – її елементи.

Наприклад, множина A може складатися з натуральних чисел 1, 2,…,6 при цьому її елементами будуть $\alpha_1 = 1$, $\alpha_2 = 2$, …, $\alpha_6 = 6$. Тоді ми запишемо її таким чином: $A = \{1, 2, 3, 4, 5, 6\}$.

Належність елемента a до множини A можна записати в такий спосіб: $a \in A$. Якщо елемент a не належить множині A, записують: $a \notin A$.

Будь-який набір, кожний елемент якого належить множині A, є його *підмножиною*. Так, множина $B = \{1, 2, 3\}$ є підмножиною множини $A = \{1, 2, 3, 4, 5, 6\}$. Цей факт записується таким чином: $B \subset A$.

Якщо множина не містить жодного елемента, вона називається *порожньою* і позначається символом $\varnothing$.

Підмножина, яка не співпадає із вихідною множиною і не є порожньою називається її *власною підмножиною*.

Всі використані в дискретній математиці множини містять елементи з найбільшої множини S, яка називається *простором елементів*. Отже, всі використані множини є підмножинами S.

У загальному випадку, якщо множина S містить n елементів, в ній можна виділити 2^n підмножин.

Приклад 1.1. Нехай $S = \{1, 2, 3, 4\}$. Розглянемо всілякі підмножини множини S. З урахуванням порожньої підмножини $\varnothing$ в множині S може бути виділено в цілому $2^4 = 16$ підмножин, а саме:

$\varnothing$,

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

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

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

$\{1, 2, 3, 4\} = S$,

Однієї з причин застосування теорії множин у дискретній математиці є той факт, що для множин визначено важливі перетворення, які мають просте геометричне тлумачення, використовуючи так звану *діаграму Ейлера*[1] *– Венна*[2]. У ній простір S подається у вигляді квадрата, а множини – у вигляді плоских фігур, обмежених замкненими лініями.

Наприклад, на рис.1.1 показано діаграму Ейлера – Венна для трьох множин C, U, A, де $C \subset U \subset A$.

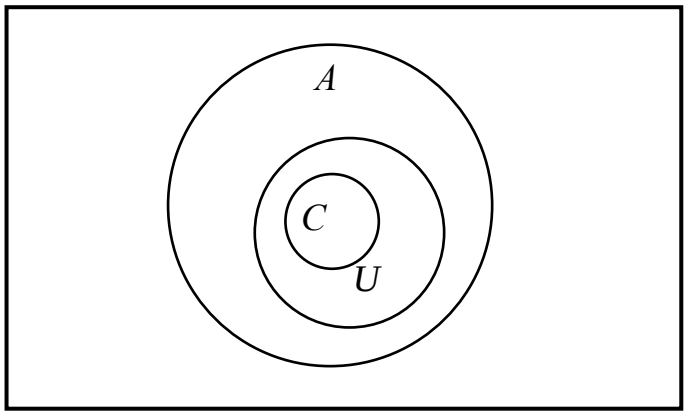


Рис.1.1. Діаграма Ейлера – Венна для визначення множин $C \subset U \subset A$.

Розглянемо сукупності, що складаються з m об'єктів. Якщо усі m об'єктів із сукупності M попарно різні, то таку сукупність називають *множиною*, або *m-*

[1] Леонард Ейлер (1707 – 1783) – великий математик-енциклопедист та механік XVIII сторіччя, який зробив фундаментальний внесок у розвиток цих наук (а також фізики, астрономії та інших прикладних наук). Вважається одним з из найвидатніших математиков в історії, автор більш ніж 850 робіт (серед них близько 20 фундаментальних монографій) з математичного аналізу, диференціальної геометрії, теорії чисел, наближених обчислень, небесної механіки, математичної фізики, оптики, балістики, кораблебудування, теорії музики та ін. Вивчав медицину, химію, ботаніку, повітроплавання, велику кількість європейських та давніх мов. Використовував кругові ілюстрації (кола Ейлера) для ілюстрації операцій з множинами.

[2] Джон Венн (1834 – 1923) – англійський математик, логік і філософ, запропонував графічний спосіб зображення формул математичної логіки. Основною сферою його інтересу була логіка, він опублікував три роботи з цієї теми: «Логіка випадку» (англ. The Logic of Chance) 1866 р., в якій вводиться інтерпретація частоти або частотна теорія ймовірностей; «Символьна логіка» (англ. Symbolic Logic) 1881 р., де були введені діаграми Венна; «Принципи емпіричної логіки» (The Principles of Empirical Logic)1889 р., в якій наводяться обґрунтування зворотних операцій у булевій логіці. Венн розширив математичну логіку Буля і найбільше відомий серед математиків і логіків через введений ним схематичний спосіб подання множин та їх об'єднань й перетинів.

елементною множиною, при цьому число m називають *кардинальним числом* або *потужністю* множини M. Зазвичай це записують у такий спосіб: $m = |M|$.

Отже, потужність множини – це те спільне, що є в будь-яких двох еквівалентних множин. Для скінченної множини поняття потужності збігається із поняттям кількості її елементів.

Потужність порожньої множини дорівнює нулю, тобто якщо |M| = 0, то M *– порожня множина*, $M = \varnothing$.

Якщо серед об'єктів, що входять у сукупність M, є однакові, то таку сукупність називають *мультимножиною*.

Усі множини можна поділити на два класи: скінченні й нескінченні.

Скінченними називають множини, кількість елементів яких скінченна. Наприклад, це множина студентів, які навчаються в одній групі; множина вершин багатогранника; множина жителів міста; множина сполучень, які можна скласти з n елементів та ін.

Множини, які мають нескінченне число елементів, називають *нескінченними*. Наприклад, множина N всіх натуральних чисел, множина всіх точок на прямій.

Потужність скінченної множини – це деяке натуральне число, тобто скінченна величина. Якщо x є елементом множини M, то цей факт записують у вигляді $x \in M$ і кажуть «x належить M». Запис $x \in M \mid P(x)$ означає, що розглядається множина, яка складається з елементів, що мають властивість P, а записи $M_1 = \{ x \in M: P(x)\}$ та $M_1 = \{ x \in M \mid P(x)\}$ означають, що розглядається частина множини M, причому $x \in M_1 \Leftrightarrow (x \in M \,\&\, P(x))$. (Позначення: $A \Rightarrow Y$ – із твердження A випливає твердження Y; $A \Leftrightarrow B$ – з A випливає B і навпаки).

Можливим є факт, що $M_1 = \varnothing$, або $M_1 = M$. У будь-якому випадку про множину M_1 кажуть, що вона являє собою підмножину множини M, і пишуть $M_1 \subseteq M$. Якщо $M_1 \subseteq M$ і $M \subseteq M_1$, то пишуть $M_1 = M$. В цьому випадку множини M_1 та M називають *рівними* множинами. Якщо $M_1 \subseteq M$, але $M_1 \neq M$, то пишуть $M_1 \subset M$, і говорять, що множина M_1 *строго включена* в множину M. Якщо ж $M_1 \subseteq M$, то кажуть про *включення* M_1 у M.

Припустимо, $A \subseteq R$. Будемо позначати надалі $A^+ = \{x \in A : x > 0\}$, $R^+ = \{ x \in R : x > 0\}$, $Z^+ = \{ x \in Z : x > 0\} = \{1, 2, \ldots n\ldots,\} = N$.

Якщо хоча б одним засобом можна пронумерувати (перелічити) за допомогою усіх натуральних чисел $n \in N$ всі елементи нескінченної множини M, то кажуть, що M має *зліченну потужність* або є *зліченною множиною*, і пишуть $|M| = |N|$. Замість $|N|$ пишуть іноді одну літеру α (готичне а). Наприклад, множина Z зліченна, тому що це легко вбачається в наступному записі чисел одне під іншим :

$$Z = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\},$$
$$N = \{\ldots, 6, 4, 2, 0, 1, 3, 5, \ldots\}.$$

У загальному випадку, якщо між елементами множин X та Y можна встановити взаємно однозначну відповідність (тобто кожному елементу $x \in X$ поставити у відповідність один і тільки один елемент $y \in Y$, і при цьому кожному

елементу $y \in Y$ також буде поставлений у відповідність єдиний елемент $x \in X$), то кажуть, що множини X і Y мають одне і те саме кардинальне число, або мають однакову *потужність*, або є *рівнопотужними*, і пишуть $|X| = |Y|$. Запис $|X| < |Y|$ означає, що $|X| \neq |Y|$, але для деякої підмножини $Y_1 \subset Y$ виконується $|X| = |Y_1|$.

1.2. Операції з множинами

Розглянемо тепер операції, які можна виконувати із множинами.

Рівність множин. Множини A та B називають *рівними* тоді і тільки тоді, коли кожний елемент множини A є елементом множини B, і навпаки, кожний елемент множини B є елементом множини A, тобто

$$A \subseteq B \text{ й } B \subseteq A.$$

Рівність множин A і B показана за допомогою діаграми Ейлера – Венна на рис. 1.2.

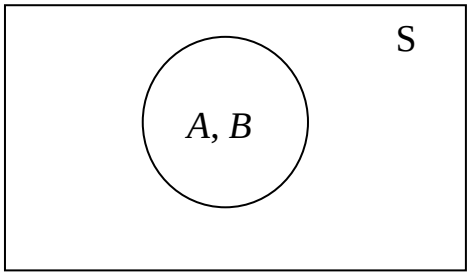


Рис.1.2. Зображення поняття «Рівність множин»

Об'єднання множин. *Об'єднанням* або *сумою* двох множин A та B називається множина, що складається з усіх елементів, кожний із яких належить хоча б одній з даних множин (рис. 1.3). Наприклад, якщо множина $A = \{2,5,8\}$, $B = \{0,5,6,7\}$, то їх об'єднання $A \cup B = \{0,2,5,6,7,8\}$.

Зображення об'єднання множин за допомогою діаграм Ейлера бачимо на рис.1.2, де штрихуванням показано результат об'єднання множин A та B.

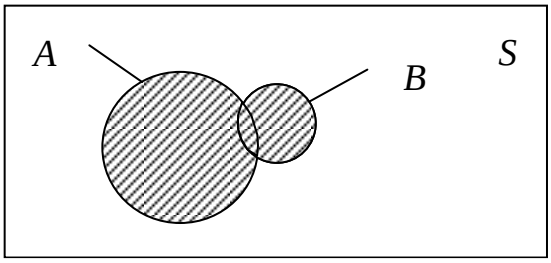


Рис.1.3. Графічне подання об'єднання множин за допомогою діаграм Ейлера

Відносно операції об'єднання будуть справедливими такі закони:

асоціативний: $(A \cup B) \cup C = A \cup (B \cup C)$;

комутативний: $A \cup B = B \cup A$.

Крім того, для будь-яких множин A та B справедливими також є співвідношення:

$A \cup A = A;\quad A \cup \varnothing = A;\quad A \cup S = S;\quad A \cup B = B$, якщо $A \subset B$.

Перетин множин. *Перетином* або *добутком* двох множин A і B (позначається $A \cap B$) називають сукупність елементів, які належать множинам A і B одночасно. Наприклад, перетином множин A = {2, 5, 8} і B = {0, 5, 6, 7} буде множина $A \cap B = \{5\}$.

Ілюстрацію операції перетину множин за допомогою діаграм Ейлера показано на рис.1.4.

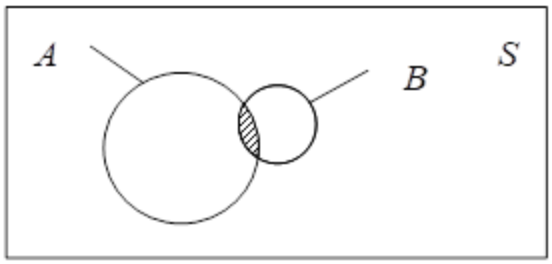


Рис.1.4. Графічне подання перетину множин A і B

Для цієї операції будуть справедливими такі закони:

асоціативний: $(A \cap B) \cap C = A \cap (B \cap C)$;

комутативний: $A \cap B = B \cap A$.

Крім того, для всіляких множин A та B правильними є також співвідношення:

$$A \cap A = A;\ A \cap \varnothing = \varnothing;\ A \cap S = A;\ A \cap B = A,\ якщо\ A \subset B.$$

Операції об'єднання й перетину пов'язані дистрибутивними законами, а саме:

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

У цьому неважко переконатися за допомогою діаграм Ейлера (рис.1.5).

а *б*

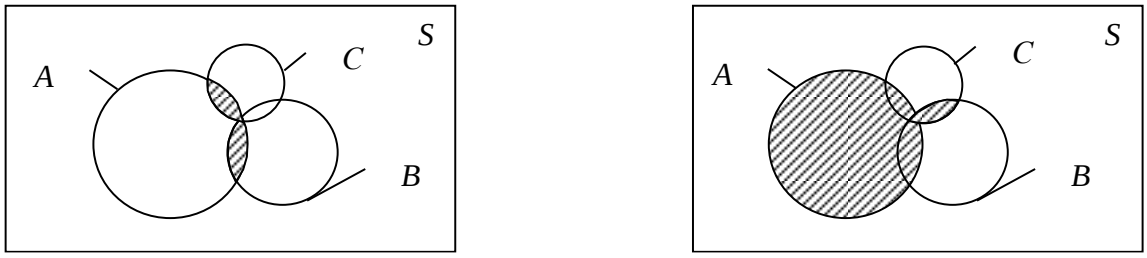


Рис. 1.5. Графічне подання дистрибутивних законів за допомогою діаграм Ейлера: *а* – $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$, *б* – $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Дві множини A и B є *взаємовиключними,* або несумісними, якщо їхнім перерізом є порожня множина, тобто $A \cap B = \varnothing$.

Доповнення множин. *Доповненням* множини *A* до універсальної множини *S* називається множина $\overline{A}$, яка включає в себе всі елементи універсальної множини *S*, крім тих, що належать множині *A*. На рис. 1.6. подано зображення доповнення множини $\overline{A}$ за допомогою діаграм Ейлера.

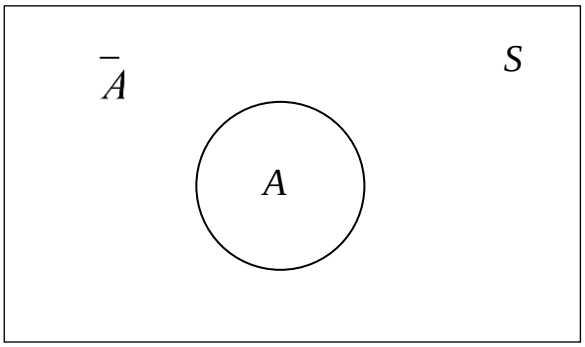


Рис.1.6. Графічна інтерпретація доповнення множини *A*

Відносно операції доповнення будуть справедливими такі твердження:

$$\overline{\varnothing} = S;\ \overline{S} = \varnothing;\ \overline{\overline{A}} = A;$$
$$A \cup \overline{A} = S;\ A \cap \overline{A} = \varnothing.$$

Крім того, із поняттям доповнення пов'язані закони де Моргана, а саме:

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

Завдання: доведіть справедливість цих співвідношень, використовуючи діаграми Ейлера.

Різниця. *Різницею* двох множин A та B (позначається як $A - B$ або A / B) називається множина, що складається з тих елементів множини A, які не містяться в множині B. На рис.1.7 її показано штриховкою.

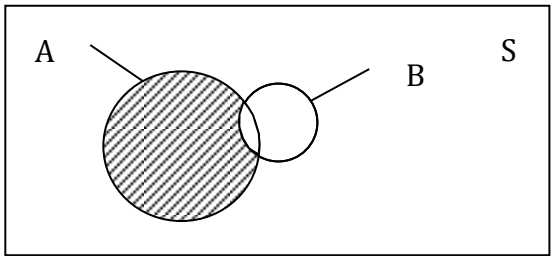


Рис.1.7. Графічне подання різниці множин *A* та *B*

Зауважимо, що різниці $A\backslash B$ та $B\backslash A$ не співпадають (див рис. 1.8).

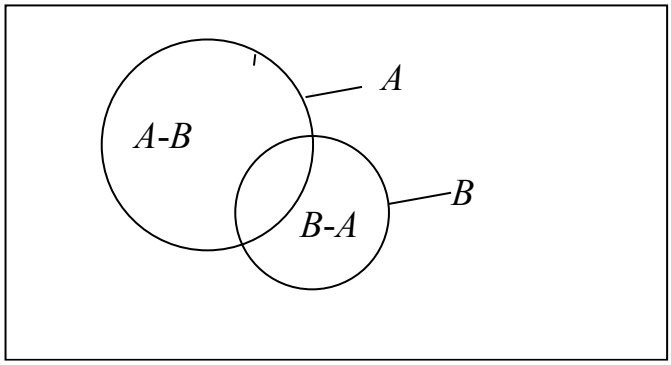


Рис.1.8. Графічна інтерпретація різниці множин A та B

Приклад 1.2. Нехай множини $A = \{2, 5, 8\}$, $B = \{0, 5, 6, 7\}$, тоді їх різницею буде множина $A - B = \{2, 8\}$.

Відносно різниці множин будуть справедливими такі твердження:

$$A - A = \varnothing;$$

$$A - \varnothing = A;$$

$$A - S = \varnothing;$$

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

Запам'ятайте! Вираз, який включає різницю, необхідно записувати із використанням дужок.

Розглянемо приклад, який ілюструє описані вище операції з множинами.

Приклад 1.3. Припустимо, що елементами простору S є натуральні числа від 1 до 6, тобто $S = \{1, 2, 3, 4, 5, 6\}$, і на цьому просторі задано такі підмножини:

$$A = \{2, 4, 6\};\ B = \{1, 2, 3, 4\};\ C = \{1, 3, 5\}.$$

З огляду на наведені співвідношення можна записати:

$$A \cup B = \{1,2,3,4,6\},\ \ B \cup C = \{1,2,3,4,5,6\},$$

$$A \cup B \cup C = \{1,2,3,4,5,6\} = S,$$

$$A \cap B = \{2,4\},\ \ B \cap C = \{1,3\},\ \ A \cap C = \varnothing,$$

$$A \cap B \cap C = \varnothing,$$

$$\overline{A} = \{1,3,5\} = C,\ \ \overline{B} = \{5,6\},\ \ \overline{C} = \{2,4,6\} = A.$$

$$A - B = \{6\},\ B - A = \{1,3\},$$

$$A - C = \{2,4,6\} = A,\ \ C - A = \{1,3,5\} = C,$$

$$B - C = \{2,4\},\ \ C - B = \{5\}.$$

Завдання. Для закріплення матеріалу проілюструйте наведені вище операції за допомогою діаграм Ейлера – Венна.

1.3. Розбиття множин

Розглянемо приклад. Припустимо, що A – множина учнів певної школи. Позначимо через A_1 множину учнів першого класу, A_2 – множину учнів другого класу і т.д., A_{12} – множину учнів дванадцятого класу цієї школи. Ясно, що кожна з множин $A_1, A_2, \ldots, A_{12}$ є підмножиною множини A, причому всі вони не порожні (у кожному класі є учні). Ці множини попарно не перетинаються (не мають спільних елементів), оскільки один і той самий учень не може навчатися одночасно в двох різних класах, і об'єднання всіх цих множиг дорівнює A.

Говорять, що сукупність підмножин множини M утворює його розбиття, якщо виконано такі три умови: кожна з них не порожня, будь-які дві з них не мають спільних елементів і їх об'єднання дорівнює М.

Відтак, у розглянутому вище прикладі сукупність підмножин $A_1, A_2, \ldots, A_{12}$ утворює розбиття множини A.

Якщо, наприклад, $A = \{1, 2, 3, 4, 5, 6\}$, то сукупність таких підмножин множини A: $\{1, 3\}, \{2, 4, 6\}, \{5\}$ утворить його розбиття, а сукупність підмножин $\{1, 2, 3\},\{4, 5\},\{3, 6\}$ – ні, оскільки дві з них мають непорожній переріз.

Отже, під розбиттям множини ми будемо розуміти «розбивку» його на частини або підмножини. Ясно, що одна і та сама множина може бути «розбита» на частини по-різному. Так, наприклад, якщо множина $B = \{a, b, c, d, e, f, g\}$, то сукупності її підмножин $\{a, b, c, \}, \{d, e, f, g\}$, і $\{a, b\}, \{c, d\}, \{e, f, g\}$ утворюють два її різні розбиття. Тобто множину B «розбито» на підмножини по-різному.

Розбиття множини будемо позначати малими літерами грецького алфавіту: ρ, σ, δ і т.д.

Класи розбиття. Нехай дано множину M і деяке її розбиття σ. Кожна з підмножин сукупності, яка утворює розбиття ρ, називається *класом розбиття*.

Так, наприклад, якщо $B = \{a, b, c, d, e, f, g\}$ і σ – розбиття множини B, яке утворене такою сукупністю його підмножин: $\{a, b\}, \{e, d, f\}, \{c\}, \{g\}$, то кожна з цих підмножин є класом даного розбиття σ.

Якщо зобразити множину за допомогою діаграми Ейлера – Венна, то її розбиття можна уявити наочно, у вигляді поділеного на частини кола (рис. 1.9). Кожна така частина і є класом даного розбиття.

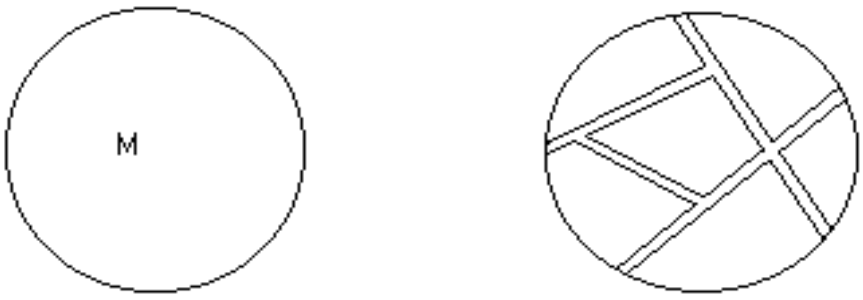


Рис.1.9. Розбиття множини M

У одному і тому самому класі розбиття множини M може утримуватися декілька (і навіть нескінченно багато) елементів із M. Якщо елементи x, y множини M належать до одного і того ж самого класу даного розбиття ρ, то цей факт ми будемо позначати в такий спосіб: $x \sim y$ (ρ).

Наприклад, для розглянутого вище розбиття σ множини B можна записати $a \sim b$ (σ), $d \sim e$ (σ), а запис $c \sim d$ (σ) є помилковим, оскільки c і d належать до різних класів розбиття σ. Враховуючи, що елементи a і a, зазвичай, належать до одного класу розбиття, позаяк це один і той самий елемент, то можна записати $a \sim a$(σ). Аналогічно $b \sim b$(σ), $c \sim c$(σ) і т.і.

Припустимо далі, що для деякого розбиття ρ множини M і деяких елементів x, y, z множини M мають місце такі твердження: $x \sim y$ (ρ) і $y \sim z$ (ρ) . Запис $x \sim y$ (ρ) означає, що елементи x і y належать до одного і того ж самого класу нашого розбиття. Аналогічно y і z знаходяться в одному й тому самому класі. Якщо тепер припустити, що x і z лежать у різних класах розбиття ρ, то одержимо, що ці класи мають загальний елемент y, а це неможливо. Виходить, що $x \sim z$ (ρ).

Відтак, ми довели, що коли $x \sim y$ (ρ) і $y \sim z$ (ρ), то $x \sim z$ (ρ).

Кожний клас даного розбиття ρ множини M, за визначенням, не порожній, тобто в кожному класі знаходяться елементи множини M.

Наприклад, для розбиття σ множини B, розглянутої вище, клас розбиття $\{a, b\}$ можна позначити $\overline{a}$, оскільки він містить елемент a. Але цей клас містить також елемент b, отже, його можна позначити і через $\overline{b}$. Таким чином, $\{a, b\} = \overline{a} = \overline{b}$. Аналогічно $\{e, d, f\} = \overline{e} = \overline{d} = \overline{f}$.

Отже, як бачимо, один і той самий клас розбиття можна позначити по-різному. Це не має вас дивувати. І в математиці, і в житті ми не одноразово зустрічалися з такою ситуацією. Наприклад, дроби 1/2, 2/4, 3/6, … – це одне і те саме число, яке просто записано по-різному.

Повернемося тепер до загального випадку розбиття ρ множини M. Якщо деякий клас розбиття містить елементи x і y множини M, то, з одного боку, він позначається $\overline{x}$, а з другого – $\overline{y}$, тобто $\overline{x} = \overline{y}$. Відтак, ми довели, що з $x \sim y$ (ρ) випливає $\overline{x} = \overline{y}$.

Припустимо тепер, що $\overline{x} = \overline{y}$. Але $\overline{x}$ – це клас, що містить елемент x, $\overline{y}$ – це клас, що містить елемент y, і ці класи рівні, тобто це один і той самий клас. Виходить, що x і y перебувають в одному класі розбиття, й $x \sim y$ (ρ). Отже, можна зробити висновок, що $\overline{x} = \overline{y}$ тоді і тільки тоді, коли $x \sim y$ (ρ).

Фактор-множина. Розглянемо побудоване вище розбиття σ множини B. Підмножини $\{a, b\}$, $\{e, d, f\}$, $\{c\}$, $\{g\}$ є класами цього розбиття. Відповідно до нашої домовленості, їх можна позначити, наприклад, $\overline{a}$, $\overline{e}$, $\overline{c}$, $\overline{g}$ відповідно. Таким чином, із розбиттям σ пов'язана сукупність його класів розбиття, тобто множина $\{ \overline{a}, \overline{e}, \overline{c}, \overline{g} \}$.

В загальному випадку, якщо ρ – деяке розбиття довільної множини M, то з цім розбиттям пов'язано сукупність його класів.

Сукупність усіх класів даного розбиття ρ множини *M* називається *фактор-множиною* множини *M* за розбиттям ρ і позначається *M*/ρ.

Відтак, елементами множини *M* /ρ є класи даного розбиття ρ і інших елементів у ньому немає.

Якщо ми повернемося до розбиття ρ множини *M*, зображеної на рис.1.9, то множина *M* /ρ складається зі шматків (рис. 1.10), на які ми розрізали коло, котре відповідає множині *M*. Далі для розглянутого вище розбиття множини *B* маємо B/σ = { $\bar{a}$, $\bar{e}$, $\bar{c}$, $\bar{g}$ }. Звісно ж елементи множини *B*/σ можна позначити і по-іншому, і тоді, наприклад, *B*/σ = { $\bar{b}$, $\bar{f}$, $\bar{c}$, $\bar{g}$}.

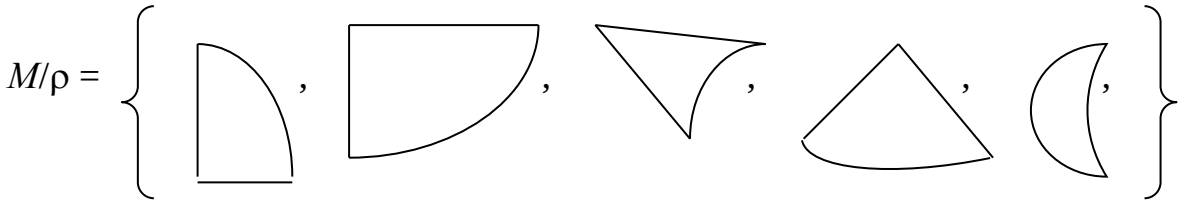


Рис.1.10. Компоненти розбитої множини

Нехай тепер δ – розбиття множини *B*, класами якого є підмножини {*a, b, c, d*}, {*e, f, g*}. Тоді *B*/δ = { $\bar{a}$, $\bar{e}$}. Звертаємо вашу увагу на те, що $\bar{a}$ в *B*/σ і $\bar{a}$ в *B*/δ – це різні елементи. Наприклад, ТРЦ "Україна" у Харкові і ТРЦ "Україна" у Києві – це різні торгівельні центри, і хоча вони називаються однаково, це не викликає непорозумінь. Не буде їх і у випадку із розбиттями, оскільки завжди ясно про яке саме розбиття йде мова.

Зауважимо, що *M* /ρ – це сукупність лише деяких (не усіх) підмножин множини *M*, а 2^M – сукупность всіх підмножин цієї множини. Виходить, що *M* /ρ $\subset 2^M$, більш того, *M* /ρ являє собою власну підмножину множини 2^M.

Ясно, що коли множина *M* нескінченна, то будь-яка її фактор-множина в залежності від розбиття може бути як скінченною, так і нескінченною.

1.4. Декартів добуток множин

Введемо тепер деякі поняття, необхідні для опису операцій над множинами.

Вектор являє собою деякий упорядкований набір елементів. Інший синонім цього терміну – "*кортеж*".

Елементи, що утворюють вектор, називаються його *координатами* або *компонентами*. Координати нумеруються зліва направо. Кількість координат називається *довжиною* або *розмірністю* вектора. Вектор записують, використовуючи круглі дужки, наприклад, (0, 5, 4, 5). Іноді дужки і навіть коми опускаються.

Вектори, довжина яких дорівнює двом, часто називаються *упорядкованими парами.*

Два вектори називають *рівними*, якщо вони мають однакову довжину і відповідні їх координати дорівнюють одна одній. Інакше кажучи, вектори $(a_1, \dots, a_m)$ і $(b_1, \dots, b_n)$ будуть рівними, якщо $m = n$ і $a_1 = b_1$, $a_2 = b_2$, ... , $a_m = b_n$.

Прямим (декартовим) добутком множин A та B (позначають $A \times B$) називається множина усіх пар (a, b), при яких $a \in A$ і $b \in B$. Зокрема, якщо $A = B$, то обидві координати належать множині A. Такий добуток позначається A^2. Аналогічно прямим добутком множин $A_1, \dots, A_n$ (позначаємо через $A_1 \times A_2 \times \dots \times A_n$) називається множина усіх векторів $(a_1, \dots, a_n)$ довжини n, коли $a_1 \in A_1$, $a_2 \in A_2$, … , $a_n \in A_n$. Добуток $\underbrace{A \times A \times \dots \times A}_{n \text{ разів}}$ позначається через A^n.

Приклад 1.4. $R^2 = R \times R$ – це множина точок площини, точніше пар (a, b), де $a, b \in R$ і є координатами точок площини.

Приклад 1.5. Якщо $A = \{a,b,c,d,e,f,g,h\}$, $B = \{1,2,3,4,5,6,7,8\}$, тоді $A \times B = \{a1, a2, a3, a4, a5, a6, a7, a8, b1, \dots, h7, h8\}$ – множина, що містить позначення всіх 64 клітинок шахівниці.

Приклад 1.6. Розглянемо множину числових матриць 3×4, тобто матриць такого вигляду:

$$\begin{Vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{Vmatrix},$$

де a_{ij} належить множині R дійсних чисел.

Рядки матриці – це елементи множини R^4 (вектори розмірністю 4). Сама матриця, розглянута як впорядкований набір (тобто вектор) рядків, – це елемент множини $\left(R^4\right)^3 = R^4 \times R^4 \times R^4$.

Компонентами матриці, заданої у такий спосіб є рядки, а не числа. Тому $(R^4)^3 \neq R^{12}$. Змістовний сенс цієї нерівності в тому, що у векторі R^{12} не міститься ніякої інформації про будову матриці, той самий вектор міг би перераховувати елементи 4×3 або 2×6, а ці математичні об'єкти не співпадають із матрицями 3×4.

1.5. Відношення

Підмножина $R \subseteq M^n$ називається n-місним відношенням на множині M. Будемо говорити, що елементи $a_1, a_2, \dots a_n$ знаходяться у відношенні R, якщо $(a_1, a_2, \dots a_n) \in R$.

Одномісне відношення – це просто підмножина множини M. Такі відношення називають *ознаками*. Говоримо, що елемент a має ознаку R, якщо $a \in R$, $R \subseteq M$. Властивості одномісних відношень – це особливості підмножин множини M; тому для випадку $n = 1$ термін відношення вживається рідко.

Прикладом тримісного відношення є множина трійок нападників у хокейній команді. Кожний гравець знаходиться у цьому відношенні з тими, з ким він грає в трійці.

Серед найбільш вивчених і часто використовуваних є двомірні або *бінарні* відношення. Факт, що елементи a та b перебувають у бінарному відношенні R, можна записати $a\,R\,b$, $(a,b)\in R$. Відношення R, задане на множині Ω, позначимо як (R,Ω).

Приклад 1.7. Розглянемо відношення, які задано на множині N натуральних чисел:

– відношення « $\le$ » виконується для пар (7, 9) і (7, 7), але не для пари (9, 7);

– відношення «мати спільний дільник, відмінний від одиниці» виконується для пар (6, 9), (4, 2), (2, 4), (4, 4), але не для пар (7, 9) і (9, 7);

– відношення «бути дільником» ($a\,R\,b$ коли a дільник b) виконується для пар (2, 4) і (4, 4), але не для пар (4, 2) і (7, 9).

Приклад 1.8. Відношення, які задано на множині точок дійсної площини:

– відношення «знаходитися на однаковій відстані від початку координат» виконується для пар точок, які знаходяться на одному й тому самому колі з центром на початку координат;

– відношення «знаходитися на різній відстані від початку координат» виконується для тих і тільки тих пар точок, для яких не виконується попереднє відношення;

– відношення «бути симетричним щодо вісі X» виконується для усіх пар точок (x_1, y_1) і (x_2, y_2), що задовольняють умові $x_1 = x_2$ і $y_1 = -y_2$.

Приклад 1.9. Відношення, які задано на множині людей: «жити в одному місті», «бути молодше», «бути сином», «вчитися в одному університеті».

Аби задати відношення (R,Ω), необхідно задати всі пари елементів $(x,y)\in\Omega\times\Omega$, які включено в множину R. Крім повного переліку всіх пар, існують три способи задання відношень: за допомогою матриці, графа й розрізів. Перші два способи застосовують, щоб задати відношення на скінченних множинах, задання відношення розрізами може бути застосовано й до нескінченних множин.

Опишемо названі способи задання відношень.

Задання відношення за допомогою матриці. Нехай множина Ω складається з n елементів, R – подане на цій множині бінарне відношення. Пронумеруємо елементи множини Ω цілими числами від 1 до n. Аби задати відношення, побудуємо квадратну таблицю розміром $n\times n$. Її i-й рядок відповідає елементу x_i множини Ω, j-й стовпець – елементу x_j з множини Ω. На перетині i-го рядка та j-го стовпця ставимо 1, якщо елемент x_i перебуває у відношенні R з елементом x_j, і нуль в інших випадках, а саме:

$$a_{ij}(R)=\begin{cases}1, & x_i\ R\ x_j,\\ 0 & \text{в інших випадках.}\end{cases}$$

П р и к л а д 1.10. Нехай $X=\{1,2,\ldots,5\}$, R – відношення “більше” на множині X. Тоді його можна описати у вигляді матриці:

$$R=\begin{pmatrix}0&0&0&0&0\\1&0&0&0&0\\1&1&0&0&0\\1&1&1&0&0\\1&1&1&1&0\end{pmatrix}.$$

П р и к л а д 1.11. Для скінченної множини {1, 2, ... ,6} матриці відношень R_1 – « $\le$ » , R_2 – «мати спільний дільник», R_3 – «бути дільником» мають такий вигляд:

$$R_1=\begin{pmatrix}1&1&1&1&1&1\\0&1&1&1&1&1\\0&0&1&1&1&1\\0&0&0&1&1&1\\0&0&0&0&1&1\\0&0&0&0&0&1\end{pmatrix},\quad R_2=\begin{pmatrix}0&0&0&0&0&0\\0&1&0&1&0&1\\0&0&1&0&0&1\\0&1&0&1&0&1\\0&0&0&0&1&0\\0&1&1&1&0&1\end{pmatrix},\quad R_3=\begin{pmatrix}1&1&1&1&1&1\\0&1&0&1&0&1\\0&0&1&0&0&1\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{pmatrix}$$

Задання відношення за допомогою графа. У загальному значенні графом називають непорожню множину вершин (вузлів), з’єднаних ребрами[3]. Аби задати відношення за допомогою графа, поставимо у взаємно однозначну відповідність елементам скінченної множини Ω, на якій визначено відношення, вершини графа $x_1,\ldots,x_n$ (за будь-якою нумерацією).

Провести дугу від вершини x_i до x_j можна тоді й тільки тоді, коли елемент x_i перебуває у відношенні R з елементом x_j, коли ж $i=j$, то дуга (x_i,x_j) перетворюється на петлю при вершині x_i.

П р и к л а д 1.12. Задамо відношення з прикладу 1.10 за допомогою графа.

[3] Строге математичне визначення графів, їхні характеристики, властивості і застосування буде розглянуто далі у розділі 2.

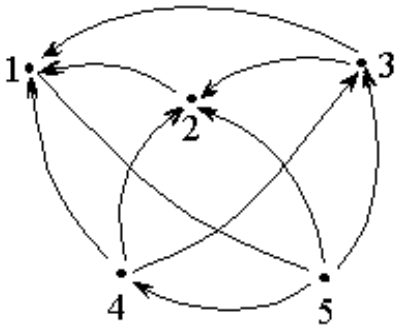


Рис. 1.11. Задання відношення “більше” на множині $X = \{1, 2, 3, 4, 5\}$ за допомогою графа

Отже, коли задано будь-який орієнтований граф *G*, що має *n* вершин, і вибрано нумерацію на множині Ω, яка складається з *n* елементів, то тим самим на цій множині задано деяке відношення $R = R(G)$, а саме, твердження $x_i \; R \; x_j$ буде справедливим тоді і тільки тоді, коли в графі *G* наявна дуга (x_i, x_j). Відтак, граф виступає як геометричне зображення відношення.

Задання відношень за допомогою розрізів. Розглянемо відношення *R* на множині Ω.

Верхнім розрізом відношення (R, Ω) в елементі *x*, що позначається через $R^+(x)$, називається множина елементів $y \in \Omega$, для яких виконано умову: $(y, x) \in R$, тобто

$$R^+(x) = \{y \in \Omega | (y, x) \in R\}. \qquad (1.2)$$

Нижнім розрізом $R^-(x)$ відношення (R, Ω) в елементі x називається множина елементів $y \in \Omega$, для яких $(x, y) \in R$, а саме:

$$R^-(x) = \{y \in \Omega | (x, y) \in R\}. \qquad (1.3)$$

Отже, верхній розріз (множина R^+) являє собою множину всіх таких елементів *y*, що перебувають у відношенні *R* з фіксованим елементом *x* $(y \; R \; x)$. Нижній розріз (множина R^-) – це множина всіх таких елементів *y*, з якими фіксований елемент *x* перебуває у відношенні *R* $(x \; R \; y)$.

Таким чином, аби задати відношення за допомогою розрізів, необхідно описати всі верхні або всі нижні його розрізи. Інакше кажучи, відношення *R* буде задано, якщо для кожного елемента $x \in \Omega$ задано множину $R^+(x)$ або для кожного елемента $x \in \Omega$ задано множину $R^-(x)$.

П р и к л а д 1.13. Нехай задано множину: $\Omega = \{1, 2, 3, ..., 10\}$. Відношення *R* означає «бути дільником», тобто $x \, R \, y$, якщо x – дільник y. Задати це відношення можна в такий спосіб:

за допомогою верхніх розрізів:

$R^+(1)=\{1\}$, $R^+(6)=\{1;2;3;6\}$
$R^+(2)=\{1;2\}$, $R^+(7)=\{1;7\}$,
$R^+(3)=\{1;3\}$, $R^+(8)=\{1;2;4;8\}$,
$R^+(4)=\{1;2;4\}$, $R^+(9)=\{1;3;9\}$,
$R^+(5)=\{1;5\}$, $R^+(10)=\{1;2;5;10\}$;

або за допомогою нижніх розрізів:

$R^-(1)=\{1;2,...,10\}$, $R^-(6)=\{6\}$,
$R^-(2)=\{2;4,...,10\}$, $R^-(7)=\{7\}$,
$R^-(3)=\{3;6;9\}$, $R^-(8)=\{8\}$,
$R^-(4)=\{4;8\}$, $R^-(9)=\{9\}$,
$R^-(5)=\{5;10\}$, $R^-(10)=\{10\}$.

Розглянемо відношення спеціального вигляду та описані вище способи їх задання.

Відношення називається *порожнім* (позначається $\varnothing$), якщо воно не виконується для жодної пари $(x, y) \subset \Omega\times\Omega$.

Для порожнього відношення справедливі такі твердження:

1. У матриці $A(\varnothing)$ величини $a_{i,j}(\varnothing) = 0$ для всіх значень i, j.
2. Граф $G(\varnothing)$ не має дуг.
3. $R^+(x) = R^-(x) = \varnothing$ для всякого елемента $x \in \Omega$.

Відношення називається *повним* (позначається U), якщо воно виконується для всіх пар $(x, y) \subset \Omega\times\Omega$. Для повного відношення правильні такі ознаки:

1. У матриці $A(U)$ величини $a_{i,j}(U) = 1$ для всіх значень i, j.
2. У графі $G(U)$ дуги з'єднують будь-яку пару вершин.
3. Розрізи $R^+(x) = R^-(x) = \Omega$ для всіх елементів $x \in \Omega$.

Відношення називається *діагональним* або *рівністю* (позначається E), коли воно виконується для всіх пар $(x, y) \subset \Omega\times\Omega$, які складаються із збіжних елементів. Тобто $x\,E\,y$, якщо x та y – це один і той самий елемент множини Ω. Для діагонального відношення E мають місце такі твердження:

1. У матриці $A(E)$

$$a_{i,j}(E)=\begin{cases}1, \textit{ якщо} & i=j,\\ 0 \textit{ в інших випадках.}\end{cases}$$

2. У графі $G(E)$ наявні тільки петлі при вершинах, інші дуги відсутні.
3. Розрізи $R^+(x) = R^-(x) = x$ для всіх елементів $x \in \Omega$.

Відношення називається *антидіагональним* (позначається $\overline{E}$), коли воно виконується для всіх пар $(x, y) \subset \Omega\times\Omega$, котрі складаються із незбіжних елементів. Для відношення $\overline{E}$ справедливі такі ознаки:

1. У матриці $A(E)$

$$a_{i,j}\left(\overline{E}\right)=\begin{cases}1, \textit{якщо} & i \neq j,\\ 0 \ \textit{в інших випадках.}\end{cases}$$

2. У графі $G(\overline{E})$ наявні всі дуги (x_i, x_j), якщо $i \neq j$ (відсутні тільки петлі при вершинах).

3. Розрізи $R^+(x) = R^-(x) = \Omega \setminus \{x\}$ для всіх елементів $x \in \Omega$.

1.6. Операції над відношеннями

Відношення R_1 *включено* у відношення R_2 (записується як $R_1 \le R_2$), коли множину пар, для яких виконується відношення R_1, включено в множину пар, для котрих виконується R_2.

Будемо говорити, що відношення R_1 *строго включено* в R_2 $(R_1 < R_2)$, якщо $R_1 \le R_2$ й $R_1 \neq R_2$. Рівність відношень реалізується так само, як і рівність множин.

Для матричного задання відношень буде діяти таке правило:

якщо $R_1 \le R_2$, то $a_{ij}(R_1) \le a_{ij}(R_2)$, $i, j = \overline{1,n}$.

П р и к л а д 1.14. R_1 – відношення « $\le$ » на множині дійсних чисел, R_2 – відношення « $<$ » на тій самій множині, тоді $R_2 \le R_1$.

Відношення $\overline{R}$ називається *доповненням* відношення R, тоді і тільки тоді, коли воно пов'язує тільки ті пари елементів, для яких не виконується відношення R.

Очевидно, що

$$\overline{\boldsymbol{R}} = \boldsymbol{\Omega}^2 \setminus \boldsymbol{R}. \tag{1.4}$$

З огляду на це в матричному записі

$$a_{ij}\left(\overline{R}\right) = 1 - a_{ij}(R), \quad i, j = \overline{1,n}. \tag{1.5}$$

У графі $G\left(\overline{R}\right)$ наявні ті і тільки ті дуги, що відсутні у графі $G(R)$.

Для розрізів відношення $\overline{R}$ справедливі такі твердження:

$$\overline{R}^+(x) = \Omega \setminus R^+(x),$$

$$\overline{R}^-(x) = \Omega \setminus R^-(x).$$

П р и к л а д 1.15. Нехай R – відношення «$\geq$», задане на множині дійсних чисел, тоді $\overline{R}$ – відношення «$<$», задане на тій самій множині.

Перетином відношень R_1 та R_2 (записується $R_1 \cap R_2$) називається відношення, визначене перетином відповідних підмножин множини Ω^2.

У матричному записі це означає, що

$$a_{ij}\left(R_1 \cap R_2\right) = \min\left\{a_{ij}\left(R_1\right), a_{ij}\left(R_2\right)\right\}, \quad i, j = \overline{1,n}. \tag{1.6}$$

Об'єднанням відношень R_1 та R_2 (позначається $R_1 \cup R_2$) називається відношення, отримане шляхом об'єднання відповідних підмножин множини Ω^2.

У матричному записі це можна подати таким чином:

$$a_{ij}\left(R_1 \cup R_2\right) = \max\left\{a_{ij}\left(R_1\right), a_{ij}\left(R_2\right)\right\}, \quad i, j = \overline{1,n}. \tag{1.7}$$

Оберненим до відношення R називається відношення R^{-1}, яке задовольняє таку умову:

$$x\, R^{-1} y \;\Leftrightarrow\; y\, R\, x. \tag{1.8}$$

Для матриць відношень R та R^{-1} буде мати місце така формула:

$$a_{ij}\left(R^{-1}\right) = a_{ji}(R). \tag{1.9}$$

П р и к л а д 1.16. Нехай R – відношення «$\geq$» на множині дійсних чисел. Тоді оберненим до нього відношенням R^{-1} буде відношення «$\leq$» на множині дійсних чисел.

П р и к л а д 1.17. Нехай відношення R задано на множині $X = \{x_1,\ x_2,\ x_3, x_4, x_5\}$ такою матрицею:

$$R = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

Побудувати відповідні йому обернене відношення та доповнення.

Розв'язування

Згідно з формулою (1.4) доповнення відношення R можна задати такою матрицею:

$$\overline{R} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

Обернене відношення будуємо, використовуючи формулу (1.9). Отже,

$$R^{-1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Добутком (або *композицією*) відношень R_1 та R_2 (позначається як $R_1 \cdot R_2$) називається відношення, яке будується за таким правилом: $x\,(R_1 \cdot R_2)\,y$, коли існує елемент $z \in \Omega$, який задовольняє умови $x\,R_1\,z$ та $z\,R_2\,y$.

П р и к л а д 1.18. Розглянемо відношення R_1 та R_2, подані на множині дійсних чисел. Причому, R_1 – відношення «менше», R_2 – відношення «більше». Пара чисел $(x, y) \subset R_1 \cdot R_2$, коли існує число z, для якого виконано такі вимоги: $x < z$ та $z > y$. Вочевидь, ця умова виконується для всіх чисел x, y, а тому $R_1 \cdot R_2$ – це повне відношення (тобто таке, що пов'язує всі елементи даної множини).

Відповідно до визначення, $x(R_1 \cdot R_2)y$, коли існує елемент $z \in \Omega$, який задовольняє умови $x\,R_1\,z$ та $z\,R_2\,y$. У матричному записі це означає, що

$$a_{ij}(R_1 \cdot R_2) = \max_{k=\overline{1,n}} \min\{a_{ik}(R_1), a_{kj}(R_2)\}, \qquad (1.10)$$

де n – порядок матриці.

Інакше кажучи, композиція відношень обчислюється як максимінний добуток відповідних їм матриць.

П р и к л а д 1.19. Нехай множина $X = \{x_1,\ x_2,\ x_3, x_4, x_5\}$, на ній подано два відношення R_1 та R_2, а саме:

$$R_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \qquad R_2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Визначити їх композицію.

Розв'язування

Для обчислення використовуємо формулу (1.10). Тоді отримуємо такий результат:

$$R_1 \cdot R_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

Відношення (R_1, Ω_1) називається *звуженням* відношення (R, Ω) на множину Ω_1, якщо $\Omega_1 \subset \Omega$ та $R_1 = R \bigcap \Omega_1 \times \Omega_1$. Звуження відношення (R, Ω) на множину Ω_1 називають також відношенням R на множині Ω_1.

Можна сказати, що R_1 утворюється з R видаленням усіх пар, що містять елементи, які не належать множині Ω_1. Строго кажучи, R і R_1 – це різні відношення із різними областями визначення. Проте, якщо не виникає протиріч, цього педантизму не дотримуються; наприклад, цілком можна говорити «бути дільником» не уточнюючи, задано воно на множині N або на якійсь його підмножині.

П р и к л а д 1.20. Відношення « >» на множині натуральних чисел є звуженням відношення « >» на множині дійсних чисел.

1.7. Властивості відношень

Відношення R називається *рефлексивним*, якщо $x\,R\,x$ для будь-якого елемента $x \in \Omega$, тобто кожен елемент множини перебуває у відношенні R сам із собою.

Наприклад, відношення «бути схожими», «бути не старшим», «менше або дорівнює» – рефлексивні; «бути братом», «бути старшим», «більше» – не рефлексивні.

У матриці рефлексивного відношення на головній діагоналі розміщуються одиниці, тобто елемент матриці $a_{ij} = 1,$ якщо $i = j$.

Граф рефлексивного відношення обов'язково має петлі при вершинах. Стосовно верхнього й нижнього розрізів справедливі твердження: $x \in R^+(x)$, $x \in R^-(x)$ для всіх елементів $x \in \Omega$.

Відношення R називається *антирефлексивним*, коли твердження $x\,R\,y$ означає, що $x \neq y$ для $\forall x \in \Omega$, тобто жоден елемент не перебуває у відношенні із собою.

У матриці антирефлексивного відношення елементи головної діагоналі дорівнюють нулю, тобто $a_{ij} = 0,$ якщо $i = j$.

Граф антирефлексивного відношення не має петель при вершинах, а верхні та нижні розрізи задовольняють такі умови: $x \notin R^{+}(x)$, $x \notin R^{-}(x)$ для всіх елементів $x \in \Omega$.

Антирефлексивними будуть відношення «більше», «менше», «бути старшим», «бути сином».

Відношення R називається *симетричним*, якщо $R = R^{-1}$ $(x\,R\,y \Rightarrow y\,R\,x)$.

Матриця симетричного відношення є симетричною, тобто $a_{ij} = a_{ji}$ для всіх значень i, j. У графі такого відношення всі дуги парні, а верхні й нижні розрізи збігаються для всіх елементів $x \in \Omega$, тобто $R^{+}(x) = R^{-}(x)$ $\forall x \in \Omega$.

Симетричними є відношення рівності, «бути схожим», «вчитися в одній групі», «бути симетричним щодо осі X» на множині точок площини.

Неважко переконається в тому, що відношення R є симетричним тоді і тільки тоді, коли $R = R^{-1}$.

Відношення R називається *асиметричним*, якщо $R \cap R^{-1} = \varnothing$ (тобто з двох виразів $x\,R\,y$ та $y\,R\,x$ хоча б один не відповідає дійсності).

У матриці асиметричного відношення $a_{ij} \wedge a_{ji} = 0$ для всіх значень i, j, тобто з двох симетричних елементів a_{ij} і a_{ji} хоча б один обов'язково дорівнює 0.

Асиметричними, наприклад, є відношення «більше» та «менше».

Зауважимо, що антирефлексивність – це обов'язкова умова асиметричності.

Відношення R називається *антисиметричним*, якщо твердження $x\,R\,y$ та $y\,R\,x$ можуть бути правильними одночасно тоді і тільки тоді, коли $x = y$.

У матриці антисиметричного відношення $a_{ij} \wedge a_{ji} = 0$, коли $i \neq j$.

Прикладами антисиметричних будуть відношення «більше або дорівнює», «менше або дорівнює», «не більше», «не гірше».

Розглянемо відношення «менше або дорівнює»: дійсно, якщо $a \le b$ й $b \le a$, то $a = b$. Антисиметричність інших відношень можна обґрунтувати аналогічно.

Відношення R називається *транзитивним*, якщо $R^2 \le R$ (тобто, коли з тверджень $x\,R\,z$ та $z\,R\,y$ випливає $x\,R\,y$).

Транзитивними є відношення «більше або дорівнює», «менше», «бути старшим», «вчитися в одній групі», «жити в одному місті». Відношення «бути сином» не є транзитивним.

Умова $R^2 \le R$ дає зручний спосіб перевірки транзитивності відношення в разі, коли воно задано за допомогою матриці. Для цього необхідно обчислити матрицю відношення R^2 (тобто піднести до квадрата матрицю вихідного відношення) і перевірити умову. Якщо $a_{ij}(R^2) \le a_{ij}(R)$ для всіх значень i, j, то відношення транзитивне. Коли ж цю умову порушено хоча б для однієї пари індексів i, j, то відношення не буде транзитивним.

Відношення R називається *ациклічним*, якщо $R^k \bigcap R^{-1} = \varnothing$, тобто з умов $x\,R\,z_1,\ z_1\,R\,z_2,\ \ldots,\ z_{k-1}\,R\,y$ випливає, що $x \neq y$.

Це означає, що граф такого відношення не містить циклів.

Відношення R називається *від'ємно транзитивним*, коли його доповнення $\overline{R}$ транзитивне.

Відношення R називається *сильно транзитивним*, якщо воно одночасно транзитивне і від'ємно транзитивне.

Транзитивним замиканням відношення R називають відношення $\hat{R}$, яке отримують за таким правилом:

$$\hat{R} = R \cup R^2 \cup R \cup \ldots \cup R^n \cup \ldots$$

Вочевидь $a\ \hat{R}\ b$, якщо, у множині M існує ланцюжок із n елементів $a = a_1$, $a_2, \ldots, a_{n-1}, a_n = b$, у якому між сусідніми елементами виконується відношення R: $a_1\,R\,a_2,\ a_2\ R\,a_3, \ldots\ldots, a_{n-}1\,R\,b$. Якщо відношення R транзитивне, то $\hat{R} = R$. Дійсно, якщо $a\,R\,b$, то $a\ \hat{R}\ b$ (ланцюжок складається з двох елементів a та b), тому $R \subseteq \hat{R}$. Якщо ж $a\ \hat{R}\ b$, то існує ланцюжок $a_1\,R\,a_2,\ a_2\ R\,a_3, \ldots\ldots, a_{n-}1\,R\,b$. Але оскільки R транзитивне, то $a\,R\,b$, тому $\hat{R} \subseteq R$. Звідси випливає, що $R = \hat{R}$.

Транзитивним замиканням відношення "бути сином" є відношення "бути прямим нащадком", що є об'єднанням відношень "бути сином", "бути онуком", "бути правнуком" і т. д.

Стосовно транзитивного замикання має місце таке твердження:

Т е о р е м а 1.1. Транзитивне замикання будь-якого бінарного відношення R являє собою найменше транзитивне бінарне відношення, що містить у собі R.

Наведемо формулювання двох теорем, які дозволяють побудувати транзитивне замикання в деяких випадках.

Т е о р е м а 1.2. Якщо існує число k, для якого $R^k = R^{k+1}$, то

$$\hat{R} = R \cup R^2 \cup \ldots \cup R^k.$$

Т е о р е м а 1.3. Якщо R являє собою нечітке відношення на скінченній множині E, причому $m(E) = n$, то $\hat{R} = R \cup R^2 \cup R \cup \ldots \cup R^n$ або існує число: $k \leq n$, для якого $R^k = R^{k+1}$.

Властивості ациклічності й транзитивності відіграють особливу роль у теорії прийняття рішень, оскільки вони виражають природні взаємозв'язки між об'єктами. Дійсно, якщо об'єкт x у деякому сенсі не гірший за об'єкт y, а об'єкт y в тому самому сенсі не гірший за об'єкт z, то природно чекати, що об'єкт x буде не гіршим від об'єкта z (транзитивність), і в будь-якому разі об'єкт z не кращий за об'єкт x (ациклічність).

П р и к л а д 1.21. Визначити властивості такого відношення:

$$R = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Розв'язування

Дане відношення є рефлексивним, бо його матриця містить на головній діагоналі тільки одиниці. Воно не буде симетричним, позаяк серед симетричних елементів є такі, що не дорівнюють один одному, наприклад елементи a_{12} та a_{21}. Оскільки елемент $a_{13} = a_{31} = 1$, то відношення не буде також асиметричним й антисиметричним.

Для перевірки його транзитивності помножимо дане відношення само на себе, тобто

$$R^2 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

Оскільки $R^2 \not\subset R$, то вихідне відношення не є транзитивним.

1.8. Відношення еквівалентності та порядку

Еквівалентність. R є відношенням *еквівалентності* (*еквівалентністю*), якщо воно рефлексивне, симетричне й транзитивне. Позначимо його R_e , або символом $\sim$.

Наведемо приклади відношень, які є еквівалентністю.

Відношення рівності E на будь-якій множині є еквівалентністю. Рівність – це мінімальне відношення еквівалентності в тому сенсі, що при видаленні з нього довільної пари (тобто будь-якої одиниці на діагоналі матриці *E*) воно перестає бути рефлексивним, і, отже, вже не буде еквівалентністю.

Відношення конгруентності на множині трикутників. Розглянемо множину трикутників на площині, вважаючи трикутник заданим, якщо відомі координати його вершин. Два трикутники називаються рівними, якщо при накладенні вони збігаються, тобто можуть бути переведені один в інший шляхом деякого переміщення. Відношення рівності (конгруентності) є еквівалентністю на множині трикутників. Ним буде також *відношення подібності* на множині трикутників

Відношення «мати однаковий залишок при діленні на 7» є еквівалентністю на множині цілих додатних чисел *N*. Воно виконується, наприклад, для пар (11, 46), (14, 70) і не виконується для пар (12, 13), (14, 71).

Відношення «*вчитися на одному курсі*», «*учитися в одній групі*» задані на множині студентів факультету.

Характерним для еквівалентності є те, що вона розподіляє елементи на класи. У останньому прикладі – це курси або групи студентів факультету, у другому – множини конгруентних (або подібних) трикутників, у третьому – множини чисел, що мають однакову остачу при діленні на 7. Отже, задання еквівалентності на множині тісно пов'язане з її розбиттям на неперетинні підмножини. Розглянемо цю властивість еквівалентності докладніше.

Нехай задано деяке розбиття множини Ω, тобто відомо її підмножини Ω_1, Ω_2, … Ω_N, які задовольняють умову $\Omega = \bigcup_{i=1}^{N} \Omega_i$, причому $\Omega_i \bigcap \Omega_j = \varnothing$, коли $i \neq j, \quad i, j, = 1, 2, \ldots N$. Уведемо на множині Ω відношення R у такий спосіб: $x\,R\,y$ тоді і тільки тоді, коли існує множина Ω_i, що відповідає таким умовам: $x \in \Omega_i$ і $y \in \Omega_i$.

З а в д а н н я. Доведіть, що уведене таким чином відношення являє собою еквівалентність.

Як бачимо, задання еквівалентності на деякій множині Ω рівносильне розбиттю цієї множини на класи еквівалентних між собою елементів. І навпаки, будь-яке розбиття множини Ω визначає на ній відповідну йому еквівалентність.

Відношення порядку. Відношенням *нестрогого порядку* «$\leq$» (*нестрогим порядком*) називається відношення, що має властивості рефлексивності, антисиметричності й транзитивності.

Відношенням *строгого порядку* «$<$» (*строгим порядком*) називається відношення, яке має властивості антирефлексивності, асиметричності й транзитивності.

Якщо на множині Ω задано відношення «$\leq$», тобто деякий нестрогий порядок, то йому можна поставити у відповідність строгий порядок «$<$» , що визначається за таким правилом: $x < y$ тоді і тільки тоді, коли $x \leq y$ та $x \neq y$. І навпаки, якщо «$<$» – відношення строгого порядку, задане на множині Ω, то йому можна поставити у відповідність відношення «$\leq$» у такий спосіб: $x \leq y$ тоді і тільки тоді, коли $x < y$ або $x = y$. Отже, за нестрогим порядком ми можемо визначити відповідний йому строгий порядок і навпаки.

Припустимо, що на деякій множині задано відношення порядку (для всіх, або деяких пар її елементів), тоді кажуть, що на цій множині задано *частковий порядок.*

Частковий порядок на множині Ω називається *лінійним порядком*, якщо для будь яких елементів $x, y \in \Omega$ справедливе одне з трьох тверджень: $x < y$, $x = y$ або $x > y$ (тобто ми можемо порівняти будь-які два елементи множини Ω).

Множина M, на якому задане відношення порядку, називається, *цілком упорядкованою*, якщо будь-які два елементи M можна порівняти, *і частково упорядкованою* у протилежному випадку.

Наведемо приклади відношень порядку.

1. Відношення «$\le$» та «$\ge$» задані на множині чисел є нестрогоими порядками. Відношення «$<$» та «$>$» являють собою строгі порядки. Всі ці відношення цілком упорядковують множину N.

2. Розглянумо системи всіляких підмножин множини M. Відношення включення «$\subseteq$» задає на ній нестрогий частковий порядок, а відношення строгого включення «$\subset$» задає строгий порядок. Наприклад, $\{1, 2\} \subset \{1, 2, 3\}$, а множини $\{1, 2\}$ і $\{1, 3, 4\}$ не можна порівняти.

3. Відношення підпорядкованості на підприємстві задає строгий частковий порядок. У ньому непорівнянними будуть співробітники різних відділів.

4. Припустимо, у списку скінченного алфавіту A порядок літер зафіксовано, тобто він завжди той самий. Тоді такий список визначає повне упорядкування літер. Назвемо його *відношенням передування* й позначимо, що $a_i \prec a_j$ (якщо a_i передує a_j у списку літер). На ньому базується відношення передування слів. Воно задає повне впорядкування множини усіх скінченних слів в алфавіті A і називається *лексикографічним упорядкуванням* слів. Прикладом такого упорядкування є словник.

5. Якщо розглянути числа в позиційних системах числення (наприклад у десятковій), як слова в алфавіті цифр, то їх лексикографічне упорядкування збігається зі звичайним, якщо всі числа, що порівнюються, мають однакове число розрядів. Наприклад:

$$20 < 1073\ ;$$

$$20 \succ 1073;$$

$$0020 \prec 1073.$$

Отже, відношення упорядкування для множин залежить від додаткових факторів, які пов’язані з характеристиками елементів цих множин.

1.9. Відповідність, відображення і функції

Нехай X й Y – дві довільні множини. Говорять, що на множині X задано *відображення* f, яке набуває значення з множини Y, якщо кожному елементу $x \in X$ поставлено у відповідність один і тільки один елемент $y \in Y$. Коли X та Y – числові множини, відображення f називається *функцією*.

Для позначення функції (відображення) з множини X в множину Y користуються таким записом: $f: X \to Y$.

Якщо a – елемент множини X, то відповідний йому елемент $b = f(a)$ множини Y називається *образом* елемента a при відображенні f.

Сукупність усіх елементів $a \in X$, образом яких є даний елемент $b \in Y$, називається *повним прообразом* елемента b і позначається $f^{-1}(b)$, тобто

$$f^{-1}(b) = \{a \in X \mid f(a) = b\}.$$

Образ будь-якої множини A з множини X ($A \subset X$) при відображенні f позначають $f(A)$ і визначають у такий спосіб:

$$f(A) = \{b \in Y \mid \exists\, a \in A : b = f(a)\}.$$

Повний прообраз будь-якої множини B з множини Y ($B \subset Y$) визначають таким чином:

$$f^{-1}(B) = \{a \in X \mid f(a) \in B\}.$$

Дамо ще кілька визначень.

Відображення $f: X \to Y$ називається *взаємно однозначним*, коли з рівності $f(a_1) = f(a_2)$ випливає, що $a_1 = a_2$.

Відображення $f: X \to Y$ є відображенням множини A *на* множину B (відображення «*на*» , або *сюр'єкція*), якщо $f(A) = B$.

Відображення $f: X \to Y$ є відображенням множини A *в* множину B (відображення «*в*», або *ін'єкція*), якщо $f(A) \subset B$.

Відображення $f: X \to Y$ називається *бієкцією*, якщо воно взаємно однозначне і являє собою відображення «*на*».

Якщо f – бієкція, то існує відображення $g: Y \to X$, зумовлене співвідношенням: $g(f(a)) = a$, $a \in X$, яке називається *оберненим* до відображення f й позначається f^{-1}.

Відповідність. *Відповідністю* між множинами A та B називається підмножина $G \subseteq A \times B$. Відповідність G називається *функціональною* або *однозначною*, якщо образом будь-якого елемента множини A є єдиний елемент множини B, причому $A, B \subseteq G$.

Наприклад, англо-український словник встановлює відповідність між множиною англійських і українських слів. Вона не є функціональною.

Функція являє собою функціональну відповідність. Якщо функція f встановлює відповідність між множинами A й B, то кажуть, що функція f має тип $A \to B$ (позначення $f: A \to B$).

Якщо образ $f(A)$ множини A складається з єдиного елемента, то f називають функцією-константою.

Відображення типу $A \to A$ називається *перетворенням* множини A.

Важливим при побудові математичних моделей дискретної математики є поняття відношення між елементами двох множин. Якщо для кожного елемента $x \in X$ можна сказати, що елемент x знаходиться або не знаходиться у даному відношенні α з елементом $y \in Y$, то кажуть, що між елементами множин X і Y задано відношення α, і пишуть $x \alpha y$, у тому випадку, коли x знаходиться у відношенні α з елементом y. Наприклад, між множиною X студентів даного курсу і множиною Y усіх навчальних дисциплін за даною спеціальністю існує відношення α, де $x \alpha y$ означає «студент x вивчає дисципліну y». Зауважимо, що коли X та Y змістовні множини, між їх елементами може існувати багато різноманітних змістовних відношень.

Тепер ми можемо визначити поняття відображення, використовуючи термін відношення.

Відношення α між елементами множин X і Y називають *відображенням* множини X в множину Y, якщо кожний елемент $x \in X$ знаходитися у відношенні з деяким елементом $y \in Y$, причому y є єдиним для даного x.

Відображення множини X в множину Y позначають через φ (або іншою літерою) і пишуть $\varphi: X \to Y$, при цьому замість $x \varphi y$ для $x \in X$ й $y \in Y$ часто пишуть $y = \varphi(x)$ і кажуть, що y є образом x при відображенні φ. Відображення $\varphi: X \to Y$ зручно подавати у вигляді *двочасткового графа*.

Наприклад, на рис.1.12 показано відображення множини $X = \{x_1, x_2, x_3, x_4\}$ у множину $Y = \{y_1, y_2, y_3, y_4\}$.

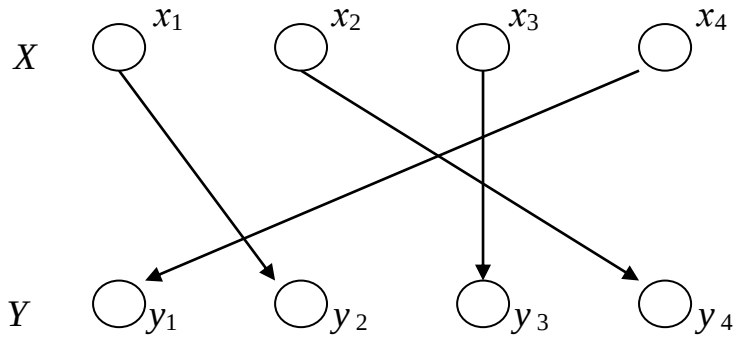


Рис.1.12. Двочастковий граф

Тут точки верхнього ряду зображують елементи множини X, нижнього ряду – множини Y, і, якщо має місце відповідність $y = \varphi(x)$, то з точки x виходить стрілка, яка входить в точку y. Таку форму подання називають *двочастковим графом відображення* $\varphi: X \to Y$, причому точки цього рисунка називають *вершинами графа*, а стрілки – *дугами графа* і кажуть, що відображення φ є :

– *ін'єктивним*, якщо в кожну вершину $y \in Y$ заходить або входить не більш однієї дуги;

– *сюр'єктивним*, якщо в кожну вершину $y \in Y$ входить не менше однієї дуги ;

– *бієктивним* або *бієкцією*, якщо φ ін'єктивне і сюр'єктивне одночасно.

Аби забезпечити строгість суджень, приймемо, що $\varphi(\varnothing) = \varnothing$. Якщо $\varphi(X) = Y$, то, вочевидь, відображення φ є сюр'єктивним .

У додатках часто використовується відображення $\varphi: X \to R^+$, при цьому $\varphi(x)$ виступає як певна характеристика (ціна, вага, довжина, або пропускна здатність та ін) елемента $x \in X$.

Очевидно, завдання бієкції $\varphi: X \to Y$ означає встановлення взаємно однозначної відповідності між елементами множин X та Y.

Відображення $\varphi: X \to Y$ і $\varphi_1: X \to Y$ називаються *рівними*, якщо з $x \varphi y$ випливає $x \varphi_1 y$ і навпаки, тобто якщо ці відображення можна зобразити одним і тим самим двочастковим графом відображення.

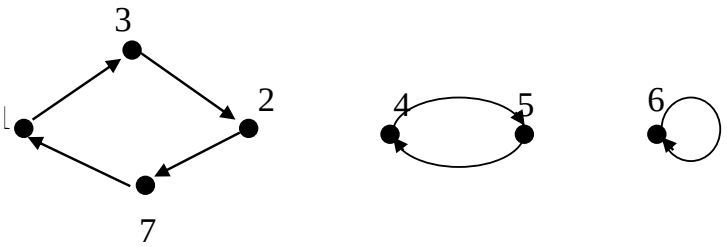
Нехай X та Y – скінченні непорожні множини, причому $|X| = m$, $|Y| = n$. Тоді кількість усіх можливих відображень $\varphi: X \to Y$ дорівнює n^m. Дійсно, щоб одержати довільне відображення $\varphi: X \to Y$, необхідно побудувати його двочастковий граф, для чого вибрати для кожної вершини $x \in X$ одну і тільки одну дугу серед n дуг ($n = |Y|$), які можуть виходити з вершини x і заходити у вершини $y \in Y$, причому для різних вершин $x, x_1 \in X$ зазначений вибір дуг робиться незалежно. Тому кількість усіх таких виборів, а також і всіх двочасткових графів відображень буде $n \cdot n \cdot \ldots \cdot n$, де кількість співмножників $m = |X|$. Отже, шукана кількість дорівнює n^m.

Множину усіх відображень $\varphi: X \to Y$ іноді позначають як Y^X, оскільки для скінченних X і Y тоді одержують формулу $| Y^X | = | Y^{|X|} |$. Але ніщо не заважає розглядати останню формулу для нескінченних X і Y і вважати її правилом зведення в кардинальний степінь $|X|$ кардинального числа $|Y|$.

Наприклад, $c = 2^a$. Далі, суму $|X| + |Y|$ довільних кардинальних чисел визначають як потужність $|X + Y|$ множини $X + Y$, створеної з усіх вершин $x \in X$ та $y \in Y$ двочасткового графа довільного відображення $\varphi: X \to Y$, а добуток $|X| \cdot |Y|$ визначають як потужність $|X \times Y|$ множини $X \times Y$, створеної з всіляких упорядкованих пар (x, y), де $x \in X$, $y \in Y$, тобто, із дуг двочасткового графа, в котрому з кожної вершини $x \in X$ виходять дуги в кожну вершину $y \in Y$. Таке визначення суми і добутку кардинальних чисел узгоджується зі звичним змістом $m + n$ і mn для скінченних множин X та Y, коли $m = |X|$, $n = |Y|$.

Перетворення. Відображення $\varphi: X \to X$ називають *перетворенням* множини X й на рисунку зображують у вигляді *орграфа* (орієнтованого графа) *перетворення*, вершинами якого є елементи $x \in X$, а дугами – пари (x_1, x_2), для яких $x_1 \varphi x_2$, тобто $x_2 = \varphi(x_1)$. Перетворення $\varphi: X \to X$ можна зображати дворядковим записом, вміщуючи в першому рядку цього запису елементи $x \in X$, а в другому – під ними їх подання $\varphi(x)$. Наприклад, задамо на множині $X = \{1, 2, 3, 4, 5, 6, 7\}$ одне з перетворень за допомогою дворядкового запису і орграфа (рис.1.13):

$$\varphi = \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 3 & 7 & 2 & 5 & 4 & 6 & 1 \\ \hline \end{array}$$



а) *б*)

Рис.1.13. Зображення перетворення: *а* – за допомогою дворядкового запису; *б* – орієнтованого графа

Очевидно, що це перетворення бієктивне, причому $\varphi(A) = A$ для множин $A=\varnothing$, $A=\{1,2,3,7\}$, $A=\{4,5\}$, $A=\{6\}$ і тільки для них.

Бієктивні перетворення називають *підстановками*. Очевидно орграф підстановки φ скінченної множини розпадається на частини, однозначно обумовлені підмножинами A, при яких $A = \varphi(A)$. Ці частини (вершини разом із дугами) прийнято називати *циклами* підстановки φ. Максимальна кількість циклів має *тотожня підстановка* φ, тобто, коли рівність $x = \varphi(x)$ виконується для довільного $x \in X$. Підстановки, які мають один цикл, називаються *одноцикловими*. Вони слугують, наприклад, моделями для опису обходу одною людиною n різноманітних пунктів (почавши рух у вихідному пункті, людина обходить всі інші по одному разу і повертається у вихідний). Зауважимо, що кількість усіх підстановок n-елементної множини X становить $n! = n(n-1)(n-2) \ldots 2 \cdot 1$, оскільки кожному елементу $x_1 \in X$ можна поставити у відповідність аби-який елемент із n елементів множини X, але елементу x_2 коли $x_2 \neq x_1$, можна поставити у відповідність згідно з цією підстановкою будь-який елемент із тих $(n-1)$, що залишилися і не є образом елемента x_1 і т.д.

У зв'язку зі сказаним вище зручно позначити через $X!$ множину всіх бієктивних перетворень множини X, бо тоді $|X!| = |X|!$ для скінченної множини X.

Запам'ятайте, що $10! = 3628800$, а $0! = 1$.

Зрозуміло, що кількість всіх одноциклових підстановок n-елементної множини дорівнює $(n-1)!$.

Приклад 1.22. Кожне натуральне число можна розкласти на добуток простих чисел. Якщо розташувати прості дільники у певному порядку наприклад, за «не зменьшенням», то одержимо таку функцию:

G(42)=(2, 3, 7);

G(23)=23;

G(100)=(2, 2, 5, 5) і т.д.

Приклад 1.23. Кожній людині відповідає множина його знайомих. G(Клочко) = (Коваленко, Петрівський, ... ,Семененко).

Методи задання функцій. Функції можна задавати за допомогою таблиці, графічно, аналітично.

Таблиця є найбільш простим способом задання функції. Вона подає функцію у вигляді пар $(x, f(x))$ (див. рис. 1.4) Проте в такий спосіб можуть задаватися тільки функції, визначені на кінцевій множині. Таблиці функцій, визначених на нескінченних множинах (наприклад, тригонометричних) задають їх тільки в кінцевому числі точок.

X	$f(x)$
x_1	$f(x_1)$
x_2	$f(x_2)$
x_m	$f(x_m)$

Рис. 1.14. Табличний спосіб задання функції

Іншим засобом завдання функцій є аналітичний, тобто формулою, яка описує функцію як суперпозицію інших (вихідних) функцій.

Для задання функції можуть використовуватися також графіки.

Ще два засоби завдання функцій:

а) за допомогою рекурсивної процедури;

б) $f(x) = \begin{cases} x, x \in M, \\ 0, x \notin M. \end{cases}$

Всі наведені положення використовуються в інших розділах дискретної математики.

Висновки

Множина являє собою деякий набір об’єктів, які не повторюються. Будь-яка множина має принаймні дві підмножини – порожню і саму себе. Вони називаються невласними. Важливою характеристикою множини є її потужність або кардинальне число. Для скінченної множини воно дорівнює кількості її елементів. Визначено такі операції над множинами: доповнення, перетин, об’єднання, добуток, розбиття.

Поняття відношення дозволяє формалізувати операції попарного порівняння об’єктів і математично обґрунтувати вибір одного або кількох об’єктів у тому разі, коли неможливо задати критерій на множині альтернатив.

Бінарні відношення можна задавати за допомогою матриці, графа або розрізів. До них застосовують операції перетину, об’єднання, доповнення та інші. Важливе значення для моделювання за допомогою відношень мають такі їхні властивості як рефлексивність, симетричність (асиметричність), транзитивність.

Задання на множині відношення дозволяє встановити на ній певне упорядкування.

Якщо X й Y – дві довільні множини і кожному елементу $x \in X$ поставлено у відповідність один і тільки один елемент $y \in Y$, то кажуть, що на множині X задано *відображення* f, яке набуває значення з множини Y, коли X та Y – числові множини, відображення f називається *функцією*.

Існують різні види відображень, особливе значення має взаємно однозначне відображення (бієкція), яка дозволяє встановити еквівалентність множин.

Питання, викладені в цьому розділі, розглянуто у літературі [1, 10, 11, 16, 19, 21].

Питання для самоконтролю

1. Дайте визначення поняття «множина».
2. Якими способами можна задати множину?
3. Скільки існує підмножин скінченної множини, складеної з n елементів?
4. Що являють собою діаграми Ейлера? З якою метою їх використовують?
5. Чому дорівнює потужність скінченної множини?
6. У якому випадку дві нескінченні множини будуть рівнопотужними?
7. Які множини називаються лічильними?
8. Назвіть основні операції над множинами.
9. Що являє собою різниця двох множин? Їх об'єднання? Перетин?
10. Дайте визначення доповнення множини.
11. Сформулюйте закони, яким підпорядковуються операції над множинами.
12. Які властивості характерні для кожної з операцій над множинами?
13. Чи виконується в загальному випадку комутативний закон для операції «різниця множин»?
14. Що називають відображенням?
15. Яке відображення називається функціональним?
16. Яким чином можна задати відображення множин?
17. Що називають образом (прообразом) елемента при відображенні f?
18. Яким чином визначають образ (прообраз) множини при відображенні?
19. Яке відображення називають ін'єктивним? Сюр'єктивним? Бієкцією?
20. Яке відображення називають взаємно однозначним?
21. Яке відображення називають функцією?
22. Яким чином можна задати функцію?
23. Яке відображення називають перетворенням?
24. Дайте визначення бінарного відношення.
25. Які існують способи задання відношень?
26. Яким чином можна задати відношення за допомогою матриці?
27. Як можна задати відношення у вигляді графа?
28. Яким чином задають відношення за допомогою розрізів?
29. Сформулюйте визначення верхнього (нижнього) розрізу відношення.
30. Які із способів задання відношень можна використовувати на нескінченній множині елементів?
31. Які математичні операції виконують над відношеннями?
32. Яке відношення називається рефлексивним (антирефлексивним)?

33. Яке відношення називається симетричним, антисиметричним, асиметричним?

34. Які відношення називають транзитивними, сильно транзитивними, від'ємно транзитивними?

35. Яким чином обчислюють транзитивне замикання відношення?

36. Які властивості характерні для відношення еквівалентності? Не строгого порядку? Строгого порядку?

37. Які множини називають лінійно впорядкованими? Частково впорядкованими?

Приклади розв'язування задач

1. Нехай задано множини $A=\{3,6,7,9\}$ та $B=\{1,6,8,9\}$. Знайти їх об'єднання, перетин і різниці $A-B$ й $B-A$.

Розв'язування

Об'єднання множин включає елементи, які належать принаймні одній із множин, тому $A\cup B=\{1,3,6,7,8,9\}$.

Перетин включає лише спільні елементи цих множин, отже, $A\cap B=\{6,9\}$.

Тепер знайдемо різницю $A-B$. Вона включає елементи множини *A*, які не належать множині *B*, тобто $A-B=\{3,7\}$.

Різниця $B-A$ включає елементи множини *B*, які не належать множині *A*, отже, $B-A=\{1,8\}$.

2. Нехай $A=N;\ B=Z;\ (A\subset B)$. Знайти їхнє об'єднання, перетин та різниці $A-B$ і $B-A$.

Розв'язування

Оскільки $A\subset B$, то $A\cup B=B$, $A\cap B=A$ і $A-B=\varnothing$.

За визначенням $B-A=\{0,-1,-2,-3...\}$.

3. Задано множини: $A=\{x\,|\,x\in R;\ 2<x\le 5\}$; $B=\{x\,|\,x\in R;\ 0\le x<3\}$. Знайти їхнє об'єднання, перетин і різниці $A-B$ та $B-A$.

Розв'язування

$A\cup B=\{x\,|\,x\in R;0\le x\le 5\}$,

$A\cap B=\{x\,|\,x\in R;2<x<3\}$,

$A-B=\{x\,|\,x\in R;3\le x\le 5\}$,

$B-A=\{x\,|\,x\in R;0\le x\le 2\}$.

4. Знайти множини *A* й *B*, якщо їх різниці такі: $A-B=\{a\}$, $B-A=\{b,e\}$, а $A\cup B=\{a,b,c,d,e\}$.

Розв'язування

Елементи множин A й B знайдемо за допомогою діаграм Ейлера (див. рис.1.15). Елемент a міститься тільки в множині A, елементи b,e – лише в множині B, тоді (враховуючи останню умову) елементи c,d належать перетину множин A та B.

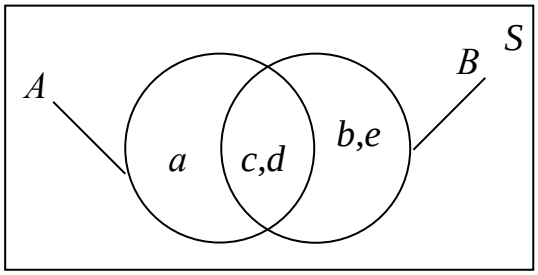


Рис. 1.15. Графічна інтерпретація задачі 4

Отже, $A=\{a,c,d\}; B=\{b,c,d,e\}.$

5. Виразити за допомогою множин A, B й C заштриховану ділянку D на діаграмі (рис. 1.16).

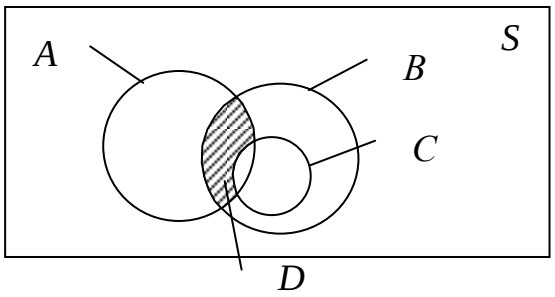


Рис.1.16. Графічне подання умови задачі 5

Розв'язування

Згідно з діаграмою Ейлера заштрихована ділянка відповідає такому виразу: $D=(A\cap B)-C=(B-C)\cap A=(A-C)\cap B$.

6. Заштрихувати на діаграмі Ейлера ділянку, відповідну такій множині:

$$D=(B\cap C)-(A\cap C).$$

Розв'язування

Спочатку зобразимо на діаграмі множини B, C та їхній перетин (рис.1.17, *а*). Потім перетин множин A й C (рис.1.17, *б*) і, нарешті, розв'язок задачі подано на рис. 1.17, *в*.

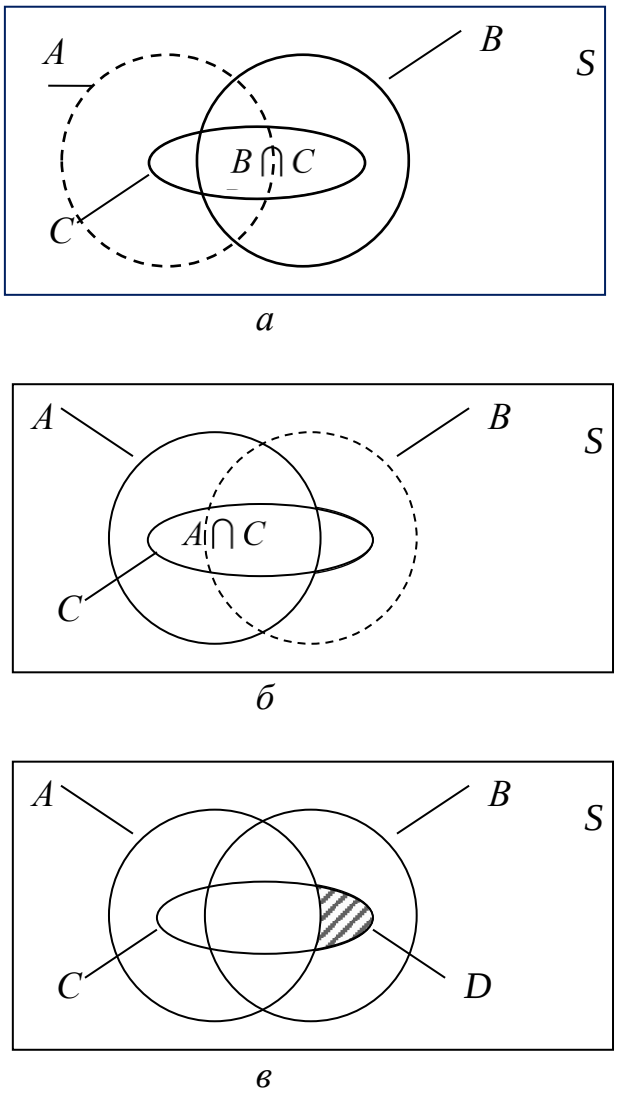


Рис.1.17. Графічне подання етапів розв’язання задачі 7

7. Уведемо такі позначення: A – множина працівників заводу; B – множина працівників-чоловіків того самого заводу. Що означають множини $A-B$ і $(A\cap B)-B$?

Розв’язування

$A-B$ являє собою множину працівників заводу, не враховуючи чоловіків, тобто це множина працівників-жінок цього заводу. Оскільки множина $A\cap B=B$ (враховуючи, що $B\subset A$), то $(A\cap B)-B$ являє собою порожню множину.

8. Задано множини: $A=\{x\,|\,x\in R;-5<x<5\}$, $B=Z$,$C=N$. Знайти такі множини: $D=A\cap(B-C)$, $E=A\cap(C-B)$, $F=(C-A)\cap B$, $G=A\cap B\cap C$.

Розв’язування

Множина $B-C$ являє собою множину цілих від’ємних чисел, тому множина D включає цілі від’ємні числа, які належать інтервалу (–5; 5), тобто $D=\{-4,-3,-2,-1,0\}$.

Оскільки $C \subset B$, то $C - B = \varnothing$, і, відповідно $E = \varnothing$.

Множина $C - A$ складається з натуральних чисел, більших п'яти, і, враховуючи, що $C \subset B$, маємо такий результат: $F = \{5, 6, 7, 8...\}$.

Аналогічно: $G = \{1, 2, 3, 4\}$.

1. Знайти множини A і B, якщо: *а*) $A \cap B = \{3,5\}$; $B \subset A$; $A - B = \{7\}$; *б*) $A \cup B = \{2,5,7,8,10\}$; $A - B = \varnothing$; $B - A = \{7,10\}$.

Розв'язування

а) Оскільки $B \subset A$, то $A \cap B = B$, тому $B = \{3,5\}$. З огляду на це, множина A включає всі елементи множини B й елементи множини $A - B$, таким чином, $A = \{3, 5, 7\}$.

б) Враховуючи, що $A - B = \varnothing$, можемо зробити висновок: $A \subset B$, тоді $A \cup B = B$ й $B = \{2, 5, 7, 8, 10\}$. Множина $B - A$ містить елементи B, які не входять в A, тому $A = \{2, 5, 8\}$.

10. Задано множини: $A = \{1, 3, 5, 7, 9\}$, $B = \{1, 2, 5, 7\}$, $C = \{4, 7\}$. Виразити через A, B, C множини: $D = \{1, 5\}$; $E = \{2, 4\}$.

Розв'язання

$D = (A - C) \cap B,$

$E = (B \cup C) - A.$

Задачі для самостійного розв'язування

1. Визначте множини $A \cap Y$, $A \cup Y$, $A \setminus Y$, $Y \setminus A$ для поданих нижче множин A та Y.

а) $A = \{3,5,6,7,9\}$, $Y = \{4,6,7,8\}$;

б) $A = \{3,5,6,7,9\}$, $Y = \{2,4,8\}$;

в) $A = \{2,3,5,6\}$, $Y = \{3,4,5\}$;

г) $A = Z$, $Y = N$;

д) $A = R$, $Y = Q$;

е) $A = Z$, $Y = Q$;

ж) $A = \{x \in R \mid -1 \le x \le 5\}$, $Y = \{x \in R \mid -3 \le x \le 2\}$;

і) $A = \{x \in R \mid -1 \le x \le 3\}$, $Y = \{x \in R \mid 3 \le x \le 5\}$;

и) $A = \{x \in R \mid -1 \le x \le 5\}$, $Y = \{x \in R \mid -3 \le x \le 2\}$;

к) $A = \{x \in R \mid -1 \le x \le 3\}$, $Y = \{x \in R \mid 3 \le x \le 5\}$;

л) A – множина всіх прямокутників, Y – множина всіх ромбів.

2. Знайдіть множини A і Y якщо:

а) $A \setminus Y = \{a,b\}$, $Y \setminus A = \{c,d\}$, $A \cap Y = \{x,y,z\}$;

б) $A \cup Y = \{a,b,c,d,e,f\}$, $A \cap Y = \{c,d\}$, $A \setminus Y = \{a,e,f\}$;

в) $A \cup Y = \{a,b,c,d\}$, $A \setminus Y = \{a\}$, $A \cap Y = \varnothing$.

3. Нехай A – множина розв'язків рівняння $f(x) = 0$, Y – множина розв'язків рівняння $g(x) = 0$. Опишіть, використовуючи множини A й B, множину розв'язків таких рівнянь:

а) $f(x)g(x) = 0$;

б) $f(x) + g(x) = 0$;

в) системи рівнянь: $\begin{cases} f(x) = 0, \\ g(x) = 0; \end{cases}$

4. Опишіть множину дійсних коренів рівняння $f(x) = 0$, використовуючи множини $A = \{x \in R \mid f(x) \geq 0\}$ і $Y = \{x \in R \mid f(x) \leq 0\}$.

5. Визначте такі множини: $A \cup \varnothing$, $A \cap \varnothing$, $A \cup A$, $A \cap A$, $A \setminus A$, $A \setminus \varnothing$, $\varnothing \setminus A$.

6. Дано множини A й B, причому $A \subset B$. Знайдіть такі множини: *а*) $A \cup B$; *б*) $A \cap B$; *в*) $A \setminus B$.

7. Задано множини: $A = \{0,3,5,8\}$ і $B = \{0,4,5,9\}$. Знайти їх об'єднання, перетин і різниці.

8. Дано множини: $A = N$, $B = \{x \mid x - \textit{натуральне парне}\}$. Знайти для цих множин об'єднання, перетин і різниці $A - B$ та $B - A$.

9. Маючи множини $A = \{x \mid x \in R; -1 \leq x < 4\}$, $B = \{x \mid x \in R; 0 < x \leq 6\}$, визначити їх об'єднання, перетин і різниці $A - B$ та $B - A$.

10. Знайти множини A і B, якщо:

$$A - B = \{a,b\};$$
$$B - A = \{d\};$$
$$A \cap B = \{c\}.$$

11. Дано множини: $A = Z$, $B = \{x \mid x \in R; -2 \leq x < 2\}$, $C = \{x \mid x \in R; 0 < x \leq 5\}$. Визначити множини:

$$D = A \cap B \cap C,$$
$$E = (A \cap C) - B,$$
$$F = A - (C \cup B),$$
$$G = (A \cap B) - C.$$

12. Знайти множини A та B, якщо:

а) $A \cap B = \{2,3\}; A \subset B; B - A = \{6\};$

б) $A \cup B = \{1,4,5,7,9\}; B - A = \varnothing; A - B = \{4,7,9\}.$

13. Дано множини: $A = \{2, 3, 5, 6, 8\}$, $B = \{1, 3, 5, 7\}$, $C = \{4, 6\}$. Виразити через них такі множини: $D = \{2, 8\}$, $E = \{1, 6\}$.

14. На діаграмі Ейлера – Венна зображено множини A, B, C (рис. 1.18). Зазначте (заштрихуйте) на цій діаграмі такі множини:

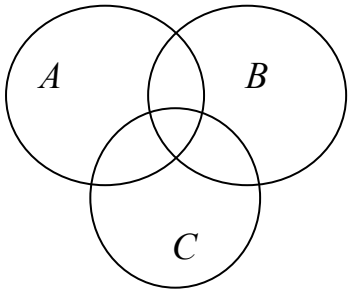


Рис. 1.18. Діаграма Ейлера – Венна до завдання 14

а) $A \cap (B \cup C)$;
б) $A \cap (B \cap C)$;
в) $(A \setminus B) \cap C$;
г) $(A \setminus B) \cup (B \setminus C)$.

15. Виразити через множини A, B і C заштриховану ділянку D на діаграмі Ейлера (рис. 1.19).

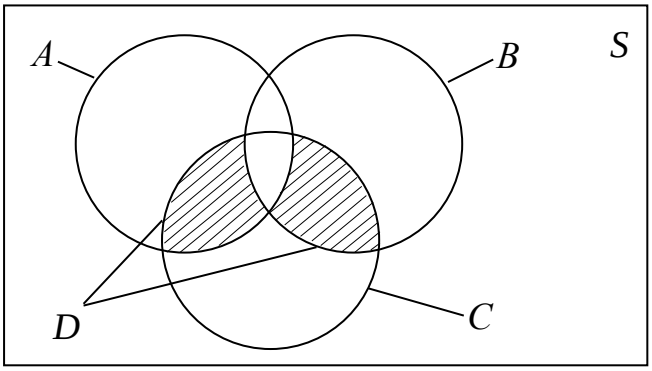


Рис. 1.19. Графічне подання умови задачі 15

16. На діаграмі Ейлера (рис.1.20) заштрихувати ділянку, відповідну такій множині: $D = (A \cap C) - B$.

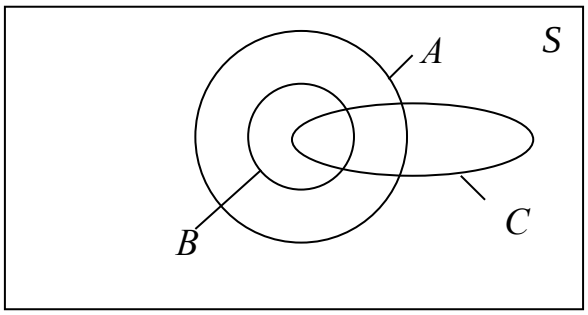


Рис.1.20. Графічне подання умови задачі 16

17. Доведіть,що для будь-яких множин A, B, C будуть правильними такі твердження:

а) $A\cap(B\cup C)=(A\cap B)\cup(A\cap C)$; *б*) $A\cup(B\cap C)=(A\cup B)\cap(A\cup C)$;

в) $A\setminus(B\cap C)=(A\setminus B)\cup(A\setminus C)$.

18. Доведіть такі співвідношення:

а) $X\subset Y\Leftrightarrow X\cup Y=Y\Leftrightarrow X\cap Y=X$;

б) $X\subset Z$ й $Y\subset Z\Leftrightarrow X\cup Y\subset Z$;

в) $Z\subset X$ й $Z\subset Y\Leftrightarrow Z\subset X\cap Y$;

г) $X\cup(Y\cap Z)=(X\cup Y)\cap(X\cup Z)$;

д) $X\cap(Y\cup Z)=(X\cap Y)\cup(X\cap Z)$;

е) $X\setminus Y=X\setminus(X\cap Y)=(X\cup Y)\setminus Y$;

ж) $X\setminus(Y\setminus Z)=(X\setminus Y)\cup(X\cap Z)$;

и) $(X\setminus Y)\cap(Z\setminus U)=(X\cap Z)\setminus(Y\cup U)$;

к) $X\cap(Y\setminus Z)=(X\cap Y)\setminus(X\cap Z)$;

л) $(X\setminus Z)\cap(Y\setminus Z)=(X\cap Y)\setminus Z$;

м) $(X\cup Y)\setminus Z=(X\setminus Z)\cup(Y\setminus Z)$;

н) $(X\setminus Y)\setminus Z=X\setminus(Y\cup Z)$.

19. Доведіть такі співвідношення:

а) $X\times Y=\varnothing\Leftrightarrow X=\varnothing$ або $Y=\varnothing$;

б) $X_1\times Y_1\subset X\times Y\Leftrightarrow X_1\subset X, Y_1\subset Y$;

в) $X\times Y=A\times B\Leftrightarrow X=A, Y=B$;

г) $(X\times Y)\cap(X_1\times Y)=(X\cap X_1)\times Y$;

д) $X\times(Y\cup Z)=(X\times Y)\cup(X\times Z)$;

е) $(X\cap X_1)\times(Y\cap Y_1)=(X\times Y)\cup(X_1\times Y_1)$.

20. Нехай A та B – підмножини множини M. Доведіть , що:

а) $\overline{A}\cup\overline{B}=A\cap B$; *б*) $\overline{\overline{A}\cap\overline{B}}=A\cup B$.

21. Множина A складається з n элементов, а множина B – з m елементів ($m > n$). Яке найбільше і найменше число елементів можуть містити такі множини: $A\cup B$, $A\cap B$, $A\setminus B$, $B\setminus A$?

22. Доведіть, що коли множина A складається з n елементів, множина B – із m елементів, а $A\cap B$ з k елементів, то множина $A\cup B$ складається з $n + m – k$ елементів.

23. Кожний учень у класі вивчає англійську або французку мову. Англійської навчаються 25 учнів, французької – 27, а обидво –18. Скільки учнів у класі?

24. У класі 30 учнів. Кожен з них займається футболом або хокеєм. Половина учнів класу грає лише у хокей, а п’ятеро – і в хокей і у футбол. Скільки учнів у класі займається футболом ?

25. Нехай A – множина студентів групи; B – множина студентів-відмінників тієї самої групи. Що означають такі твердження:

$$A - B = A,$$
$$A - B = \varnothing,$$
$$A \cap B = A?$$

РОЗДІЛ 2

ОСНОВИ ТЕОРІЇ ГРАФІВ

Мета розділу: вивчення основні понять теорії графів та набуття навичків їх застосування до розв'язування оптимізаційних задач.

Багато структур, що є цікавими із точки зору математики, інформатики та їх практичного застосування, можуть бути описані за допомогою *графів.*

У загальному значенні *графом* називають непорожню множину вершин (вузлів), з'єднаних ребрами. Для різних галузей застосування графи можуть розрізнятися напрямами, обмеженнями на кількість зв'язків, додатковими відомостями про їх вершини та ребра.

У строгому математичному визначенні *граф* являє собою пару множин: $G = (V, E)$, де V – підмножина будь-якої лічильної множини (вершини), а E – підмножина декартового добутку $V \times V$ (ребра або зв'язки).

Теорія графів є розділом дискретної математики, який вивчає властивості графів. Її почали розробляти для розв'язування деяких задач про геометричні конфігурації, що складаються з точок і ліній. Потім вона набула широкого застосування в різних галузях науки та виробництва. Наприклад, у геоінформаційних системах (ГІС). Тут наявні або проєктовані будинки, споруди, квартали і т. п. розглядаються як вершини графа (множина V), а дороги, інженерні мережі, лінії електропередачі та ін., що їх з'єднують, як ребра (множина E). Обчислення, виконані на такому графі, дають можливість, приміром, знайти найкоротший об'їзний шлях або найближчий продуктовий магазин, спланувати оптимальний транспортний маршрут. Коли за вершини графа взяти роботи, які мають бути виконані, а ребра відповідають порядку їх виконання, то, використовуючи методи теорії графів, можна, зокрема, визначити оптимальний порядок проведення робіт.

У теорії графів існує велика кількість невирішених проблем і поки що не доведених гіпотез. Тож розглянемо основні її поняття.

2.1. Основні поняття теорії графів

Нехай дано множину X, яка включає елементи, називані *вершинами графа*, а також закон (правило) G, що встановлює відповідність між кожним елементом x множини X та її підмножинами. Тоді граф буде повністю визначено множиною X і відповідністю G.

Приклад 2.1. Припустимо, що $X = \{x_1, x_2, x_3, x_4, x_5\}$, $G(x_1) = \{x_2, x_3\}$, $G(x_2) = \{x_1, x_3, x_5\}$, $G(x_4) = \{x_3, x_5\}$, $G(x_3) = \{x_1, x_2, x_4\}$, $G(x_5) = \{x_2, x_4\}$.

Цей запис означає, що граф містить п’ять вершин (множина X). Відповідність G задає множину ребер графа. Зокрема, запис $G(x_1)=\{x_2, x_3\}$ означає, що є ребра, які з’єднують вершину x_1 з вершинами x_2 та x_3.

Наочне подання графа можна отримати, якщо співставити множині X певні точки площини (вершини), а множині ребер – множину відрізків U, що з’єднують усі або деякі з вершин. Тоді граф G можна визначити як пару множин X та U, а саме $G = (X,U)$. Тому часто граф зображують геометрично. Відповідне наведеному вище прикладу зображення графа показано на рис. 2.1.

У геометричному поданні графа ребра – це лінії, що з’єднують вершини. Для їх позначення іноді використовують запис: $g(x_i, x_j)$, тоді говорять, що ребро g *інцидентне* вершинам x_i і x_j, а вершини x_i і x_j називають *суміжними*. У зображеному на рис. 2.1 графі суміжними будуть, наприклад, вершини x_2 і x_1, x_2 і x_3, x_2 і x_5. У той же час вершини x_2 та x_4 не будуть суміжними.

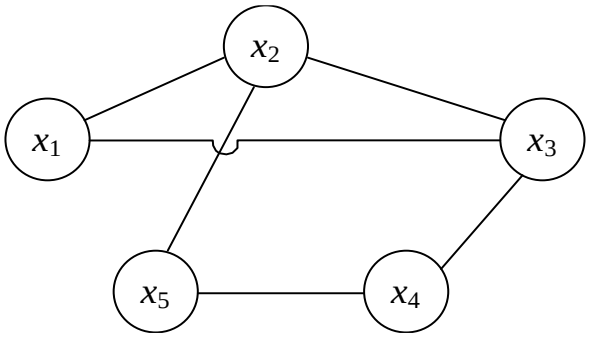
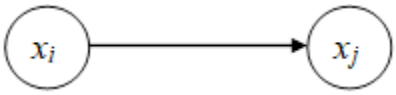


Рис. 2.1. Приклад геометричного зображення графа

Якщо порядок проходження вершин у записі $g(x_i, x_j)$ задано, то таке ребро називають *орієнтованим* або *дугою*, геометрично такий порядок позначають стрілкою.



Тут вершина x_i є початком дуги, а вершина x_j – її кінцем. Кажуть також, що дуга *виходить* з вершини x_i та *входить* у вершину x_j.

П р и к л а д 2.2. На рис. 2.2 зображений граф, вершинами якого є точки a, b, c, d, e, g, h, а дугами – відрізки (a, a), (c, b), (c, d), (c, e), (d, c), (d, d), (e, d), (g, h). У термінах відображень цей граф можна описати у такий спосіб:

$G(a) =\{a\}$; $G(b) = \varnothing$; $G(c) = \{b, d, e\}$; $G(d) = \{c, d\}$; $G(e) = \{d\}$; $G(g) = \{h\}$; $G(h) = \varnothing$.

Неважко помітити, що обидва способи описують один і той самий граф.

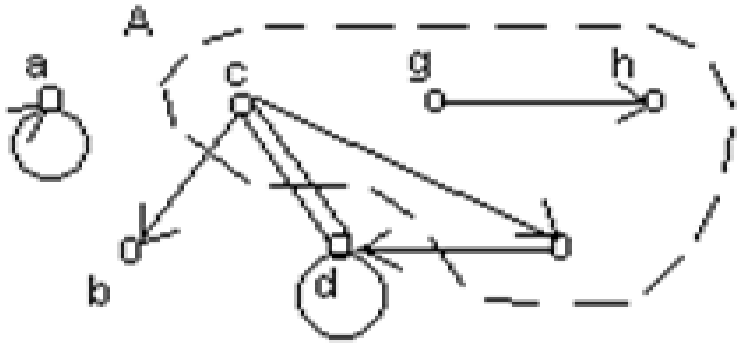


Рис. 2.2. Зображення графа до прикладу 2.2

Отже, розрізняють *неорієнтовані*, тобто ті, що складаються із ребер, і *орієнтовані*, котрі складаються із дуг, *графи.*

Прикладами графів є відношення батьківства і материнства на множині людей (дерево родоводу), карта доріг на місцевості, схема з'єднань електричних приладів, відношення переваги одних учасників турніру над іншими і т.і.

Приклади орієнтованих графів показано на рис. 2.3.

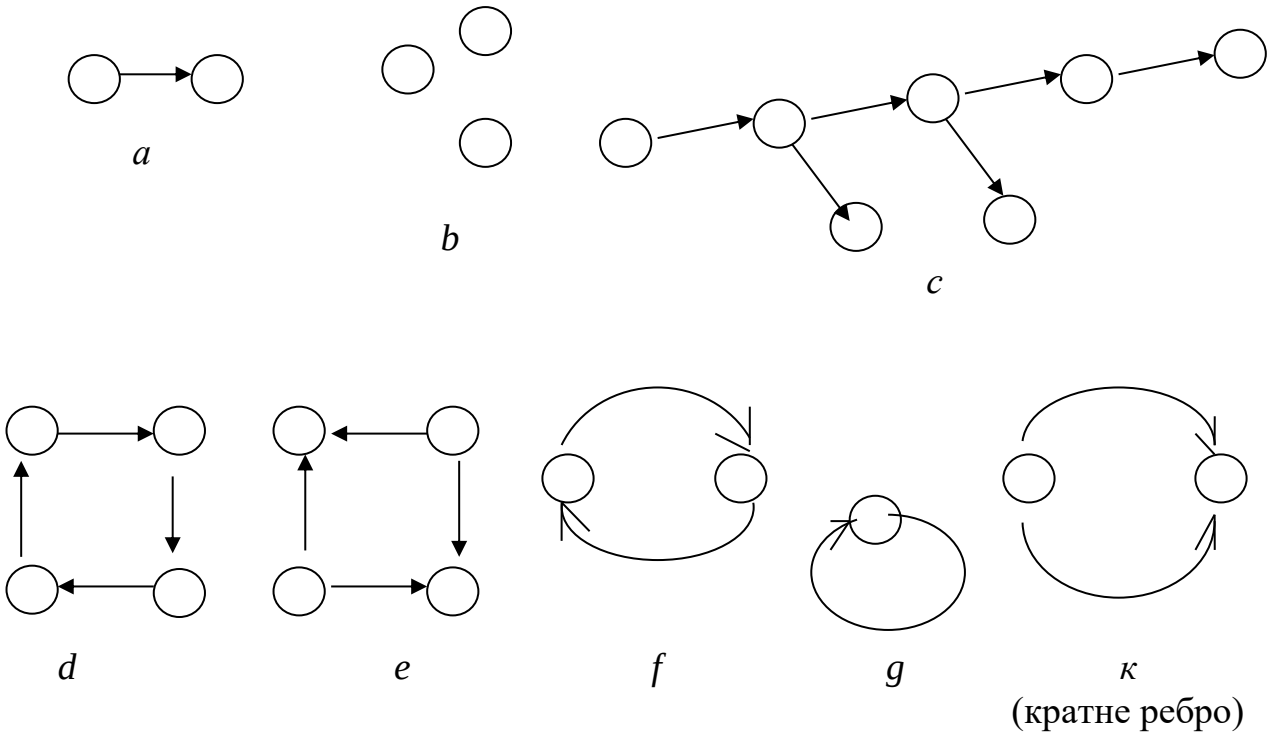


Рис. 2.3. Приклади орієнтованих графів

В наведених прикладах варіант "*b*" подає граф із порожньою множиною ребер. Граф "*e*" ілюструє недосяжність двох вершин, а "*g*" – граф із петлею.

Зробимо ще кілька визначень.

Повним називається граф, у якого будь-яка вершина з'єднується (ребрами або дугами) з усіма іншими вершинами. Так, граф, зображений на рис. 2.1, не повний.

Підграфом G_A графа $G = (X, \Gamma)$ називається граф, що включає лише ті з вершин графа G, які утворюють множину A, а також дуги, що з'єднують ці вершини. Наприклад, підграфом є окреслена пунктиром область A на рис. 2.2.

У математичному записі підграф позначається у такий спосіб:

$$G_A = (A, \Gamma_A), \text{ де } A \subseteq X, \Gamma_{AX} = (\Gamma_X) \cap A. \qquad (2.1)$$

Частковим графом G_Δ стосовно графа $G = (X, \Gamma)$, є такий, що містить тільки частину дуг графа G, тобто визначений умовою:

$$G_\Delta = (X, \Delta), \text{ де } \Delta x \subseteq \Gamma_X \qquad (2.2)$$

Приміром, нехай $G = (X, \Gamma)$ – карта шосейних доріг України. Тоді мапа шосе Дніпропетровської області являє собою підграф, а мапа головних доріг країни – це частковий граф.

Дугу, що з'єднує вершини a і b, і спрямовану від a до b часто позначають у такий спосіб: $u = (a, b)$.

Петлею називається ребро (дуга) $g(x_i, x_j)$, у якого початкова й кінцева вершини збігаються.

Ланцюг – це послідовність ребер, які проходять через кілька вершин. Наприклад, послідовність ребер $g(x_1, x_2)$, $g(x_2, x_3)$, $g(x_3, x_4)$ на рис. 2.1 являє собою ланцюг.

Ланцюг, у якого початкова й кінцеві вершини збігаються, будемо називати *циклом*. Ребра $g(x_1, x_2)$, $g(x_2, x_3)$, $g(x_3, x_1)$ на рис. 2.1 утворюють цикл.

Для орієнтованого графа мають місце аналогічні поняття «*шлях*» (він є послідовністю дуг, які проходять через кілька вершин) і «*контур*» (шлях, у якого початкова й кінцеві вершини збігаються).

Шляхом у графі G називають таку послідовність дуг $\mu = (u_1, \ldots, u_k)$, у якій кінець кожної попередньої дуги збігається із початком наступної. Шлях μ, послідовними вершинами котрого є вершини $a, b, \ldots, m$, позначається таким чином: $\mu = (a, b, \ldots, m)$.

Довжиною шляху $\mu = (u_1, \ldots, u_k)$ називають число $l(\mu) = k$, яке дорівнює числу дуг, що складають шлях.

Іноді кожній дузі u_i приписують деяке число $l(u_i)$, яке називається *довжиною дуги*. Тоді довжина шляху визначається як сума довжин дуг, що складають шлях, а саме:

$$l(\mu) = \sum_{i=1}^{k} l(u_i), \qquad (2.3)$$

Шлях, у котрому жодна дуга не трапляється двічі, називається *простим*.

Шлях, де жодна вершина не зустрічається двічі, називається *елементарним*.

Контур – це кінцевий шлях $\mu = (x_1, \ldots, x_k)$, у якого початкова вершина x_1 обов'язково збігається із кінцевою x_k.

При цьому контур називається *елементарним*, якщо усі його вершини різні (за винятком початкової і кінцевої, які співпадають).

Контур одиничної довжини, утворений дугою вигляду (*a*, *a*), називається *петлею*. Так, наприклад, на рис. 2.2 (*e*, *d*, *c*, *b*) – шлях, (*c*, *e*, *d*, *c*) – контур, а (*d*, *d*) – петля.

Один і той самий граф можна зобразити по-різному. Так, на рис. 2.4 подано три варіанти зображення одного графа. Ця властивість графів називається *ізоморфізмом*.

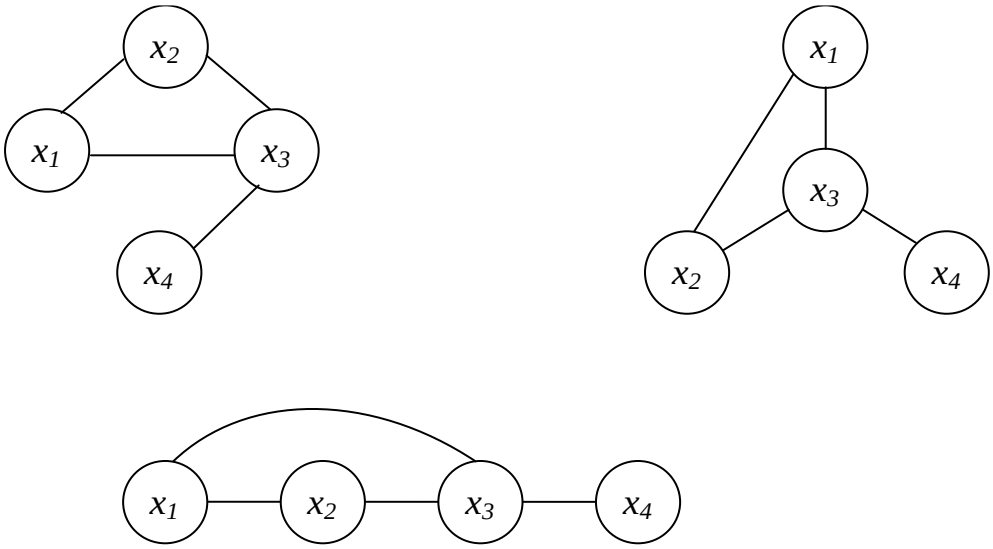


Рис. 2.4. Приклади ізоморфних графів

Взагалі, кожен граф можна задати трьома способами: графічно, аналітично й за допомогою матриць. Розглянемо їх докладніше.

1. *Графічний спосіб*. Він полягає в тому, що вершини графа зображуються точками або колами, його ребра (або дуги) – лініями (стрілками). Приклад такого задання графа наведено на рис. 2.5. Тут символами x_i позначено вершини графа, g_j – ребра або дуги. На рис. 2.4, *а* зображено неорієнтований, а на рис.2.3, *б* – орієнтований графи.

а *б*

Рис. 2.5. Графічний спосіб задання графа: *а* – неорієнтований, *б* – орієнтований графи

2. *Аналітичний спосіб*. Включає опис множини X та відповідності G. Два графи із попереднього прикладу можна задати аналітично у такий спосіб:

а) неорієнтований граф: $X=\{x_1,x_2,x_3,x_4\}$, $G(x_1)=\{x_2,x_3\}$, $G(x_2)=\{x_1,x_3\}$, $G(x_3)=\{x_1,x_2,x_4\}$, $G(x_4)=\{x_3\}$;

б) орієнтований граф: $X=\{x_1x_2x_3x_4\}$, $G(x_1)=\{x_2x_4\}$, $G(x_2)=\{x_4\}$, $G(x_3)=\{x_2\}$, $G(x_4)=\{x_3\}$.

3. *Матричний спосіб*. Граф може бути заданий також у зручному для машинної обробки вигляді за допомогою матриці суміжності або матриці інциденцій.

Матрицею суміжності графа є квадратна матриця A, розмір якої дорівнює числу його вершин, а кожний елемент a_{ij} визначається у такий спосіб: він дорівнює одиниці, якщо з вершини x_i можна безпосередньо потрапити у вершину x_j (тобто існує ребро, або дуга, що з'єднує вершину x_i і вершину x_j), і нулю у протилежному випадку. Для графів, показаних на рис. 2.4, матриці суміжності мають такий вигляд:

$$A_{нор}=\begin{pmatrix}0&1&1&0\\1&0&1&0\\1&1&0&1\\0&0&1&0\end{pmatrix}$$ – для неорієнтованого графа;

$$A_{ор}=\begin{pmatrix}0&1&0&1\\0&0&0&1\\0&1&0&0\\0&0&1&0\end{pmatrix}$$ – для орієнтованого.

Зауважимо, що матриця суміжності неорієнтованого графа завжди симетрична.

Позначимо ребра (дуги) графа через $g_1, g_2, \dots g_m$, а вершини через $x_1, x_2, \dots x_n$.

Матрицею інциденцій ребер графа є матриця $R = \|r_{ij}\|_{n\times m}$, кожний елемент якої визначено в такий спосіб:

$$r_{ij} = \begin{cases} 1, \textit{ коли вершина } x_i \textit{ інцидентна ребру } g_j, \\ 0, \textit{ якщо ні.} \end{cases}$$

Матрицею інциденцій дуг графа є матриця: $S = \|s_{ij}\|_{n\times m}$, кожний елемент якої визначається таким чином:

$$s_{ij} = \begin{cases} 1, \textit{ коли ребро } g_j \textit{ виходить з вершини } x_i, \\ -1, \textit{ якщо ребро } g_j \textit{ заходить у вершину } x_i, \\ 0, \textit{ коли ребро } g_j \textit{ не інцидентне вершині } x_i. \end{cases}$$

У нашому прикладі (див. рис. 2.5)

$$R = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \qquad S = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & -1 \end{pmatrix}.$$

Зауважимо, що матриця інцидентності в описаному вигляді застосовуються тільки до графів без петель. Коли в графі є петлі цю матрицю варто розділити на дві півматриці: позитивну і негативну.

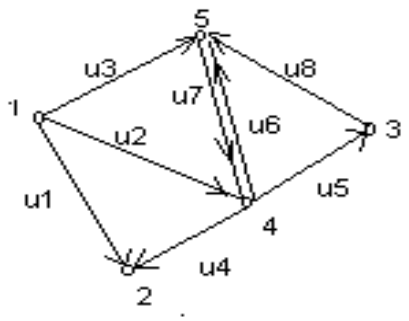


Рис. 2.6.

Приклад 2.3. Розглянемо граф без петель, показаний на рис. 2.6. Відповідні йому матриці суміжності і інцидентності мають такий вигляд:

Матриця суміжності: $A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

Матриця інцидентності: $S = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 1 & -1 \end{pmatrix}$.

При розв'язуванні практичних задач ребрам або дугам графа ставиться у відповідність певне число, яке в кожному конкретному випадку відображає ту чи іншу характеристику, наприклад, відстань, витрати, пропускну здатність, часовий інтервал між подіями і т. і.

Граф, кожній дузі якого відповідає певне число, називається *зваженим*.

Зазвичай, графи, які розглядаються, є кінцевими, тобто кінцевими є множини їхніх елементів (вершин і ребер). Тому скінченність цих графів не буде додатково визначатися.

2.2. Неорієнтовані графи

Тепер розглянемо докладніше неорієнтовані графи.

Наведемо декілька визначень, які використовуються для характеристики неорієнтованих графів.

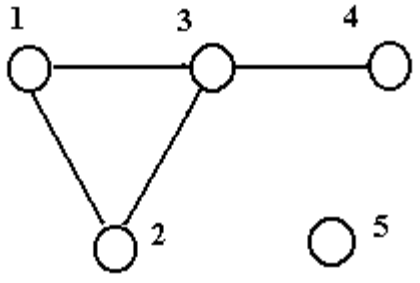


Рис. 2.7.

Степенем вершини x, що позначається $deg(x)$ або dx, називають число ребер, інцидентних вершині x. Так, для графа на рис. 2.7 маємо d1=2, d2=2, d3=3, d4=1, d5=0.3 маємо d1=2, d2=2, d3=3, d4=1, d5=0. Якщо dx=1, то вершину називають *тупиковою*; якщо dx = 0, то *ізольованою*.

Теорема. Нехай G – неорієнтований граф із n вершинами і m ребрами і d_j – степінь j-ої вершини. Тоді $\Sigma n_j = 1$ і $d_j = 2m$.

Доведення цієї теореми випливає з того факту, що кожне ребро додає одиницю до степеня кожної з двох вершин, які воно з'єднує, тобто добавляє 2 до суми степенів вже наявних вершин.

Наслідок. У кожному графі число вершин непарного степеня парне.

Неорієновані графи мають важливу характеристику, яка називається *зв'язністю* графа. Кажуть, що граф *зв'язний*, якщо будь-які дві його вершини можна з'єднати ланцюгом. Коли граф G незв'язний, то його можна розбити на такі підграфи G_i, що усі вершини в кожному з них будуть зв'язаними, а вершини із різних підграфів – не зв'язані. Такі підграфи G_i називають *компонентами зв'язності* графа G.

Приклад 2.4.Якщо з графа (рис. 2.7) виключити ізольовану вершину 5, то отриманий граф буде зв'язним. Граф на рис. 2.8 незв'язний і має дві компоненти зв'язності.

Його можна перетворити в зв'язний, додавши ребро (*міст*), що з'єднує вершини 3 і 5 (штрихова лінія). Видалення моста перетворює зв'язний граф у незв'язний.

Аби визначити зв'язність орієнтованого графа не потрібно звертати увагу на орієнтацію дуг.

Граф, зображений на рис. 2.2, є незв'язним. Проте його підграф, що складається з вершин *b, c, d, e*, вже буде зв'язним.

Для орієнтованого графа існує поняття сильної зв'язності.

Граф є *сильно зв'язним*, якщо для будь-яких двох його вершин x і y існує шлях, що йде з x в y.

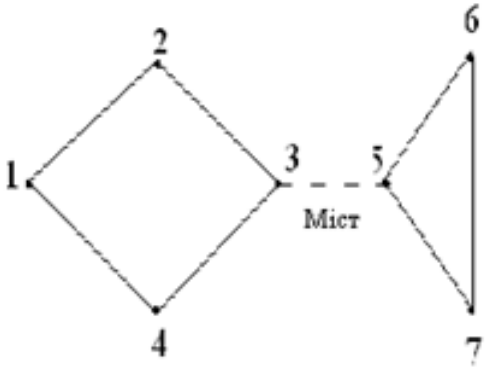


Рис. 2.8.

Важливим окремий випадком неорієнтованого графа є *дерево* (рис. 2.9).

Деревом називають кінцевий зв'язний неорієнтований граф, який не має циклів.

Якщо дано множину вершин *a, b, c,...* , то дерево можна побудувати таким чином. Одну з вершин, наприклад, *a*, візьмемо за початкову і назвемо її *коренем дерева*.

З цієї вершини проводимо ребра в сусідні вершини *b*, *c*, *d*,..., із них проводимо ребра в сусідні з ними вершини *e*, *f*, *g*, *h*,... і т.д. Відтак, дерево можна побудувати, послідовно додаючи ребра в його вершини. Це дає можливість встановити зв'язок між кількістю вершин і числом ребер дерева. Найпростіше дерево складається з 2-х вершин, які з'єднані ребром. Кожен раз, коли до дерева добавляємо ще одне ребро, до нього додається також і вершина (кінець цього ребра). Отже, дерево з *n* вершинами має *n* – 1 ребро.

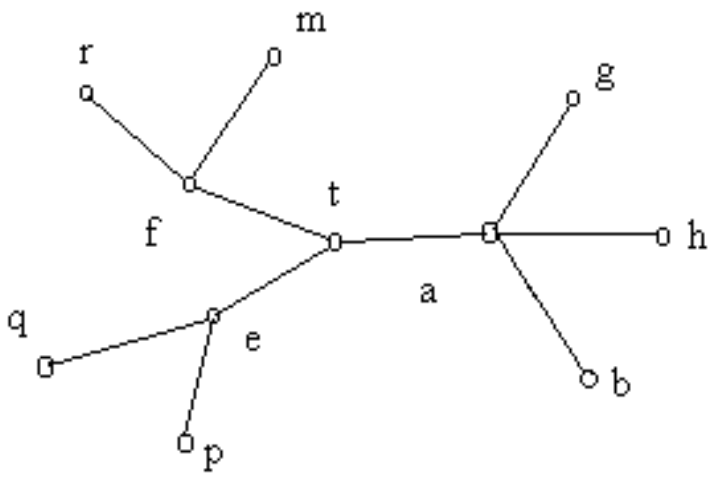


Рис. 2.9. Граф у формі дерева

2.3. Ізоморфізм графів

Як вже було зазначено вище, один і той самий граф геометрично можна зобразити різними засобами. На рис. 2.10 наведено три подання одного і того самого графа. Такі графи називаються *ізоморфними.*

Довільна підстановка φ на множині вершин графа *G*, що зберігає відношення суміжності, тобто така, при якій образи $\varphi(u)$ і $\varphi(v)$ вершин *u* і *v* суміжні тоді і тільки тоді, коли суміжними є вихідні вершини *u* і *v*, називається *автоморфізмом* графа *G*. Іншими словами, автоморфізм графа – це ізоморфізм графа на себе.

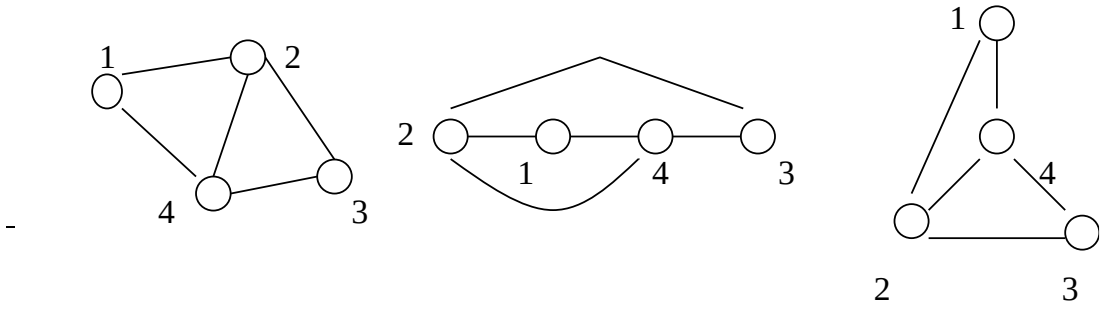


Рис. 2.10. Приклад ізоморфних графів

Довільний граф G має щонайменше один автоморфізм – тотожне перетворення E: $VG \to VG$, при якому $E(v) = v$ для будь-якої вершини v. Очевидно, що коли φ – автоморфізм графа G, то й обернена підстановка φ^{-1} також є автоморфізмом. Якщо обидві підстановки φ і ϕ є автоморфізмами, то і їхня композиція $\varphi\phi$ також буде автоморфізмом.

2.4. Відношення порядку і відношення еквівалентності на графі

Із зазначених вище властивостей очевидно, що граф дає зручне геометричне подання відношень на множині, тому теорія графів і теорія відношень на множині взаємно доповнюють одна одну.

Визначення. На графі $G = (X, \Gamma)$ введено відношення порядку, якщо для будь-яких двох вершин x і y, котрі задовольняють умову $x \le y$, існує шлях із x в y. У цьому випадку говорять, що вершина x *передує* вершині y.

Покажемо, що таке визначення відповідає усім властивостям відношення порядку.

Рефлексивність. Умова $x \le x$ означає еквівалентність вершини самій собі, тобто таку умову: $x \equiv x$. Проте, цю умову можна також інтерпретувати як наявність шляху з x в x, тобто як петлю у вершині x (рис. 2.11, *а*).

Транзитивність. Умова $(x \le y,\ y \le z) \to x \le z$ означає, що вершини x, y, z послідовно зустрічаються на одному і тому самому шляху (рис. 2.11, *б*, *в*).

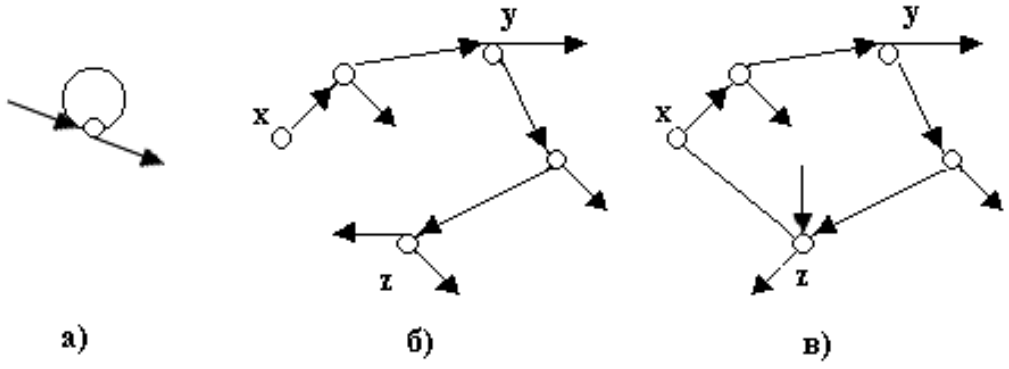


Рис. 2.11. Відношення порядку на графах

Антисиметричність. Покажемо справедливість умови $x \le y, y \le x \to x \equiv y$.

Дійсно, ліва частина цього виразу означає, що існує шлях із x в y, і одночасно існує шлях з y в x. Отже, в графі є контур, на якому лежать обидві вершини x і y.

З правої частини умови випливає, що вершини, котрі лежать на одному і тому самому контурі, є еквівалентними.

Будемо вважати цей висновок визначенням еквівалентності на графі і покажемо, що воно задовольняє всім умовам, висунутим щодо відношення еквівалентності.

Нагадаємо, що відношення еквівалентності має бути рефлексивним, симетричним і транзитивним. Відтак, впевнимось, що ці властивості мають місце.

Виконання умов рефлексивності ($x \equiv x$) і симетричності ($x \equiv y \to y \equiv x$) є очевидними і випливають із даного вище визначення еквівалентності. Умова транзитивності $x \equiv y, y \equiv z \to x \equiv z$ також є очевидною, оскільки означає, що коли в графі є контур із вершинами x і y, а також контур із вершинами y і z , то існує також і контур, якому належать вершини x і z.

Таким чином, всіляке відношення порядку разом із відношенням еквівалентності визначає деякий граф.

На графі може бути також уведено відношення строгого порядку. У цьому випадку для будь-яких двох вершин x і y, що задовольняють умову: $x < y$, існує шлях, котрий йде з вершини x до вершини y. Умова транзитивності для відношення строгого порядку $x < y < z \to x < z$ означає, як і в попередньому випадку, що вершини x, y і z зустрічаються послідовно на одному і тому самому шляху. Умова антирефлексивності ($x < x$ хибно) означає відсутність петель на графі, а умова асиметричності (якщо $x < y$, то $y < x$ хибно) означає відсутність контурів.

Отже, відношенню строгого порядку відповідає граф без контурів.

Запишемо всі дані про властивості відношень порядку в табл. 2.1.

Таблиця 2.1. Визначення різних видів відношень порядку (порядків)

Порядки	Відношення					
	Антисиметричне	Транзитивне	Рефлексивне	Антирефлексивне	Повне	Неповне
Строгий	+	+		+		
Нестрогий	+	+	+			
Повний	+	+			+	
Неповний	+	+				+

2.5. Характеристики графів

Розв’язування багатьох технічних задач методами теорії графів зводиться до визначення тих або інших характеристик графів. Хоча вивчення технічних застосувань теорії графів не є метою цієї книги, знайомство з найважливішими характеристиками графів може виявитися корисним при опануванні інших дисциплін.

Розглянемо деякі характеристики графів.

Цикломатичне число. Нехай G – неорієнтований граф, що має n вершин, m ребер і r компонент зв'язності. Цикломатичним числом графа G є число $C(G) = m - n + r$.

Воно має цікавий фізичний зміст, а саме: дорівнює найбільшому числу незалежних циклів у графі. Цю характеристику використовують, наприклад, для визначення числа незалежних контурів, коли розраховують електричні кола.

Хроматичне число. Нехай p – натуральне число. Граф G називають p-хроматичним, якщо його вершини можна пофарбувати p різними кольорами таким чином, щоб ніякі дві суміжні вершини не були пофарбовані однаково. Найменше число p, при якому граф є p-хроматичним, називають хроматичним числом графа і позначають $\Psi(G)$.

Якщо $\Psi(G) = 2$, то граф називають *біхроматичним*. Необхідною і достатньою умовою аби граф був біхроматичним, є відсутність у ньому циклів непарної довжини. Хроматичне число грає важливу роль при розв'язуванні задачі найбільш економного використання пам'яті у програмуванні. Проте його визначення, за винятком випадку біхроматичного графа, являє собою досить складну задачу і потребує застосування сучасних обчислювальних засобів.

Зв'язним називається граф, у якому всі вершини пов'язані між собою (тобто існує ланцюг, який з'єднує їх).

Нехай G – зв'язний неорієнтований граф, v_1 і v_2 – будь-які дві його вершини. Тоді існує простий ланцюг, котрий пов'язує ці дві вершини: $M(L_1, L_2, \dots, Lq)$. Якщо кількість ребер q цього ланцюга не мінімальна з можливих, то є інший ланцюг $M^*(l_1, l_2, \dots, lq^*)$, що зв'язує v_1 і v_2 і має меншу кількість ребер. Отже, існує ланцюг M, з мінімальною кількістю ребер p, який зв'язує вершини v_1 і v_2. Мінімальна довжина простого ланцюга з початком v_1 і кінцем v_2 називається *відстанню* $d(v_1, v_2)$ між цими вершинами.

Відстань $d(v_1, v_2)$ задовольняє таким умовам (аксіомам метрики):

1. $d(v_1, v_2) \geq 0$, для всіх v_1, v_2, $d(v_1, v_2) = 0$ якщо $v_1 = v_2$;
2. $d(v_1, v_2) = d(v_2, v_1)$;
3. $d(v_1, v_2) + d(v_1, v_3) \geq d(v_1, v_3)$

Діаметр графа визначається у такий спосіб:

$$d(G) = \max_{v_1, v_2 \in G} d(v_1, v_2), \tag{2.6}$$

Нехай v – довільна вершина графа G. *Максимальним віддаленням* у графі G від вершини v є величина $p(v) = \max_{v_1 \in G} d(v, v_1)$.

Вершина U називається центром графа, якщо максимальне віддалення від неї набуває мінімального значення, а саме:

$$P(U) = \min_{v \in G} p(v), \tag{2.7}$$

Максимальне віддалення $p(G)$ від центра називається *радіусом* графа.

Центр не обов'язково має бути єдиним.

Наприклад, у повному неорієнтованому графі, у котрому будь-які дві різні вершини v_1,v_2 з'єднані ребром, радіус дорівнює 1 і будь-яка вершина є центром.

Множина внутрішньої сталості. Множину $S \subseteq X$ графа $G = (X, \Gamma)$ називають внутрішньо стійкою, якщо жодні дві вершини з S не є суміжними, тобто для будь-якого $X \in S$ має місце рівність $\Gamma_x \cap S = \varnothing$

Множину внутрішньої сталості, що містить найбільше число елементів, називають найбільшою внутрішньо стійкою множиною, а кількість її елементів – *числом внутрішньої сталості* графа G. Найбільша внутрішньо стійка множина грає важливу роль у теорії зв'язку.

Множина зовнішньої сталості. Множину $T \subset X$ графа $G = (X, \Gamma)$ називають зовнішньо стійкою, якщо будь-яка вершина, що не належить T, сполучена дугами із вершинами з множини T, тобто для довільної вершини $x \notin T$ має місце $\Gamma_x \cap T \neq \varnothing$.

Множину зовнішньої сталості, яка містить найменше число елементів, називають найменшою зовнішньо стійкою множиною, а кількість її елементів– *числом зовнішньої сталости* графа G.

2.6. Ейлерові графи і гамільтонові цикли

Розрізняють ейлерові цикли і ейлерові графи.

У теорії графів ланцюг, який проходить кожне ребро тільки один раз, називають *ланцюгом Ейлера*. Так само, цикл Ейлера – це ланцюг Ейлера, який розпочинається та завершується в одній і тій самій вершині. Ейлеровим циклом можна вважати слід олівця, що викреслює цей цикл, не відриваючись від паперу

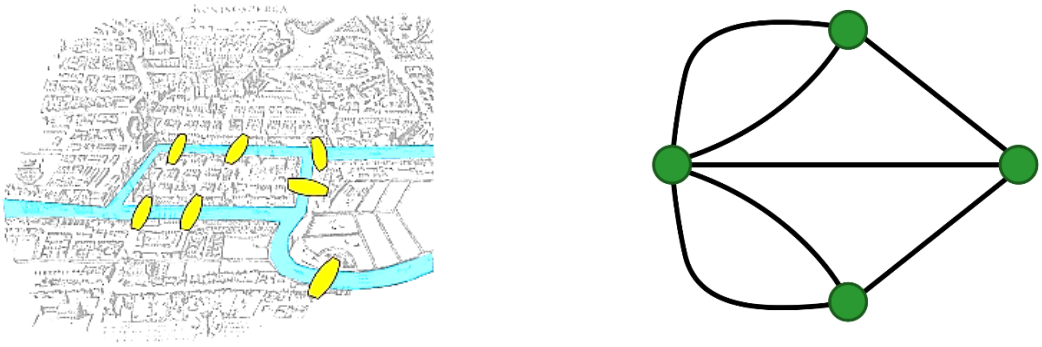
Ці поняття були вперше введені Леонардом Ейлером під час розв'язання відомої задачі кенігсберзьких мостів[4] 1736 року.

Математично задача формулюється таким чином: чи можна для графа на рис. 2.12 побудувати ланцюг (або цикл), який проходить кожне ребро тільки один раз?

Ейлер довів, що необхідна умова існування циклу — парність степеня кожної вершини графу, і стверджував без доведення, що зв'язний граф в якому всі вершини мають парний степінь має цикл Ейлера[5].

[4] Сім мостів Кенігсберґа — відома історична задача математики. Доведення неможливості її розв'язання Леонардом Ейлером в 1735 році спричинило створення теорії графів і передувало виникненню топології. Місто Кенігсберґ в Східній Пруссії (нині Калінінград у Росії) було розташоване на берегах річки Преголя, рукави якої ділили його тоді на чотири частини, в тому числі й два острови — Kneiphof (Кнайпгоф) і Lomse (Ломзе), що поєднувалися сімома мостами: Krämerbrücke (Крамарський), Grünebrücke (Зелений), Köttelbrücke (Робітничий), Schmiedebrücke (Ковальський), Holzbrücke (Дерев'яний), Hohebrücke (Високий) і Honigbrücke (Медовий).
Необхідно було знайти такий маршрут через місто, аби пройти всі сім мостів і кожним мостом пройти рівно один раз. На острови не можна було потрапити інакше як через міст. Кожен з мостів мав бути пройденим повністю один раз (тобто не можна було піти на середину мосту і повернутися назад, а потім з іншого берега перетнути другу половину). Ейлер довів, що розв'язку не існує.

[5] Перше повне доведення цього твердження 1873 р. оприлюднив Карл Гіргольцер



а *б*

Рис. 2.12. Задача про сім кенінгсберзських мостів

Отже, сформулюємо умови, за яких граф є ейлеровим.

Скінченний неорієнтований граф G буде ейлеровим тоді і тільки тоді, коли він є зв'язним і степені всіх його вершин парні.

У незв'язному графі кожний цикл належить до якої-небудь його зв'язної частини, тобто не проходить через усі його ребра.

У кожну вершину може входити кілька дуг за умови, що виходять вони з різних вершин. Отже, їх має бути парне число.

Ейлеровим називають ланцюг, що включає всі ребра даного кінцевого неорієнтованого графа G , але має різні початок (v_1) і кінець (v_2).

Аби у графі існував ейлерів ланцюг, необхідна його зв'язність і парність степенів усіх вершин, крім початкової і кінцевої. Останні дві вершини мусять мати непарні степені: із v_1 ми зайвий раз виходимо, а в v_2 зайвий раз входимо. Ці умови є достатніми для існування ейлерового ланцюга.

Випадок скінченного орієнтованого графа.

Аби у скінченному орієнтованому графі існував ейлеровий цикл, необхідно і достатньо, щоб степені вершин графа за вхідними і вихідними дугами були однакові.

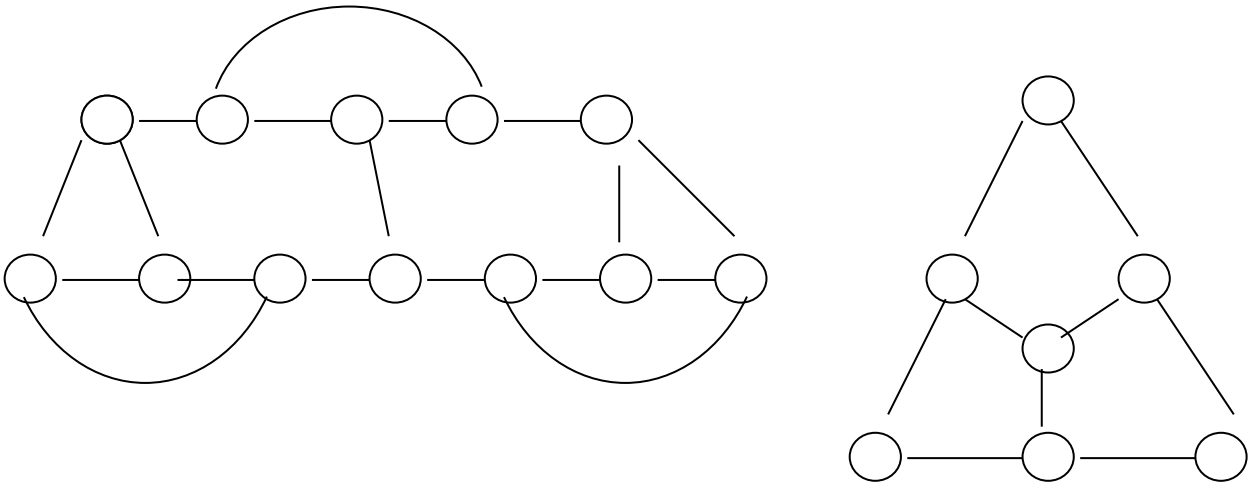
Справедливість цього твердження випливає із факту, що будь-якому неорієнтованому графу канонічно відповідає орієнтований, у котрому кожне ребро замінюється двома спрямованими дугами, котрі інцидентні тим самим вершинам і є такими, що йдуть у протилежному напрямку.

У скінченному зв'язному графі завжди можна побудувати орієнтований цикл, який проходить через кожне ребро один раз в кожному з двох напрямків. Такий цикл іноді називається *засобом обходу всіх дуг графа*. Він використовується в багатьох прикладних задачах, пов'язаних з графами.

Га́мільтоновим називається шлях, який містить кожну вершину графа тільки один раз. Якщо його початкова і кінцева вершини збігаються, то мова йде про гамільтонів цикл[6].

Отже, *гамільтоновим* називається простий цикл, який проходить через усі вершини графа, що аналізується. Він існує не у всякому графі (рис. 2.13)

[6] Гамільтонові шлях, цикл і граф названі на честь ірландського математика Вільяма Гамільтона, який вперше визначив ці класи, дослідивши задачу «навколосвітньої подорожі» по додекаедру, вузлові вершини якого символізували найбільші міста Землі, а ребра — дороги, що їх з'єднують.



Існує Не існує

Рис. 2.13. Приклади графів

Незважаючи на зовнішню подібність, задачі розпізнавання ейлеровості і гамільтоновості графа принципово різні. Правило визначення ейлеровості описано вище. Що ж стосується гамільтоновості графа, то відповідь на це питання дає теорема, яка без доведення наводиться нижче.

Теорема. Граф із степеневою послідовністю $d_1 \le d_2 \le ... \le d_n$ є гамільтоновим, якщо для всякого числа k, яке задовольняє нерівностям $1 \le k < n/2$, буде істинною така імплікація (відповідність):

$$(d_k \le k) \Rightarrow (d_{n-k} \ge n - k) \qquad (2.4)$$

Існують і інші як більш сильні, так і більш слабкі теореми й умови визначення гамільтоновості графів.

Розглянемо типові оптимізаційні задачі на графах і методи їх розв'язування.

2.7. Задача про найкоротший шлях у графі

Наведемо приклади практичних задач, розв'язування яких може бути зведено до пошуку найкоротшого шляху в графі.

Приклад 2.5. Маємо мережу автомобільних доріг, яка з'єднує міста Дніпропетровської області. Деякі з них односторонні. Знайти найкоротший шлях від м. Дніпро до будь-якого іншого міста області (якщо рухатися можна лише по дорогах).

Приклад 2.6. Наявна певна кількість авіарейсів між містами світу, вартість кожного з них відома. До того ж вартість перельоту з міста A в місто B не обов'язково дорівнює вартості перельоту у зворотному напрямку. Визначити маршрут між двома заданими містами (можливо з пересадками), який має мінімальну вартість.

Сформулюємо цю задачу математично.

Кожній дузі (x,y) вихідного графа G поставимо у відповідність число $a(x,y)$. Якщо в графі G відсутня деяка дуга (x,y), то припустимо, що $a(x,y)=\infty$. Будемо називати число $a(x,y)$ *довжиною дуги* (x,y), хоча $a(x,y)$ можна інтерпретувати і як відповідні витрати, або ваговий коефіцієнт. *Довжиною шляху* будемо вважати суму значень довжин окремих дуг, які створюють цей шлях.

Для будь-яких двох вершин *s* і *t* графа *G* можуть існувати кілька шляхів, які з'єднують вершину *s* із вершиною *t*. Необхідно визначити шлях мінімальної довжини, котрий з'єднує вершину *s* і вершину *t*. Його називають *найкоротшим* шляхом між вершинами *s* і *t*.

Нижче буде розглянуто алгоритм, який дозволяє визначити такий шлях.

Алгоритм пошуку найкоротшого шляху (алгоритм Дейкстри)

Ідея алгоритму. Кожній вершині з множини *X* поставимо у відповідність число *d*(*x*) – найменшу відому відстань від цієї вершини до *s*. Алгоритм включає кілька кроків, зокрема, на кожному з них опрацьовується одна вершина з метою зменшення чисел *d*(*x*). Робота алгоритму закінчується, коли всі вершини опрацьовано.

Опишемо цей алгоритм.

Крок 1. *Ініціалізація*.

Задамо, що $d(s)=0$ і $d(x)=\infty$ для всіх вершин *x*, відмінних від *s* (це відображає той факт, що відстані до цих вершин на даний момент не відомі). Жодну вершину ще не опрацьовано (не забарвлено). Через *y* позначимо останню із забарвлених вершин. Позначимо (пофарбуємо) вершину *s* і будемо вважати, що $y = s$.

Крок 2. *Основний крок алгоритму*.

Для кожної незабарвленої вершини *x* перерахуємо величину відстані $d(x)$ в такий спосіб:

$$d(x)=\min\{d(x);d(y)+a(y,x)\}. \qquad (2.5)$$

Якщо $d(x)=\infty$ для всіх незабарвлених вершин *x*, то належить закінчити процедуру алгоритму, оскільки у вихідному графі відсутні шляхи з вершини *s* у незабарвлені.

Коли ні, то слід пофарбувати ту з вершин *x*, для якої величина $d(x)$ найменша. Крім того, потрібно пофарбувати дугу, що веде до обраної на даному кроці вершини *x* [саме для цієї дуги досягався мінімум виразу (2.5)]. Вважати, що $y=x$.

Крок 3. Якщо $y=t$, то процедуру закінчити, оскільки найкоротший шлях від вершини *s* до вершини *t* знайдений (це єдиний шлях від *s* до *t*, котрий складається із забарвлених дуг). А якщо ні, то перейти до *кроку* 2.

Алгоритм описано.

Забарвлені дуги утворюють у вихідному графі орієнтоване дерево з коренем у вершині *s*. Його називають *орієнтованим деревом найкоротших шляхів.* Єдиний шлях від вершини *s* до будь-якої вершини *x*, що йому належить, буде найкоротшим між зазначеними вершинами.

Оскільки на всіх етапах алгоритму Дейкстри забарвлені шляхи утворюють у вихідному графі орієнтоване дерево, то алгоритм можна розглядати як процедуру нарощування орієнтованого дерева з коренем у вершині *s*. Коли в процедурі нарощування досягнута вершина *t*, процедуру припиняють.

Для визначення найкоротших шляхів від вершини *s* до всіх вершин вихідного графа процедуру нарощування дерева слід продовжити, допоки всі вершини графа не будуть включені в орієнтоване дерево найкоротших шляхів. При цьому для вихідного графа буде отримано *покривне дерево* (звісно, якщо в даному графі існує хоча б одне з таких).

Отже, аби описаний вище алгоритм дозволив одержати дерево найкоротших шляхів від вершини *s* до всіх інших вершин, його третій крок повинен бути скорегований у такий спосіб:

Крок 3а. Якщо всі вершини виявляються забарвленими, то процедуру закінчують (існує єдиний найкоротший шлях від вершини *s* до будь-якої вершини *x* і він складається тільки із забарвлених дуг). А якщо ні, то повертаються до *кроку* 2.

Зауважимо, що алгоритм Дейкстри застосовується при роботі з орієнтованими графами, які не мають петель, а довжини дуг не набувають від'ємних значень. Але, він може бути узагальнений на випадок, коли деякі з дуг мають від'ємні значення довжини. Необхідна модифікація описаного вище алгоритму (Форда) передбачає такі дії:

– На *другому кроці* алгоритму перерахування величини відстані $d(x)$ за допомогою співвідношення (2.1) проводиться для всіх вершин, а не тільки для незабарвлених. Отже, числа $d(x)$ можуть зменшуватися як для незабарвлених, так і для пофарбованих вершин.

– Якщо для деякої пофарбованої вершини *x* відбувається зменшення величини $d(x)$, то із неї та з інцидентної ній пофарбованої дуги забарвлення знімається.

– Процедура алгоритму закінчується тільки тоді, коли всі вершини пофарбовані й якщо після виконання кроку 2 жодне із чисел $d(x)$ не змінюється.

Приклад 2.7. За допомогою алгоритму Дейкстри визначити найкоротший шлях між вершинами *s* і *t* у графі, зображеному на рис. 2.7.

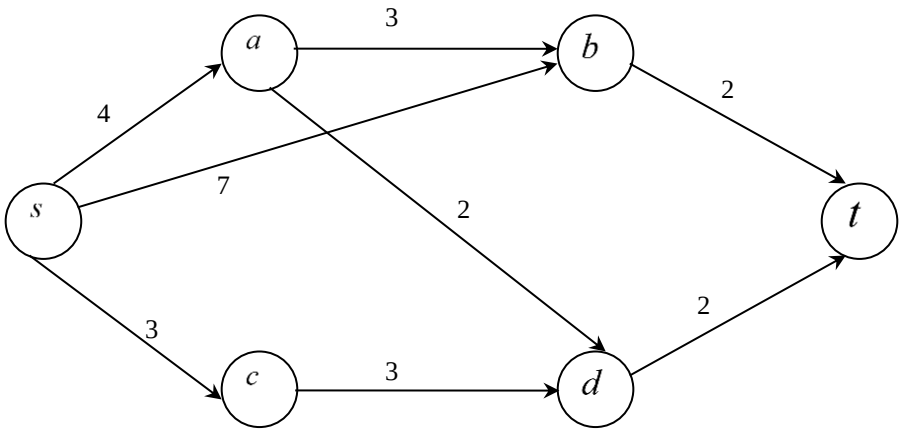


Рис. 2.14. Граф до прикладу 2.7

Розв'язування

Крок 1. Зафарбуємо вершину *s*. Будемо вважати, що $d(s)=0$ і $d(a)=d(b)=d(c)=d(d)=d(t)=\infty;\ \ (y=s).$ На графі запишемо значення величин $d(x)$ поряд із вершиною. Тоді він набуває поданого на рис. 2.15 вигляду.

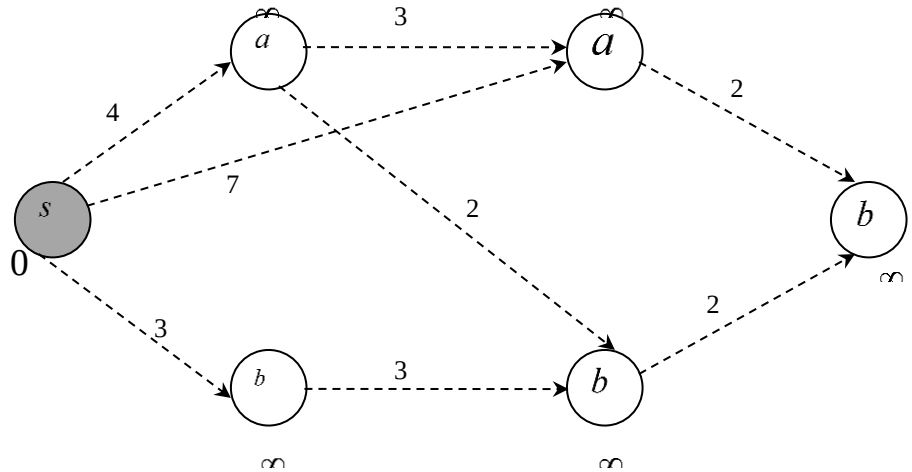


Рис. 2.15. Перший крок алгоритму пошуку найкоротшого шляху

Крок 2. Для незабарвлених вершин зробимо перерахування відстаней $d(x)$, а саме:

$$d(a)=\min\{d(a),d(s)+a(s,a)\}=\min\{\infty,0+4\}=4;$$

$$d(b)=\min\{d(b),d(s)+a(s,b)\}=\min\{\infty,0+7\}=7;$$

$$d(c)=\min\{d(c),d(s)+a(s,c)\}=\min\{\infty,0+3\}=3;$$

$$d(d)=\min\{d(d),d(s)+a(s,d)\}=\min\{\infty,0+\infty\}=\infty;$$

$$d(t)=\min\{d(t),d(s)+a(s,t)\}=\min\{\infty,0+\infty\}=\infty.$$

Оскільки $\min\{d(a),d(b)d(c),d(d),d(t)\}=\min\{4,7,3,\infty,\infty\}=3=d(c)$, то зафарбовуємо вершину c і дугу (s,c). Поточне дерево найкоротших шляхів показано на графі (рис. 2.16) товстою лінією.

Крок 3. Через те, що вершина t залишилася незабарвленою, повертаємось до кроку 2, передбачивши, що $y=c$.

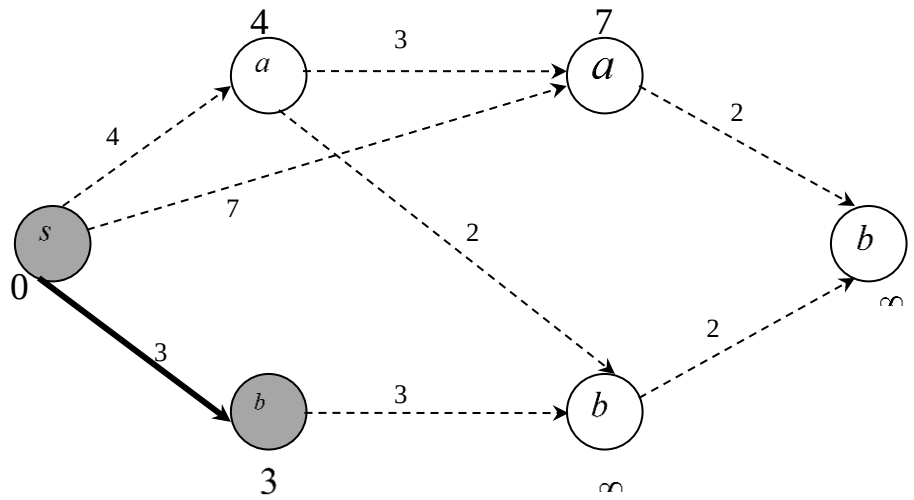


Рис. 2.16. Побудова дерева найкоротших шляхів (крок 2, $y=s$)

Крок 2 $(y=c)$.

Перераховуємо величини $d(x)$ для незабарвлених вершин x, а саме:

$$d(a)=\min\{d(a),d(c)+a(c,a)\}=\min\{4,3+\infty\}=4;$$

$$d(b)=\min\{d(b),d(c)+a(c,b)\}=\min\{7,3+\infty\}=7;$$

$$d(d)=\min\{d(d),d(c)+a(c,d)\}=\min\{\infty,3+3\}=6;$$

$$d(t)=\min\{d(t),d(c)+a(c,t)\}=\min\{\infty,3+\infty\}=\infty.$$

Оскільки $\min\{d(a),d(b),d(d),d(t)\}=\min\{4,7,6,\infty\}=4=d(a)$, то зафарбовуємо вершину a й дугу (s,a). Поточне дерево найкоротших шляхів набуває вигляду, зображеного на рис. 2.17.

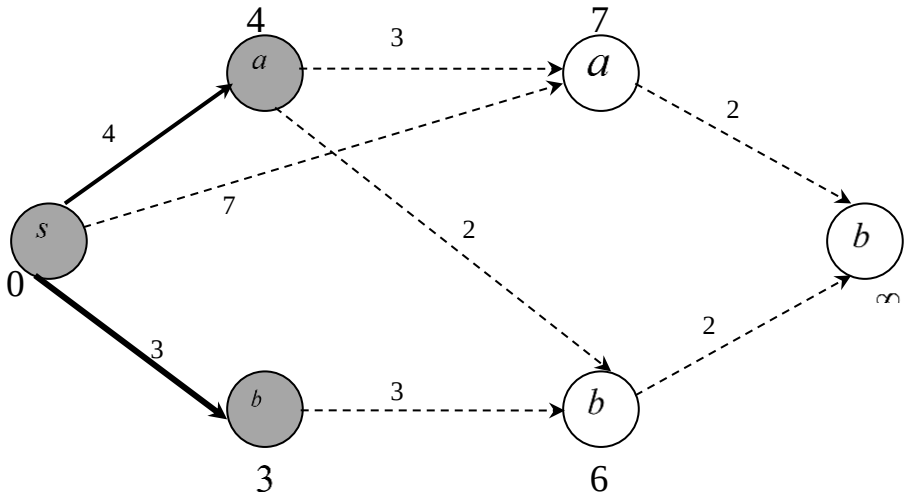


Рис. 2.17. Побудова дерева найкоротших шляхів (крок 2, $y = c$)

Крок 3**.** Вершина t залишилася незабарвленою, тому повертаємось до кроку 2, зважаючи, що $y = a$.

Крок 2 $(y = a)$.

Для незабарвлених вершин робимо перерахування чисел $d(x)$, а саме:

$$d(b) = \min\{d(b), d(a) + a(a,b)\} = \min\{7, 4+3\} = 7;$$

$$d(d) = \min\{d(d), d(a) + a(a,d)\} = \min\{6, 4+2\} = 6;$$

$$d(t) = \min\{d(t), d(a) + a(a,t)\} = \min\{\infty, 4+\infty\} = \infty.$$

Враховуючи, що $\min\{d(b), d(d), d(t)\} = \min\{7, 6, \infty\} = 6 = d(d)$, зафарбовуємо вершину d й дугу (a,d). Зауважимо, що можна було зафарбувати дугу (c,d).

Поточне дерево найкоротших шляхів набуває вигляду, показаного на рис. 2.18.

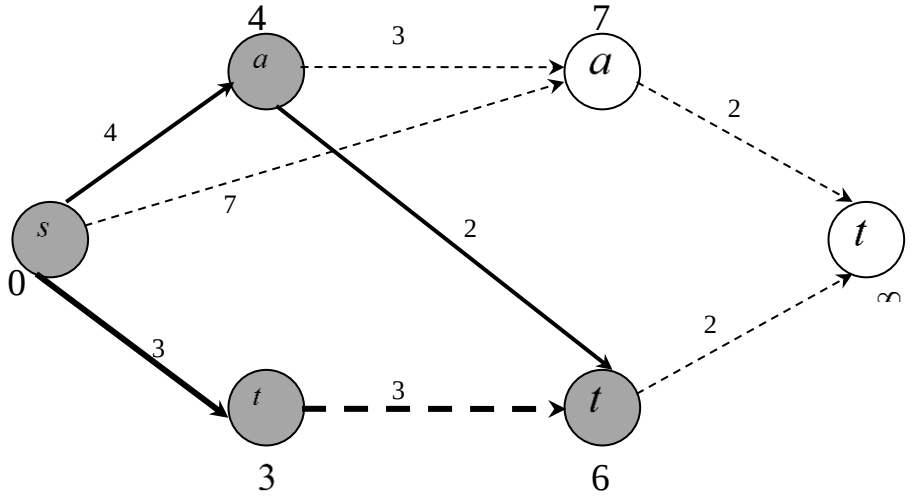


Рис. 2.18. Поточне дерево найкоротших шляхів (крок 2, $y = a$)

Крок 3. Оскільки вершина t залишилася незабарвленою, то знов повертаємось до кроку 2, вважаючи, що $y = d$.

Крок 2 $(y = d)$.

Для незабарвлених вершин робимо перерахування відстаней $d(x)$:

$$d(b) = \min\{d(b), d(d) + a(d,b)\} = \min\{7, 6 + \infty\} = 7;$$

$$d(t) = \min\{d(t), d(d) + a(d,t)\} = \min\{\infty, 6 + 2\} = 8.$$

Оскільки $\min\{d(b), d(t)\} = \min\{7,8\} = 7 = d(b)$, то зафарбовуємо вершину b й дугу (s,b). Можна було б також зафарбувати дугу (a,b).

Поточне дерево найкоротших шляхів набуває вигляду, поданого на рис. 2.19.

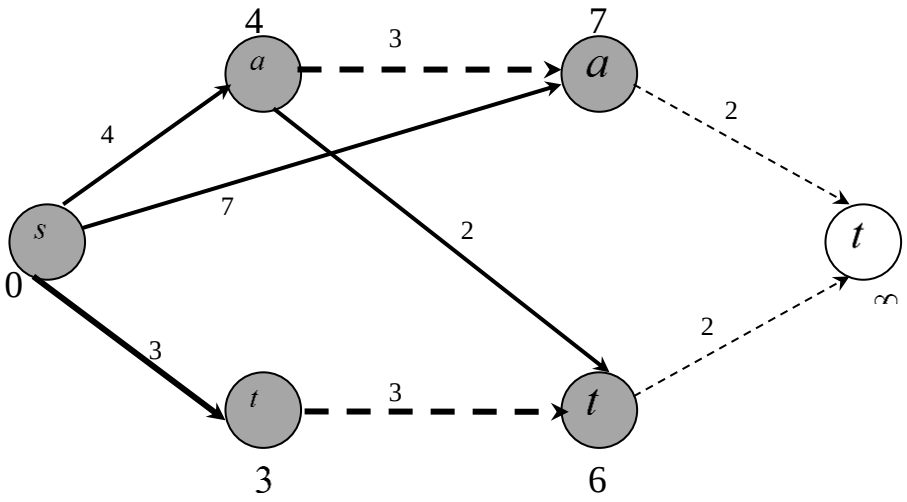


Рис. 2.19. Поточне дерево найкоротших шляхів (крок 2, $y = d$)

Крок 3. Вершина t залишилася незабарвленою, тому знову повертаємось до кроку 2, передбачивши, що $y = b$.

Крок 2. $(y = b)$.

Для незабарвлених вершин робимо перерахування чисел $d(x)$, а саме:

$$d(t) = \min\{d(t), d(b) + a(b,t)\} = \min\{8, 7 + 2\} = 8.$$

Отже, вершину t, нарешті, можна пофарбувати. Разом з нею фарбуємо й дугу (d,t), яка визначає величину $d(t)$.

Остаточно побудоване дерево найкоротших шляхів має вигляд, показаний на рис. 2.20.

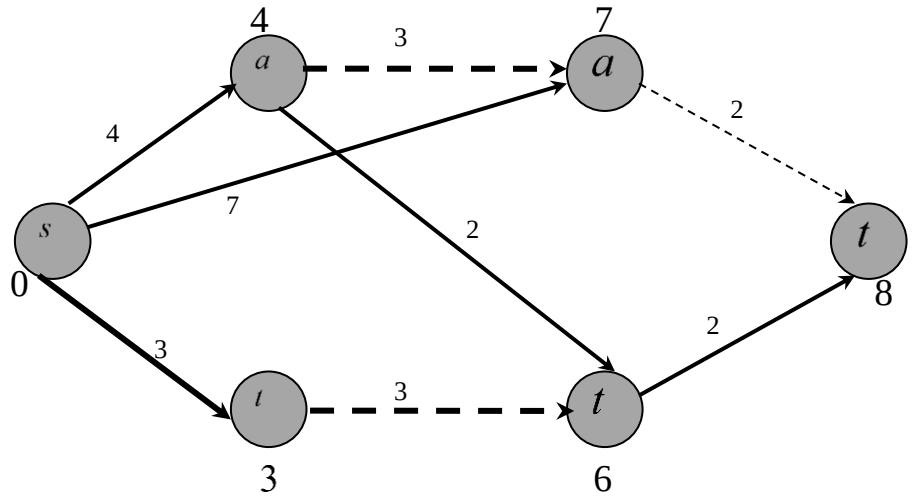


Рис. 2.20. Дерево найкоротших шляхів у графі з прикладу 1

Очевидно, що у вихідному графі є два найкоротші шляхи однакової довжини: перший шлях складається з дуг $(s,a),(a,d),(d,t)$; другий включає дуги $(s,c),(c,d),(d,t)$.

Зауважимо, що найкоротший шлях буде єдиним тільки тоді, коли під час процедури алгоритму жодного разу не виникає неоднозначності у виборі забарвленої дуги.

Приклад 2.8. Застосуємо алгоритм Форда до графа, поданого на рис. 2.21.

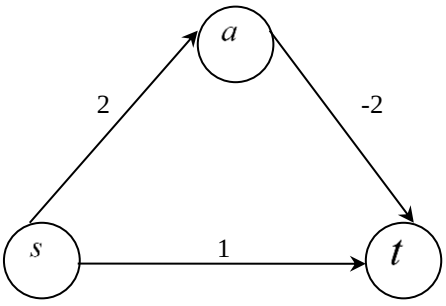


Рис. 2.21. Граф до прикладу 2.8

Розв'язування

Крок 1. Зафарбуємо вершину s. Задамо, що $d(s)=0$ і $d(a)=d(t)=\infty$, $y=s$. Позначимо величини $d(x)$ поряд з вершинами графа (рис. 2.12).

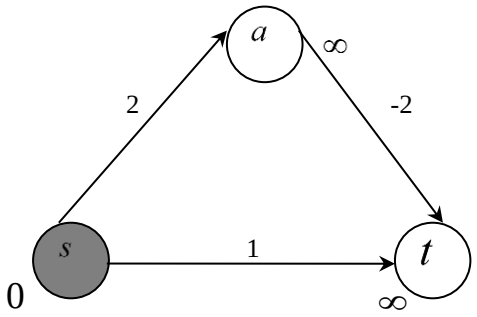


Рис. 2.22. Крок 1 алгоритму Форда в прикладі 2

Крок 2 $(y = s)$.

Для незабарвлених вершин *x* робимо перерахування чисел $d(x)$, а саме:

$$d(a) = \min\{d(a), d(s) + a(s,a)\} = \min\{\infty, 0+2\} = 2;$$

$$d(t) = \min\{d(t), d(s) + a(s,t)\} = \min\{\infty, 0+1\} = 1.$$

Беручи до уваги, що $\min\{d(a), d(t)\} = 1 = d(t)$, зафарбовуємо вершину *t* й дугу (s,t). Поточне дерево найкоротших шляхів набуває вигляду, зображеного на рис. 2.23.

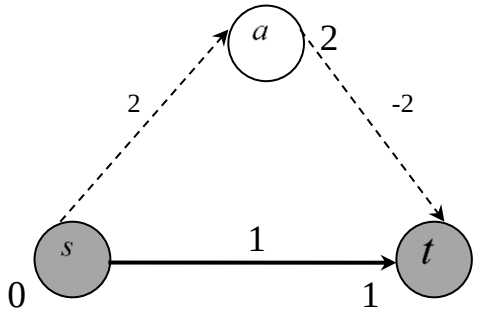


Рис. 2.23. Крок 2 алгоритму Форда у прикладі 2 $(y = s)$

Крок 3. Оскільки забарвленими виявилися не всі вершини, повертаємось до кроку 2, вважаючи, що $y = t$.

Крок 2 $(y = t)$

Враховуючи те, що з вершини *t* не виходить жодна дуга, усі числа $d(x)$ залишаються незмінними, тому фарбуємо вершину *a* і разом з нею дугу (s,a). Поточне дерево найкоротших шляхів позначено на графі (див. рис. 2.24) товстою лінією.

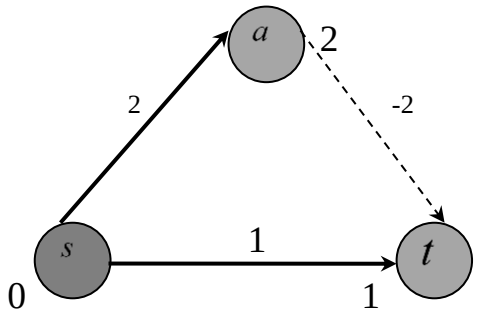


Рис. 2.24. Крок 2 алгоритму Форда в прикладі 2 $(y=t)$

Крок 3. Здійснюємо повернення до кроку 2, аби спробувати зменшити значення чисел $d(x)$.

Крок 2 $(y=a)$.

Обчислюємо, що

$$d(t)=\min\{d(t),d(a)+a(a,t)\}=\min\{1,2-2\}=0,$$

$$d(s)=\min\{d(s),d(a)+a(a,s)\}=\min\{0,2+\infty\}=0.$$

Позаяк величина $d(t)$ зменшується від 1 до 0, то з вершини *t* і дуги (s,t) фарбування знімається.

Тепер поточне дерево найкоротших шляхів має вигляд, показаний на рис. 2.25.

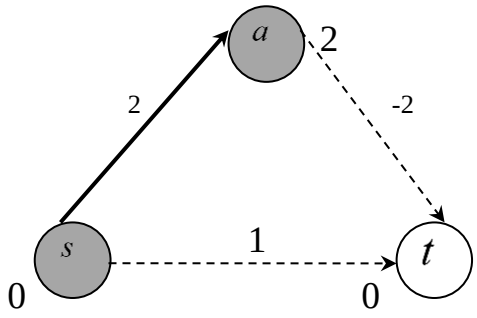


Рис. 2.25. Крок 2 алгоритму Форда в прикладі 2 $(y=a)$

У вихідному графі залишається незабарвленою тільки вершина *t*. Зафарбовуємо її разом з дугою (a,t), оскільки на даному кроці $y=a$.

Дерево найкоротших шляхів тепер набуває вигляду, зображеного на рис. 2.16.

Крок 3. Здійснюємо чергове повернення до кроку 2.

Крок 2 $(y=t)$

Враховуючи те, що з вершини *t* не виходить жодна дуга, значення чисел $d(x)$ не змінюються. Незабарвлених вершин у графі немає.

Крок 3. З огляду на те, що всі вершини в графі виявляються забарвленими й на попередньому кроці алгоритму жодну з величин $d(x)$ зменшити не вдалося, процедура алгоритму завершується, найкоротший шлях з вершини *s* у вершину *t* складається з дуг $(s,a),(a,t)$ і має таку довжину: 2 – 2 = 0 (див. рис. 2.26).

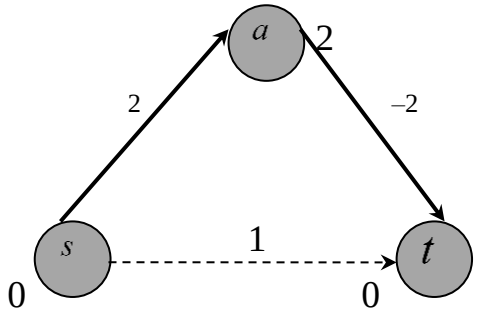


Рис. 2.26. Найкоротший шлях у графі з прикладу 2

Зауважимо, що алгоритм Форда не забезпечує розв'язування зазначеної задачі за умови наявності у вихідному графі контуру, довжина якого становить від'ємну величину. Дійсно, у цьому випадку можна нескінченно «повторюючи» цей контур, одержати шлях як завгодно малої довжини.

У ситуації, коли немає даних про відсутність або наявність у розглянутому графі контурів від'ємної довжини, можна застосовувати алгоритм Форда, але під час його роботи необхідно враховувати, скільки разів забарвлюються окремі вершини. Коли кількість забарвлень однієї з вершин досягає величини *N* (число вершин графа), процедуру алгоритму можна зупинити, бо вихідний граф містить контур, довжина якого має від'ємне значення. Якщо ж цього не відбувається, то процедура алгоритму Форда завершується протягом скінченної кількості кроків, а вихідну задачу буде розв'язано правильно.

2.8. Задача побудови мінімального кістякового дерева

Одним з поширених практичних питань, які в математичному формулюванні являють собою задачу про знаходження мінімального кістякового (покривного) дерева, буде таке: припустимо, існує *n* міст, які потрібно з'єднати дорогами таким чином, щоб можна було дістатися з кожного міста в будь-яке інше (прямим сполученням або через інші міста). Дозволяється будувати дороги між заданими парами міст, причому відома вартість будівництва кожної з них. Необхідно вирішити, які саме дороги треба прокласти, аби мінімізувати загальну вартість будівництва.

З огляду на теорію графів, ця задача може бути сформульована в такий спосіб: знайти мінімальне кістякове дерево в графі, вершини якого відповідають містам, а ребра – парам міст, між якими можна прокласти пряму дорогу. При цьому вага ребра дорівнює вартості її будівництва.

Сформульована задача розв'зується досить просто. Виконання алгоритму починається з вибору довільного вузла мережі та найкоротшої дуги із множини дуг, що з'єднують цей вузол з іншими. Після цього з'єднуємо два вузли обраною дугою і вибираємо найближчий до них третій вузол. Далі додаємо цей вузол і відповідну дугу до шуканої мережі. Продовжуємо допоки всі вузли не будуть з'єднані між собою. Алгоритм, що базується на «поглинанні» найкоротших дуг, описуємо нижче.

Алгоритм побудови мінімального кістякового дерева

Крок 1. Використовуючи вузли (вершини) вихідної мережі, визначаємо такі дві множини:

s – множину з'єднаних вузлів;

$\overline{s}$ – множину несполучених вузлів.

Спочатку всі вузли будуть належати множині $\overline{s}$.

Крок 2. Із множини $\overline{s}$ вибираємо довільний вузол і з'єднуємо його з найближчим сусіднім (після виконання даного кроку множина s буде містити два вузли).

Крок 3. Серед усіх дуг, що з'єднують вузли множини s із вузлами множини $\overline{s}$, вибираємо найкоротшу дугу. Кінцевий вузол цієї дуги, який перебуває в множині $\overline{s}$, позначимо через δ. Вилучаємо вузол δ із множини $\overline{s}$ і поміщаємо його в множину s.

Крок 4. Виконуємо крок 3, допоки всі вузли не будуть належати множині s.

Алгоритм описано.

Приклад 2.9. Знайти мінімальне дерево в мережі, зображеній на рис. 2.27.

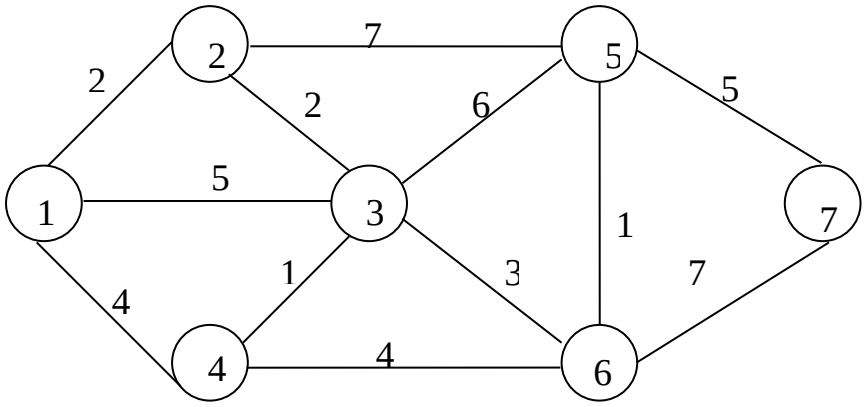


Рис. 2.27. Мережа до прикладу 2.9

Розв'язування

Крок 1. Визначаємо множини S, $\overline{s}$, а саме:

$$\overline{s} = \{1, 2, 3, 4, 5, 6, 7\},\ s = \varnothing.$$

Крок 2. Вибираємо для розгляду вузол 6, тоді $s = \{6, 5\}$, $\bar{s} = \{1, 2, 3, 4, 7\}$.

Побудована на цьому кроці мережа на рис. 2.28. зображена товстою лінією. Вартість побудованої мережі p = 1.

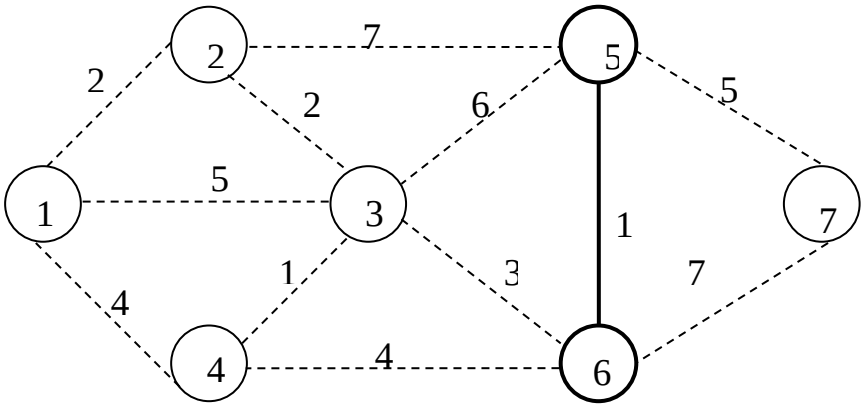


Рис. 2.28. Перший етап побудови мінімального кістякового дерева

Крок 3.

1. Вибираємо вузол 3. Множини матимуть такий вигляд: $s = \{6,5,3\}$, $\bar{s} = \{1,2,4,7\}$. Вартість побудованої мережі p = 1 + 3 = 4 одиниці, її вигляд подано на рис. 2.29.

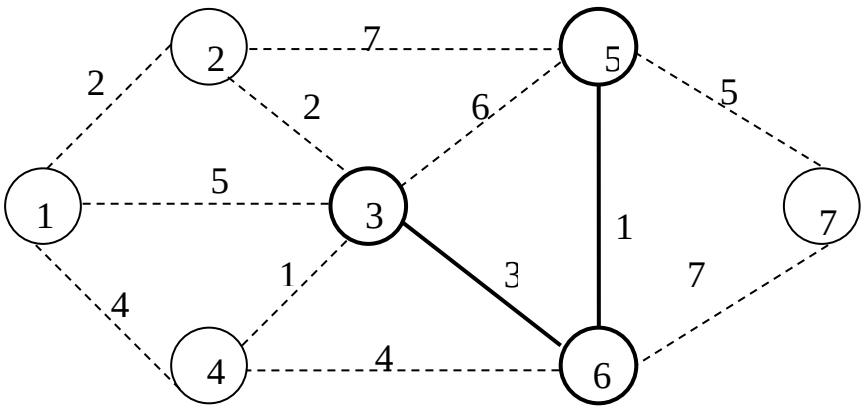


Рис. 2.29. Побудова мінімального кістякового дерева (крок 3, п. 1)

2. Вибираємо вузол 4, тоді $s = \{6,5,3,4\}$, $\bar{s} = \{1,2,7\}$. Вартість побудованої мережі p = 1 + 3 +1 = 5 одиниць (див. рис. 2.30).

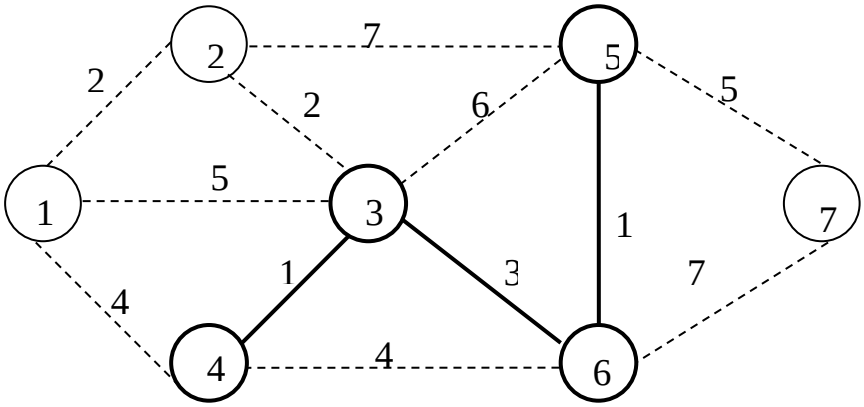


Рис. 2.30. Побудова мінімального кістякового дерева (крок 3, п. 2)

3. Вибираємо вузол 2. У цьому випадку множини $s = \{6, 5, 3, 4, 2\}$, $\overline{s} = \{1, 7\}$. Вартість побудованої мережі p = 1 + 3 +1 + 2 = 7 одиниць (див. рис. 2.21).

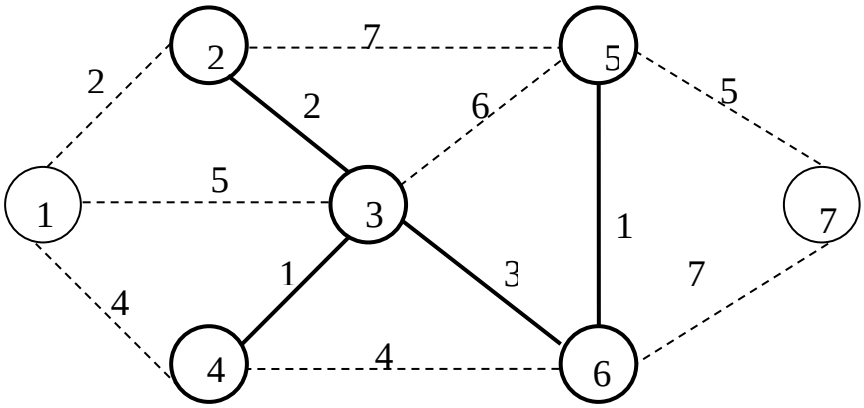


Рис. 2.31. Побудова мінімального кістякового дерева (крок 3, п. 3)

4. Вибираємо вузол 1, тоді $s = \{6, 5, 3, 4, 2, 1\}$, $\overline{s} = \{7\}$. Вартість побудованої мережі p = 1 + 3 +1 + 2 + 2 = 9 одиниць (див. рис. 2.32).

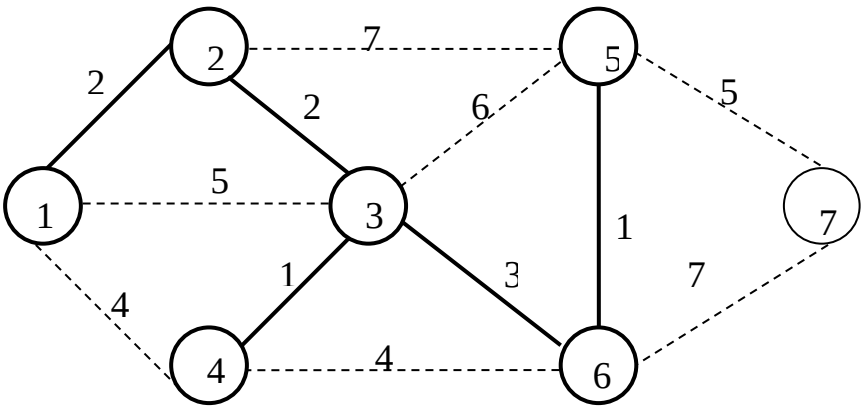


Рис. 2.32. Побудова мінімального кістякового дерева (крок 3, п. 4)

5. Вибираємо вузол 7, тоді $s=\{6,5,3,4,2,1,7\}$, $\overline{s}=\{\varnothing\}$. Вартість побудованої мережі *p* = 1 + 3 +1 + 2 + 2 +5 = 14 одиниць. Його вигляд подано на рис. 2.33.

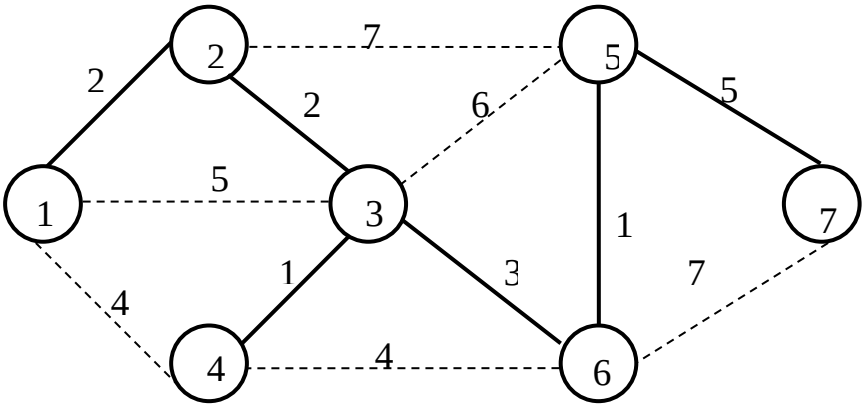


Рис. 2.33. Мінімальне кістякове дерево мережі з прикладу 2.9

Оскільки множина $\overline{s}=\varnothing$, то виконання алгоритму завершено. Вартість мінімального кістякового дерева дорівнює 14 одиниць.

2.9. Задача про максимальний потік у графі

Перш, ніж розглядати постановку задачі й алгоритми її розв'язку, уведемо деякі визначення.

Вершина, з якої починається переміщення об'єктів, називається *джерелом* і зазвичай позначається через *s*.

Вершина, у якій закінчується переміщення об'єктів, називається *стоком* і, як правило, позначається через *t*.

Об'єкти, котрі переміщаються або «плинуть» із джерела в стік, називаються *одиницями потоку*, або просто *одиницями*.

Коли кількість одиниць потоку, які можуть проходити по дузі (x,y), обмежена, то говорять, що дуга (x,y) має обмежену *пропускну здатність*.

Задача про максимальний потік полягає у відшукуванні способу пересилання максимальної кількості одиниць із джерела в стік без перевищення пропускної здатності кожної дуги вихідного графа.

Максимальну величину пропускної здатності будемо позначати через $c(x,y)$.

Кількість одиниць, котрі проходять по дузі (x,y), прийнято називати *потоком* у даній дузі. Позначимо його через $f(x,y)$. Очевидно, що $0\le f(x,y)\le c(x,y)$.

Дуги графа можна віднести до трьох різних категорій, а саме:

– ті, у яких потік не може ні збільшуватися, ні зменшуватися (множина таких дуг позначається через N);

– ті, у яких потік може збільшуватися (множина таких дуг позначається через I);

– ті, у яких потік може зменшуватися (множина таких дуг позначається через R).

Дуги із множини I називаються *збільшувальними*, а дуги із множини R – *зменшувальними*.

Уведемо такі позначення:

$i(x,y)$ – максимальна величина, на яку може бути збільшений потік у дузі (x,y), причому

$$i(x,y)=c(x,y)-f(x,y); \tag{2.6}$$

$r(x,y)$ – максимальна величина, на яку може бути зменшений потік у дузі (x,y), а саме:

$$r(x,y)=f(x,y). \tag{2.7}$$

Переслати додаткову кількість одиниць із джерела в стік можна кількома способами.

1. Існує шлях P від вершини *s* до вершини *t*, який цілком складається із збільшувальних дуг (див. рис. 2.34).

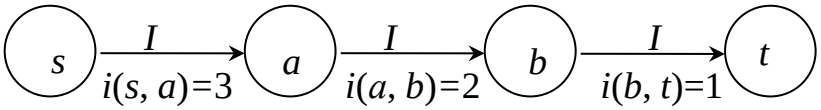


Рис. 2.34. Шлях, що цілком складається із збільшувальних дуг

Оскільки $i(x,y)$ являє собою максимально можливе збільшення потоку в дузі (x,y), то максимальна величина додаткового потоку, що переноситься з вершини *s* у вершину *t* по шляху *P*, буде визначатися величиною $\min\limits_{(x,y)\in P}\{i(x,y)\}$. Наприклад, для шляху, показаного на рис. 2.34. вона буде такою:

$$\min\{i(s,a),i(a,b),i(b,t)\}=\min\{3,2,1\}=1.$$

2. Існує шлях P від вершини *t* до вершини *s* , який цілком складається із зменшувальних дуг (рис. 2.35).

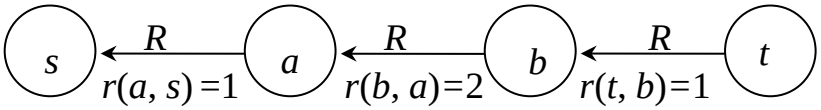


Рис. 2.35. Шлях, що цілком складається із зменшувальних дуг

Через те, що в кожній дузі (x,y) шляху P потік можна зменшити якнайбільше на величину $r(x,y)$, максимальне зменшення потоку вздовж шляху P визначається величиною $\min\limits_{(x,y)\in P}\{r(x,y)\}$.

Для даного прикладу (рис. 2.25) по шляху P можна переслати з вершини s у вершину t максимум одну одиницю потоку, оскільки

$$\min\{r(t,b),r(b,a),r(a,s)\}=\min\{1,2,1\}=1.$$

3. Цей спосіб поєднує в собі два перших. Тут проводиться пошук ланцюга, який з'єднує вершини s і t, і дуги якого задовольняють таким умовам:

– усі ті дуги ланцюга, що мають напрямок від вершини s до вершини t, тобто *прямі*, належать до множини I;

– усі ті дуги ланцюга, що мають напрямок від вершини t до вершини s, тобто *зворотні*, належать множині R.

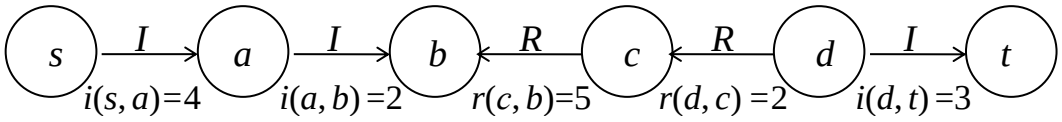


Рис. 2.36. Шлях, який включає збільшувальні й зменшувальні дуги

Переслати додатковий потік з вершини s у вершину t можна за рахунок збільшення потоку в прямих збільшувальних дугах, і зменшення в зворотних зменшувальних дугах.

Максимальна величина додаткового потоку, який можна передати вздовж такого ланцюга з вершини s у вершину t, визначається в такий спосіб:

$$\min\{\min\{i(s,a),i(a,b),i(d,t)\},\min\{r(c,b),r(d,c)\}\}=$$

$$=\min\{\min\{4,2,3\},\min\{2,5\}\}=2.$$

Ця величина називається *максимальним збільшенням потоку* ланцюга.

Усякий ланцюг (кожного із трьох типів, розглянутих вище), що веде від вершини s до вершини t і яким можуть бути додатково надіслані одиниці потоку, називається *збільшувальним*.

2.4.1. Алгоритм пошуку збільшувального ланцюга

Основна ідея алгоритму пошуку збільшувального ланцюга полягає в побудові дерева, котре зростає від вершини s та складене із забарвлених дуг, по

яких із неї можуть передаватися додаткові одиниці потоку. Опишемо цей алгоритм.

Крок 1. Визначаємо склад множин N, I, R. Дуги множини N з подальшого розгляду вилучаються, оскільки в них зміни потоку неможливі. Зафарбовуємо вершину S (джерело).

Крок 2. Зафарбовуємо дуги й вершини відповідно до наведених нижче правил, допоки або не буде пофарбовано вершину t, або забарвлення нових вершин стане неможливим.

Правила забарвлення вершини y й дуги (x,y), коли вершину x вже пофарбовано, можна сформулювати таким чином:

– якщо $(x,y)\in I$, то фарбують вершину y й дугу (x,y);

– коли $(y,x)\in R$, то фарбують вершину y і дугу (y,x);

– в інших випадках забарвлення вершини y й відповідної дуги не відбувається.

Алгоритм описано.

Приклад 2.10. Знайти збільшувальний ланцюг у графі, поданому на рис. 2.37. Поруч із дугами графа зазначено літери R, I та N, які встановлюють їх належність до відповідних множин.

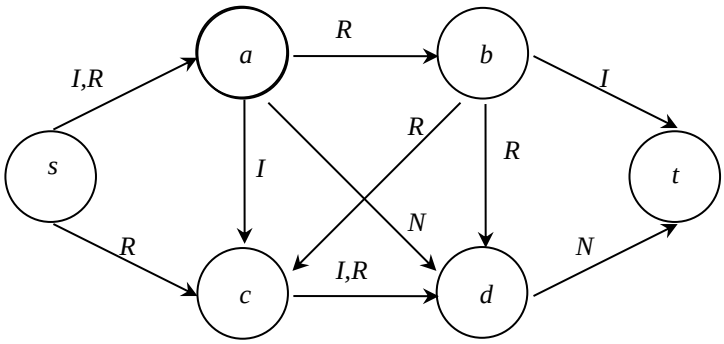


Рис. 2.37. Граф до прикладу 2.10

Розв'язування

Крок 1. Дуги (a,d) й (d,t) належать множині N, тому вони з подальшого розгляду вилучаються.

Крок 2. Насамперед зафарбовується вершина s. Від неї можуть бути пофарбовані вершини a й дуга (s,a), оскільки $(s,a)\in I$. Вершина c й дуга (s,c) не можуть бути пофарбовані від вершини s, тому що $(s,c)\in R$. Отже, поточне дерево забарвлених дуг має вигляд (дуги позначено товстою лінією), показаний на рис. 2.38.

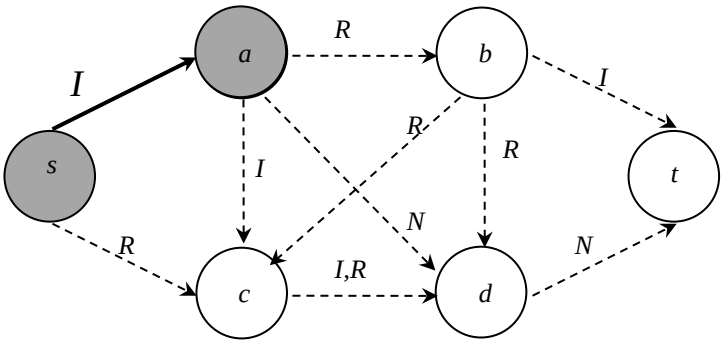


Рис. 2.38. Перший етап побудови збільшувального ланцюга у графі

Далі будемо зафарбовувати вершини й дуги від вершини *a*. Вершина b й дуга (a,b) не можуть бути пофарбовані, оскільки $(a,b)\in R$. Вершина c й дуга (a,c) можуть бути пофарбовані, тому що $(a,c)\in I$. Це завершує процедуру забарвлення елементів графа від вершини *a*.

Поточне дерево пофарбованих дуг набуває вигляду, показаного на рис. 2.39.

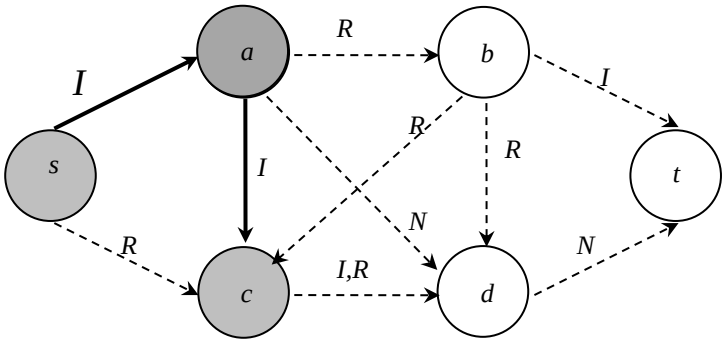


Рис. 2.39. Другий крок побудови збільшувального ланцюга у графі

Далі будемо зафарбовувати вершини й дуги від вершини *c*. Оскільки вершини *s* й *a* уже пофарбовані, їх можна не розглядати.

Вершина b й дуга (b,c) можуть бути пофарбовані, позаяк $(b,c)\in R$. Так само можуть бути пофарбовані d й дуга(c,d), оскільки $(c,d)\in I$. Це завершує процедуру фарбування від вершини *c* й поточне дерево забарвлених дуг має вигляд, наведений на рис. 2.40.

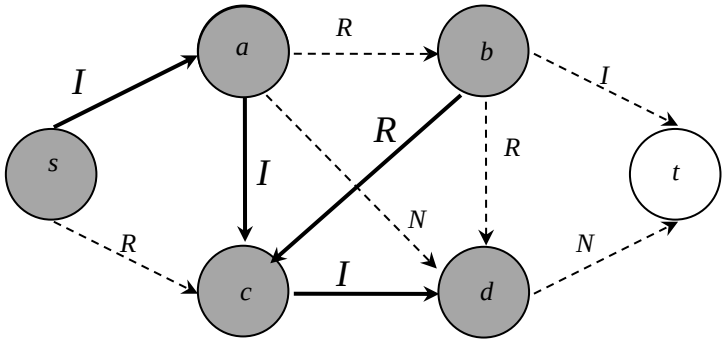


Рис. 2.40. Третій крок побудови збільшувального ланцюга у графі

Далі будемо зафарбовувати вершини й дуги від вершини b. Вершини a,c й d, які вже були забарвлені, можна не розглядати. Вершина t й дуга (b,t) можуть бути пофарбовані, позаяк $(b,t)\in I$.

На цьому процедура забарвлення закінчується, оскільки виявилася пофарбованою вершина *t*.

Дерево, яке складається з пофарбованих вершин і дуг вихідного графа, показано на рис. 2.41.

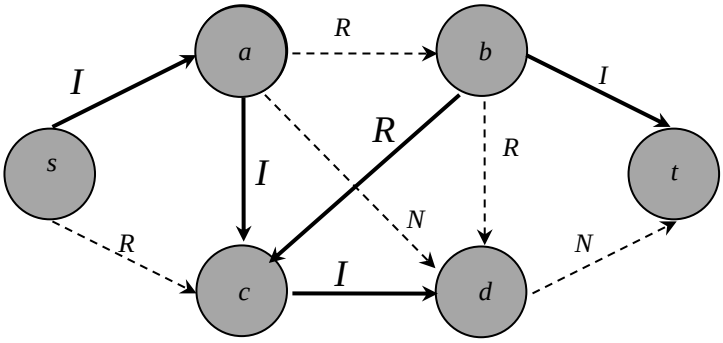
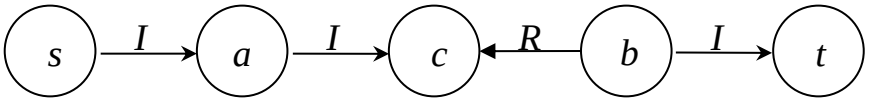


Рис. 2.41. Дерево пофарбованих вершин і дуг з прикладу 2.10

На рис. 2.41. бачимо, що збільшувальним ланцюгом з вершини *s* до вершини *t* є такий:



Нехай $i(s,a)=4,\ \ i(a,c)=3,\ \ r(b,c)=2,\ \ i(b,t)=2$. Тоді максимальне збільшення потоку по цьому ланцюгу має таке значення:

$$\min\{i(s,a),i(a,c),r(b,c),i(b,t)\}=\min\{4,3,2,2\}=2.$$

Таким чином, потоки в прямих дугах знайденого ланцюга, тобто в дугах $(s,a),(a,c)$ і (b,t), можуть бути збільшені на 2 одиниці. Останнє свідчить про те, що зазначені 2 одиниці, які раніше проходили по дузі (b,c), «повертаються» по дузі (b,t), а відповідна «нестача потоку» у вершині *c* компенсується за рахунок двох одиниць, пропущених по дузі (a,c).

Щоразу, коли в процесі виконання розробленого алгоритму визначають збільшувальний ланцюг від вершини s до вершини t, із джерела в стік може бути відправлено додаткову кількість одиниць потоку, і це не перевищує максимальної величини потоку по даному ланцюгу. При цьому в збільшувальних дугах знайденого ланцюга потік зростає, а в зменшувальних дугах спадає на зазначену максимальну величину.

2.4.2. Алгоритм пошуку максимального потоку (Форда й Фалкерсона)

Ідея цього алгоритму полягає в тому, що береться початковий потік від вершини *s* до вершини *t* й за допомогою алгоритму пошуку збільшувального ланцюга виконується його пошук. Якщо він виявляється успішним, то потік уздовж знайденого ланцюга збільшується до максимально можливого значення. Потім здійснюють пошук нового збільшувального ланцюга і т. д. Коли ж на якомусь етапі цієї процедури ланцюг, що збільшує потік, знайти не вдається, то це означає, що потік від вершини *s* до вершини *t* є максимальним.

Опишемо алгоритм пошуку максимального потоку.

Крок 1. Вибрати будь-який початковий потік від вершини *s* (джерело) мережі A до вершини *t* (стік), тобто будь-який набір величин $f(x,y)$, що задовольняє такі співвідношення:

– кількість одиниць, які надходять у вершину x, дорівнює кількості одиниць, котрі виходять з цієї вершини (x – множина вершин мережі A), тобто

$$\sum_{y\in x} f(x,y) - \sum_{y\in x} f(y,x) = 0, \qquad (x \neq s,\ x \neq t);$$

– кількість одиниць потоку, що проходить по дузі, не перевищує пропускну здатність дуги, а саме:

$$0 \le f(x,y) \le c(x,y), \quad (x,y) \in A;$$

– сумарне число одиниць потоку, що виходить із джерела, повинно дорівнювати сумарному числу одиниць (*V*), які потрапляють у стік, тобто

$$\sum_{y\in x} f(s,y) - \sum_{y\in x} f(y,s) = V,$$
$$\sum_{y\in x} f(y,t) - \sum_{y\in x} f(t,y) = V.$$

Якщо величина жодного з початкових потоків від вершини s до вершини t невідома, то можна її задати, зважаючи що для всіх дуг $(x,y)\in A$ $f(x,y)=0$.

Крок 2. Кожну з дуг (x,y) графа віднести до однієї з множин I або R за таким правилом:

Коли $f(x,y)<c(x,y)$, то вважати, що $i(x,y)=c(x,y)-f(x,y)$, а дуга (x,y) належить множині I.

Коли $f(x,y)>0$, то вважати, що $r(x,y)=f(x,y)$, а дуга (x,y) належить множині R.

Крок 3. На множинах I і R, сформованих у попередньому кроці, застосувати до вихідного графа алгоритм пошуку збільшувального ланцюга. Якщо при цьому його знайти не вдається, то закінчити процедуру виконання алгоритму: поточний потік є максимальним. А якщо ні, то здійснити максимально можливе збільшення потоку вздовж знайденого збільшувального ланцюга і повернутися до кроку 2.

Алгоритм описано.

Приклад 2.11. Застосуємо алгоритм пошуку максимального потоку до графа, зображеного на рис. 2.42. Величини $c(x,y)$ позначено біля дуг.

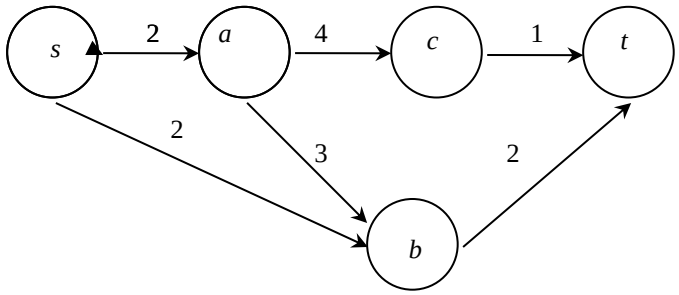


Рис. 2.42. Граф до прикладу 2.11

Розв'язування

Крок 1. Виконання алгоритму починається із задання нульового потоку, тобто для всіх дуг (x,y) графа вважаємо, що $f(x,y)=0$.

Крок 2. Оскільки для всіх дуг виконується умова, що $f(x,y)<c(x,y)$, то всі вони можуть будуть віднесені до множини I. При цьому $i(x,y)=c(x,y)-f(x,y)=c(x,y)$. Крім того, враховуючи, що для всіх дуг $f(x,y)=0$, то на кожному кроці жодну з них не можна віднести до множини R, тобто маємо граф, зображений на рис. 2.43.

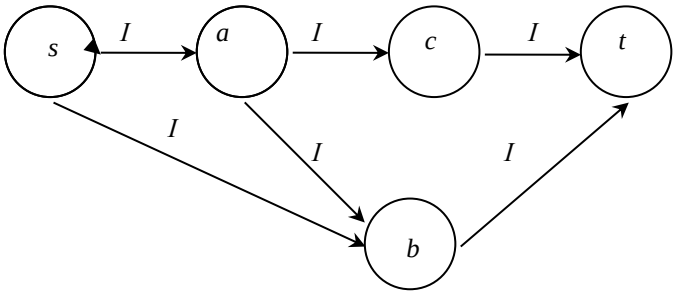


Рис. 2.43. Перший етап пошуку максимального потоку (приклад 2.11)

При цьому $i(s,a)=c(s,a)=2;$

$i(a,c)=c(a,c)=4;$

$i(c,t)=c(c,t)=1;$

$i(s,b)=c(s,b)=2;$

$i(b,t)=c(b,t)=2;$

$i(a,b)=c(a,b)=3.$

Крок 3. Скористаємося алгоритмом пошуку збільшувального ланцюга від вершини s до вершини t, який було описано вище.

На даному етапі виконання алгоритму в графі існує кілька таких ланцюгів (оскільки початковий потік нульовий). Їх показано на рис. 2.44.

Розглянемо один з них, а саме, ланцюг $(s,a),(a,b),(b,t)$, зображений на рис. 2.44, *б*.

Максимальне збільшення потоку на цьому ланцюзі обчислюється таким чином:

$$\min\{i(s,a),i(a,b),i(b,t)\}=\min\{2,3,2\}=2.$$

Отже, потік у кожній із трьох його дуг збільшується на дві одиниці, в інших дугах потік не змінюється, тобто

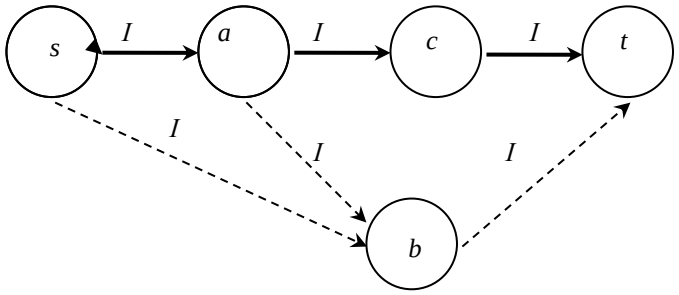
$$f(s,a)=2,\quad f(a,b)=2,\quad f(b,t)=2,\quad f(s,b)=0,\quad f(a,c)=0,\quad f(c,t)=0.$$

Крок 2. Обчислюємо знову величини $i(x,y)$ й $r(x,y)$ за формулами (2.6), (2.7) і коректуємо склад множин I і R, а саме:

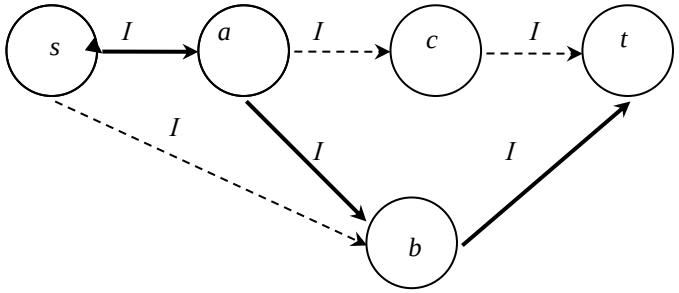
$$i(x,y)=c(x,y)-f(x,y);$$

$$r(x,y)=f(x,y).$$

а



б



в

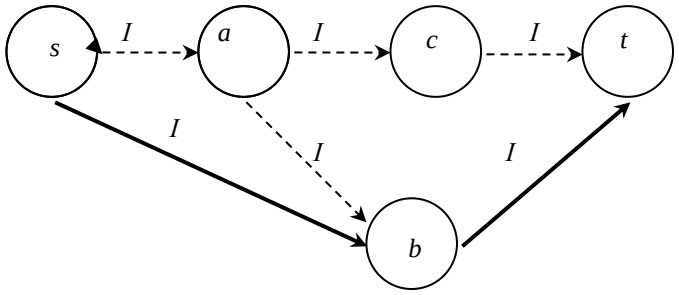


Рис. 2.44. Результат застосування алгоритму пошуку збільшувального ланцюга для графа з прикладу 2.11 (перший прохід алгоритму)

Оскільки $f(s,a)=2=c(s,a)$, а отже $i(s,a)=0$ і тому дуга $(s,a)\notin I$, $r(s,a)=2$, це означає, що дуга $(s,a)\in R$.

Через те, що $f(a,b)=2<c(a,b)=3$, дуга $(a,b)\in I$, а значення $i(a,b)=1$, крім того $(a,b)\in R$, а значення $r(a,b)=2$.

Аналогічно для дуги (b,t) отримуємо такі результати:

$f(b,t)=2=c(b,t)$, тому $(b,t)\notin I$, оскільки $r(b,t)=2$, то $(b,t)\in R$.

Дуги: $(s,b),(a,c)(c,t)$, як і раніше, належать до множини I.

Зазначимо символи I й R над відповідними дугами графа (див. рис. 2.45).

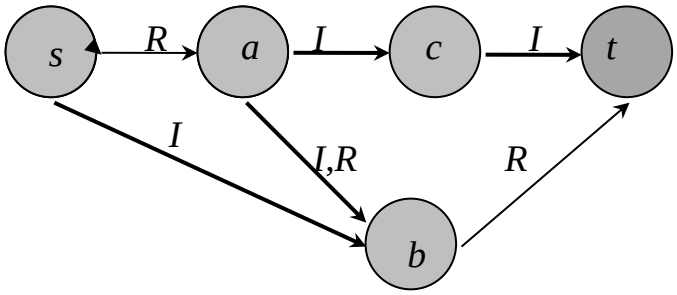


Рис. 2.45. Ілюстрація до розв'язування прикладу 2.11

Крок 3. Знову скористаємося алгоритмом пошуку збільшувального ланцюга від вершини *s* до вершини *t*. У цьому випадку (див. рис. 2.45) такий ланцюг єдиний. Його зображено на рис. 2.46

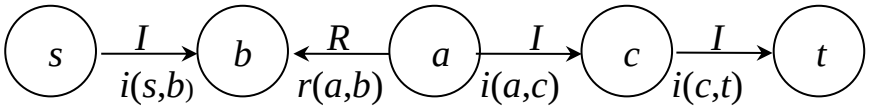


Рис. 2.46. Збільшувальний ланцюг до прикладу 2.11 (другий прохід алгоритму)

Максимальне збільшення потоку вздовж ланцюга, поданого на рис. 2.45, набуває такого значення:

$$\min\{i(s,b),r(a,b),i(a,c),i(c,t)\}=\min\{3,2,4,1\}=1.$$

Таким чином, від вершини s до вершини t може бути додатково пропущено одну одиницю потоку. При цьому величина потоку в кожній із трьох прямих дуг $(s,b),(a,c)(c,t)$, що належать знайденому ланцюгу, на одиницю збільшиться, а значення потоку в зворотній дузі (a,b) на одиницю зменшиться.

Отже, тепер потоки в дугах розглянутого графа будуть мати такі значення:

$$f(s,a)=2,\quad f(a,b)=2,\quad f(b,t)=2,$$

$$f(s,b)=1,\quad f(a,c)=1,\quad f(c,t)=1.$$

Кожна із трьох одиниць побудованого на даний момент потоку проходить такими маршрутами:

– перша одиниця – від вершини *s* до вершини *t* таким шляхом: $(s,a),(a,b)$ і (b,t);

– друга одиниця – від вершини s до вершини t по такому шляху: $(s,b),(b,t)$;

– третя одиниця – від вершини s до вершини t по такому шляху: $(s,a),(a,c)$ і (c,t).

Крок 2. Обчислюємо знову величини $i(x,y)$ й $r(x,y)$ та коректуємо склад множин I і R, а саме:

$$f(a,b)=1<c(a,b)=3,\ (a,b)\in I,\ i(a,b)=2;\ (a,b)\in R,\ r(a,b)=1;$$
$$f(s,b)=1<c(s,b)=3,\ (s,b)\in I,\ i(s,b)=2;\ (s,b)\in R,\ r(s,b)=1;$$
$$f(a,c)=1<c(a,c)=4,\ (a,c)\in I,\ i(a,c)=3;\ (a,c)\in R,\ r(a,c)=1;$$
$$f(c,t)=1<c(c,t)=1,\ (c,t)\notin I,\ r(c,t)=1;\ (c,t)\in R.$$

Висновки стосовно дуг (s,a) і (b,t) зроблено вище.

Проставимо на графі символи I й R, які отримано для інших дуг (рис. 2.47).

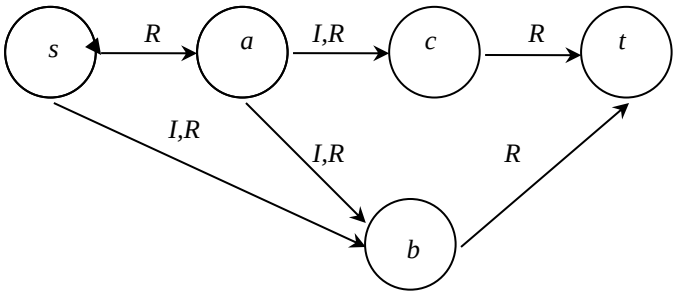


Рис. 2.47. Пошук збільшувального ланцюга в прикладі 5 (третій прохід алгоритму)

Крок 3. Знову скористаємося алгоритмом пошуку збільшувального ланцюга, від вершини s до вершини t.

Для цього проаналізуємо граф, поданий на рис. 2.47. Виявляється, що з огляду на знайдені значення потоку в дугах графа він не має жодного збільшувального ланцюга. Під час виконання алгоритму у вихідному графі (рис. 2.47.) виявляються забарвленими вершини s, b, a і c (у зазначеному порядку), однак вершина t не зафарбовується. Отже, оскільки збільшувального ланцюга тут не існує, то виконання алгоритму пошуку максимального потоку завершується.

Останній із побудованих потоків є максимальним, бо від вершини s до вершини t можна пропустити 3 одиниці потоку.

2.10. Розрахунок мережевого графіка

На теорії графів базується математичний апарат мережевих моделей.

Мережеве планування (network planning) – метод наукового планування та управління виробничими процесами, які виконують великі обсяги робіт.

Методи мережевого планування знаходять широке застосування в багатьох галузях виробництва. Всякий запланований комплекс робіт, необхідних для досягнення деякої мети, називають проєктом. Так, можна, наприклад, говорити про будівництво нового будинку, модернізацію устаткування, створення нового пристрою.

Мережеве планування має ряд переваг: забезпечує наочність технологічної послідовності робіт; дозволяє скласти оперативні та поточні плани, а також прогнозувати складні процеси; дозволяє виявити приховані ресурси часу і матеріальних засобів при виконанні виробничих процесів

Мережева модель (мережевий графік, мережа) являє собою економіко-математичну модель, яка відображає комплекс робіт (операцій) і подій, пов'язаних з реалізацією певного проєкту (науково-дослідного, виробничого та інших), в їх логічній послідовності і зв'язку.

Аналіз мережевих моделей, поданих в графічній чи табличній (матричній) формі, дозволяє, по-перше, більш чітко виявити взаємозв'язки етапів реалізації проєкту і, по-друге, визначити оптимальний порядок виконання всіх етапів. Оптимальність визначається залежно від мети моделювання, наприклад, метою може бути скорочення термінів виконання всього комплексу робіт.

Методи мережевого планування і управління забезпечують:

• складання календарного плану виконання певного комплекту робіт;

• оцінку необхідних трудових, матеріальних і фінансових ресурсів, затрат часу;

• контроль комплексу робіт з прогнозуванням і запобіганням можливих зривів при виконанні робіт;

• ефективне управління при чіткому розподілі відповідальності між керівниками різних рівнів і виконавцями робіт;

• оцінку дієздатності та якості системи стосовно певних критеріїв.

Математичний апарат мережевих моделей базується на теорії графів.

Граф, деякі вершини якого виділені, *називається мережею.* Виділені вершини називаються *полюсами мережі.* Наприклад, дерево з коренем можна розглядати, як однополюсну мережу.

Отже, мережева модель – це граф спеціального вигляду, а саме *мережа*.

Ізоморфізмом мереж називається ізоморфне відображення їхніх графів, при якому полюси обов'язково переходять у полюси (іноді задають умову, щоб певні полюси переходили у визначені).

Вершини, відмінні від полюсів, називаються *внутрішніми вершинами мережі*.

Ребро, інцидентне хоча б одному полюсу, називається *полюсним*. Інші ребра називаються *внутрішніми*.

Розглянемо використання положень теорії графів при побудові мережі складного комплексу робіт або операцій.

Вихідними даними мережевого планування (моделювання) того чи іншого виробничого процесу є перелік і тривалість здійснення операцій, які виконуються під час нього, інформація про їх технологічну послідовність, тривалість і потрібні для їх виконання ресурси.

Визначимо основні терміни, що використовуються при мережевому плануванні. Перш за все, це *робота* і *подія*. *Робота* в плані – це певна діяльність, яка необхідна для досягнення конкретних результатів. *Подія* – це кінцевий результат попередніх робіт, тобто вона фіксує факт виконання роботи. При цьому терміни виконання всього проєкту задано і відомо тривалість виконання окремих робіт або операцій, що входять до його складу. Кожна така операція починається і закінчується певними подіями. Зазвичай вихідною подією є початок роботи, а кінцевою стосовно неї подією є її закінчення, і можливо, початок нової.

Робота характеризує матеріальну дію, яка потребує використання ресурсів або ні. При графічному поданні робота зображується стрілкою, яка з'єднує дві події. В таблиці вона позначається парою розміщених в дужках чисел (i, j), де i – номер події, з якої робота виходить (починається), а j – номер події, в яку вона входить (закінчується). Ці ж числа можуть бути показані і як нижній індекс. Робота не може початися раніше, ніж відбудеться подія, з якої вона виходить. Кожна робота має певну тривалість $t(i, j) = t_{ij}$, яка на графі зазвичай подається цифрою, що записана над стрілкою. Наприклад, запис $t(2,5) = t_{2,5} = 4$ означає, що робота (2,5) має тривалість 4 одиниці.

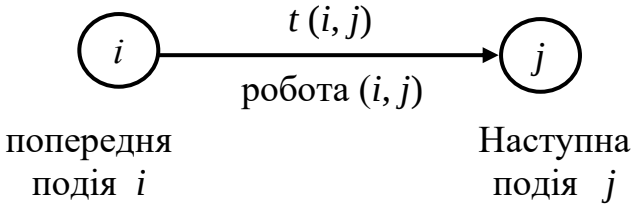


Рис. 2. 48. Кодування роботи на графі

До робіт також відносяться такі процеси, які не вимагають ні ресурсів, ні часу виконання. Вони полягають у встановленні логічного взаємозв'язку робіт і показують, що одна з них безпосередньо залежить від інших, тобто її здійснення не може початися, доки не буде закінчено виконання певної іншої події; їх називають *фіктивними роботами* і на графіку зображають пунктирними стрілками

Подіями називаються результати виконання однієї або декількох робіт. Вони не мають протяжності в часі. Подія здійснюється в той момент, коли закінчується остання з робіт, які входять в неї. Вона позначається одним числом і при графічному поданні мережевої моделі зображується колом (або іншою геометричною фігурою), всередині якого проставляється його порядковий номер (i = 1, 2, ..., N). В мережевій моделі обов'язково мають бути початкова подія (з

номером 1), з якої роботи тільки виходять, і кінцева подія (з номером *N*), в яку роботи тільки входять.

Шлях – це ланцюг робіт, які виконуються одна за одною і з'єднують початкову і кінцеву вершини. Тривалість шляху визначається сумою тривалостей складаючих його робіт. Шлях, що має максимальну довжину, називають *критичним* і *позначають* $L_{Кр}$, а його тривалість – $t_{Кр}$.

Роботи, котрі належать до критичного шляху, називаються *критичними*. Їх невчасне виконання веде до зриву термінів закінчення всього комплексу робіт.

Мережевий графік — це динамічна модель виробничого процесу, яка відображає технологічну залежність і послідовність виконання комплексу робіт, погоджує їх здійснення в часі з урахуванням витрат ресурсів і вартості, а також виділенням вузьких (критичних) місць.

Мережеве планування супроводжується побудовою робочих таблиць і мережевих графіків. Виходячи з розподілу процесу реалізації проєкту на роботи і терміни їх виконання, будують відповідний мережевий граф. Потім виконують його аналіз та оптимізацію за обраними критеріями.

Проте перед розрахунком мережевої моделі слід переконатися, що вона задовольняє наступним основним вимогам.

1. Події правильно пронумеровані, якщо для кожної роботи (i, j) виконується умова $i < j$.

При невиконанні цієї вимоги необхідно використовувати алгоритм перенумерації подій, який полягає в наступному:

– нумерація починається з вихідної події, якій привласнюється № 1;

– з вихідної події викреслюють стрілками всі роботи, що виходять з неї, і серед них знаходимо подію, в яку не входить жодна робота, їй і привласнюють № 2;

– потім викреслюють стрілками роботи, що виходять з події № 2, і знов знаходять подію, в яку не входить жодна робота, і їй привласнюють № 3,

так триває до завершальної події, номер якої має дорівнювати кількості подій в мережевому графіку;

– якщо при черговому викреслюванні робіт одночасно кілька подій не мають робіт, що входять в них, то їх нумерують по черзі у довільному порядку.

2. Відсутні тупикові ("глухі") події (окрім завершальної), тобто такі, за якими не слідує хоча б одна робота;

3. Відсутні події (за винятком початкової), яким не передує хоча б одна робота ("хвостові" події);

4. Відсутні цикли (контури, петлі), тобто замкнуті шляхи, які з'єднують певні події з ними ж самими;

5. Дві довільні події мають бути безпосередньо пов'язані не більше ніж однім ребром (дугою).

6. На графі має бути лише одна вихідна та лише одна завершальна подія.

Якщо це не так (наприклад, початок реалізації комплексу робіт можна розпочинати паралельно з декількох робіт), то необхідно ввести фіктивні події та роботи.

При невиконанні вказаних вимог не має сенсу приступати до обчислень характеристик подій, робіт, визначення критичного шляху і оптимізації.

Для подій розраховують такі характеристики: ранній і пізній термін здійснення події, а також її резерв.

Ранній термін настання події t_p визначається величиною найбільш тривалого відрізка шляху від початкової до даної події, причому $t_p\,(1) = 0$, а $t_p\,(N) = t_{кр}(L)$:

$$t_P(j) = \begin{cases} t_P(j) + t(i,j), & \text{якщо до події j підходить одна операція,} \\ \max\limits_i\{t_P(i) + t(i,j)\}, & \text{якщо до події j підходить декілька } \{i\} \text{ операцій,} \end{cases} \quad (2.8)$$

$$j = \overline{2,N}\ .$$

Пізній термін настання події t_n характеризує найпізніший допустимий термін, до якого подія має бути виконана, не викликаючи при цьому зриву терміну виконання кінцевої події:

$$t_{\text{п}}(j) = \begin{cases} t_{\text{п}}(j) - t(i,j), & \text{якщо від події j відходить одна операція,} \\ \min\limits_i\{t_{\text{п}}(j) - t(i,j)\}, & \text{якщо від події j відходить декілька } \{i\} \text{ операцій,} \end{cases} \quad (2.9)$$

$$i = \overline{2,N-1}\ .$$

Цей показник визначається “зворотним ходом”, починаючи із завершальної події, з урахуванням співвідношення $t_n(N) = t_p(N)$.

Всі події, за винятком тих, які належать критичному шляху, мають *резерв часу* $R(i)$:

$$R(i) = t_n(i) - t_p(i). \quad (2.10)$$

Резерв показує, на який гранично допустимий термін можна затримати настання цієї події, не викликаючи при цьому збільшення терміну виконання всього комплексу робіт.

Для всіх робіт (i, j) на основі ранніх і пізніх термінів виконання всіх подій можна визначити таки показники:

Ранній термін початку $t_{рп}\,(i, j) = t_p(i)$; (2.11)

Ранній термін закінчення $t_{рз}\,(i, j) = t_p(i) + t(i, j)$; (2.12)

Пізній термін закінчення $t_{пз}\,(i, j) = t_n(j)$; (2.13)

Пізній термін початку $t_{nn}(i, j) = t_n(j) - t(i, j)$; (2.14)

Резерв часу операції (i, j) $R(i, j) = t_n(j) - t_p(i)$; (2.15)

Повний резерв часу $R_n(i, j) = t_n(j) - t_p(i) - t(i, j)$; (2.16)

Незалежний резерв $R_н(i, j) = \max\{0;\ t_p(j) - t_n(i) - \mathrm{t}(i, j)\} =$

$$= \max\{0; R_n(i, j) - R(i) - R(j)\}. \quad (2.17)$$

Повний резерв часу операції R_n показує, на скільки можна збільшити час виконання конкретної роботи за умови, що термін здійснення всього комплексу робіт не зміниться. Це максимальний час, на який можна відкласти або збільшити тривалість роботи (i , j), не змінюючи директивного або раннього терміну настання завершальної події; R_n має мінімальні значення для операцій, які перебувають на критичному шляху; вони дорівнюють нулю, якщо директивний термін настання завершальної події не заданий або перевищує початок виконання операцій на час, який дорівнює тривалості критичного шляху. Повний резерв часу роботи стосується не лише неї, а й усіх повних шляхів, які її містять. Якщо використати повний резерв часу тільки для цієї однієї роботи, то резерви часу решти робіт даного шляху зменшуватимуться на величину використаного резерву (або будуть дорівнювати 0, якщо було використано максимальний резерв для даного шляху).

Незалежний резерв часу відповідає випадку, коли всі попередні роботи закінчуються в пізні терміни, а всі подальші починаються в ранні терміни. Використовування цього резерву не впливає на величину резервів часу інших робіт.

Шлях характеризується двома показниками – тривалістю і резервом.

Тривалість шляху визначається сумою тривалостей робіт, які входять до нього.

Резерв визначають як різницю між довжинами критичного і даного шляхів. З цього визначення виходить, що роботи, які лежать на критичному шляху, і сам критичний шлях мають нульовий резерв часу (тобто не мають резерву часу), оскільки будь-яка затримка із завершенням подій, розташованих на критичному шляху, приведе до такої самої затримки у виконанні завершальної події. Отже, *критичний шлях* L_{kp} можна визначити як послідовність операцій із найменшим повним резервом часу.

Резерв часу шляху показує, на скільки може збільшитися тривалість робіт, що його складають, без зміни тривалості загального терміну виконання всіх робіт. Відмінний від нуля (ненульовий) резерв часу певної події означає, що термін її настання може бути збільшений на величину її резерву часу без затримки терміну виконання всього комплексу робіт.

Для оптимізації мережевої моделі, яка полягає в перерозподілі ресурсів з ненапружених робіт на критичні для прискорення їх виконання, необхідно якомога більш точно оцінити степінь важкості своєчасного виконання всіх робіт, а також “ланцюгів” шляху. Більш точним інструментом розв’язування цієї задачі

в порівнянні з повним резервом є коефіцієнт напруженості, який може бути обчислений одним з двох способів за наведеними нижче формулами:

$$K_{\text{н}}(i,j) = \frac{t(L_{max}) - t_{\text{кр}}}{t_{\text{кр}} - t'_{\text{кр}}},$$

$$K_{\text{н}}(i,j) = 1 - \frac{R_{\text{п}}}{t_{\text{кр}} - t'_{\text{кр}}},$$

де $t(L_{max})$ – тривалість максимального шляху, що проходить через роботу (i, j); $t_{\text{кр}}$ – тривалість критичного шляху; $t'_{\text{кр}}$ – тривалість відрізка даного шляху, що співпадає із критичним шляхом (максимальне співпадіння).

Коефіцієнт напруженості змінюється від нуля до одиниці, причому чим він ближчий до одиниці, тим складніше виконати дану роботу у встановлений термін. За коефіцієнтом напруженості всі роботи можуть бути поділені на кілька груп (див. табл. 2.2). Самими напруженими є роботи критичного шляху, для яких він дорівнює 1. В результаті перерозподілу ресурсів прагнуть максимально зменшити загальну тривалість робіт, що можливе при переведенні всіх робіт в першу групу.

Таблиця 2.2

Класифікація робіт за коефіцієнтом напруженості

Група робіт	Коефіцієнт напруженості
Критичні	$K_{\text{н}} = 1$
Напружені	$K_{\text{н}} > 0{,}8$
Підкритичні	$0{,}6 < K_{\text{н}} \leq 0{,}8$
Резервні	$K_{\text{н}} \leq 0{,}6$

Розрахунок часових параметрів t_{po} і t_p ведеться від початку мережевого графіка до кінця, а розрахунок t_n і $t_{nз}$ – від кінця до початку. При цьому для кінцевої події $t_p = t_n$.

Зауважимо, що під час розрахунку часових параметрів мережевих графіків із детермінованим часом виконання операцій не враховуються випадкові зміни їх тривалості, а це може істотно впливати на термін завершення всього комплексу операцій. На мережевому графіку часові параметри відображаються, як показано на рис. 2.49.

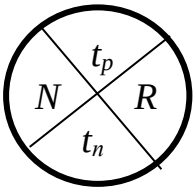


Рис. 2.49. Відображення часових параметрів подій в вершинах мережевого графіка

Приклад 2.12. У табл. 2.3 записано роботи (i, j) і час їхнього виконання t_{ij}.

Таблиця 2.3

Вихідні дані до прикладу 2.12

i , j	1, 2	1, 3	2, 3	2, 5	3, 4	3, 6	4, 5	4, 6	4, 7	5, 7	6,7
t_{ij}	2	8	5	4	7	23	12	4	5	10	8

Необхідно побудувати мережевий графік виконання робіт і визначити його параметри за подіями і роботами.

Розв'язування

За даними про роботи i, j будуємо мережевий графік. За умовами завдання ми маємо 7 подій, отже у графі буде 7 вершин. При зображенні графа необхідно так розташувати вершини, аби роботи (i, j) не перетиналися. Отриманий мережевий граф показано на рис. 2.50.

Перевіримо, чи задовольняє він вимогам 1 – 6. Нумерацію подій здійснено правильно (згідно із даними таблиці). Граф має одну вхідну і одну вихідну вершину, цикли відсутні, відтак, граф побудовано правильно і можна приступати до розрахунків.

Обчислимо параметри кожної з подій.

1) Ранній термін настання події i, $t_р(i)$. Це максимальний шлях від початкової події до i-ї події, він обчислюється за формулою (2.8):

$$t_p(1) = 0; \quad t_p(2) = t(1, 2) = 2.$$

У третю подію входять дві роботи: (2, 3) і (1, 3), значить,

$$t_p(3) = \max\{ t_p(2) + t(2, 3) ; t_p(1) + t(1, 3)\} = \max\{2+5, 8\} = 8.$$

У четверту подію входить одна робота – (3, 4), тому

$$t_p(4) = t_p(3) + t(3, 4) = 8 + 7 = 15.$$

.

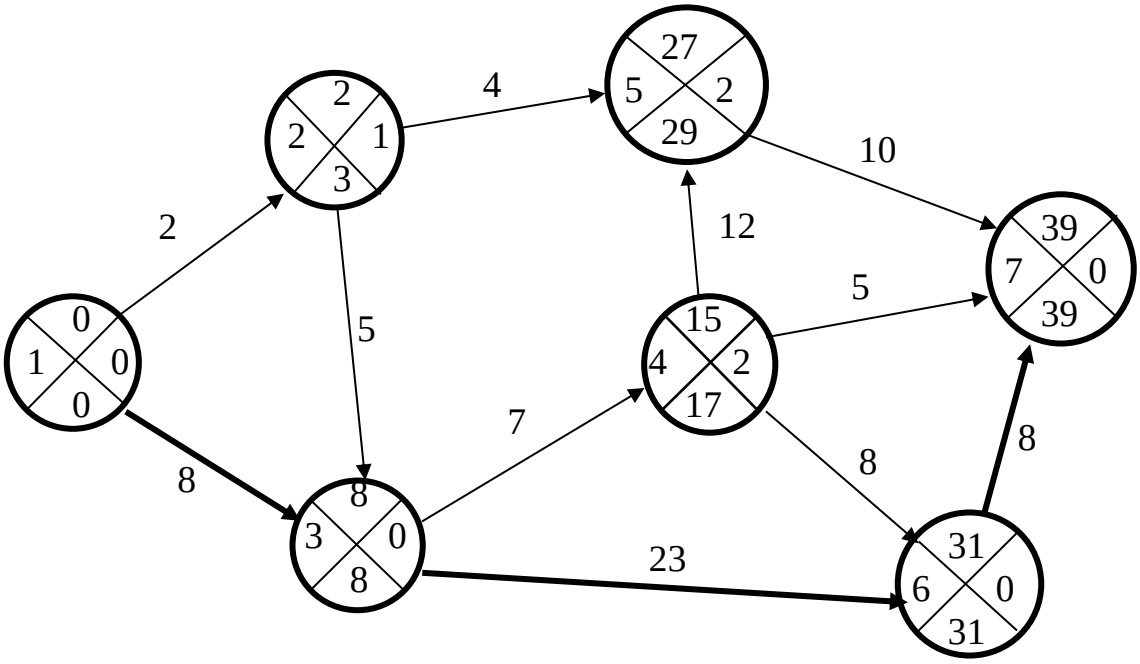


Рис. 2.50. Мережевий графік

У п’яту подію входять дві роботи: (2, 5) і (4, 5), отже,

$t_p(5) = \max\{t_p(2) + t(2, 5); t_p(4) + t(4, 5)\} = \max\{2+4, 15+12\} = 27.$

У шосту подію входять дві роботи: (4, 6) і (3, 6), відтак,

$t_p(6) = \max\{t_p(4) + t(2, 5); t_p(3) + t(3, 6)\} = \max\{15+4, 8+23\} = 31.$

У сьому подію входять три роботи: (5, 7); (4, 7); (6, 7), значить

$t_p(7) = \max\{t_p(5) + t(5, 7); t_p(4) + t(4, 7); t_p(6) + t(6, 7)\} =$

$= \max\{5+5, 27+10, 31+8\} = 39.$

2) Пізній термін настання події i, $t_n(i)$ – це різниця між тривалістю максимального шляху $L_{\max}$ і шляху найбільшої тривалості від даної події і до кінцевої.

Він розраховується за формулою (2.9) і оберненою схемою. Тобто розрахунок починаємо від кінцевої події, орієнтуємося на вихідні роботи, і обчислюємо мінімум різниці.

Для кінцевої події

$t_n(7) = t_p(7) = 39.$

З шостої події виходить одна робота: (6, 7), тому

$t_n(6) = t_n(7) - t(6, 7) = 39 - 8 = 31.$

З п’ятої події також виходить одна робота: (5, 7)

$t_n(5) = t_n(7) - t(5, 7) = 39 - 10 = 29.$

З четвертої події виходить три роботи: (4, 5), (4, 6), (4, 7)

$t_n(4) = \min\{ t_n(5) - t(4, 5) ; t_n(6) - t(4, 6) ; t_n(7) - t(4, 7) \} =$

$= \min\{29 - 12; 31 - 4; 39 - 5\} = 17.$

З третьої події виходить дві роботи: (3, 4), (3, 6)

$t_n(3) = \min\{ t_n(4) - t(3, 4) ; t_n(6) - t(3, 6) \} = \min\{17 - 7; 31 - 23\} = 8.$

З другої події виходить 2 роботи: (2, 5); (2 , 3),

$t_n(2) = \min\{ t_n(5) - t(2, 5) ; t_n(3) - t(2, 3) \} = \min\{8 - 5; 29 - 4\} = 3.$

З початкової події виходить 2 роботи: (1, 2); (1, 3),

$t_n(1) = \min\{ t_n(2) - t(1, 2) ; t_n(3) - t(1, 3) \} = \min\{3 - 2; 8 - 8\} = 0.$

Для початкової події має виконуватися умова:

$$t_p(1) = t_n(1) = 0 .$$

3) За формулою (2.10) обчислюємо резерв часу для кожної з подій.

$$R(i) = t_n(i) - t_p(i).$$

$R(1) = 0;$ $R(5) = 29 - 27 = 2;$

$R(2) = 3 - 2 = 1;$ $R(6) = 31 - 31 = 0;$

$R(3) = 8 - 8 = 0;$ $R(7) = 39 - 39 = 0.$

$R(4) = 17 - 15 = 2;$

4) Критичний шлях проходить по подіях із нульовим резервом часу ($R(i) = 0$), тобто, це події 1, 3, 6, 7 (виділено на графі, рис. 2.50). Довжина критичного шляху $L_{кр}$ – це самий довгий шлях від початкової події до кінцевої:

$$L_{кр} = t_p(7) = 39.$$

Тепер обчислимо часові параметри стосовно робіт.

5) Ранній термін закінчення роботи (i, j), обчислюємо за формулою (2.11):

$$t_{рз}(i, j) = t_p(i) + t(i, j).$$

$t_{рз}(1, 2) = t_p(1) + t(1, 2) = 0+2 = 2;$

$t_{рз}(1,3) = t_p(1) + t(1,3) = 0+8 = 8;$

$t_{рз}(2, 3) = t_р(2) + t(2, 3) = 2+5 = 7;$

$t_{рз}(2, 5) = t_р(2) + t(2, 5) = 2+4 = 6;$

$t_{рз}(3, 4) = t_р(3) + t(3, 4) = 8+7 = 15;$

$t_{рз}(3, 6) = t_р(3) + t(3, 6) = 8+23 = 31;$

$t_{рз}(4, 5) = t_р(4) + t(4, 5) = 15+12 = 27;$

$t_{рз}(4, 6) = t_р(4) + t(4, 6) = 15+4 = 19;$

$t_{рз}(4, 7) = t_р(4) + t(4, 7) = 15+5 = 20;$

$t_{рз}(5, 7) = t_р(5) + t(5, 7) = 27+10 = 37;$

$t_{рз}(6, 7) = t_р(6) + t(6, 7) = 31+8 = 39.$

6) Пізній термін закінчення роботи (i, j), обчислюємо за формулою (2.13):

$t_{пз}(1,2) = t_п(2) = 3;$ $t_{пз}(2,3) = t_п(3) = 8;$

$t_{пз}(1,3) = t_п(3) = 8;$ $t_{пз}(2,5) = t_п(5) = 29;$

$t_{пз}(3,4) = t_п(4) = 17;$ $t_{пз}(4,5) = t_п(5) = 29;$

$t_{пз}(3,6) = t_п(6) = 31;$ $t_{пз}(4,6) = t_п(6) = 31;$

$t_{пз}(5,7) = t_п(7) = 39;$ $t_{пз}(4,7) = t_п(7) = 39.$

$t_{пз}(6,7) = t_п(7) = 39.$

7) Повний резерв часу роботи (i, j) – це час, на який можна збільшити її тривалість, не змінюючи при цьому тривалість критичного шляху $L_{кр}$. Його обчислюємо за формулою (2.16).

$R_п(i, j) = t_п(j) - t_р(i) - t(i, j);$

$R_п(1, 2) = t_п(2) - t_р(1) - t(1, 2) = 3 - 0 - 2 = 1;$

$R_п(1, 3) = t_п(3) - t_р(1) - t(1, 3) = 8 - 0 - 8 = 0;$

$R_п(2, 3) = t_п(3) - t_р(2) - t(2, 3) = 8 - 2 - 5 = 1;$

$R_п(2, 5) = t_п(5) - t_р(2) - t(2, 5) = 29 - 2 - 4 = 23;$

$R_п(3, 4) = t_п(4) - t_р(3) - t(3, 4) = 17 - 8 - 7 = 2;$

$R_п(3, 6) = t_п(6) - t_р(3) - t(3, 6) = 31 - 8 - 23 = 0;$

$R_п(4, 5) = t_п(5) - t_р(4) - t(4, 5) = 29 - 15 - 12 = 2;$

$R_п(4, 6) = t_п(6) - t_р(4) - t(4, 6) = 31 - 15 - 8 = 8;$

$R_п(4, 7) = t_п(7) - t_р(4) - t(4, 7) = 39 - 15 - 5 = 19;$

$R_п(5, 7) = t_п(7) - t_р(5) - t(5, 7) = 39 - 27 - 10 = 2;$

$R_п(6, 7) = t_п(7) - t_р(6) - t(6, 7) = 39 - 31 - 8 = 0.$

8) Розрахуємо коефіцієнти напруженості для кожної роботи.

$$K_{\text{н}}(1,2) = 1 - \frac{1}{39} = 0{,}974,$$

$$K_{\text{н}}(1,3) = 1 - \frac{0}{39} = 1,$$

$$K_{\text{н}}(2,3) = 1 - \frac{1}{39} = 0{,}974,$$

$$K_{\text{н}}(2,5) = 1 - \frac{23}{39} = 0{,}41,$$

$$K_{\text{н}}(3,4) = 1 - \frac{2}{39} = 0{,}949,$$

$$K_{\text{н}}(3,6) = 1 - \frac{0}{39} = 1,$$

$$K_{\text{н}}(4,5) = 1 - \frac{2}{39} = 0{,}949,$$

$$K_{\text{н}}(4,7) = 1 - \frac{19}{39} = 0{,}513\ ,$$

$$K_{\text{н}}(4,6) = 1 - \frac{8}{39} = 0{,}795,$$

$$K_{\text{н}}(5,7) = 1 - \frac{2}{39} = 0{,}949,$$

$$K_{\text{н}}(6,7) = 1 - \frac{0}{39} = 1.$$

Аналізуючи роботи за коефіцієнтом напруженості отримуємо, що критичними є роботи (1, 3), (3, 6) та (6, 7). Вони мають нульовий резерв часу і коефіцієнт напруженості 1.

Роботи (2, 5) та (4, 7) мають великий резерв часу (23 та 19), відтак, можна на даному етапі з них зняти ресурси і перекинути їх на ті, що перебувають на критичному шляху. Аналогічно, роботи (3, 4), (4, 5), (5, 7) мають резерв часу 2. Роботи (2, 3) та (1, 2) вважаємо підкритичними. На рис. 2.50 критичний шлях позначено товстою лінією.

Розглянуті положення теорії графів можна використовувати при розв’язанні задач моделювання об’єктів із складною внутрішньою структурою.

Алгоритми, побудовані з використанням теорії графів, відрізняються високою швидкодією, а моделі об’єктів і процесів є наочними і простими в програмуванні.

Висновки

Теорія графів має широке практичне застосування. Розроблені в ній алгоритми, через свою узагальненість і високій рівень абстракції, використовуються для вирішення задач у транспортних і комп'ютерних мережах, будівельному проектуванні, молекулярному моделюванні, в геоінформаційних системах, в задачах планування и організації проєктів у різних сферах.

У загальному значенні *графом* називають непорожню множину вершин (вузлів), з’єднаних ребрами. Для різних галузей застосування графи можуть

розрізнятися напрямами, обмеженнями на кількість зв'язків, додатковими відомостями про їх вершини та ребра.

Розрізняють орієнтовані і неорієнтовані графи. Кожен граф можна задати трьома способами: графічно, аналітично й за допомогою матриць.

Граф є зручним способом подання бінарних відношень і відповідностей.

Найбільш відомими задачами теорії графів є задача про найкоротший шлях у графі, задача про максимальний потік, задача побудови мінімального кістякового дерева, задача про фарбування графа та ін.

Питання, викладені в цьому розділі, розглянуто у літературі [1, 2, 7, 10, 11, 16, 19, 20]

Питання для самоконтролю

1. Дайте визначення графа.
2. Якими способами можна задати граф?
3. Яка властивість графа називається ізоморфізмом?
4. Дайте визначення матриці суміжності графа.
5. У якому випадку матриця суміжності графа буде мати на своїй головній діагоналі відмінні від нуля елементи?
6. Чим визначається розмір матриці інциденцій графа?
7. Що являє собою орієнтоване дерево найкоротших шляхів графа?
8. Що є ознакою завершення виконання алгоритму побудови найкоротшого кістякового дерева?
9. Які дуги в задачі про максимальний потік у графі називаються збільшувальними? Зменшувальними?
10. Чи може збільшувальний шлях у задачі про максимальний потік містити зменшувальні дуги?
11. Наведіть приклади практичних задач, які можна розв'язувати за допомогою теорії графів.
12. Який граф називається мережею?
13. Що являє собою полюс мережі?
14. Для чого використовують мережеве планування.
15. Дайте визначення таких понять: критичний шлях, ранній термін виконання події, пізній термін виконання події, ранній термін початку операції, ранній термін закінчення операції; пізній термін закінчення; пізній термін початку операції; повний резерв часу; тривалість шляху; резерв часу шляху.
16. Як визначають характеристики мережевого графа, описані в п.15?

Задачі для самостійного розв'язування

1. Намалюйте повний граф з N вершинами, якщо: а) $N = 2$; б) $N = 4$; в) $N = 5$.

2. Скільки ребер у повному графі з N вершинами, якщо: а) $N = 2$; б) $N = 4$; в) $N = 5$.

3. Наведіть приклад графа з п’ятьма вершинами, у якого дві вершини мають однаковий степінь.

4. Побудуйте граф з п’ятьма вершинами, степені яких всі різні між собою, тобто дорівнюють 0, 1, 2, 3, 4.

5. Побудуйте матриці суміжності, інцидентності та список ребер для поданих нижче графів (рис. 2.51 – 2.54).

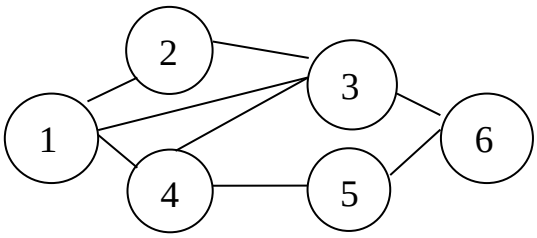


Рис. 2.57.

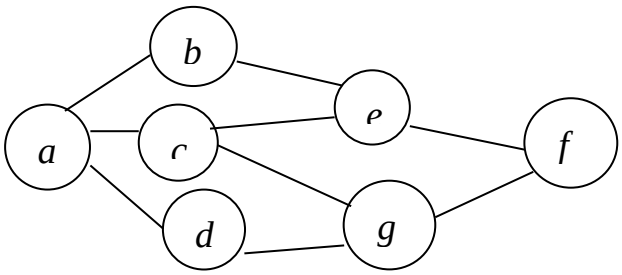


Рис. 2.58

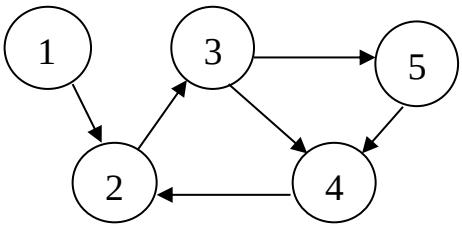


Рис.2.59.

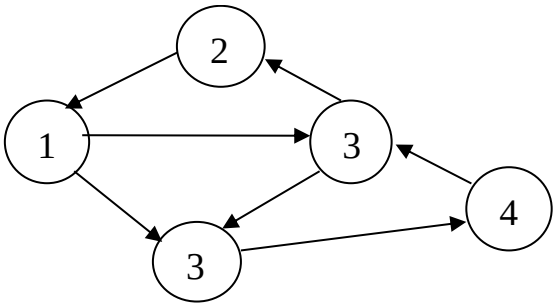


Рис.2.60.

6. Перелічить шляхи і контури графів на рис. 2.51 – 2.54.

7. Знайдіть довжину шляхів для графів на рис. 2.51 – 2.54.

8. Наведіть приклади часткових графів та підграфів для графів на рис. 2.51 – 2.54.

9. Наведіть приклади графів з одним, двома та трьома компонентами зв’язності.

10. Знайдіть групи ізоморфізмів для графів, зображених на рис. 2.51 – 2.54.

11. Перевірте властивості рефлексивності, транзитивності та симетричності для графів на рис. 2.51 – 2.54.

12. Визначте вид відношення порядку для графів на рис. 2.51 – 2.54.

13. Знайдіть цикломатичне та хроматичне числа для графів на рис. 2.51 – 2.54.

14. Знайдіть центр, радіус та діаметр для графів на рис. 2.51 – 2.54.

15. Знайдіть гамільтонові цикли для графів на рис. 2.51 – 2.54.

16. Складіть матриці суміжності й інциденцій для заданих нижче неорієнтованих графів:

а) $X=\{x_1,x_2,x_3,x_4\}; G(x_1)=\{x_2,x_4\}; G(x_2)=\{x_1,x_3,x_4\}; G(x_3)=\{x_2,x_3\};$ $G(x_4)=\{x_1,x_2,x_3\}.$

б) $X=\{x_1,x_2,x_3,x_4\}; G(x_1)=\{x_1,x_4\}; G(x_2)=\{x_3,x_4\}; G(x_3)=\{x_2,x_3,x_4\};$ $G(x_4)=\{x_1,x_2,x_3\}.$

17. Зобразіть графічно неорієнтований граф, заданий такою матрицею суміжності:

$$A=\begin{pmatrix}0&0&1&0\\0&1&1&1\\1&1&0&1\\0&1&1&1\end{pmatrix}.$$

18. Задайте аналітично орієнтований граф, зображений нижче.

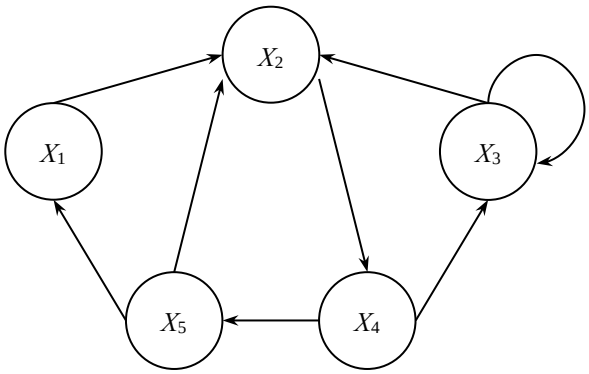


Рис. 2.55. Граф до задачі 18

19. Зобразіть графічно орієнтований граф, заданий такою матрицею інциденцій:

$$S=\begin{pmatrix}-1&1&0&0&-1&0\\1&0&0&-1&0&1\\0&0&-1&1&1&0\\0&-1&1&0&0&-1\end{pmatrix}.$$

20. Знайдіть найкоротший шлях від вершини *s* до вершини *t* за допомогою алгоритму Дейкстри для заданого нижче графа.

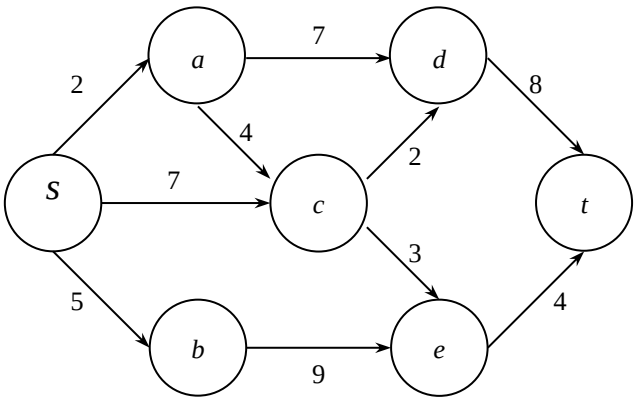


Рис. 2.56. Граф до задачі 20

21. Для заданого неорієнтованого графа (рис. 2.57) побудуйте найкоротше кістякове дерево.

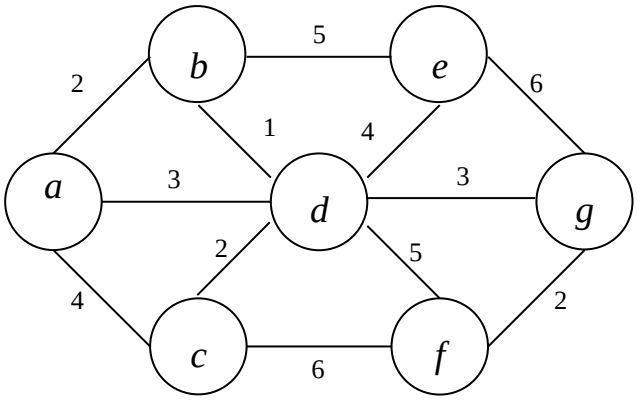


Рис. 2.57. Граф до задачі 21

22. Знайдіть збільшувальний шлях від джерела s до стоку t для зображеного нижче графа.

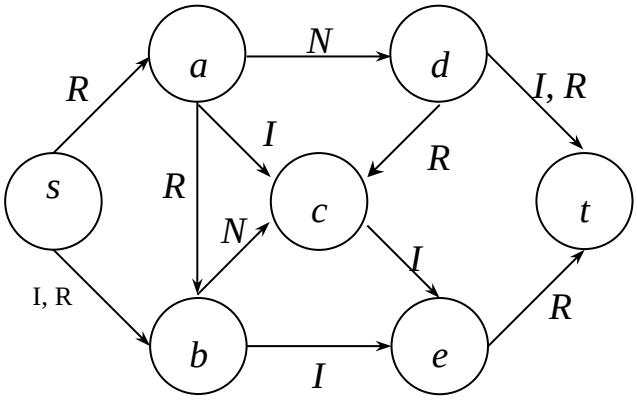


Рис. 2.58. Граф до задачі 22

23. Знайдіть максимальну величину потоку від вершини s до вершини t, використовуючи алгоритм Форда й Фалкерсона. На дугах графа (рис. 2.59) зазначено їхні максимальні пропускні здатності.

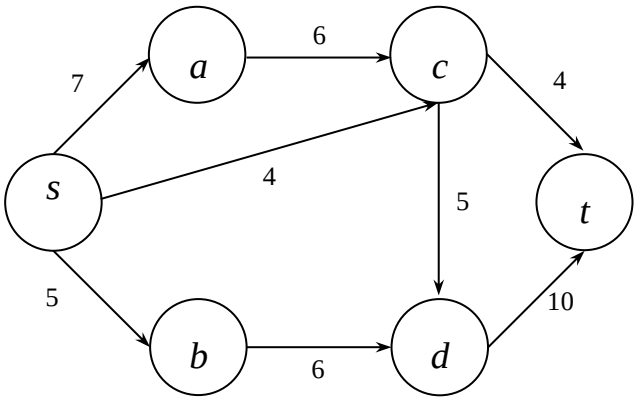


Рис. 2.59. Граф до задачі 23

24. Для графа, поданого на рис. 2.60, за допомогою алгоритму Дейкстри побудуйте дерево найкоротших шляхів із коренем у вершині s.

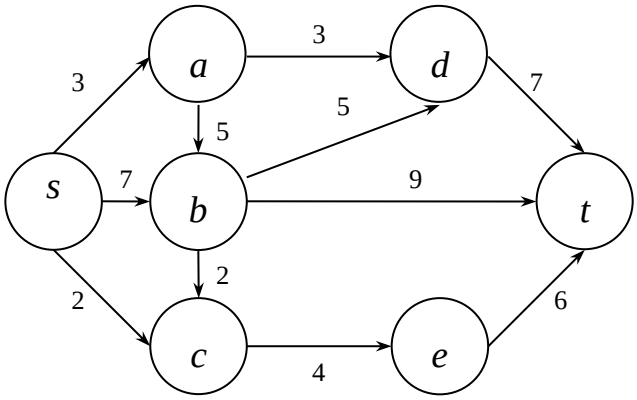


Рис. 2.60. Граф до задачі 24

25. Складіть аналітичний опис неорієнтованого графа за такою матрицею суміжності:

$$A=\begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

26. Наведіть приклад мережі з одним, двома та трьома найкоротшими шляхами.

27. Наведіть приклад мережі з максимальним потоком, що дорівнюю сумі потоків стоку.

28. За яких умов в мережі може існувати декілька критичних шляхів? Наведіть приклад.

Індивідуальне завдання до розділу 2

Тема завдання: Побудова та розрахунок мережевого графіка.

Мета завдання: ознайомитись з поняттям мережевого планування та базовими характеристиками мережевого графа, навчитися розраховувати основні його параметри.

Порядок виконання завдання

1. Опрацювати необхідний теоретичний матеріал.
2. Побувати мережевий граф, використовуючи вихідні дані відповідно до свого варіанта[7] (див. табл. 2.3).
3. Розрахувати основні характеристики побудованого мережевого графа. Визначити критичний шлях. Зробити висновки.
3. Оформити звіт про виконання завдання такого змісту:
   - номер варіанта,
   - вихідні дані варіанта, побудований мережевий графік,
   - розрахунки із необхідним рівнем деталізації,
   - висновки.
4. Захистити виконану роботу.

При виконанні завдань можна застосовувати стандартні засоби ЕОМ або власноруч написані програми.

Зміст завдання

Припустимо, для деякого проєкта відомо необхідні роботи та час їх виконання (табл. 2.4). Скласти мережевий граф та провести розрахунки його параметрів.

Критерії оцінювання індивідуального завдання

При оцінюванні індивідуального завдання враховується правильність виконання завдання (50 %), пояснення щодо обраних методів і правил обчислення, знання визначень і термінології (40 %), своєчасне подання завдання на перевірку (10 %).

[7] Вибір варіанта визначає викладач.

Таблиця 2.3

Вихідні дані до індивідуального завдання

i, j	1, 2	1, 3	2, 3	2, 5	3, 4	3, 6	4, 5	4, 6	4, 7	5, 7	6,7
Варіант	Значення $t(i, j)$										
1	3	14	8	11	9	5	6	8	10	5	7
2	12	5	9	10	12	8	10	6	9	14	9
3	8	9	10	10	8	12	8	7	10	7	9
4	6	10	11	12	7	12	8	10	3	5	6
5	4	9	5	9	10	5	9	8	11	10	7
6	13	4	8	5	8	5	9	6	8	6	10
7	3	12	8	11	9	8	6	8	10	5	7
8	12	5	9	10	9	10	10	6	9	14	9
9	8	9	10	10	8	12	8	7	10	7	9
10	6	10	11	12	7	10	9	10	8	5	6
11	4	6	5	9	10	5	9	8	12	10	6
12	3	4	8	5	5	5	9	6	8	6	12
13	3	7	8	11	9	5	6	8	10	5	7
14	12	5	9	10	12	8	10	6	9	14	9
15	8	9	4	10	8	12	8	7	10	7	9
16	6	10	11	8	7	12	8	10	3	5	6
17	4	9	5	9	14	5	9	10	11	10	7
18	13	4	6	5	8	5	9	8	8	6	10
19	3	14	8	11	9	5	4	8	10	5	8
20	12	5	9	10	12	8	10	6	9	14	9
21	8	9	10	6	8	9	8	7	10	7	9
22	6	10	6	9	7	12	8	10	3	5	6
23	4	9	9	9	10	5	9	8	7	10	7
24	13	9	8	5	8	5	9	6	10	6	10
25	3	12	8	11	9	5	6	8	10	5	7
26	12	5	8	10	11	8	10	6	9	14	9
27	8	9	10	10	8	12	8	7	10	7	9
28	6	10	11	12	8	12	8	10	3	5	6
29	4	9	5	9	8	5	9	8	11	10	7
30	13	4	8	5	9	5	9	6	8	6	10
31	4	6	9	3	5	9	10	5	6	7	9
32	2	8	5	4	7	23	12	4	5	10	8

РОЗДІЛ 3

ОСНОВИ ТЕОРІЇ АВТОМАТІВ

Мета розділу: вивчення основних булевих функцій, їхніх властивостей та методів мінімізації. Набуття практичних навичок синтезу скінченних автоматів

3.1. Основні поняття

Двозначна логіка дозволяє математично описати реальні об'єкти, які перебувають в одному із двох можливих станів. Вони описуються за допомогою *булевих змінних*, кожна з яких має лише два можливі значення – нуль або одиницю.

Відношення між булевими змінними задають за допомогою *булевих функцій*, які, подібно до числових функцій, можуть залежати в загальному випадку від кількох булевих змінних, тобто $y = f(x_1, x_2, ..., x_n)$.

Найважливішою властивістю булевих функцій є те, що вони, як і булеві змінні, можуть мати тільки два можливих значення – нуль або одиницю. Ця властивість дозволяє використовувати їх як булеві змінні в інших функціях.

Довільна булева функція задається одним з трьох способів: матричним (табличним), геометричним й аналітичним. При матричному способі булева функція $f(x_1, x_2, ..., x_n)$ задається за допомогою *таблиці істинності* (див. табл. 3.1 і 3.2), у лівій частині якої подано всі можливі двійкові набори довжини n, а у правій наведено значення функції, що відповідають цим наборам.

Під *двійковим набором* $y = y_1, y_2, ..., y_n$ (тут y_i набуває значень 0 або 1, для всіх значень індекса i) розуміють сукупність значень аргументів $x_1, x_2, ..., x_n$ булевої функції f. Двійковий набір має довжину n, якщо він включає n цифр із множини {0,1}. Наприклад, у табл. 3.1 подано всі двійкові набори довжини 3.

Таблиця 3.1

$x_1\ x_2\ x_3$	$f(x_1, x_2, x_3)$
000	0
001	1
010	0
011	0
100	1
101	1
110	0
111	1

Таблиця 3.2

номери наборів	$f(x_1, x_2, x_3)$
0	0
1	1
2	0
3	0
4	1
5	1
6	0
7	1

Іноді двійкові набори в таблиці істинності булевої функції зручно записувати, використовуючи *номери наборів*. Для цього запишемо аргументи $x_1, x_2, ..., x_n$ в порядку зростання їх індексів. Тоді будь-який двійковий набір $y = y_1, y_2, ..., y_n$, $y_i \in \{0,1\}$ можна розглядати як ціле двійкове число N, а саме: $N = y_1 \cdot 2^{n-1} + y_2 \cdot 2^{n-2} + \ldots + y_n$. Його називають *номером набору y*. Наприклад, двійкові набори 101 і 111 мають номери 5 і 7 відповідно. Очевидно, що будь-яка булева функція може бути задана таблицею істинності, у якій двійкові набори замінено на їхні номери (табл. 3.2).

Булеві функції, що залежать від великого числа змінних, задавати таблицею істинності незручно через їхню громіздкість. Приміром, таблиця істинності булевої функції 8 змінних буде містити 2^8, тобто 256 рядків. У зв'язку з цим для задання функцій багатьох змінних зручно використовувати модифікацію таблиці істинності.

Розглянемо спосіб її побудови для функції n змінних. Множина n змінних функції розбивається на дві підмножини $x_1, x_2, ..., x_{j-1}$ і $x_j, x_{j+1}, x_{j+2}, ..., x_n$. Змінними $x_1, x_2, ..., x_{j-1}$ позначають рядки таблиці істинності, записуючи в кожному з них відповідні двійкові набори довжини $j-1$. Змінними $x_j, x_{j+1}, x_{j+2}, ..., x_n$ відмічають стовпчики таблиці, записуючи в кожному з них відповідні двійкові набори довжини $n-j+1$. Значення функції записується в клітинці на перетині відповідного рядка й стовпця (табл. 3.3).

Таблиця 3.3

$x_1, x_2, ..., x_{j-1}$	$x_j, x_{j+1}, x_{j+2}, ..., x_n$.			
	00…0	00…1	…	11…1
00…0			…	
00…1			…	
…	…	…	…	…
11…1			…	

Основними в булевій алгебрі виступають три функції:

1. *Заперечення* (позначається як $y = \bar{x}$).

Вона має одну змінну та набуває протилежного (інверсного) стосовно неї значення, тобто

$$y = \bar{x} = \begin{cases} 0, & \text{якщо } x = 1, \\ 1, & \text{якщо } x = 0. \end{cases}$$

2. *Диз'юнкція* (логічне додавання).

В неї дві змінні x_1 й x_2. Диз'юнкція набуває значення «0» тільки тоді, коли $x_1 = 0$ й $x_2 = 0$ одночасно, тобто

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Диз'юнкція позначається таким чином: $y = x_1 \vee x_2$ або $y = x_1 + x_2$.

3. *Кон'юнкція* (логічне множення)

Це функція двох змінних x_1 і x_2, яка набуває значення «1» тільки тоді, коли $x_1 = 1$ і $x_2 = 1$ одночасно, тобто

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Кон'юнкція позначається як $y = x_1 \cdot x_2$ або $y = x_1 \wedge x_2$, $y = x_1$ & x_2.

Вирази $\bar{x}$, $x_1 \vee x_2$, $x_1 \wedge x_2$ являють собою *логічні формули*. Більш складні формули створюють шляхом заміщення змінних іншими формулами, які зазвичай беруться в дужки. Наприклад: $(x_1 \vee x_2) \vee \overline{(x_1 \wedge x_3)}$. Кожна формула визначає деяку булеву функцію. Це *аналітичний* спосіб задання функції.

Дві функції (і відповідні формули) є *рівносильними*, якщо при будь-яких значеннях аргументів вони набувають однакових значень. Рівносильні формули з'єднуються знаком «дорівнює» (=).

Універсальний спосіб перевірки тотожності функцій полягає у використанні таблиць істинності.

Приклад 3.1. Перевіримо тотожність таких функцій: $y_1 = \overline{x_1 \vee x_2}$ і $y_2 = \bar{x}_1 \wedge \bar{x}_2$.

Складемо таблиці істинності. Для цього обчислимо значення функцій y_1 і y_2 для будь-яких можливих комбінацій значень змінних x_1 і x_2:

x_1	x_2	y_1	y_2
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Оскільки для одних і тих самих значень змінних функції y_1 й y_2 набувають однакових значень, то вони рівносильні.

Наведемо основні тотожності булевої алгебри, які неважко довести за допомогою таблиць істинності.

1. *Комутативність* логічного додавання й множення, тобто

$$x \vee y = y \vee x;$$
$$x \wedge y = y \wedge x.$$

2. *Асоціативність* логічного додавання й множення

$$x \vee (y \vee z) = (x \vee y) \vee z;$$
$$x \wedge (y \wedge z) = (x \wedge y) \wedge z.$$

3. *Дистрибутивність* множення відносно додавання й додавання відносно множення:

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z);$$
$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z).$$

4. Властивості констант:

$$x \vee 0 = x \quad x \vee 1 = 1;$$
$$x \wedge 0 = 0 \quad x \wedge 1 = x.$$

5. Властивості заперечення:

$$x \vee \bar{x} = 1;$$
$$x \wedge \bar{x} = 0.$$

6. Закони де Моргана:

$$\overline{x \vee y} = \bar{x} \wedge \bar{y};$$
$$\overline{x \wedge y} = \bar{x} \vee \bar{y}.$$

7. Закони поглинання:

$$x \vee (x \wedge y) = x;$$
$$x \wedge (x \vee y) = x.$$

8. Закони ідемпотентності:

$$x \vee x = x;$$
$$x \wedge x = x.$$

3.2. Мінімізація логічних функцій

Із положень **§** 3.1 випливає, що одна й та сама логічна функція може мати різну форму запису. Особливо виділяють диз'юнктивну й кон'юнктивну нормальні форми (ДНФ і КНФ).

Диз'юнктивною нормальною формою (ДНФ) називається диз'юнкція скінченного числа різних членів, кожний з яких являє собою кон'юнкцію окремих змінних або їх заперечень, що входять у даний член не більше одного разу.

Кон'юнктивною нормальною формою (КНФ) називається кон'юнкція скінченного числа різних членів, кожний з яких являє собою диз'юнкцію окремих змінних або їх заперечень, котрі входять у даний член не більше одного разу.

Використовуючи наведені вище закони, будь-яку формулу можна звести до ДНФ або до КНФ.

Приклад 3.2. Запишемо функцію $\left(x_1 \cdot x_2 \vee \overline{x_2} x_3\right) \cdot \overline{\overline{x_1} \cdot x_4}$ у вигляді ДНФ і КНФ.

Спочатку перетворимо вираз $\overline{\overline{x_1} \cdot x_4}$. Використовуючи закони де Моргана й властивості заперечення, отримуємо, що $\overline{\overline{x_1} \cdot x_4} = \overline{\overline{x_1}} \vee \overline{x}_4 = x_1 \vee \overline{x}_4$.

Тепер перетворимо вихідну функцію, застосувавши дистрибутивний закон множення відносно додавання та закони поглинання, а саме:

$$\left(x_1 \cdot x_2 \vee \overline{x_2} x_3\right) \cdot \overline{\overline{x_1} \cdot x_4} = \left(x_1 \cdot x_2 \vee \overline{x_2} x_3\right) \cdot \left(x_1 \vee \overline{x_4}\right) = x_1 x_2 \vee x_1 \overline{x_2} x_3 \vee x_1 x_2 \overline{x_4} \vee \overline{x_2} x_3 \overline{x_4}.$$

Отже, ДНФ даної функції має такий вигляд: $x_1 x_2 \vee x_1 \overline{x_2} x_3 \vee x_1 x_2 \overline{x_4} \vee \overline{x_2} x_3 \overline{x_4}$.

Виконаємо перетворення вихідної функції за допомогою другого дистрибутивного закону, а саме:

$$\left(x_1 \cdot x_2 \vee \overline{x_2} x_3\right) \cdot \overline{\overline{x_1} \cdot x_4} = \left(x_1 \cdot x_2 \vee \overline{x_2} x_3\right) \cdot \left(x_1 \vee \overline{x_4}\right) = \left(x_1 \vee \overline{x_2} x_3\right)\left(x_2 \vee \overline{x_2} x_3\right) \cdot \left(x_1 \vee \overline{x_4}\right) =$$
$$= \left(x_1 \vee \overline{x_2}\right)\left(x_1 \vee x_3\right)\left(x_2 \vee \overline{x_2}\right) \times \left(x_2 \vee x_3\right)\left(x_1 \vee \overline{x_4}\right) =$$
$$= \left(x_1 \vee \overline{x_2}\right)\left(x_1 \vee x_3\right)\left(x_2 \vee x_3\right)\left(x_1 \vee \overline{x_4}\right).$$

Останнє перетворення отримано з огляду на властивості заперечення.

Таким чином, КНФ даної функції має такий вигляд:

$$\left(x_1 \vee \overline{x_2}\right)\left(x_1 \vee x_3\right)\left(x_2 \vee x_3\right)\left(x_1 \vee \overline{x_4}\right).$$

Члени диз'юнктивної нормальної форми, що включають k змінних називають *мінтермами* k-го рангу.

Члени кон'юнктивної нормальної форми, складені з k змінних, називають *макстермами* k-го рангу.

Наприклад: x_1x_2 – мінтерм 2-го рангу; $x_1\overline{x_2}x_3$ – мінтерм 3-го рангу; $x_1 \vee \overline{x_2}$ – макстерм 2-го рангу.

Якщо в кожному члені нормальної форми містяться всі змінні (або в прямому, або в інверсному вигляді), то вона називається *досконалою* (диз'юнктивною або кон'юнктивною) *нормальною* формою і позначається як ДДНФ або ДКНФ відповідно. Перехід від ДНФ (КНФ) до досконалої форми виконується на основі такого співвідношення: $x \vee \bar{x} = 1\left(x \wedge \bar{x} = 0\right)$.

Приклад 3.3. Записати функцію: $x_1x_2 \vee x_1\overline{x_2}x_3 \vee x_1x_2\overline{x_4} \vee \overline{x_2}x_3\overline{x_4}$ у вигляді досконалої форми.

Розв'язування

$$x_1x_2 \vee x_1\overline{x_2}x_3 \vee x_1x_2\overline{x_4} \vee \overline{x_2}x_3\overline{x_4} = x_1x_2 \cdot 1 \cdot 1 \vee x_1\overline{x_2}x_3 \cdot 1 \vee x_1x_2 \cdot 1 \cdot \overline{x_4} \vee 1 \cdot \overline{x_2}x_3\overline{x_4} =$$

$$= x_1x_2\left(\overline{x_3} \vee x_3\right)\left(\overline{x_4} \vee x_4\right) \vee x_1\overline{x_2}x_3\left(x_4 \vee \overline{x_4}\right) \vee x_1x_2\left(x_3 \vee \overline{x_3}\right) \cdot \overline{x_4} \vee \left(x_1 \vee \overline{x_1}\right)\overline{x_2}x_3\overline{x_4} =$$

$$= x_1x_2x_3x_4 \vee x_1x_2x_3\overline{x_4} \vee x_1x_2\overline{x_3}x_4 \vee x_1x_2\overline{x_3}\,\overline{x_4} \vee x_1\overline{x_2}x_3x_4 \vee$$
$$\vee x_1\overline{x_2}x_3\overline{x_4} \vee x_1x_2x_3\overline{x_4} \vee x_1x_2\overline{x_3}\,\overline{x_4} \vee x_1\overline{x_2}x_3\overline{x_4} \vee \overline{x_1}\overline{x_2}x_3\overline{x_4} =$$
$$= x_1x_2x_3x_4 \vee x_1x_2x_3\overline{x_4} \vee x_1x_2\overline{x_3}x_4 \vee x_1x_2\overline{x_3}\,\overline{x_4} \vee x_1\overline{x_2}x_3x_4 \vee x_1\overline{x_2}x_3\overline{x_4} \vee \overline{x_1}x_2x_3\overline{x_4}.$$

ДКНФ має такий вигляд:

$$x_1x_2x_3x_4 \vee x_1x_2x_3\overline{x_4} \vee x_1x_2\overline{x_3}x_4 \vee x_1x_2\overline{x_3}\,\overline{x_4} \vee x_1\overline{x_2}x_3x_4 \vee x_1\overline{x_2}x_3\overline{x_4} \vee \overline{x_1}x_2x_3\overline{x_4}.$$

Оскільки булева функція може бути записана по-різному, виникає потреба пошуку найбільш компактної форми її подання. Ця процедура називається *мінімізацією логічних функцій*. Для розв'язування задачі мінімізації логічна функція попередньо зводиться до ДДНФ або ДКНФ.

Мінімізація функцій у булевому базисі відбувається згідно із законами поглинання й зводиться до пошуку мінімальної диз'юнктивної або кон'юнктивної форми. За критерій зазвичай беруть змінну C – ціну покриття, що обчислюється за таким виразом:

$$C = \sum_s g_s\left(n - S\right),$$

де g_s – число термів рангу S, що утворюють покриття функції від n змінних.

Найпоширеніші методи мінімізації – Квайна, Квайна – Мак-Класкі та Вейча – Карно.

Вихідною формою для кожного з них виступає одна з досконалих форм: ДДНФ або ДКНФ.

У загальному випадку кожний з вищезазначених методів передбачає триетапний алгоритм мінімізації, а саме:

Етап 1. Проводиться перехід від *досконалої* Д(К)НФ до *скороченої* Д(К)НФ шляхом об'єднання спочатку конституент, а потім усіх похідних членів більш низького рангу.

Скороченою формою називається Д(К)НФ, членами якої служать тільки ізольовані елементарні кон'юнкції (диз'юнкції).

Члени скороченої Д(К)НФ в алгебрі логіки називаються *первинними імплікантами* (*імпліцентами*).

Етап 2. Проводиться перехід від *досконалої* Д(К)НФ до *тупикової* Д(К)НФ.

Тупиковою називається Д(К)НФ, членами якої є прості імпліканти (імпліценти), причому серед них немає жодної зайвої.

Зайвим називається такий член функції, вилучення якого не впливає на її значення істинності. Назва «тупикова» показує, що подальша мінімізація функції в рамках нормальних форм уже неможлива.

Етап 3. Виконується перехід від *тупикової* (мінімальної серед нормальних) форми функції до її *мінімальної* форми.

Цей етап, називаний звичайно *факторизацією*, уже не є регулярним, як два попередні, а вимагає певної вправності, інтуїції й досвіду.

Іншими словами, пошук можливостей спрощення функції здійснюється шляхом спроб і випробувань. Для зменшення числа операцій заперечення слід застосовувати закон інверсії, а для зменшення числа кон'юнкцій і диз'юнкцій – відповідні розподільні закони.

3.2.1. *Метод Квайна*

Ідея мінімізації логічної функції за методом Квайна полягає в попарному порівнянні всіх імплікант, які входять до складу ДДНФ, з метою виявлення можливості поглинання якоїсь змінної (ця операція називається склеюванням)

Опишемо алгоритм цього методу.

Крок 1. *Перетворення логічної функції у* ДДНФ.

Якщо функція містить диз'юнктивні члени, що не належать до мінтермів, то їх необхідно розгорнути в мінтерми.

Крок 2. *Знаходження первинних імплікант*.

Кожний мінтерм ДДНФ порівнюється з основними мінтермами цієї форми. Щоб при порівнянні не пропустити жодного мінтерма, операцію склеювання бажано робити за допомогою спеціальних таблиць, у верхньому рядку й лівому стовпчику яких розташовуються всі мінтерми, що брали участь у порівнянні. Якщо якісь з них відрізняються меж собою тільки однією змінною (тобто в одному це x, а в іншому $\bar{x}$), то в клітинці таблиці, розташованої проти склеюваних мінтермів, виписується тільки їхня спільна частина. Заміна двох

мінтермів їх спільною частиною (склеювання) еквівалентна такій операції: $XY + \bar{X}Y = Y$. Процес склеювання за допомогою таблиць триває, допоки не буде одержано терми, склеювання яких неможливе.

На цьому процедура складання скороченої ДНФ закінчується.

Крок 3. *Розміщення позначок.*

Складається таблиця, число рядків якої дорівнює кількості отриманих первинних імплікант, а число стовпців збігається із кількістю мінтермів ДДНФ. Якщо в деякий мінтерм ДДНФ входить одна з первинних імплікант, то в місці перетину відповідного стовпця й рядка ставиться позначка.

Крок 4. *Знаходження істотних імплікант.*

Коли якийсь стовпчик складеної на кроці 3 таблиці має тільки одну позначку, то первинна імпліканта у відповідному рядку є істотною, оскільки без неї не буде отримано усієї множини заданих мінтермів. Стовпці, що відповідають істотним імплікантам, з таблиці викреслюються.

Крок 5. *Викреслювання зайвих стовців.*

Якщо після виконання кроку 4 таблиця містить два однакові стовпці (тобто такі, що мають позначки в однакових рядках), то один з них викреслюється. Покриття стовпця, що залишився, буде здійснювати відкинутий мінтерм.

Крок 6. *Викреслювання зайвих первинних імплікант.*

Якщо після відкидання якихось стовпців на кроці 5 з'являються рядки, що не мають жодної позначки, то первинні імпліканти, відповідні цим рядкам, вилучаються з подальшого розгляду, оскільки вони не покривають мінтерми, котрі залишилися в розгляді.

Крок 7. *Вибір мінімального покриття.*

У таблиці, яку було складено на кроці 3 і скореговано на кроках 4, 5 і 6, вибирається така сукупність первинних імплікант, що включає позначки в усіх стовпцях (принаймні по одній позначці в кожному). Якщо існує кілька можливих варіантів такого вибору, то перевага надається варіанту покриття з мінімальною сумарною кількістю літер в імплікантах, котрі утворюють покриття.

Таким чином, тупикова форма заданої функції буде складатися із суми істотних імплікант (крок 4) і первинних імплікант, які покривають мінтерми, що залишилися (крок 2).

Приклад 3.4. Знайти тупикову форму для поданої нижче ДНФ.

$f(x_1, x_2, x_3, x_4) = x_2\bar{x}_3 + \bar{x}_1x_3x_4 + x_1\bar{x}_2x_4 + x_1x_2\bar{x}_3x_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4$.

Розв'язування

Крок 1

Знаходимо ДДНФ заданої функції $f(x_1, x_2, x_{3,} x_4)$ (див. приклад 8), відтак:

$$f(x_1,x_2,x_3,x_4)=\left(x_1+\overline{x_1}\right)x_2\overline{x_3}\left(x_4+\overline{x_4}\right)+\overline{x_1}\left(x_2+\overline{x_2}\right)x_3x_4+$$
$$+x_1\overline{x_2}\left(x_3+\overline{x_3}\right)x_4+x_1x_2\overline{x_3}x_4+\overline{x_1}x_2\overline{x_3}\,\overline{x_4}=$$
$$=x_1x_2\overline{x_3}x_4+x_1x_2\overline{x_3}\,\overline{x_4}+\overline{x_1}x_2\overline{x_3}x_4+\overline{x_1}x_2\overline{x_3}\,\overline{x_4}+\overline{x_1}x_2x_3x_4+x_1\overline{x_2}x_3x_4+x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}$$

Крок 2

Визначаємо первинні імпліканти. Для цього спочатку складемо таблицю вихідних мінтермів. Вона буде містити стільки рядків і стовпців, скільки мінтермів у ДДНФ заданої функції. На перехресті рядка й стовпця запишемо загальну частину відповідних мінтермів. Наприклад, перший стовпчик відповідає мінтерму $x_1x_2\overline{x}_3x_4$ а другий рядок – мінтерму $x_1x_2\overline{x}_3\overline{x}_4$, вони відрізняються один від одного лише змінною x_4, вона входить до першого мінтерму в прямому вигляді, а до другого – в інверсному. Тому в місці перетину першого стовпця й другого рядка ми запишемо тільки спільну частину цих мінтермів, а саме: $x_1x_2\overline{x}_3$. Так заповнюються всі клітинки таблиці.

Вихідні терми	$x_1x_2\overline{x_3}x_4$	$x_1x_2\overline{x_3}\,\overline{x_4}$	$\overline{x_1}x_2\overline{x_3}x_4$	$\overline{x_1}x_2\overline{x_3}\,\overline{x_4}$	$\overline{x_1}x_2x_3x_4$	$\overline{x_1}x_2x_3x_4$	$x_1\overline{x_2}x_3x_4$	$x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}$
$x_1x_2\overline{x_3}x_4$		$x_1x_2\overline{x_3}$	$x_2\overline{x_3}x_4$					$x_1\overline{x_3}x_4$
$x_1x_2\overline{x_3}\,\overline{x_4}$	$x_1x_2\overline{x_3}$			$x_2\overline{x_3}\,\overline{x_4}$				
$\overline{x_1}x_2\overline{x_3}x_4$	$x_2\overline{x_3}x_4$			$\overline{x_1}x_2\overline{x_3}$	$\overline{x_1}x_2x_4$			
$\overline{x_1}x_2\overline{x_3}\,\overline{x_4}$		$x_2\overline{x_3}\,\overline{x_4}$	$\overline{x_1}x_2\overline{x_3}$					
$\overline{x_1}x_2x_3x_4$			$\overline{x_1}x_2x_4$			$\overline{x_1}x_3x_4$		
$\overline{x_1}\,\overline{x_2}x_3x_4$					$\overline{x_1}x_3x_4$		$\overline{x_2}x_3x_4$	
$x_1\overline{x_2}x_3x_4$						$x_2\overline{x_3}x_4$		$x_2\overline{x_2}x_4$
$x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}$	$x_1\overline{x_3}x_4$						$x_1\overline{x_2}x_4$	

Тепер складемо таблицю первинних імплікант. Включаємо в неї всі імпліканти, отримані в першій таблиці. У нашому випадку їх дев’ять. Далі заповнюємо таблицю так само, як і попередню. Прямокутниками позначено первинні імпліканти, які не склеюються.

Первинна імпліканта	$x_1x_2\overline{x_3}$	$x_2\overline{x_3}x_4$	$x_2\overline{x_3}\,\overline{x_4}$	$\overline{x_1}x_2\overline{x_3}$	$\boxed{x_1\overline{x_3}x_4}$	$\boxed{\overline{x_1}x_3x_4}$	$\boxed{\overline{x_2}x_3x_4}$	$\boxed{x_1\overline{x_2}x_4}$	$\boxed{\overline{x_1}x_2x_4}$
$x_1x_2\overline{x_3}$				$\boxed{x_2\overline{x_3}}$					
$x_2\overline{x_3}x_4$			$x_2\overline{x_3}$						
$x_2\overline{x_3}\,\overline{x_4}$		$x_2\overline{x_3}$							
$\overline{x_1}x_2\overline{x_3}$	$x_2\overline{x_3}$								
$x_1\overline{x_3}x_4$									
$\overline{x_1}x_3x_4$									
$\overline{x_2}x_3x_4$									
$x_1\overline{x_2}x_4$									
$\overline{x_1}x_2x_4$									

Крок 3. Розміщення позначок.

Складаємо таблицю, рядки якої відповідають первинним імплікантам, отриманим на попередньому кроці, а стовпчики – вихідним термам. Якщо відповідна імпліканта міститься в термі – то на перетині цього рядка й стовпця ставимо позначку V. Наприклад, мінтерм $x_1x_2\overline{x_3}x_4$ містить первинні імпліканти $x_2\overline{x_3}x_4$ та $x_2\overline{x_3}$. Тому на перетині відповідних стовпця (у нашому прикладі він перший) і рядків (перший та останній) ставимо позначку V.

Первинні імпліканти	Вихідні терми							
	$x_1x_2\overline{x_3}x_4$	$x_1x_2\overline{x_3}\,\overline{x_4}$	$\overline{x_1}x_2\overline{x_3}x_4$	$\overline{x_1}x_2\overline{x_3}\,\overline{x_4}$	$\overline{x_1}x_2x_3x_4$	$\overline{x_1}\,\overline{x_2}x_3x_4$	$x_1\overline{x_2}x_3x_4$	$x_1\overline{x_2}\,\overline{x_3}x_4$
$x_2\overline{x_3}x_4$	V							V
$\boxed{\overline{x_1}x_3x_4}$					V	V		
$\overline{x_2}x_3x_4$						V	V	
$\boxed{x_1\overline{x_2}x_4}$							V	V
$\overline{x_1}x_2x_4$			V		V			
$\boxed{x_2\overline{x_3}}$	V	V	V	V				

↑ ↑ ↑ ↑

Крок 4. Знаходження істотних імплікант. До них належить первинна імпліканта $x_2\overline{x_3}$, оскільки в стовпці $\overline{x_1}x_2\overline{x_3}\,\overline{x_4}$ наявна тільки одна позначка і вона відповідає рядку $x_2\overline{x_3}$.

З останньої таблиці викреслюємо стовпці, що відповідають істотній імпліканті (їх позначено стрілками). Після цього таблиця набуває такого вигляду:

Первинні імпліканти	Вихідні терми			
	$\overline{x_1}x_2x_3x_4$	$\overline{x_1}\,\overline{x_2}x_3x_4$	$x_1\overline{x_2}x_3x_4$	$x_1\overline{x_2}\,\overline{x_3}x_4$
$x_2\overline{x_3}x_4$				V
$\boxed{\overline{x_1}x_3x_4}$	V	V		
$\overline{x_2}x_3x_4$		V	V	
$\boxed{x_1\overline{x_2}x_4}$			V	V
$\overline{x_1}x_2x_4$	V			

Крок 5. Не виконується, оскільки в отриманій після викреслювання таблиці немає стовпців, які мають однакові позначки.

Крок 6. Не здійснюємо, тому що в кожному рядку отриманої таблиці є позначки.

Крок 7. Вибір мінімального покриття.

Мінімальне покриття вихідних термів, що залишилися, здійснюють первинні імпліканти $\overline{x_1}x_3x_4$ й $x_1\overline{x_2}x_4$ (у таблиці їх позначено прямокутниами).

Таким чином, тупикова форма заданої ДНФ включає істотні імпліканти (у нашому випадку $x_2\overline{x_3}$) та імпліканти, які здійснюють мінімальне покриття ($\overline{x_1}x_3x_4$ й $x_1\overline{x_2}x_4$). Вона має такий вигляд:

$$f\left(x_1,x_2,x_3,x_4\right)=x_2\overline{x_3}+\overline{x_1}x_3x_4+x_1\overline{x_2}x_4.$$

Подальша мінімізація отриманої ДНФ можлива, наприклад, шляхом уведення в її запис дужок: $f\left(x_1,x_2,x_3,x_4\right)=x_2\overline{x_3}+\left(\overline{x_1}x_3+x_1\overline{x_2}\right)x_4$.

3.2.2. *Метод Квайна – Мак-Класкі*

Недоліком методу Квайна є необхідність повного попарного порівняння всіх мінтермів на етапі визначення первинних імплікант. Це стає суттєвим із зростанням числа мінтермів, оскільки збільшується і число попарних порівнянь.

Числова форма функцій алгебри логіки дозволяє спростити процес визначення первинних імплікант. Це враховано в методі Квайна – Мак-Класкі.

Його особливість – формалізація пошуку простих (первинних) імплікант. ЇЇ виконують у такий спосіб:

– усі конституенти одиниці з ДДНФ булевої функції *f* записують у вигляді їх двійкових номерів;

– усі номери розбиваються на неперетинні групи. Ознака створення *i*-ї групи: наявність *i* одиниць у кожному двійковому номері конституенти одиниці;

– склеювання проводять тільки між номерами сусідніх груп;

– склеювані номери позначаються в який-небудь спосіб (наприклад, їх закреслюють);

– склеювання проводять всілякі, як і в методі Квайна. Непозначені після склеювання номери і є простими імплікантами.

Зазначимо, що групу термів, ранг яких дорівнює $n - r$, називають r-кубом.

Опишемо алгоритм методу Квайна – Мак-Класкі.

Крок 1. Усі мінтерми ДДНФ записуються у вигляді їх двійкових номерів, а саме: змінній x ставиться у відповідність 1, а її інверсії $\bar{x}$ – 0. Наприклад, мінтерму $\overline{x_1}x_2x_3x_4$ відповідає двійковий номер 0111.

Крок 2. Пошук первинних імплікант.

Усі двійкові номери розбиваються відповідно до числа одиниць на неперетинні групи. При цьому в i-ту групу ввійдуть усі номери (набори), що мають у своєму двійковому записі i одиниць. Оскільки умовою створення r-куба є наявність розбіжності тільки за однією координатою (тобто в одному двійковому розряді) в $(r-1)$-кубах та однакових незалежних координат, то групи, які відрізняються двома й більше розрядами, просто не варто порівнювати. Отже, попарне порівняння можна робити тільки з наборами сусідніх за номерами груп.

Кроки $3-7$ виконуються аналогічно тим самим крокам в алгоритмі методу Квайна. Різниця лише в тому, що в них фігурують числові мінтерми.

Приклад 3.5. Знайти тупикову ДНФ функції, записаної аналітично, а саме:

$$f(x_1, x_2, x_3, x_4) = \bigvee_1 (3,4,5,7,9,11,12,13).$$

Тут у дужках перераховано десяткові еквіваленти наборів, на яких функція f дорівнює 1.

Розв'язування

Крок 1. Спочатку випишемо всі 0-куби (терми максимального рангу), а саме:

$$K^0 = \{0011, 0100, 0101, 0111, 1001, 1011, 1100, 1101\}.$$

Розіб'ємо 0-куби на групи за кількістю одиниць у кожному двійковому наборі, у результаті чого утворюємо три групи:

$$K_1^0 = \{0 \quad 1 \quad 0 \quad 0\}; \qquad K_2^0 = \begin{Bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{Bmatrix}; \qquad K_3^0 = \begin{Bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{Bmatrix}.$$

Крок 2. Пошук первинних імплікант.

а) порівнюємо K_1^0 й K_2^0 і знаходимо мінтерми, які відрізняються тільки в одному розряді. Набори, які склеюються, позначимо зірочкою (∗).

$$\begin{array}{l} 0 \quad 1 \quad 0 \quad 0* \end{array} \qquad \begin{array}{l} 0 \quad 0 \quad 1 \quad 1 \\ 0 \quad 1 \quad 0 \quad 1* \\ 1 \quad 0 \quad 0 \quad 1 \\ 1 \quad 1 \quad 0 \quad 0* \end{array}$$

За результатами порівняння будуємо 1-куби, у яких поглинута координата замінюється символом «–»:

$$\begin{array}{l} 0 \quad 1 \quad 0 \quad 0 \\ \underline{0 \quad 1 \quad 0 \quad 1} \\ 0 \quad 1 \quad 0 \quad - \end{array} \qquad \begin{array}{l} 0 \quad 1 \quad 0 \quad 0 \\ \underline{1 \quad 1 \quad 0 \quad 0} \\ - \quad 1 \quad 0 \quad 0 \end{array}$$

б) Тепер порівняємо K_2^0 й K_3^0.

$$\begin{array}{l} 0 \quad 0 \quad 1 \quad 1* \\ 0 \quad 1 \quad 0 \quad 1* \\ 1 \quad 0 \quad 0 \quad 1* \\ 1 \quad 1 \quad 0 \quad 0* \end{array} \qquad \begin{array}{l} 0 \quad 1 \quad 1 \quad 1 \ * \\ 1 \quad 0 \quad 1 \quad 1 \ * \\ 1 \quad 1 \quad 0 \quad 1 \ * \end{array}$$

На підставі порівняння будуємо 1-куби:

$$\begin{array}{l} 0 \quad 0 \quad 1 \quad 1 \\ \underline{0 \quad 1 \quad 1 \quad 1} \\ 0 \quad - \quad 1 \quad 1 \end{array} \qquad \begin{array}{l} 0 \quad 0 \quad 1 \quad 1 \\ \underline{1 \quad 0 \quad 1 \quad 1} \\ - \quad 0 \quad 1 \quad 1 \end{array}$$

$$\begin{array}{l} 0 \quad 1 \quad 0 \quad 1 \\ \underline{0 \quad 1 \quad 1 \quad 1} \\ 0 \quad 1 \quad - \quad 1 \end{array} \qquad \begin{array}{l} 0 \quad 1 \quad 0 \quad 1 \\ \underline{1 \quad 1 \quad 0 \quad 1} \\ - \quad 1 \quad 0 \quad 1 \end{array}$$

$$\begin{array}{cccc} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ \hline 1 & 0 & - & 1 \end{array} \qquad \begin{array}{cccc} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ \hline 1 & - & 0 & 1 \end{array}$$

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & - \end{array}.$$

Оскільки всі мінтерми 4-го рангу були «склеювані», тобто непозначених мінтермів не залишилось, робимо висновок, що первинних імплікант 4-го рангу немає.

Тепер розіб'ємо всі отримані тут 1-куби на групи відповідно до положення незалежної координати «–». У нас вийшло чотири групи:

$$K_1^1 = \begin{Bmatrix} 0 & 1 & 0 & - \\ 1 & 1 & 0 & - \end{Bmatrix},\ K_2^1 = \begin{Bmatrix} 0 & 1 & - & 1 \\ 1 & 0 & - & 1 \end{Bmatrix},\ K_3^1 = \begin{Bmatrix} 0 & - & 1 & 1 \\ 1 & - & 0 & 1 \end{Bmatrix},\ K_4^1 = \begin{Bmatrix} - & 1 & 0 & 0 \\ - & 0 & 1 & 1 \\ - & 1 & 0 & 1 \end{Bmatrix}.$$

Порівняння усередині кожної з груп K_1^1, K_2^1, K_3^1 і K_4^1 дає такий результат:

$$K_1^1: \begin{array}{cccc} 0 & 1 & 0 & - \\ 1 & 1 & 0 & - \\ \hline - & 1 & 0 & - \end{array} \qquad K_2^1: \begin{array}{cccc} 0 & 1 & - & 1 \times \\ 1 & 0 & - & 1 \times \end{array} \qquad K_3^1: \begin{array}{cccc} 0 & - & 1 & 1 \times \\ 1 & - & 0 & 1 \times \end{array}$$

$$K_4^1: \begin{array}{cccc} - & 1 & 0 & 0 \\ - & 1 & 0 & 1 \\ \hline - & 1 & 0 & - \end{array} \qquad - \ 0 \ 1 \ 1 \times .$$

Символом « × » позначено мінтерми, які не можна склеїти, отже, вони являють собою первинні імпліканти 3-го рангу.

За результатами склеювання отримано один 2-куб $\{- \ 1 \ 0 \ -\}$.

Таким чином, первинними імплікантами будуть:

1-куби: $0\ 1 - 1,\ \ 1\ 0 - 1,\ \ 0 - 1\ 1,\ \ 1 - 0\ 1,\ \ - 0\ 1\ 1$
і
2-куб: $- \ 1\ 0 - .$

Крок 3. Розміщення позначок у таблиці.

Цей крок здійснюється так само, як і в алгоритмі Квайна (див. приклад 9).

Первинні імпліканти	Вихідні терми							
	0 1 0 0	0 1 0 1	1 1 0 0	1 1 0 1	0 0 1 1	0 1 1 1	1 0 0 1	1 0 1 1
0 1 – 1		V				V		
1 0 – 1							V	V
0 – 1 1					V	V		
1 – 0 1				V			V	
– 0 1 1					V			V
–1 0 –	V	V	V	V				

Крок 4. Істотною імплікантою є 2-куб: $\{-\ 1\ 0\ -\}$, що відповідає кон'юнкції $x_2\overline{x_3}$, оскільки стовпці, вихідних термів 0100 і 1100 містять позначки тільки в рядку, відповідному цій імпліканті.

Викреслюємо стовпці, які містять позначки в рядку імпліканти {–1 0 –}. У таблиці їх виділено іншим кольором.

Кроки 5 і 6 не потрібні, враховуючи вигляд отриманої таблиці.

Крок 7. Вибір мінімального покриття термів, що залишилися.

Вочевидь, мінімальне покриття термів, які залишились, здійснюють імпліканти {10–1} і {0–11}. Їм відповідають кон'юнкції: $x_1\overline{x_2}x_4$ і $\overline{x_1}x_3x_4$.

Відтак, тупикова форма вихідної функції має вигляд:

$$f\left(x_1, x_2, x_3, x_4\right) = x_2\overline{x_3} + x_1\overline{x_2}x_4 + \overline{x_1}x_3x_4.$$

3.2.3. *Метод Вейча – Карно*

Цей метод передбачає побудову тупикової форми логічної функції за допомогою спеціальної таблиці, яка називається *діаграмою* (*картою*) *Вейча – Карно* і являє собою спеціально перебудовану таблицю істинності функції. Вона виступає графічним способом мінімізації функції і забезпечує відносну простоту роботи із виразами великого обсягу.

Опишемо алгоритм мінімізації логічних функцій за допомогою діаграми Вейча – Карно.

Крок 1. Будують прямокутну систему координат, вісь ординат якої збігається з лівим вертикальним краєм діаграми, а вісь абсцис – з нижнім краєм.

Крок 2. У даній системі координат зображують прямокутну таблицю, рядки й стовпчики якої нумеруються двійковими числами. Кожному двійковому розряду ставиться у відповідність певна логічна змінна.

За таких умов вісь ординат є віссю складного аргументу, який включає (коли n парне) $n/2$ перших логічних змінних: $x_1, x_2, ..., x_{n/2}$, а вісь абсцис служить для зміни складного аргументу, що містить $n/2$ наступних логічних змінних, а саме: $x_{\frac{n}{2}+1}, x_{\frac{n}{2}+2}, ..., x_n$. При непарному значенні n складні аргументи включають неоднакову кількість змінних.

Аби за допомогою описаної вище таблиці можна було не тільки записати будь-яку функцію, але й зробити її мінімізацію, необхідно виконати одну дуже важливу *умову спряженого сусідства* – за якою у сусідніх по вертикалі або по горизонталі клітинках (у фізичному, звичайно, сенсі) повинні перебувати сусідні конституенти (у логіко-математичному сенсі)

Крок 3. З метою виконання умови спряженого сусідства замінюємо двійковий код, яким були позначені стовпці й рядки діаграми, двійковим циклічним *кодом Грея*, користуючись при цьому записаним нижче правилом.

Цей двійковий код має ту особливість, що сусідні два числа відрізняються одне від одного тільки в одному розряді.

Правило перетворення двійкового коду в код Грея

1. Найстарша значуща цифра числа в коді Грея збігається із найстаршою значущою цифрою цього самого числа у двійковому коді.
2. Цифра в будь-якому іншому, молодшому розряді числа коду Грея має такі ознаки:

– збігається з відповідною цифрою у двійковому коді, якщо ліворуч від даної цифри коду Грея розташоване парне число одиниць;

– збігається із запереченням відповідної цифри у двійковому коді, якщо ліворуч від даної цифри коду Грея міститься непарна кількість одиниць;

Наприклад, послідовність у двійковому коді становить: 10110, а перетворення її на код Грея: 11101.

Застосування коду Грея до нумерації клітинок по осях координат у діаграмі Вейча – Карно забезпечує розміщення в сусідніх клітинках діаграми сусідніх членів ДД(К)НФ будь-якої логічної функції.

Крок 4. У самих клітинках діаграми необхідно проставити значення істинності реалізацій функції на кожному з наборів аргументів. Причому, якщо функцію задано за допомогою ДДНФ, то досить проставити тільки одиниці, а коли функцію задано у ДКНФ, тільки нулі.

Крок 5. Проводиться склеювання мінтермів таким чином, що 2^r сусідніх клітинок (0-кубів), об'єднуючись, утворюють r-куб (r = 1, 2, …).

Слід зауважити, що коли кількість змінних більша або дорівнює п'яти, відобразити графічно функцію у вигляді єдиної плоскої діаграми (карти) неможливо. У таких випадках будують комбіновану діаграму, яка складається із сукупності більш простих діаграм, наприклад, чотиривимірних. Тоді процедура мінімізації буде полягати в тому, що спочатку знаходять мінімальні форми всередині чотиривимірних кубів, а потім, розширюючи поняття сусідніх клітинок, відшукують терми мінімального рангу для сукупності діаграм.

Приклад 3.6. Знайти тупикову ДНФ функції записаної в аналітичному вигляді:

$$f\left(x_1, x_2, x_3, x_4\right) = \bigvee_1 \left(0,1,4,5,7,10,11,13,15\right).$$

Розв'язування

Застосуємо діаграми Вейча – Карно.

Кроки 1, 2.

Складемо таблицю, перенумерувавши її рядки й стовбці відповідно до коду Грея (див. рис. 3.1).

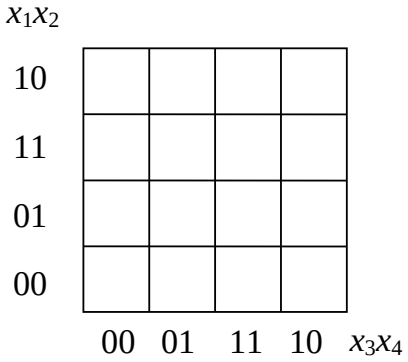


Рис. 3.1. Мінімізація логічних функцій методом Вейча – Карно (кроки 1, 2)

Кроки 3, 4, 5.

Даній функції відповідають такі 0-куби:

$K^0 = \{0000, 0001, 0100, 0101, 0111, 1010, 1011, 1101, 1111\}$. Позначимо їх на діаграмі, записуючи у відповідні клітинки одиниці (рис. 3.2), і мінімізуємо функцію відповідно до сформульованих вище правил.

Для цього виконуємо склеювання суміжних наборів, у клітинках яких записано одиниці. Зони склеювання позначено в таблиці (див. рис. 3.2) прямокутниками. При склеюванні діємо у такий спосіб: спочатку беремо одну з позначених зон і дивимось, які змінні залишаються сталими в її межах, виписуємо їх кон'юнкцію, причому якщо змінна в даній зоні нульова, то беремо її інверсію. Потім розглядаємо наступну зону і т.д. Кон'юнкції зон об'єднуємо за допомогою диз'юнкції.

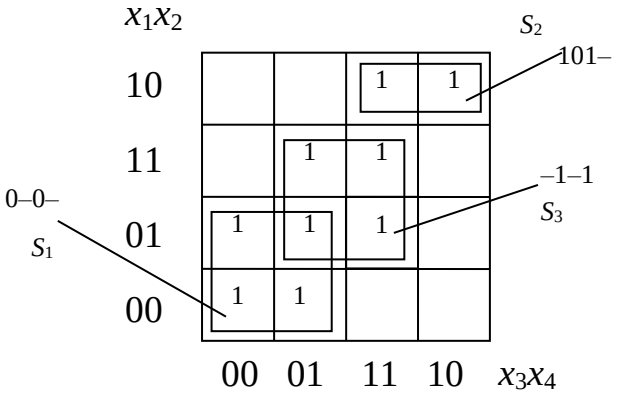


Рис. 3.2. Мінімізація логічних функцій методом Вейча – Карно (кроки 3, 4, 5)

У нашому прикладі в зоні S_1 незмінними залишаються x_1 та x_3. Відповідна їй кон'юнкція має вигляд $\bar{x}_1\bar{x}_3$. Аналогічно для зони S_2 отримуємо кон'юнкцію $x_1\bar{x}_2x_3$, а для S_3 – x_2x_4.

Таким чином, унаслідок склеювання отримуємо таку тупикову форму вихідної функції:

$$f\left(x_1, x_2, x_3, x_4\right) = \bar{x}_1\bar{x}_3 + x_2x_4 + x_1\bar{x}_2x_3.$$

Зауважимо, що при визначенні двоклітинного куба необхідно розрізняти, чи є він горизонтальним або вертикальним. Перший випадок (горизонтальний куб) відповідає тому, що сталі значення мають змінні рядка (у нашому прикладі це куб S_2). Другий випадок (вертикальний куб) відповідає тому, що сталі значення мають змінні стовпця.

3.3. Мінімізація частково визначених двійкових функцій

У практичній діяльності нерідко доводиться мати справи з функціями, які визначено не для всіх наборів змінних. Подібні випадки трапляються тоді, коли за умовами функціонування деякі з наборів не використовуються і тому байдуже, якого значення на них набуває функція. Цю обставину можна використовувати при мінімізації функції, задавши її на «байдужих» наборах так, щоб забезпечити найбільш економічну реалізацію.

Пояснимо сенс такого прийому. Нехай дано частково визначену функцію: $f = f\left(x_1, x_2, ..., x_n\right)$. Позначимо через $\varphi_1 = \varphi_1\left(x_1, x_2, ..., x_n\right)$ функцію, яку задано на всіх «байдужих» наборах одиницями; а через $\varphi_0 = \varphi_0\left(x_1, x_2, ..., x_n\right)$ ту, що задано на всіх «байдужих» наборах нулями. Задача оптимального визначення даної функції f зводиться до вибору із скороченого покриття для функції φ_1 мінімальної кількості кубів максимальної розмірності, сукупність яких покривала б усі вершини функції φ_0.

Ця сукупність якраз і утворює мінімальне покриття частково визначеної функції f. При цьому воно може покривати й деякі 0-куби, відповідні «байдужим» наборам, що свідчить про те, що функцію задано на цих наборах одиничними значеннями.

Для мінімізації частково визначених функцій використовують розглянуті раніше методи мінімізації Квайна, Квайна – Мак-Класкі й діаграми Вейча – Карно.

Приклад 3.7. Знайти тупикову форму функції чотирьох змінних, яку задано таким чином:

$$f\left(x_1, x_2, x_3, x_4\right) = \bigvee_1\left(0, 1^*, 2^*, 4^*, 6, 7^*, 8^*, 9, 11^*, 13^*, 14, 15^*\right).$$

У цьому виразі десяткові номери наборів, на яких функцію недовизначено, записано зі знаком «$*$».

Розв'язування

Складемо таблицю істинності заданої функції *f*, включаючи значення функцій φ_1 і φ_0.

№	x_1	x_2	x_3	x_4	f	φ_1	φ_0
0	0	0	0	0	1	1	1
1	0	0	0	1	*	1	0
2	0	0	1	0	*	1	0
3	0	0	1	1	0	0	0
4	0	1	0	0	*	1	0
5	0	1	0	1	0	0	0
6	0	1	1	0	1	1	1
7	0	1	1	1	*	1	0

№	x_1	x_2	x_3	x_4	f	φ_1	φ_0
8	1	0	0	0	*	1	0
9	1	0	0	1	1	1	1
10	1	0	1	0	0	0	0
11	1	0	1	1	*	1	0
12	1	1	0	0	0	0	0
13	1	1	0	1	*	1	0
14	1	1	1	0	1	1	1
15	1	1	1	1	*	1	0

Крок 1. Об'єднаємо 0-куби в групи за кількістю одиниць у кожному двійковому наборі, тобто

$$K_0^0 = \{0\ \ 0\ \ 0\ \ 0\};\qquad K_1^0 = \begin{Bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{Bmatrix};\qquad K_2^0 = \begin{Bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{Bmatrix};$$

$$K_3^0 = \begin{Bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{Bmatrix};\quad K_4^0 = \{1\ \ 1\ \ 1\ \ 1\},$$

Крок 2. Пошук первинних імплікант, а саме:

а) Порівнюємо K_0^0 й K_1^0 (набори, які склеюються, відзначимо символом $*$);

$$0\ \ 0\ \ 0\ \ 0\ *\qquad \begin{matrix} 0\ 1\ 0\ 0\ * \\ 0\ 0\ 0\ 1\ * \\ 0\ 0\ 1\ 0\ * \\ 1\ 0\ 0\ 0\ * \end{matrix}$$

На підставі порівняння будуємо 1-куби, у яких поглинуту координату замінюємо символом «–», а саме:

0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
0 1 0 0	0 0 0 1	0 0 1 0	1 0 0 0
0 – 0 0	0 0 0 –	0 0 – 0	– 0 0 0

б) Порівнюємо K_1^0 й K_2^0:

0 0 0 1 *	
0 0 1 0 *	0 1 1 0 *
0 1 0 0 *	1 0 0 1 *
1 0 0 0 *	

На базі порівняння будуємо такі 1-куби:

0 0 1 0	0 1 0 0	0 0 0 1	1 0 0 0
0 1 1 0	0 1 1 0	1 0 0 1	1 0 0 1
0 – 1 0	0 1 – 0	– 0 0 1	1 0 0 –

в) Порівнюємо K_2^0 й K_3^0:

	0 1 1 1 *
0 1 1 0 *	1 0 1 1 *
1 0 0 1 *	1 1 0 1 *
	1 1 1 0 *

На підставі порівняння будуємо 1-куби:

0 1 1 0	0 1 1 0	1 0 0 1	1 0 0 1
0 1 1 1	1 1 1 0	1 0 1 1	1 1 0 1
0 1 1 –	– 1 1 0	1 0 – 1	1 – 0 1

г) Порівнюємо K_3^0 й K_4^0:

0 1 1 1 *	
1 0 1 1 *	1 1 1 1 *
1 1 0 1 *	
1 1 1 0 *	

На основі порівняння будуємо 1-куби:

$$\begin{array}{cccc} 0&1&1&1 \\ \underline{1&1&1&1} \\ -&1&1&1 \end{array} \qquad \begin{array}{cccc} 1&0&1&1 \\ \underline{1&1&1&1} \\ 1&-&1&1 \end{array} \qquad \begin{array}{cccc} 1&1&0&1 \\ \underline{1&1&1&1} \\ 1&1&-&1 \end{array} \qquad \begin{array}{cccc} 1&1&1&0 \\ \underline{1&1&1&1} \\ 1&1&1&- \end{array}$$

Як бачимо, первинних імплікант четвертого рангу функції φ_1 немає.

Розіб'ємо всі 1-куби на групи відповідно до числа одиниць, враховуючи положення незалежної змінної «–», а саме:

$$K_1^1 = \begin{Bmatrix} 0&0&0&- \\ 1&0&0&- \\ 0&1&1&- \\ 1&1&1&- \end{Bmatrix}, \quad K_2^1 = \begin{Bmatrix} 0&0&-&0 \\ 0&1&-&0 \\ 1&0&-&1 \\ 1&1&-&1 \end{Bmatrix},$$

$$K_3^1 = \begin{Bmatrix} 0&-&0&0 \\ 0&-&1&0 \\ 1&-&0&1 \\ 1&-&1&1 \end{Bmatrix}, \quad K_4^1 = \begin{Bmatrix} -&0&0&0 \\ -&0&0&1 \\ -&1&1&0 \\ -&1&1&1 \end{Bmatrix}.$$

Порівняння груп, що мають різну кількість одиниць, усередині кубів K_1^1, K_2^1, K_3^1 і K_4^1 зумовлює такий результат:

$$\begin{array}{cccc} 0&0&0&- \\ 1&0&0&- \\ \hline -&0&0&- \end{array} \qquad \begin{array}{cccc} 0&1&1&- \\ 1&1&1&- \\ \hline -&1&1&- \end{array} \qquad \begin{array}{cccc} 0&0&-&0 \\ 0&1&-&0 \\ \hline 0&-&-&0 \end{array} \qquad \begin{array}{cccc} 1&0&-&1 \\ 1&1&-&1 \\ \hline 1&-&-&1 \end{array} \qquad \begin{array}{cccc} 0&-&0&0 \\ 1&-&1&0 \\ \hline 0&-&-&0 \end{array}$$

$$\begin{array}{cccc} 1&-&0&1 \\ 1&-&1&1 \\ \hline 1&-&-&1 \end{array} \qquad \begin{array}{cccc} -&0&0&0 \\ -&0&0&1 \\ \hline -&0&0&- \end{array} \qquad \begin{array}{cccc} -&1&1&1 \\ -&1&1&0 \\ \hline -&1&1&- \end{array}$$

Отже, первинних імплікант функції φ_1 третього рангу немає.

Відтак, за результатами склеювання отримано такі первинні імпліканти:

$$\{-0\,0-\},\ \{0--0\},\ \{-1\,1-\},\ \{1--1\}.$$

Крок 3. Розміщення позначок.

Результати виконання цього кроку показано в поданій нижче таблиці.

Первинні імпліканти φ_1	Вихідні терми φ_0			
	0 1 1 0	1 1 1 0	0 0 0 0	1 0 0 1
– 0 0 –			V	V
– 1 1 –	V	V		
1 – – 1				V
0 – – 0	V		V	

Крок 4. Пошук істотних імплікант.

Істотною тут виявляється первинна імпліканта { – 1 1 – }. Із таблиці викреслюємо відповідні їй стовпці.

Кроки 5 і 6 відсутні.

Крок 7. Вибір мінімального покриття.

Мінімальне покриття термів, що залишилися, здійснює первинна імпліканта { – 0 0 – }.

Таким чином, тупикова форма заданої частково визначеної функції має такий вигляд:

$$f\left(x_1, x_2, x_3, x_4\right) = x_2 x_3 + \overline{x}_2 \overline{x}_3 .$$

Тепер розв’яжемо цю задачу, використовуючи діаграми Вейча – Карно.

Розв’язування

Побудуємо діаграми Вейча – Карно функцій φ_1 та φ_0.

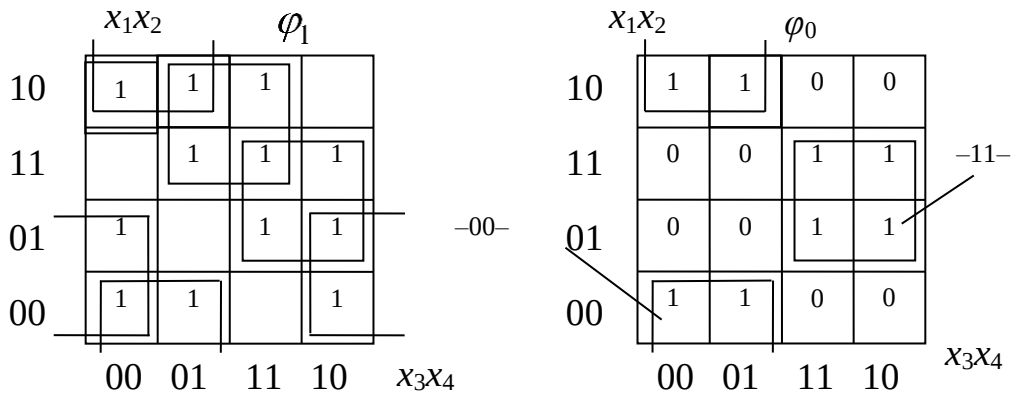


Рис. 3.3. Діаграми Вейча – Карно функцій φ_1 та φ_0

Діаграма Вейча – Карно, що характеризує функцію φ_0, містить результати операції пошуку вихідної частково визначеної функції f, які забезпечують одержання тупикової ДНФ такого вигляду: $f(x_1, x_2, x_3, x_4) = x_2x_3 + \bar{x}_2\bar{x}_3$.

Результат тут збігається з результатом мінімізації, отриманим за допомогою методу Мак-Класкі.

3.4. Пошук мінімальних КНФ

У деяких випадках схема, що реалізує мінімальну КНФ, виявляється більш економічною ніж та, що реалізує ДНФ тієї самої логічної функції. Тому з метою одержання оптимального розв'язку бажано знайти мінімальні ДНФ і КНФ логічної функції.

Описані вище методи мінімізації ДДНФ можуть бути застосовані й для мінімізації ДКНФ, якщо користуватися двоїстими формами відповідних логічних законів, а замість конституент одиниці вживати конституенти нуля.

Разом з тим на практиці дуже часто замість використання методів мінімізації КНФ застосовують підхід, що дозволяє виявляти тупикові КНФ за допомогою методів мінімізації ДДНФ і правила де Моргана.

Цей підхід характеризується трьома кроками, а саме:

Крок 1. Записуємо заперечення заданої функції як диз'юнкцію елементарних кон'юнкцій максимального рангу, що мають номери тих наборів, на яких логічна функція набуває нульового значення.

Крок 2. Отриманий вираз мінімізуємо за допомогою кожного з методів мінімізації ДДНФ. Внаслідок цього одержуємо мінімальні диз'юнктивні форми заперечення заданої функції.

Крок 3. Від мінімальних (тупикових) ДНФ заперечення заданої функції за допомогою правила де Моргана $\left(\bar{x}_1\bar{x}_2 = \bar{x}_1 + \bar{x}_2, \overline{x_1 + x_2} = \bar{x}_1\bar{x}_2\right)$ переходимо до тупикових КНФ заданої функції.

Приклад 3.8. Для записаної аналітично ДНФ знайти тупикову КНФ.

$$f(x_1, x_2, x_3, x_4) = \bigvee_1 (3,4,5,7,9,11,12,13).$$

Розв'язування

З умови задачі випливає, що функція f має нульове значення на таких наборах: 0, 1, 2, 6, 8, 10, 14, 15. Двійкові еквіваленти цих номерів мають такий вигляд: 0000, 0001, 0010, 0110, 1000, 1010, 1110, 1111.

Крок 1. Записуємо заперечення заданої функції:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 +$$
$$+ x_1x_2x_3\bar{x}_4 + x_1x_2x_3x_4.$$

Крок 2. Знайдемо тупикову ДНФ функції f за допомогою діаграми Вейча – Карно.

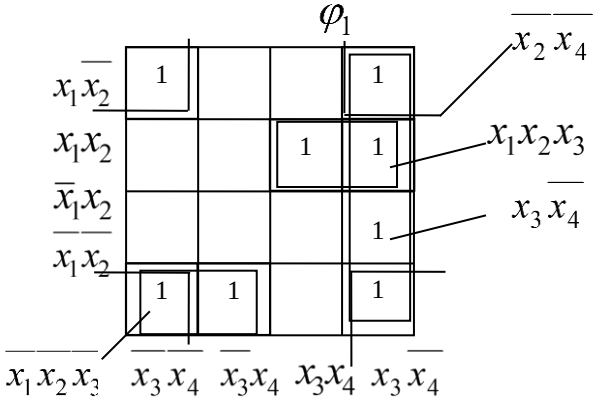


Рис. 3.4. Діаграма Вейча – Карно до прикладу 13

Після мінімізації функція набуває такого вигляду:

$$f\left(x_1x_2x_3x_4\right)=\overline{x_2}\,\overline{x_4}+x_3\overline{x_4}+x_1x_2x_3+\overline{x_1}x_2\overline{x_3}\,.$$

Крок 3. Використовуючи правило де Моргана, визначимо тупикову КНФ, а саме:

$$f\left(x_1x_2x_3x_4\right)=\overline{\overline{x_2}\,\overline{x_4}+x_3\overline{x_4}+x_1x_2x_3+\overline{x_1}x_2\overline{x_3}}=\overline{\overline{x_2}\,\overline{x_4}}\cdot\overline{x_3\overline{x_4}}\cdot\overline{x_1x_2x_3}\cdot\overline{\overline{x_1}x_2\overline{x_3}}=$$
$$=\left(x_2+x_4\right)\left(\overline{x_3}+x_4\right)\left(\overline{x_1}+\overline{x_2}+\overline{x_3}\right)\left(x_1+x_2+x_3\right).$$

3.5. Синтез логічних (комбінаційних) схем

Обладнання, яке реалізує елементарні булеві функції, називають *логічними елементами*. Їхні входи відповідають булевим змінним, а вихід – реалізованій функції. Види логічних елементів показано на рис. 3.3.

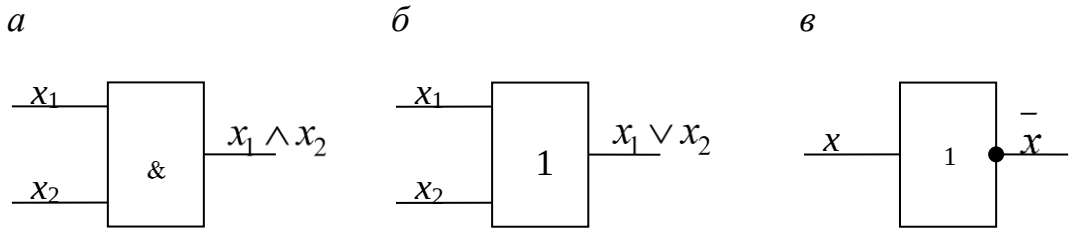


Рис. 3.5. Види логічних елементів: *а* – кон’юнктор, *б* – диз’юнктор, *в* – інвертор

Подібно до суперпозиції функцій, логічні схеми утворюються шляхом суперпозиції елементів за допомогою визначення їх зовнішніх вузлів (полюсів). Коректно побудовані логічні схеми повинні задовольняти такі умови:

– не допускати замкнених контурів, які можуть призвести до неоднозначності на виходах елементів;

– будь-який вхід елемента має бути з'єднаним тільки з одним входом схеми або виходом іншого елемента;

– виходи елементів, що не являють собою виходи схеми та не з'єднані із входами інших елементів, вважаються зайвими й вилучаються зі схеми.

Зауважимо, що записати булеву функцію для заданої логічної схеми досить легко. Так само просто будується й логічна схема даного аналітичного виразу булевої функції. Успішне проєктування логічних схем полягає в тому, щоб забезпечити *найбільш економічну* реалізацію булевої функції.

Опишемо алгоритм синтезу логічних схем.

Крок 1. Складаємо таблиці істинності синтезованого вузла згідно з його визначенням, призначенням і словесним описом принципу роботи.

Крок 2. Будуємо математичну формулу логічної функції, що описує роботу синтезованої схеми (вузла) згідно з наявною таблицею істинності.

Крок 3. Аналізуємо отриману функцію з метою побудови різних варіантів її математичного запису (на підставі законів булевої алгебри) та знаходимо найкращий з них відповідно до заданого критерію.

Крок 4. Складаємо функціональну (логічну) схему з елементів «І», «АБО», «НІ».

Алгоритм описано.

Побудова логічної схеми базується на прямому заміщенні елементарних добутків, сум і заперечень кон'юнкторами, диз'юнкторами й інверторами відповідно.

Для отримання аналітичного виразу заданої таблично (у досконалій диз'юнктивній нормальній формі) функції потрібно скласти суму конституент одиниці тих наборів значень вхідних двійкових змінних, для яких реалізації логічної функції f дорівнюють 1, причому символ будь-якої змінної в конституенті береться зі знаком заперечення, якщо конкретне значення змінної x_i у розглянутому наборі становить 0.

Для того, щоб записати аналітично функцію, задану таблично у досконалій кон'юнктивній нормальній формі, потрібно скласти логічний добуток конституент нуля тих наборів значень вхідних двійкових змінних, для яких реалізація функції f дорівнює 0, причому символ будь-якої змінної в конституенті береться зі знаком заперечення, якщо конкретне значення змінної x_i в розглянутому наборі дорівнює 1.

Приклад 3.9. Роботу трьох верстатів координують таким чином, що коли будь-які два з них вийдуть із ладу, то автоматично включається в роботу

четвертий (аварійний) верстат. Потрібно синтезувати логічну схему обладнання, яка керує включенням у роботу четвертого верстата.

Розв'язування

Крок 1. Ведемо позначення. Нехай змінні x_1, x_2, x_3 відображають стани верстатів 1, 2, 3 відповідно. Коли змінна x_i дорівнює 1, то це означає, що *i*-й верстат працює справно, а коли нулю – то верстат вийшов з ладу. Складемо таблицю істинності функції f. Одиничне значення функції відповідає включенню в роботу четвертого верстата.

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
f	1	1	1	0	1	0	0	0

Крок 2. З урахуванням даних складеної вище таблиці істинності запишемо ДДНФ і ДКНФ логічної функції, а саме:

$$f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2}\,\overline{x_3} + \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} + x_1\overline{x_2}\,\overline{x_3},$$

$$f(x_1, x_2, x_3) = \left(x_1 + \overline{x_2} + \overline{x_3}\right)\left(\overline{x_1} + x_2 + \overline{x_3}\right)\left(\overline{x_1} + \overline{x_2} + x_3\right)\left(\overline{x_1} + \overline{x_2} + \overline{x_3}\right).$$

Крок 3. Зробимо мінімізацію ДДНФ і ДКНФ, використовуючи діаграми Вейча – Карно (див. рис. 3.6).

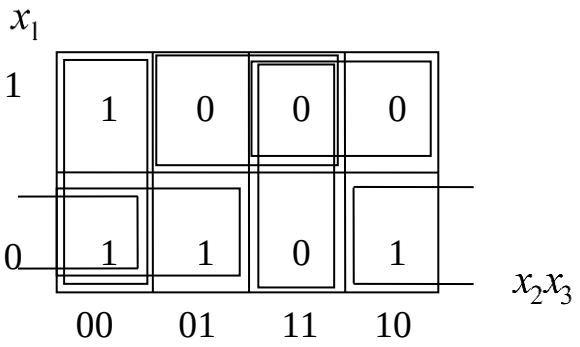


Рис. 3.6. Діаграма Вейча – Карно для функціїї з прикладу 14

Одержимо таку мінімальну ДНФ функції:

$$f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2} + \overline{x_1}\,\overline{x_3} + \overline{x_2}\,\overline{x_3}\ .$$

Беручи заперечення від виразу $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$ й застосувавши закони де Моргана, визначимо мінімальну КНФ в такому вигляді:

$$f\left(x_1x_2x_3\right)=\left(\overline{x_1}+\overline{x_2}\right)\left(\overline{x_1}+\overline{x_3}\right)\left(\overline{x_2}+\overline{x_3}\right).$$

Крок 4. Використовуючи мінімальні ДНФ і КНФ функції, побудуємо відповідні їм логічні схеми. Процес синтезу виконується у напрямку від вхідних значень змінних до отримання вихідних значень. Спочатку використовуючи вхідні змінні, отримують необхідні інверсні значення, а потім, поетапно застосовуючи необхідні елементи для реалізації операцій, маємо кінцеве значення функції. Отримані схеми подано на рис. 3.7 та 3.8.

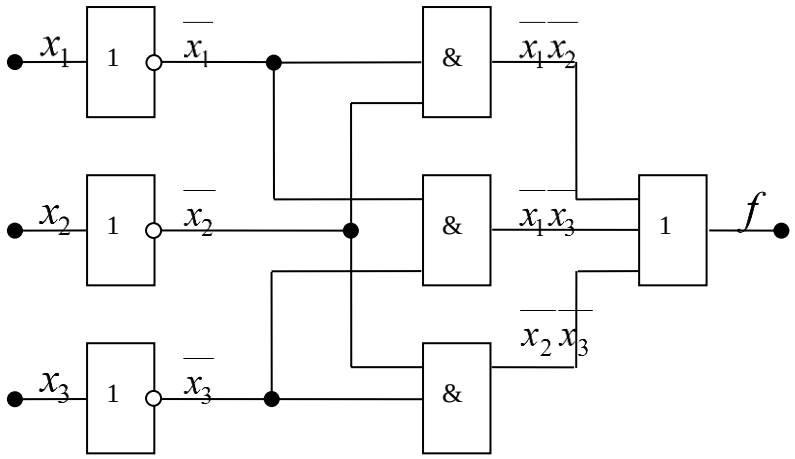


Рис. 3.7. Логічна схема, що базується на мінімальній ДНФ функції (приклад 14)

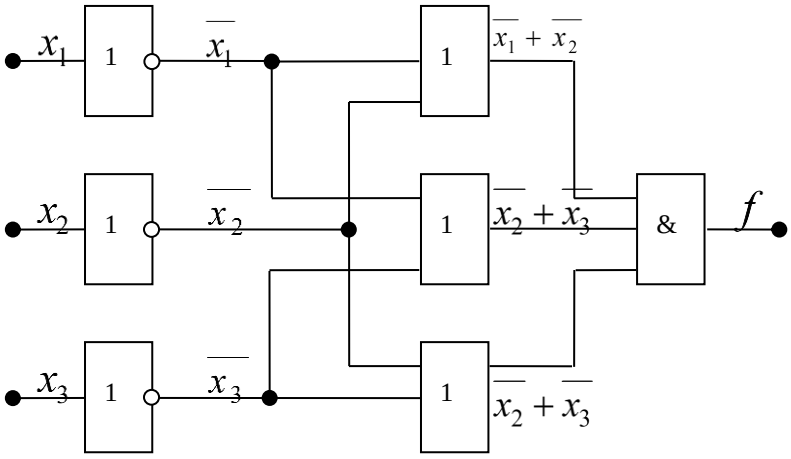


Рис. 3.8. Логічна схема, виконана на базі мінімальної КНФ (приклад 14)

3.6. Синтез скінченних автоматів

Скінченний автомат являє собою математичну модель пристрою із скінченною пам’яттю. Він переробляє вхідну множину впливів X у множину

вихідних сигналів Y, залежно від свого внутрішнього стану S, і характеризується тим, що множини X, Y та S, що містять усі можливі значення входів, виходів та внутрішніх станів, є скінченими. Іншими словами, існує *скінчений перелік* з n можливих значень сигналів на вході, m можливих значень сигналів на виході, а також k можливих внутрішніх станів системи.

Скінченний автомат дуже легко задати у вигляді таблиці розміру $n \times k$, у кожній клітинці якої в чисельнику міститься наступний внутрішній стан системи, а в знаменнику – сигнал на виході (див. рис. 3.9, *а*). Будь-який скінчений автомат можна також подати у вигляді орієнтованого графа, вершинами якого будуть внутрішні стани, а дугами – пари із вхідного і вихідного сигналів (див. рис. 3.9, *б*).

З викладеного раніше випливає, що автомати повинні містити елементи пам'яті, аби зберігати значення стану системи від такту до такту, один чи декілька вхідних та вихідних каналів.

а

$s(n)$ \ $x(n)$	x_1	x_2	x_3
S_1	S_2/y_1	S_2/y_2	S_3/y_2
S_2	S_3/y_2	S_2/y_1	S_1/y_2
S_3	S_1/y_2	S_2/y_1	S_3/y_1

б

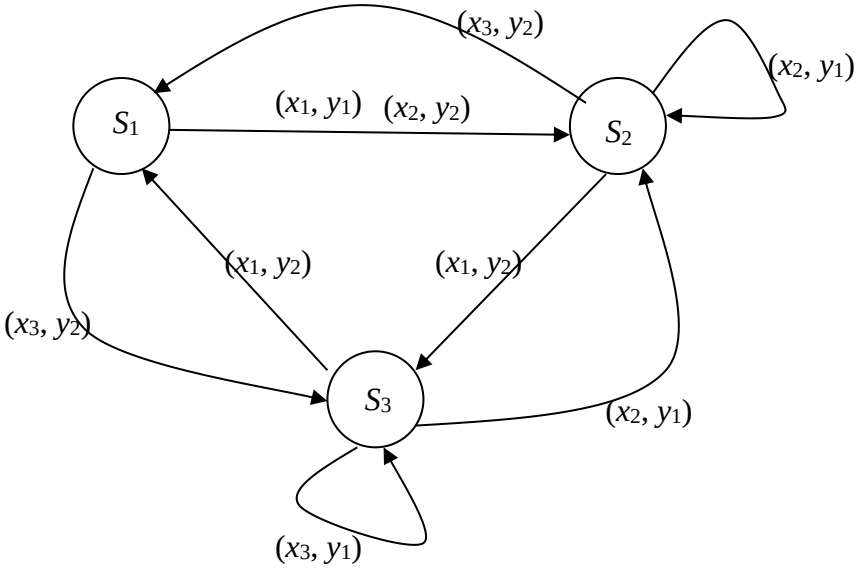


Рис. 3.9. Приклад задання скінченного автомата:
а – за допомогою таблиці переходів; *б* – за допомогою графа

Реалізація скінченних автоматів зводиться до синтезу відповідної комбінаційної схеми, що перетворює вхідні змінні $x(n)$ й $s(n)$ на вихідні змінні $s(n+1)$ й $y(n)$ із заданими характеристичними функціями, тобто

$$s(n+1) = f\left(s(n), x(n)\right),$$

$$y(n) = \varphi\left(s(n), x(n)\right).$$

Щоб зберегти стан $s(n+1)$ до наступного такту, у ланцюг зворотного зв'язку потрібно ввести необхідну кількість елементів пам'яті.

При реалізації автоматів у двійковому структурному алфавіті можна використовувати розглянуті вище методи синтезу комбінаційних схем. Для цього необхідно закодувати кожен стан схеми й записати характеристичні функції у вигляді булевих функцій двійкових змінних. Таке кодування можна здійснити шляхом перетворення загальної таблиці переходу автомата до таблиці істинності в двійковому структурному алфавіті. Якщо елементи множин X, Y, S пронумеровано порядковими числами, починаючи з нуля, то їм відповідають коди, котрі являють собою двійкові еквіваленти цих чисел.

Приклад 3.10. Синтезувати скінченний автомат, заданий загальною таблицею переходів, яку подано нижче.

$s(n)$ \ $x(n)$	0	1	2	3
0	3/0	2/0	1/1	3/0
1	3/0	2/0	1/1	3/0
2	3/0	2/0	2/0	3/0
3	3/0	0/1	0/1	1/1

Розв'язування

Запишемо вихідну таблицю переходів скінченного автомата в такому вигляді:

$x(n)$	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
$s(n)$	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
$s(n+1)$	3	3	3	3	2	2	2	0	1	1	2	0	3	3	3	1
$y(n)$	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	1

Замінюючи десяткові числа їх двійковими еквівалентами, одержуємо таблицю істинності (табл. 3.4), у якій $x(n), s(n)$, $s(n+1)$ і $y(n)$ записано двійковими кодами.

Таблиця 3.4

$x(n)$		$s(n)$		$s(n+1)$		$y(n)$
$x_1(n)$	$x_2(n)$	$s_1(n)$	$s_2(n)$	$s_1(n+1)$	$s_2(n+1)$	
0	0	0	0	1	1	0
0	0	0	1	1	1	0
0	0	1	0	1	1	0
0	0	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	0	0	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	0	1	1

Із цієї таблиці видно, що комбінаційна схема скінченного автомата повинна мати чотири входи, відповідні вхідним змінним $x_1(n)$, $x_2(n)$ і змінним стану $s_1(n)$, $s_2(n)$, а також три виходи, які відповідають змінним стану $s_1(n+1)$, $s_2(n+1)$ і виходу $y(n)$.

Користуючись розглянутим у § 3.5 алгоритмом, синтезуємо комбінаційну схему скінченного автомата. Для цього проведемо роздільну мінімізацію функцій $s_1(n+1)$, $s_2(n+1)$ і $y(n)$ за допомогою діаграм Вейча – Карно (див. рис. 3.10).

Наслідком мінімізації буде такий вигляд функцій:

$$s_1(n+1) = x_2(n)\cdot\overline{s_1}(n) + \overline{x_1}(n)\overline{x_2}(n) + s_1(n)\cdot\overline{s_2}(n);$$

$$s_2(n+1) = \overline{x_1}(n)\cdot\overline{x_2}(n) + x_1(n)x_2(n) + x_1(n)\cdot\overline{s_1}(n);$$

$$y(n) = \overline{s_1}(n+1).$$

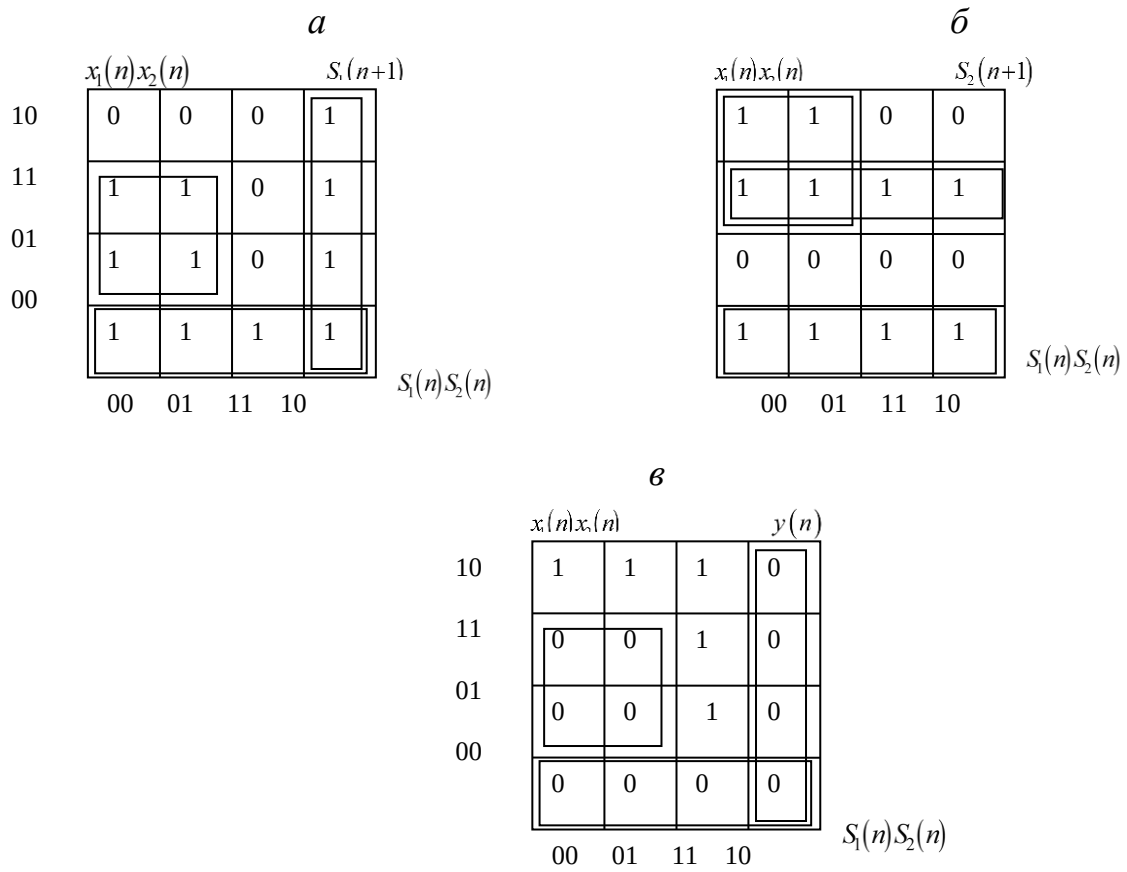


Рис. 3.10. Мінімізація функцій за допомогою діаграм Вейча – Карно:
$а - s_1(n+1)$; $б - s_2(n+1)$; $в - y(n)$

Синтезувавши комбінаційну схему, відповідну вихідній таблиці, а також увівши два елементи затримки C_1 і C_2, одержимо структурну схему скінченного автомата (див. рис. 3.11).

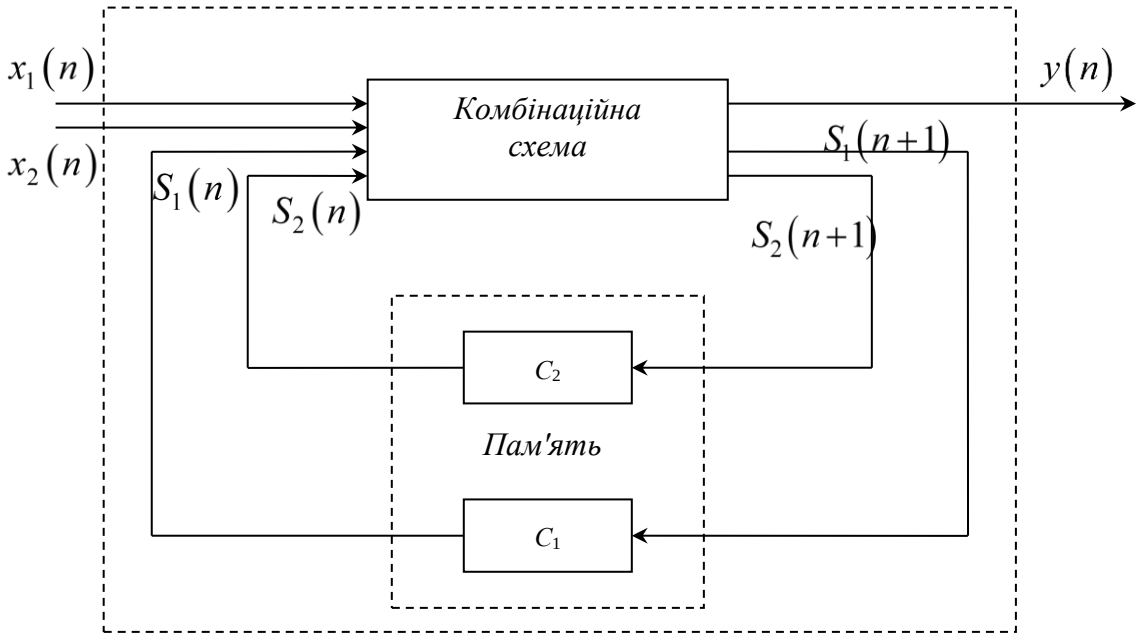


Рис. 3.11. Структурна схема скінченного автомата (приклад 15)

Виходячи з того, що праві частини логічних виразів для одержання функцій $s_1(n+1)$ й $s_2(n+1)$ містять загальний член $\overline{x_1}(n)\cdot\overline{x_1}(n)$, а $y(n)=\overline{s_1}(n+1)$, схема скінченного автомата може бути також подана у такому вигляді (див. рис. 3.12):

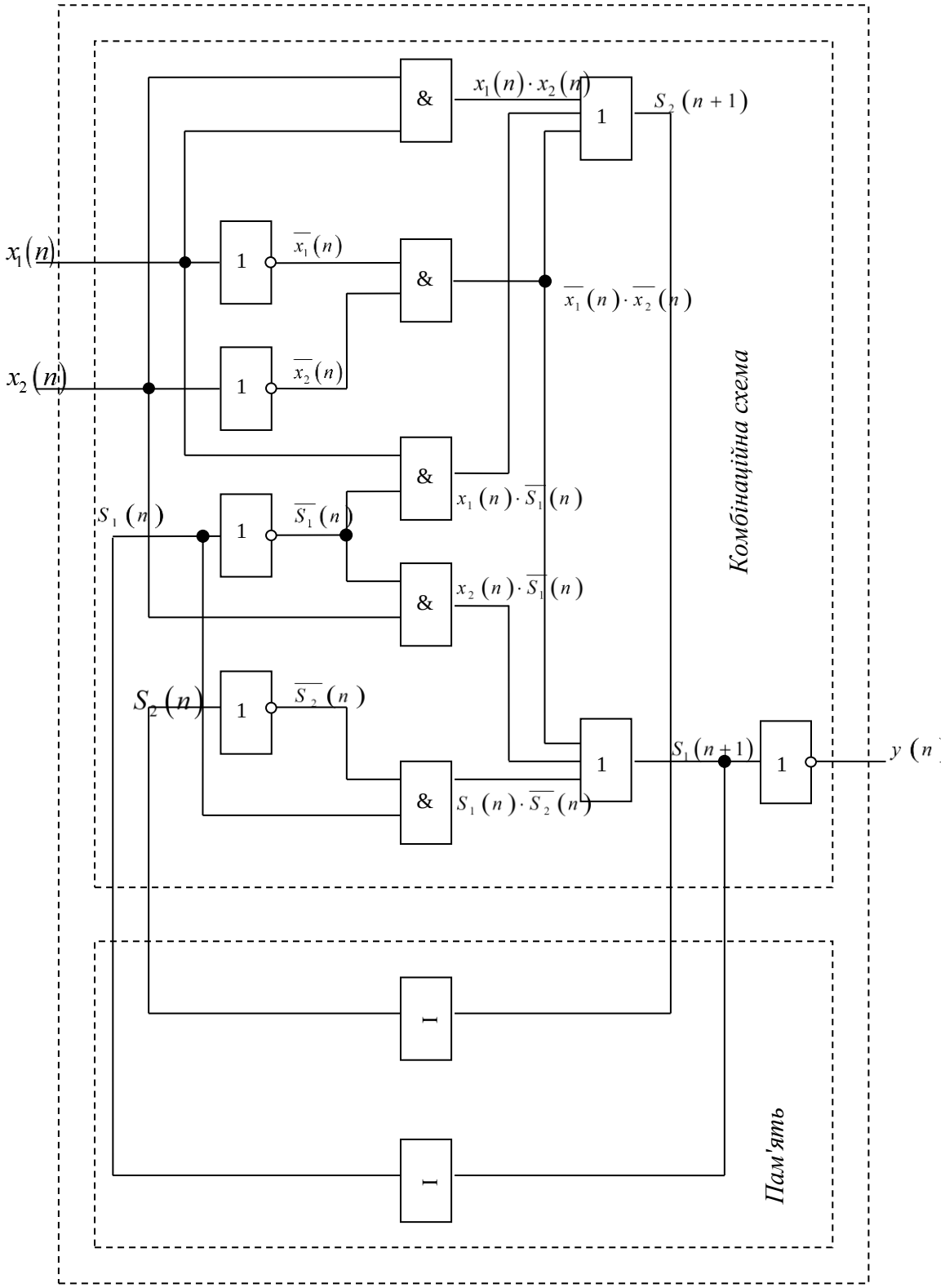


Рис. 3.12. Схема скінченного автомата (до прикладу 15)

Висновки

Скінченний автомат являє собою математичну модель пристрою із скінченною пам'яттю. Він переробляє вхідну множину впливів X у множину вихідних сигналів Y, залежно від свого внутрішнього стану S, і характеризується тим, що множини X, Y та S, що містять усі можливі значення входів, виходів та внутрішніх станів, є скінченими. При реалізації автоматів у двійковому структурному алфавіті використовуються методи синтезу комбінаційних схем і математичний апарат булевих функцій. Оскільки двозначна логіка дозволяє математично описати реальні об'єкти, які перебувають в одному із двох можливих станів. Вони описуються за допомогою *булевих змінних*, кожна з яких має лише два можливі значення – нуль або одиницю.

Основними в булевій алгебрі виступають три функції: з*аперечення*, д*из'юнкція*, *кон'юнкція*.

Одна й та сама логічна функція може мати різну форму запису. Особливо виділяють диз'юнктивну й кон'юнктивну нормальні форми (ДНФ і КНФ).

Диз'юнктивною нормальною формою (ДНФ) називається диз'юнкція скінченного числа різних членів, кожний з яких являє собою кон'юнкцію окремих змінних або їх заперечень, що входять у даний член не більше одного разу.

Кон'юнктивною нормальною формою (КНФ) називається кон'юнкція скінченного числа різних членів, кожний з яких являє собою диз'юнкцію окремих змінних або їх заперечень, котрі входять у даний член не більше одного разу.

Будь-яку логічну функцію можна звести до ДНФ або до КНФ.

Булева функція може бути записана по-різному, тому виникає потреба пошуку найбільш компактної форми її подання. Ця процедура називається *мінімізацією логічних функцій*. Для розв'язування задачі мінімізації логічна функція попередньо зводиться до ДДНФ або ДКНФ.

Найпоширеніші методи мінімізації – Квайна, Квайна – Мак- Класкі та Вейча – Карно.

Розглянуті питання основою теорії інформаційних систем та мають широке застосування у комп'ютерних науках та криптографії: спеціальні форми зображення булевих функцій у алгебрі Буля, повнота системи булевих функцій, мінімізація булевих функцій.

Питання, викладені в цьому розділі, розглянуто у літературі [1, 3, 4, 5, 6, 9, 10, 11, 13, 16, 19, 20, 21, 22]

Питання для самоконтролю

1. Яких значень можуть набувати булеві змінні й функції?
2. Яким чином визначають основні булеві функції однієї та двох змінних?
3. У якому випадку дві булеві функції будуть рівносильними?

4. Який запис булевих функцій називається досконалою диз'юнктивною (кон'юнктивною) нормальною формою?

5. Який закон алгебри логіки лежить в основі методів мінімізації булевих функцій?

6. Які методи мінімізації логічних функцій ви знаєте?

7. Що являє собою код Грея?

8. Що являє собою частково визначена логічна функція?

9. Який вигляд має загальна структура скінченного автомата?

10. У чому полягає відмінність скінченного автомата від комбінаційної схеми?

Задачі для самостійного розв'язування

1. Використовуючи основні тотожності булевої алгебри, записати дану функцію у вигляді досконалої диз'юнктивної нормальної форми.

$$y = \overline{x_1} \vee \left(\overline{x_2 \wedge x_3}\right) \vee x_1 \wedge \overline{x_3}.$$

2. Застосовуючи основні тотожності булевої алгебри, записати дану функцію у вигляді досконалої кон'юнктивної нормальної форми.

$$y = \left(\overline{x_1 \wedge x_2}\right) \wedge \overline{x_3} x_2 \vee \overline{x_1 \wedge x_2}.$$

3. За допомогою таблиць істинності перевірити тотожність таких функцій:
$y_1 = \left(x_1 \vee x_2\right) \wedge \overline{x_3}$ і $y_2 = \overline{\left(\overline{x_1} \wedge \overline{x_2}\right) \vee x_3}$.

4. Застосовуючи метод Квайна, знайти тупикову форму для диз'юнктивної нормальної форми такої функції:
$f\left(x_1, x_2, x_3, x_4\right) = \overline{x_1} x_3 + \overline{x_2} x_3 x_4 + x_1 \overline{x_3}\,\overline{x_4} + \overline{x_1}\,\overline{x_2} x_3 \overline{x_4} + x_1 x_2 \overline{x_3}\,\overline{x_4} + x_1 \overline{x_2}\,\overline{x_3} x_4$.

5. За допомогою методу Квайна – Мак-Класкі знайти тупикову ДНФ такої функції:

$$f\left(x_1, x_2, x_3, x_4\right) = x_2 x_4 + x_1 \overline{x_2} x_3 + x_1 \overline{x_3}\,\overline{x_4} + x_1 x_2 \overline{x_3} x_4 + \overline{x_1} x_2 \overline{x_3} x_4 + x_1 \overline{x_2}\,\overline{x_3} x_4.$$

6. Задану нижче логічну функцію записати у вигляді ДДНФ, а потім мінімізувати методом Вейча – Карно.

$$f\left(x_1, x_2, x_3, x_4\right) = \overline{x_1 x_2} + x_1 \overline{x_2} \left(\overline{x_3 + x_2 x_4}\right) + \overline{x_1} x_2 x_4 + x_1 \overline{x_2 x_3} x_4.$$

7. Методом Вейча – Карно знайти тупикову форму такої частково визначеної функції чотирьох змінних:

$$f\left(x_1, x_2, x_3, x_4\right) = \bigvee_1 \left(1^*, 2, 3^*, 5^*, 6, 7^*, 8^*, 9, 11^*, 12^*, 13, 14\right).$$

Тут символом «*» позначено десяткові номери наборів, на яких функцію не визначено.

8. Методом Вейча – Карно знайти тупикову форму записаної нижче функції чотирьох змінних й побудувати відповідну їй комбінаційну схему.

$$f\left(x_1, x_2, x_3 x_4\right) = \bigvee_1 \left(0, 2, 3, 5, 6, 8, 10, 11, 13, 14\right).$$

9. На основі заданої таблиці істинності функції трьох змінних $f\left(x_1, x_2, x_3\right)$ скласти відповідну їй комбінаційну схему. Функцію $f\left(x_1, x_2, x_3\right)$ спочатку мінімізувати одним із відомих методів.

x_1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1
f	1	1	1	0	0	0	1	1

10. Синтезувати скінченний автомат, заданий загальною таблицею переходів.

$X(n)$ / $S(n)$	0	1	2	3
0	2/1	3/0	2/1	2/1
1	2/1	3/0	1/1	1/0
2	2/1	1/1	0/0	0/0
3	2/1	1/1	0/1	0/1

РОЗДІЛ 4

МАТЕМАТИЧНА ЛОГІКА І ФОРМАЛЬНІ СИСТЕМИ

Мета розділу: опанування змісту основних понять математичної логіки та теорії формальних систем.

Поняття «логіка» бере початок від Аристотеля і характеризує науку, яка вивчає форми і закони мислення, а також методи, за допомогою яких люди роблять висновки, встановлюють зв'язок логічних форм із мовою.

Логічні форми та категорії є результатом суспільної практики. Пізнання людиною навколишньої дійсності під час її діяльності уможливило формулювання законів логіки, яка являє собою в певному сенсі відображення дійсності. Отже, логіка є наукою про закони та форми правильного мислення. Вона вивчає форми міркувань, відволікаючись від їх конкретного змісту, встановлює, що з чого випливає, шукає відповідь на питання про спрямування міркувань.

Математична (символічна або теоретична) логіка є розширенням традиційної. З одного боку, ця наука застосувала математичні методи вивчення загальних структур (форм) правильного мислення і тим самим стала розділом математики, з іншого – предметом її вивчення є також процес доведення математичних теорем і самі математичні теорії. Отже, математична логіка є також інструментом для досліджень у галузі основ математики. Цей розділ математичної логіки отримав назву «теорія доведення» чи «метаматематика».

Таким чином, можна сказати, що логіка – це наука про закони мислення, а математична логіка є наукою про закони математичного мислення.

4.1. Вступ у формальні системи

Формальними називаються системи операцій над об’єктами, котрі являють собою послідовність символів (тобто слова в певному визначеному алфавіті). Отже, операції здійснюються над символами. Термін «формальний» підкреслює той факт, що об’єкти й операції над ними розглядаються формально, без будь-яких змістовних інтерпретацій символів. Передбачається також, що між символами не існує жодних зв’язків і відношень крім тих, що явно описані засобами самої формальної системи.

Якщо вам запропонують впорядкувати об’єкти 53, 109, 3, то, скоріш за все, без усяких додаткових питань ви розташуєте їх у такому порядку: 3, 53,109. Інакше кажучи, цій задачі буде дана звичайна арифметична інтерпретація, а саме: послідовності цифр вважаються зображенням чисел у звичайній десятковій системі. Упорядкуванням цих послідовностей буде розташування чисел за зростанням, а правило порівняння таких зображень чисел відомо настільки добре, що зазвичай про нього ніхто не замислюється.

У дійсності таке упорядкування не є очевидним. Можливість неоднозначного виведення задач із тексту обмежень означає, що текст не містить формального визначення задачі. Для такого визначення необхідно чітко описати клас об'єктів, для яких задача розв'язується, явно ввести для них поняття упорядкування, охарактеризувати його як систему локальних операцій над символами, із котрих ці об'єкти складаються.

По суті при такому розумінні формальний опис системи означає її потоковий, явний опис – усе, що є істотним для розв'язування задачі, має бути описано явно. Таке уточнення задачі зазвичай називають її *формалізацією*.

Подібні уявлення про те, що є точним описом можна простежити на прикладі таких понять як «алгоритм», «дані» і т.п. У визначеному сенсі проблему точного опису деякої множини можна розглядати як проблему побудови алгоритму, котрий обчислює або породжує цю множину.

Так, приміром, визначення за допомогою формули – це опис обчислювальної множини у котрому використані всі істотні складові частини поняття алгоритму, крім одного – детермінованості. Відкидаючи несуттєвий тут порядок обчислення елементів множини, ми виграємо в компактності опису, який при цьому не стає менше відкритим. Такий опис, не будучи алгоритмом, являє собою формальну систему, що однозначно описує множину формул.

Історично теорія формальних систем виникла в межах основ математики при дослідженні побудови аксіоматичних теорій і методів доведення у них. З їхнього вивчення і починається знайомство з формальними системами.

4.2. Принципи побудови формальних теорій

Будь-яка точна теорія визначається, по-перше, мовою, тобто певною множиною висловлювань, котрі мають зміст з точки зору готової теорії, і, по-друге, сукупністю теорем – множиною мови, що складається із висловлювань, істинних у даній теорії.

Яким чином теорія одержує свої теореми?

У математиці з античних часів існував зразок систематичної побудови теорії – геометрія Евкліда, у якій усі вихідні передумови сформульовані явно, у вигляді аксіом, а з них вже виводяться теореми за допомогою ланцюжків логічних міркувань, котрі називаються доказами. Проте до середини 19 сторіччя математичні теорії, як правило, не вважали потрібним явно виділяти дійсно усі вихідні принципи. Критерії ж строгого доказу й очевидності тверджень у математиці в різні часи були різноманітними і також явно не формулювалися. Час від часу це призводило до необхідності перегляду основ тієї або іншої теорії. Відомо, наприклад, що основи диференціального й інтегрального числень, розроблені у 18-му сторіччі Ньютоном і Лейбніцем, у 19 столітті піддалися серйозному перегляду; математичний аналіз у його сучасному вигляді спирається на роботи Коші, Больцано, Веєрштрасса з теорії границь. Наприкінці 19 сторіччя перегляд торкнувся й загальних принципів організації математичних теорій.

Це спричинило створення нової галузі математики, так званих *основ математики*, предметом якої стала саме побудова математичних теорій і тверджень. Вона поставила своєю метою відповісти на питання типу: «яким чином має бути побудована теорема, щоб у ній не виникало протиріч?», «які властивості повинні мати методи доведення, аби їх можна було вважати достатньо строгими?».

Однієї з фундаментальних ідей, на яку спираються дослідження з основ математики, є формалізації теорій, тобто послідовне проведення аксіоматичного методу їх побудови. При цьому не дозволяється користуватися якимись припущеннями про об'єкти теорії, крім тих, що описано явно у вигляді аксіом; аксіоми розглядаються як формальні послідовності символів (виразів), а доведення – як методи одержання одних виразів з інших за допомогою операцій над символами. Такий підхід гарантує чіткість вихідних тверджень і однозначність висновків, проте, може скластися враження, що осмисленість і істинність у формальних теоріях не відіграє ніякої ролі. У дійсності й аксіоми, і правила виводу прагнуть вибирати таким чином, щоб побудована з їхньою допомогою формальна теорія мала змістовний сенс.

Більш конкретно формальна теорія (або числення) будується в такий спосіб:

1. Визначають множину формул або правильно побудованих виразів, які утворюють мову теорії.

Цю множину задають конструктивними засобами (зазвичай, індуктивним визначенням). Як правило, ця множина нескінченна і часто *розв'язувана*.

2. Виділяють підмножину формул, які називаються *аксіомами теорії*.

Підмножина аксіом може бути скінченною і нескінченною. Якщо множина аксіом є нескінченною, то, як правило, вона задається за допомогою скінченної множини схем аксіом та правил породження конкретних аксіом зі схеми аксіом. Зазвичай аксіоми поділяються на два види: *логічні* (загальні для цілого класу формальних теорій або систем) та *нелогічні* (або власні), які визначають специфіку та зміст конкретної теорії або системи. Множина аксіом, здебільшого, має бути розв'язною.

3. Задають *правила виводу теорії*.

Правило виводу $R(F_1, \dots, F_n, \sigma)$ являє собою певне відношення на множині формул. Якщо формули $F_1, \dots, F_n, \sigma$ знаходяться у відношенні R, то формула σ називається безпосередньо виведеною з $F_1, \dots, F_n$ за правилом R.

Часто правило $R(F_1, \dots, F_n, \sigma)$ записують у такому вигляді:

$$\frac{(F1, \dots, Fn)}{\sigma} R.$$

Формули $F_1, \dots, F_n$ називаються *припущеннями, посилками або гіпотезами* правила R, а σ – його *наслідком* або *висновком*. Приклади аксіом і правил виводу будуть наведені трохи пізніше.

Виведенням (виводом) формули B з формул $A_1, \ldots, A_n$ називається послідовність формул $F_1, \ldots, F_m$, де $F_m = B$, а будь-яка формула F_i ($i = 1, 2, \ldots, m$) є або аксіомою, або однією з вихідних формул $A_1, \ldots, A_n$, або безпосередньо виведена з формул $F_1, \ldots, F_{i-1}$ (або із будь якої їхньої підмножини) за одним із правил виведення. Якщо існує виведення U з $A_1, \ldots, A_n$, то це свідчить, що B виведена з $A_1, \ldots, A_n$ Цей факт позначається таким чином:

$$A_1, \ldots, A_n \vdash B.$$

Формули $A_1, \ldots, A_n$ називаються *гіпотезами* або *посилками* виведення. Перехід у виведенні від F_{i-1} до F_i є i-м кроком виведення.

Доведенням формули B у теорії T називається виведення B з порожньої множини формул, тобто виведення, у якому вихідними формулами є тільки аксіоми.

Формула B, для якої існує доведення, називається формулою, доказуваною (виведеною) в теорії T, або *теоремою* теорії T; факт доведеності формули B позначається у такий спосіб: $\vdash B$.

Очевидно, що приєднання формул до гіпотез не порушує виведеності, оскільки, коли $\vdash B$, то $A \vdash B$, і якщо $A_1, \ldots, A_n \vdash B$, то $A_1, \ldots, A_n, A_{n+1} \vdash B$ для будь-яких $A_1, \ldots, A_n$ і A_{n+1}. Порядок гіпотез у списку несуттєвий.

При вивченні формальних теорій ми маємо справу з двома типами висловлювань. Передусім, з тими, що є елементами самої теорії (теореми), і розглядаються як чисто формальні об'єкти, визначені раніше; а також з висловлюваннями про теорію (властивостями її теорем, доведень і т. і.), які формулюються на мові, що є зовнішньою стосовно теорії (*метамова теорії*), і називаються *метатеоремами.*

Різницю між теоремами і метатеоремами не завжди описано явно, але її обов'язково потрібно брати до уваги.

Наприклад, якщо вдалося побудувати виведення B з $A_1, \ldots, A_n$, то твердження «$A_1, \ldots, A_n \vdash B$» є метатеоремою. Її можна розглядати як додаткове («довільне») правило виводу, яке можна приєднати до початкових правил і використовувати у подальших конструюваннях.

Ясно, що загальнозначущі (тотожньо-істинні) висловлення типу $A \vee \bar{A}$ або $\forall x\, P(x) \Rightarrow P(Y)$, які мають силу загальних логічних законів, мають входити у склад будь-якої теорії, котра претендує на логічний сенс. Тому вивчення конкретних формальних теорій почнемо із числень, які породжують усі загальновідомі формули.

4.3. Числення висловлень. Аксіоми і правила виводу

У численні висловлювань ми знову зустрічаємося з об'єктами, із котрими вже мали справу, а саме, із формулами алгебри логіки. Проте формули ми тут будемо розглядати не як засіб подання функцій, а як складові висловлювання, утворені з елементарних висловлювань (змінних) за допомогою логічних

операцій (зв'язок): $\vee$ (або), $\&$ (і), $\neg$ (заперечення) , $\Rightarrow$ (якщо X, то Y). Особлива увага при цьому приділяється тотожньо-істинним висловлюванням, оскільки, як уже відзначалося, вони мають входити в будь-яку теорію як загальнологічні закони. Їхнє породження і є основною задачею числення висловлювань.

Аби визначити числення висловлювань, ми маємо задати його складові як формальну систему: алфавіт, формули, аксіоми та правила виведення.

Алфавіт числення висловлювань складається з пропозиційних висловлювань (пропозиційних літер): A, B, C..., знаків логічних зв'язок (пропозиційних зв'язок) $\vee$, $\&$, $\neg$, $\Rightarrow$ і допоміжні символи – дужки – (,).

Формули:

а) пропозиційне висловлення є формулою;

б) якщо τ і π – формули, то $(\tau \vee \pi)$, $(\tau \& \pi)$, $(\tau \Rightarrow \pi)$ і $\neg \tau$ також є формулами;

в) інших формул немає.

Зовнішні дужки у формулах зазвичай опускаються: наприклад, пишуть не $(\tau \vee \pi)$, а $\tau \vee \pi$. Замість синтаксично більш зручного знака $\neg$ часто вживають риску над формулою.

Аксіоми. Наведемо тут дві системи аксіом. Перша з них безпосередньо використовує всі логічні зв'язки:

Система аксіом I (аксиоматизація Кліні)

I 1. $A \to (B \to A)$;

I 2. $((A \to (B \to C)) \to ((A \to B) \to (A \to C))$;

I 3. $(A \,\&\, B) \to A$

I 4. $(A \,\&\, B) \to B$;

I 5. $A \to (B \to (A \,\&\, B))$;

I 6. $A \to (A \vee B)$;

I 7. $B \to (A \vee B)$;

I 8. $(A \to C) \to ((B \to C) \to ((A \vee B) \to C)))$;

I 9. $(A \to B) \to ((A \to \neg B) \to \neg A)$;

I 10. $\neg \neg A \to A$.

Інша система використовує тільки дві зв'язки $\neg$ і $\to$. При цьому скорочується алфавіт числення (викидаються знаки $\vee$, $\&$) і відповідно визначення формули. Операції $\vee$, $\&$ розглядаються не як зв'язки числення висловлювань, а як скорочення (їх вживати зручно, але не обов'язково) для деяких його формул: $A \vee B$ замінюємо на $\neg A \to B$, $A \,\&\, B$ замінюємо на $\neg (A \to \neg B)$

Система аксіом II (аксиоматизація Лукашевича)

II 1. $A\to (B\to A)$;

II 2. $(A\to (B\to C))\to ((A\to B)\to (A\to C)$;

II 3. $(\neg A\to \neg B)\to ((\neg A\to B)\to A)$.

Наведені системи аксіом рівносильні в тому сенсі, що породжують одну і ту саму множину формул. Яка із систем краще? Це залежить від точки зору. Система II компактніша й більш наглядна. Відповідно, більш компактні і доведення різноманітних її властивостей. З іншого боку, у більш багатій системі I виведення різноманітних формул коротші.

Правила виведення

1) Правило підстановки. Якщо u – виведена формула, що містить букву A (позначимо цей факт через $u(A)$), то виведена формула $u(\beta)$ утворюється з u заміною усіх входжень A на довільну формулу β, а саме:

$$\frac{u(\mathrm{A})}{u(\beta)}.$$

2) правило виведення (Modus Ponens). Якщо u і $u\to\beta$ являють собою виведені формули, то формула β також буде виведеною:

$$\frac{u, u\to\beta}{\beta}.$$

У цьому описі числення висловлювань аксіоми є формулами числення (відповідно до визначення формули); формули ж використані в правилах виводу (u, та u $\to$ β і т.і.), це ”метаформули“ або *схеми формул*. Схема формул, наприклад $u\to\beta$, означає множину всіх тих формул числення, що утворюються, якщо її метазмінні замінити формулами числення: наприклад, якщо u замінити на A, а β – на $A\&B$, то зі схеми формул $u\to\beta$ одержимо формулу $A\to A\&B$.

4.4. Числення предикатів і теорії першого порядку

Числення предикатів — це формальна система в математичній логіці, в якій допускаються висловлювання відносно змінних, фіксованих функцій і предикатів. Вона являє собою розширення логіки висловлювань.

Аксіоми і правила виводу.

1. **Алфавіт** числення предикатів складається з пропозиційних змінних x_1, x_2, … , предметних констант a_1, a_2, … , предикатних букв $P_1^1, P_2^1, P_3^1, \cdots P_k^j$,

функціональних букв $f_1^1, f_2^1, f_3^1, \cdots f_k^j$, а також знаків логічних зв'язок $\vee$, &, $\neg$, $\rightarrow$, кванторів загальності $\forall$ та існування $\exists$ і дужок (,).

Верхні індекси предикатних і функціональних букв показують число аргументів, їхні нижні індекси слугують для звичайної нумерації букв. Змінні висловлювання у численні предикатів вводяться або як 0-арні предикати P_1^0, P_2^0, P_3^0 , … , тобто як предикати без пропозиційних змінних.

2. **Формули.** Поняття формули визначається у два етапи.

1) *Терми:*

а) пропозиційні змінні і константи є термами;

б) якщо f_n – функціональна буква, а t_1, … , t_n – терми, то $f_n(t_1, \dots , t_n)$ також терм.

2) *Формули*:

а) якщо P^n – предикатна буква, а t_1, … , t_n – терми, то $P^n(t_1, \dots , t_n)$ також буде формулою; усі входження пропозиційних змінних у формулу $P^n(t_1, \dots , t_n)$ називаються вільними;

б) якщо F_1, F_2 – формула, то формулами будуть також такі вирази: $\neg F_1$, $(F_1 \& F_2)$, $(F_1 \vee F_2)$, $(F_1 \rightarrow F_2)$; усі входження змінних, які вільні у F_1, F_2, є такими і в зазначених чотирьох видах формул;

в) якщо $F(x)$ – формула, яка містить вільні входження змінної x, то $\forall x F(x)$ і $\exists x F(x)$ є також формулами; у них всі входження змінної x називаються *пов'язаними*; входження інших змінних в F залишаються вільними.

3) *Аксіоми числення предикатів* поділяють на дві групи:

1) аксіоми числення висловлювань (можна взяти будь-яку із систем I або II);

2) предикатні аксіоми:

$$\text{P1.}\ \ \forall x\, F(x) \rightarrow F(y);$$

$$\text{P2.}\ \ F(y) \rightarrow \exists x\, F(x).$$

У цих аксіомах $F(x)$ знаходиться в області дії квантора за y; формула $F(y)$ отримана з $F(x)$ заміною усіх вільних входжень x на y.

Щоб пояснити істотність вимоги до входжень x в F, розглянемо приклад, коли $F(x)$ являє собою формулу $\exists y\, P(y, x)$, де ця вимога порушена: тобто вільне входження змінної x знаходиться в області дії квантора $\exists$ для y. Підстановка цієї формули в аксіому P1 дає формулу $\forall x \exists y\, P(y, x) \rightarrow \exists y\, P(y, y)$. Якщо її розглянути на множині N натуральних чисел із предикатом P – «бути більше», то одержимо таке висловлювання: «якщо для всякого x знайдеться y більший, ніж x, то знайдеться y, більший самого себе». Посилка цієї імплікації істинна на множині N, а її висновок хибний, і, отже, саме висловлювання є хибним.

4. Правила виведення.

1) правило виведення (Modus Ponens) – таке саме, що й у численні висловлювань;

2) правило узагальнення (GEN) ($\forall$ - введення), а саме

$$F \rightarrow G(x),$$

$$F \rightarrow \forall x \; G(x),$$

де $G(x)$ містить вільні входження x, а F їх не містить;

3) правило $\exists$ - введення:

$$\frac{G(x) \rightarrow F}{\exists x \, G(x) \rightarrow F}$$

за тих самих вимог до F і G, що й у попередньому правилі.

Порушення цих вимог можуть призвести до помилкових висновків із істинних висловлень.

Наприклад, $P(x)$ – предикат «x ділиться на 6», $Q(x)$ – предикат «x ділиться на 3». Тоді висловлювання $P(x) \rightarrow Q(x)$, очевидно, є істинним для довільного x, проте застосування до нього правила узагальнення дає висловлювання $P(x) \rightarrow \forall x \, Q(x)$, що не є завжди істинним.

Якщо ж до висловлювання $P(x) \rightarrow Q(x)$ застосувати правило $\exists$- введення, то одержимо висловлювання $\exists x \; P(x) \rightarrow Q(x)$,) із котрого шляхом (вже коректного!) застосування правила узагальнення одержимо такий вислів: $\exists x \, P(x) \rightarrow \forall x \, Q(x)$, що є хибним на множині натуральних чисел.

Розглянемо приклади формальних систем.

Історично формальні системи створювалися з конкретною метою більш точного обґрунтування методів побудови математичних теорій. Проте поступово стало ясно, що на основі тих самих принципів, а саме: вихідного набору аксіом, правил виведення і поняття виведення – можна описувати не тільки множину виразів, які є висловлюваннями, але і довільні обчислювальні множини об'єктів. Основи теорії таких формальних систем були закладені Э. Постом. Цю теорію можна називати абстрактною або загальною, оскільки вона, на відміну від числення висловлювань, не розглядає властивості формальних систем щодо їхніх конкретних інтерпретацій, а вивчає їхні внутрішні синтаксичні властивості.

Розглянемо приклади формальних систем абстрактного типу.

Приклад 4.1. Множину припустимих шахових позицій можна описати як формальну систему, у котрій єдиною аксіомою є початкова позиція, правилами виведення є правила гри, а теореми – це позиції, отримані за правилами гри із початкових. Проте таке визначення потребує абстрактного уточнення, а саме: потрібно знати чий черговий хід, чи ходив раніш король, чи не був останній хід ходом через два поля вперед.

Приклад 4.2. Розглянемо абстрактне визначення орієнтованої двополюсної схеми. Два варіанти таких схем показано на рис. 4.1.

а *б*

Рис. 4.1. Двополюсні схеми.

Аксіомами такої системи є елементи із виділеними полюсами, правилами виведення – правила з'єднання елементів у схему.

Інтерпретацією може бути: електрична схема, графік та інше.

4.5. Мови і граматики

До початку XX ст., говорячи про мови, мали на увазі тільки природні (українську, англійську, латинську і ін.), що є або були в минулому засобом спілкування між людьми в їхньому повсякденному житті. Наука про мови – лінгвістика – зводилася в основному до вивчення конкретних природних мов, їхньої класифікації, з'ясовування подібностей і розходжень між ними.

Теорія формальних мов, граматик та автоматів активно розвивається з 1950-х років. Її основоположником є американський вчений, професор Ноам Чомскі[8]. Головним завданням теорії є розроблення математичного апарату для опису та аналізу природних і штучних мов. У наш час теорія формальних мов, граматик та автоматів є потужним засобом математичного моделювання, який активно застосовується, зокрема, в синтаксичному аналізі, перекладі, та у розробленні трансляторів.

Виникнення і розвиток *метаматематики,* яка вивчає по суті мову математики, дослідження засобів комунікації у тварин, ідеї структуралістського підходу у лінгвістиці розширили уявлення про мову, згідно з яким під мовою розуміють всякий засіб спілкування, який складається із таких елементів:

а) знаків системи, тобто множини припустимих послідовностей знаків;

б) множини змістів цієї системи;

в) відповідності між послідовностями знаків і змістом, що робить «осмисленими» припустимі послідовності знаків.

Знаками можуть бути, наприклад, букви алфавіту, математичні позначення, звуки, напрями і т.д. Наука про осмислені знакові системи називається *семіотикою.*

[8] Авра́м Ноа́м Чо́мскі (також передається як Хо́мський, англ. Avram Noam Chomsky; нар. 7 грудня, 1928 року, Філадельфія, Пенсільванія) – американський публічний інтелектуал, відомий своєю роботою в галузі лінгвістики, політичною активністю та соціальною критикою, професор мовознавства Масачусетського технологічного інституту (MTI) у відставці. Іноді його називають «батьком сучасної лінгвістики». Автор класифікації формальних мов, що називається ієрархією Чомскі (Хомського). Його роботи про породжувальні граматики зробили значний внесок у занепад біхевіоризму і сприяли розвитку когнітивних наук. Є автором понад 150 книг на такі теми як лінгвістика, війна та політика.

Семіотичний підхід виявився дуже плідним у різноманітних галузях знання: у біології, соціології, лінгвістиці. При цьому різні гілки семіотики мають значну специфіку і не скрізь ще використовують точні математичні засоби. Найбільш просунутими є дослідження знакових систем, у яких знаками є символи алфавітів, а послідовностями знаків – тексти; до таких знакових систем відносяться природні мови, мови науки, а також мови програмування.

Саме інтерес до мов програмування і необхідність вирішення задачі машинного перекладу природних мов спричинив виникнення нової науки – математичної лінгвістики, яка розглядає мову як довільну множину осмислених текстів.

Правила, що визначають множину текстів, утворюють синтаксис мови; опис множини змістів і відповідності між текстами і змістами – семантику мови. Семантика залежить від характеру об'єктів, які описуються мовою, і засоби її вивчення відрізняються для різних типів мов.

Семантику мови математики – формальні теорії – ми розглянули вище. Використання і дослідження семантики мов програмування стало самостійною галуззю теоретичного програмування.

Розвиток семантики пов'язаний, насамперед, із машинним перекладом.

Що ж стосується синтаксису, то його особливості набагато менше залежать від призначення і цілей мови. Саме у сфері вивчення синтаксису отримано значні результати і склався спеціальний математичний апарат – теорія формальних граматик. Зауважимо, що при цьому мова розглядається вже не як засіб спілкування, а як множина формальних об'єктів, послідовностей символів алфавіту. Далі такі послідовності будуть називатися *словами.*

4.6. Формальні граматики і їхні властивості

Математичні моделі, що використовують подання текстів у вигляді послідовності символів, називають формальними мовами і граматиками.

Визначення 4.1. Кінцева множина символів, неподільних у даному розгляді, називається *словником* чи *алфавітом*, а символи, що входять у множину, – літерами (буквами) алфавіту.

Наприклад, алфавіт $A = \{2, b, c, +, !\}$ містить 5 літер, а алфавіт $B = \{00, 01, 10, 11\}$ – 4, кожна з яких складається з двох символів.

Визначення 4.2. Послідовність букв алфавіту називається *словом* або *ланцюжком* у цьому алфавіті. Число літер, що входять у слово, називається його *довжиною*.

Наприклад, слово $a = 2bc$ в алфавіті A має довжину $l(a) = 3$, а слово $b = 0000110010$ в алфавіті B має довжину $l(b) = 5$.

Якщо задано алфавіт A, то через A^* позначимо множину всіляких ланцюжків, які можуть бути побудовані з літер алфавіту A. При цьому передбачається, що порожній ланцюжок, який позначимо знаком $, також входить у множину A^*.

Нехай задано алфавіт V. Тоді можна вважати заданою і множину V^* усіх скінченних слів або ланцюжків в алфавіті V.

Формальна мова L в алфавіті V являє собою довільну підмножину $L \subseteq V^*$. Конструктивний опис формальних мов здійснюється за допомогою формальних систем спеціального виду, які називаються *формальними породжувальними граматиками*.

Формальна породжувальна граматика G – це формальна система, обумовлена четвіркою об'єктів: $G = \langle V, W, I, P \rangle$, де V – алфавіт термінальних символів; W – алфавіт нетермінальних символів ($V \cap W = \varnothing$); I – початковий символ (аксіома) граматики; P – кінцева $\xi \to \beta$ множина правил ξ виду β, де i – ланцюжки в алфавіті $V \cup W$.

Ланцюжок β безпосередньо виведений із ланцюжка α в граматиці G (позначається $\alpha \to_G \beta$). Індекс G опускається, коли ясно, про яку саме граматику йде мова.

Якщо $\alpha = \gamma\xi\delta$, $\beta = \gamma\eta\delta$ і $\xi \to \eta$.

Ланцюжок β називається *виведеним* із α, якщо існує послідовність $E_0 = \tau$, $E_1, E_2 \ldots, E_n = \beta$ така, що для всіх $i = 0, 1, \ldots n-1$, $E_i \Rightarrow E_{i+1}$. Ця послідовність називається *виведенням* β *із* α, а n – довжиною виведення. Виведеність β із α, позначається таким чином: $\alpha \overset{n}{\to} \beta$.

Мовою $L(G)$, породженою граматикою G, називається множина всіх ланцюжків у термінальному алфавіті V, виведених з I.

Граматики G і G' еквівалентні, якщо $L(G) = L(G')$.

У теорії граматик склалися свої традиції позначень, яких ми будемо дотримуватися. Символи термінального алфавіту прийнято позначати малими латинськими літерами, ланцюжки в алфавіті $V \cup W$ – грецькими. Довжина ланцюжка α позначається $l(\alpha)$ або $|\alpha|$, множина всіх ланцюжків в алфавіті V через V^*. Множина всіх непорожніх ланцюжків – V^+.

Лема 4.1. Для довільної граматики G існує еквівалентна їй граматика G_1, ліві частини правил якої не містять входжень основних символів.

Нехай $G = \langle V, W, I, P \rangle$, Кожному символу $a \in V$ поставимо у відповідність двійник, а саме символ A, який не міститься в $V \cup W$. Множину всіх двійників позначимо через V'.

Побудуємо тепер граматику $G_1 = \langle V_1, W_1, I_1, P_1 \rangle$ у такий спосіб: $V_1 = V$, $I_1 = I$, $W_1 = V' \cup W$, а $R_1 = R' \cup R''$, де R' – множина всіх правил вигляду: $A \to \alpha$ ($a \in V$, A – двійник a), а R'' отримано з R заміною в кожному правилі усіх входжень термінальних символів входженнями їхніх двійників. Кожному виводу I, ε_1, ... ε_n у G відповідає висновок I, ε_1, ... ε_n, ε_{n+m} у G', де ε_i ($i \le n$) отримано з ε заміною всіх символів із V їхніми двійниками, а ε_{n+J} ($J \le m$) отримано з ε_{n+J-1} застосуванням правила R'з , причому до ε'_{n+m} правила з R уже незастосовні. Ясно, що $\varepsilon'_{n+m} \le \varepsilon_n$, тому $L(G) \subseteq L(G_1)$. Оскільки тільки правила з R' містять термінальні символи в правих частинах, то будь-який висновок із G_1 ланцюжка довжини m повинен містити m застосувань правил із R'. Видаливши з виведення застосування цих правил і привівши в ньому обернене

перейменування двійників у символи V, одержимо висновок того ж ланцюжка в G. Звідси

$$L(G_1) \subseteq L(G) \text{ і отже}$$

$$L(G) = L(G_1). \ \blacksquare$$

Спосіб уведення двійників, тільки що продемонстрований при дослідженні граматик, є дуже поширеним. Сама ж лема дозволяє стверджувати, що для будь-якої мови L, що породжена певною формальною граматикою, існує породжувана нею граматика, для якої L – множина її заключних слів.

Таким чином, формальні граматики спроможні містити будь-які перелічені множини.

Розглянемо кілька прикладів, що ілюструють введені поняття.

1. Задана граматика $\Gamma_{1.0}$ і потрібно визначити мову, породжувану нею:

$$\Gamma_{1.0}: \quad V_\tau = \{a, b, c\}, V_A = \{I\rangle\}, R = \{I\rangle \to abc\}.$$

Схема граматики містить одне правило, тому $\Gamma_{1.0}$ породжує мову з одного слова, а саме:

$$L(\Gamma_{1.0}) = \{\text{abc}\}.$$

2. Задана граматика $\Gamma_{1.1}$. Потрібно визначити мову, породжувану цією граматикою:

$$\Gamma_{1.1}: \ V_\tau = \{a, b, c, d\}, V_A = \{I, B, C\}$$
$$R = \{ I \to aB$$
$$B \to Cd$$
$$B \to dc$$
$$C \to \$\}.$$

Побудуємо всі виводи в цій граматиці:

$$I \Rightarrow aB \Rightarrow aCd \Rightarrow ad, I \Rightarrow aB \Rightarrow adc.$$

Отже, мова $L(\Gamma_{1.1}) = \{adc, ad\}$.

3. Задана граматика $\Gamma_{1.2}$. Потрібно визначити мову, породжувану цією граматикою:

$$\Gamma_{1.2}: V_A = \{I, A\}, V_\tau = \{0, 1\},$$
$$R = \{ I \to 0A1$$
$$0A \to 00A1$$
$$A \to \$\}.$$

Розглянемо кілька виводів за допомогою правил граматики Г1.2. Застосовуючи перше і третє правила, одержуємо:

$$I \Rightarrow 0A1 \Rightarrow 01.$$

Застосовуючи два рази перше правило і третє, маємо

$$I \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 0011.$$

У загальному випадку, застосовуючи K разів перше правило, одержимо в результаті ланцюжок, що містить K нулів і K одиниць.

Отже, мова, породжувана граматикою Г1.2, містить всякі ланцюжки, в яких число нулів дорівнює числу одиниць.

4. Задана граматика Г1.3. Потрібно визначити мову, породжувану цією граматикою .

$$\Gamma_{1.3}: V_\tau = \{a, b\}, V_A = \{I, A\},$$

$$R = \{ I \rightarrow aA$$
$$A \rightarrow bA\}.$$

Спроба побудови виведення в цій граматиці приводить нас до ланцюжка:

$$I \Rightarrow aA \Rightarrow abA \Rightarrow abbA \Rightarrow \ldots ,$$

який виявляється нескінченним. Іншими словами, $\Gamma_{1.3}$ породжує порожню мову.

Специфіка формально-лінгвістичного підходу до опису множин ланцюжків починає виявлятися при розгляді більш вузьких класів граматик. Узвичаєною класифікацією граматик і породжуваних мов є *ієрархія Чомскі* (*Хомського*), яка містить чотири типи граматик.

Граматика типу 0 є найбільш загальною, без обмежень на правила виведення, тобто в ній немає ніяких обмежень на правила породження. Будь-яке правило

$$r = \eta \rightarrow \psi,$$

може бути побудоване з використанням довільних ланцюжків $\eta, \psi \in (V_\tau \cup V_A)^*$.

Наприклад, $CCLW \Rightarrow WT\,xSAb \Rightarrow xrtHD$.

Граматика типу 1 (контекстно-залежна). Всі її правила виведення мають такий вигляд:

$$\alpha A \beta \rightarrow \alpha \omega \beta, \text{ де } \omega \in (V \cup W)^+.$$

Ланцюжки α і β залишаються незмінними при застосуванні правила, тому їх називають контекстом (відповідно лівим і правим), а граматику – контекстно-залежною.

Граматики типу 1 значно зручніші на практиці, ніж граматики типу 0, оскільки в лівій частині правила замінюється завжди один нетермінальний символ, який можна зв'язати з деяким синтаксичним поняттям, у той час як у граматиці типу 0 можна заміняти відразу кілька символів, у тому числі і термінальних.

Наприклад, граматика

$$\Gamma_{1.4}: V_\tau = \{a, b, c, d\}, V_A = \{I, A, B\}$$

$$R = \{\ I \to aAI,$$
$$AI \to AAI$$
$$AA \to ABA$$
$$A \to b$$
$$bBA \to bcdA$$
$$bI \to ba\ \}$$

є контекстно-залежною, оскільки друге і шосте правила мають непорожній лівий контекст, а третє і п'яте містять обидва контексти. Виведення у такій граматиці може мати вигляд:

$$I \Rightarrow aAI \Rightarrow aAAI \Rightarrow abAI \Rightarrow abbI \Rightarrow abba.$$

Граматика типу 2 (контекстно-вільна або безконтекстна, КВ). Всі її правила мають такий вид:

$$A \to \alpha, \text{ де } \alpha \in (V \cup W)^*.$$

Очевидно, що вони виходять із правил граматики типу 1 за умови $\alpha = \beta = \$$. Оскільки контекстні умови відсутні, то правила КВ-граматик виходять простішими, ніж правила граматик типу 1. Саме такі граматики використовують для опису мов програмування. Прикладом КВ-граматики може слугувати $\Gamma_{1.5}$:

$$V_\tau = \{a, b\},\ V_A = \{I\},$$
$$R = \{\ I \to aIa$$
$$I \to bIb$$
$$I \to aa$$
$$I \to bb\}.$$

Вона породжує мови, що складаються з ланцюжків, кожний з яких у свою чергу складається з двох частин, ланцюжка $\beta \in V_\tau^*$ і його дзеркального відображення β'.

$$L(\Gamma_{1.5}) = \{\ \beta\beta' \mid \beta \in V_\tau^+\},$$

де V_τ^+– це множина V_τ^* без порожнього ланцюжка.

За допомогою правил цієї граматики може бути побудований, наприклад, такий ланцюжок:

$$I \Rightarrow aIa \Rightarrow abIba \Rightarrow abaIaba \Rightarrow ababbaba.$$

Граматика типу 3 (регулярна або автоматна, А-граматика). Всі її правила мають такий вигляд:

$$A \to \alpha, \text{ або } A \to \alpha B \text{ або } A \to B\alpha,$$

де $\alpha \in V_\tau$, $A, B \in V_A$. причому граматика може мати тільки правила вигляду $A \to \alpha B$ – правосторонні, або тільки вигляду $A \to B\alpha$ – лівосторонні. Прикладами автоматних можуть слугувати правостороння $\Gamma_{1.6}$ і лівостороння $\Gamma_{1.7}$ граматики.

$\Gamma_{1.6}$: $V_\tau = \{a, b\}$, $V_A = \{I, A, Z\}$,

$$R = \{ I \to aI$$
$$I \to aA$$
$$A \to bA$$
$$A \to aZ$$
$$A \to bZ$$
$$Z \to \$ \}.$$

$\Gamma_{1.7}$: $V_\tau = \{a, b\}$, $V_A = \{I, A, Z\}$,

$$R = \{ I \to Ab$$
$$A \to Ab$$
$$A \to Za$$
$$Z \to Za$$
$$Z \to \$ \}.$$

Ці граматики є еквівалентними і породжують мову

$$L(\Gamma_7) = \{ aa\ldots abb\ldots b \mid n, m\ 0\}.$$

Мова L належить до типу i (i = 0,1, 2, 3), якщо існує граматика типу i, яка її породжує.

Між множинами мов різних типів існує відношення включення:

$$\{ L \text{ типу } 3 \} \in \{ L \text{ типу } 2 \} \in \{ L \text{ типу } 1 \} \in \{ L \text{ типу } 0\}.$$

Доведено, що існують мови типу 0, які не відносяться до типу 1, мови типу 2, котрі не належать до типу 1, і мови типу 3, що не співпадають із типом 2.

З огляду на те, що найбільше практичне застосування знаходять граматики типу 2, далі ми більш детально розглянемо самі їх.

2.7. Контекстно-вільні граматики

КВ-мови є найвивченішим класом. Це пояснюється тим, що з одного боку КВ-граматики виявилися дуже зручним апаратом для опису побудови природних мов і особливо мов програмування; з іншого боку, їм притаманна відносна простота та змістовність структури і наявність зрозумілих засобів опису. Дослідження КВ-мов також має значний теоретичний інтерес.

Виникнення поняття КВ-граматики практично збіглося із появою метамови Бекуса (або нормальної форми Бекуса), яка була вперше використана при описі мови програмування АЛГОЛ-60 і швидко стала узвичаєним засобом їх формального опису. Опис мови за допомогою нормальних форм Бекуса являє собою сукупність так званих «*металінгвістичних формул*» – виразів вигляду

$$X:: = Y_1 \mid \ldots \mid Y_n,$$

де X – деякий текст, укладений у кутові дужки і називаний *металінгвістичною змінною*, а $Y_{1\ldots}Y_n$ – послідовності металінгвістических змінних і основних символів мови (букв, цифр, роздільників, неподільних слів типу «beqin, end, else» і т.д.). Знак « ::= » називається металінгвістичною зв'язкою і читається як «є» або «це». Знак | – це металінгвістична зв'язка «або». Металінгвістична змінна являє собою ім'я конструкції мови.

Металінгвістична формула в цілому – це опис різноманітних синтаксичних варіантів будівлі конструкції X, що знаходиться в лівій частині, через інші конструкції й основні символи мови, зазначені в правій частині. Перерахування варіантів проводиться за допомогою зв'язок.

Приклад 1. Множина ідентифікаторів мови програмування – це ланцюжки із букв і цифр, що починаються з літери. Тоді поняття ідентифікатора може бути описане за допомогою таких нормальних форм Бекуса (металінгвістичних формул), у яких ланцюжок із букв і цифр скорочено називається БЦ-ланцюжком:

<ідентифікатор> ::=<буква>| <буква><БЦ-ланцюжок>

<БЦ-ланцюжок >::=<буква> | <цифра> | <буква>

<БЦ-ланцюжок > |<цифра><БЦ-ланцюжок >

<буква>::=*a* |*b*|... | *z*

<цифра>::=0 |1|... | 9

Основні символи мови – це 26 букв латинського алфавіту і 10 цифр.

Приклад 2. Арифметичні вирази (без констант і з фіксованою множиною змінних *a, b, c*) у метамові Бекуса описується в такий спосіб:

<арифм. вираз>::=<терм>|<арифм.вираз>+<терм>|<арифм. вираз>-<терм>

<терм>::=<множник>|<терм>*<множник>| <терм>/<множник>

<множник>::=(<арифм.вираз>) | <змінна>

<змінна> ::=a |b| c.

Вже з цих прикладів ясно, що нормальні форми Бекуса неважко перетворити в КВ-граматику.

Дійсно, основні символи відповідають термінальним символам граматики, металінгвістичні змінні – нетермінальним символам, зв'язка ::= відповідає знаку →, а без зв'язки | можна обійтися, якщо формулу $X::=Y_1|\ldots\ |Y_n$ замінити системою формул $X::=Y_1,\ldots\ X::=Y_n$. Якщо зазначені перетворення провести для останнього приклада, замінивши <арифм.вираз> на символ I, <терм > на T, <множник> на M, <змінна> на L, те одержимо КВ-граматику:

$$I \rightarrow T;$$

$$I \rightarrow I + T;$$

$$I \to I - T;$$
$$T \to M;$$
$$T \to T \times M;\ T \to T / M;$$
$$M \to I;\ M \to L; L \to a; L \to b; L \to c.$$

Така відповідність є взаємно однозначною, і всяка КВ-граматика перетворюється в нормальні форми Бекуса. Це свідчить про близькість теоретичних і практичних цілей дослідження граматик, які породжують ту чи іншу мову, і обумовлює їх використання для формалізації алгоритмічних мов програмування.

Висновки

Однієї з фундаментальних ідей, на яку спираються дослідження з основ математики, є формалізації теорій, тобто послідовне проведення аксіоматичного методу їх побудови. При цьому не дозволяється користуватися якимись припущеннями про об'єкти теорії, крім тих, що описано явно у вигляді аксіом; аксіоми розглядаються як формальні послідовності символів (виразів), а доведення – як методи одержання одних виразів з інших за допомогою операцій над символами. Такий підхід гарантує чіткість вихідних тверджень і однозначність висновків. Аксіоми і правила виводу необхідно вибирати таким чином, щоб побудована з їхньою допомогою формальна теорія мала змістовний сенс.

Будь яка алгоритмічна мова в своїй основі повинна базуватися на принципах формальної системи, що забезпечить відсутність протиріч та помилок в її складі. Останнім часом мало уваги приділяється навчальним варіантам псевдомов, які спрощують сприйняття та вивчення коректних основ програмування.

Питання, викладені в цьому розділі, розглянуто у літературі [1, 3 – 6, 8 – 13, 16 – 19, 22 – 25]

Питання для самоконтролю.

1. Дайте визначення формальної системи?
2. Які складові має точна формальна теория?
3. Що є предметом основ математики?
4. Яким чином будують формальну теорія?
5. Які типи висловлювань використовують в формальній теорії?
6. Які логічні операції застосовують у численні висловлювань?
7. Що означають індекси предикатів?
8. Які квантори використовуються в численні висловлювань?
9. Які правила виводу існують в численні висловлювань?
10. Поясніть правило виводу Modus Ponens.
11. В чому полягає правило узагальнення ($\forall$ - уведення)?
12. В чому полягає правило $\exists$ - уведення:

13. Назвіть складові будь-якої мови.
14. Що називається семіотикою?
15. Які типи граматик існують за класифікацією Чомскі?
16. Що являє собою граматика типу 0?
17. Які особливості граматики типу 1?
18. Що являє собою контекстно-вільна граматика?
19. Назвіть ознаки граматики типу 3.
20. Які металінгвістичні зв'язки використовуються в контекстно-вільних граматиках?

Задачі для самостійного розв'язування

1. Дайте визначення умов точного опису поняття «формула».

2. Дайте визначення умов точного опису поняття «масив».

3. Виконайте аналіз як формальної системи а) алгоритмічної мови C++; б) HTML.

4. Наведіть приклади виведення формул в елементарній алгебрі, базуючись на формальній теорії.

5. Наведіть приклад гіпотези з таких розділів математики: математичний аналіз, алгебра, геометрія, алгебра логіки та теорія графів.

6. Наведіть приклад аксіом в таких розділах математики як математичний аналіз, алгебра, геометрія, алгебра логіки та теорія графів.

7. Наведіть приклад застосування правила виведення в алгебрі та геометрії.

8. Наведіть приклад застосування правила підстановки в алгебрі логіки.

9. Наведіть приклади одно-, дво- та тримісних предикатів.

10. Наведіть приклад знакової системи.

11. Застосуйте правила виведення в граматиці типу 0 для визначення понять в довільній алгоритмічній мові.

12. Застосуйте правила виведення в граматиці типу 1 для визначення понять в довільній алгоритмічній мові.

13. Застосуйте правила виведення в граматиці типу 2 для визначення понять в довільній алгоритмічній мові.

14. Застосуйте правила виведення в граматиці типу 3 для визначення понять в довільній алгоритмічній мові.

15. Побудуйте нормальну форму Бекуса для визначення перелічених нижче операторів:

а) арифметичного; б) умовного; в) логічного; г) циклу.

16. За допомогою КВ-граматики дайте визначення операторів із задачі 15.

РОЗДІЛ 5

КОМБІНАТОРНІ КОНФІГУРАЦІЇ

Мета розділу: вивчення основних задач та формул комінаторного аналізу, їх властивостей та практичного застосування

Розділ математики, присвячений розв'язанню задач вибору та розташування деякої, зазвичай, скінченної множини відповідно до заданих правил, називається *комбінаторикою*. Кожне таке правило визначає спосіб побудови певної конструкції із елементів вихідної множини, що зветься *комбінаторною конфігурацією*. Тому на меті комбінаторного аналізу стоїть дослідження комбінаторних конфігурацій, алгоритмів їх побудови, оптимізація таких алгоритмів, а також розв'язання задач переліку. Найпростішими прикладами комбінаторних конфігурацій є перестановки, розміщення, комбінація та розбиття. Комбінаторика пов'язана з багатьма іншими розділами математики.

Предмету комбінаторики не так просто дати стисле та вичерпне визначення. У деякому сенсі слово «комбінаторика» можна розуміти як синонім терміну «дискретна математика», тобто дослідження дискретних скінченних математичних структур. На шкільному рівні з терміном «комбінаторика» пов'язують просто набір відомих формул, які використовуються для обчислення так званих *комбінаторних чисел*. Може здаватися, що ці формули корисні тільки для розв'язування навчальних задач і не мають відношення до практичного застосування. Насправді це далеко не так. Обчислення на дискретних скінченних математичних структурах, які часто називають *комбінаторними обчисленнями*, потребують комбінаторного аналізу для встановлення властивостей і виявлення оцінки придатності використовуваних алгоритмів. Розглянемо приклад із практики.

Припустимо деяке агентство нерухомості має у своєму розпорядженні базу даних із n записів, причому кожний запис містить одну пропозицію (що є) і один запит (що потрібно) щодо об'єктів нерухомості. Необхідно знайти всі такі пари записів, в яких пропозиція першого запису збігається із запитом другого, і, навпаки, пропозиція другого запису збігається із запитом першого. На побутовій мові це називається добором варіантів обміну. Припустимо, що база даних дозволяє перевірити варіант за одну мілісекунду. Неважко зміркувати, що при «лобовому» алгоритмі пошуку варіантів (кожний запис порівнюється із кожним) буде потрібно $n(n-1)/2$ порівнянь. Якщо $n = 100$, то все в порядку – відповідь буде отримана за 4,95 секунди. Але якщо $n = 100\,000$ (більш реальний випадок), то відповідь буде одержана за 4 999 950 секунд, що складає майже 1389 годин і навряд чи може вважатися прийнятною. Зверніть увагу, що ми оцінили тільки трудомісткість добору прямих варіантів, а існують ще варіанти, коли кількість учасників угоди більше двох...

Цей приклад показує, що комбінаторні обчислення потребують попереднього аналізу і кількісної оцінки вихідних задач і алгоритмів, які використовуються. Задачі зазвичай оцінюються з огляду на розмір, тобто

загальну кількість різних варіантів, серед яких потрібно знайти розв'язок, а алгоритми оцінюються із погляду складності. При цьому розрізняють складність за часом (або *часову складність*), тобто кількість необхідних кроків алгоритму, і складність за пам'яттю (або *ємнісну складність*), тобто обсяг пам'яті, який потрібен для роботи алгоритму.

Назвемо деякі з основних задач комбінаторики.

1. Довести існування чи відсутність конфігурації із заданими властивостями.

Тут спочатку потрібно сформулювати вимоги до класу конфігурацій, які потрібно побудувати. Потім проводиться теоретичне доведення існування (або ні) хоча б однієї конфігурації із заданими властивостями (її побудова – це окреме і часто дуже складне завдання)

2. Знайти загальну кількість конфігурацій із заданими властивостями.

Існує досить багато задач, які можна сформулювати у такому вигляді. Наприклад, скількома способами можна провести матчі з футболу між 10 різними командами?

3. Знайти конфігурацію елементів множини, що має зазадалегідь задані властивості.

Наприклад, може виникнути потреба скласти конкретний розклад матчів між 10 футбольними командами, а не тільки знати їх кількість.

4. Описати всі методи розв'язування даної комбінаторної задачі, подати алгоритми їх переліку. На розв'язування цієї задачі зараз спрямовані зусилля багатьох вчених.

5. З усіх розв'язків даної комбінаторної задачі обрати оптимальний за тими чи іншими параметрами.

До цього класу відносяться задачі комбінаторної оптимізації. Наприклад, задача комівояжера[9], яка досі не має остаточного розв'язання, задачі розміщення та ін.

У всіх випадках основним інструментом такого аналізу є формули і методи, які розглянуто у цьому розділі.

5.1. Елементи комбінаторики

У багатьох практичних застосуваннях виникає необхідність підрахувати кількість можливих комбінацій об'єктів, які задовольняють визначені умови. Такі задачі називаються *комбінаторними*.

Наведемо їх приклади.

[9] Полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Прості способи розв'язання задачі комівояжера: повний лексичний перебір, жадібні алгоритми (метод найближчого сусіда), метод включення найближчого міста, метод найдешевшого включення, метод мінімального кістяка дерева. На практиці застосовують різні модифікації ефективних методів, а саме: гілок і меж, генетичні та мурашині алгоритми. Всі ефективні (такі, що скорочують повний перебір) методи розв'язання задачі комівояжера є евристичними.

1. Система зв'язку складається з n однакових антен, які розташовані в лінійному порядку. Отримана система зможе приймати всі вхідні сигнали і буде називатися функціональною, допоки жодні дві послідовні антени не будуть несправними. Якщо виявиться, що рівно m з n антен несправні, яка ймовірність того, що отримана система буде справною? Наприклад, у випадку, коли $n = 4$ і $m = 2$, існує 6 можливих конфігурацій системи, а саме:

$$\begin{matrix} 0\,1\,1\,0 \\ 0\,1\,0\,1 \\ 1\,0\,1\,0 \\ 0\,0\,1\,1 \\ 1\,0\,0\,1 \\ 1\,1\,0\,0 \end{matrix},$$

де 1 означає, що антена працює, а 0, що ні. Оскільки отримана система буде функціональною в перших трьох схемах і не функціонуватиме в решті, ми можемо обчислити ймовірність, а саме: $\frac{3}{6} = 0{,}5$. У випадку загальних n і m також можна обчислити ймовірність правильного функціонування системи, але для цього потрібно вміти підрахувати кількість конфігурацій системи, які забезпечують її працездатність і загальну кількість усіх можливих конфігурацій.

2. Двадцять працівників мають бути призначені на 20 різних робочих місць по одному на кожну роботу. Скільки різних призначень можливо?

З попереднього викладення ми бачимо, що було б корисно мати ефективний метод для підрахунку кількості способів, якими може відбуватися та чи інша подія. Насправді багато проблем теорії ймовірностей можна вирішити, просто зробивши це. Комбінаторний аналіз застосовується наразі не тільки в завданнях теорії ймовірності, але й в багатьох інших науках. Наприклад, однією із найскладніших загадок у біології XX ст. була будова «ниток життя» – молекул білка і нуклеїнових кислот. Виявилось, що молекули білка – це об’єднання декількох довгих ланцюгів, складених із 20 амінокислот Після відкриття структури ДНК постало питання: яким чином її молекули передають організму інформацію про побудову ланцюгів амінокислот, котрі складають білки. Дослідження показали, що мова йдеться про 20 амінокислот. Американський фізик Г. Гамов сформулював задачу так: як за допомогою 4 видів нуклеотидів можна зашифрувати 20 видів амінокислот?

Ще одне застосування комбінаторних методів – криптографія (наука про шифрування). Наприклад, при розшифруванні давніх мов, вирішенні питань кібербезпеки.

Отже, розмаїтість комбінаторних задач не піддається вичерпному опису, але серед них є ціла низка таких, що трапляються особливо часто, і для них відомі способи підрахунку.

Для формулювання і розв’язування комбінаторних задач використовуються різноманітні моделі *комбінаторних конфігурацій*. Розглянемо дві найбільш популярні.

1. Дано n предметів. Їх потрібно розмістити по m ящиках таким чином, щоб виконувалися задані обмеження. Скількома способами це можна зробити?

2. Розглянемо множину функцій, а саме:

$$F: X \to Y, \quad \text{де} \quad |X| = n, \quad |Y| = m, \qquad X = \{1, \ldots, n\}.$$

Не обмежуючи загальності, можна вважати, що

$$Y = \{1, \ldots, m\}, \quad F = \langle F(1), \ldots, F(n)\rangle, \quad 1 \le F(i) \le m.$$

Скільки існує функцій F, які задовольняють задані обмеження?

5.1.1. Основні правила комбінаторного аналізу. Поняття вибірки

Всі обчислення у комбінаториці можна зробити за допомогою правил суми та добутку.

Правило суми. Якщо об'єкт x можна вибрати n способами, а об'єкт y – m способами, то x або y можна вибрати $(n + m)$ способами.

Приклад 5.1. Припустимо, потік складається з двох груп. У першій 20 студентів, у другій – 23. Тоді обрати одного студента з потоку для доповіді можна 20 + 23 = 43 способами.

Правило добутку. Якщо об'єкт x можна вибрати n способами, і після кожного такого вибору об'єкт y можна обрати m способами (тобто вибір об'єкта x не впливає на кількість способів вибору об'єкта y), то пару (x, y) можна вибрати nm способами.

Приклад 5.2. Припустимо маємо три міста: А, Б, В. З міста А до міста Б можна дістатися 5 способами, а з міста Б до міста В – 7. Тоді за правилом добутку з міста А до міста В можна дістатися 5×7 = 35 способами.

Приклад 5.3. У їдальні є вибір з 3 перших і 5 других блюд. Тоді обід з двох блюд можна обрати 3×5 = 15 способами.

Вибіркою із множини $A = \{a_1, a_2, \ldots a_n\}$ називають сукупність об'єктів $\{a_{i1},\ldots,a_{ik}\}$, яка містить k елементів множини A. У зв'язку з тим, що у комбінаториці вивчають кількість вибірок, то від значень елементів множини A часто абстрагуються, і говорять про вибірку із n елементів по k, або k-вибірку із n елементів.

Залежно від способу вибору існують два типи вибірок: *без повторень* – коли при виборі кожного нового елемента його обирають так, щоб він не збігався з уже вибраними, і *з повтореннями.*

У вибірці з повтореннями один і той самий елемент із множини A може зустрітись кілька разів, але це не обов'язково. Назва характеризує саме спосіб вибору.

Вибірку називають *упорядкованою*, якщо задано порядок її елементів (тобто вибірки будуть різними, навіть якщо вони відрізняються тільки порядком розташування в них елементів), а якщо ні – то *невпорядкованою.* Зрозуміло, що впорядкована *k*-вибірка – це кортеж (вектор) з *k* компонентами.

Приклад 5.4. Припустимо, множина $A = \{a, b, c\}$. Складемо можливі вибірки довжиною 2 з елементів цієї множини.

Впорядковані вибірки із повтореннями: (a, a), (a, b), (a, c), (b, b), (b, a), (b, c), (c, c), (c, a), (c, b).

Невпорядковані вибірки із повтореннями: (a, a), (a, b), (a, c), (b, b), (b, c), (c, c). При такому способі вибору (a, b) та (b, a), (a, c) та (c, a), (b, c) та (c, b) не розрізрізняють, оскільки порядок об'єктів у вибірці не має значення.

Впорядковані вибірки без повторень: (a, b), (a, c), (b, a), (b, c), (c, a), (c, b).

Невпорядковані вибірки без повторень: (a, b), (a, c), (b, c).

5.1.2. Розміщення

Впорядковані вибірки із повтореннями називають *розміщеннями з повтореннями*.

З цього визначення випливають такі характерні ознаки розміщень з повтореннями:

1) елементи можуть бути однаковими;

2) порядок елементів важливий.

Кількість розміщень з повтореннями з n елементів по m елементів позначають через $U(n, m)$ і обчислюють за формулою:

$$U(n, m) = n^m \ . \qquad (5.1)$$

Її справедливість випливає із правила добутку: на першому місці може бути будь-який із n даних елементів, на другому – теж будь-який із n даних елементів, ... , на m -му місці – те саме.

Отже, розміщенням із n елементів по m називаються комбінації, які включають m елементів вибраних із n елементів.

Приклад 5.5. Скількома способами можна пофарбувати 5 іграшок в 4 кольори?

Розв'язування.

$$U(4, 5) = 4^5 = 1024.$$

Приклад 5.6. Дано цифри 1, 2, 3, 4. Скільки різних двозначних чисел можна скласти, якщо цифри в числі можуть повторюватися?

Розв'язування.

$$U(4,2) = 4^2 = 16.$$

Впорядковані вибірки без повторень називають *розміщеннями без повторень*.

Характерні ознаки розміщень без повторень:

1) елементи не можуть бути однаковими;

2) порядок елементів важливий.

Число ін'єктивних функцій або кількісь всіх можливих способів розмістити k предметів по n ящиках не більше ніж по одному в ящик називається *кількістю розміщень без повторень* і позначається $A(n, k)$ або $(n)_k$ і обчислюється за такою формулою:

$$A(n,k) = \frac{n!}{(n-k)!}. \qquad (5.2)$$

Дійсно, ящик для першого предмета можна вибрати n способами, для другого – $(n - 1)$ способом і т.д. Таким чином:

$$A(n,k) = n(n-1)(n-2)\dots(n-k+1) = \frac{n!}{(n-k)!}.$$

За визначенням вважають, що $A(n, k) := 0$ коли $k > n$ і $A(n, 0) := 1$.

Приклад 5.7. У деяких видах спортивних змагань виходом є визначення учасників, яки посіли перше, друге і третє місця. Скільки можливо різних виходів, якщо в змаганні беруть участь n учасників?

Розв'язування

Кожен можливий вихід відповідає такій функції:

$$F:\{1, 2, 3\} \to \{ 1 \dots n\},$$

тут аргумент – це номер призового місця, результат – номер учасника. Відтак, усього можливо $A(n, 3) = n(n - 1)(n - 2)$ різних виходів.

5.1.3. Перестановки

Перестановка з n елементів – це розміщення без повторень з n по n елементів, тобто коли в розміщення входять усі елементи. Перестановки з n елементів називають також *n-перестановками.*

Характерні ознаки перестановок:

1) елементи не можуть бути однаковими, в перестановку входять всі елементи множини;

2) порядок елементів важливий.

Кількість взаємно однозначних функцій $F:\{ 1 \dots n \} \to \{ 1 \dots n\}$, або *кількість перестановок n* предметів, позначається $P(n)$.

$$P(n) = n!$$

Доведення

$$P(n) = A(n, n) = n(n-1) \ldots (n-n+1) = n(n-1) \ldots 1 = n!.$$

Приклад 5.8. Припустимо, множина $A = \{a, b, c\}$. Складемо всі можливі перестановки з цієї множини.

Розв'язування

Кількість можливих перестановок з трьох елементів: $P(3) = 3!=6$. перелічимо їх. Це будуть комбінації з елементів множини A, які відрізняються тільки порядком елементів, а саме: (a, b, c), (a, c, b), (b, c, a), (b, a, c), (c, a, b), (c, b, a).

Приклад 5.9. Розглянемо послідовність $\varepsilon = (E_1, \ldots, E_m)$ непорожніх підмножин множини E ($\varepsilon \subset 2^E, E_i \subset E, E_i \neq \varnothing$), яка називається ланцюжком в E, коли $\forall i \in 1..m-1 \quad E_i \subset E_{i+1} \,\&\, E_i \neq E_{i+1}$. Ланцюжок ε називається повним в E, якщо $|\varepsilon| = |E|$. Скільки існує повних ланцюжків?

Розв'язування

Очевидно, що в повному ланцюжку кожна така підмножина E_{i+1} отримана з попередньої підмножини E_i додаванням тільки одного елемента з E а, отже, $|E_1| = 1$, $|E_2| = 2, \ldots,$ $|E_m| = |E| = m$. Відтак, повний ланцюжок визначається порядком, в якому елементи множини E додаються для утворення чергового елемента повного ланцюжка. Звідси кількість повних ланцюжків – це число перестановок елементів множини E, вона дорівнює $m!$.

Перестановкою з повтореннями з n елементів називають будь-яке впорядкування n-множини, серед елементів якої є однакові. Якщо серед елементів множини є n_1 елементів першого типу, n_2 елементів другого типу, … n_k елементів k-го типу, причому $n_1 + n_2 + \ldots + n_k = n$, то кількість всіх перестановок такої множини з повтореннями позначують $P_n(n_1, n_2, \ldots n_k)$:

$$P_n(n_1, n_2, \ldots n_k,) = \frac{n!}{n_1!\, n_2! \cdots n_k!}.$$

5.1.4. Сполучення

Невпорядковані k-вибірки з n-елементної множини називають *сполученнями з n елементів по k.*

За допомогою сполучень можна описати, наприклад, кількість способів розмістити k нерозрізнених предметів по n коробках.

Кількість сполучень позначається $C(n, k)$ або $\left(\frac{k}{n}\right)$ і обчислюється за такою формулою:

$$C(n,\ k) = \frac{n!}{k!\,(n-k)!}.$$

Доведемо її.

1. Кількість розміщень без повторень потрібно розділити на кількість перестановок.

2. Кількість сполучень є числом строго монотонних функцій, позаяк строго монотонна функція $F: 1\ldots n \to 1\ldots m$ визначається набором своїх значень, причому $1 \le F(1) < \cdots < F(n) \le m$. Іншими словами, кожна строго монотонна функція визначається вибором n чисел із діапазону $1...m$. Таким чином, кількість строго монотонних функцій дорівнює кількісті n-елементних підмножин m-елементної множини, що, у свою чергу, дорівнює кількості способів вибрати n ящиків із предметами з m ящиків.

За визначенням $C(m, n) := 0$ при $n > m$.

Характерні ознаки сполучень без повторень:

1) елементи не можуть бути однаковими;

2) порядок елементів не важливий.

Приклад 5.10. На початку гри в доміно кожному гравцю видається сім із наявних 28 різноманітних кісток. Скільки існує різноманітних комбінацій кісток, що гравець може одержати на початку гри?

Розв'язування

Очевидно, що шукане число дорівнює числу 7-елементних підмножин 28-елементної множини. Маємо

$$C(28,\ 7) = \frac{28!}{7!\,(28-7)!} = \frac{28 \cdot 27 \cdot 26 \cdot 25 \cdot 24 \cdot 23 \cdot 22}{7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 1184040.$$

Число монотонних функцій або число розміщень n нерозрізнених предметів по m ящиках називається *кількістю сполучень із повтореннями* і позначається $V(m,n)$.

$$V(m, n)=\mathrm{C}(n + m - 1, n).$$

Довести цю формулу можна у такий спосіб.

Монотонній функції $f: \{1, \ldots, n\} \to \{1, \ldots, m\}$ однозначно відповідає строго монотонна функція $f': \{1, \ldots, n\} \to \{1, \ldots, n + m - 1\}$.

Характерні ознаки сполучень із повтореннями:

1) елементи можуть бути однаковими;

2) порядок елементів не важливий.

Приклад 5.11. Скількома засобами можна розсадити n щойно прибулих гостей серед m гостей, які уже перебувають за круглим столом?

Розв'язування

Очевидно, що між m гостями, котрі вже знаходяться за круглим столом, є m проміжків, в які можна розсаджувати щойно прибулих. Таким чином, маємо

$$V(m,\ n) = \mathrm{C}(m + n - 1) = \frac{(m+n-1)!}{n!(m-1)!}$$ варіантів.

Класифікацію і співвідношення між різними типами комбінаторних схем показано на рис. 5.1.

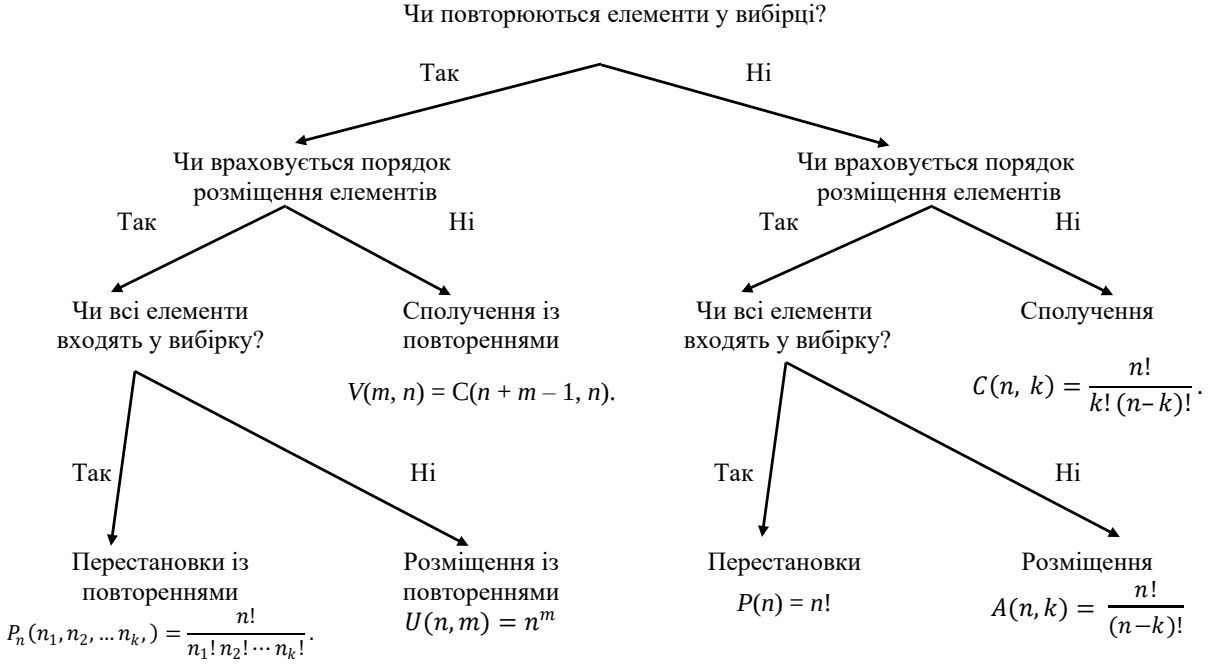


Рис. 5.1. Класифікація комбінаторних конфігурацій

5.2. Підстановки

У цьому параграфі розглядаються підстановки і перестановки, що насправді є рівнозначними поняттями. Для обчислення кількості перестановок у підрозділі 5.1.3 використовується дуже проста формула: $P(n) = n!$. Застосовуючи її при розв'язанні практичних завдань, не варто забувати, що факторіал – це функція, яка дуже швидко зростає, зокрема, хутчіше експоненти. Дійсно, застосовуючи відому з математичного аналізу *формулу Стірлінга*

$$n! \approx \sqrt{2\pi n}\, n^n e^{-n}$$

або, більш точно,

$$\sqrt{2\pi n}\, n^n e^{-n} \prec n! \prec \sqrt{2\pi n}\, n^n e^{-n+1/(12n)},$$

неважко показати, що

$$\lim_{n\to+\infty} \frac{n!}{2^n} = +\infty.$$

5.2.1. Група підстановок

Взаємно однозначне відображення множини X ($f : X \to X$) називається *підстановкою* на X.

Зазвичай, підстановку зручно задавати таблицею з двох рядків. У першому рядку записують значення аргументів, в іншому – відповідні значення функції, наприклад:

$$f = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 1 & 4 & 3 \end{vmatrix} \quad g = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \end{vmatrix}.$$

Оскільки байдуже, у якому порядку записувати впорядковані пари елементів, то одна й та сама підстановка допускає різні подання. Приміром, підстановку f можна записати так:

$$f = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 1 & 4 & 3 \end{vmatrix} = \begin{vmatrix} 2 & 1 & 4 & 3 & 5 \\ 2 & 5 & 4 & 1 & 3 \end{vmatrix} = \begin{vmatrix} 5 & 4 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 & 5 \end{vmatrix}.$$

Добутком підстановок f і q називається їхня суперпозиція $f°q$.

Приклад 5.12. Знайти суперпозицію підстановок f і q, поданих вище.

Розв'язування

$$f°g = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 5 & 3 & 2 \end{vmatrix}.$$

Тотожною називається підстановка *e*, яка кожний елемент переводить в себе, тобто *e* (*x*) = *x*. Наприклад,

$$e = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{vmatrix}.$$

Обернена підстановка являє собою обернену функцію. Вона завжди існує, оскільки підстановка є бієкцією. Таблицю оберненої підстановки можна отримати, якщо просто поміняти місцями рядки таблиці вихідної підстановки.

Приклад 5.13. Знайти обернену підстановку до підстановки

$$f = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 2 & 1 & 5 \end{vmatrix} \quad .$$

Розв'язування

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

Отже, множина підстановок утворить групу щодо операції суперпозиції. Вона називається *симетричною групою* степеня *n*.

5.2.2. Графічне подання підстановок

Підстановки зручно подавати в графічній формі, проводячи стрілки від кожного елемента *x* до елемента *f* (*x*).

Приклад 5.14. Графічне подання підстановки $f = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 4 & 5 \end{vmatrix}$ показано на рис. 5.2.

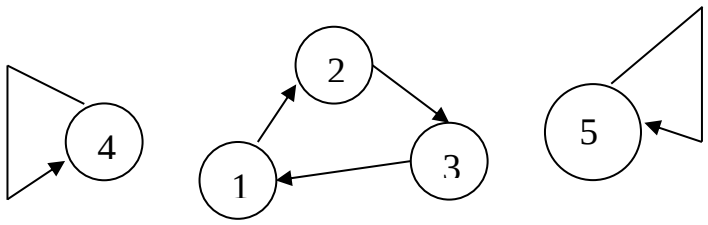


Рис. 5.2. Графічне подання підстановки

Цикл підстановок – це послідовність елементів $x_0,...x_к$, при якій

$$f(x_i) = \begin{cases} x_{i+1}, & 0 \le i \prec k, \\ x_0, & i = k. \end{cases}$$

Цикл довжини 2 називається *транспозицією*.

5.2.3. Підстановки і перестановки

У таблиці підстановки нижній рядок (значення функції) є перестановкою елементів верхнього рядка (значення аргументу). Якщо домовитись, що елементи верхнього рядка (аргументи) завжди розташовуються у визначеному порядку (наприклад, за зростанням), то верхній рядок можна не вказувати – підстановка визначається одним нижнім рядком. Відтак, підстановки однозначно відповідають перестановкам.

Перестановку (і відповідну їй підстановку) елементів 1, ... *n* будемо позначати $\langle a_1, a_2, \cdots a_k \rangle$, де всі a_i – довільні числа з діапазону 1, ... , *n*.

Якщо в перестановці $f = \langle a_1, a_2, \cdots a_k \rangle$ для елементів a_i і a_j має місце нерівність $a_i > a_j$, коли $i < j$, то пара (a_i, a_j) називається *інверсією*. Позначимо через *I*(*f*) *кількість інверсій* у перестановці *f*.

Має місце така теорема.

Теорема 5.1. Довільну підстановку *f* можна подати у вигляді суперпозиції *I*(*f*) транспозицій сусідніх елементів.

Доведення

Нехай $f = \langle a_1, \cdots 1, \cdots a_k \rangle$ Переставимо 1 на перше місце, помінявши її місцями із сусідніми зліва елементами. Позначимо послідовність цих транспозицій через t_1. При цьому всі інверсії, в яких брала участь 1 (і тільки вони), зникнуть. Потім переставимо 2 на друге місце і т.д. Таким чином, $f^{\circ}t_1{}^{\circ}t_2{}^{\circ}\ldots{}^{\circ}t_n = e$ і, за властивостями групи, $f^{\circ}t_1{}^{\circ}t_2{}^{\circ}\ldots{}^{\circ}t_n = t_n^{-1\circ}\ldots{}^{\circ}t_2^{-1\circ}t_1^{-1}$, причому $|t_1| + |t_2| + \ldots + |t_n| = I(f)$. Теорему доведено.

Наслідком цієї теореми є твердження, що всяке сортування може бути виконано перестановкою сусідніх елементів.

Доведена теорема підтверджує, що довільну перестановку можна подати у вигляді композиції визначеної кількості транспозицій, але вона не стверджує, що таке подання є ефективним. Сортування, засноване на цій теоремі, відоме як *метод бульбашки*.

Зауважимо, що при переміщенні елемента на своє місце транспозиціями сусідніх елементів усі інші залишаються на своїх місцях, крім тих, що переміщуються і знаходяться на цільовому місці (вони міняються місцями). Відтак, метод бульбашки може бути виражений у формі алгоритму. Це простий, але далеко не найефективніший алгоритм сортування. Опишемо його.

Алгоритм сортування методом бульбашки

Вхід: масив A: **array** [1..*n*] **of** B, де значення елементів масиву розташовані в довільному порядку і для значень типу B задане відношення < .

Вихід: масив A: **array** [1..*n*] **of** B, в якому значення розташовані в порядку зростання

for i **from** 1 **to** *n* –1 **do**
m: = *i*{ індекс кандидата в мінімальні елементи}
    for *j* **from** *i*+1 **to** n **do**
        if A [*j*]<A [*m*] **then**
        m:=*j* { новий кандидат у мінімальні}
    end for
end for
A[*i*]↔ A[*m*] { ставимо мінімальний елемент на місце}
end for.

5.2.4. Генерація перестановок

На множині перестановок природно можна визначити впорядкування на основі впорядкування елементів. Як-от, говорять, що перестановка $\langle a_1, a_2, \cdots a_n \rangle$ лексикографічно передує перестановці $\langle b_1, b_2, \cdots b_n \rangle$, якщо

$$\exists\, k \leq n, a_k < b_k \text{ та } \forall i < k,\ a_i = b_i.$$

Аналогічно кажуть, що перестановка $\langle a_1, a_2, \cdots a_n \rangle$ антилексикографічно передує перестановці $\langle b_1, b_2, \cdots b_n \rangle$, якщо

$$\exists\, k \leq n, a_k > b_k \text{ та } \forall i > k,\ a_i = b_i.$$

Розглянемо алгоритм генерації всіх перестановок елементів 1, ... , *n* в антилексикографічному порядку.

Масив *P*: **array** [1..*n*] **of** 1..*n* є глобальним і призначений для збереження перестановок.

Вхід: *n* – кількість елементів

Вихід: послідовність перестановок елементів 1, ... *n* в антилексикографічному порядку.

```
for i from 1 to n do
    P [i] : = i { ініціалізація}
end for
Antilex (n) {виклик рекурсивної процедури Antilex}.
```

Основна робота з генерації перестановок виконується рекурсивною процедурою Antilex.

Вхід: *m* – параметр процедури – кількість перших елементів масиву *P*, для яких генеруються перестановки.

Вихід: послідовність перестановок 1, ... , *m* в антилексикографічному порядку.

```
if m =1 then
    yield P{чергова перестановка}
else
for i from 1 to m do
Antilex (m – 1) {рекурсивний виклик}
if i < m then
    P[i]↔ P[m] {такий елемент}
    Reverse (m– 1) {зміна порядку елементів}
end if
end for
end if.
```

Допоміжна процедура Reverse переставляє елементи заданого відрізка масиву P в оберненому порядку.

Вхід: *k* – номер елемента, що задає відрізок масиву *P*, який підлягає перестановці в оберненому порядку.

Вихід: перші *k* елементів масиву *P* переставлені в оберненому порядку

```
j:=1{нижня межа діапазону, що повертається}
while j < k do
    P [j]↔ P[k]
    j: = j +1
    k: = k–1
end while.
```

Зауважимо, що шукану послідовність перестановок *n* елементів можна одержати з послідовності перестановок $(n-1)$ елементів в такий спосіб.

Потрібно виписати n блоків по $(n-1)!$ перестановок у кожному, що відповідають послідовності перестановок $(n-1)$ елемента в антилексикографічному порядку. Потім до всіх перестановок у першому блоці потрібно приписати справа n, у другому – $(n-1)$ і т.д. у спадному порядку, потім у кожному блоці (крім першого), до перестановок якого справа приписаний елемент i, треба в перестановках блока замінити усі входження елемента i на елемент n. В одержаній послідовності всі перестановки будуть різними і їх $n(n-1)! = n!$, тобто перераховано всі перестановки. При цьому антилексикографічний порядок дотриманий для послідовностей усередині одного блока, оскільки те саме було у вихідній послідовності і послідовностях на межах двох блоків (позаяк відбувається зменшення самого правого елемента).

Звернемося до процедури Antilex. Легко побачити, що в ній реалізована зазначена побудова. В основному циклі спочатку будується черговий блок – послідовність перестановок перших $(m-1)$ елементів масиву P (при цьому елементи $P[m]$, ... , $P[n]$ залишаються незмінними). Потім елемент $P[m]$ міняється місцями з черговим елементом $P[i]$. Виклик допоміжної процедури Reverse необхідний, оскільки остання перестановка в блоці є обертанням першої, а для генерації такого блока на черговому кроці циклу потрібно відновити вихідний порядок.

Розглянемо приклад.

Послідовність перестановок в антилексикографічному порядку для $n = 3$ буде такою: (1, 2, 3), (2, 1, 3), (1, 3, 2), (3, 1, 2), (2, 3, 1), (3, 2, 1).

5.3. Біноміальні коефіцієнти

Кількість сполучень $C(m, n)$ – це число різних n-елементних підмножин з m-елементної множини. Числа $C(m, n)$ використовуються у формулах розв'язування багатьох комбінаторних задач, їх також називають біноміальними коефіцієнтами.

Дійсно, розглянемо таку типову схему суджень при розв'язанні комбінаторної задачі.

Припустимо, потрібно визначити кількість підмножин m-елементної множини, які задовольняють деякій умові. Розіб'ємо задачу на підзадачі: Розглянемо окремо одноелементні підмножини, двоелементні і т.д., а потім просумуємо отримані результати. На кожному з кроків нам потрібно визначити кількість підмножин, і це можно зробити, використовуючи $C(m, n)$.

Числа $C(m, n)$ мають цілу низку корисних властивостей, які проаналізовані у цьому розділі, і допомагають при обчисленнях.

З основної формули для кількості сполучень, а саме:

$$C(m,n) = \frac{m!}{n!\,(m-n)!}$$

випливає кілька властивостей, які ми сформулюємо нижче.

Теорема 5.2.

1. $C(m, n) = C(m, m – n)$.
2. $C(m, n) = C(m – 1, n) + C(m – 1, n – 1)$.
3. $C(n,i)\, C(i,m) = C(m, n)C(n – m,I – m)$.

Доведення

1. $C(m, m-n) = \dfrac{m!}{(m-n)!\,(m-(m-n))!} = \dfrac{m!}{(m-n)!\,n!} = C(m,n).$

2. $C(m-1,n) + C(m-1,n-1) = \dfrac{(m-1)!}{n!\,(m-n-1)!} + \dfrac{(m-1)!}{(n-1)!\,\big(m-1-(n-1)\big)!} =$

$= \dfrac{(m-1)!}{n(n-1)!\,(m-n-1)!} + \dfrac{(m-1)!}{(n-1)!\,(m-n)(m-n-1)!} =$

$= \dfrac{(m-n)(m-1)! + n(m-1)!}{n(n-1)!\,(m-n)(m-n-1)!} = \dfrac{(m-n+n)(m-1)!}{n!\,(m-n)!} =$

$= \dfrac{m!}{n!\,(m-n)!} = C(m,n).$

3. $C(n,i)C(i,m) = \dfrac{n!}{i!\,(n-i)!} \cdot \dfrac{i!}{m!\,(i-m)!} = \dfrac{n!}{m!\,(i-m)!\,(n-i)!} =$

$= \dfrac{n!\,(n-m)!}{m!\,(i-m)!\,(n-i)!\,(n-m)!} = \dfrac{n!}{m!\,(n-m)!} \cdot \dfrac{(n-m)!}{(i-m)!\,(n-i)!} =$

$= C(n,m)C(n-m,i-m).$

Біном Ньютона

Кількість сполучень $C(m, n)$ називають також *біномними коефіцієнтами*. Зміст цієї назви встановлюється теоремою, відомою як *формула бінома Ньютона*.

Теорема 5.3. (*формула бінома Ньютона*)

$(x+y)^m = \sum_{n=0}^{m} C(m,n)x^n y^{m-n}$.

Доведення

Проведемо доведення за індукцією.

1. База, $m = 1$:

$$(x+y)^1 = x+y = 1x^1y^0 + 1x^0y^1 = C(1,0)x^1y^0 + C(1,1)x^0y^1 =$$
$$= \sum_{n=0}^{1} C(1,n)x^n y^{1-n}.$$

2. Індукційний перехід.

$$(x+y)^m = (x+y)(x+y)^{m-1} = (x+y)\sum_{n=0}^{m-1} C(m-1,n)x^n y^{m-n-1} =$$

$$= \sum_{n=0}^{m-1} xC(m-1,n)x^n y^{m-n-1} + \sum_{n=0}^{m-1} yC(m-1,n)x^n y^{m-n-1} =$$
$$= \sum_{n=0}^{m-1} C(m-1,n)x^{n+1} y^{m-n-1} + \sum_{n=0}^{m-1} C(m-1,n)\, x^n y^{m-n} =$$
$$= \sum_{n=0}^{m-1} (C(m-1,n-1) + C(m-1,n))x^n y^{m-n} + C(m-1,m-1)x^m y^0 =$$
$$= \sum_{n=0}^{m-1} C(m,n)x^n y^{m-n} + C(m,m)x^m y^{m-m} = \sum_{n=0}^{m} C(m,n)x^n y^{m-n}.$$

Наступні два твердження є наслідками доведеної теореми.

Твердження 5.1. $\sum_{n=0}^{m} C(m,n) = 2^m$.

Доведення

$$2^m = (1+1)^m = \sum_{n=0}^{n} C(m,n)1^n 1^{m-n} = \sum_{n=0}^{m} C(m,n).$$

Твердження 5.2. $\sum_{n=0}^{m}(-1)^n C(m,n) = 0$.

Доведення

$0 = (-1+1)^m = \sum_{n=0}^{m} C(m,n)(-1)^n 1^{m-n} = \sum_{n=0}^{m}(-1)^n C(m,n)$.

Зауважимо, що біномні коефіцієнти мають цілу низку цікавих властивостей.

Теорема 5.3. Мають місце такі тотожності:

1. $\sum_{n=0}^{m} nC(m,n) = m2^{m-1}$,

2. $C(m+n,k) = \sum_{i=0}^{k} C(m,i)C(n,k-i)$.

Доведення

1. Розглянемо послідовність чисел 1, ..., m. Спочатку всі підмножини довжини 0, потім усі підмножини довжини 1 і т.д. Існує $C(m, n)$ підмножин потужності n, і кожна з них має довжину n, таким чином, усього в цій послідовності буде $\sum_{n=0}^{n} nC(m, n)$ чисел. З іншого боку, кожне число x входить у цю послідовність $2^{|\{1,\dots,m\}\setminus\{x\}|} = 2^{m-1}$ разів, а усього чисел m.

2. $C(n + m,\ k)$ – це кількість способів вибрати k предметів із $m + n$ предметів. Їх можна вибирати в два прийоми. Спочатку вибрати i предметів із перших m предметів, а потім вибрати відсутні $k – i$ предметів із тих n предметів, що залишилися. Звідси загальна кількість способів вибрати k предметів складає $\sum_{i=0}^{k} C(m, i)C(n, k - i)$.

Цікавий ефективний спосіб рекурентного обчислення значень біномних коефіцієнтів, випливає з теореми 5.2 (властивість 2). Його можна подати в графічній формі, відомій як *трикутник Паскаля.*

У цьому рівнобедреному трикутнику кожне число (крім одиниць на бічних сторонах) є сумою двох чисел, що стоять над ним (рис .5.3). Кількість сполучень $C(m, n)$ знаходиться в $(m + 1)$-му ряду на $(n + 1)$-му місці.

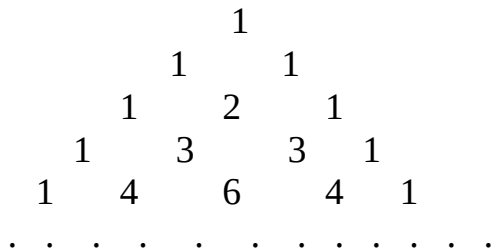


Рис. 5.3. Трикутник Паскаля

Висновки

Комбінаторика, або комбінаторний аналіз – це розділ математики, що розглядає задачі вибору та розташування елементів деякої, зазвичай, скінченної множини відповідно до заданих правил. Отже, якщо є множина, яка складається з n елементів (тобто потужність множини дорівнює n), то кожне таке правило визначає спосіб, за яким із елементів цієї вихідної множини побудована певна комбінаторна конфігурація. Відповідно метою комбінаторного аналізу є визначення алгоритмів побудови комбінаторних конфігурацій, їх дослідження та кількісне розв’язання задач переліку.

Основні формули комбінаторики використовуються у теорії ймовірностей. Зауважимо, що елементи вихідної множини вважаються різними і при побудові комбінаторних конфігурацій не можуть відбуватись повторення. Отже, комбінаторні конфігурації мають бути побудовані без повторень.

Питання, викладені в цьому розділі, розглянуто у літературі [1, 10, 11, 19, 20, 21]

Питання для самоконтролю.

1. Сформулюйте поширені задачі комбінаторного аналізу.
2. Дайте визначення розміщення.
3. Що являє собою розміщення без повторень?
4. Які типи перестановок можливі?
5. Як можна обчислити кількість перестановок?
6. Як обчислити кількість перестановой із повтореннями?
7. Яким чином визначають кількість сполучень?
8. В чому полягає алгоритм сортування методом бульбашки?
9. Що являє собою лексикографічний (антилексикографічний) порядок?
10. Які властивості мають біномні коефіцієнти? Опишіть їх.
11. Що характеризує трикутник Паскаля?

Задачі для самостійного розв'язування

1. Один із рядів глядацької зали містить 15 крісел. Скількома способами можна розмістити на них 15 людей?

2. Скількома способами можна розфарбувати повний граф на 6-ти вершинах шістьма кольорами? (Два способи вважаються різними, якщо деяка вершина при першому з них має один колір, а при другому – інший.)

3. З квадратної матриці розміром 10×10 вибирають 10 елементів таким чином, щоб ніякі два з них не належали до одного рядка. Скільки таких наборів з 10 елементів можна скласти?

4. Десять співробітників необхідно розподілити на шість робочих місць. Відомо, що кожний може працювати на будь-якому з них. Скількома способами можна здійснити таке призначення?

5. На трьох комп'ютерах C_1, C_2, C_3 необхідно розв'язати три задачі P_1, P_2, P_3. Для розв'язування кожної з них можна використати будь-який комп'ютер. Скількома способами можна спрямувати задачі на розв'язок?

6. В дитячій садок, в якому є 7 груп, приєднались 5 нових дітей, яких необхідно розподілити по групах, причому в одну групу можна спрямувати не більше однієї дитини. Скільки можливих варіантів розподілу існує?

7. Список контрольних завдань, з яких складають екзаменаційні білети, включає 19 запитань. Кожен білет має містити два різних запитання. Скільки різних екзаменаційних білетів можна скласти?

8. У виразі $(x+y)^{12}$ розкрили дужки і привели подібні члени. Який коефіцієнт буде відповідати виразу $x^4 y^8$?

9. Обчислити значення такої суми:

а) $C_7^1 + C_7^3 + C_7^5 + C_7^7$; *б*) $C_8^2 + C_8^4 + C_8^6 + C_8^8$.

11. У множині з 10 елементів зафіксовано чотири властивості p_1, p_2, p_3, p_4, якими можуть володіти або не володіти елементи множини. Як за допомогою методу включення-виключення описати ті елементи, в яких немає жодної з даних властивостей?

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бондаренко М.Ф. Комп'ютерна дискретна математика: підручник /М.Ф.Бондаренко, Н.В.Білоус, А.Г Руткас. – Харків : «Компанія СМІТ», 2004. – 480 с.

2. Вовк В.М. Оптимізаційні моделі економіки : навч. посібник / В.М. Вовк, Л.М. Зомчак. – Львів: ЛНУ імені Івана Франка, 2013. – 318 с.

3. Гавриленко С.Ю., Клименко А.М. Любченко Н.Ю. та ін. Теорія цифрових автоматів та формальних мов. – Харків, НТУ «ХПІ», 2010. – 176 с.

4. Гавриленко С.Ю., Клименко А.М. Носков. В.І Логіка дискретних автоматів: навч. посіб.–Х: НТУ «ХПІ», 2014.–129 с.

5. Гавриленко С.Ю. Формальні мови, граматики та автомати: Навчальний посібник/ Гавриленко С.Ю. – Харків: НТУ « ХПІ », 2021. – 133 с.

6. Гаврилків В.М. Формальні мови та алгоритмічні моделі: навч. посіб. – Івано-Франківськ: «Сімик», 2012. – 172 с.

7. Дацко М. В. Дослідження операцій в економіці: навч. посіб. / М. В. Дацко, М. М. Карбовник. – Львів: ПАІС, 2009. – 288 с.

8. Захарія Л. М., Заяць М. М. Формальні мови та граматики: навч. посіб.– Львів, «Львівська політехніка», 2016.– 196 с.

9. Зубенко В.В., Шкільняк С.С. Основи математичної логіка: навчальний посібник. Київ : НУБіП України, 2020. 102 с.

10. Кривий С.Л. Дискретна математика : вибрані питання / С. Л. Кривий. – Київ : Вид. дім "Києво-Могилянська акад.", 2007.

11. Кривий С.Л. Збірник задач з дискретної математики : вибрані питання / С. Л. Кривий, О. М. Ходзинський. – Київ : Бізнесполіграф, 2008.

12. Математична логіка. Практикум [Електронний ресурс]: навч. посіб. для студ. спеціальності 113 «Прикладна математика», освітньої програми «Наука про дані та математичне моделювання» / О.Л. Темнікова ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,37 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 76 с.

13. Нікітченко М. С. Прикладна логіка / М. С. Нікітченко, С. С. Шкільняк. – Київ : ВПЦ "Київ. ун-т", 2013.

14. Нікітченко М. С. Математична логіка та теорія алгоритмів / М. С. Нікітченко, С. С. Шкільняк. – Київ : ВПЦ "Київ. ун-т", 2008.

15. Нечіткі множини в системах управління та прийняття рішень: навч. посіб. / Т.А. Желдак, Л.С.Коряшкіна, С.А. Ус; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2020. – 383 с.

16. Новицький І.В. Дискретна математика у прикладах і задачах [Текст]: навч. посібник / І.В.Новицький, С.А. Ус. – Д.: НГУ, 2013, – 90 с.

17. Сопронюк Т.М. Системне програмування. Частина І. Елементи теорії формальних мов: Навчальний посібник у двох частинах. – Чернівці: ЧНУ, 2008. – 84 с.

18. Спекторський І.Я., Статкевич В.М. Формальні мови та автомати . Київ: КПІ ім. Ігоря Сікорьского, 2019. – 167 с.

19. Трохимчук Р.М. Дискретна математика / Р. М. Трохимчук. – Київ : Вид. дім "Персонал", 2010.

20. Трохимчук Р.М. Збірник задач і вправ з дискретної математики / Р. М. Трохимчук. – Київ : ВПЦ "Київ. ун-т", 2008.

21. Трохимчук Р.М. Збірник задач і вправ з теорії множин і відношень : навч. посіб. / Р. М. Трохимчук. – Київ : ВПЦ "Київ. ун-т", 2012.

22. Халецька З.П., В.В. Нарадовий Математична логіка та теорія алгоритмів: навч. посіб./ З.П. Халецька, В.В. Нарадовий. – Кропивницький : РВВ КДПУ ім. В. Винниченка, 2017. – 128 с

23. Хромой Я.В. Математична логіка / Я. В. Хромой. – Київ : Вища шк., 1983.

24. Хромой Я.В. Збірник задач і вправ з математичної логіки / Я. В. Хромой. – Київ : Вища шк., 1978.

25. Шкільняк, С. С. Математична логіка; Основи теорії алгоритмів : навч. посіб. / С. С. Шкільняк. — Київ : ДП «Вид. дім «Персонал», 2009. – 280 с.

Список позначень

$\vdash$ виведення с.

$\vee$ або

$\&$ і

$\neg$ заперечення

$\Rightarrow$ наслідування,

$\in$ належить

$\notin$ не належить

$\subset$ належність

$\varnothing$ порожня множина

$\rightarrow$ відображення

$\exists$ існує

$\forall$ будь-який

ДНФ диз’юнктивна нормальна форма

КНФ кон’юнктивна нормальна форма

ДДНФ досконала диз’юнктивна нормальна форма

ДКНФ досконала кон’юнктивна нормальна форма

$U(m,n)$ кількість розміщень (із повтореннями)

$A(m, n)$ або $(m)_n$ кількість розміщень (без повторень)

$P(n)$ кількість перестановок

$C(m, n)$ або $(\frac{n}{m})$ кількість сполучень, біномні коефіцієнти

Предметний покажчик

Т

У

Ф

Х

Ц

Ч

Ш

Навчальне видання

Слєсарєв Володимир Вікторович
Новицький Ігор Валерійович
Ус Світлана Альбертівна

ДИСКРЕТНА МАТЕМАТИКА

Навчальний посібник

Редактор Є.М. Ільченко

Підписано до друку 25.04.23. Формат 30х42/4.
Папір офсет. Ризографія. Ум. друк. арк. 10,5
Обл.-вид. арк. 13,6 . Тираж 100 пр. Зам. №

Підготовлено до друку та видруковано
в Національному технічному університеті
«Дніпровська політехніка»
Свідоцтво про внесення до Державного реєстру ДК № 1842 від 11.06.2004 р.
49005, м Дніпро, просп. Д.Яворницького, 19

Новицький Ігор Валерійович – доктор технічних наук, професор кафедри системного аналізу та управління Національного технічного університету «Дніпровська політехніка», автор понад 80 наукових та методичних праць, серед яких дві монографії, 7 навчальних посібників, 5 авторських свідоцтв. *Основні напрямки наукової діяльності*: автоматична оптимізація технологічних процесів підготовки руд перед збагаченням, моделювання і оптимізація технологічних процесів.

Слєсарєв Володимир Вікторович – доктор технічних наук, професор кафедри системного аналізу та управління Національного технічного університету «Дніпровська політехніка». Опублікував понад 130 наукових та методичних праць, серед яких дві монографії, 4 навчальних посібника, більше ніж 80 наукових статей. *Наукові інтереси:* питання моделювання та автоматизації складних потокових технологій, методи автоматизованого керування та діагностики технологічних процесів; управління складними об'єктами гірничо-металургійних підприємств; розробка інтелектуальних систем прийняття рішень та інформаційних систем.

Ус Світлана Альбертівна – кандидат фізико-математичних наук, доцент, професор кафедри системного аналізу та управління Національного технічного університету «Дніпровська політехніка», експерт Національного агентства забезпечення якості вищої освіти. Опублікувала понад 170 наукових та методичних праць, серед яких дві монографії, 16 навчальних посібників, понад 70 наукових статей. *Наукові інтереси:* прийняття рішень в умовах невизначеності, моделювання складних систем, оптимізація розподілу матеріальних потоків у транспортно-логістичних системах з неперервно-розподіленим ресурсом.