МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

М. В. Новожилова

ПРИКЛАДНІ ЗАДАЧІ ДИСКРЕТНОГО АНАЛІЗУ

КОНСПЕКТ ЛЕКЦІЙ

(для здобувачів першого (бакалаврського) рівня вищої освіти всіх форм навчання зі спеціальностей 122, F3 – Комп'ютерні науки, 126, F6 – Інформаційні системи та технології)

Харків
ХНУМГ ім. О. М. Бекетова
2025

УДК 519.8:004](075.8)

Новожилова М. В. Прикладні задачі дискретного аналізу : конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти всіх форм навчання зі спеціальностей 122, F3 – Комп'ютерні науки, 126, F6 – Інформаційні системи та технології / М. В. Новожилова ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2025. – 83 с.

Автор
д-р фіз.-мат. наук, проф. М. В. Новожилова

Рецензент
А. Л. Литвинов, доктор технічних наук, професор, професор кафедри комп'ютерних наук та інформаційних технологій ХНУМГ ім. О. М. Бекетова

Рекомендовано кафедрою комп'ютерних наук та інформаційних технологій, протокол № 17 від 02.05.2025

У конспекті лекцій розглянуто такі теми, як теорія множин, математична логіка, алгебра висловлень, постановка та розв'язання задач комбінаторики характеристики та складність задач дискретної оптимізації, теорія графів, задачі дискретної оптимізації на графах, методи розв'язання задачі комівояжера, а також генетичні алгоритми.

Досягнутий рівень компетентності дозволить ефективно застосовувати інструментарій дискретного аналізу та комбінаторної оптимізації під час виконання завдань у професійній діяльності студентами освітніх програм «Комп'ютерні науки», «Комп'ютерні науки. Управління проєктами», спеціальностей 122, F3 – Комп'ютерні науки та 126, F6 – Інформаційні системи та технології.

ЗМІСТ

ВСТУП

Метою проведення лекційного курсу з дисципліни «Прикладні задачі дискретного аналізу» є засвоєння студентами знань про моделі та методи дискретного аналізу та комбінаторної оптимізації на прикладах реальних ситуацій предметної області з використанням сучасних інформаційних середовищ проєктування, які можуть безпосередньо використовуватися в галузі комп'ютерних наук та інформаційних технологій.

Вивчення дисципліни «Прикладні задачі дискретного аналізу» безпосередньо спирається на дисципліни «Оптимізаційні методи та моделі», «Програмування».

У результаті вивчення дисципліни здобувач першого (бакалаврського) рівня вищої освіти за спеціальностями 122, F3 – Комп'ютерні науки та 126, F6 – Інформаційні системи та технології знатиме технології моделювання дискретного аналізу при розв'язанні задач галузі комп'ютерних наук, отримає навички застосування знання фундаментальних і природничих наук до розв'язання задач дискретного аналізу та програмування щодо складання та реалізації алгоритмів розв'язання задач дискретного аналізу, навчиться оцінювати складність та ефективність алгоритмів та способів передачі інформації, набуде практичних навичок застосування математичного та алгоритмічного апарату дискретного аналізу для моделювання та розв'язання задач інженерної та інформаційної інфраструктури міста.

Дисципліна складається з трьох змістових модулів та 8 тем, у яких розглядаються такі питання.

Змістовий модуль 1 Математичний апарат дискретного аналізу (теми 1–3).

Дискретний аналіз – математична дисципліна, у якій досліджуються явища та процеси, що мають дискретний характер, або такі, що можуть бути приведені до дискретного виду для спрощення обчислень. Складовими дискретного аналізу є теорія множин, математична логіка, комбінаторний аналіз, теорія графів, комбінаторика.

Змістовий модуль 2 Задачі дискретної оптимізації: постановка, властивості, обчислювальна складність (теми 4–6).

Задачі дискретної оптимізації – це задачі знаходження екстремуму функції, заданої на дискретній множині точок. Предметна галузь дискретної оптимізації – це дослідження структури і властивостей різних класів NP-складних задач, розробки та обґрунтування методів їхнього розв'язання,

дослідження ефективності запропонованих методів, створення відповідного програмного забезпечення, що дає можливість будувати нові інформаційні технології та розв’язувати нові прикладні задачі. Теорія графів – невід’ємна складова дискретного аналізу як математичного фундаменту дискретної оптимізації.

Змістовий модуль 3 Методи розв’язання задач дискретних оптимізаційних задач ресурсного забезпечення міста (теми 7, 8).

До задач дискретної оптимізації на графах зводяться задачі проєктування інженерної інфраструктури, геометричного проєктування, визначення найкоротшого шляху, задачі про призначення та багато інших. За математичною постановкою ці задачі переважно є задачами пошуку мінімального шляху, мінімального покриття та задачею комівояжера. Для розв’язання таких задач застосовується цілий арсенал різних оптимізаційних методів: точні (метод гілок та меж), наближені та різноманітні евристики, яскравим прикладом яких є генетичні алгоритми.

Цей конспект лекцій містить теоретичний матеріал за дисципліною, який супроводжується контрольними запитаннями та завданнями, а також перелік використаних та додаткових джерел, зокрема посилання на відповідні ресурси неформальної освіти.

ТЕМА 1
ТЕОРІЯ МНОЖИН. ВЛАСТИВОСТІ, ОСНОВНІ ОПЕРАЦІЇ НАД МНОЖИНАМИ

Основні питання

1.1 Поняття множини. Відображення множин. Відношення.
1.2 Способи визначення множин.
1.3 Операції над множинами.
1.4 Поняття про алгебру Кантора.
1.5 Декартовий добуток кількох множин.

1.1 Поняття множини. Відображення множин. Відношення

Визначення 1.1. Множина – це сукупність певних об’єктів, що називаються елементами множини, які можна добре розрізняти.

Приклад 1.1. Наведемо приклади множин:

- множина закладів вищої освіти України;
- множина студентів певного навчального закладу;
- множина підприємств міста;
- множина літер українського алфавіту.

Зокрема, у математиці виокремлюють такі види множин:

- множина дійсних чисел;
- множина прямокутних трикутників;
- множина векторів у тривимірному просторі.

Довільна множина сама може бути елементом деякої множини, котра може входити як елемент до складу інших множин.

Надалі елементи, з яких складаються множини, позначаються малими латинськими літерами, а самі множини – великими латинськими літерами:

$$a \in A \qquad a \notin A.$$

Визначення 1.2. Множина A називається підмножиною множини B, якщо всі елементи множини A є елементами множини B:

$$A \subset B \text{ або } A \subseteq B .$$

Визначення 1.3. Дві множини називаються рівними, якщо всі елементи однієї множини є елементами другої множини, і навпаки:

$$A = B \text{ означає, що } A \subseteq B \text{ та } B \subseteq A.$$

Визначення 1.4. Множина, яка не містить жодного елемента, називається порожньою і позначається ∅.

Визначення 1.5. Кількість елементів множини A називається її потужністю і позначається |A|.

Виділимо серед множин довільного характеру числові множини. Числові множини поділяються на скінченні та нескінченні.

Потужність скінченної множини дорівнює кількості елементів цієї множини.

Нескінченні числові множини (неперервна математика, зокрема, математичний аналіз) також поділяються на злічені та незлічені (рис. 1.1).

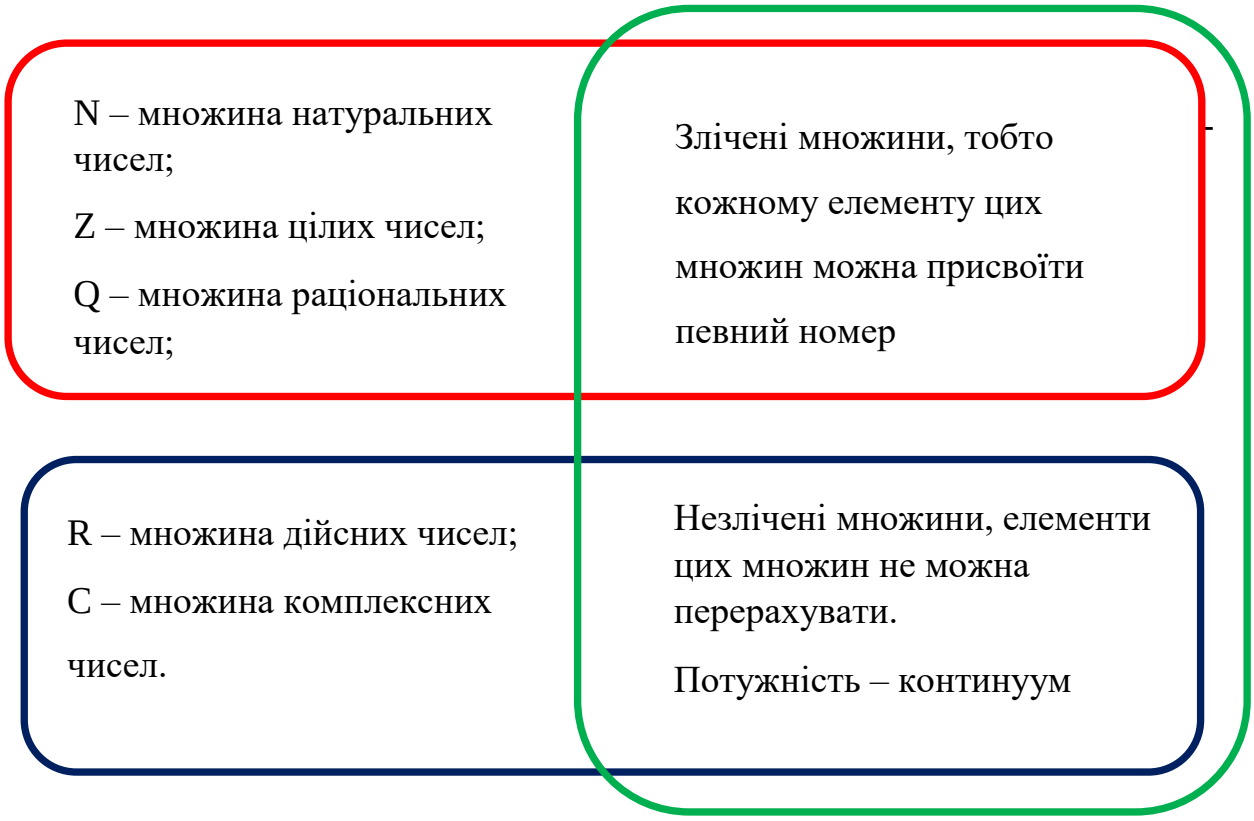


Рисунок 1.1 – Класифікація числових множин за потужністю

Визначення 1.6. Злічена множина – це така нескінченна множина, елементи якої можна занумерувати натуральними числами.

Приклад 1.2. Множина усіх точок площини з цілими координатами є зліченою.

Говорять, що множина натуральних чисел має потужність $\aleph_0$ (читається «алеф-нуль»). Таким чином, всі злічені множини мають потужність $\aleph_0$.

Визначення 1.7. Незлічена множина – це така нескінченна множина, потужність якої більша за потужність $\aleph_0$.

Незлічені множини мають потужність континуум, або $\aleph_1$ (рис. 1.1).

Теорема 1.1 (Кантор Г). Множина дійсних чисел, що міститься у відрізку [0,1] є незліченою.

Довільний відрізок дійсної числової осі має потужність континуум, як і вся числова пряма.

1.2 Способи визначення множин

Існують три основні засоби визначення множин:

1. Перерахування елементів для скінченних множин. У цьому випадку елементи множин записуються у фігурних дужках:

$$M_1 = \{-2, -1, 0, 1, 2, 3\}.$$

2. Визначення множини за допомогою породжуючої процедури.

Це індуктивні або рекурентні правила або формули:

– індуктивне або рекурсивне визначення:

$$M_2 = \{a_k \in R,\ a_k = k \cdot a_k,\ k \in N\};$$

– рекурентне визначення:

$$M_3 = \{a_k \in R,\ a_k = 5 + a_{k-1},\ k \in N\}.$$

3. Опис властивостей елементів множини:

$$M_4 = \{\text{натуральні числа менші за число } 100\}.$$

Для останнього способу визначення множини важливим є існування розпізнавальної процедури, яка дає змогу точно визначити, є цей об’єкт елементом множини, чи ні.

Приклад 1.2. Множина

$$M_5 = \{\text{хороші книжки, видані у XX столітті}\}$$

не може вважатися заданою коректно, бо поняття «хороша книга» суб'єктивне, а отже, неоднозначне.

Множина

$$M_6 = \{\text{літературні твори, що відзначені Нобелівською премією}\}$$

задана цілком коректно. Розпізнавальною процедурою тут є перелік всіх творів, відзначених цією премією.

Визначення 1.8. Множина A всіх підмножин, що складається з елементів скінченної множини A, включаючи порожню множину $\varnothing$ та саму множину A, називається булеаном.

Позначається булеан P(A).

Приклад 1.3. Побудуємо булеан для множини A = {a, b, c}.

Тоді P(A) = {$\varnothing$, {a, b, c}, {a}, {b}, {c}, {ab}, {ac}, {bc}}

1.3 Операції над множинами

Для ілюстрації наступних означень використовуються так звані кола Ейлера або діаграми Венна. На цих діаграмах множини зображуються на площині у вигляді кіл або інших плоских фігур, незалежно від їхньої потужності. Результат виконання операцій позначається відповідно кольором або штриховкою.

Визначення 1.9. Об'єднанням двох множин A та B називається множина C,

$$C = A \cup B,$$

яка складається як з елементів множини A, так із елементів множини B, взятих по одному разу.

На рисунку 1.2 множина C, що є об'єднанням двох множин A та B, є заштрихованою.

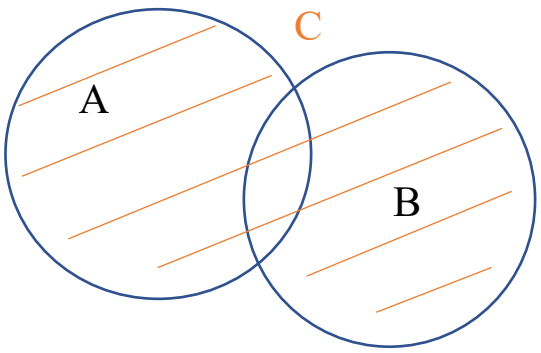


Рисунок 1.2 – Множина C є об'єднанням двох множин A та B

Визначення 1.10. Перетином двох множин A і B називається множина C:

$$C = A \cap B,$$

елементами якої є ті і тільки ті елементи, що належать одночасно множинам A та B (рис. 1.3).

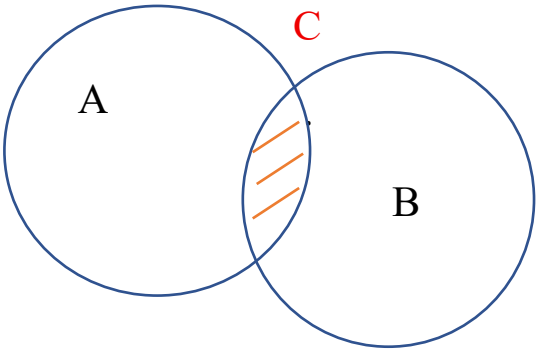


Рисунок 1.3 – Множина C є перетином двох множин A та B

Визначення 1.11. Різницею двох множин A та B називається множина C, елементами якої є ті і тільки ті елементи множини A, що не є елементами множини B (на рис. 1.4 множина C заштрихована).

$$C = A \setminus B.$$

Визначення 1.12. Симетричною різницею двох множин A та B називається множина C, елементами якої є ті і тільки ті елементи множин A та B, що не належать цим множинам одночасно.

$$C = A \Delta B, \text{ причому } A \Delta B = (A \setminus B) \cup (B \setminus A).$$

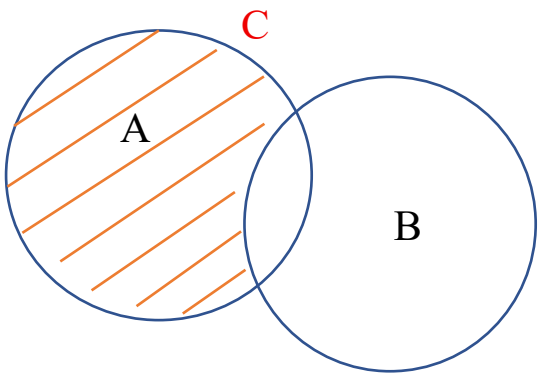


Рисунок 1.4 – Множина C є різницею двох множин A та B

Визначення 1.13. Якщо $A \supseteq B$, різниця $A \backslash B$ називається доповненням множини B у множині A і позначається $\overline{B}_A$.

Визначення 1.14. Універсум U – це множина, що містить всі можливі елементи.

Зазвичай доповнення розглядається в універсумі, тобто $\overline{A} = U \backslash A$ (на рис. 1.5 універсум U позначений зеленим прямокутником).

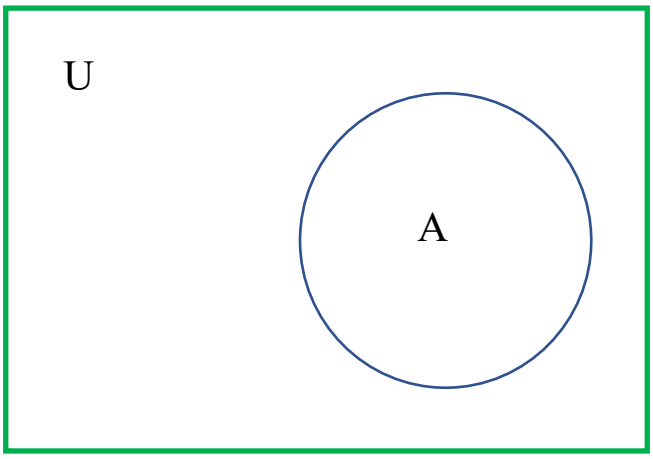


Рисунок 1.5 – Універсум U містить множину A

1.4 Поняття про алгебру Кантора

Виникнення теорії множин пов’язане з іменем німецького математика Георга Кантора (1845–1918). Наприклад, Г. Кантор ввів поняття потужності (об’єму) та еквівалентності множин.

Множина усіх підмножин деякої універсальної множини U разом із заданими на ній операціями $\cup$, $\cap$, $\overline{X}$, де X – це певна множина, така, що $X \subseteq U$, називається алгеброю множин.

Іншими словами, алгебра множин на U – це непорожня підмножина $\Re$ степеню множини X(P(X)), яка є замкненою відносно операцій об’єднання ($\cup$), перетину ($\cap$) та доповнення ($\overline{X}$) відносно X.

Поняття алгебри Кантора у контексті множин тісно пов’язане з булевою алгеброю.

Визначення 1.15. Якщо ми розглянемо всі підмножини {X} універсуму U, тобто $|\{X\}| = P(U)$, то кортеж

$$(P(X), \cup, \cap, \overline{X}, \emptyset, X)$$

утворює булеву алгебру.

Основні властивості булевої алгебри множин:

Комутативність: $A \cup B=B \cup A$,
$A \cap B=B \cap A$.

Асоціативність: $(A \cup B) \cup C=A \cup (B \cup C)$,
$(A \cap B) \cap C=A \cap (B \cap C)$.

Дистрибутивність: $A \cup (B \cap C)=(A \cup B) \cap (A \cup C)$,
$A \cap (B \cup C)=(A \cap B) \cup (A \cap C)$.

Тотожні елементи: $A \cup \emptyset=A$ ($\emptyset$ – нульовий елемент алгебри),
$A \cap U = A$ (U – одиничний елемент).

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

Закони де Моргана: $\overline{A \cap B} = \overline{A} \cup \overline{B}$,
$\overline{A \cup B} = \overline{A} \cap \overline{B}$.

Розглянемо на прикладі, як виконуються в алгебрі Кантора операції, якщо множини задані двійковим представленням.

Приклад 1.4. Нехай множина A має код 01010111, а множина B – 11100011.

Тоді:

– об’єднання $A \cup B$ має код 11110111 (умовно кажучи, операції об’єднання відповідає «майже сума», для якої 1 + 1 = 1);

– перетин $A \cap B$ має код 01000011 (операції перетину відповідає звичайний добуток);

– доповнення $\overline{A}$ має код 10101000 (операції доповнення відповідає так звана операція інверсії, яка нулі замінює одиницями і навпаки);

– симетрична різниця $A \Delta B$ має код 10110100 (операції симетричної різниці відповідає так зване логічне додавання, або додавання за модулем 2, для якої 1 + 1 = 0);

– різниця A\B має код 10100000.

1.5 Декартовий добуток кількох множин

Визначення 1.15. Декартовим або прямим добутком множин $A_1,\dots A_n$

$$A_1 \times A_2 \times \dots \times A_n$$

називається множина всіх впорядкованих наборів $\langle a_1,\dots,a_n\rangle$, $a_i \in A_i$, що називаються векторами або кортежами.

Якщо всі множини A_i рівні між собою, то декартів добуток позначається A^n і називається степенем множини A.

Приклад 1.5. Нехай множини A = B = R, тоді $A \times B = R^2$ є множиною пар (x, y) дійсних чисел. Зображуються елементи цього добутку точками на площині xOy декартової системи координат.

Точку тривимірного простору визначають трійкою дійсних чисел (x, y, z), де $x, y, z \in R$, отже,

$$(x, y, z) \in R^3.$$

Контрольні запитання та завдання

1. Що таке множина? Як позначаються множини та їхні елементи?
2. Які множини називаються рівними?
3. Яка множина називається порожньою? Як вона позначається ?
4. Формула симетричної різниці двох множин. Надайте приклад застосування.
5. Формула різниці двох множин. Надайте приклад застосування.
6. Надайте графічну ілюстрацію симетричної різниці двох множин за визначенням 1.12.
7. Поясніть поняття універсуму U.
8. Надайте графічну ілюстрацію доповнення множини до універсуму за визначеннями 1.13 та 1.14.
9. Як в алгебрі Кантора задаються операції над множинами?
10. Надайте визначення декартового (прямого) добутку множин $A_1,\dots A_n$.
11. Як визначають точку тривимірного простору?
12. Як можна виразити операції перетину, різниці та симетричної різниці через сумісне застосування операцій доповнення та об’єднання?

ТЕМА 2 МАТЕМАТИЧНА ЛОГІКА: АЛГЕБРА ВИСЛОВЛЮВАНЬ, ФУНКЦІЇ ІСТИННОСТІ

Основні питання

2.1 Відношення. Відображення множин.
2.2 Функції.
2.3 Алгебра висловлювань.
2.4 Операції над висловлюваннями.
2.5 Формули алгебри висловлювань. Таблиця істинності.

2.1 Відношення. Відображення множин

Визначення 2.1. Відношенням G на декартовому добутку множин A та B називається підмножина

$$G \subset A \times B, < a, b > \in G. \qquad (2.1)$$

Вираз $< a, b > \in G$ означає, що елементу $a \in A$ поставлений у відповідність елемент $b \in B$.

Визначення 2.2. Множина елементів $b \in B$, які відповідають елементу $a \in A$, називається образом елемента a відношення G.

Визначення 2.3. Множина елементів $a \in A$, які співставляються елементу $b \in B$, називається прообразом елемента b відношення G.

Введемо у розгляд множини A_1 та B_1 такі, що $A_1 \subseteq A$ і $B_1 \subseteq B$.

Нехай $G \subset A_1 \times B_1$, тоді $A_1 = пр_1G$ називається областю визначення відношення G, а $B_1 = пр_2G$ – областю значень.

Розглянемо декартовий добуток множин $A_1 \times A_2 \times \ldots \times A_n$.

Визначення 2.4. Підмножина $\Re \subseteq A^n$ називається n-місцевим (n-арним) відношенням на множині A.

Одномісцеве відношення – це просто підмножина в A. Найчастіше доводиться мати справу з двомісцевими відношеннями, які називають бінарними, тобто бінарне відношення на множині A – це підмножина в A^2.

Той факт, що елементи a та b знаходяться у відношенні $\Re$, позначається так:

$$a \, \Re \, b.$$

Приклад 2.1. Нехай A = N (A – множина натуральних чисел).

Задані відношення:

$\Re_1$ – мати спільний дільник, відмінний від 1;

$\Re_2$ – елемент a є дільником елемента b;

$\Re_3$ – $a < b$.

Розглянемо пари чисел: ⟨5,8⟩, ⟨7,13⟩, ⟨4,12⟩, ⟨9,6⟩. Необхідно встановити відношення на цих парах чисел.

Відповідь. Зрозуміло, що $4\ \Re_1\ 12$, $9\ \Re_1\ 6$, $4\ \Re_2\ 12$, $5\ \Re_3\ 8$, $7\ \Re_3\ 13$, $4\ \Re_3\ 12$.

Визначення 2.5. Відношення G називається функціональним, якщо образ кожного елемента $a \in пр_1G$ є одноелементною множиною в $пр_2G$.

Функціональне відношення називається відображенням множини A у множину B.

Надалі розглядатимемо відображення множин. Якщо

$$A_1 = пр_1G = A,$$

то відображення називається скрізь визначеним. У протилежному випадку – частково визначеним.

Якщо $B_1 = пр_2G = B$, то відображення G називається «відображенням на» або сюр'єктивним.

У протилежному випадку відображення G називається «відображенням в».

Визначення 2.6. Нехай $a_i, b_i \in G$, $i = 1,2$. Відображення G називається ін'єктивним або однозначним, якщо

$\forall a_1, a_2 \in пр_1G$ такі, що $a_1 \neq a_2$, їхні образи теж різні, тобто $b_1 \neq b_2$.

$\forall$ – квантор загальності, логічна константа, що інтерпретується «для всіх», «для будь-якого».

$\exists$ – квантор існування, логічна константа, що інтерпретується «існує», «є принаймі один», «для деяких».

Визначення 2.7. Якщо відображення G скрізь визначене, сюр'єктивне та ін'єктивне, то воно називається бієктивним або взаємно однозначним.

2.2 Функції

Відображення будемо називати просто функціями.

Відображення $f : A \to B$, яке ставить у відповідність кожному елементу множини A єдиний елемент множини B, є функцією. Для функції f використовується одне з позначень:

$$f : A \to B$$

або

$$b = f(a),\ a \in A \text{ та } b \in B.$$

Також застосовуються такі позначення:

$\forall a \in A$, $b = f(a)$ – визначення образу елемента a функції f, a – аргумент функції f, b – значення функції f.

$\forall b \in B$, $a = f^{-1}(b)$ – визначення прообразу $a \in A$ елемента b.

Визначення 2.8. Функція

$$f: A_1 \times A_2 \times \ldots \times A_n \to B$$

називається n-місцевою або n-вимірною.

2.3 Алгебра висловлювань

Висловлюванням називають оповідальне речення тієї чи іншої мови, про яке можна сказати, що воно або хибне, або істинне.

Просте (елементарне) висловлювання – це висловлювання, яке не включає як самостійні частини інші висловлювання.

Поставимо кожному висловлюванню числове значення за таким правилом: якщо висловлювання істинне, то йому відповідає 1, якщо хибне, то 0.

Приклад 2.2. Чи є наведені вирази простим висловлюванням?

Якщо вираз є висловлюванням, то яким саме – істинним чи хибним.

(1) Число 357 кратне 7.

(2) Це речення складається із шести слів.

(3) Хай живе кібернетика!

(4) Женева – столиця Швейцарії.

(5) Не існує найбільшого простого числа.

(6) Чи існує найменше просте число?

(7) Будьте уважні та взаємно ввічливі.

(8) Сонце зійшло.

(9) Виходячи з кімнати, вимикайте світло.

(10) Нерівність $3 < 2$ хибна.

(11) Це висловлювання – хибне.

2.4 Операції над висловлюваннями

Зазвичай конкретні елементарні висловлювання позначають малими латинськими літерами: a, b, c, … (інколи з індексами), а значення висловлювань – істинно та хибно – відповідно символами 1, та 0 (або I та X, а в англомовній літературі – відповідно T і F).

Операції над висловлюваннями задаються за допомогою так званих таблиць істинності, які визначають істинність або хибність новоутвореного висловлювання.

Визначення 2.9. Заперечення – унарна операція. Нехай a – висловлювання, то його заперечення позначається $\bar{a}$ і визначається таблицею істинності (табл. 2.1):

Таблиця 2.1 – Таблиця істинності для операції заперечення

a	$\bar{a}$
1	0
0	1

Висловлювання a істинне тоді і тільки тоді, коли $\bar{a}$ є хибним, читається «не a».

Визначення 2.10. Кон’юнкція – бінарна операція, що діє над двома висловлюваннями: a та b. Позначається $a \wedge b$, читається «a і b».

Кон’юнкція визначається таблицею істинності (табл. 2.2):

Таблиця 2.2 – Таблиця істинності для операції кон’юнкції

a	b	$a \wedge b$
1	1	1
1	0	0
0	1	0
0	0	0

Висловлювання a ∧ b – істинне лише тоді і тільки тоді, коли a і b – істинні одночасно.

Визначення 2.11. Диз'юнкція – бінарна операція, що діє над двома висловлюваннями: a та b. Позначається як a ∨ b, читається «a або b». Визначається таблицею істинності (табл. 2.3):

Таблиця 2.3 – Таблиця істинності для операції кон'юнкції

a	b	a ∨ b
1	1	1
1	0	1
0	1	1
0	0	0

Висловлювання a ∨ a хибне тоді і лише тоді, коли a і b хибні одночасно.

Розглянемо ще дві важливі операції.

Визначення 2.12. Імплікація – бінарна операція, що діє над двома висловлюваннями: a та b. Позначається a → b, читається «Якщо a, то b». Імплікація є хибною тоді й лише тоді, коли перше висловлювання є істинним, а друге – хибним.

Визначення 2.13. Еквівалентність – бінарна операція, що діє над двома висловлюваннями: a та b. Позначається a ~ b, читається «a тоді і тільки тоді, коли b». Еквівалентність є істинною тоді і тільки тоді, коли обидва висловлювання мають однакове значення.

Таблиці істинності операцій імплікації та еквівалентності мають вигляд (табл. 2.4).

Крім того, розглядаються змінні висловлення, які позначатимемо латинськими літерами x, y, z, … (інколи з індексами) і називатимемо також пропозиційними змінними.

Після підстановки замість пропозиційної змінної певного елементарного висловлення ця змінна набуде відповідного значення (1 або 0).

Таблиця 2.4 – Таблиця істинності для операцій імплікації та еквівалентності

x	y	x → y	x ~ y
0	0	1	1
0	1	1	0
1	0	0	0
1	1	1	1

Окремі елементарні висловлення можна з’єднувати між собою за допомогою певних зв’язок (сполучників), утворюючи складені висловлення.

Приклад 2.3. Вирази:

– число 5 – просте та число 23 – просте;

– Київ – столиця України, а Берн – столиця Швейцарії;

– якщо число 27 кратне 3, то число 27 кратне 6;

– неправильно, що 2 < 3, або неправильно, що 4 < 3 – є складеними висловленнями, що утворені з простіших (елементарних) висловлень шляхом використання зв’язок

і; а; якщо …, то; неправильно, що та або.

Використання цих мовних зв’язок трактується як виконання над висловленнями зазначених вище логічних операцій, різні назви та позначення яких наведені в таблиці 2.5.

Таблиця 2.5 – Опис логічних операцій

Назва	Визначення
Кон’юнкція (логічне множення, логічне і)	∧ &
Диз’юнкція (логічне додавання, логічне або)	∨
Заперечення (логічне ні)	¬ ′ ¯
Імплікація (логічне якщо, …то…)	→ ⊃ ⇒
Еквівалентність (рівнозначність)	~ ↔ ≡

Зазвичай використовуються перші із наведених назв і позначень.

Отже, з елементарних висловлювань і пропозиційних змінних за допомогою означених операцій і дужок утворюються складені висловлювання, яким відповідають формули або вирази.

Зауважимо, що символам логічних операцій відповідають у звичайній мові такі мовні зв’язки, або сполучники:

– ∧ – і; та; а; але; хоч; разом із; незважаючи на; …

– ∨ – або; чи; хоч (принаймні) одне з; …

– ¬ – не; неправильно, що; …

– → – якщо (коли) … , то (тоді)…; … імплікує …; із … випливає …; у разі … має місце …; …

– ~ – … тоді й тільки тоді, коли …; … якщо й тільки якщо …; … еквівалентне …; … рівносильне.

Застосовуючи пропозиційні змінні та символи логічних операцій, будь-яке складене висловлювання можна формалізувати, тобто перетворити на формулу, яка виражатиме (задаватиме) його логічну структуру.

Приклад 2.4. Висловлювання.

Якщо число 24 кратне 2 і 3, то число 24 кратне 6, має таку логічну структуру:

$$(a \wedge b) \rightarrow c.$$

Пропозиційній змінній a відповідає елементарне висловлювання «Число 24 кратне 2», змінній b – висловлювання «Число 24 кратне 3», а змінній c – висловлювання «Число 24 кратне 6».

2.5 Формули алгебри висловлювань. Таблиця істинності

За аналогією зі звичайними числовими алгебрами, де змінні можуть набувати значень із певної множини чисел (напр., цілі, раціональні або дійсні числа), розглянемо алгебру висловлювань, у якій значеннями відповідних змінних будуть якісь висловлювання.

Операціями алгебри висловлювань будуть означені вище кон'юнкція, диз'юнкція, заперечення, імплікація та еквівалентність. Застосовуючи до елементарних висловлювань і пропозиційних змінних ці операції, діставатимемо складені висловлювання, яким відповідатимуть формули (або вирази) алгебри висловлювань. Для запису і дослідження цих формул властивостей використовують формальні мови, тобто певні множини слів у деякому алфавіті.

Алфавіт найбільш поширеної формальної мови алгебри висловлювань складається з трьох груп символів:

1) символи елементарних висловлювань і пропозиційних змінних: a, в, c, … та x, y, z, … (інколи з індексами);

2) символи операцій: $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\sim$;

3) допоміжні символи – круглі дужки: (і).

Із символів цього алфавіту будують пропозиційні формули або просто формули алгебри висловлювань за індуктивним правилом:

1) усі пропозиційні змінні та елементарні висловлювання є формулами;

2) якщо A та B – формули, то вирази $(A \wedge B)$, $(A \vee B)$, $(\neg A)$, $(A \to B)$, $(A \sim B)$ також є формулами (для всіх цих виразів формули A та B є підформулами);

3) інших формул, крім тих, що побудовані за правилами 1) та 2), немає.

Формули алгебри висловлювань позначатимемо великими латинськими літерами A, B, …, F,….

Приклад 2.5. Визначити, чи є послідовність символів формулою алгебри висловлювань.

(а) $((x \to y) \wedge z) \sim ((\neg x) \to (y \vee z))$; (так)

(б) $(((\neg x) \wedge y) \to (x \sim ((\neg y) \vee x)$. (ні)

Нехай $p_1, p_2, \ldots, p_n$ – це всі пропозиційні змінні, що входять до формули A, позначатимемо цей факт $A(p_1, p_2, \ldots, p_n)$.

Формулі $A(p_1, p_2, \ldots, p_n)$ поставимо у відповідність бінарну функцію

$$f(p_1, p_2, \ldots, p_n), \tag{2.2}$$

що визначена на множині впорядкованих наборів $(p_1, p_2, \ldots, p_n)$, де кожна p_i набуває значення у множині $B = \{0, 1\}$.

Значення (0 або 1) функції $f(p_1, p_2, \ldots, p_n)$ на наборі значень $a_1, a_2, \ldots, a_n$ її змінних $p_1, p_2, \ldots, p_n$ дорівнює значенню формули $A(p_1, p_2, \ldots, p_n)$ при підстановці до неї замість пропозиційних змінних $p_1, p_2, \ldots, p_n$ значень $a_1, a_2, \ldots, a_n$, відповідно.

Зауважимо, що кількість елементів в області визначення функції $f(p_1, p_2, \ldots, p_n)$ дорівнює 2^n.

Приклад 2.6. Обчислити значення формули алгебри висловлювань

$$(((a \to (\neg b)) \to (b \wedge ((\neg c) \to a))) \sim (\neg a))$$

на наборі (1,1,0) значень її змінних, тобто за умови, що a = 1, b = 1, c = 0.

Для цього за допомогою індексів занумеруємо порядок виконання операцій у цій формулі:

$$(((a \to_4 (\neg_1 b)) \to_7 (b \wedge_6 ((\neg_2 c) \to_5 a))) \sim_8 (\neg_3 a)).$$

Підставимо замість пропозиційних змінних a, b, c задані вище значення, дістанемо

$$(((1 \to_4 (\neg 11)) \to_7 (1 \wedge_6 ((\neg_2 0) \to_5 1))) \sim_8 (\neg_3 1)).$$

Обчисливши операції 1, 2 та 3, дістанемо вираз

$$(((1 \to_4 0) \to_7 (1 \wedge_6 (1 \to_5 0))) \sim_8 0),$$

який після обчислення операцій 4 та 5 набуде вигляду

$$((0 \to_7 (1 \wedge_6 0)) \sim_8 0).$$

Результатом операції 6 є 0, отже, матимемо $((0 \to_7 0) \sim_8 0)$.

Після виконання операції 7 дістанемо $(1 \sim_8 0)$. Останній вираз має значення 0. Таким чином, значенням цієї формули на наборі (1,1,0) буде 0.

Аналогічні обчислення можна виконати й для решти семи наборів значень змінних a, b, c.

Функцію $f(p_1, p_2, ..., p_{n-1}, p_n)$ (2.2) називають функцією істинності для формули A або відповідного складеного висловлювання. Для функції істинності $f(p_1, p_2, ..., p_{n-1}, p_n)$ можна побудувати таблицю істинності (табл. 2.5):

Таблиця 2.5 – Таблиця істинності функції $f(p_1, p_2, ..., p_{n-1}, p_n)$

p_1 p_2 ... p_{n-1} p_n	$f(p_1, p_2, ..., p_{n-1}, p_n)$
0 0 ... 0 0	f (0, 0, ..., 0, 0)
0 0 ... 0 1	f (0, 0, ..., 0, 1)
0 0 ... 1 0	f (0, 0, ..., 1, 0)
0 0 ... 1 1	f (0, 0, ..., 1, 1)
.....	.....
1 1 ... 1 1	f (1, 1, ..., 1, 1)

Традиційно набори значень змінних у таблиці істинності розташовують у лексикографічному порядку.

Контрольні запитання та завдання

1. Яке відношення називається відображенням множини A у множину B?
2. За яких умов відображення на множині є взаємно однозначним?
3. Які об’єкти називають відношеннями?
4. Наведіть визначення терміна «висловлення».
5. Які висловлення називають простими?
6. За яких умов відображення на множині є сюр’єктивним?
7. Якими символами позначають істинність та хибність висловлень?

ТЕМА 3
ПОСТАНОВКА ТА РОЗВ'ЯЗАННЯ ЗАДАЧ КОМБІНАТОРИКИ

Основні питання

3.1 Комбінаторні обчислення для теоретико-множинних операцій.

3.2 Формула включення-виключення.

3.3 Сполуки, перестановки та розміщення.

3.1 Комбінаторні обчислення для теоретико-множинних операцій

Комбінаторика (комбінаторний аналіз) вивчає різноманітні властивості множин, які можна утворити як підмножини певної скінченної множини.

Задачі, в яких мова йде про підрахунок певної кількості підмножин основної скінченної множини із визначеними властивостями, називаються комбінаторними.

Кількість елементів (потужність) основної скінченної множини називають розмірністю комбінаторної задачі.

Оскільки комбінаторні задачі формулюють для скінченних множин, то для переважної більшості таких задач існує тривіальний метод їхнього розв'язання – перебір.

Пошук більш ефективних алгоритмів і методів порівняно з повним перебором для задач малої розмірності не викликає інтересу, тому що виграш виявляється незначним. Водночас комбінаторні задачі великої розмірності навіть за умови застосування найефективніших алгоритмів потребують такої кількості операцій (обсягу обчислень), що стають практично нерозв'язними.

З появою комп'ютерів з'явилась реальна можливість для розв'язання комбінаторних задач достатньо великої розмірності. Розробка й дослідження загальних принципів побудови оптимальних комбінаторних алгоритмів для розв'язування різноманітних комбінаторних задач – одні з найважливіших проблем сучасної теорії та практики програмування.

3.2 Формула включення-виключення

Якщо A та B – довільні скінченні множини, то безпосередньо із визначення теоретико-множинних операцій випливають співвідношення

$$| A \setminus B | = | A | - | A \cap B | \text{ і } | B \setminus A | = | B | - | A \cap B |.$$

Як і раніше, через |M| позначено кількість елементів скінченної множини M.

Очевидно також, що, коли множини A та B не перетинаються, тобто

$$A \cap B = \emptyset,$$

то

$$| A \cup B | = | A | + | B |.$$

Зокрема,

$$| A \Delta B | = | A \setminus B | + | B \setminus A |.$$

Для довільних множин A та B маємо

$$| A \cup B | = | A | + | B | – | A \cap B |.$$

Для трьох множин A, B, C спостерігається така рівність:

$$| A \cup B \cup C| =$$
$$= | A | + | B | + |C| – (| A \cap B | + | A \cap C| + | B \cap C|) + | A \cap B \cap C|. \quad (3.1)$$

При обчислюванні за формулою (3.1) потрібно послідовно додавати й віднімати певні кількості. Тому метод обчислення за цією формулою дістав назву **методу (принципу) включення та виключення**.

Приклад 3.1. Зі 100 студентів факультету мову програмування C++ знає 52 студенти, Python – 41, SQL – 29, C++ і Python – 25, C++ та SQL – 17, Python й SQL – 14, усі три мови знають 8 студентів.

Скільки студентів не знають жодної з трьох мов?

Позначимо через A, B, C множини студентів, які знають C++, Python та SQL відповідно.

Тоді кількість студентів, які знають принаймні одну мову, згідно з (3.1) становить:

$|A \cup B \cup C|=|A| + |B| + |C| – (|$

$\times A \cap B| + |A \cap C| + |B \cap C|) + |A \cap B \cap C|=52 + 41 +$

Укана Отже, ш кількість дорівнює 100 – 74 = 26.

Розглянемо важливу теоретико-множинну операцію – прямий добуток.

Якщо $A_1, A_2, \ldots, A_n$ – скінченні множини,

то

$$A_2 \times \ldots \times A_n| = |A_1| 29 – (25 |A_1 + 17 + 14) + 8 = 74.$$

$$\cdot |A_2| \cdot \ldots \cdot |A_n|. \quad (3.2)$$

Формулу (3.2) часто називають основним правилом комбінаторики (або правилом множення).

Нехай потрібно виконати одну за одною n дій. Якщо першу дію можна виконати k_1 способами, другу – k_2 способами і т. д., а n-ту – k_n способами, то всі n дій разом можна виконати $k_1k_2\ldots k_n$ способами.

Приклад 3.2 Кількість слів довжиною m в алфавіті A ($|A| = n$) становить $|A^m| = |A|^m = n^m$.

Цей результат випливає також із того, що побудову одного зі слів довжиною m можна розкласти на m кроків (або дій): перший крок – вибір першої літери слова, другий – вибір другої літери слова і т. д., m-й – вибір останньої літери.

Кожну з цих дій можна виконати n способами.

3.3 Сполуки, перестановки та розміщення

Позначимо через $B_k(M)$ множину всіх k-елементних підмножин певної скінченної множини M, що містить n елементів, а через C(n, k) – кількість елементів множини $B_k(M)$, де $n = |M|$ і $0 \le k \le n$.

Зокрема:

– $B_0(M)$ складається лише з одного елемента – порожньої множини ∅;

– $B_1(M)$ складається з n елементів – одноелементних підмножин множини M;

– $B_n(M)$ містить лише один елемент – саму множину M.

Отже, $C(n, 0) = C(n, n) = 1$ і $C(n, 1) = n$.

Визначення 3.1. Довільну k-елементну підмножину n-елементної множини називають сполукою (або комбінацією) k елементів з n.

Кількість усіх k-елементних підмножин множини M, яка складається з n елементів, становить

$$C(n, k) = \frac{n!}{k!(n-k)!} \qquad (3.3)$$

Для кількості сполук з n по k крім уведеного позначення C(n, k) використовують також позначення C_n^k.

Приклад 3.3. Визначимо, скількома способами можна розподілити 15 екзаменаційних білетів між 11 студентами.

Очевидно, кількість екзаменаційних білетів для цього прикладу дорівнює кількості сполук із 15 елементів (чисел) по 11, тобто

$$C(15, 1) = 12 \times 13 \times 14 \times 15 = 32\,760.$$

Визначення 3.2. Перестановкою множини M називають будь-який утворений з елементів множини M кортеж довжиною n, в якому кожен елемент із M зустрічається лише один раз.

Позначимо через P_n кількість усіх перестановок множини M.

Послідовно утворюватимемо кортежі довжиною n з елементів множини M (| M | = n). Є n можливостей для вибору першої координати кортежу. Після того, як обрано елемент для першої координати, залишиться n – 1 елемент, з яких можна вибрати другу координату кортежу і т. д. За основним правилом комбінаторики всі n дій разом можна виконати

$$n(n-1)(n-2)\ldots 2\cdot 1 = n!$$

способами.

Отже, є n! кортежів довжиною n, утворених з елементів множини M, і $P_n = n!$

Приклад 3.4. Скільки п’ятизначних чисел можна утворити із цифр 0, 1, 2, 3, 4? Кожну цифру можна використовувати в числі тільки один раз.

Існує $P_5 = 5!$ перестановок зазначених цифр. Однак частина з цих перестановок матиме на першому місці цифру 0, тобто відповідатиме чотиризначним числам.

Кількість чисел вигляду 0abcd, де (a, b, c, d) – перестановка з цифр 1, 2, 3, 4, становить $P_4 = 4!$ Отже, шуканим числом є $P_5 – P_4 = 5! – 4! = 96$.

Визначення 3.3. Кортеж довжиною k, утворений з елементів множини M (|M| = n), у якому елементи не повторюються, називають розміщенням з n по k ($0 \le k \le n$). Різні розміщення з n по k відрізняються або складом елементів, або їхнім порядком.

Кількість усіх k-елементних підмножин множини M з n елементів дорівнює C(n, k). Із кожної з цих підмножин можна утворити стільки кортежів, скільки існує різних перестановок її елементів. За попередньою теоремою їхня кількість дорівнює k!

Отже, загальна кількість кортежів довжиною k, які можна побудувати з n елементів, дорівнює k!C(n, k).

$$A(n, k) = k!C(n, k) = \frac{n!}{(n-k)!} \qquad (3.4)$$

Приклад 3.5. Визначити, скільки тризначних чисел можна утворити з цифр 0, 1, 2, 3, 4 за умови, що цифри кожного числа різні.

Існує $A(5, 3)$ кортежів довжиною 3, координати яких обрано із зазначених цифр. Виключимо із них кортежі, перша координата яких дорівнює 0. Таких кортежів – $A(4, 2)$.

Отже, шукане число дорівнює $A(5, 3) – A(4, 2) = 5 \times 4 \times 3 – 4 \times 3 = 48$.

Контрольні запитання та завдання

1. Що вивчає комбінаторика?
2. Сформулюйте метод (принцип) включення та виключення.
3. Сформулюйте основне правило комбінаторики.
4. Що таке сполука (комбінація)?
5. Що називається перестановкою множини?
6. Що називають розміщенням?
7. Сформулюйте правило обчислення факторіалу.

ТЕМА 4
ХАРАКТЕРИСТИКИ ТА СКЛАДНІСТЬ ЗАДАЧ ДИСКРЕТНОЇ ОПТИМІЗАЦІЇ

Основні питання

4.1 Загальна постановка і класифікація задач оптимізації.

4.2 Обчислювальна складність задач комбінаторної оптимізації.

4.3 Класичні дискретні задачі.

4.1 Загальна постановка і класифікація задач оптимізації

Позначимо через R^n множину всіх n-вимірних векторів із неперервними компонентами).

Визначення 4.1. Ізольована точка x множини X: $x \in X \subset R^n$ – це точка, яка належить множині ε (певному околу) в R^n:

$$x \in \varepsilon \subset R^n,$$

в яку не входять ніякі інші точки множини X: $\varepsilon \cap \{X/x\} = \varnothing$.

Визначення 4.2. Дискретна множина – множина, яка складається з ізольованих точок. Дискретна множина є або скінченною або зліченою.

Визначення 4.3. Загальна задача дискретної оптимізації – це задача знаходження точок максимумів чи мінімумів функції, яка є визначеною на дискретній множині.

Вимога дискретності змінних в явному вигляді або в прихованій формі притаманна багатьом класам задач.

Розглянемо задачу математичного програмування:

$$f(x) \to \max. \qquad (4.1)$$
$$x \in X$$

Нагадаємо.

Визначення 4.4. Задача (4.1) є задачею неперервної оптимізації, якщо множина $X \subset R^n$.

Сформулюємо таке.

Визначення 4.5. Задача (4.1) є загальною задачею дискретного програмування, якщо множина X є дискретною чи такою, що частина змінних набуває значень, які належать дискретній множині.

Визначення 4.6. Якщо $X \subset Z^n$ (Z^n – множина всіх n-вимірних векторів з цілочисловими компонентами), то ця задача є задачею цілочислового програмування.

Визначення 4.7. Нехай $X \subset Z^k \times R^m$, $k + m = n$. Тоді ця задача називається частково цілочисловою (змішаною) задачею.

Визначення 4.8. У тому випадку, якщо $X \subset B^n$ (B^n – множина всіх n – вимірних векторів з компонентами, які приймають тільки два значення 0 і 1), то задача називається задачею булевого програмування.

Існує множина класів практичних задач, які на перший погляд не мають нічого спільного чи зовсім слабо пов’язані з оптимізацією дискретної моделі, але які тим не менш, можна сформулювати як задачі дискретного програмування.

Дійсно, вимога дискретності змінних, якщо в неявному вигляді, то в прихованій формі, притаманна багатьом практично важливим класам задач, які забезпечують широку сферу застосування дискретного програмування в численних теоретичних и прикладних дисциплінах.

Сфери застосування дискретного програмування:

- складання послідовності виробничих процесів;
- календарне планування роботи підприємства;
- планування і забезпечення матеріально-технічного забезпечення;
- розміщення підприємств;
- складання кошторису капіталовкладень;
- планування використання обладнання.

Найбільш вивченим класом задач дискретної оптимізації є задачі цілочислового лінійного програмування.

4.2 Обчислювальна складність задач комбінаторної оптимізації

Розрізняють два класи проблем: P та NP. Ці класи можна визначити більш точно за допомогою будь-якого формального визначення алгоритмів, такого як машина Тюрінга. Не заглиблюючись в більш формальний опис, для подальшого буде нам достатньо визначити їх так.

Клас P складають задачі, для розв’язання яких відомі алгоритми з поліноміальною складністю (polynomial-time algorithms). До поліноміальних відносять алгоритми, складність (трудомісткість) яких обмежена поліномом від розміру входу оптимізаційної задачі.

Нагадаємо, многочленом, багаточленом або поліномом однієї змінної в математиці називається вираз вигляду

$$c_0 + c_1x + \ldots + c_nx^n,$$

де c_i є сталими коефіцієнтами (константами), а x – змінна.

Отже, це клас відносно простих задач, для яких існують ефективні алгоритми.

Наприклад, стандартний алгоритм множення матриць вимагає n^3 операцій множення, де n – ранг матриці.

До класу NP (nondeterministic polynomial), який здається більш широким, відносять задачі, що можуть бути розв'язаними недетермінованим поліноміальним алгоритмом.

Такі алгоритми мають дві фази: на першій знаходиться припустимий варіант розв'язку, а на другій цей варіант перевіряється детермінованим поліноміальним алгоритмом верифікації.

Визначення 4.9. Задача комбінаторної оптимізації π у варіанті розпізнавання є NP-повною (NP-complete), якщо виконуються дві умови:

1) $\pi \in$ NP;

2) всі задачі із NP можуть бути зведеними до π за допомогою поліноміального алгоритму.

Зауважимо, що на практиці для верифікації другої умови зазвичай лише показують, що деяку відому NP-повну задачу можна поліноміально перетворити в досліджувану задачу.

Інколи можна показати, що всі задачі із NP поліноміально зводяться до деякої задачі π, але не вдається довести, що $\pi \in$ NP. Такі задачі відносять до числа NP-складних.

Визначення 4.10. Задача відноситься до класу NP-складних (NP-hard), якщо до неї поліноміально зводяться всі задачі із класу NP.

Для потреб практики важливо, щоб трудомісткість оптимального алгоритму була обмежена поліномом від вхідних даних задачі. На сьогодні предметом дискусій є питання, чи P = NP?

У разі негативної відповіді теоретично не існує поліноміального алгоритму, тому при розв'язанні таких задач відразу варто зосередитися на розробці потужних наближених алгоритмів, які хоча б і не гарантують отримання оптимального розв'язку, здатні знаходити оптимальний чи близький до нього розв'язок за прийнятний час.

Ця проблема Математичним інститутом Клея з Кембриджа у 2000 р. визначена як одна із семи проблем тисячоліття.

4.3 Класичні дискретні задачі

З фізичних умов багатьох практичних задач випливає вимога цілочисельності, тобто дробове значення змінних може не мати сенсу (тобто виявитися нереалізованим).

Задача про рюкзак

Мандрівник, збираючись в похід, хоче укласти в свій рюкзак деяку кількість предметів: 1, 2, ... , n. Відома вага p_j і об'єм v_j кожного предмета j.

Загальна вага рюкзака не повинна перевищувати величину P, а об'єм не повинен перевищувати величину V.

Мандрівник, спираючись на суб'єктивні оцінки корисності предметів, кожному предмету j приписує коефіцієнт корисності (цінності) c_j.

Які предмети потрібно вибрати мандрівнику, щоб сумарна цінність рюкзака була максимальною?

Позначимо через x_1, x_2, x_n – кількість взятих предметів.

Функція мети (цільова функція) має вигляд:

$$z = \sum_{j=1}^{n} c_j x_j.$$

Тоді математична модель задачі про рюкзак така:

$$z \to \max,$$

$$\sum_{j=1}^{n} p_j x_j \le P,$$

$$\sum_{j=1}^{n} v_j x_j \le V,$$

$$x_j \ge 0,\ x_j \text{ – ціле, } j=1,2,\ldots, n.$$

Найбільш поширені окремі випадки цієї задачі, коли $x_j = 0$ або 1, тобто присутнім є лише одне із функціональних обмежень.

Задача про призначення

Нехай є n виконавців робіт R_i, i =1,…, n та n робочих місць W_j, j=1,…, n. Для кожного призначення (R_i,W_j) відома вартість виконання відповідного виду робіт $c_{ij} \geq 0$, i,j=1,…, n (деякі вартості можуть бути нескінченно великими, якщо відповідне призначення неможливо).

Потрібно так розподілити n виконавців по n робочих місць, щоб кожен виконавець був завантажений одним видом робіт, кожна робота виконувалась тільки одним виконавцем, а сумарні витрати на виконання всіх видів робіт були мінімальними.

Побудуємо відповідну математичну модель.

Визначимо незалежні (ендогенні) змінні:

$$x_{ij} = \begin{cases} 1, \text{ якщо виконавець } R_i \text{ призначений на робоче місце } W_j, \\ 0, \text{ в іншому випадку.} \end{cases}$$

Функція мети має вигляд:

$$z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

Отже, математична модель задачі про призначення:

$$z \to \min,$$

$$\sum_{i=1}^{n} x_{ij} = 1,$$

$$\sum_{j=1}^{n} x_{ij} = 1,$$

x_{ij}=1 або 0, i,j=1,2,…n.

Задача про експертів

Нехай є n тематичних напрямів (j = 1,2, …, n), за якими необхідно дати кваліфіковані висновки.

Для цього можна запросити m експертів. Відомо, за якими питаннями і-й експерт може одночасно дати кваліфікований висновок – множина $B_i = \{b_{i1}, b_{i2},\ldots, b_{ij},\ldots b_{in}\}$ потужності n, $b_{ij} = \{0,1\}$, де $b_i = 0$ – експерт не може дати відповідь, $b_i = 1$ – експерт може надати відповідь.

Відомий також розмір c_i. i=1,..., m, оплати послуг експертів.

Необхідно знайти такий набір експертів, при якому за кожним направленням буде надано висновок, і при цьому витрати на послуги будуть мінімальними.

Побудуємо відповідну математичну модель.

Незалежні змінні:

$$x_i = \begin{cases} 1, \text{ якщо запрошуємо i-го експерта,} \\ 0, \text{ якщо не запрошуємо i-го експерта, } i = 1,..., m. \end{cases}$$

Цільова функція:

$$z = \sum_{i=1}^{m} c_i x_i.$$

Обмеження задачі: за кожним напрямом повинен бути запрошений хоча б один експерт:

$$\sum_{i=1}^{m} b_{ij} x_i \geq 1, j = 1,2, \dots, n.$$

Остаточна математична модель

$$\sum_{i=1}^{m} c_i x_i \rightarrow \min,$$
$$\sum_{i=1}^{m} b_{ij} x_i \geq 1, j = 1,2, \dots, n.$$

Задачі з дихотомією

Дихотомія (від грецького διχοτομια) – поділ на дві частини.

У деяких задачах допустиме рішення має відповідати одній із двох альтернативних умов:

$$f(x) \rightarrow \max;$$
$$G(x) \leq 0,$$
$$g_1(x) \leq 0 \text{ або } g_2(x) \leq 0.$$

Зведемо задачу до задачі дискретної оптимізації. Для цього введемо допоміжну змінну y:

$$y = \begin{cases} 1, \text{ якщо вірна перша умова } g_1(x) \le 0, \\ 0, \text{ якщо вірна друга умова } g_2(x) \le 0. \end{cases}$$

Також введемо до розгляду достатньо велике число Z (становить верхню межу значень функцій $g_i(x)$, i = 1,2, на всій допустимій множині рішень). Тоді обмеження виду «або-або» еквівалентно такій системі нерівностей:

$$\begin{cases} g_1(x) \le Zy, \\ g_2(x) \le Z(1- y), \\ y \in \{0, 1\}. \end{cases}$$

Права частина однієї з нерівностей буде дорівнювати Z і таким чином відповідна нерівність буде надлишковою. Задача запишеться у такий спосіб:

$$f(x) \to \max;$$
$$G(x) \le 0 ;$$
$$g_1(x) \le Zy,$$
$$g_2(x) \le Z(1- y),$$
$$y \in \{0, 1\}.$$

Контрольні запитання та завдання

1. Що таке дискретна множина?
2. Що таке загальна задача дискретної оптимізації?
3. Яка задача називається частково цілочисловою (змішаною) задачею?
4. Які задачі складають клас P щодо алгоритмічної складності?
5. Які задачі належать до класу NP?
6. Яка проблема Математичним інститутом Клея визначена як одна із семи проблем тисячоліття?
7. Сформулюйте задачу про рюкзак.
8. У чому полягає задача про призначення?
9. Надайте математичну модель задачі про експертів.
10. Сформулюйте задачу з дихотомією.

ТЕМА 5
ТЕОРІЯ ГРАФІВ

Основні питання

5.1 Основні поняття.

5.2 Ізоморфізм графів.

5.3 Представлення графів матрицями інцидентності та суміжності.

5.4 Паросполучення.

5.1 Основні поняття

Визначення 5.1. Графом G називається множина V із заданим на ній бінарним відношенням $E \subset V^2$, тобто

$$G = < V, E >.$$

Граф можна визначити інакше, спираючись на елементарні геометричні поняття точки та лінії. При цьому геометричні характеристики цих ліній до уваги не приймаються.

Визначення 5.2. Граф G – це сукупність двох множин V = {v} та E = {e}, де V – множина точок або вершин, E – множина ліній або ребер, що з'єднують точки, на яких визначено відношення інцидентності $\Im$, тобто кожний елемент $e \in E$ інцидентний двом елементам $\{v', v''\} \in V$.

Відношення інцидентності дозволяє розглядати граф як сім'ю пар

$$(v', v''), v', v'' \in V,$$

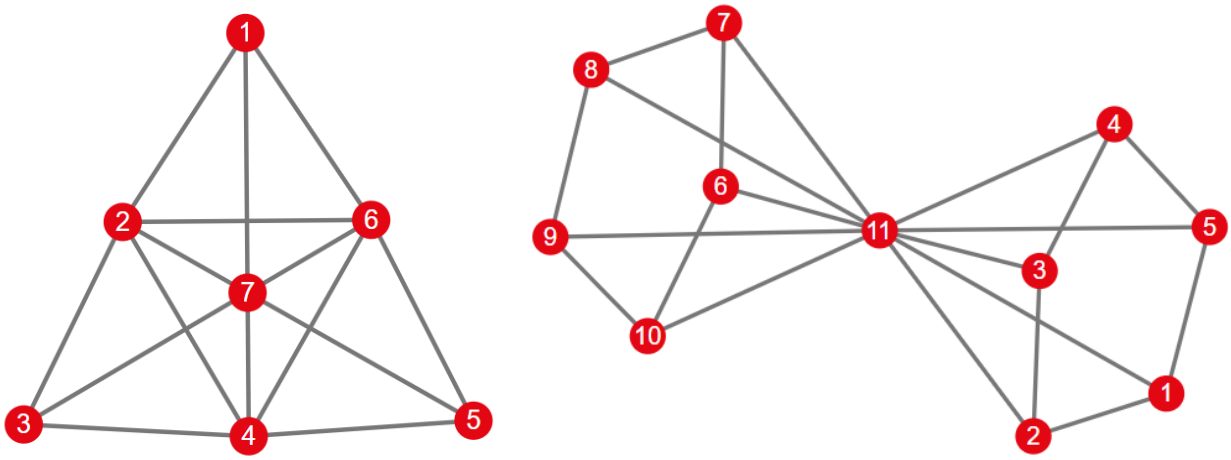
що визначає, які вершини з'єднані лініями (ребрами).

Граф G зображується на площині у вигляді геометричної фігури, що має вершини, з'єднані відрізками прямих або кривих ліній.

Рисунок 5.1, а представляє граф G, що має 7 вершин та 13 ребер, на рисунку 5.1, б подано граф з 11 вершинами та 20 ребрами.

Ці та деякі інші зображення графових структур згенеровані за допомогою програми Graph Driver, розробленої українським розробником програмного забезпечення Іваном Куцкіром.

Якщо порядок елементів у парі (v', v") є важливим, то граф називається орієнтованим, в іншому випадку – неорієнтованим (на рис. 5.2 подано орієнтований граф).



а) б)

Рисунок 5.1 – Приклади неорієнтованих графів:

а – граф G = < V, E >, |V| = 7, |E| = 13; б – граф G = < V, E >, |V| = 11, |E| = 20

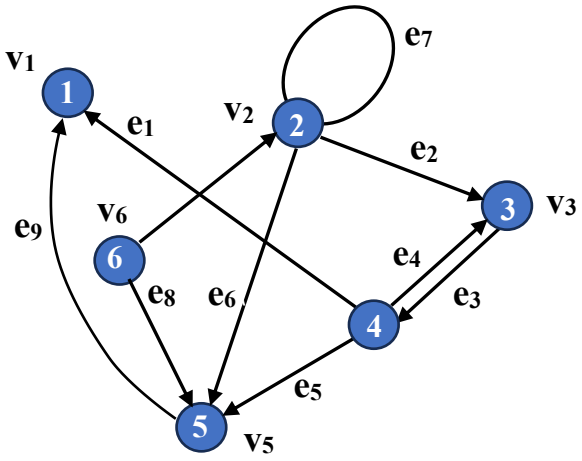


Рисунок 5.2 – Приклад орієнтованого графа

Ребра орієнтованого графа G зображуються стрілками, які вказують на порядок переліку вершин. Часто орієнтовані ребра називають дугами.

Визначення 5.3. Граф G' називається підграфом G, якщо: V (G′) ⊂ V(G) та E(G') ⊂ E(G).

Визначення 5.4. Ребро e_k графа G називається інцидентним до вершини v_i, якщо v_i є одним з кінців e_k.

Визначення 5.5. Два ребра графа G називаються суміжними, якщо вони інцидентні одній вершині; дві вершини – суміжні, якщо вони інцидентні одному ребру.

Визначення 5.6. Граф називається простим, якщо його суміжні вершини з'єднані не більше, як одним ребром.

Інакше граф називається загальним (мультиграфом).

У загальному графі допускаються:

– кратні ребра, тобто кілька ребер може з'єднувати одну й ту саму пару вершин (ребра e4, e3 на рис. 5.2) та петлі, тобто ребро замкнене на одну вершину (e7 на рис. 5.2).

Визначення 5.7. Кількість ребер, інцидентних цій вершині називається її ступенем.

При визначенні ступеня вершини ребро, що утворює петлю, враховується двічі.

Вершина ступеня 0 називається ізольованою, а вершина ступеня 1 – підвішеною або кінцевою.

Наявна лема Ейлера про «потискання рук».

Лема 5.1. Сума степенів усіх вершин графа – парна, це парне число дорівнює подвоєній кількості ребер.

Визначення 5.8. Граф називається зваженим, якщо кожній вершині $v_i \in V$ та кожному ребру $e_k \in E$ ставиться у відповідність елемент деякої множини, яка називається вагою w_i вершини v_i або вагою p_i ребра e_k:

$$v_i \in V \rightarrow w_i \in W = \{ w_1, w_2, \ldots, w_n\},$$
$$e_k \in E \rightarrow p_i \in P = \{ p_1, p_2, \ldots, p_n\}.$$

Визначення 5.9. Простий граф називається повним (або клікою), якщо дві довільні вершини графа є суміжними.

Повний граф з n вершинами позначається K_n. На рисунку 5.2 зображено граф K_4.

Визначення 5.10. Дводолевим графом (також біграфом, двочастковим графом) називається граф, множина вершин якого може бути розбита на дві підмножини так, що кожне ребро графа має одну вершину з першої підмножини і одну з другої підмножини

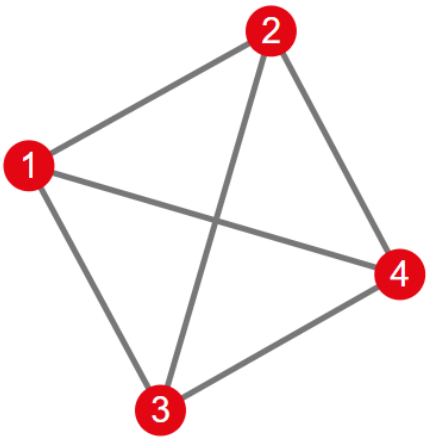


Рисунок 5.2 – Кліка K_4

На рисунку 5.3 подано приклади дводолевих графів, підмножини вершин виділені кольором.

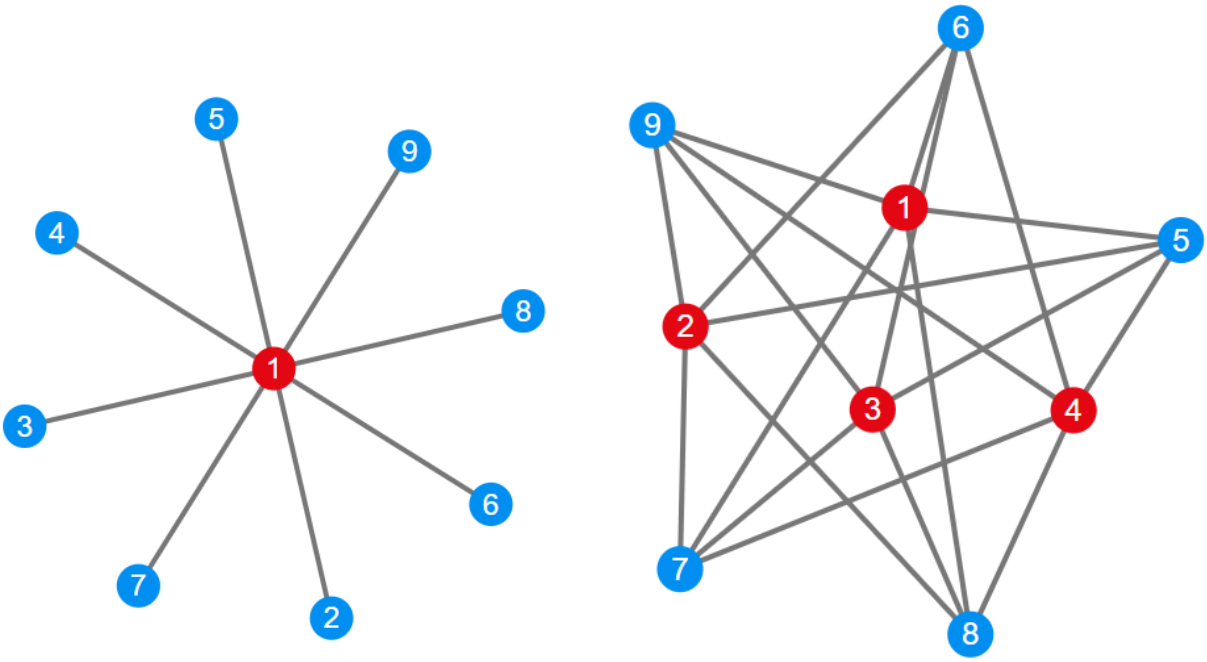


Рисунок 5.3 – Приклади дводолевих графів

Дводолеві графи часто використовуються у практичних застосуваннях. Наприклад, задачі призначення, розподілу, плани перевезень описуються дводолевими графами (виконавці та роботи, джерела ресурсів та споживачі, склади та магазини).

5.2 Ізоморфізм графів

Один і той же граф може мати різні зображення, тому важливу роль відіграє поняття ізоморфних графів, яке дає можливість різні зображення одного графа вважати представниками одного класу і не розрізняти їх під час досліджень.

Визначення 5.11. Два графи G = < V, E > та G' = < V', E' > називають ізоморфними, якщо існує взаємно однозначна відповідність між множинами V та V' вершин така, що вершини одного графа сполучені тоді і тільки тоді, коли сполучені вершини другого графа. Якщо графи орієнтовані, то напрямок дуг теж повинен збігатися.

На рисунку 5.4 зображено два графи, які є ізоморфними за відповідності

$$1 \leftrightarrow s,\ 5 \leftrightarrow m,\ 2 \leftrightarrow p,\ 6 \leftrightarrow q,\ 4 \leftrightarrow r,\ 3 \leftrightarrow n$$

Ці графи можна вважати ізоморфними, тобто одним графом, поданим різними зображеннями.

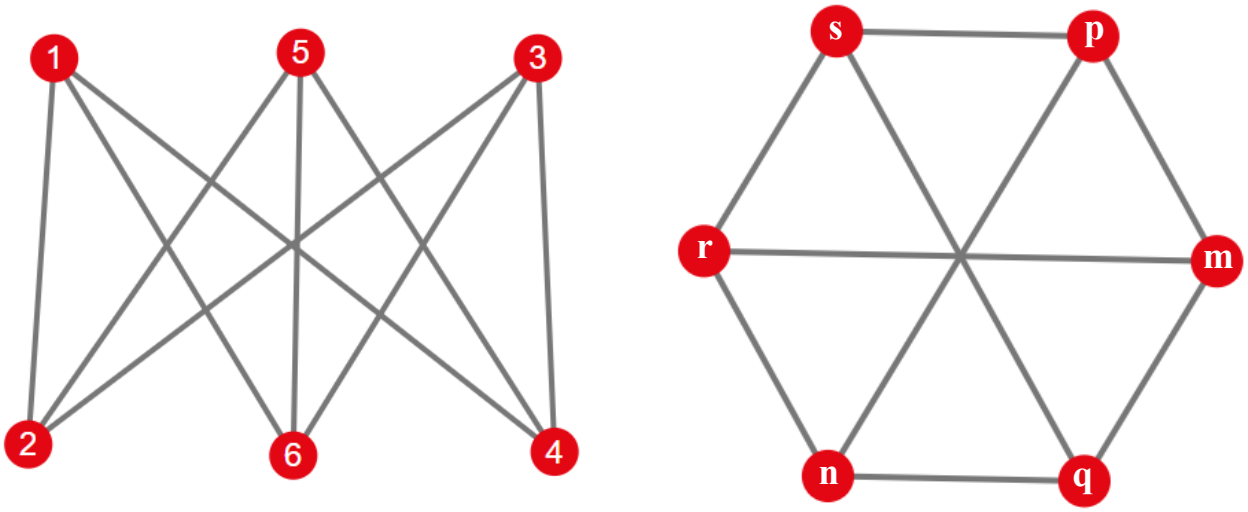
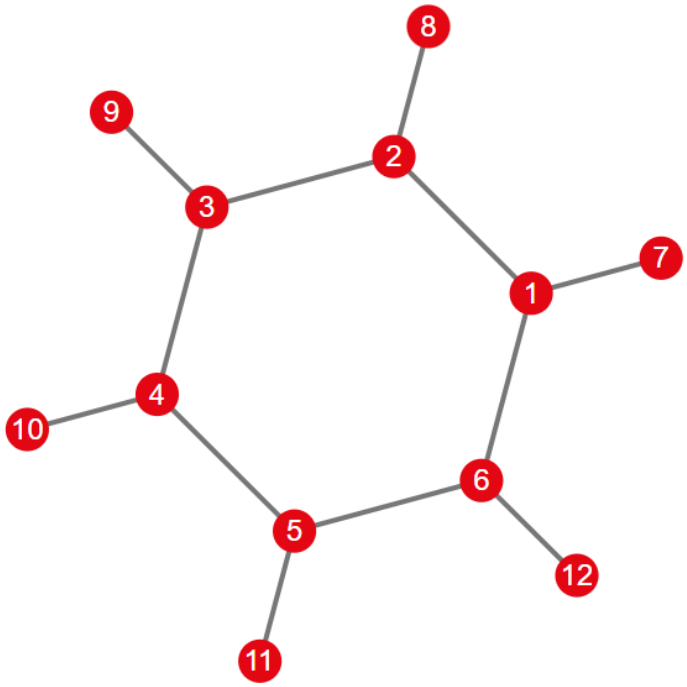


Рисунок 5.4 – Приклад ізоморфних графів

5.3 Представлення графів матрицями інцидентності та суміжності

Загалом граф зручно подавати у вигляді списку ребер – масиву натуральних чисел, розміром m × 2, де в кожному рядку записуються номери вершин, що з’єднуються відповідним ребром (рис. 5.5).



Ребро	1-ша вершина	2-га вершина
e_1	1	2
e_2	2	3
e_3	3	4
e_4	4	5
e_5	5	6
e_6	6	1
e_7	1	7
e_8	2	8
e_9	3	9
e_{10}	4	10
e_{11}	5	11
e_{12}	6	12

Рисунок 5.5 – Подання графа подавати у вигляді списку ребер

Визначення 5.13. Матрицею інцидентності графа G є матриця (n × m), де n – кількість вершин, m – кількість ребер. Це одна з форм подання графа, у якій вказуються зв'язки між інцидентними елементами графа ребро (дуга) і вершина.

Стовпці матриці інцидентності відповідають ребрам, рядки – вершинам. Ненульове значення в клітинці матриці вказує на зв'язок між вершиною і ребром (їхня інцидентність).

Кожна комірка матриці інцидентності неорієнтованого графа може набувати трьох значень:

– 1: якщо ребро e_k інцидентне вершині v_i;
– 2: якщо ребро e_k є петлею (починається та закінчується) у вершині v_i;
– 0: якщо вершина v_i не має стосунку до ребра e_j.

Кожна комірка матриці інцидентності орієнтованого графа може набувати чотирьох значень:

– 1: якщо ребро e_k виходить з вершини v_i;
– 1: якщо ребро e_k входить у вершину v_i;
– 0: якщо вершина v_i не має стосунку до ребра e_k;
– 2: якщо ребро e_k є петлею (починається та закінчується) у вершині v_i.

Приклад 5.1 матриці інцидентності (табл. 5.1) для неорієнтованого графа, зображеного на рисунку 5.5.

Таблиця 5.1 – Матриця інцидентності

Вершина	Ребро											
	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}
1	1	0	0	0	0	1	1	0	0	0	0	0
2	1	1	0	0	0	0	0	1	0	0	0	0
3	0	1	1	0	0	0	0	0	1	0	0	0
4	0	0	1	1	0	0	0	0	0	1	0	0
5	0	0	0	1	1	0	0	0	0	0	1	0
6	0	0	0	0	1	1	0	0	0	0	0	1
7	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	1

Визначення 5.14. Матрицею суміжності графа G зі скінченною кількістю вершин n є квадратна матриця A розміру n, у якій значення елементу a_{ij} дорівнює числу ребер з i-ї вершини графа в j-ту вершину.

Матриця суміжності є симетричною відносно головної діагоналі.

Іноді, особливо у разі неорієнтованого графа, петля (ребро з деякої i-ї вершини в саму себе) вважається за два ребра, тобто значення діагонального елементу a_{ii} у цьому випадку дорівнює подвоєному числу петель навколо i-ї вершини.

Матриця суміжності простого графа (що не містить петель і кратних ребер) є бінарною матрицею і містить нулі на головній діагоналі.

Приклад 5.2 матриці інцидентності (табл. 5.2) для неорієнтованого графа, зображеного на рисунку 5.5.

5.4 Паросполучення

Визначення 5.15. Реберним пакуванням у графі G = (V, E) називається підмножина E_1 попарно несуміжних ребер $E_1 \subseteq E$.

Таблиця 5.2 – Матриця суміжності

Вершина	Вершина											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	1	1	0	0	0	0	0
2	1	0	1	0	0	0	0	1	0	0	0	0
3	0	1	0	1	0	0	0	0	1	0	0	0
4	0	0	1	0	1	0	0	0	0	1	0	0
5	0	0	0	1	0	1	0	0	0	0	1	0
6	1	0	0	0	1	0	0	0	0	0	0	1
7	1	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	0	0	0	0	0	0
12	0	0	0	0	0	1	0	0	0	0	0	0

Визначення 5.16. Максимальна кількість попарно несуміжних ребер графа G = (V, E) називається реберним числом незалежності графа $\varepsilon_1(G)$.

Якщо ребро інцидентне вершині, то кажуть, що вони покривають одне одного.

Множина вершин, що покривають усі ребра графа, називається вершинним покриттям графа.

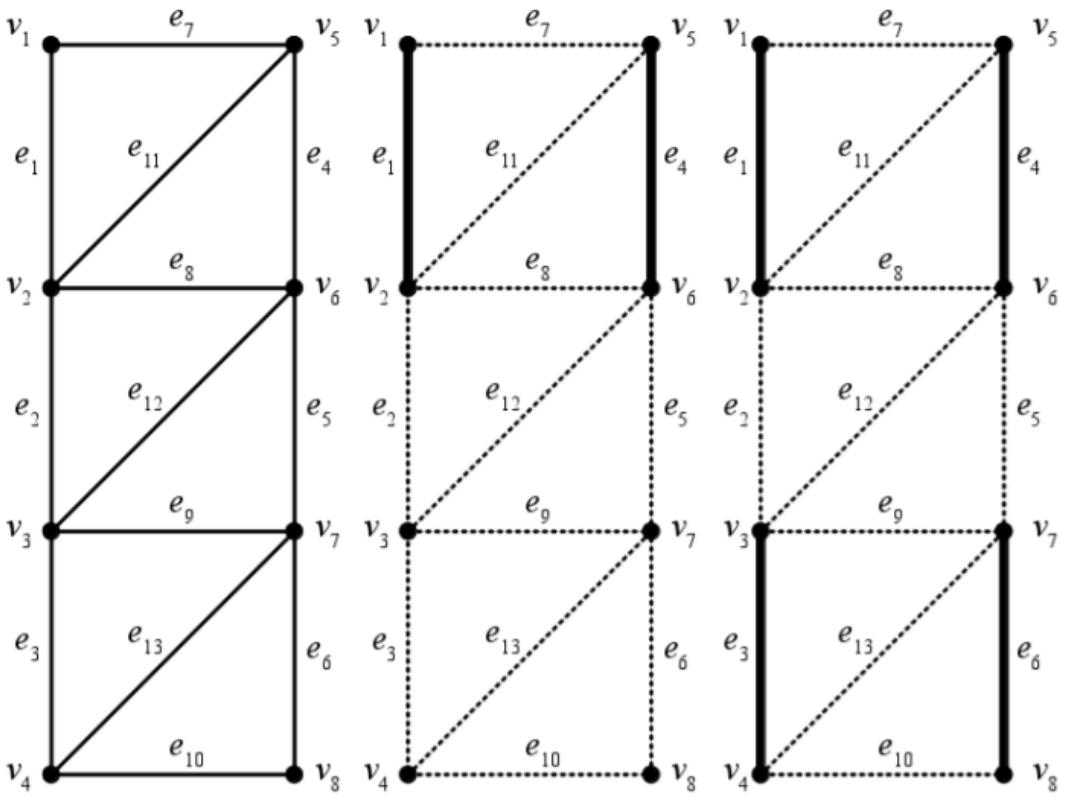
Множина ребер, які покривають усі вершини графа, називається реберним покриттям графа.

Визначення 5.17. Множина ребер графа G, у якій жодна пара ребер не є суміжною, називається паросполученням (рис. 5.6) в графі G.

Паросполучення, у якому кількість ребер дорівнює $\varepsilon_1(G)$, називається максимальним.

Максимальне паросполучення не обов’язково покриває всі вершини графа і не обов’язково має найбільшу можливу кількість ребер серед усіх паросполучень.

Визначення 5.18. Досконалим паросполученням з V_1 у V_2 в дводольному графі $G = < V, E >$, $V = V_1 \cup V_2$, $V_1 \cap V_2 = \varnothing$, називається бієктивне відображення між вершинами підмножин V_1 та V_2, при якому кожна вершина з V_1 з’єднується ребром з деякою вершиною з V_2.



а) б) в)

Рисунок 5.6 – Графічна ілюстрація парасполучень графа:
а – граф G; б – довільне паросполучення графа G;
в – максимальне паросполучення графа G

Досконале паросполучення в графі з n вершинами (де n парне) завжди містить n/2 ребер. Максимальне паросполучення може мати меншу кількість ребер.

Задача пошуку максимального досконалого паросполучення безпосередньо зводиться до відомого типу оптимізаційних задач – транспортної задачі.

Контрольні запитання та завдання

1. Дайте визначення поняттю «граф».
2. Який граф називається орієнтованим?
3. Який граф називається зваженим?
4. Що таке кліка?
5. Що таке ізоморфізм графів? Наведіть приклад.

6. Що таке матриця інцидентності?

7. Скільки одиниць буде в кожному стовпчику в матриці інцидентності простого графа?

8. Що таке матриця суміжності? Наведіть приклад.

9. Дайте визначення поняттю дводолевий граф.

10. Що таке паросполучення?

11. Яке паросполучення називається максимальним?

12. Яке паросполучення називається досконалим?

13. Поясніть відмінність між досконалим та максимальним паросполученнями.

ТЕМА 6
ЗАДАЧІ ДИСКРЕТНОЇ ОПТИМІЗАЦІЇ НА ГРАФАХ

Основні питання

6.1 Постановка задачі комівояжера.

6.2 Формальне визначення задачі комівояжера.

6.3 Алгоритмічна складність задачі комівояжера.

6.1 Постановка задачі комівояжера

Задача комівояжера (англ. *Travelling salesman problem*, TSP) – це одна з найбільш відомих задач дискретної оптимізації, що полягає в знаходженні самого вигідного (оптимального) маршруту, що проходить через зазначені міста хоча б по одному разу з наступним поверненням у початкове місто.

В умовах задачі вказуються критерій оптимальності маршруту (найкоротший, найдешевший, сукупний критерій і т. п.) і відповідні матриці відстаней, вартості і т. п.

Вказується зазвичай, що маршрут повинен проходити через кожне місто тільки один раз – у такому випадку вибір здійснюється серед гамільтонівих циклів.

Гамільтонів цикл – цикл у графі, що містить всі вершини графа рівно по одному разу, тобто це простий цикл, що містить всі вершини графа.

Зі свого боку, цикл – це замкнений шлях у графі (замкнений ланцюг).

Існує кілька окремих випадків загальної постановки задачі, зокрема:

– геометрична задача комівояжера (так звана планарна або евклідова, коли матриця відстаней відображає відстані між точками на площині);

– трикутна задача комівояжера (коли на матриці вартостей виконується нерівність трикутника);

– симетрична й асиметрична задача комівояжера.

Задача комівояжера належить до класу NP-повних задач.

Багато сучасних методів дискретної оптимізації, такі як метод Гоморі, метод гілок та меж і різні варіанти евристичних алгоритмів, були розроблені на прикладі задачі комівояжера.

Важливий внесок у дослідження задачі належить Джорджу Данцигу, Делберту Рею Фалкерсону і Селмеру Джонсону.

Герхард Райнельт створив базу даних TSPLIB – набір стандартизованих примірників задачі комівояжера різного ступеня складності для порівняння результатів роботи різних груп дослідників.

У березні 2005 року задача з 33 810 вузлами була вирішена програмою Конкорд був обчислений шлях довжиною в 6 6048 945 од. і доведено відсутність коротких шляхів.

У квітні 2006 року був знайдений розв'язок для екземпляра графа з 85 900 вершинами.

Використовуючи методи декомпозиції, можна обчислити рішення для випадків завдання з мільйонами вузлів, довжина яких менше ніж на 1 % більше оптимальної.

6.2 Формальне визначення задачі комівояжера

Задачу комівояжера можна подати у вигляді графової моделі G = < V, E >, використовуючи вершини графа і ребра між ними.

На рисунку 6.1 подана графова модель задачі комівояжера, що включає чотири міста. Таким чином, вершини графа (на рис. 6.1 позначені цифрами {**1**, **2**, **3**, **4**}) відповідають містам, а ребра (i, j) між вершинами i і j – шляхи сполучення між цими містами.

Кожному ребру (i, j) можна зіставити вагу $c_{ij} \geq 0$ (на рис. 6.1 це значення 20, 42, ...), яку можна розуміти як, наприклад, відстань між містами, час або вартість поїздки.

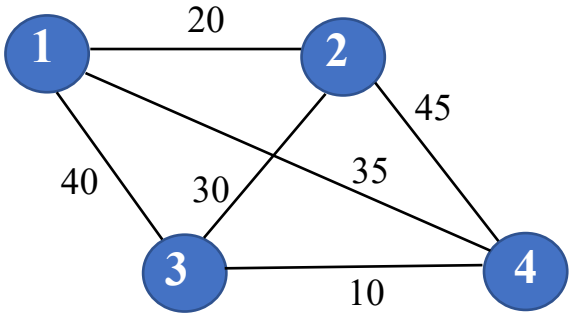


Рисунок 6.1 – Симетрична задача комівояжера для чотирьох міст

З метою спрощення завдання і гарантії існування маршруту, звичайно вважається, що модельний граф задачі є повністю зв'язним, тобто, що між кожною довільною парою вершин існує ребро.

У тих випадках, коли між окремими містами не існує шляху, цей факт можна формалізувати шляхом введення ребер з максимальною довжиною. Через велику довжину таке ребро ніколи не потрапить до оптимального маршруту, якщо він існує.

Залежно від того, який критерій оптимальності маршруту зіставляється з вагою ребер, розрізняють різні варіанти задачі, найважливішими з яких є симетрична, асиметрична і метрична задачі.

У разі симетричної задачі всі пари ребер між одними і тими ж вершинами мають однакову довжину, тобто для ребра (i, j) довжини $c_{ij} = c_{ji}$. є однаковими. Симетрична задача моделюється неорієнтованим графом.

Асиметрична задача комівояжера відрізняється тим, що вона моделюється орієнтованим графом. При цьому можливий випадок, що для ребра (i, j) довжини $c_{ij} \neq c_{ji}$. Таким чином, варто також враховувати, у якому напрямку розглядаються ребра.

У симетричному випадку кількість можливих маршрутів удвічі менше асиметричного випадку.

На практиці задача комівояжера у випадку реальних міст може бути як симетричною, так і асиметричною залежно від тривалості або довжини маршрутів і від напрямку руху.

Визначення 6.1 Симетричну задачу комівояжера називають метричною, якщо стосовно довжин ребер виконується **нерівність трикутника**. Умовно кажучи, в таких завданнях обхідні шляхи довше прямих, тобто, ребро від вершини i до вершини j ніколи не буває довшим шляху через проміжну вершину k:

$$c_{ij} \leq c_{ik} + c_{kj}.$$

Така властивість довжини ребер визначає міру відстані, що задовольняє інтуїтивне розуміння відстані.

Поширені на практиці функції відстані є також метриками і задовольняють нерівності трикутника:

– *Евклідова відстань* в евклідовій задачі комівояжера;

– *Мангетенська метрика* (також квартальна метрика) прямокутної задачі комівояжера, у якій відстань між вершинами на решітці дорівнює сумі відстаней по осі ординат і абсцис;

– *Максимальна метрика,* яка визначає відстань між вершинами ґратчастого графа як максимальне значення відстані уздовж осі ординат і абсцис.

Мангетенська метрика відповідає випадку, коли пересування в обох напрямках відбувається послідовно, а максимальна – випадку, коли пересування в обох напрямках відбувається синхронно, а загальний час дорівнює максимальному часу пересування вздовж осі ординат або абсцис.

Не-метрична задача комівояжера може виникати, наприклад, у випадку мінімізації тривалості перебування за наявності вибору транспортних засобів у різних напрямках. У такому випадку обхідний шлях літаком може бути коротше прямого сполучення автомобілем.

Якщо на практиці в умовах задачі дозволяється відвідувати міста кілька разів, то симетричну задачу можна звести до метричної. Для цього задачу розглядають на так званому графі відстаней. Цей граф має таку ж множину вершин, як і вихідний, і є повністю зв'язним.

Довжина ребер c_{ij} між вершинами i та j на графі відстаней відповідає довжині найкоротшої відстані між вершинами i та j у вихідному графі.

Таким чином, для довжин c_{ij} виконується нерівність трикутника, і кожному маршруту на графі відстаней завжди відповідає маршрут із можливими повтореннями вершин у вихідному графі.

Сформулюємо симетричну задачу комівояжера у вигляді задачі дискретної оптимізації.

Скористаємося поданням симетричної задачі комівояжера у вигляді множини ребер V графа G = < V, E >. Припустимо, кількість ребер графа G дорівнює N.

Кожному ребру $\{ j_i , j_k \} \in V$ ставиться у відповідність бінарна змінна $x_{j_i j_k} \in \{0, 1\}$, що дорівнює 1, якщо ребро належить маршруту, і 0 – у протилежному випадку.

Тоді функція мети в задачі комівояжера подається у вигляді

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{j_i j_k} x_{j_i j_k}.$$

Довільний маршрут можна представити у вигляді значень вектора змінних $\{x_{j_1 j_2}, x_{j_2 j_3}, \dots x_{j_N j_1}\}$ приналежності відповідних ребер маршруту, але не кожен такий вектор визначає маршрут.

Для вершин маршруту має виконуватися умова кратності: кожна вершина повинна мати одне вхідне і вихідне одне ребро маршруту, тобто кожна вершина має сполучатися через пару ребер із рештою вершин.

$$\forall i \in V, \sum_{j \in V\{i\}} x_{ij} = 2$$

Таким чином, в умові кратності отримана сума дорівнює кількості ребер у маршруті, що мають вершину і на одному з кінців. Ця сума дорівнює 2, тому що кожна вершина має вхідне і вихідне ребро.

На рисунку 6.2 вершина 1 сполучається з вершинами 3 та 4. Ребра маршруту позначені товстими лініями та поряд із ребрами вказані значення x_{ij}, що додаються до умови кратності.

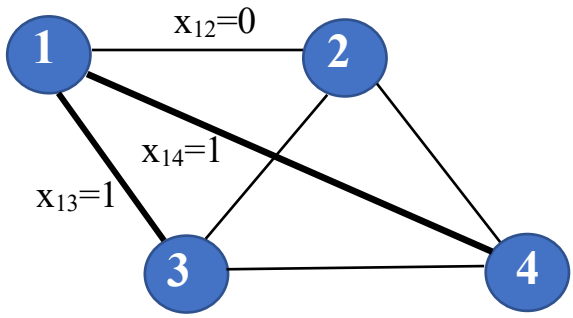


Рисунок 6.2 – Ілюстрація умови кратності

Однак у загальному випадку в побудованому маршруті можуть бути цикли, тобто змінні x_{ij} можуть задовольняти умові кратності, але не визначати маршрут (рис. 6.3). І знов на рисунку 6.3, а ребра маршруту позначені товстими лініями. Поряд із ребрами вказані значення x_{ij}.

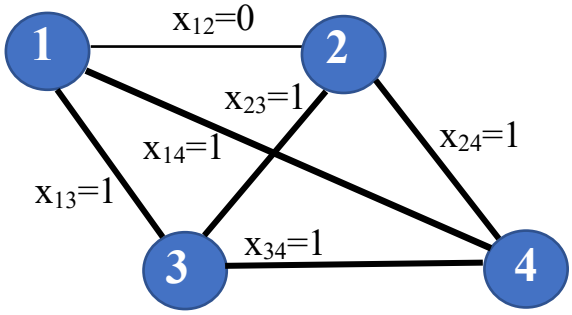


Рисунок 6.3 – Маршрут на графі показано товстими лініями. Побудований маршрут має один цикл

Щоб уникнути подібних випадків, мають виконуватися так звані нерівності циклів (або умови усунення підмаршрутов), які були визначені Данцигом, Фалкерсоном і Джонсоном під назвою *умови петель* (*loop conditions*).

Цими нерівностями визначається додаткова умова того, що кожна множина вершин $S \subset V$ є або порожньою, або містить всі вершини, що зв'язані з іншими вершинами через мінімум два ребра:

$$\sum_{i \in S,\, j \notin S} x_{ij} \geq 2$$

для всіх множин вершин S, де $1 \leq |S| \leq |V| - 1$.

Виключення підциклів забезпечується поліноміальною кількістю обмежень

$$u_i - u_j + nx_{ij} \leq n - 1,\ u_i, u_j \geq 0,\ i, j = 2,\ldots,N,$$

де змінні $u_i \geq 0$ – номер кроку, на якому відвідали місто i.

6.3 Алгоритмічна складність задачі комівояжера

Оскільки комівояжер в кожному з міст постає перед вибором наступного міста з тих, що він ще не відвідав, існує $(n - 1)!$ маршрутів для асиметричної та $(n - 1)! / 2$ маршрутів для симетричної задачі комівояжера.

Таким чином, вимір простору пошуку залежить експоненціально від кількості міст.

Задача комівояжера є NP-важкою проблемою, що означає, що немає відомого поліноміального алгоритму для її вирішення за всіх можливих вхідних даних. Це означає, що немає ефективного способу знайти оптимальний шлях для комівояжера в масштабах всієї задачі, коли кількість міст (точок) стає дуже великою.

Різні варіанти завдання комівояжера (метрична, симетрична і асиметрична) NP-еквівалентні. Згідно з поширеною, але недоведеною гіпотезою про нерівність класів складності P і NP, не існує детермінованої машини Тьюрінга, здатної знаходити рішення примірників завдання за поліноміальний час залежно від кількості міст.

Однак існують алгоритми пошуку наближених рішень для метричної задачі за поліноміальний час і знаходження маршруту максимум удвічі довше оптимального.

На сьогодні не відомий жоден алгоритм з поліноміальним часом, який би гарантував розв'язок з точністю, за якою генерується розв'язок, кращий ніж 1,5 від оптимального.

Контрольні запитання та завдання

1. Наведіть неформальну постановку задачі комівояжера.

2. Що таке гамільтонів цикл?

3. Які методи оптимізації були засновані під час розв’язання задачі комівояжера?

4. Надайте характеристику асиметричної та симетричної задач комівояжера.

5. Що таке нерівність трикутника, застосування нерівності трикутника.

6. Коли може виникнути неметрична задача комівояжера?

7. Яка алгоритмічна складність задачі комівояжера?

8. Функція мети, умова кратності задачі комівояжера у вигляді задачі дискретної оптимізації.

ТЕМА 7
РОЗВ'ЯЗАННЯ ЗАДАЧІ КОМІВОЯЖЕРА МЕТОДОМ ЛІТТЛА

Основні питання

7.1 Метод гілок та меж розв'язання задачі комівояжера.

7.2 Метод Літтла – модифікація методу гілок та меж.

7.3 Приклад застосування методу Літтла.

7.4 Визначення процедури розгалуження в методі Літтла.

7.1 Метод гілок та меж розв'язання задачі комівояжера

Метрична задача комівояжера полягає у знаходженні найвигіднішого маршруту х, який проходить через вказані міста хоча б по одному разу з поверненням у вихідне місто.

Вершини графа відповідають містам, а ребра (i, j) між вершинами і та j – шляхи сполучення цих міст. Якщо задача розглядається на площині, то обчислюється евклідова відстань між парами міст у певному маршруті з урахуванням умови замкнутості і задача полягає у визначенні гамільтонового циклу на графі.

Маршрут комівояжера, який задовольняє умову задачі, задається такою перестановкою цілих чисел

$$p = \{(j_1, j_2),(j_2 , j_3),(j_3, j_4),...,(j_N , j_1)\}.$$

Кожному маршруту р ставиться у відповідність функція вартості (довжина)

$$f(p) = L(j_1, j_2) + L(j_2 , j_3) + L(j_3, j_4) + ... + L(j_N , j_1), \qquad (7.1)$$

де $L(j_j, j_k)$ – елементи матриці вартості (відстаней).

Одним із точних методів розв'язання задачі комівояжера є метод гілок та меж, що гарантує визначення глобально-оптимального розв'язку.

Метод гілок і меж – один із комбінаторних методів. Узагальнено його суть полягає в упорядкованому усіченому переборі можливих варіантів маршрутів, визначенні для кожного з них значенні функції мети (оцінки вартості або довжини маршруту) і розгляді лише тих з них, які виявляються за певними ознаками перспективними, при цьому безперспективні варіанти відкидаються.

У межах методу гілок і меж вся множина припустимих рішень (планів) деяким способом розбивається на підмножини, кожна з яких цим же способом знову розбивається на підмножини. На кожній з цих множин послідовно визначається розв'язок задачі, краще значення функції мети береться як оцінка зверху оптимального розв'язку. Процес триває до тих пір, поки не визначиться оптимальний розв'язок вихідної задачі.

Варто зазначити, що з погляду обчислювальної складності, задача комівояжера відноситься до NP-важких, тобто її перебірна складність не дозволяє знаходити точне рішення для великої кількості вершин графа за прийнятний час, навіть при введенні деяких обмежень.

Тому на практиці застосовують методи, які визначають певний квазіоптимальний маршрут (можливо, зовсім не найкращий, але близький до нього). Часто цю задачу зводять до задачі цілочисельного лінійного програмування, для вирішення якої теж використовується метод гілок та меж, який базується на стратегії «розділяй та володарюй».

Групою авторів (Дж. Літтл, К. Мурті, Д. Суїні, К. Керолл) була запропонована модифікація методу гілок і меж, розроблена спеціально для розв'язання задачі комівояжера. Згодом вона отримала назву «метод Літтла» (за прізвищем першого автора наукової праці).

Проведемо аналіз методу Літтла – Мурті – Суїні – Керолла, який є універсальним вдосконаленим методом повного перебору з послідовним відсівом рішень, що здаються невигідними, для пошуку квазіоптимального розв'язання задачі комівояжера.

7.2 Метод Літтла – модифікація методу гілок та меж

Алгоритм Літтла – Мурті – Суїні – Керолла можна сформулювати у вигляді таких правил.

Крок 0. Знайти матрицю найкоротших відстаней $L = \| l_{ij} \|$ із кожної вершини в іншу за алгоритмом Дейкстри щодо побудови дерева найкоротших шляхів [1].

Крок 1. Знайти в кожному рядку матриці відстаней L мінімальний елемент і відняти його від усіх елементів відповідного рядка. Отримаємо матрицю, приведену по рядках.

Якщо в матриці, яка приведена по рядках, існують стовпці, що не містять нуля, то приводимо її також і по стовпцях.

Крок 2. Знайти суму значень мінімальних елементів, тобто визначити константу приведення, яка буде нижньою межею множини всіх допустимих гамільтонових циклів.

Крок 3. Для кожного нульового елемента приведеної матриці визначити степінь нульового елемента. Для цього нуль у матриці замінити на знак «*» і знайти суму мінімальних елементів рядка і стовпця, які відповідають цьому нулю.

Крок 4. Вибрати ребро (i_0, j_0), для якого степінь нульового елемента досягає максимального значення.

Крок 5. Розбити множину всіх гамільтонових циклів Ω^0 на дві підмножини $\Omega^1_{i_0 j_0}$ та $\Omega^1_{\overline{i_0 j_0}}$.

Підмножина $\Omega^1_{i_0 j_0}$ містить (i_0, j_0), а підмножина $\Omega^1_{\overline{i_0 j_0}}$ його не містить.

Для отримання матриці вартості для підмножини $\Omega^1_{i_0 j_0}$ викреслюємо рядок i_0 і та стовпець j_0.

Щоб не допустити виникнення негамільтонового циклу, потрібно знайти рядок j і стовпець і без знака «∞», та на їхньому перетині поставити «∞».

Крок 6. Привести матрицю гамільтонових циклів $\Omega^1_{i_0 j_0}$ та обчислити нижню межу цієї множини $\varphi(\Omega^1_{i_0 j_0})$.

Аналогічні дії виконати для множини гамільтонових циклів $\Omega^1_{\overline{i_0 j_0}}$.

Крок 7. Виконати приведення матриці гамільтонових циклів $\Omega^1_{\overline{i_0 j_0}}$ та обчислити нижню межу цієї множини $\varphi(\Omega^1_{\overline{i_0 j_0}})$.

Крок 8. Порівняти нижні межі підмножин гамільтонових циклів $\Omega^1_{i_0 j_0}$ та $\Omega^1_{\overline{i_0 j_0}}$.

Якщо $\varphi(\Omega^1_{i_0 j_0}) < \varphi(\Omega^1_{\overline{i_0 j_0}})$, то подальшому розгалуженню підлягає множина $\Omega^1_{i_0 j_0}$, інакше $\Omega^1_{\overline{i_0 j_0}}$.

Процес розбиття множин на підмножини супроводжується побудовою дерева розгалужень.

Крок 9. Якщо в результаті розгалужень отримано матрицю 2 × 2, то визначити гамільтоновий цикл, який отримано розгалуженням, та обчислити його довжину f(p).

Крок 10. Порівняти довжину гамільтонового циклу f(p) з нижніми межами обірваних гілок.

Якщо довжина f(p) не перевищує їхніх нижніх меж, то задача вирішена. В іншому випадку розбиваємо гілки підмножин з нижньою межею, меншою

отриманого шляху, поки не отримаємо маршрут з меншою довжиною або не переконаємося, що такого не існує.

Зазвичай необхідно повторити алгоритм для N – 2 рівнів (етапів) дерева, де N – кількість пунктів у задачі комівояжера. Задовільних теоретичних оцінок цього алгоритму немає, але практика показує, що на сучасних комп'ютерах він дозволяє знайти оптимальний розв'язок задачі комівояжера для графів з кількістю вершин $N \leq 100$.

7.3 Приклад застосування методу Літтла

Приклад 7.1. Розглянемо повний зважений граф G = < V, E >, який подано на рисунку 7.1. Множина V містить 5 вершин графа G, множина E – 10 ребер графа G.

Задано матрицю вартості L, що містить ваги ребер графа G і має вигляд таблиці 7.1.

Необхідно знайти найвигідніший маршрут x (гамільтоновий цикл на графі G), який проходить через всі вершини графу хоча б по одному разу з поверненням у вихідну вершину.

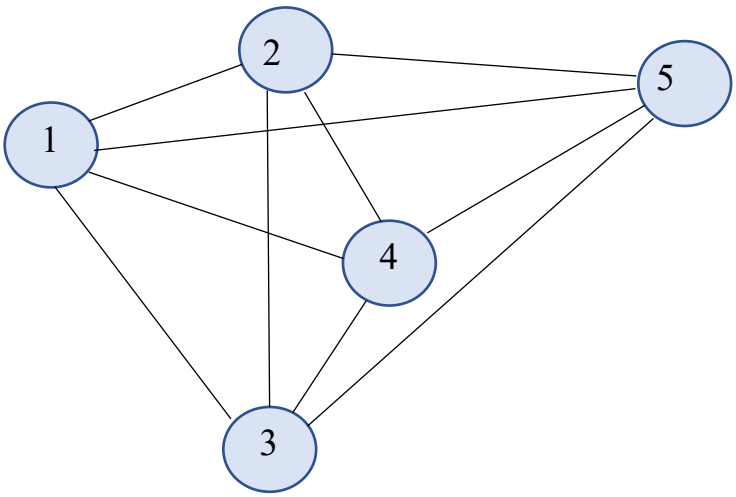


Рисунок 7.1 – Повний граф G з 5 вершинами

Визначимо один із допустимих гамільтонових циклів графа, що розглядається.

На першому кроці алгоритму знайдемо мінімальний елемент у кожному рядку матриці L1 і потім віднімемо його з інших елементів рядка (мінімальні елементи записані праворуч відповідних рядків таблиці 7.1 червоним кольором). Сума вилученої вартості = 57. Отримаємо матрицю L1 (табл. 7.2), подану нижче.

Таблиця 7.1 – Вихідна матриця вартості L (ваги ребер графа G)

L =

	1	**2**	**3**	**4**	**5**	
1	∞	12	22	28	32	12
2	12	∞	10	43	20	10
3	22	10	∞	15	10	10
4	28	43	15	∞	15	15
5	32	20	10	15	∞	10

$\Sigma_{\text{рядки}} = 57$

Те ж виконаємо і зі стовпцями матриці L1, що не містять нуля (табл. 7.2).

Таблиця 7.2 – Матриця вартості L1, приведена по рядках (ваги ребер графа G)

	1	**2**	**3**	**4**	**5**
1	∞	0	10	16	20
2	2	∞	0	33	10
3	12	0	∞	5	0
4	13	28		∞	0
5	22	10	0	5	∞
	2			5	

$\Sigma_{\text{стовпці}} = 7$

Отримаємо приведену матрицю L2 (табл. 7.3), що містить нулі в кожному рядку і кожному стовпці.

Поточна нижня межа вартості маршруту дорівнює сумі $\Sigma_{\text{рядки}} + \Sigma_{\text{стовпці}} = 64$.

Нижня межа вартості дорівнює сумі всіх забраних елементів в рядках і стовпцях.

Таблиця 7.3 – Приведена матриця вартості L2

	1	**2**	**3**	**4**	**5**
	∞	0	10	11	20
2	0	∞	0	28	10
3	10	0	∞	0	0
4	11	28		∞	0
5	20	10	0	0	∞

Для кожного нульового елемента матриці L2 розрахуємо значення Γ_{ij}, яке дорівнює сумі найменшого елемента і рядку (виключаючи елемент $l_{ij} = 0$) і найменшого елемента j стовпця:

$$\Gamma_{12} = 10, \Gamma_{21} = 10, \Gamma_{23} = 0, \Gamma_{32} = 0, \Gamma_{34} = 0, \Gamma_{35} = 0, \Gamma_{45} = 11, \Gamma_{53} = 0, \Gamma_{54} = 0.$$

Порівнюючи оцінки Γ_{ij}, визначаємо одну максимальну оцінку $\Gamma_{45} = 11$.

Максимальна оцінка $\Gamma_{45} = 11$ визначає момент розгалуження.

Внесемо в результуючий маршрут (орієнтований граф) дугу (4,5).

Видалимо з матриці L2 вартості рядок 4 і стовпець 5 отримаємо матрицю L4 (табл. 7.4).

Таблиця 7.4 – Матриця вартості L4

	1	**2**	**3**	**4**	**5**
1	∞	0	10	11	
2	0	∞	0	28	
3	10	0	∞	0	
4					
5	20	10	0	0	

У рядку 5 і стовпчику 4 відсутній елемент, що дорівнює ∞. Надамо елементу (5,4) значення нескінченності, щоб уникнути передчасного замикання контуру (табл. 7.5).

Таблиця 7.5 – Редукована матриця вартості L5

	1	**2**	**3**	**4**
1	∞	0	10	11
2	0	∞	0	28
3	10	0	∞	0
5	20	10	0	∞

Нагадуємо, що поточна нижня межа функції вартості маршруту: 64. Підсумкове значення нижньої межі має збігтися з довжиною результуючого контуру.

Матриця L5 має нульові елементи в кожному рядку та стовпчику. Тому для кожного такого елемента визначаємо оцінку Γ_{ij}, яка дорівнює сумі найменшого елемента і рядку (виключаючи елемент $l_{ij} = 0$) і найменшого елемента j стовпця:

$$\Gamma_{12} = 10, \Gamma_{21} = 10, \Gamma_{23} = 0, \Gamma_{32} = 0, \Gamma_{34} = 11, \Gamma_{53} = 10.$$

Є максимальний елемент $Г_{34} = 11$. Видалимо з матриці вартості рядок 3 і стовпець 4 (табл. 7.6). Внесемо в результуючий орієнтований граф дугу (3,4): {(3,4), (4,5)}.

Таблиця 7.6 – Матриця вартості L6

	1	**2**	**3**	**4**
1	∞	0	10	
2	0	∞	0	
3				
5	20	10	0	

У рядку 5 и стовпчику 3 матриці L6 відсутній елемент, що дорівнює ∞. Надамо елементу (5,3) значення нескінченності, щоб уникнути передчасного замикання контуру (табл. 7.7).

Таблиця 7.7 – Редукована матриця вартості L7

	1	**2**	**3**
1	∞	0	10
2	0	∞	0
5	20	10	∞

Знайдемо мінімальний елемент в рядку 5 (це 10) і потім віднімемо його з інших елементів рядка. Отримаємо матрицю L8, подану нижче (табл. 7.8).

Таблиця 7.7 – Редукована матриця вартості L8

	1	**2**	**3**
1	∞	0	10
2	0	∞	0
5	10	0	∞

При цьому поточна нижня межа вартості маршруту стає 64 + 10 = 74.

Визначення оцінок $Г_{ij}$

$$Г_{12} = 10, Г_{21} = 10, Г_{23} = 10, Г_{52} = 10$$

надає 4 однакових максимальних оцінки Г = 10.

Це означає, що алгоритм розгалужується, тому в загальному випадку необхідно розглянути всі отримані варіанти по черзі.

Видалимо з матриці вартості рядок 1 і стовпець 2. Внесемо в результуючий орієнтований граф дугу (1,2): {(1,2), (3,4), (4,5)}.

Виконуючи аналогічні кроки трансформації матриці вартості, отримаємо матрицю L9 вигляду (табл. 7.8):

Таблиця 7.7 – Редукована матриця вартості L9

	1	**3**
2	∞	0
5	10	∞

Визначивши мінімальний елемент в першому стовпчику матриці L9 та додавши його до наявної оцінки поточної нижньої межі вартості маршруту, отримаємо остаточне значення вартості маршруту: 84 та приведену матрицю вартості L10 (табл. 7.8).

Таблиця 7.8 – Редукована матриця вартості L10

	1	**3**
2	∞	0
5	0	∞

Згідно з формулюванням кроку 9 алгоритму Літтла ребра (2, 3) та (5,1) графа за інформацією матриці L10 вносяться до допустимого маршруту (гамільтонового циклу).

Таким чином, допустимий маршрут має вигляд: {(1,2), (2,3), (3,4), (4,5), (5,1)}, його вартість залишається 84 одиниці.

Якщо вихідний граф є не повним, то у вихідній матриці вартості L будуть відсутні певні елементи. У цьому випадку необхідно замінити нескінченні дуги на довжини найкоротших шляхів між відповідними вершинами (за винятком діагональних елементів, яким надається значення нескінченності).

Оскільки елементи отриманої матриці є вже не дугами, а відстанями між вершинами, то нерівність трикутника в цьому випадку виконується автоматично.

7.4 Визначення процедури розгалуження в методі Літтла

Ще раз нагадаємо, що суть методу гілок і меж як одного з комбінаторних методів – в упорядкованому усіченому переборі можливих варіантів маршрутів, визначенні для кожного з них значенні функції мети (оцінки вартості або довжини маршруту) і розгляді лише тих з них, які виявляються за певними ознаками перспективними, при цьому безперспективні варіанти відкидаються.

У межах методу гілок і меж вся множина припустимих рішень (планів) деяким способом розбивається на підмножини, кожна з яких цим же способом знову розбивається на підмножини.

На кожній з цих множин послідовно визначається розв'язок задачі, краще значення функції мети береться як оцінка зверху оптимального розв'язку. Процес триває до тих пір, поки не визначиться оптимальний розв'язок вихідної задачі.

Алгоритм Літтла – Мурті – Суїні – Керолла, що є модифікацією усіченого перебору за методом гілок та меж, викладено у пп. 7.2.

У пп. 7.3. наведено приклад визначення допустимого розв'язку – деякого гамільтонового циклу на графі. Розглянемо тепер процедуру розгалуження, використовуючи приклад нижче.

Приклад 7.1 Нехай є граф G(V,E).

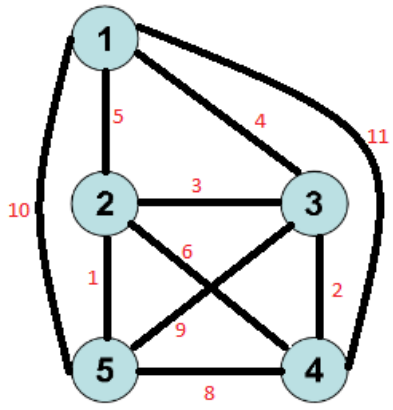


Рисунок 7.2 – Вихідний граф задачі комівояжера

Позначимо множину всіх маршрутів через Ω. Множина всіх маршрутів Ω містить 5! маршрутів.

Покладемо, що матриця відстаней графа G(V,E) після виконання першого кроку алгоритму Літтла має вигляд (табл. 7.9).

Таблиця 7.9 – Матриця відстаней графа G(V,E)

	1	2	3	4	5
1	0	5	4	11	10
2	5	0	3	6	1
3	4	3	0	2	9
4	11	6	2	0	8
5	10	1	9	8	0

Замінимо діагональні елементи матриці відстаней на ∞, забороняючи тим самим перехід з міста і до міста і.

Ітерація 1.

Крок 2. Знаходимо приведену матрицю відстаней. Для цього визначаємо мінімальний елемент у кожному рядку матриці відстаней та віднімаємо мінімальний елемент рядка від кожного елемента рядка. (табл. 7.10):

Таблиця 7.10 – Матриця відстаней, приведена по рядках

	1	2	3	4	5	min
1	∞	5	**4**	11	10	**4**
2	5	∞	3	6	**1**	**1**
3	4	3	∞	**2**	9	**2**
4	11	6	**2**	∞	8	**2**
5	10	**1**	9	8	∞	**1**

У матриці (табл. 7.10), яка приведена по рядках, є перший стовпчик, що не містить нуля, то приводимо матрицю також і по елементам першого стовпчика (табл. 7.11):

Крок 3. Визначаємо константу приведення φ(Ω) як суму мінімальних елементів рядків:

$$\varphi(\Omega) = 4 + 1 + 2 + 2 + 1 + 2 = 12.$$

Таблиця 7.11 – Приведена матриця відстаней

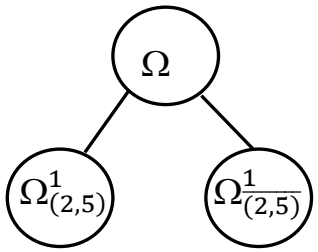
	1	2	3	4	5
1	∞	1	0	7	6
2	2	∞	2	5	0
3	0	1	∞	0	7
4	7	4	0	∞	6
5	7	0	8	7	∞

Крок 4. Для кожного нульового елемента приведеної матриці (табл. 7.11) визначаємо константи Γ_{ij}

$$\Gamma_{13} = 1;\ \underline{\Gamma_{25} = 8};\ \Gamma_{31} = 2;\ \Gamma_{34} = 5;\ \Gamma_{43} = 4;\ \Gamma_{52} = 8.$$

Крок 5. Найбільша сума констант приведення дорівнює (2 + 6) = 8 для ребра (2,5), отже множина маршрутів Ω розбивається на дві підмножини (рис. 7.3):

$$(2,5) \in \Omega^1_{(2,5)},\ (2,5) \notin \Omega^1_{\overline{(2,5)}},\ \Omega = \Omega^1_{(2,5)} \cup \Omega^1_{\overline{(2,5)}}.$$



$(2,5) \in \Omega^1_{(2,5)}$ $(2,5) \notin \Omega^1_{\overline{(2,5)}}$

Рисунок 7.3 – Побудова дерева рішень задачі. Перше розгалуження

Крок 6.

6.1 Приводимо матрицю гамільтонових циклів $\Omega^1_{(2,5)}$ та обчислюємо нижню межу цієї множини $\varphi(\Omega^1_{(2,5)})$.

Для отримання приведеної матриці для підмножини $\Omega^1_{(2,5)}$, викреслюємо рядок 2 і та стовпець 5. Щоб не допустити виникнення негамільтонового циклу, потрібно знайти рядок 5 і стовпець 2, та на їхньому перетині поставити «∞», а потім привести матрицю відстаней по рядках (табл. 7.12) і стовпцях (табл. 7.13).

Таблиця 7.12 – Матриця відстаней, приведена по рядках

	1	2	3	4	min
1	∞	1	0	7	
3	0	1	∞	0	
4	7	4	0	∞	
5	7	∞	8	7	7

Таблиця 7.13 – Матриця відстаней, приведена по стовпчиках

	1	2	3	4
1	∞	1	0	7
3	0	1	∞	0
4	7	4	0	∞
5	0	∞	1	0
min		1		

Отже, оцінка лівої гілки дерева розв'язків $\varphi(\Omega^1_{(2,5)}) = 12 + 8 = 20$.

6.2 Приводимо матрицю гамільтонових циклів $\Omega^1_{(2,5)}$ та обчислюємо нижню межу цієї множини $\varphi(\Omega^1_{(2,5)})$.

Замінюємо елемент (2,5) матриці відстаней на ∞, після чого здійснюємо чергове приведення матриці відстаней, у результаті отримаємо редуковану матрицю за рядками та за стовпцями (табл. 7.14).

Нижня границя гамільтонових циклів підмножини $\varphi(\Omega^1_{(2,5)})$ дорівнює

$$\varphi(\Omega^1_{(2,5)}) = 12+2+6= 20.$$

Таблиця 7.14 – Редукована матриця відстаней за рядками та за стовпцями

	1	2	3	4	5
1	∞	1	0	7	0
2	0	∞	0	3	∞
3	0	1	∞	0	1
4	7	4	0	∞	0
5	7	0	8	7	∞

Оскільки нижні границі підмножини $\Omega^1_{(2,5)}$ та підмножини $(\Omega^1_{(2,5)})$ дорівнюють один одному: $\varphi(\Omega^1_{(2,5)}) = \varphi(\Omega^1_{(2,5)}) = 20$, розвиваємо ліву гілку дерева рішень, та ребро (2,5) включаємо в маршрут з новою границею 20 (рис. 7.4).

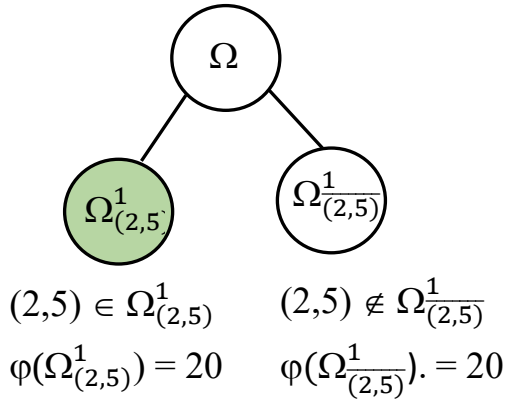


Рисунок 7.4 – Розвиваємо ліву гілку дерева рішень

Ітерація 2.

Крок 4. Розглядаємо приведену матрицю відстаней (табл. 7.15), сформовану з таблиці 7.13.

Таблиця 7.15 – Приведена матриця відстаней на другому кроці алгоритму

	1	2	3	4
1	∞	0	0	7
3	0	0	∞	0
4	7	3	0	∞
5	0	∞	1	0

Для кожного нульового елемента приведеної матриці (табл. 7.15) визначаємо величину Γ_{ij}.

$\Gamma_{12} = 0$; $\Gamma_{13} = 0$; $\Gamma_{31} = 0$; $\Gamma_{32} = 0$; $\Gamma_{34} = 0$; $\Gamma_{43} = 3$; $\Gamma_{54} = 0$; $\Gamma_{51} = 0$.

Найбільша сума констант приведення дорівнює (3 + 0) = 4 для ребра (4,3), отже, множина маршрутів $\Omega^1_{(2,5)}$ розбивається на дві підмножини $\Omega^2_{(4,3)}$ та $\Omega^2_{\overline{(4,3)}}$ (рис. 7.5).

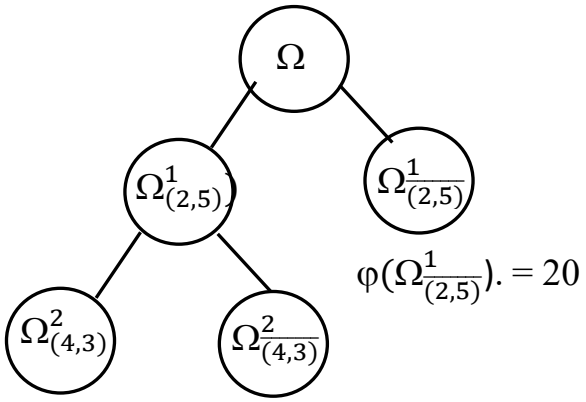


Рисунок 7.5 – Подальше формування дерева рішень

Крок 6.

6.1 Приводимо матрицю гамільтонових циклів $\Omega^2_{(4,3)}$ та обчислюємо нижню межу цієї множини $\varphi(\Omega^2_{(4,3)})$.

Для отримання приведеної матриці для підмножини $\Omega^2_{(4,3)}$ викреслюємо рядок 4 і та стовпець 3. Щоб не допустити виникнення негамільтонового циклу, у комірці (3, 4) ставимо «∞» (табл. 7.16).

Таблиця 7.16 – Попередження виникнення негамільтонового циклу

	1	2	4
1	∞	0	7
3	0	0	∞
5	0	∞	0

Матриця відстаней (табл. 7.16) є приведеною, тому оцінка $\varphi(\Omega^2_{(4,3)}) = 20$.

7.2 Аналогічні дії виконаємо для множини гамільтонових циклів $\Omega^2_{\overline{(4,3)}}$ Приводимо матрицю гамільтонових циклів $\Omega^2_{\overline{(4,3)}}$ та обчислюємо нижню межу цієї множини $\varphi(\Omega^2_{\overline{(4,3)}})$.

Замінюємо елемент (4,3) матриці відстаней (табл. 7.14) на ∞, після чого здійснюємо чергове приведення матриці відстаней, у результаті отримаємо редуковану матрицю (табл. 7.17) за рядками.

Таблиця 7.17 – Редукована матриця відстаней на поточній ітерації

	1	2	3	4	min
1	∞	0	0	7	
3	0	0	∞	0	
4	7	3	∞	∞	3
5	0	∞	1	0	

Нижня границя $\varphi(\Omega^2_{\overline{(4,3)}})$ гамільтонових циклів цієї підмножини

$$\varphi(\Omega^2_{\overline{(4,3)}}) = 20 + 3 = 23.$$

Оскільки нижня границя підмножини $\Omega^1_{(2,5)}$ менше за нижню границю підмножини ($\Omega^1_{(4,3)}$), $\varphi(\Omega^2_{(4,3)}) \le \varphi(\Omega^2_{(4,3)})$, розвиваємо ліву гілку дерева рішень, та ребро (4,3) включаємо в маршрут (рис. 7.6).

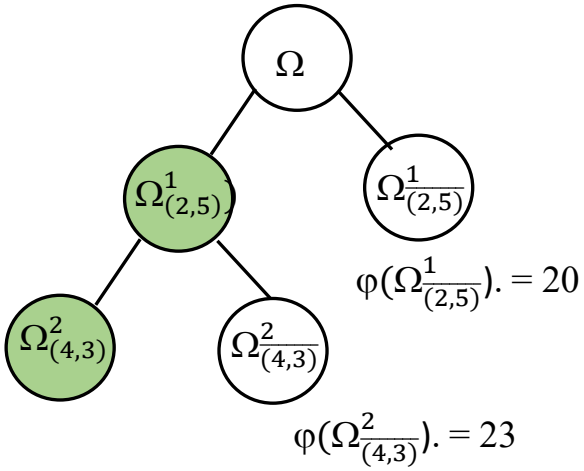


Рисунок 7.6 – Розвиток лівої гілки дерева рішень

Складові маршруту: {(2,5); (4,3))

Ітерація 3.

Крок 3. Розглядаємо матрицю відстаней (табл. 7.16). Вона є приведеною, обчислюємо константи Γ_{ij}.

$$\Gamma_{31} = 0;\ \Gamma_{32} = 0;\ \underline{\Gamma_{54} = 7};\ \Gamma_{12} = 0;\ \Gamma_{51} = 0.$$

Найбільша сума констант приведення дорівнює (0 + 7) = 7 для ребра (5,4), отже множина маршрутів $\Omega^2_{(4,3)}$ розбивається на дві підмножини:
$\Omega^3_{(5,4)}$ та $\Omega^3_{(5,4)}$ (рис. 7.7).

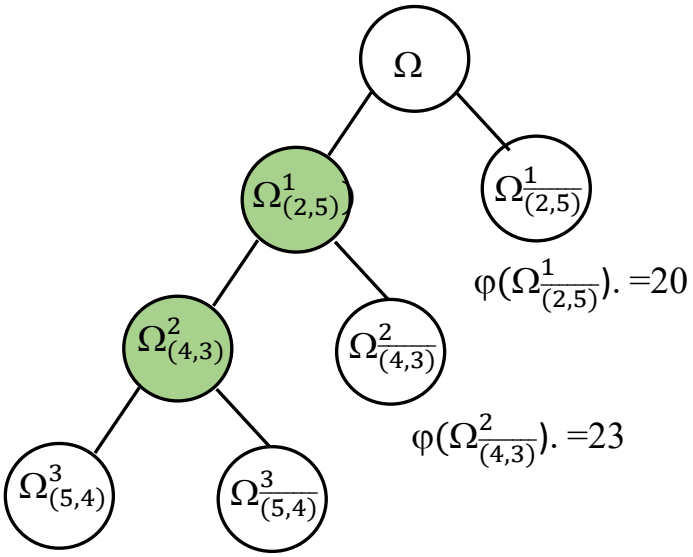


Рисунок 7.7 – Подальше розгалуження дерева рішень

Крок 6.

6.1 Приводимо матрицю гамільтонових циклів $\Omega^3_{(5,4)}$ та обчислюємо нижню межу цієї множини $\varphi(\Omega^3_{(5,4)})$.

Для отримання приведеної матриці для підмножини $\Omega^3_{(5,4)}$ викреслюємо рядок 5 і та стовпець 4. (табл. 7.18).

Таблиця 7.18 – Приведена матриця відстаней на кроці 6

	1	2
1	∞	0
3	0	0

Матриця відстаней (табл. 7.18) є приведеною, тому оцінка $\varphi(\Omega^3_{(5,4)}) = 20$.

6.2 Аналогічні дії виконаємо для множини гамільтонових циклів $\Omega^3_{\overline{(5,4)}}$. Приводимо матрицю гамільтонових циклів $\Omega^3_{\overline{(5,4)}}$ та обчислюємо нижню межу цієї множини $\varphi(\Omega^3_{\overline{(5,4)}})$.

Замінюємо елемент (5,4) матриці відстаней (табл. 7.16) на ∞, після чого здійснюємо чергове приведення матриці відстаней та отримаємо редуковану матрицю (табл. 7.19) за рядками. Приведемо цю матрицю за стовпчиками:

Таблиця 7.19 – Приведена матриця відстаней за стовпчиками

	1	2	4
1	∞	0	7
3	0	0	∞
5	0	∞	∞
min			7

$\varphi(\Omega^3_{\overline{(5,4)}}) = 20 + 7 = 27$.

Оскільки нижня границя підмножини $\Omega^3_{(5,4)}$ менше за нижню границю підмножини $(\Omega^3_{\overline{(5,4)}})$, $\varphi(\Omega^3_{(5,4)}) \le \varphi(\Omega^3_{\overline{(5,4)}})$, розвиваємо ліву гілку дерева рішень, та ребро (5,4) включаємо в маршрут (рис. 7.8).

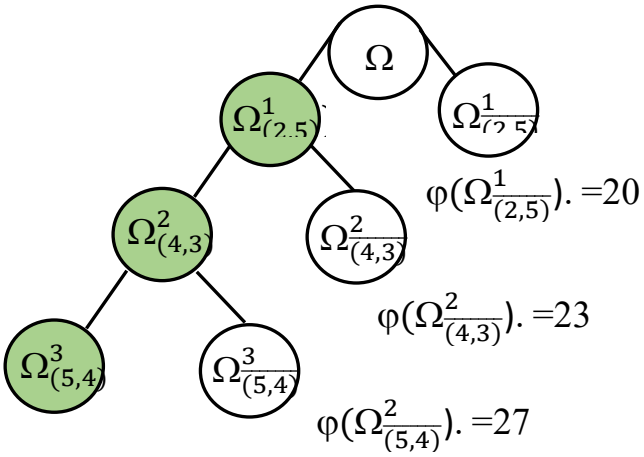


Рисунок 7.8 – Подальше розгалуження дерева рішень задачі

Складові маршруту: { (2,5); (4,3); (5,4)}.

Ітерація 3.

Крок 3. Розглядаємо матрицю відстаней (табл. 7.18).

Додаємо до маршруту ребро (1,2) – за умови задачі, що кожна вершина має бути в маршруті:

{ (1,2); (2,5); (4,3); (5,4)}.

Більш того, за умовою задачі кожна вершина повинна мати степінь 2, тому обираємо ребро (3,1).

Отже, побудовано маршрут

{(1,2); (2,5); (5,4); (4,3); (3,1)}.

Його довжина

5 + 1 + 8 + 2 + 4 = 20.

За оцінками термінальних вершин дерева рішень – цей маршрут є оптимальним.

Контрольні запитання та завдання

1. Викладіть суть методу гілок та меж.
2. Наведіть оцінку обчислювальної складності задачі комівояжера.
3. Опишіть кроки алгоритму Дейкстри побудови дерева найкоротших шляхів. Наведіть приклад застосування.

4. Яку роль відіграє алгоритм Дейкстри для процесу розв’язання задачі комівояжера.

5. Що таке константа приведення в алгоритмі Літтла?

6. Як в алгоритмі Літтла не допускається виникнення негамільтонового циклу?

7. Як відбувається розгалуження у методі Літтла?

ТЕМА 8
ГЕНЕТИЧНІ АЛГОРИТМИ

Основні питання

8.1 Класифікація методів дискретної оптимізації за точністю отриманого розв'язку.

8.2 Загальні відомості про генетичні алгоритми.

8.3 Критерії завершення роботи генетичного алгоритму.

8.1 Класифікація методів дискретної оптимізації за точністю отриманого розв'язку

Розглянемо задачу дискретної оптимізації.

Необхідно знайти елемент $x^* \in X$ (X – дискретна множина припустимих рішень), такий, що

$$x^* = \arg\operatorname*{extr}_{x \in X} f(x), \qquad (8.1)$$

де f(x) – цільова функція.

Будемо вважати, що метод дискретної оптимізації – це певна процедура A, яка переводить задану підмножину $Z \subset X$ (початкові наближення) у множину $X^* \subset X$.

Існує багато підходів до класифікації методів дискретної оптимізації – за точністю, типом використаних просторів, структурою обчислювальної схеми тощо.

Проведемо класифікацію методів дискретної оптимізації за точністю отриманого розв'язку (рис. 8.1).

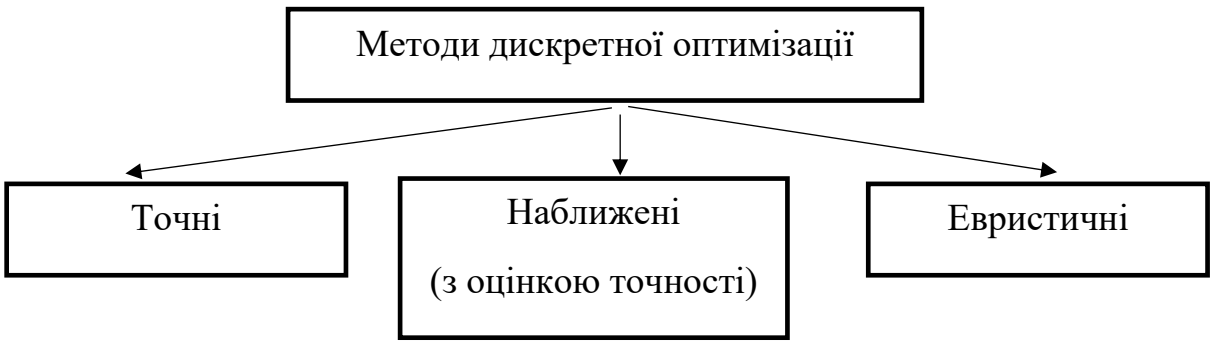


Рисунок 8.1 – Класифікація методів дискретної оптимізації за точністю отриманого розв'язку

Точні методи дискретної оптимізації – це такі методи, які знаходять глобальний розв'язок, тобто для цих методів

$$X^* \subseteq \text{Arg } ext\ f(x).$$

Наприклад, це метод повного перебору, метод гілок та меж та його варіанти – усічений перебір, динамічне програмування.

Наближені методи дискретної оптимізації діляться на алгоритми з апріорною та апостеріорною оцінками точності.

Евристичні алгоритми будуються на основі правдоподібних міркувань (наприклад, алгоритм «прямуй в найближче місто», що використовується для розв'язання задачі комівояжера), але оцінити точність знайденого розв'язку не видається можливим.

Однак часто дослідники називають наближеними як алгоритми з оцінкою точності, так і евристичні.

8.2 Загальні відомості про генетичні алгоритми

Генетичні алгоритми (ГА, genetic algorithms) виникли на основі спостереження за природними процесами еволюції та селекції популяцій живих істот – **особин** певного виду – і моделювання їхніх принципів функціонування.

При описі генетичних алгоритмів зазвичай використовують терміни, запозичені з генетики. Надамо перелік цих термінів.

Популяція – це скінченна множина особин.

Особини, що входять до популяції, у генетичних алгоритмах подаються хромосомами із закодованими в них множинами параметрів задачі, переважно розв'язків, які інакше називаються ***точками в просторі пошуку*** (search points) задачі.

Хромосоми (інші назви – ланцюжки або кодові послідовності) – це впорядковані послідовності генів.

Ген (який також називається властивістю, знаком чи детектором) – це атомарний елемент генотипу, зокрема хромосоми.

Генотип, або **структура** – це набір хромосом цієї особини.

Отже, особинами популяції можуть бути генотипи або одиничні хромосоми.

Фенотип – це сукупність зовнішніх і внутрішніх ознак, які відповідають цьому генотипу, тобто декодована структура, або множина параметрів задачі (розв'язок, точка простору пошуку).

Алель – це альтернативне значення конкретного гена, також визначається як значення властивості або варіант властивості.

Локус (позиція) – указує на місце розміщення певного гена в хромосомі (ланцюжку амінокислот).

Множина позицій генів – це локи.

Основну схему еволюційного відбору, покладену в основу розробки генетичних алгоритмів, наведено на рисунку 8.2.

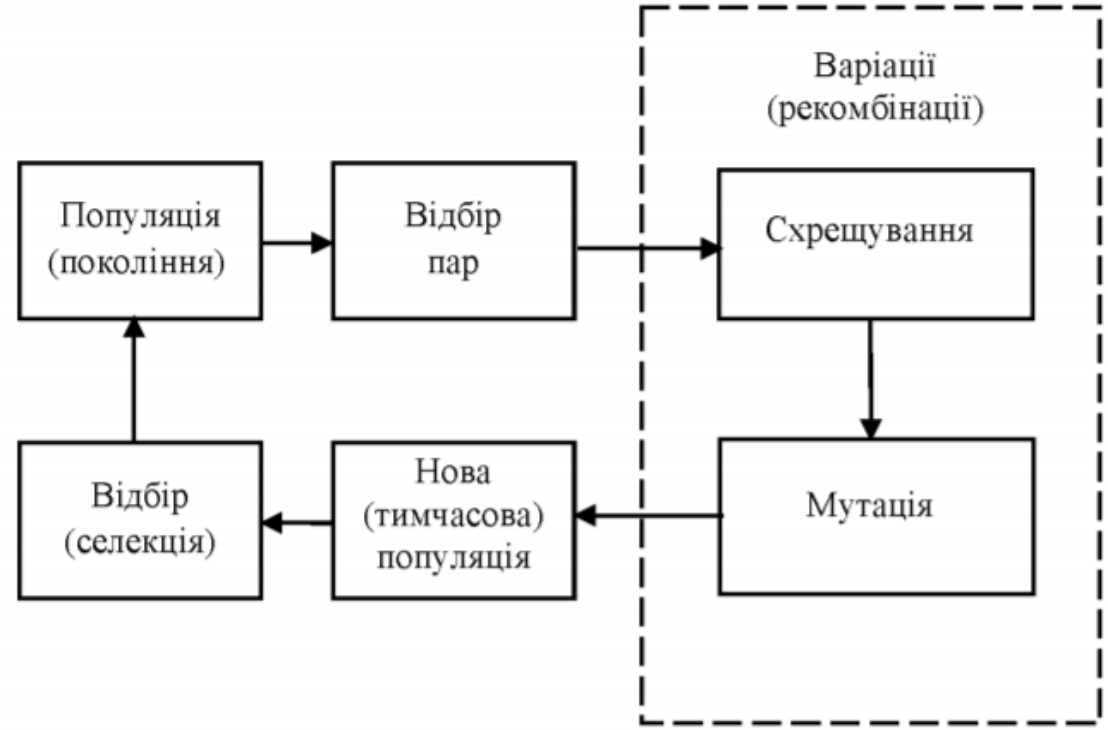


Рисунок 8.2 – Схема еволюційного відбору

Функція пристосованості (Fitness function), яка ще називається функцією корисності чи придатності, виражає міру пристосованості особин у популяції.

Ця функція дозволяє оцінити ступінь пристосованості конкретних особин у популяції й вибрати з них найпридатніші відповідно до еволюційного принципу виживання найсильніших.

У задачах оптимізації функція пристосованості зазвичай формується на основі цільової функції й оптимізується. Процес появи в гені нових рис або підсилення вже наявних сильних рис, відображених у алелях, під впливом зовнішнього середовища називається **мутацією**.

Отже, генетичний алгоритм – це еволюційний алгоритм пошуку, що використовується для розв'язання задач оптимізації, моделювання чи підтримки ухвалення рішень шляхом послідовного відбору, комбінування й варіації шуканих параметрів із використанням механізмів, що нагадують біологічну еволюцію.

Загальна ідея генетичного алгоритму – перенесення принципу еволюційного відбору на процес оптимізації.

Мета таких алгоритмів – підвищення загального рівня пристосованості популяції.

Загальний підхід, що моделює процес еволюції в живій природі та використовується в еволюційних алгоритмах, подано на рисунку 8.3.

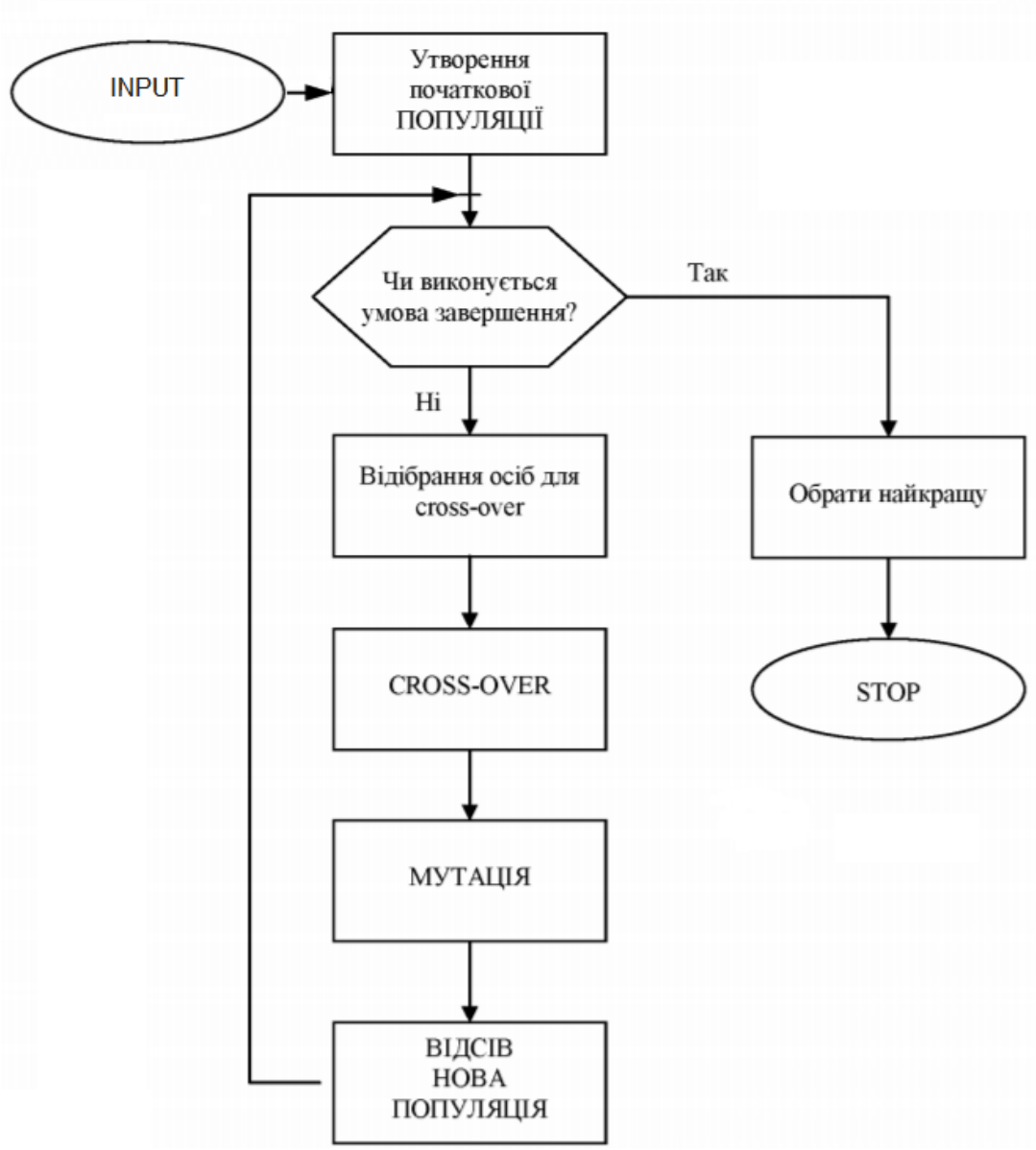


Рисунок 8.3 – Рулетка як тема вибору особин

Розробка генетичного алгоритму для розв'язання певної задачі дискретної оптимізації вимагає конкретизації таких аспектів:

– принципи формування початкової популяції;

– механізми відбору;

– вибір типу кросовера (схрещування);

– вибір оператора мутації;

– умова завершення.

Батьки – особини (хромосоми, генотипи) з популяції на поточній ітерації генетичного алгоритму.

Нащадки – особини (хромосоми, генотипи) з популяції на наступній ітерації генетичного алгоритму.

Кросовер – генетичний оператор схрещування, який здійснює обмін генами або частинами хромосом між двома батьками з метою отримання нових хромосом, які є кодуваннями їхніх нащадків.

При цьому генетичний матеріал з хромосом батьків згідно з властивістю спадковості безпосередньо передається в хромосоми нащадків.

Формування початкової популяції. Під час вибору початкової популяції часто застосовують процедури породження випадковим чином m варіантів розв'язку ЗКО, де m – параметр алгоритму, який задається. При такому формуванні використовуються чотири основні принципи:

1) дробовик – випадковий вибір з усього простору розв'язків X;

2) фокусування – випадковий вибір із заданої підобласті X;

3) ковдра – генерування повної популяції, що включає всі можливі розв'язку із заданої підобласті X;

4) комбінування – сумісна реалізація всіх трьох попередніх принципів.

Таким чином, початкова популяція створюється випадковим чином.

Навіть якщо початкова популяція виявиться абсолютно неконкурентоспроможною, ймовірно, що генетичний алгоритм все одно достатньо швидко переведе її в життєздатну популяцію.

Тому на першому етапі достатньо, щоб особини відповідали формату особин популяції, і на них можна було підрахувати функцію пристосованості (Fitness).

Підсумком першого кроку є популяція H, що складається з N особин.

Приклад 8.1.

Для задачі комівояжера як початкова популяція H виступає скінченна множина N довільних маршрутів комівояжера, що є гамільтоновими циклами.

8.3 Критерії завершення роботи генетичного алгоритму

Під час реалізації генетичного алгоритму найчастіше використовуються такі умови завершення обчислень:

1. Перевищення заданої кількості К ітерацій (змін поколінь / популяцій), Л – параметр алгоритму.

2. Відсутність (або невелика зміна) поліпшення наявного рекорду чи середньої пристосованості популяції на кількох останніх ітераціях.

3. Вичерпання заданого часу на обчислення.

На **етапі відбору** потрібно з усієї популяції вибрати певну її частку, яка залишиться «в живих» на цьому етапі еволюції.

Є різні способи проводити відбір. Імовірність виживання особини h повинна залежати від значення функції пристосованості Fitness (h). Сама частка **s** тих, хто вижив, зазвичай є параметром генетичного алгоритму, і її задають заздалегідь.

За підсумками відбору з N особин популяції H повинні залишитися sN особин, які увійдуть до підсумкової популяції H'. Решта особин гинуть.

У генетичному алгоритмі (як і в інших еволюційних алгоритмах) може бути виділено дві форми відбору.

У першій формі – **відбір для варіації** – особини вибираються для рекомбінації та/або мутації.

У другій – **відбір популяції** – особини відбираються для нового покоління. Це інколи називається заміною, оскільки деякі або всі батьки замінюються деякими або всіма нащадками.

Таким чином створюється наступне «покоління». Особини наступного покоління також оцінюються, потім проводиться селекція, застосовуються генетичні оператори і так далі.

Так моделюється «еволюційний процес», що триває кілька життєвих циклів (поколінь), поки не буде виконано критерій зупинки алгоритму.

Традиційний вид генетичного алгоритму базується на виборі **пропорційної пристосованості.**

При виборі пропорційної пристосованості ймовірність вибору деякого варіанта x_i із поточної популяції P у більшості ГА задається формулою

$$p(x_i) = \frac{f_i}{\sum_k f_k}.$$

Приклад 8.2. Якщо поточне покоління полягає з 4 особин

$$S_1, \quad S_2, \quad S_3 \quad \text{і} \quad S_4$$

зі значеннями придатності

$$f_1 = 2,\ f_2 = 4,\ f_3 = 1,\ f_4 = 1,$$

то у формуванні нового покоління вони братимуть участь з можливостями відповідно 0,25; 0,5; 0,125 і 0,125.

Чим вище пристосованість особини (чим вона «сильніше»), тим більше у неї шансів залишити потомство.

Цю схему відбору можна розглядати як вибір особини за допомогою рулетки (рис. 8.4), відносні площі q_i секторів якої дорівнюють ймовірностям

$$q_i = p(x_i) = \frac{f_i}{\sum_k f_k}.$$

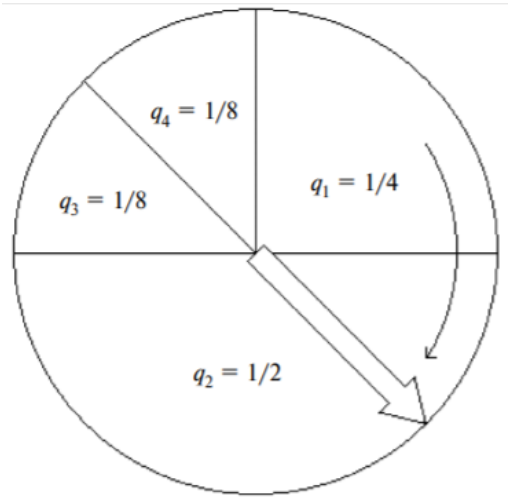


Рисунок 8.4 – Вибір особин для схрещування та мутації

За аналогією з реальним життям – чотири рази обертається рулетка і чотири рази вибирається особина для схрещування та мутації. Одна і та саме особина може брати участь у формуванні потомства кілька разів, найімовірніше це буде найбільш пристосована особа.

Схеми схрещування.

Розглянемо схему одноточкового схрещування.

Одноточкове схрещування організовується у такий спосіб. Якщо є два батьки з хромосомами $S_1 = (S_{11}, S_{12}, ..., S_{1N})$ і $S_2 = (S_{21}, S_{22}, ..., S_{2N})$, то хромосоми їхніх нащадків у новому поколінні є

$$(S_{11}, ..., S_{1m}, S_{2m+1}, ..., S_{2N}) \text{ і } (S_{21}, ..., S_{2m}, S_{1m+1}, ..., S_{1N});$$

тобто «голова» і «хвіст» хромосом нащадка беруться від різних батьків.

Точка схрещування вибирається випадковим чином, у наведеному прикладі вона розміщується між m-м та (m+1)-м символами.

Приклад 8.3. Розглянемо хромосоми $S_1 = (1, 2, 3, 4, 5)$ та $S_2 = (0, 0, 0, 0, 0)$. Нехай точка схрещування розташована між 3 та 4 генами.

Тоді m = 3.

$$S_1 = (1, 2, 3, \mid 4, 5) \text{ та } S_2 = (0, 0, 0, \mid 0, 0).$$

нащадки мають вигляд

$$S_1 = (1, 2, 3, 0, 0) \text{ та } S_2 = (0, 0, 0, 4, 5).$$

На цей час продовжуються подальші дослідження щодо впровадження генетичних алгоритмів у прикладні області діяльності професіоналів з комп'ютерних наук та інформаційних технологій.

Генетичні алгоритми можна застосовувати до широкого спектра задач у різних галузях, включаючи інженерію, інформатику, фінанси, біологію, логістику тощо, не вимагаючи при цьому особливих математичних властивостей цільової функції.

Важливою властивістю генетичних алгоритмів є те, що генетичні алгоритми за своєю природою є паралельними, оскільки оперують з популяціями рішень, уможливлює їхню придатність для паралельних обчислень, значно прискорюючи процес пошуку.

Контрольні запитання та завдання

1. Як у генетичному алгоритмі відбувається ініціалізація, або вибір вихідної популяції хромосом.

2. Наведіть визначення оцінки пристосованості хромосом у популяції.

3. Як відбувається перевірка умови зупинки алгоритму.
4. Наведіть суть алгоритму селекції хромосом.
5. Визначить основні особливості застосування генетичних операторів.
6. Як відбувається формування нової популяції.
7. Наведіть алгоритм вибору «найкращої» хромосоми.

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Дистанційний курс «Прикладні задачі дискретного аналізу» [Електрон. ресурс]. – Електрон. текст. дані. – Режим доступу: https://dl.kname.edu.ua/course/view.php?id=1752, вільний (дата звернення 05.05.2025). – Назва з екрана.

2. Федак І. В. Курс лекцій з функціонального аналізу та теорії міри. Ч.1. Вимірні множини та вимірні функції [Електрон. ресурс] : навч. посіб. / І. В. Федак. – Електрон. текст. дані. – Івано-Франківськ : ПНУ імені Василя Стефаника, 2020. – 52 с. – Режим доступу: https://kmfa.pnu.edu.ua/wp-content/uploads /sites/64/2020/02-Федак-І.В.-Курс-лекцій-з-функціонального-аналізу-та-теорії-міри-Ч1.pdf, вільний (дата звернення 05.05.2025). – Назва з екрана.

3. Rothvoss T. Discrete Optimization [Electronic resource] : Course Notes / T. Rothvoss. – Electronic text data. – Washingtong: University of Washingtong, 2020 – 86 p. – Regime of access: https://sites.math.washington.edu/~rothvoss/lecturenotes/DisOpt409-Spring2020.pdf, free (application date: 05.05.2025). – Header from the screen.

4. Темнікова О. Л. Дискретна математика : конспект лекцій (Частина 1) [Електрон. ресурс] : навч. посіб. / О. Л. Темнікова. – Електрон. текст. дані. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 154 с. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/42839/1/LectureDM1Temnikova.pdf, вільний (дата звернення 05.05.2025). – Назва з екрана.

5. Кузьменко І. М. Теорія графів [Електрон. ресурс] : навч. посіб. / І. М. Кузьменко. – Електрон. текст. дані. – Київ : КПІ ім. Ігоря Сікорського. – 2020. – 71 с. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/35854 /1/Teoriia_hrafiv.pdf, вільний (дата звернення 05.05.2025). – Назва з екрана.

6. W. J. Martin III. Discrete Optimization [Electronic resource]: Course Notes / Martin III W. J. – Electronic text data. – Worcester: Worcester Polytechnic Institute. – 2022. – 232 p. – Regime of access: https://users.wpi.edu/~martin/TEACHING/3233 /MA3233LectureNotes.pdf, free (application date: 05.05.2025). – Header from the screen.

7. Жданова О. Г. Дослідження операцій: Вступ до дискретного програмування: Практикум [Електрон. ресурс] : навч. посіб. / О. Г. Жданова, В. Д. Попенко, М. О. Сперкач. – Електрон. текст. дані. – Київ : КПІ ім. Ігоря Сікорського, 2019. – 47 с. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/32225/1/Praktykum2_Dosl.operats.-DyskrProhr.pdf, вільний (дата звернення 05.05.2025). – Назва з екрана.

8. Course of Discrete Mathematics [Electronic resource]. – Electronic text data. – Regime of access: https://www.coursera.org/learn/discrete-mathematics, free (application date: 05.05.2025). – Header from the screen.

9. Course of Data Structures and Algorithms Specialization. Algorithms on Graphs [Electronic resource]. – Electronic text data. – Regime of access: https://www.coursera.org/learn/algorithms-on-graphs?specialization=data-structuresalgorithms, free (application date: 05.05.2025). – Header from the screen10.

10. Discrete Mathematics. Master Discrete Math with 400+ Practice Questions and Quizzes: Foundational for Computer Science and Math Students [Electronic resource]. – Electronic text data. – Regime of access: https://www.udemy.com/course/discrete-math/?srsltid=AfmBOoq6m7ASVGKewZKeoxFrxiuYhRgLkLYXZCVfBP4FXm9daW2wcbSI, free (application date: 05.05.2025). – Header from the screen.

Електронне навчальне видання

НОВОЖИЛОВА Марина Володимирівна

ПРИКЛАДНІ ЗАДАЧІ ДИСКРЕТНОГО АНАЛІЗУ

КОНСПЕКТ ЛЕКЦІЙ

(для здобувачів першого (бакалаврського) рівня вищої освіти всіх форм навчання зі спеціальностей 122, F3 – Комп'ютерні науки, 126, F6 – Інформаційні системи та технології)

Відповідальний за випуск *А. Л. Литвинов*

Редактор *О. В. Михаленко*

Комп'ютерне верстання *М. В. Новожилова*

План 2022, поз. 118Л

Підп. до друку 26.05.2025. Формат 60 × 84/16.
Ум. друк. арк. 4,8.

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О. М. Бекетова,
вул. Чорноглазівська (Маршала Бажанова), 17, Харків, 61002.
Електронна адреса: office@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 5328 від 11.04.2017.