МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ЛЕСІ УКРАЇНКИ

Кафедра математичного аналізу та статистики

На правах рукопису

Давидюк Наталія Вячеславівна

ЗНАХОДЖЕННЯ НАЙКОРОТШИХ МАРШРУТІВ У ГРАФАХ

Спеціальність: 111 Математика

Освітньо-професійна програма «Математика»

Робота на здобуття освітнього ступеня «Магістр»

Науковий керівник:
ШВАЙ ОЛЬГА ЛЕОНІДІВНА,
кандидат педагогічних наук, доцент
кафедри математичного аналізу та
статистики

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри математичного аналізу
та статистики
від _____ 20___ р.
Завідувач кафедри
доц. Федуник-Яремчук О.В. _____

Луцьк – 2024

Зміст

ВСТУП

У сучасному світі теорія графів є однією з основних галузей дискретної математики, яка знаходить численні застосування в різних сферах діяльності, таких як комп'ютерні науки, транспортні системи, логістичні системи, соціальні мережі та інше. У теорії графів окремо можна виділити завдання по знаходженню найкоротших маршрутів, воно є критично важливим для оптимізації шляхів у транспортних системах, мінімізації витрат у комунікаційних мережах, а також допомагає у пошуку найкоротших зв'язків у мережах обміну даними.

Алгоритми пошуку найкоротших маршрутів, як от алгоритм Дейкстри та алгоритм Флойда-Воршелла, забезпечують ефективні розв'язки для багатьох практичних задач з оптимізації.

Алгоритм Дейкстри дозволяє знаходити найкоротші шляхи від однієї вершини до всіх інших у графі з додатними вагами ребер, що допогамає у вирішенні завдань, пов'язаних із транспортними системами та зв'язком.

За допомогою алгоритма Флойда, можна знайти найкоротші шляхи між усіма парами вершин, такий алгоритм широко використовується у задачах глобальної оптимізації шляхів у мережах з різними вагами ребер.

Мета магістерської роботи – проаналізувати основні алгоритми, які використовуються для з пошуку найкоротших маршрутів у графах та проілюструвати їх застосування при розв'язанні практичних задач.

Для досягнення поставленої мети визначені такі завдання:

1. Провести аналіз літературних джерел з теорії графів, зокрема, щодо шляхів і циклів у графах.
2. Описати та порівняти найбільш ефективні алгоритми пошуку найкоротших маршрутів у графах.
3. Дослідити принципи роботи алгоритму Дейкстри та провести аналіз його ефективності.
4. Дослідити алгоритм Флойда-Воршелла, порівняти його з алгоритмом Дейкстри та проаналізувати його ефективність у задачах.

5. Дослідити історичні аспекти розвитку поняття «граф».

Для виконання завдань магістерської роботи були використані наступні *методи дослідження*: теоретичний аналіз наукової літератури з метою виявлення сучасних поглядів на досліджувану проблему; порівняльний аналіз та синтез; методи математичного моделювання графів тощо.

Об'єкт дослідження – графи як математичні моделі, що застосовуються для опису та аналізу різноманітних мережевих структур, зокрема транспортних і комунікаційних.

Предмет дослідження – методи пошуку найкоротших маршрутів у графах, зокрема алгоритми Дейкстри та Флойда-Воршелла, а також їхні властивості, обмеження та застосування.

Наукова новизна дослідження полягає в тому, що в роботі досліджено та проілюстровано на прикладах самостійно розв'язаних автором вправ застосування основних алгоритмів, які використовуються для пошуку найкоротших маршрутів у графах

Практичне значення роботи – матеріал наукової роботи може бути використаний студентами при вивченні теорії графів та учнями ЗЗСО на заняттях математичних гуртків та факультативів.

Структура та об'єм дослідження: робота складається зі вступу; двох розділів, які в свою чергу містять підрозділи; висновків, додатків, та списку використаної літератури.

Зміст роботи викладений на 41 сторінках друкованого тексту.

*Апробація дослідження***.** Результати магістерської роботи були представлені у повідомленні на XVIII Міжнародній науково-практичній конференції студентів, аспірантів та молодих вчених «Молода наука Волині: пріоритети та перспективи досліджень» та опубліковані у матеріалах цієї конференції [9].

РОЗДІЛ 1

ЗАГАЛЬНІ ВІДОМОСТІ ПРО ГРАФИ

1.1. Поняття графа. Основні визначення.

Свій розвиток теорія графів почала з праць Леонарда Ейлера, який у 1736 році розв'язав задачу про Кенігсберзькі мости. Довгий час дослідження Леонарда Ейлера були єдиними результатами теорії графів. Лише із середини 19 століття були отримані нові результати теорії графів, а інтенсивний розвиток теорія графів отримала уже у 20 столітті.

Графи є однією з головних структур, яка має застосування в інформаційних технологіях, логістиці, мережевих технологіях та інших сферах. Граф – це система, що складається з множини вершин та ребер, які з'єднують пари вершин. За допомогою графів можна моделювати відносини та зв'язки між об'єктами, що є дуже важливим для аналізу різноманітних систем, у яких важливими є зв'язки та їхні властивості.

Означення 1.1. Графом G називається пара (V, E), де V – множина вершин, а E – множина ребер.

Вершини графа часто позначаються символами $v_1, v_2, \dots, v_n$, а ребра – парами вершин, які вони з'єднують, наприклад, $e = (v_i, v_j)$.

Означення 1.2. Орієнтованим графом або оргарафом називається граф, у якому кожне ребро має свій напрямок, $D = (V, E)$, де V – множина вершин, $E \subseteq V \times V$ – множина орієнтованих ребер.

Розглянемо основні поняття теорії графів.

Вершина – це основний елемент графа, який представляє окремий об'єкт або точку. Множина вершин позначається через V, її елементи називають вершинами.

Ребро – це елемент графа, який з'єднує пару вершин. Множина ребер позначається через E, і кожне ребро з'єднує дві вершини.

У неорієнтованих графах ребра не мають напрямку, тоді як у орієнтованих графах ребра мають певний напрямок від однієї вершини до іншої, і називаються орієнтованими ребрами (дугами).

Степінь вершини – це кількість ребер, що з'єднують дану вершину з іншими вершинами.

Вершина називається *непарною*, якщо її степінь непарний, *парною* − якщо степінь парний.

У орієнтованих графах розрізняють вхідний та вихідний степінь, залежно від напрямку ребер. Степінь входу – кількість дуг, які входять у вершину, степінь виходу – виходять з вершини.

Позначається степінь вершини $\deg(A)$.

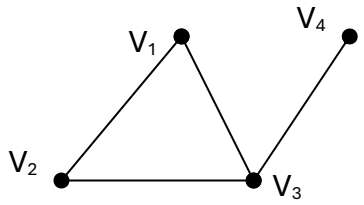


Рис. 1.1. Діаграма графа

Так, наприклад, на рис.1.1 зображено неорієнтований граф, степені якого $\deg(V_1) = 2$, $\deg(V_2) = 2$, $\deg(V_3) = 3$, $\deg(V_4) = 1$.

Вершини *A* та *B* називаються *суміжними*, якщо існує ребро, кінцями якого є ці вершини.

Говорять, вершини *A* та *B інцидентні* ребру *AB*. Ці вершини називають *кінцями* ребра *AB*.

Два різні ребра графа називаються *суміжними*, якщо вони мають хоча б одну спільну вершину.

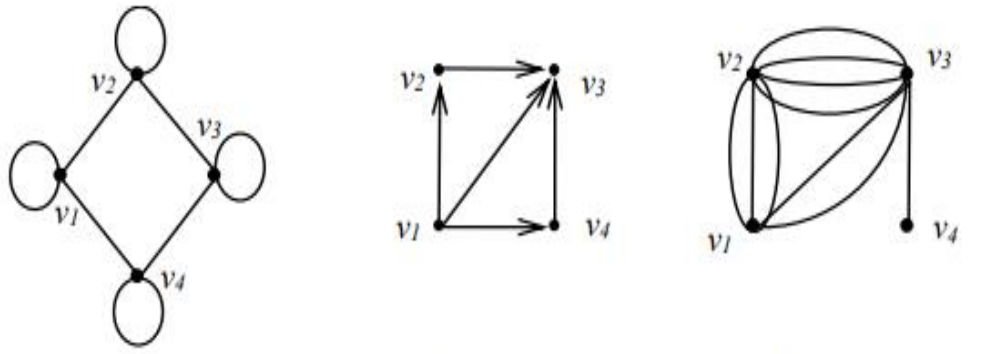
Так, наприклад, на рис.1.1 вершини V_1 та V_2 є суміжними, а вершини V_1 та V_4 не є суміжними.

Граф називається *простим*, якщо між будь-якою парою його вершин може існувати не більше одного ребра.

У випадку, коли хоча б одна пара вершин з'єднана більше ніж одним ребром, граф називають *мультиграфом*.

Ребра мультиграфа, які з'єднують одну і ту ж пару вершин, називаються *кратними.*

Ребро, яке з'єднує вершину саму із собою, називається *петлею.* Граф, який містить кратні ребра й петлі, називається *псевдографом.*



Простий неорієнтований граф з петлями G

Орієнтований граф D

Неорієнтований мультиграф M

Рис. 1.2. Приклади графів.

Теорема 1.1. *(Лема про рукостискання)*

Сума степенів усіх вершин графа завжди є парним числом, тобто $\sum_{v \in G} \deg(v) = 2m$, де m — кількість ребер графа.

Доведення

Оскільки кожне ребро має дві кінцеві вершини, то його додавання збільшує степінь кожної з цих вершин на 2. Таким чином, кожне ребро додає 2 одиниці до загальної суми степенів усіх вершин.

Отже, сума степенів вершин графа дорівнює подвоєному числу ребер, а отже, є парним числом.

Доведення завершено.

Теорема 1.2 У будь-якому графі кількість вершин з непарним степенем є парною.

Доведення

Доведемо від супротивного.

Припустимо, що в графі існує непарна кількість вершин з непарним степенем. Оскільки сума степенів вершин з парним степенем є парною, то загальна сума степенів всіх вершин графа (яка є сумою степенів вершин парного і непарного степеня) також буде непарною. Це суперечить теоремі 1.1, згідно з якою сума степенів усіх вершин графа повинна бути парною.

Таким чином, ми отримали протиріччя, і отже, кількість вершин з непарним степенем у будь-якому графі має бути парною.

Доведення завершено.

Приклад

Є 30 студентів, серед яких деякі знайомі між собою. Довести, що кількість студентів, які мають непарну кількість знайомих, парна [13].

Розв'язання

Поставимо у відповідність кожному студентові вершину графа і з'єднаємо ребрами попарно вершини, що відповідають студентам, які знайомі між собою.

Отримаємо граф із 30 вершинами. Сума всіх степенів вершин цього графа згідно леми про рукостискання – число парне. Звідси випливає, що непарних доданків у цій сумі мусить бути парна кількість.

Отже, кількість студентів, які мають непарну кількість знайомих, парна.

Вершина B графа називається досяжною з вершини A, якщо $B=A$ або існує маршрут, який сполучає вершини B та A.

Означення 1.3. Граф називається зв'язним, якщо всі його вершини досяжні одна з одної.

Якщо у графа існують недосяжні одна з одної вершини, то такий граф називається незв’язним.

Незв'язний граф розпадається на зв'язні компоненти, які не з’єднані між собою.

Орграф називається *сильно зв’язним*, якщо для довільних двох його вершин існує маршрут, який сполучає ці вершини.

Якщо в орграфі кожну дугу замінити ребром, то отримаємо неорієнтований граф, який називається асоційованим із даним графом.

Орграф називається *слабко зв’язним*, якщо асоційований із ним граф – зв’язний.

Компонентою зв’язності (сильної) графа G називається зв’язний (сильно зв’язний) підграф, який не є власним підграфом довільного іншого зв’язного (сильно зв’язного) підграфа графа G.

Граф називається *щільним*, якщо кількість його ребер близька до максимальної.

На противагу, граф з малою кількістю ребер називається *розрідженим*. Поняття «щільний граф» і «розріджений граф» відносні і залежить від контексту задачі.

Означення 1.4. Графи G_1і G_2 називаються ізоморфними, якщо існує спосіб нумерації їхніх вершин так, щоб з'єднання між вершинами у цих графах співпадали, тобто, кожне ребро графа G_1, що з'єднує дві певні вершини, має відповідне ребро у графі G_2 , яке з'єднує ті самі вершини, але з можливим іншим іменуванням. В такому випадку кажуть, що між графами G_1та G_2 існує ізоморфізм, або що ці графи є ізоморфними.

Властивості ізоморфних графів:

1) ізоморфні графи мають однакову кількість вершин;
2) ізоморфні графи мають однакову кількість ребер;
3) ізоморфні графи мають однакову кількість вершин певного степеня;
4) для неорієнтованих графів степені відповідних вершин збігаються;
   для орієнтованих – степені входу та виходу збігаються.

Ці властивості ізоморфних графів лише необхідні, але не є достатніми. Не існує набору інваріантів для визначення ізоморфізму графів.

Відношення ізоморфізму графів є відношенням еквівалентності. Тому множина усіх графів розбивається на класи ізоморфних графів.

Про графи, які належать до одного класу, говорять, що вони розрізняються з точністю до ізоморфізму, або вживають термін *абстрактний граф* [13].

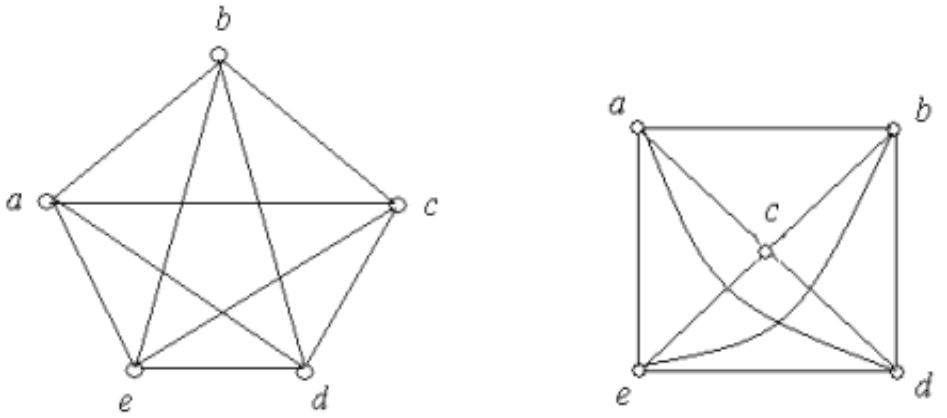


Рис. 1.3. Приклад двох ізоморфних графів.

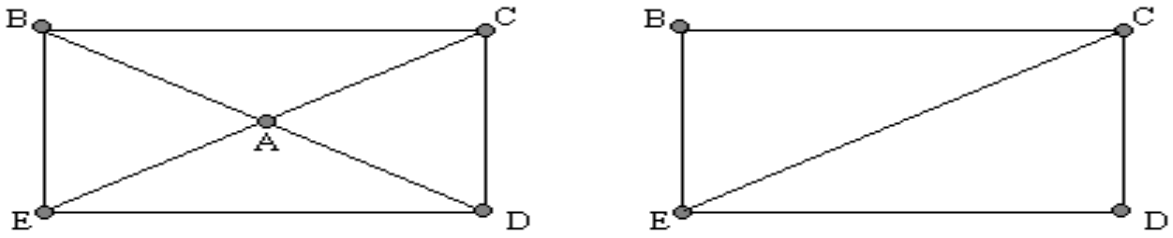


Рис. 1.4. Приклад двох неізоморфних графів.

На рисунку 1.3 зображено ізоморфні графи, бо між їх вершинами можна встановити взаємно однозначну відповідність, яка зберігає відношення суміжності.

На рисунку 1.4. зображено неізоморфні графи, бо у них різна кількість вершин, тобто не виконується необхідна умова ізоморфізму графів.

Означення 1.5. Граф називається циклічним, якщо він містить щонайменше один цикл – замкнений маршрут, який не проходить через жодну вершину більше

одного разу (за винятком початкової, яка є кінцевою). В іншому випадку граф має назву – ациклічний.

Означення 1.6. Дерево – це зв'язний граф, який не містить циклів.

Граф із однієї вершини також є деревом.

Якщо граф незв'язний, не містить циклів, то кожна його зв'язна компонента – дерево. Такий ациклічний граф називається лісом.

Дерева – це особливий і дуже важливий клас графів. Особлива роль дерев визначається як широким їхнім застосуванням у різних галузях науки і практики, так і тим особливим положенням, яке дерева займають у самій теорії графів. Останнє випливає з граничної простоти будови дерев.

Якщо усі ребра графа мають однакову довжину, рівну одиниці, то граф називається *ненавантаженим.* З допомогою ненавантажених графів описуються системи, які можуть переходити з одного стану в інший і всі стани однаково ймовірні.

Навантажений (зважений) граф – це граф, кожному ребру якого ставиться у відповідність довільне дійсне число (вага ребра). Наприклад, відстань між населеними пунктами, вартість проїзду тощо.

Основні поняття та визначення теорії графів є фундаментом для подальшого вивчення алгоритмів пошуку найкоротших шляхів, які відіграють ключову роль у розв'язанні широкого спектру практичних задач, де важливим є оптимальне використання ресурсів і мінімізація витрат.

1.2. Способи задання графів

Спосіб задання графів за допомогою рисунку на площині у вигляді точок і ліній, які ці точки з'єднують, є незручним. Найчастіше для задання графів і роботи з ними використовуються матриці.

Представлення графів у вигляді матриць є важливим для їх аналізу та ефективного зберігання. Кожен спосіб подання графа матрицею має свої переваги, що залежать від специфіки графа та завдань, які необхідно розв'язати.

Розглянемо основні способи представлення графів, зокрема матрицю суміжності, матрицю інцидентності та списком суміжності.

Означення 1.7. Матрицею суміжності графа називається квадратна матриця A розмірності $n \times n$, де n — кількість вершин у графі. Елемент матриці a_{ij} дорівнює 1, якщо між вершинами v_i і v_j існує ребро, і 0, якщо такого ребра немає.

Для неорієнтованих графів матриця суміжності є симетричною відносно головної діагоналі.

Матриця суміжності є зручною для швидкого отримання інформації про зв'язки між вершинами. Однак у графах з великою кількістю вершин і відносно малою кількістю ребер (розріджені графи) такий спосіб подання може бути неефективним з точки зору пам’яті.

Приклади

Матриці суміжності для графів, зображених на рис. 1.2. Це матриці четвертого порядку, бо графи мають по 4 вершини. Для простого неорієнтованого графа з петлями *G* матриця *A(G*) по головній діагоналі має одиниці, бо у кожної вершини графа є петля. Графи D та M не мають петель, тому по головній діагоналі у відповідних матриць – нулі.

Крім того, для неорієнтованого мультиграфа M нулі і одиниці у матриці суміжності *A(M*) замінені кратностями відповідних ребер:

$$A(G) = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \qquad A(D) = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$A(M) = \begin{pmatrix} 0 & 3 & 2 & 0 \\ 3 & 0 & 4 & 0 \\ 2 & 4 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Означення 1.8. Матрицею інцидентності називається матриця B розмірності $n \times m$, де n – кількість вершин, а m – кількість ребер у графі. Елемент матриці

b_{ij} дорівнює 1, якщо вершина v_i інцидентна ребру e_j (тобто вершина належить ребру), і 0, якщо не інцидентна.

Отже, матриця інцидентності графа – прямокутна бінарна матриця, в якій число рядків дорівнює числу вершин графа n, а число стовпців – числу ребер графа m.

У орієнтованих графах для відображення напрямку ребра використовують значення -1, якщо вершина кінець дуги і 1 , якщо – початок.

Приклади

Побудувати матрицю інцидентності для графів (Рис.1.5 та Рис.1.6)

а)

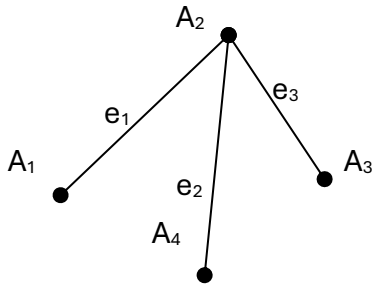


Рис.1.5

Розв'язання

Граф має чотири вершини і три ребра, тому матриця інцидентності графа матиме розмірність 4×3. Будуємо її:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

б)

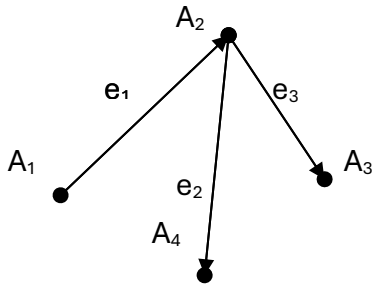


Рис.1.6.

Розв'язання

Граф орієнтований, він має чотири вершини і три ребра. Тому матриця інцидентності графа матиме розмірність 4×3. Будуємо її:

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

Матриця інцидентності широко використовується для аналізу структурованих графів та у випадках, коли потрібне відстеження напрямків ребер.

Степінь вершини можна визначити за допомогою матриць інцидентності та суміжності. Степінь вершини v_i відповідає кількості одиниць у i-му рядку матриці інцидентності або матриці суміжності.

Сума всіх степенів вершин дорівнює подвоєній кількості ребер, оскільки кожне ребро враховується в степенях двох вершин, тобто включається в суму двічі. Оскільки ця сума є парною, то й кількість вершин із непарним степенем завжди парна.

Граф називається однорідним степеня k, якщо степінь усіх його вершин дорівнює k.

Означення 1.9. Список суміжності – це таке представленя графа, у якому для кожної вершини зберігається список вершин, з якими вона пов'язана ребрами.

Списки суміжності є особливо ефективними для розріджених графів, оскільки зберігають інформацію лише про існуючі ребра.

При заданні графів у ЕОМ, список суміжності потребує менше пам'яті, ніж матриця суміжності, і дозволяє ефективно знаходити сусідні вершини, що робить його зручним для алгоритмів пошуку, таких як пошук у глибину і пошук у ширину.

Приклад

Розглянемо неорієнтований граф із трьома вершинами $V = \{v_1, v_2, v_3\}$ і трьома ребрами $E = \{e_1, e_2, e_3\}$, де:

• e_1 з'єднує v_1 і v_2 ,

• e_2 з'єднує v_2 і v_3 ,

• e_3 з'єднує v_1 і v_3 .

Задати граф матрицею суміжності, матрицею інцидентності та списком суміжності.

Розв'язання

Представлення цього графа можна здійснити матрицями суміжності та інцидентності і списком суміжності так:

- *матриця суміжності*:

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

- *матриця інцидентності:*

- $B = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$

- список суміжності:

$$v_1 : \{v_2, v_3\}, \qquad v_2 : \{v_1, v_3\}, \qquad v_3 : \{v_1, v_2\}$$

Теорема 1.3.

Кожен граф, представлений у вигляді матриці суміжності, може бути переведений у вигляд матриці інцидентності або списку суміжності і навпаки, зберігаючи повну інформацію про структуру графа.

Доведення.

Нехай дано граф $G = (V, E)$, представлений у вигляді матриці суміжності A. Для будь-якої пари вершин (v_i, v_j), якщо $a_{ij} = 1$, то вершини з'єднані ребром, яке може бути включене у відповідний список суміжності для кожної вершини або

позначене в матриці інцидентності. Таким чином, кожне представлення графа містить однакову інформацію про зв'язки між вершинами.

Доведення завершено.

Теорема 1.4.

Графи G_1 та G_2 ізоморфні тоді і тільки тоді, коли матрицю суміжності (матрицю інцидентності) одного з цих графів можна одержати з матриці суміжності (матриці інцидентності) іншого графа за допомогою відповідних перестановок рядків та стовпчиків.

Доведення.

Ізоморфні графи G_1 і G_2, які мають n вершин, відрізняються між собою лише порядком нумерації вершин, тобто існує бієкція φ, задана на множині вершин {1,2,…, n }, яка зберігає суміжність вершин.

Отже, кожен елемент a_{ij} ($i= 1,2,..., n, j= 1,2,..., n$) матриці суміжності A_1 графа G_1 збігається з елементом $a_{\varphi(i)\varphi(j)}$ (тобто елементом, який знаходиться в рядку з номером $\varphi(i)$ і стовпчику з номером $\varphi(j)$) матриці суміжності A_2 графа G_2.

Таким чином, методом послідовного одночасного обміну місцями (перестановок) рядків і стовпчиків згідно відповідності φ:

$$\begin{pmatrix} 1, & 2, & ..., & n \\ \varphi(1), & \varphi(2), & ..., & \varphi(n) \end{pmatrix}$$

матрицю суміжності A_1 можна перетворити у матрицю суміжності A_2 і, навпаки.

Для матриць інцидентності графів G_1 і G_2 справедливі аналогічні міркування. Відмінність полягає лише у тому, що коли графи G_1 і G_2 ізоморфні, то для їх множин вершин існує бієкція φ, а для множин ребер - інша бієкція ψ.

Доведення завершено.

Переваги та обмеження різних способів представлення графів у ЕОМ

1. Матриця суміжності забезпечує швидкий доступ до інформації про зв'язки між вершинами, що зручно для щільних графів, але може бути надлишковою для розріджених графів.

При цьому способі задання лише за один крок можна отримати відповідь на запитання: «Чи існує у графа ребро (дуга) $AiAj$?» Недолік способу задання – незалежно від кількості ребер (дуг) графа, об'єм пам'яті становить n^2 комірок.

2. Матриця інцидентності зручна для відстеження інцидентності вершин та ребер, але її використання не завжди доцільне для розріджених графів.

Спосіб задання графа у комп'ютері матрицею інцидентності вимагає nm комірок пам'яті, причому більшість з цих комірок зайнята нулями.

Незручним також є доступ до інформації. Відповідь на запитання : «Чи існує у графа ребро (дуга) A_iA_j ?» – вимагає, у гіршому випадку, перебору всіх стовпців матриці, тобто m кроків.

3. Список суміжності ефективний для зберігання розріджених графів і широко використовується у алгоритмах, які потребують частого звернення до суміжних вершин.

1.3. Маршрути та цикли у графах

У графах важливе місце займає дослідження маршрутів і циклів, оскільки вони дозволяють аналізувати зв'язність та структуру графів. Маршрути і цикли є основними елементами у вивченні графів, які дозволяють вирішувати задачі на шляхах, зв'язності та пошуку найкоротших маршрутів.

Означення 1.10. Маршрутом у графі називається послідовність вершин $v_1, v_2, \ldots, v_k$, у якій кожна пара суміжних вершин з'єднана ребром, тобто, для кожної пари (v_i, v_{i+1}) з послідовності існує ребро e $\in E$ графа G.

Маршрути можуть бути орієнтованими і неорієнтованими. В орієнтованому графі маршрут має враховувати напрямки ребер. У неорієнтованому графі маршрут не має враховувати напрямок зв'язків між вершинами.

Означення 1.11. Довжиною маршруту називається кількість ребер у цьому маршруті.

Для зважених графів довжина маршруту визначатися як сума ваг ребер, які входять до маршруту.

Приклад

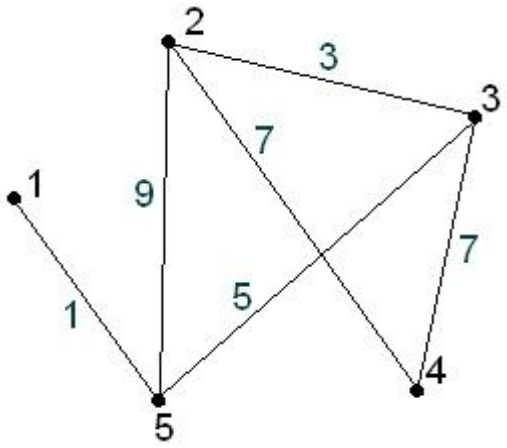


Рис.1.7

Знайти довжину маршруту 1-5-3-4 у зваженому графі (рис 1.7).

Розв'язання

Числа поряд із ребрами – це їх вага. Довжина шляху 1-5-3-4 дорівнює 1+5+7=13.

*Відповідь.*13

Приклад

Граф із вершинами $V = \{v_1, v_2, v_3, v_4\}$ і трьома ребрами $E = \{e_1, e_2, e_3, e_4\}$ заданий списком ребер:

- e_1 з'єднує v_1 і v_2,
- e_2 з'єднує v_2 і v_3,
- e_3 з'єднує v_3 і v_4,
- e_4 з'єднує v_4 і v_1.

Знайти довжини маршрутів $v_1 \to v_2 \to v_3$ та $v_1 \to v_2 \to v_3 \to v_4 \to v_1$

Розв'язання

Послідовність вершин $v_1 \to v_2 \to v_3$ є маршрутом довжини 2.

Послідовність вершин $v_1 \to v_2 \to v_3 \to v_4 \to v_1$ є циклом, який проходить усі вершини і повертається до початкової вершини. Його довжина 4.

Означення 1.12. Циклом у графі називається маршрут, у якому початкова і кінцева вершини співпадають, а всі інші вершини проходяться лише один раз.

Означення 1.13. Ейлерів цикл – це цикл, який проходить через кожне ребро графа рівно один раз і повертається до початкової вершини.

Граф, що містить такий цикл, називається ейлеровим.

Теорема 1.5. (Ейлера)

Граф має ейлерів цикл тоді і тільки тоді, коли він є зв'язним і кожна вершина в ньому має парний ступінь.

Доведення.

Необхідність. Нехай G – ейлерів граф.

Ейлерів обхід такого графа передбачає проходження через кожну його вершину, входячи в неї по одному ребру та виходячи через інше. Це означає, що кожна вершина інцидентна парному числу ребер ейлерового циклу, оскільки цей цикл включає всі ребра графа G.

Звідси випливає, що степінь кожної вершини є парним.

Достатність. Припустимо тепер, що степені всіх вершин графа G парні. Почнемо побудову ланцюга P_1 з довільної вершини v_1 (Рис.1.7) Продовжуватимемо ланцюг, кожного разу вибираючи нове ребро, поки не досягнемо вершини, з якої не залишиться невикористаних ребер. Оскільки степені вершин парні, цей ланцюг P_1 або завершиться в початковій вершині v_1 (утворивши цикл), або закінчиться в іншій вершині, але тоді також має бути продовження у вигляді циклу.

Якщо P_1 містить усі ребра графа G, то P_1 є необхідним ейлеровим циклом. В іншому випадку видаляємо всі ребра P_1 з графа G, отримуючи підграф G_1, у якому степені всіх вершин залишаються парними. Завдяки зв’язності графа G підграф G_1 також зв’язний, що дозволяє продовжити будувати ланцюги.

Далі, обираючи вершину v_2 , що належить G_1 і входить до побудованого раніше ланцюга P_1 , створимо цикл P_2 , аналогічно розвиваючи його в межах G_1 Об’єднаємо отримані цикли P_1 та P_2 , утворивши цикл $P_3 = P_1 \cup P_2$, який містить всі ребра обох ланцюгів.

Якщо об'єднаний цикл P_3 ще не є ейлеровим, то процес повторюється до тих пір, поки не буде охоплено всі ребра графа G, що завершує побудову ейлерового циклу.

Таким чином, необхідні та достатні умови для існування ейлерового циклу в графі виконуються.

Доведення завершено.

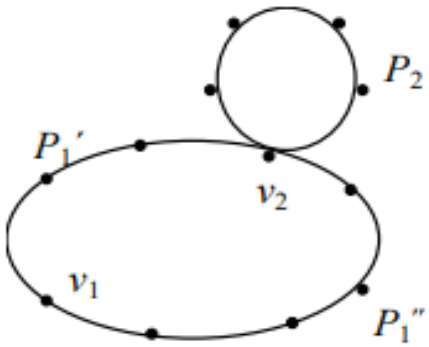
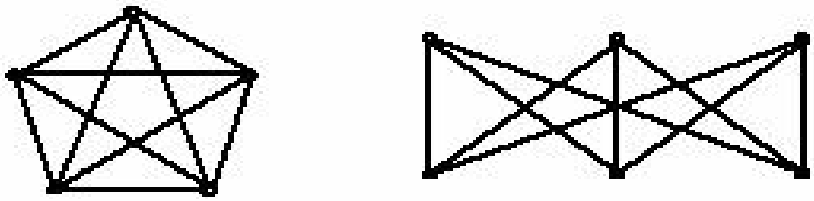


Рис. 1.8. Граф до доведення теореми Ейлера.

Приклад

Чи мають графи K_5 та $K_{3,3}$ (Рис.1.9) ейлерові цикли:



K_5 $K_{3,3}$

Рис.1.9

Розв'язання

Граф K_5 має п'ять вершин. Усі вони мають парний степінь. Тому у графі є ейлеровий цикл.

Граф $K_{3,3}$ має шість вершин. Усі вони мають непарний степінь. У графі немає ейлерового циклу.

Відповідь. $К_5$ – має ейлеровий цикл; $К_{3,3}$ – немає ейлерового циклу.

Існує дуже зручний, алгоритм побудови ейлеревого циклу – алгоритм Флері.

Алгоритм Флері

1. Вибираємо довільну вершину V_0. Йдемо по ребру, інцидентному цій вершині. Прийшли у вершину V_1 і викреслили ребро k_1.
2. Якщо на k-му кроці прийшли у вершину V_k, то для наступного ходу вибирають будь-яке ребро, інцидентне цій вершині. При цьому ребро-міст вибирають лише тоді, коли інших можливостей немає.

Приклад

Знайти ейлеровий цикл у графі (рис.1.10)

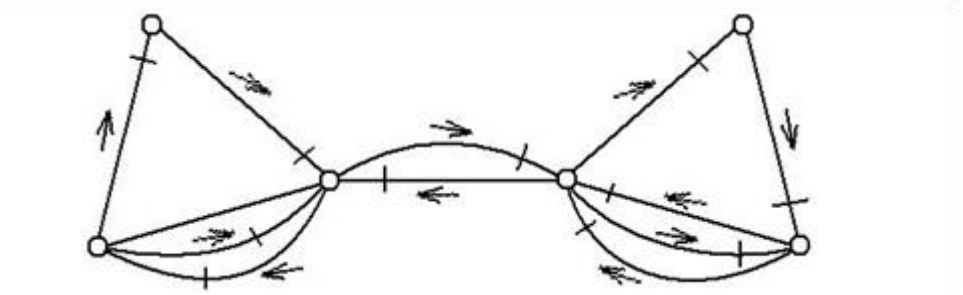


Рис. 1.10

Розв'язання

Оскільки усі вершини даного графа парні, то граф має ейлерів цикл. Застосовуючи алгоритм Флері, отримуємо один із можливих ейлеревих циклів: *А – Е – Д – С – Д – В – С – Д – Е – М – Е – М – А.*

Відповідь: А – Е – Д– С – Д – В – С – Д – Е – М – Е – М – А.

Означення1.14. Гамільтонів шлях (гамільтоновий ланцюг) – це простий ланцюг, що проходить через всі вершини графа, починаючи і закінчуючи в різних вершинах $v', v'' \in G$.

Означення1.15. Гамільтонів цикл – це простий цикл, що проходить через всі вершини графа G.

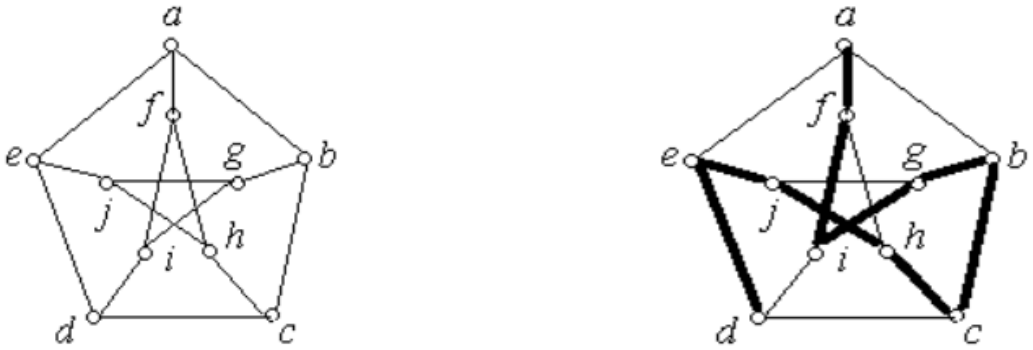
Теорема 1.6. Для кожної вершини, яка належить гамільтоновому циклу, існують рівно два ребра, що інцидентні цій вершині.

Означення 1.16. Граф, що має гамільтонів цикл, називається гамільтоновим.

На основі цього визначення та теореми 1.6 можна зробити висновок, що будь-який граф, у якому кожна вершина має степінь не менше двох, може бути гамільтоновим. Однак, для того, щоб граф мав гамільтонів цикл, необхідно, щоб він був зв'язним.

Приклад

На рисунку 1.11 а) зображено однорідний граф, степеня 3. Він називається графом Петерсена.



а) б)

Рис. 1.11

Цей граф має гамільтонів шлях (рис. 1.11, б), але не містить гамільтонового циклу.

Гамільтонові та ейлерові цикли подібні за способом задання. Проте за складністю відшукання ці задачі дуже різняться.

Пошук критерію існування гамільтонового графа – одна з невирішених проблем теорії графів. Відомі лише достатні умови існування гамільтонових графів.

Теорема (О. Оре, 1960 р.). *Якщо для довільної пари несуміжних вершин V_i та V_j графа G, який має не менше трьох вершин, виконується умова , що сума їх степенів більша рівна за n, де n – кількість вершин графа, то граф гамільтоновий.*

Теорема (Г. Дірака, 1957 р.). *Якщо число вершин графа не менше трьох і для кожної вершини графа степінь вершини більший рівнй за* $\frac{n}{2}$*, то граф гамільтоновий.*

Один з методів пошуку гамільтонового циклу – метод перебору. Для цього нумерують усі вершини та розглядають усі перестановки, вивчаючи, чи утворюють вони гамільтоновий цикл.

Якщо граф з n вершинами не гамільтоновий, то потрібно перебрати (n-1)! варіантів**.**

На практиці користуються алгоритмами часткового перебору. Проте такі алгоритми, як правило, мають високу складність.

Висновки до розділу 1

Теорія графів розпочала свій розвиток з праць Леонарда Ейлера, який у 1736 році знайшов критерій існування спеціального маршруту (ейлеревого циклу).

Інтенсивний розвиток теорія графів отримала лише 50-60 років тому. Цей розвиток пов'язаний із широким застосуванням теорії графів. Граф стає однією з найзастосовніших математичних моделей, адже за допомогою графів легко візуально ілюструювати дані та відношеня між ними. Чіткість та коректність опису даних у графах забезпечують ефективність використання графів.

У теоретико-графових термінах формулюється багато задач, які пов’язані з дискретними об’єктами. Такі задачі виникають при проектуванні схем управління, досліджені автоматів, логічних ланцюгів, блок-схем програм тощо.

Розгляд маршрутів та циклів у графах є ключовим для розуміння структури графів та вирішення багатьох практичних задач, пов'язаних з оптимізацією маршрутів і виявленням зв'язностей у системах.

Ейлерові та гамільтонові цикли надають інструменти для аналізу шляхів у різних типах мереж, включаючи комунікаційні та транспортні системи.

РОЗДІЛ 2

АЛГОРИТМИ ЗНАХОДЖЕННЯ НАЙКОРОТШИХ МАРШРУТІВ У ГРАФАХ

2.1. Найбільш ефективні алгоритми знаходження найкоротшого маршруту

Знаходження найкоротших маршрутів у графах є важливим завданням у теорії графів, яке має численні практичні застосування в різних галузях: транспортних системах, зв'язку, логістиці, навігаційних системах тощо.

Існують різні підходи та методи для вирішення цього завдання, серед яких найбільш поширеними є алгоритм Дейкстри та алгоритм Флойда.

Означення 2.1. Найкоротшим шляхом (маршрутом) між двома вершинами графа називається маршрут з мінімальною сумарною довжиною, де довжина визначається як сума ваг ребер у маршруті (для зважених графів) або кількість ребер (для незважених графів).

Алгоритм Дейкстри

У 1959 році датський математик Е. Дейкстра запропонував алгоритм визначення довжини найкоротшого маршруту від фіксованої вершини до будь-якої іншої.

Алгоритм Дейкстри є одним із найпоширеніших методів для знаходження найкоротшого шляху від однієї вершини до всіх інших у графі з додатними вагами ребер. Він використовує підхід при якому, поступово розширюється область досяжності, щоб знайти мінімальні відстані до кожної вершини, починаючи з початкової. Алгоритм працює, поступово обираючи вершину з найменшою відстанню від початкової та оновлюючи відстані до сусідніх вершин, якщо знайдено коротший шлях.

Алгоритм Флойда

Алгоритм Флойда-Воршелла призначений для знаходження найкоротших шляхів між усіма парами вершин у графі. Він базується на методі динамічного

програмування та поступово покращує відстані між вершинами за рахунок розгляду проміжних вершин. Алгоритм є універсальним і дозволяє працювати з графами, де можуть існувати від'ємні ваги, за умови відсутності циклів з від’ємною сумою ваг.

Порівняння та застосування алгоритмів

1. *Алгоритм Дейкстри* застосовується для пошуку найкоротших маршрутів від однієї вершини до інших у графі з додатними вагами. Алгоритм особливо зручний для реальних завдань у транспортних мережах та навігаційних системах.
2. *Алгоритм Флойда-Воршелла* є більш універсальним і дозволяє обчислювати найкоротші шляхи між усіма парами вершин, що є корисним для задач на графах із довільними вагами, зокрема для аналізу комунікаційних мереж.

2.2 Визначення найкоротшого шляху в алгоритмі Дейкстри.

Основний принцип роботи алгоритму Дейкстри

Алгоритм Дейкстри працює за рахунок поступового розширення області досяжності, починаючи з початкової вершини. Кожній вершині графа призначається мітка (відстань), яка спочатку дорівнює нескінченності, за винятком початкової вершини, мітка якої встановлюється на нуль. На кожному кроці алгоритм вибирає вершину з найменшою міткою з множини невідвіданих, оновлює значення міток сусідніх вершин, якщо шлях через поточну вершину є коротшим, і позначає обрану вершину як оброблену.

Опис алгоритму Дейкстри

Етап 1. Знаходження довжини найкоротшого шляху.

Крок 1. Присвоєння вершинам початкових міток.

Покладемо $d(s) = 0^*$ і вважаємо цю мітку постійною (постійні мітки помічаємо зверху зірочкою). Для решти вершин $x_i \in V, x_i \neq s$, і покладемо $d(x_i) = \infty$ і вважаємо ці мітки тимчасовими. Нехай $\tilde{x} = s, \tilde{x}$ – позначення поточної вершини.

Крок 2. Зміна міток.

Для кожної вершини x_i з тимчасовою міткою, що безпосередньо слідує за вершиною $\tilde{x}$, змінюємо її мітку у відповідності з наступним правилом:

$$d_{\text{нов.}}(x_i) = \min\{d_{\text{стар.}}(x_i); d(\tilde{x}) + \omega(\tilde{x}; x_i)\} \ ,$$

Крок 3. Перетворення тимчасової мітки в постійну.

Із всіх вершин з тимчасовими мітками вибираємо вершину x_j^* з найменшим значенням мітки

$d(x_j^*) = \min\{d(x_j)\}, x_j \in V, d(x_j)$ – тимчасова.

Перетворюємо цю мітку в постійну і покладемо $\tilde{x} = x_j^*$.

Крок 4. Перевірка на завершення першого етапу.

Якщо $\tilde{x} = t^*$ то $d(\tilde{x})$ – довжина найкоротшого шляху від s до t . В протилежному випадку повернення до кроку 2.

Етап 2. Побудова найкоротшого шляху.

Крок 5. Послідовний пошук дуг найкоротшого шляху.

Серед вершин, що передують вершині $\tilde{x}$ з постійними мітками, знаходимо вершину x_i , яка задовольняє рівність

$$d(\tilde{x}) = d(x_i) + \omega(x_i; \tilde{x})$$

Включаємо дугу $(x_i; \tilde{x})$ в шуканий шлях і покладемо $\tilde{x} = x_i$

Крок 6. Перевірка на завершення другого етапу.

Якщо $\tilde{x} = s$, то найкоротший шлях знайдено – його утворює послідовність дуг, отриманих на п’ятому кроці і записаних в оберненому порядку. В протилежному випадку повертаємося до п’ятого кроку. [3]

Приклад (Виконання алгоритму Дейкстри)

Граф представлений вершинами A, B, C, D, E та з'єднаний ребрами з відповідними вагами:

A→B вага 3,

A→C вага 1,

B→C вага 7,

B→D вага 5,

C→D вага 2,

D→E вага 7,

C→E вага 6.

Знайдемо найкоротші шляхи від вершини A до всіх інших вершин, використовуючи алгоритм Дейкстри.

Початкова ініціалізація:

- Призначаємо мітки вершинам:

$$A = 0, B = \infty, C = \infty, D = \infty, E = \infty$$

- Початкова вершина – A з міткою 0.

Крок 1: Вибір вершини A

- Оновлюємо мітки для сусідніх вершин B та C:
  - B=3 (через A),
  - C=1 (через A).
- Позначаємо вершину A як відвідану.

Крок 2: Вибір вершини C (мінімальна мітка 1)

- Оновлюємо мітки для сусідніх вершин B, D, E:
  - B залишається 3, оскільки 1+7=8, що більше від поточної мітки 3.
  - D= 3 (через C, 1+2=3),
  - E=7 (через C, 1+6=7).
- Позначаємо вершину C як відвідану.

Крок 3: Вибір вершини B (мінімальна мітка 3)

- Оновлюємо мітку для сусідньої вершини D:
  - D залишається 3, оскільки 3+5=8, що більше від поточної мітки 3.
- Позначаємо вершину B як відвідану.

Крок 4: Вибір вершини D (мінімальна мітка 3)

- Оновлюємо мітку для вершини E:
  - E залишається 7, оскільки 3+7=10, що більше від поточної мітки 7.

- Позначаємо вершину D як відвідану.

Крок 5: Вибір вершини E (мінімальна мітка 7)

- Всі сусідні вершини вже оброблені, алгоритм завершується.

Результати

Найкоротші шляхи від вершини A до інших вершин:

- A → B = 3
- A → C = 1
- A → D = 3
- A → E = 7

Переваги та недоліки алгоритму Дейкстри

1. Переваги:
   - Ефективність у графах з додатними вагами.
   - Можливість швидкого обчислення відстаней від однієї вершини до всіх інших, що робить його корисним для задач маршрутизації.
2. Недоліки:
   - Не підходить для графів із від'ємними вагами, оскільки такий підхід може призвести до некоректного результату.

2.3 Визначення найкоротшого шляху в алгоритмі Флойда-Воршелла

Алгоритм Флойда-Воршелла призначений для знаходження найкоротших відстаней між усіма парами вершин у зваженому орієнтованому графі. Цей алгоритм був розроблений у 1962 році Робертом Флойдом і Стівеном Воршеллом. Він дозволяє працювати з графами, що містять ребра з від’ємними вагами, за умови відсутності циклів із від'ємною вагою.

Нехай вершини графа $G = (V, E)$ пронумеровані від 1 до n. Для позначення довжини найкоротшого шляху між вершинами i та j, який проходить через вершини, пронумеровані від 1 до k, використовується значення d_{ij}^{k}. Очевидно,

що d_{ij}^0 – довжина (вага) ребра (i, j), якщо воно існує (в іншому разі його довжина може бути позначена як ∞).

Існує два варіанти значення $d_{ij}^k, k \in (1, \dots, n)$: .

- Найкоротший шлях між i, j не проходить через вершину k , тоді
$$d_{ij}^k = d_{ij}^{k-1}$$
- Існує більш короткий шлях між i та j , що проходить через k, який спочатку йде від i до k, а потім від k до j. У цьому випадку, очевидно $d_{ij}^k = d_{ik}^{k-1} + d_{kj}^{k-1}$

Таким чином, для знаходження значення функції досить вибрати мінімум з двох позначених значень.

Тоді рекурентна формула для d_{ij}^k має вигляд:

$$d_{ij}^k = \min\left(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\right),$$

d_{ij}^0 – довжина ребра(i, j).

Алгоритм Флойда-Воршелла обчислює значення для всіх пар вершин d_{ij}^k, поступово збільшуючи значення k від 1 до n. На кожному кроці формується нова матриця відстаней W, яка містить поточні значення найкоротших шляхів між усіма парами вершин, з урахуванням проміжних вершин.

Після завершення роботи алгоритму значення d_{ij}^n для кожної пари вершин є довжиною найкоротшого шляху між цими вершинами.

Результати обчислень зберігаються в матриці W, яка відображає всі найкоротші відстані в графі. [3]

Приклад (Виконання роботи алгоритму Флойда)

Розглянемо приклад графа з вершинами $V = \{A, B, C, D\}$, та зваженими ребрами між ними, ваги яких такі: $w(A, B) = 3,\ w(A, C) = 10,\ w(B, C) = 2,$ $w(C, D) = 1,\ w(B, D) = 5.$

1. На початку створюється матриця відстаней D:

$$D = \begin{pmatrix} 0 & 3 & 10 & \infty \\ 3 & 0 & 2 & 5 \\ 10 & 2 & 0 & 1 \\ \infty & 5 & 1 & 0 \end{pmatrix}$$

2. Далі алгоритм ітеративно перевіряє, чи існує шлях між кожною парою вершин через інші вершини, і поступово оновлює матрицю відстаней. Після врахування всіх проміжних вершин отримуємо остаточну матрицю відстаней:

$$D = \begin{pmatrix} 0 & 3 & 5 & 6 \\ 3 & 0 & 2 & 3 \\ 5 & 2 & 0 & 1 \\ 6 & 3 & 1 & 0 \end{pmatrix}$$

• Після завершення роботи алгоритму в матриці D значення $D[i][j]$ містять найкоротші відстані між кожною парою вершин i та j. Наприклад, найкоротша відстань між A та D дорівнює 6 [14; 18].

Алгоритми Дейкстри та Флойда-Воршелла знаходять широке застосування в таких галузях, як:

• Транспортні системи: для оптимізації маршрутів у дорожніх, авіаційних та інших мережах, де важливо мінімізувати відстань, час або витрати між пунктами призначення.

• Мережеві комунікації: для забезпечення оптимальних шляхів передачі даних між вузлами в мережах, що є критично важливим для скорочення затримок і підвищення продуктивності систем.

• Логістика та управління ланцюгами поставок: для знаходження найкоротших шляхів транспортування товарів між складськими та розподільчими пунктами, що допомагає мінімізувати витрати на перевезення.

Висновки до розділу 2

Графи дозволяють перевести словесне формулювання задачі в наочну модель. Така візуальна ілюстрація даних та відношень між ними за допомогою графа значно полегшує процес міркування і пошуку розв’язку практичної задачі.

Знаходження найкоротших маршрутів у графах є важливим завданням у теорії графів, яке має численні практичні застосування в різних галузях: транспортних системах, зв’язку, логістиці, навігаційних системах тощо.

У розділі досліджено два основних алгоритми для знаходження найкоротших шляхів: алгоритм Дейкстри та алгоритм Флойда-Воршелла. Акцентовано увагу на принципах роботи алгоритмів, умовах застосування, перевагах, обмеженнях та використанні.

ВИСНОВКИ

У даній магістерській роботі на основі аналізу літературних джерел розкрито основні поняття теорії графів, алгоритми знаходження найкоротших шляхів, а також їх практичне застосування в різних галузях.

У першому розділі роботи було наведено основні визначення теорії графів, зокрема означено різні типи графів, такі як орієнтовані та неорієнтовані, зважені та незважені. Було обґрунтовано важливість графів як математичної моделі для аналізу та моделювання складних систем.

Розгляд властивостей графів, таких як зв'язність, наявність циклів та мінімальні маршрути, надав основу для подальшого розуміння алгоритмів пошуку шляхів. Крім того, дослідження показало, що поняття графів дозволяє чітко моделювати зв'язки та взаємодії між об'єктами в різних системах, роблячи аналіз зручнішим та ефективнішим.

У другому розділі було досліджено два основних алгоритми для знаходження найкоротших шляхів: алгоритм Дейкстри та алгоритм Флойда-Воршелла. Кожен із цих алгоритмів було розглянуто окремо, з акцентом на принципах їхньої роботи, умовах застосування, перевагах, обмеженнях та прикладах використання.

Алгоритм Дейкстри був описаний як ефективний метод для обчислення найкоротших шляхів від однієї вершини до всіх інших у графі з додатними вагами ребер.

Алгоритм Дейкстри забезпечує оптимальний шлях від початкової вершини до кожної з інших, обираючи на кожному кроці вершину з мінімальною міткою та оновлюючи значення міток для сусідніх вершин. У розділі було наведено приклад графа з реальними вагами ребер, який ілюструє кожен крок алгоритму та показує, як значення відстаней поступово оновлюються до оптимальних.

Були вказані обмеження алгоритму Дейкстри, такі як його неспроможність працювати з графами з від'ємними вагами ребер, що є важливим аспектом для застосування алгоритму в практиці.

Алгоритм Флойда-Воршелла було розглянуто як універсальний підхід для знаходження найкоротших шляхів між усіма парами вершин у графі. Цей алгоритм використовує метод динамічного програмування та рекурентні відношення для поступового скорочення відстаней між вершинами.

Було наведено рекурентну формулу, яка забезпечує вибір мінімальної відстані між вершинами на кожному етапі, дозволяючи поступово оновлювати матрицю відстаней. Подано конкретний приклад, що ілюструє кроки алгоритму та результативність його застосування.

Алгоритм Флойда-Воршелла є особливо корисним для графів із від'ємними вагами, де він забезпечує оптимальне рішення, якщо у графі немає циклів з від'ємною вагою. Зазначено, що цей алгоритм є більш обчислювально затратним порівняно з алгоритмом Дейкстри, однак його перевага полягає в можливості обчислення найкоротших шляхів для всіх пар вершин одночасно.

Отже, теорія графів та алгоритми пошуку найкоротших шляхів мають велике значення для багатьох сфер науки та техніки, дозволяючи розв'язувати складні оптимізаційні задачі, що виникають у реальних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Балога С.І. Дискретна математика. Навчальний посібник. Ужгород : ПП «АУТДОРШАРК», 2021. 124 с.

2. Денисова Т. В., Сенчуков В. Ф. Дискретна математика [Електронний ресурс]: навч. посібник / ХНЕУ ім. С. Кузнеця. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 288 с. ISBN 978-966-676-751-9. http://repository.hneu.edu.ua/handle/123456789/22003.

3. Дискретна математика. Навчальний посібник. – Ужгород: ПП «АУТДОРШАРК», 2021. – 124 с.

4. Капітонова Ю.В., Кривий С.Л., Летичевський О.А., Луцький Г.М., Печурін М.К. Основи дискретної математики. Київ: Наукова думка, 2002. 567 с.

5. Коцовський В. М. Основи дискретної математики: навч. посібник. – Ужгород: Рік-У, 2020. – 123 с.

6. Коноваленко О. Є., Ткачук М. А., Грабовський А. В. Дискретна математика: навч.-метод. посібник / НТУ «ХПІ». – Харків : НТУ «ХПІ», 2016. – 84 с.

7. Ліхоузова Т. А. Дискретна математика. Практикум [Електронний ресурс]: навч. посіб. для студ. спец. 121 «Інженерія програмного забезпечення», 126 «Інформаційні системи та технології» / КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2,7 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 62 с. https://core.ac.uk/download/pdf/323530137.pdf.

8. Матвієнко М. П. Дискретна математика. Київ: Ліра-К, 2019. 324 с.

9. Молода наука Волині: пріоритети та перспективи досліджень. Луцьк: Луцьк: ВНУ ім. Лесі Українки, 2024. –с. 343–345. Молода наука Волині_2024 готовий файл (1)-1

10. Нікольський Ю.В., Пасічник В.В., Щербина Ю.М. Дискретна математика. Львів: Магнолія, 2024. 432 с.

11. Порубльов І. М. Дискретна математика: навч. посібник для студ. 1-го курсу бакалаврату галузі знань «Інформаційні технології». – Черкаси: видавець

ФОП Гордієнко Є.І., 2018. – 220 с. http://eprints.cdu.edu.ua/4177/1/paryblev2018%20%281%29.pdf.

12. Темнікова О. Л. Дискретна математика: Конспект лекцій (Частина 1) [Електронний ресурс]: навч. посіб. для студ. спец. 113 «Прикладна математика», освітньої програми «Наука про дані та математичне моделювання» / КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2,97 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 154 с. https://ela.kpi.ua/bitstream/123456789/42839/1/LectureDM1Temnikova.pdf.

13. Швай О.Л. Практикум із дискретної математики: навч. посіб. 2-ге вид., переробл. і допов. Луцьк: Волин. нац. ун-т ім. Лесі Українки, 2020. 236 с. Гриф «Рекомендовано до друку вченою радою Волинського національного університету імені Лесі Українки» (Протокол №14 від 26.11.2020 р.).

14. Шевченко Г.В., Шкапа В.В. Дискретна математика. Навчально-методичний посібник. Київ : ДУТ, 2018. 158 с.

ДОДАТКИ

ДОДАТОК А

Історичні аспекти становлення теорії графів

1) Ранні приклади структур графів у стародавніх цивілізаціях

Хоча формалізована теорія графів з’явилася лише у XVIII столітті, ідеї, що нагадують графи, існували значно раніше. Наприклад, у стародавньому Китаї при створенні маршруту по річках або шляхів через гори використовували схеми, схожі на графи.

У середньовіччі, коли зростала потреба у створенні карт і навігації, можна зустріти приклади використання прототипів графів для моделювання шляхів між містами та побудови торговельних маршрутів. Ці неформальні підходи підготували ґрунт для майбутнього математичного формалізму.

2) Ейлер і його внесок у початок теорії графів

У 1736 році швейцарський математик Леонард Ейлер вирішив знамениту задачу про Кенігсберзькі мости. Ця проблема стала першим випадком застосування математичного підходу до моделювання об’єктів у вигляді графів. Ейлер розробив поняття вершини та ребра, які тепер є основою теорії графів. Найважливішим результатом стало створення концепції графової зв’язності, тобто ідеї про можливість переходу між вершинами за допомогою ребер.

3) Розвиток після Ейлера: XVIII століття

Після роботи Ейлера інтерес до графів залишався обмеженим. Проте його ідеї почали застосовувати у різних задачах. Наприклад, у кінці XVIII століття графи використовувалися для моделювання логічних схем, що стали прототипами сучасних мереж.

4) Гамільтон і "Гамільтонові цикли"

У середині XIX століття ірландський математик Вільям Роуен Гамільтон зробив значний внесок у розвиток теорії графів. Його робота зосереджувалася на задачі знаходження циклів, що проходять через кожну вершину графа один раз, що нині називається гамільтоновими циклами.

Одним із прикладів використання таких графів була гра, відома як "Ікосаедровий шлях Гамільтона", яка популяризувала графову теорію серед широкої аудиторії.

5) Густав Кірхгоф: графи в електричних мережах

У 1845 році німецький фізик Густав Кірхгоф використав графи для аналізу електричних кіл. Він розробив методику обчислення струмів у складних мережах за допомогою дерев графа, що стало першим прикладом практичного застосування графів у фізиці.

6) Теорема про розфарбовування графів

Наприкінці XIX століття з’явилася ще одна важлива задача – розфарбовування графів. У 1852 році Френсіс Гатрі запропонував теорему про чотири кольори: чи можливо розфарбувати карту так, щоб жодні дві суміжні області не мали одного кольору, використовуючи лише чотири кольори. Ця проблема залишалася невирішеною понад століття, але стала потужним стимулом для розвитку теорії графів.

7) Розвиток комбінаторики та вплив Кеніга

У 1936 році угорський математик Денеш Кеніг опублікував свою книгу "Теорія графів", що стала першим фундаментальним підручником з цієї дисципліни. У своїй роботі він ввів поняття ізоморфізму графів, планарності, розфарбовування та інших основних концепцій.

8) Алгоритми на графах: перші комп’ютерні додатки

У середині XX століття розвиток інформатики суттєво вплинув на теорію графів. Зокрема, з’явилися алгоритми для знаходження найкоротших шляхів (алгоритми Дейкстри, Флойда-Воршелла) і мінімальних остовів (алгоритм Пріма, алгоритм Крускала).

9) Комп’ютерне доведення теореми про чотири кольори

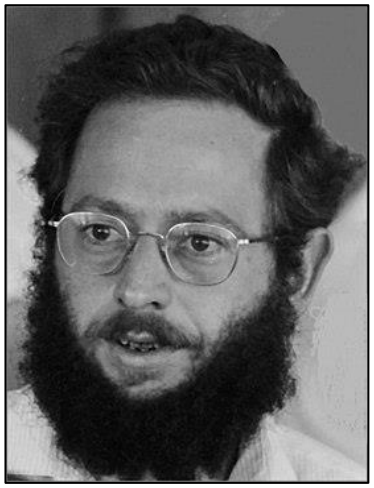
У 1976 році американські математики Кеннет Аппель і Вольфганг Хакен вперше довели теорему про чотири кольори за допомогою комп’ютера. Це стало новаторським кроком, що відкрив нові горизонти у використанні комп’ютерів для доказів математичних тверджень.

ДОДАТОК В

Бібліографічні відомості про вчених

Едсгер Дейкстра (1930–2002) – видатний нідерландський вчений, один із піонерів інформатики, який зробив значний внесок у розвиток теорії алгоритмів і структур даних. Його алгоритм для знаходження найкоротших шляхів у графах, відомий як алгоритм Дейкстри, широко використовується у різних сферах, зокрема в транспортних системах, телекомунікаційних мережах та навігаційних програмах.

Дейкстра був також відомий своєю роботою над принципами структурного програмування, що допомогло покращити якість програмного забезпечення. Він вважав, що програмування – це не просто написання коду, а справжня інженерна дисципліна. Дейкстра впровадив концепцію “Go To Statement Considered Harmful”, яка закликала програмістів уникати неконтрольованих переходів у коді для поліпшення його структури та читабельності.



Роберт Флойд (1936–2001) – американський інформатик, який зробив значний внесок у теорію алгоритмів, розробивши методи, що стали основою сучасної обчислювальної математики. Він є співавтором алгоритму Флойда-Воршелла, який дозволяє знайти найкоротші шляхи між усіма парами

вершин у графі. Цей алгоритм використовується в мережевому плануванні, маршрутизації даних і оптимізації витрат.

Окрім цього, Флойд відомий своїми роботами з формальної перевірки програмного забезпечення та визначенням правильності алгоритмів. Він розробив методи, які використовуються для доказу коректності програм, що особливо актуально для критичних систем, де навіть невелика помилка може призвести до серйозних наслідків.

Стівен Воршелл (народився у 1935 році) – американський математик і інформатик, який разом із Робертом Флойдом розробив алгоритм Флойда-Воршелла, що є одним із ключових методів для знаходження найкоротших шляхів між усіма парами вершин у графі. Цей алгоритм застосовується у багатьох сферах, включаючи мережеве планування, маршрутизацію в інтернеті та управління логістикою.

Алгоритм Флойда-Воршелла базується на методі динамічного програмування, що дозволяє ефективно знаходити найкоротші шляхи навіть у великих графах з багатьма вершинами. Воршелл вніс важливий вклад у розвиток теорії алгоритмів, забезпечивши нові способи оптимізації, які суттєво підвищують швидкодію обчислювальних процесів.

Окрім роботи над алгоритмом, Воршелл також займався іншими проблемами обчислювальної математики, приділяючи особливу увагу оптимізації та теоретичним основам штучного інтелекту, що робить його одним із провідних науковців у галузі комп'ютерних наук.

Анотація

Давидюк Н. В. Знаходження найкоротших маршрутів у графах. Магістерська робота. Луцьк, 2024. 41 с.

У магістерській роботі досліджується проблема знаходження найкоротших маршрутів у графах. Розглянуто основні алгоритми, зокрема алгоритм Дейкстри та алгоритм Флойда-Воршелла, їх застосування в різних мережевих структурах. Основний акцент зроблено на порівнянні ефективності цих алгоритмів.

Ключові слова: графи, найкоротші маршрути, алгоритм Дейкстри, алгоритм Флойда-Воршелла.

Annotation

Davydiuk N. V. Finding the Shortest Paths in Graphs. Master’s Thesis. Lutsk, 2024. 41 pages.

This master’s thesis addresses the problem of finding the shortest paths in graphs. Key algorithms, including Dijkstra's and Floyd-Warshall's algorithms, are examined for their application in various network structures. The research focuses on comparing the efficiency of these algorithms within practical contexts, particularly in transportation and communication networks. The findings contribute to route optimization and cost reduction in fields utilizing graphs.

Key words: graphs, shortest paths, Dijkstra's algorithm, Floyd-Warshall algorithm.