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

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра прикладної математики

Т.С. Кагадій, А.Г. Шпорта, О.Д. Онопрієнко

Основи дискретної математики
(частина 2)

Методичні рекомендації
до опанування лекційних занять
з дисципліни «Дискретна математика» здобувачами ступеня бакалавра
спеціальності 113 Прикладна математика

Дніпро
НТУ «ДП»
2024

Кагадій Т.С.

Основи дискретної математики (частина 2) [Електронний ресурс] : методичні рекомендації до опанування лекційних занять з дисципліни «Дискретна математика» здобувачами ступеня бакалавра спеціальності 113 Прикладна математика / Т.С. Кагадій, А.Г. Шпорта, О.Д. Онопрієнко ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2024. – 44 с.

Автори:
Т.С. Кагадій, д-р фіз.-мат. наук, проф.,
А.Г. Шпорта, канд. фіз.-мат. наук, доц.,
О.Д. Онопрієнко, PhD, доц.

Затверджено науково-методичною комісією спеціальності 113 Прикладна математика (протокол № 9-1/24 від 09.09.2024) за поданням кафедри прикладної математики (протокол № 9/24 від 02.09.2024).

Містить в скороченому вигляді теоретичні відомості з основних розділів дисципліни «Дискретна математика»: теорія графів та її застосування, деякі важливі алгоритми розв’язання задач теорії графів, означення та властивості дерев. Наведено вичерпні розв’язки великої кількості прикладів, запропоновано варіанти для самостійного розв’язування. Матеріал викладено докладно і послідовно, що дає змогу вивчати його самостійно або при дистанційній формі навчання.

Відповідальна за випуск завідувачка кафедри прикладної математики О.О. Сдвижкова, д-р техн. наук, проф.

Зміст

Вступ

Основні способи подання інформації у світі є дискретними: це слова і конструкції мов і граматик - природні і формалізовані; табличні масиви дійсних даних; відомості соціальної, демографічної, економічної статистики тощо.

Методи обробки та аналізу такої дискретної інформації вивчає дискретна математика.

Методична розробка підготовлена для студентів, які вивчають математичні моделі або методи інформатики та техніки.

В матеріалах видання містяться основні теми та поняття дискретної математики (теорія графів та її застосування, деякі важливі алгоритми розв'язання задач теорії графів, означення та властивості дерев). Всі розділи мають характер стислого конспекту лекцій і мають за мету спростити здобувачам сприйняття теоретичного матеріалу, особливо при дистанційному навчанні.

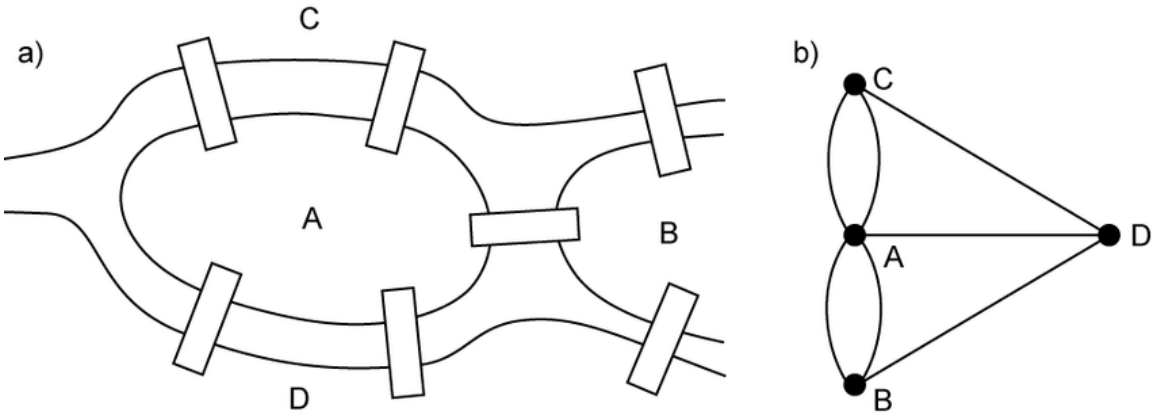
Для кращого засвоєння в тексті наведено велику кількість розв'язаних прикладів, підібрані завдання для самостійного опрацювання.

Наведено варіанти прикладних задач, для розв'язання яких використовуються методи та алгоритми дискретної математики, що свідчить про актуальність та необхідність вивчення даної дисципліни.

Методична розробка адаптована як для самостійної, так і для аудиторної роботи та містить інформацію, без якої неможливе вивчення курсу «Дискретна математика». Зміст кожної лекції логічно структурований і підпорядкований певній темі.

Розділ 1. *Теорія графів. Ключові визначення*

Виникнення теорії графів як математичної дисципліни прийнято пов'язувати з роботою Л. Ейлера, в якій він розв'язує задачу про мости Кеніґсберґа.



Чи можна прогулятися, пройшовши по кожному з мостів один раз і повернутися до початкової точки? При такому розташуванні позитивних рішень немає.

У 19 столітті Кірхгоф вивчав електричні схеми і окремо досліджував топологічну структуру, яку зараз називають графом схеми. Задача комівояжера, в якій потрібно відвідати всі точки, не завжди має рішення. Задача чотирьох кольорів – розфарбувати географічну карту так, щоб будь-які дві сусідні країни були пофарбовані різними кольорами.

Граф (орієнтований граф) — це пара множин. Елементами першої множини є точки, які називаються вершинами. Елементи другої множини являють собою невпорядковані (впорядковані) пари вершин, які називаються ребрами (дугами).

Множина *X* разом із бінарним відношенням *U* на цій множині називається графом $G=(X,U)$.

Якщо точки x_i, x_j з'єднані дугою, то вони називаються **суміжними.**

Якщо $i=j$ тоді дуга (x_i, x_j) називається петлею. Ребро (x_i, x_j) називається **інцидентним** вершинам x_i і x_j. Кількість ребер, інцидентих якійсь вершині,

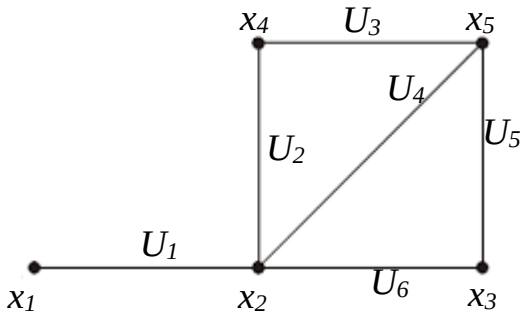
називається степенем вершини. Петлю враховують двічі. Вершина степеня 0 називається ізольованою, ступеня 1 - висячою. Граф, який не має паралельних (множинних) ребер і петель, називається **простим**, у іншому випадку - мультиграфом. **Орієнтований** граф вказує напрямок.

Граф, що складається лише з ізольованих вершин, називається нульовим або порожнім графом. У орієнтованому графі розрізняють кількість вихідних та вхідних дуг для окремої вершини.

Послідовність вершин графа називається **маршрутом (ланцюгом)**, якщо будь-які дві сусідні вершини в цій послідовності є суміжними. Довжина маршруту дорівнює кількості ребер, які його утворюють.

Простий ланцюг — це маршрут, який містить усі різні ребра. Якщо всі вершини в маршруті різні, він називається **елементарним ланцюгом**. Маршрут, перша і остання вершини якого збігаються, називається **циклом.** Як і ланцюги, існують прості та елементарні цикли.

Розглянемо наступні маршрути.



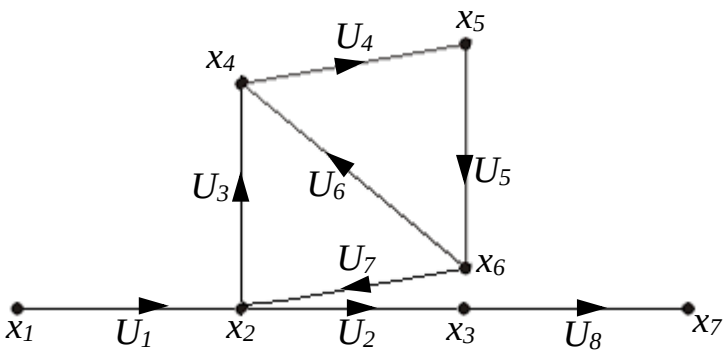
Непростий неелементарний ланцюг $\mu_1 = U_1U_4U_5U_6U_4U_3U_2$.

Простий елементарний ланцюг $\mu_1 = U_1U_2U_3U_5$.

Простий цикл $\mu_1 = U_2U_3U_4$.

Довжина цих маршрутів $l(\mu_1)=7,\ l(\mu_2)=4,\ l(\mu_3)=3$.

Для орієнтованого графа



Непростий неелементарний шлях $\mu_1 = U_1U_3U_4U_5U_6U_4U_5U_7U_2U_8$.

Простий шлях $\mu_1 = U_1U_3U_4U_5U_7U_2U_8$.

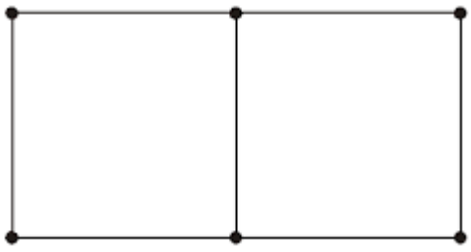
Простий елементарний цикл $\mu_1 = U_3U_4U_5U_7$.

Довжина цих маршрутів $l(\mu_1)=10,\ l(\mu_2)=7,\ l(\mu_3)=4.$

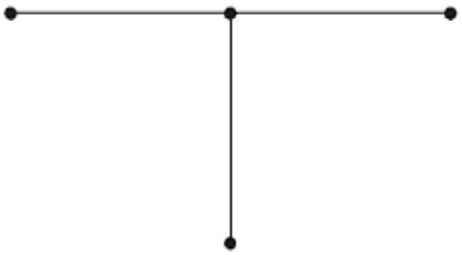
Підграфом $G_1 = (X_1, U_1)$ графа $G = (X, U)$ називається граф, множини вершин і дуг якого є підмножинами множин вершин і дуг вихідного графа

$$X_1 \subseteq X; \qquad U_1 \subseteq U.$$

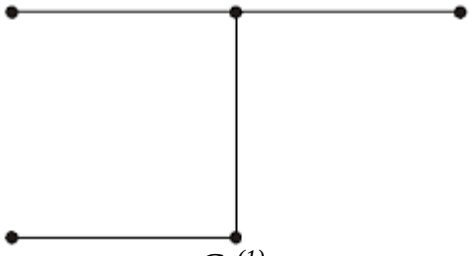
Частковим графом $G_2 = (X_2, U_2)$ графа $G = (X, U)$ називається граф, вершини якого збігаються з вершинами графа, але деякі дуги виключені.



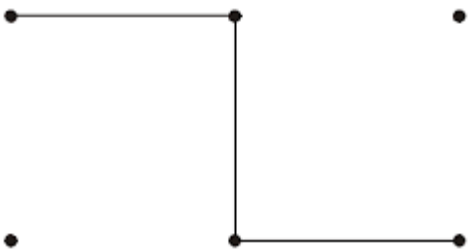
G



G1

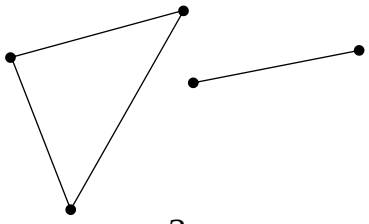


G1(1)

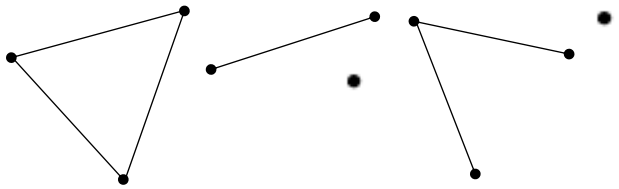


G2

Граф називається **зв'язним**, якщо існує шлях, що з'єднує будь-які дві його вершини. Максимальний підграф цього графа за вхідними компонентами називається його зв'язною компонентою. Зв’язний граф складається з однієї зв’язної компоненти. Незв'язний граф з *n* вершинами може містити від двох до *n* зв'язних компонент.

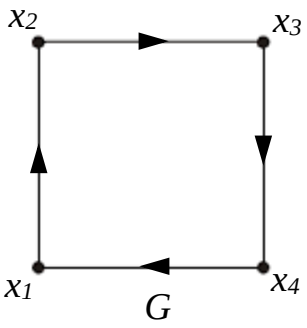


2 компоненти



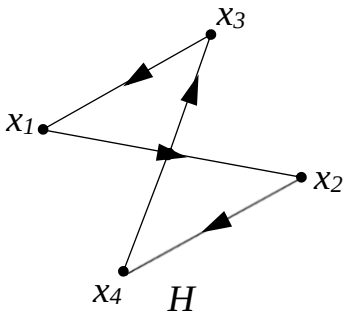
5 компонент

Два графи називаються **ізоморфними**, якщо вони мають однакову кількість вершин і дуг. Як тільки з'єднуються вершини першого графа, відразу з'єднується відповідна пара вершин другого графа і навпаки.



G

$G \cong H$



H

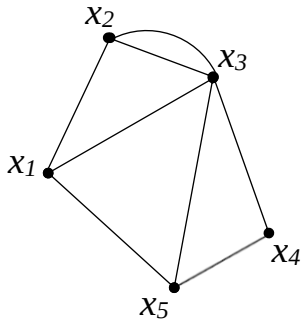
Методи задання графів:

1) граф може бути заданий списком ребер;

2) використовуючи матрицю суміжності або матрицю інцидентності.

Для графа з *n* вершинами матриця суміжності є квадратною матрицею *A* порядку *n*.

Елемент a дорівнює *k*, якщо існує *k* ребер (дуг), які з’єднують вершину з вершиною x_j. Елемент матриці дорівнює нулю, якщо такої сполучної дуги немає.



$$A=\begin{pmatrix}0&1&1&0&1\\1&0&2&0&0\\1&2&0&1&1\\0&0&1&0&1\\1&0&1&1&0\end{pmatrix}$$

Властивості матриці

1. Для неорієнтованого графа матриця симетрична відносно головної діагоналі.

2. Сума елементів рядка (або стовпця) дорівнює степеню вершини x_i.

3. Якщо транспонувати матрицю суміжності орієнтованого графа (поміняти місцями рядки та стовпці), то отримаємо матрицю суміжності графа з протилежною орієнтацією дуг.

Теорема

Якщо для орієнтованого графа будується матриця суміжності *A*, то елемент p_{ij} матриці $P=A^{\lambda}$ дорівнює числу різних шляхів довжиною λ від x_i до x_j.

Нехай орграф має *n* вершин і *m* дуг.

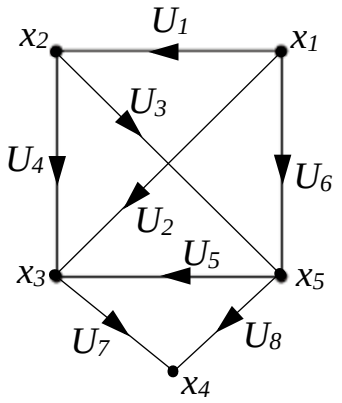
Матрицею інцидентності дуг орієнтованого графа є матриця *B* розмірності *n*x*m*, елементи якої:

+1 якщо з вершини виходить дуга;

-1 якщо дуга входить у вершину;

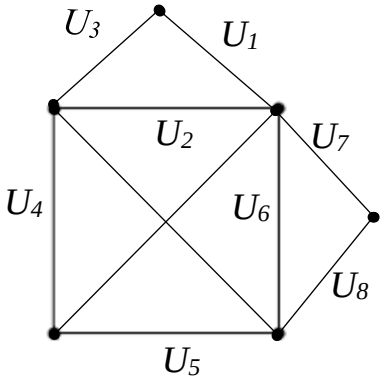
0 якщо дуга не інцидентна вершині.

Для неорієнтованого графа значення в матриці *1*, якщо ребро є випадковим до вершини, або *0*, якщо ребро не є випадковим.



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

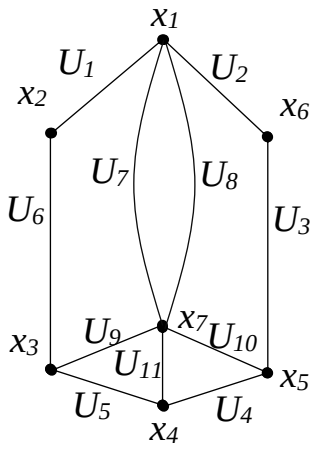
Матрицею циклів є матриця *C*, у якій кожному простому циклу відповідає рядок, а кожному ребру – стовпець. Елемент матриці *1*, якщо ребро U_j входить в цикл i, *0*, якщо ребро U_j не входить в цикл μ_j.



$$\mu_1=(U_1,U_2,U_3); \qquad \mu_2=(U_2,U_4,U_5,U_6);$$
$$\mu_3=(U_6,U_7,U_8); \qquad \mu_4=(U_1,U_3,U_4,U_5,U_6);$$
$$\mu_5=(U_2,U_4,U_5,U_8,U_7);$$
$$\mu_6=(U_1,U_3,U_4,U_5,U_8,U_7).$$

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

Приклад 1. Знайти степінь вершин неорієнтованого графа. Побудуйте матрицю суміжності та матрицю інцидентності.



1) $\deg x_1 = 4,\ \deg x_2 = 2,\ \deg x_3 = 3,$

$\deg x_4 = 3,\ \deg x_5 = 3,\ \deg x_6 = 2,$

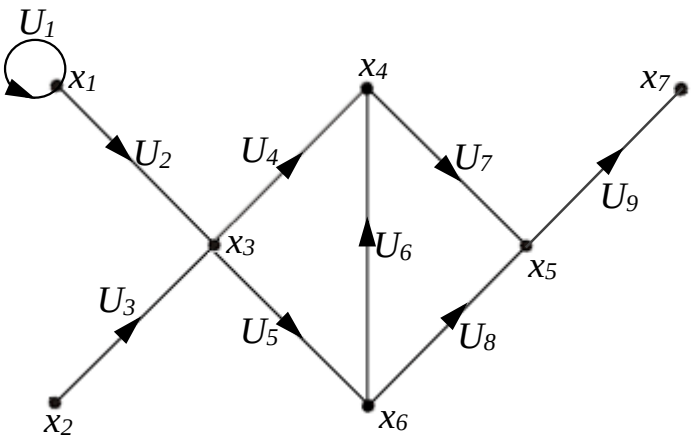
$\deg x_7 = 5;$

2) $$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 2 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix};$$

3) $$B = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} U_1 & U_2 & U_3 & U_4 & U_5 & U_6 & U_7 & U_8 & U_9 & U_{10} & U_{11} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Кожне ребро з'єднує дві вершини графа.

Приклад 2. Побудуйте матрицю суміжності та матрицю інцидентності для орієнтованого графа.



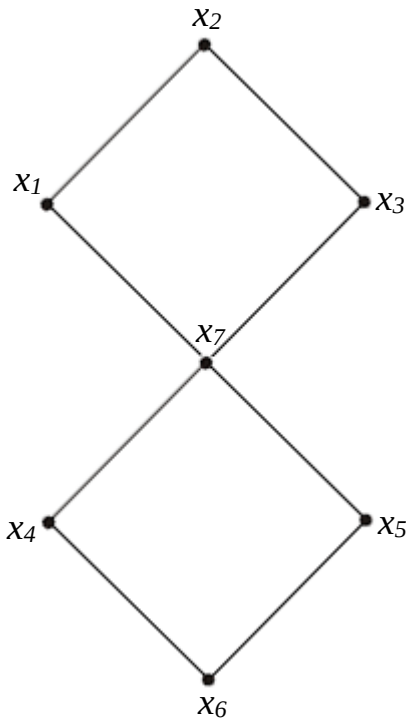
$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$B=\begin{pmatrix} \pm1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

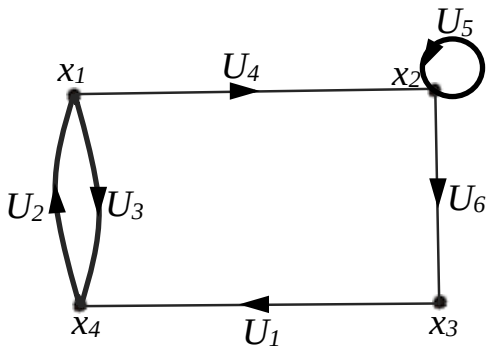
Дуга залишає вершину і входить у вершину.

Приклад 3. Обернена задача. За типом матриці відновити граф.

1) Якщо матриця A симетрична, то шуканий граф неорієнтований.
2) Якщо матриця B містить плюс і мінус одиницю, то граф є орієнтованим.



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



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

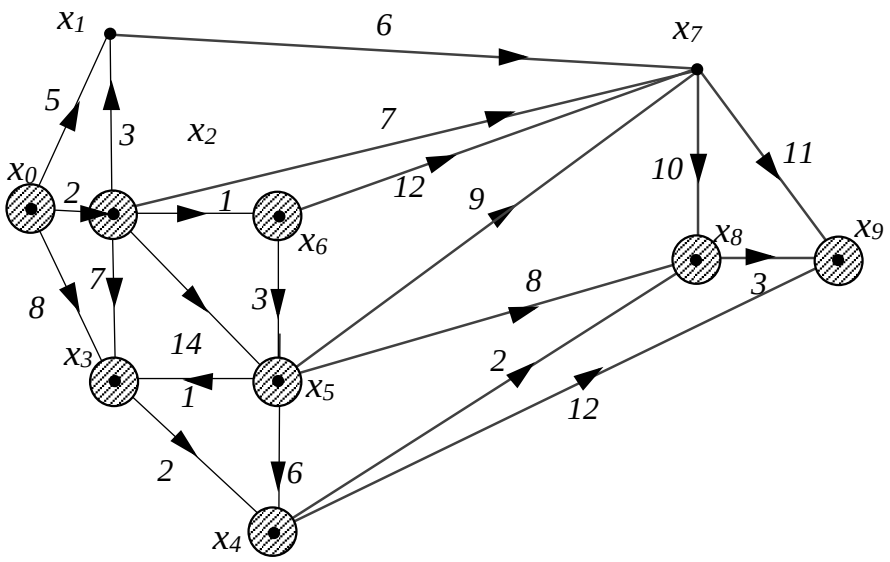
Контрольні питання

1. Що таке граф? Які бувають види графів?
2. Які типи маршрутів існують у графах?
3. Які існують способи задання графа?
4. Що таке матриця суміжності?
5. Що таке матриця інцидентності?

<u>Розділ 2.</u> Задача на знаходження найкоротшого шляху в графі

Нехай кожній дузі графа відповідає певне число $l \geq 0$, яке називається довжиною дуги. Це так звані навантажені або зважені графіки. Числа можуть нести різну інформацію. Це може бути довжина шляху, пропускна здатність трубопроводу, різні часові показники і т.д.

У такому орієнтованому графі довжина шляху, що з'єднує вершини x_0 і z обчислюється шляхом підсумовування довжин усіх дуг, що утворюють шлях. У цій задачі вам потрібно знайти найкоротший шлях від точки x_0 до точки x_9.



Для вирішення задачі скористаємося ***<u>одним із варіантів алгоритмів Дейкстри.</u>***

Метод полягає в призначенні та зміні міток вершин і виборі оптимальних результатів.

1. На першому кроці початковою точкою є **поточна**, їй присвоєно мітку *0*, усі інші вершини помічені ∞.

2. На наступних кроках мітки перепризначаються, *чим менша мітка вершини, тим краще.*

Теги змінюються відповідно до формули

$$\lambda(x_i) = \min\ \left\{\lambda(x_i), \lambda(x_{поточне}) + l(x_{поточне}, x_i)\right\}.$$

Наступною поточною точкою є точка з найменшою міткою. Її можна заштрихувати.

На нашій діаграмі ми можемо рухатися від поточної точки до точок. Позначення цих точок відрізняються. Інші теги ∞.

$$\lambda(x_1) = \min\left\{\lambda(x_1), \lambda(x_0) + l(x_0, x_1)\right\} = \min\left\{\infty, 0+5\right\} = 5;$$

$$\lambda(x_2) = \min\left\{\lambda(x_2), \lambda(x_0) + l(x_0, x_1)\right\} = \min\left\{\infty, 0+2\right\} = 2;$$

$\lambda(x_3) = \min\{\lambda(x_3), \lambda(x_0) + l(x_0, x_3)\} = \min\{\infty, 0+8\} = 8;$

$\lambda(x_4) = \min\{\lambda(x_4), \lambda(x_0) + l(x_0, x_4)\} = \min\{\infty, 0+\infty\} = \infty;$

……

Наступна поточна точка x_2. Знову змініть мітки, де це можливо.

$\lambda(x_1) = \min\{\lambda(x_1), \lambda(x_2) + l(x_2, x_1)\} = \min\{5, 2+3\} = 5;$

$\lambda(x_3) = \min\{\lambda(x_3), \lambda(x_2) + l(x_2, x_3)\} = \min\{8, 2+7\} = 8;$

$\lambda(x_4) = \min\{\lambda(x_4), \lambda(x_2) + l(x_2, x_4)\} = \min\{\infty, 2+\infty\} = \infty;$

$\lambda(x_5) = \min\{\lambda(x_5), \lambda(x_2) + l(x_2, x_5)\} = \min\{\infty, 2+14\} = 16;$

$\lambda(x_6) = \min\{\lambda(x_6), \lambda(x_2) + l(x_2, x_6)\} = \min\{\infty, 2+1\} = 3;$

$\lambda(x_7) = \min\{\lambda(x_7), \lambda(x_2) + l(x_2, x_7)\} = \min\{\infty, 2+7\} = 9;$

$\lambda(x_8) = \min\{\lambda(x_8), \lambda(x_2) + l(x_2, x_8)\} = \min\{\infty, \infty\} = \infty;$

$\lambda(x_9) = \min\{\lambda(x_9), \lambda(x_2) + l(x_2, x_9)\} = \min\{\infty, \infty\} = \infty;$

$\min\{5, 8, 16, 3, 9, \infty\} = 3$. Зобразимо x_6. $\lambda(x_6) = 3$

Наступна поточна точка x_6.

$\lambda(x_1) = \min\{\lambda(x_1), \lambda(x_6) + l(x_6, x_1)\} = \min\{5, 3+\infty\} = 5;$

$\lambda(x_3) = \min\{\lambda(x_3), \lambda(x_6) + l(x_6, x_3)\} = \min\{8, 3+\infty\} = 8;$

$\lambda(x_4) = \min\{\lambda(x_4), \lambda(x_6) + l(x_6, x_4)\} = \min\{\infty, 3+\infty\} = \infty;$

$\lambda(x_5) = \min\{\lambda(x_5), \lambda(x_6) + l(x_6, x_5)\} = \min\{16, 3+3\} = 6;$

$\lambda(x_7) = \min\{\lambda(x_7), \lambda(x_6) + l(x_6, x_7)\} = \min\{9, 3+12\} = 9;$

$\lambda(x_8) = \min\{\lambda(x_8), \lambda(x_6) + l(x_6, x_8)\} = \min\{\infty, 3+\infty\} = \infty;$

$\lambda(x_9) = \min\{\lambda(x_9), \lambda(x_6) + l(x_6, x_9)\} = \min\{\infty, \infty\} = \infty;$

$\min\{5,8,6,9,\infty\}=5$. Зобразимо x_1. $\lambda(x_1)=5$

Наступна поточна точка x_1.

$\lambda(x_3)=\min\{\lambda(x_3),\lambda(x_1)+l(x_1,x_3)\}=\min\{8,5+\infty\}=8;$

$\lambda(x_4)=\min\{\lambda(x_4),\lambda(x_1)+l(x_1,x_4)\}=\min\{\infty,5+\infty\}=\infty;$

$\lambda(x_5)=\min\{\lambda(x_5),\lambda(x_1)+l(x_1,x_5)\}=\min\{6,5+\infty\}=6;$

$\lambda(x_7)=\min\{\lambda(x_7),\lambda(x_1)+l(x_1,x_7)\}=\min\{9,5+6\}=9;$

$\lambda(x_8)=\min\{\lambda(x_8),\lambda(x_1)+l(x_1,x_8)\}=\min\{\infty,5+\infty\}=\infty;$

$\lambda(x_9)=\min\{\lambda(x_9),\lambda(x_1)+l(x_1,x_9)\}=\min\{\infty,5+\infty\}=\infty;$

$\min\{\infty,\infty,6,9,\infty,\infty\}=6$. Зобразимо x_5. $\lambda(x_5)=6$

Наступна поточна точка x_5.

$\lambda(x_3)=\min\{\lambda(x_3),\lambda(x_5)+l(x_5,x_3)\}=\min\{8,6+1\}=7;$

$\lambda(x_4)=\min\{\lambda(x_4),\lambda(x_5)+l(x_5,x_4)\}=\min\{\infty,6+6\}=12;$

$\lambda(x_7)=\min\{\lambda(x_7),\lambda(x_5)+l(x_5,x_7)\}=\min\{9,6+9\}=9;$

$\lambda(x_8)=\min\{\lambda(x_8),\lambda(x_5)+l(x_5,x_8)\}=\min\{\infty,6+8\}=14;$

$\lambda(x_9)=\min\{\lambda(x_9),\lambda(x_5)+l(x_5,x_9)\}=\min\{\infty,6+\infty\}=\infty;$

$\min\{7,12,9,14,\infty\}=7$. Зобразимо x_3. $\lambda(x_3)=7$

Наступна поточна точка x_3.

$\lambda(x_4)=\min\{\lambda(x_4),\lambda(x_3)+l(x_3,x_4)\}=\min\{12,7+2\}=9;$

$\lambda(x_7)=\min\{\lambda(x_7),\lambda(x_3)+l(x_3,x_7)\}=\min\{9,\infty\}=9;$

$\lambda(x_8)=\min\{\lambda(x_8),\lambda(x_3)+l(x_3,x_8)\}=\min\{14,7+\infty\}=14;$

$\lambda(x_9)=\min\{\lambda(x_9),\lambda(x_3)+l(x_3,x_9)\}=\min\{\infty,7+\infty\}=\infty;$

$\min\{9,9,14,\infty\}=9$. Зобразимо x_4. $\lambda(x_4)=7$

Наступна поточна точка x_4.

$\lambda(x_7)=\min\{\lambda(x_7),\lambda(x_4)+l(x_4,x_7)\}=\min\{9,\infty\}=9;$

$\lambda(x_8)=\min\{\lambda(x_8),\lambda(x_4)+l(x_4,x_8)\}=\min\{14,9+2\}=11;$

$\lambda(x_9)=\min\{\lambda(x_9),\lambda(x_4)+l(x_4,x_9)\}=\min\{\infty,9+12\}=21;$

$\min\{9,11,21\}=9$. Зобразимо x_7. $\lambda(x_7)=9$

Наступна поточна точка x_7.

$\lambda(x_8)=\min\{\lambda(x_8),\lambda(x_7)+l(x_7,x_8)\}=\min\{11,9+10\}=11;$

$\lambda(x_9)=\min\{\lambda(x_9),\lambda(x_7)+l(x_7,x_9)\}=\min\{21,9+11\}=20;$

$\min\{11,20\}=11$. Зобразимо x_8. $\lambda(x_8)=11$

Наступна поточна точка x_8.

$\lambda(x_9)=\min\{\lambda(x_9),\lambda(x_8)+l(x_8,x_9)\}=\min\{20,11+3\}=14;$

Мітка останньої вершини дорівнює довжині найкоротшого шляху.

Щоб точно вказати цей шлях (особливо якщо ви працювали з великим графом), ви повинні повернутися з кінцевої точки до початкової.

Повернення відбувається по поточних точках (вони заповнюються). У цьому випадку має бути виконана наступна умова:

$$\lambda(x_N)-\lambda(x_{N-1})=l(x_{N-1},x_N).$$

Різниця між мітками цих точок повинна збігатися з довжиною дуги між ними.

$\lambda(x_9)-\lambda(x_7)=14-9=5\neq 11;$

$\lambda(x_9)-\lambda(x_8)=14-11=3=l;$

$\lambda(x_8)-\lambda(x_7)=11-9=2\neq 10;$

$\lambda(x_8)-\lambda(x_4)=11-9=2;$

……

$\lambda(x_3)-\lambda(x_5)=1;$

$\lambda(x_5)-\lambda(x_6)=3.$

Контрольні питання

1. Яка основна ідея алгоритму Дейкстри?
2. Яка формула існує для вибору найкоротшого шляху?
3. Який найкоротший шлях?

Розділ 3. Дерева. Визначення та властивості

У теорії графів **дерево** — це неорієнтований граф, у якому будь-які дві вершини з'єднані *точно одним* шляхом, або, еквівалентно, зв'язаний ациклічний неорієнтований граф. **Ліс** — це неорієнтований граф, у якому будь-які дві вершини з'єднані *щонайбільше одним* шляхом, або, еквівалентно, ациклічний неорієнтований граф, або, еквівалентно, непересічне об'єднання дерев.

Різні типи структур даних, які в інформатиці називаються деревами, мають базові графи, які є деревами в теорії графів, хоча такі структури даних, як правило, є **кореневими деревами**.

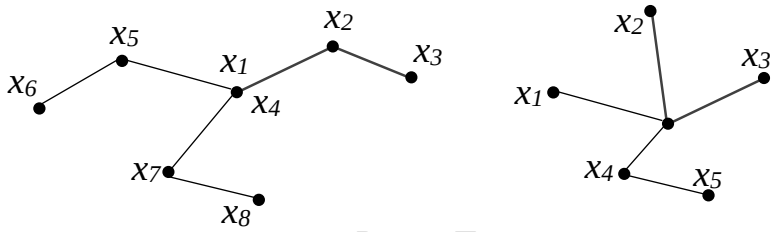


Рис1. Дерева

Дерева

Дерево — це неорієнтований граф G, який задовольняє будь-яку з наступних еквівалентних умов:

- G зв'язний і ациклічний (не містить циклів).
- G є ациклічним, і простий цикл утворюється, якщо до нього додати будь-яке ребро G.
- G зв’язний, але стане незв’язним, якщо з нього буде видалено будь-яке окреме ребро G.
- G є зв’язним і 3-вершинний повний граф K_3 не є мінором G.
- Будь-які дві вершини в G можуть бути з'єднані єдиним простим шляхом.

Якщо G має кінцеву кількість вершин, скажімо, n з них, то наведені вище твердження також еквівалентні будь-якій із наступних умов:

- G зв'язний і має $n-1$ ребер.
- G зв'язний, і кожен підграф G містить принаймні одну вершину з нульовим або одним інцидентним ребром. (Тобто G є зв’язним і 1-виродженим.)
- G не має простих циклів і має $n-1$.

Як і будь-де, в теорії графів, граф нульового порядку (граф без вершин) зазвичай не вважається деревом: хоча він порожньо зв’язаний, як граф (будь-які дві вершини можуть бути з’єднані шляхом), він не є 0 -зв'язним (або навіть (-1)-зв'язним) в алгебраїчній топології, на відміну від непорожніх дерев, і порушує співвідношення «на одну вершину більше, ніж ребер». Однак його можна розглядати як ліс, що складається з нуля дерев.

Внутрішня вершина (або **внутрішня вершина** або **вершина гілки**) є вершиною ступеня принаймні 2. Подібним чином, **зовнішня вершина** (або *зовнішня вершина*, *кінцева вершина* або *лист*) є вершиною ступеня 1.

Незвідне дерево (або *рядово-зведене дерево*) — це дерево, в якому немає вершин ступеня *2*.

Ліс

Ліс — це неорієнтований граф, у якому будь-які дві вершини з’єднані щонайбільше одним шляхом. Еквівалентно, ліс є неорієнтованим ациклічним графом. Еквівалентно, ліс є неорієнтованим графом, усі зв’язані компоненти якого є деревами; іншими словами, граф складається з непересічного об'єднання дерев. Як окремі випадки, нуль-графи (ліс, що складається з нульових дерев), одне дерево та граф без ребер є прикладами лісів.

Полідерево

Полідерево (або *спрямоване дерево*, або *орієнтоване дерево*, або *мережа з одним зв’язком*) — це орієнтований ациклічний граф (ОАГ), основним неорієнтованим графом якого є дерево. Іншими словами, якщо ми замінимо його орієнтовані ребра неорієнтованими ребрами, ми отримаємо неорієнтований граф, який є одночасно зв'язним і ациклічним.

Деякі автори обмежують фразу «спрямоване дерево» випадком, коли всі ребра спрямовані до певної вершини або всі спрямовані від конкретної вершини (див. деревоподібне).

Кореневе дерево

Кореневе дерево — це дерево, в якому одна вершина позначена ***коренем***. Ребрами кореневого дерева можна призначити природну орієнтацію або ***від кореня***, або ***до нього***, і в цьому випадку структура стає ***спрямованим кореневим деревом***. Якщо спрямоване вкорінене дерево має орієнтацію вбік від кореня,

воно називається ***деревоподібним*** або ***зовнішнім деревом***; коли він має орієнтацію в бік кореня, він називається ***анти-деревоподібним.***

У контексті, де дерева повинні мати корінь, дерево без визначеного кореня називається ***вільним деревом***.

Дерево з мітками — це дерево, у якому кожній вершині надається унікальна мітка. Вершини міченого дерева з n вершин зазвичай отримують мітки 1, 2, ..., n. ***Рекурсивне дерево*** — це кореневе дерево з мітками, де мітки вершин відповідають порядку дерева (тобто, якщо $u < v$ для двох вершин u і v, то мітка u менша за мітку v)

У кореневому дереві ***батьківська*** вершина v — це вершина, з'єднана з v на шляху до кореня; кожна вершина має унікальну батьківську, крім кореня, який не має батьківської вершини. ***Дочірня*** вершина v — це вершина, для якої v є батьківською.

Кількість різних дерев, які можна побудувати на n пронумерованих вершинах, обчислюється за формулою Кейлі $N = n^{n-2}$.

Дерево може бути ***закодовано*** наборами нулів і одиниць. Розглянемо дерево на площині. Починаючи з деякої вершини, ми рухаємося по краях дерева. Повертаємо в кожній вершині до краю, найближчого до правого боку, і повертаємо назад у кінцевих вершинах дерева.

При першому русі по краю пишемо *0*, а при другому (у зворотному напрямку) – *1*. Якщо m – кількість ребер дерева, то через $2m$ кроків ми повертаємося до початкової точки, проходячи двічі по кожному ребру. Отримана послідовність *0* і *1* (код дерева) довжиною $2m$ дозволяє відновити не тільки саме дерево, але і його розташування на площині. Довільному дереву відповідає кілька таких кодів. Каркасом (скелетом) зв'язного графа G є будь-який його підграф, який містить усі вершини G і є деревом.

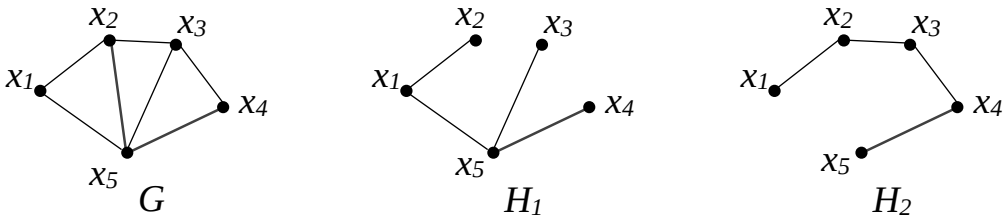


Рис 2. Скелети

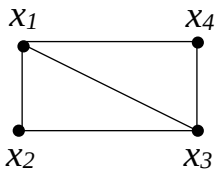
Позначимо через M матрицю з елементами m, отриману з матриці суміжності A графа шляхом заміни знаків усіх елементів на протилежні знаки. Елементи головної діагоналі замінені степенями відповідних вершин.

$$m_{ij} = \begin{cases} -a_{ij}, \text{ if } i \neq j, \ \left(a_{ij} - \text{ матриця суміжності елементів } A\right) \\ \deg x_{ii}, \text{ if } i = j. \end{cases}$$

Теорема про матричне дерево

Нехай G — зв'язний граф без петель і кратних ребер. Тоді всі алгебраїчні доповнення до елементів матриці M будуть рівні між собою і їх спільне значення дає кількість кадрів (скелетів) графа G.

Приклад 1. Знайдіть на графі кількість каркасів (скелетонів, остовів).

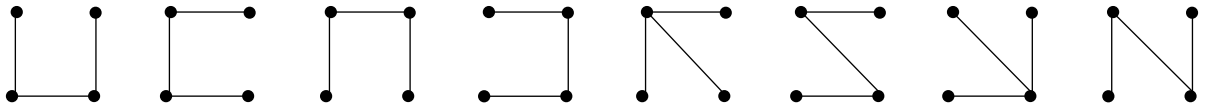


Матриця суміжності:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \quad M = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}.$$

Знайдемо алгебраїчне доповнення (або додаток) до елемента m_{23}. Це мінор з визначеним знаком.

$$A_{23}=(-1)^{2+3}\begin{vmatrix}3 & -1 & -1\\ -1 & -1 & -1\\ -1 & 0 & 2\end{vmatrix}=-6(-6-1+1-2)=8.$$



Задача про найкоротше дерево

Для зваженого або навантаженого графа актуальною є задача знаходження скелета найменшої ваги. Така задача зустрічається при проектуванні комп'ютерних і кабельних мереж, трубопроводів, доріг і т. д. Якщо зв'язний граф близький до повного графу (містить всі можливі ребра і петлі), то кількість дерев-скелетів дорівнює n^{n-2} і пряме перерахування всіх можливих дерев та їх ваг має дуже велику кількість обчислень (для $n=22$ кількість скелетів більше ніж 10^{25}). Існує кілька ефективних класичних алгоритмів. Наприклад, **алгоритм Дж. Краскала та алгоритм Прима**.

Ідея методу **алгоритму Дж. Краскала** полягає в тому, що на кожному кроці вибирається найкоротше ребро (з найменшою вагою), яке не утворює цикл із попередніми ребрами.

Нехай кілька міст пов'язані між собою авіалініями. Необхідно підібрати маршрут так, щоб з будь-якого міста можна було дістатися до іншого (з пересадками) найкоротшим шляхом.

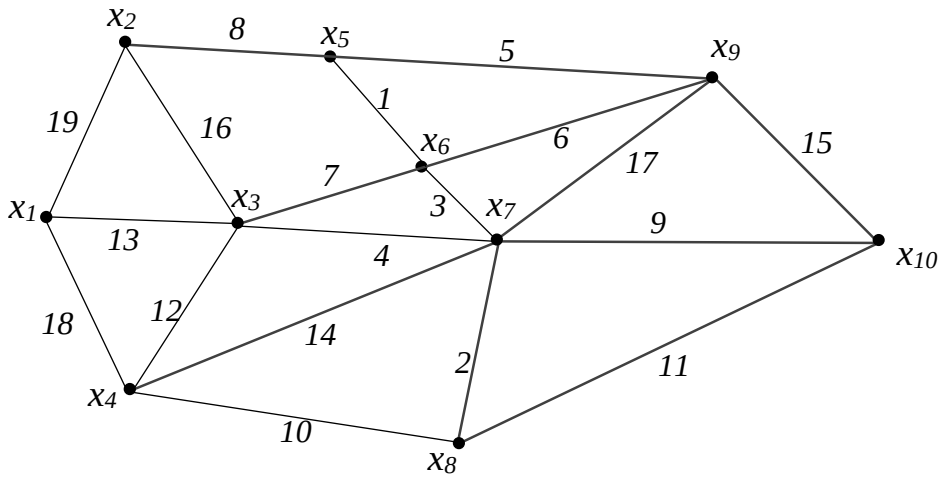
Алгоритм пошуку найкоротшого дерева у графі (алгоритм Краскала)

1. Виберіть найкоротше ребро у графі.
2. На кожному наступному етапі будуйте частину дерева, додаючи найкоротше ребро з тих, що залишилися в графі. Більш того, цикли не

повинні утворюватися. Усі ребра з однаковою мінімальною довжиною включені в дерево, за винятком тих, які утворюють цикли.

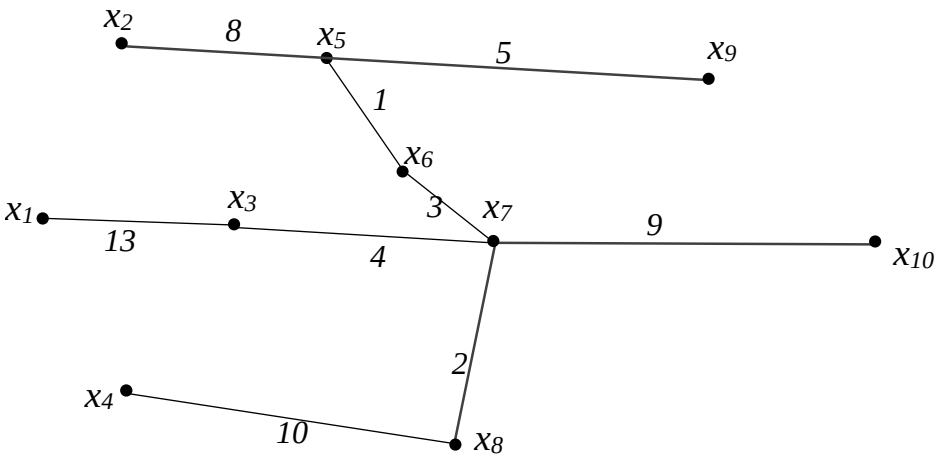
3. Після кроку $n-1$ процес завершується. (n – кількість вершин у графі). В результаті ми отримуємо дерево, яке не містить циклів і має $n-1$ ребер.

Приклад 1:



1) (x_5, x_6)- обираємо найкоротше ребро.
2) (x_7, x_8) - наступне найкоротше ребро.
3) (x_6, x_7)- наступне найкоротше ребро.
4) (x_3, x_7)
5) (x_5, x_9)
6) (x_6, x_9)- не обираємо, оскільки утворюється цикл.
7) (x_3, x_6) - не обираємо, оскільки утворюється цикл.
8) (x_2, x_5)
9) (x_7, x_{10})
10) (x_4, x_8)
11) (x_8, x_{10})

Таким чином, маємо найкоротше дерево:

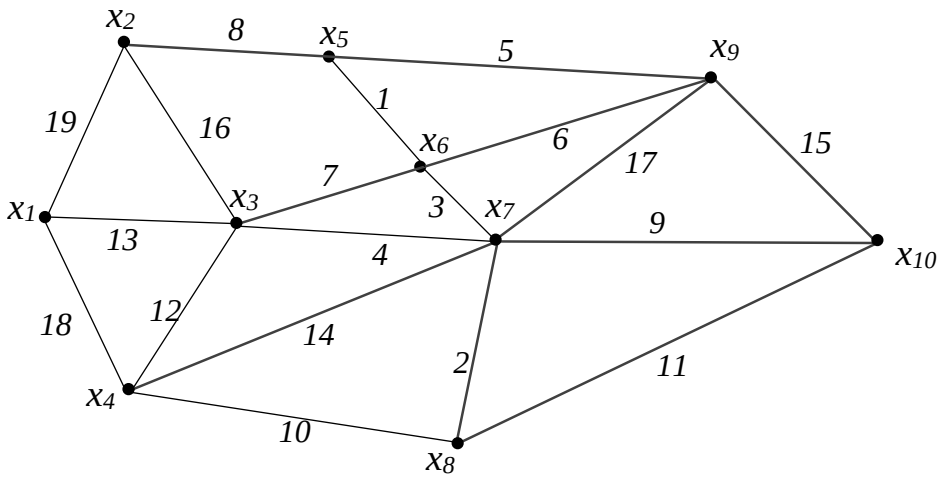


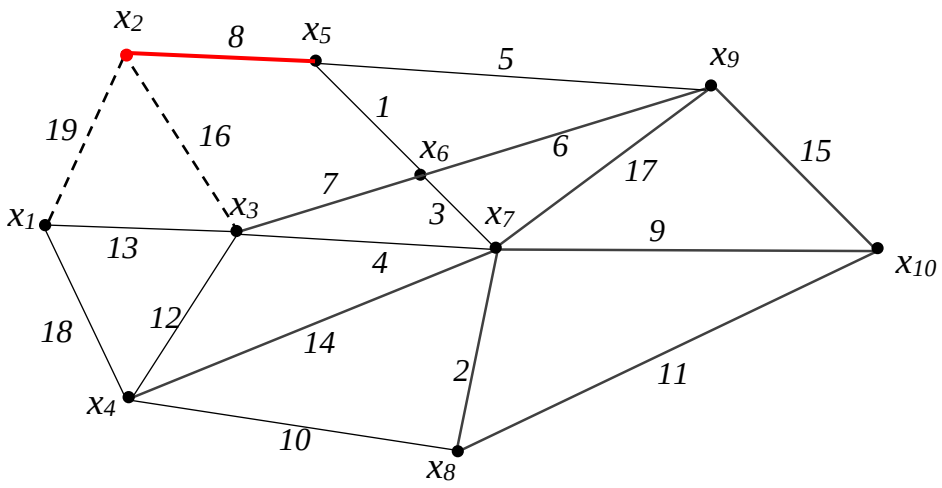
Його довжина 1+2+3+4+5+8+9+10+13=55.

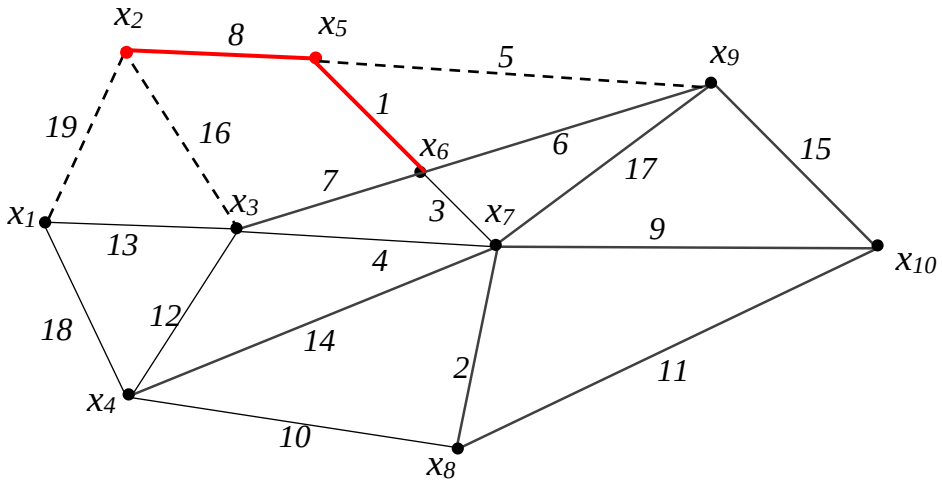
Алгоритм пошуку найкоротшого дерева у графі (алгоритм Прима)

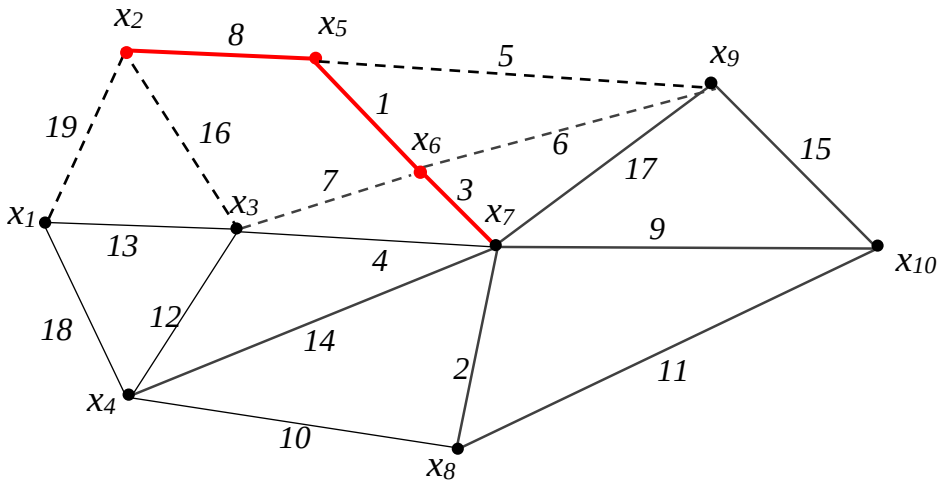
1. Виберіть довільну вершину у графі.
2. Потім, перегляньте інцидентні ребра ***на кожній із кінцевих вершин***, оберіть ребро з мінімальною вагою.
3. Процес закінчується, коли усі вершини буде включено до графу.

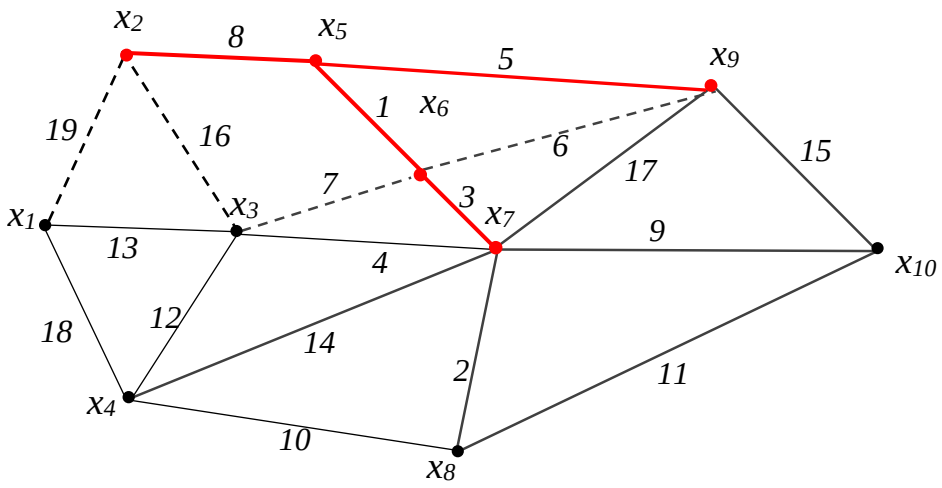
Приклад 2: Розглянемо зважений граф (кожному ребру поставлено у відповідність певне число (вага ребра), що несе різне змістове навантаження в залежності від постановки задачі).

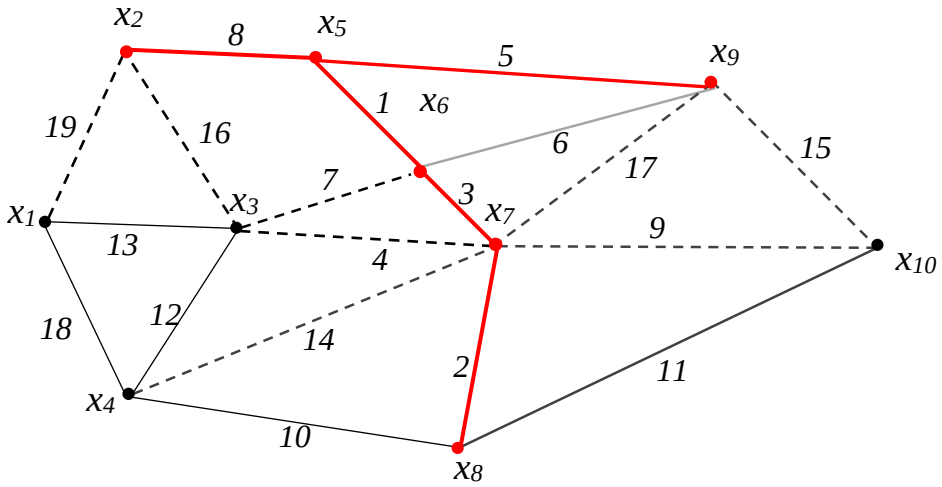


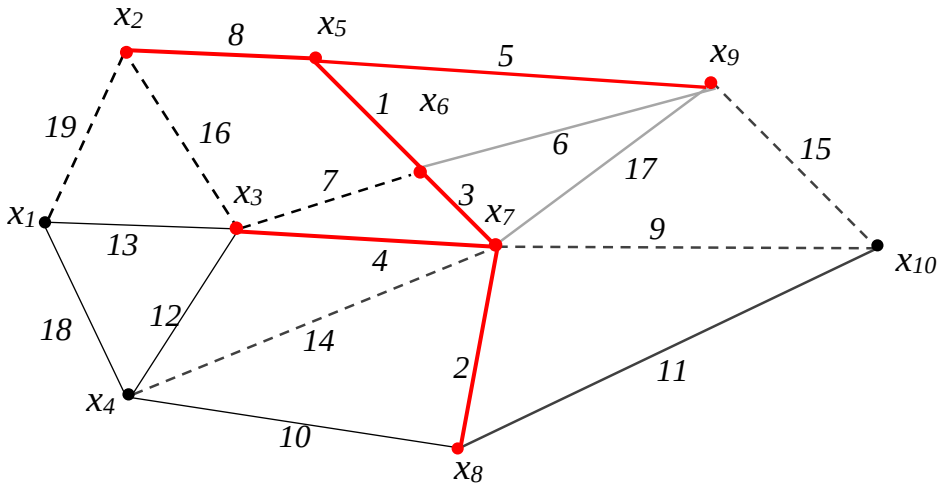

x_2
8
x_5
5
x_9
19
16
1
x_6
6
17
15
7
x_3
3
x_7
9
x_1
13
4
x_{10}
18
12
14
2
11
x_4
10
x_8

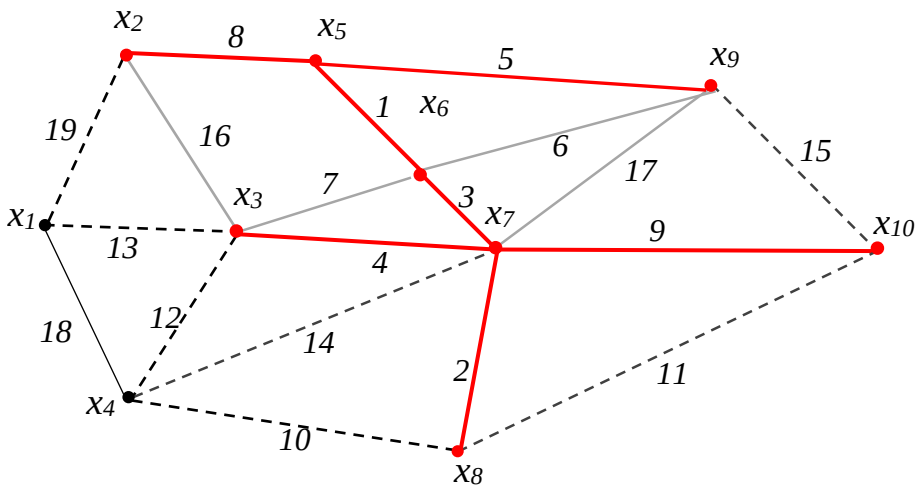

x_2
8
x_5
5
x_9
19
16
1
x_6
6
17
15
7
x_3
3
x_7
9
x_1
13
4
x_{10}
18
12
14
2
11
x_4
10
x_8

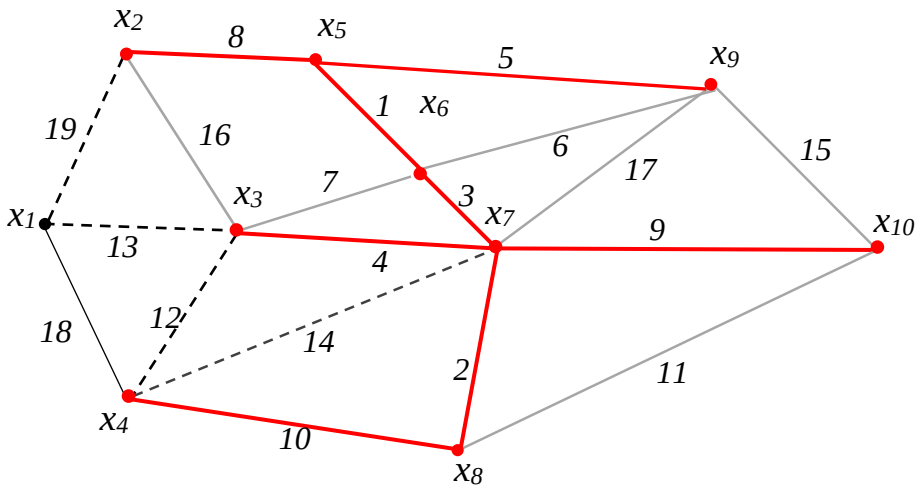

x_2
8
x_5
5
x_9
19
16
1
x_6
6
17
15
7
3
x_3
x_7
9
x_1
13
4
x_{10}
18
12
14
2
11
x_4
10
x_8

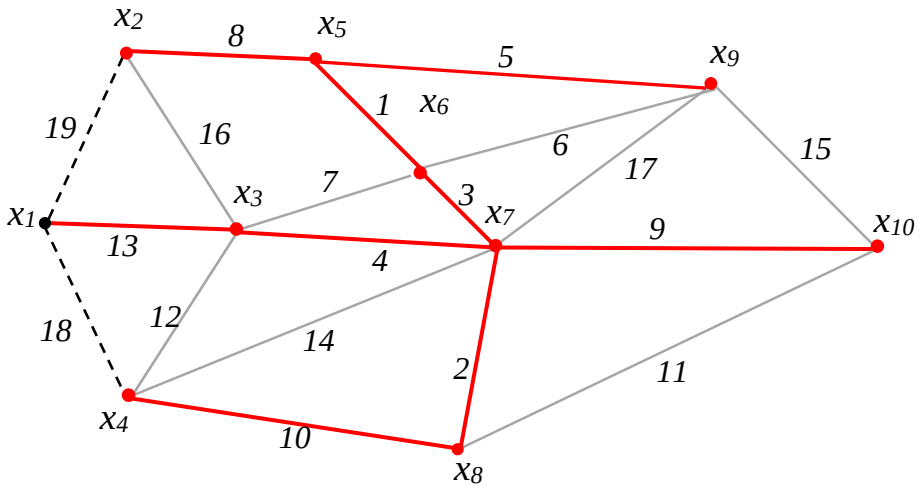

x_2
8
x_5
5
x_9
19
16
1
x_6
6
15
17
7
3
x_1
x_3
x_7
9
x_{10}
13
4
18
12
14
2
11
x_4
10
x_8

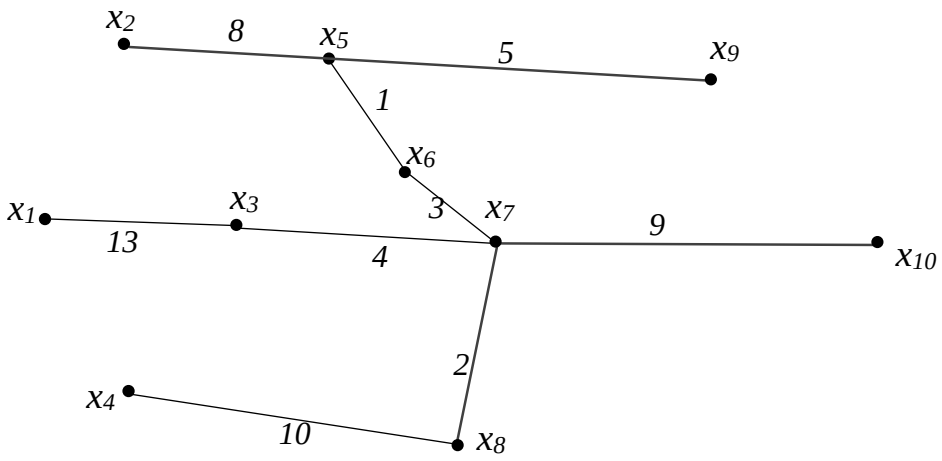

x_2
8
x_5
5
x_9
19
16
1
x_6
6
15
17
7
3
x_1
x_3
x_7
9
x_{10}
13
4
18
12
14
2
11
x_4
10
x_8


x_2
8
x_5
5
x_9
19
16
1
x_6
6
15
17
7
3
x_1
x_3
x_7
9
x_{10}
13
4
18
12
14
2
11
x_4
10
x_8


x_2
8
x_5
5
x_9
19
16
1
x_6
6
17
15
x_1
7
x_3
3
x_7
9
x_{10}
13
4
18
12
14
2
11
x_4
10
x_8


x_2
8
x_5
5
x_9
19
16
1
x_6
6
17
15
x_1
7
x_3
3
x_7
9
x_{10}
13
4
18
12
14
2
11
x_4
10
x_8


x_2
8
x_5
5
x_9
19
16
1
x_6
6
17
15
x_1
7
x_3
3
x_7
9
x_{10}
13
4
18
12
14
2
11
x_4
10
x_8



<u>Задача побудови єдиного каркасного (скелетного) дерева</u> є однією з найважливіших на практиці. Водночас це найпростіша задача для алгоритмізації та комп'ютерної реалізації.

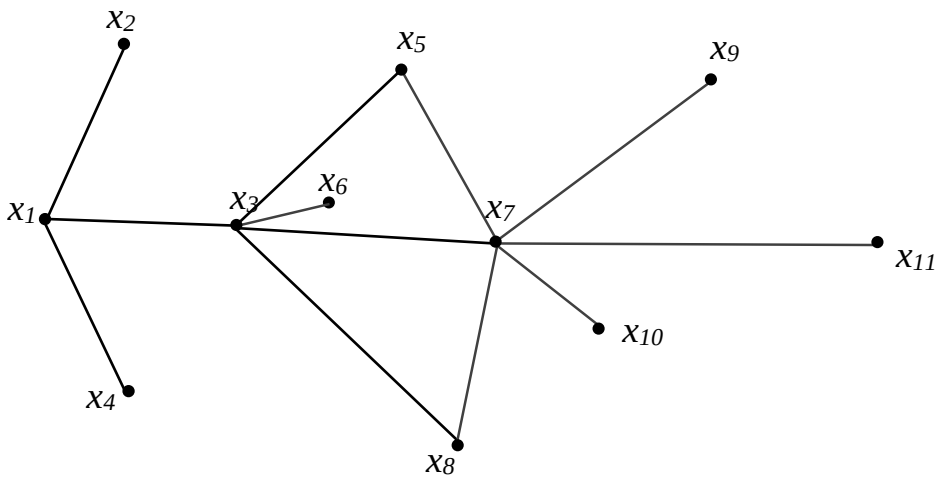
Найвідомішим алгоритмом є **<u>пошук у глибину</u>** (він має лінійну складність), метою якого є відвідати кожну вершину графа рівно один раз. Алгоритм проходить по черзі всі ребра графа і всі його вершини. Прикладом успішного застосування **<u>пошуку вглиб</u>** є дослідження лабіринту міфічним героєм Тесеєм з метою перемоги над Мінотавром.

Виконуємо **<u>обхід графа в глибину</u>** за наступними правилами:

1. Перебуваючи у вершині x_1, потрібно рухатись у будь-яку іншу, раніше не відвідану вершину x_i (якщо така знайдеться). Одночасно з цим необхідно запам'ятати ребро (шлях), по якому потрапляємо у дану вершину.
2. Якщо з вершини x_k немає можливості потрапити в раніше не відвідану вершину, або такої немає, слід повернутись у вершину x_i, з якої вперше потрапили у вершину x_k, і далі продовжуємо обхід у глибину з вершини x_i.

Іншими словами, при виконанні обходу графа у глибину прагнуть просунутись максимально «вглиб» графа, як тільки це можливо, потім відступають на крок назад і знову прагнуть пройти максимально вперед, і так далі.

Приклад 3: Розглянемо <u>неорієнтовний</u> граф.



Почнемо обхід графа з вершини x_1.

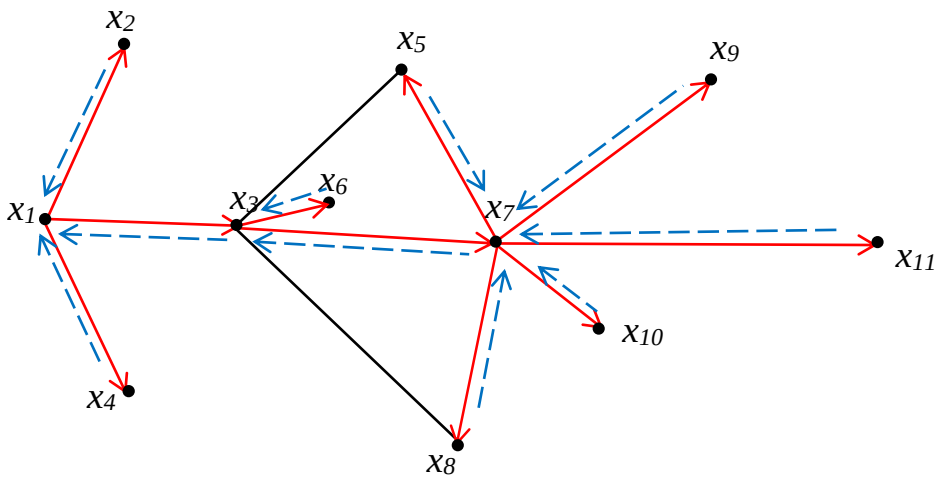
Наступною обираємо будь-яку вершину, наприклад x_3 ($x_1 \to x_3$).

Далі аналогічно переходимо у будь-яку наступну вершину, наприклад x_6 ($x_1 \to x_3 \to x_6$).Подальший рух з цієї вершини неможливий, тож повертаємось у попередню вершину x_3 і обираємо будь-який з трьох можливих варіантів (перехід у вершини $x_5, x_7, x_8,$), наприклад x_7.

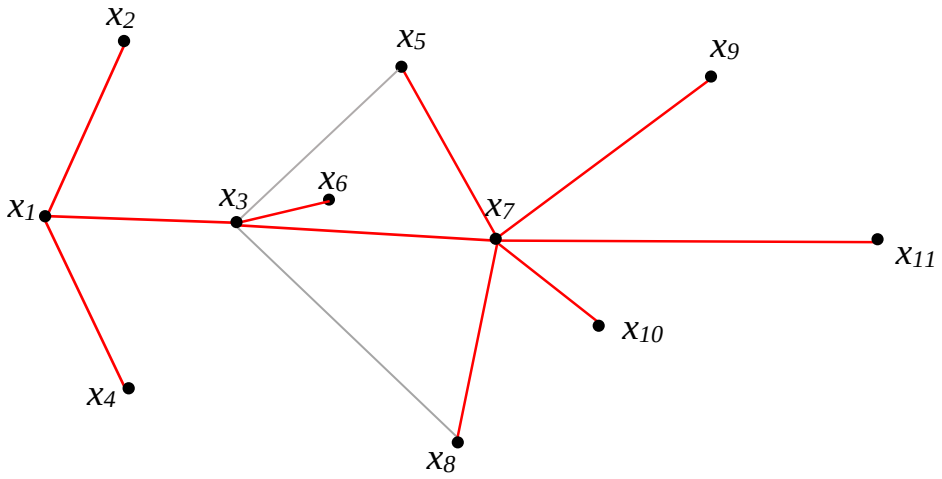
Аналогічним чином рухаємось далі, наприклад у вершину x_{11}. Подальший рух з цієї вершини неможливий, тож повертаємось у попередню вершину x_7 і обираємо будь-який з двох можливих варіантів (перехід у вершини x_9, x_{10}), наприклад x_9. Подальший рух з цієї вершини неможливий, тож повертаємось у попередню вершину x_7 і обираємо останній можливий варіант x_{10}.

Далі доведеться повернутись назад по прокладеному шляху в x_7. Обираємо будь-який з двох можливих варіантів (перехід у вершини x_5, x_8), наприклад x_5. З вершини x_5 є непройдена дуга у вершину x_3, у якій ми вже побували, тож цей перехід не виконуємо.

Знову повертаємось у x_7 і рухаємось у x_8. З вершини x_8 знову є непройдена дуга у вершину x_3, у якій ми вже побували, тож цей перехід не виконуємо. Повертаємось у вершину x_7 і з'ясовуємо, що усі суміжні вершини відвідано, тож повертаємось по пройденому раніше шляху у вершину x_3. Усі суміжні з нею вершини вже відвідано. Повертаємось у вершину x_1 і по черзі прокладаємо шлях у вершини x_2, x_4. Виконання алгоритму завершено.

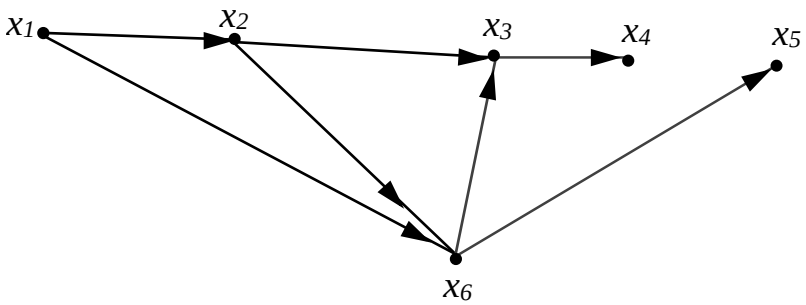


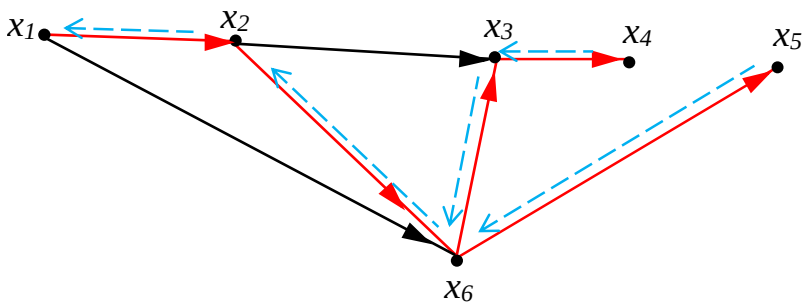
Остаточно маємо:

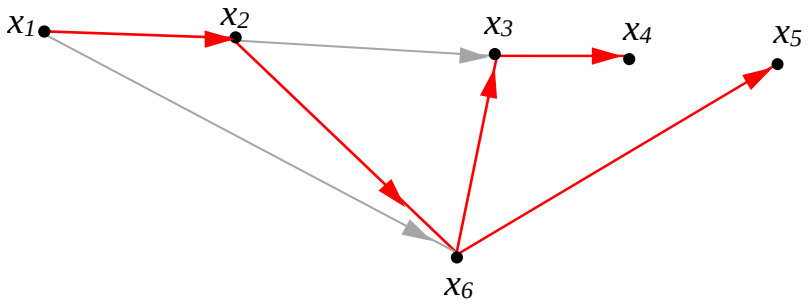


У випадку орієнтовного графа алгоритм працює аналогічно.

Приклад 4: Розглянемо орієнтовний граф.







Наступний алгоритм — **пошук в ширину**. У кожній поточній вершині розглядаються всі інцидентні ребра та їх кінцеві вершини (оточення поточної вершини).

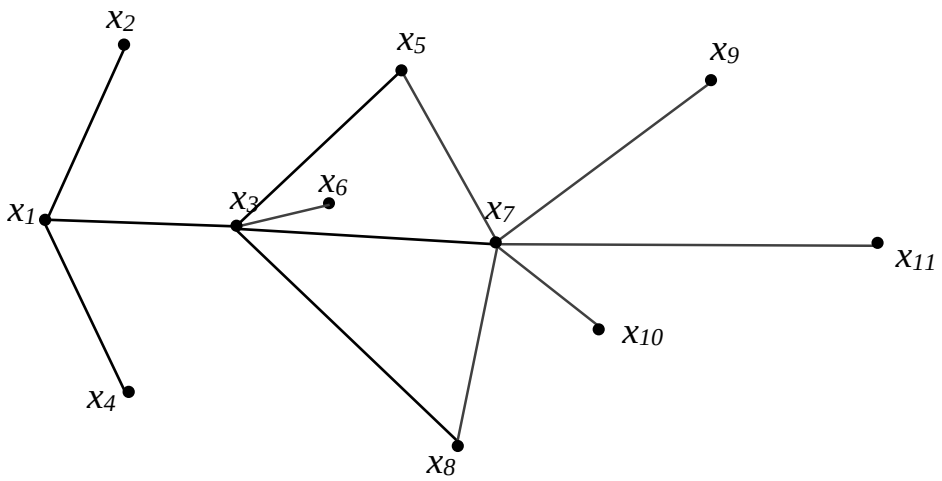
Прикладом обходу в ширину може бути поширення інформації. Наприклад, про зміни в розкладі студентів. Спочатку один студент, зазвичай староста, отримує інформацію про зміни в розкладі, далі цю важливу інформацію треба поширити серед студентів групи або потоку. Староста можливими засобами повідомляє усім, кому може цю інформацію. Потім студенти, кому відома ця інформація, повідомляють інших. Інформація про зміни поширюється майже миттєво.

Виконуємо **обхід графа в ширину** за наступними правилами:

1. Обираємо початкову вершину x_1.
2. З цієї вершини виконуємо перегляд усіх ще не переглянутих вершин, що суміжні з початковою вершиною. Тобто, ведемо пошук у всіх можливих напрямках одночасно.
3. Далі виконуємо перегляд вершин, що розташовані від вершини x_1 на відстані 2 і так далі.

Чим ближче вершина розташована до початкової вершини x_1, тим раніше вона буде відвідана. **Алгоритм пошуку в ширину дозволяє визначити найкоротший можливий шлях.**

Приклад 5: *Розглянемо неорієнтовний граф.*



Почнемо обхід графа з вершини x_1.

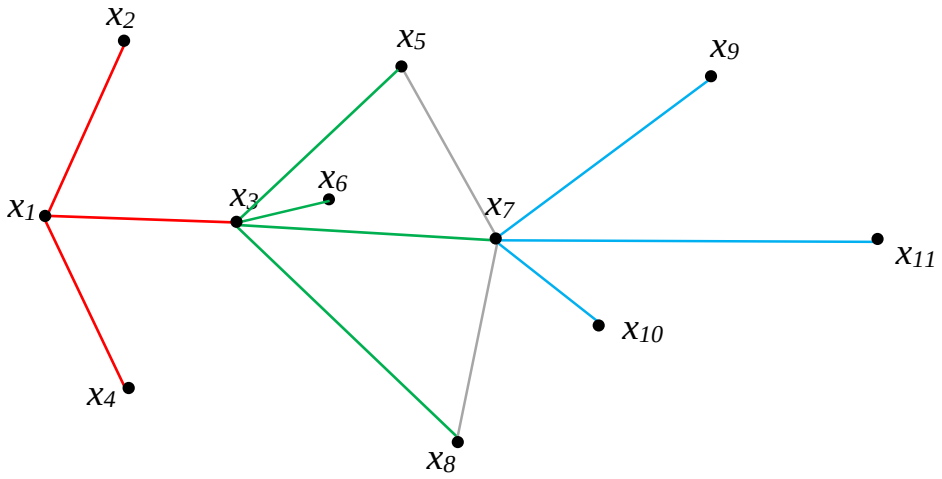
Далі виконуємо перегляд усіх суміжних з вершиною x_1 вершин x_2, x_3, x_4.

На наступному кроці виконуємо перегляд усіх суміжних вершин з вершинами x_2, x_3, x_4. Це вершини x_5, x_6, x_7, x_8.

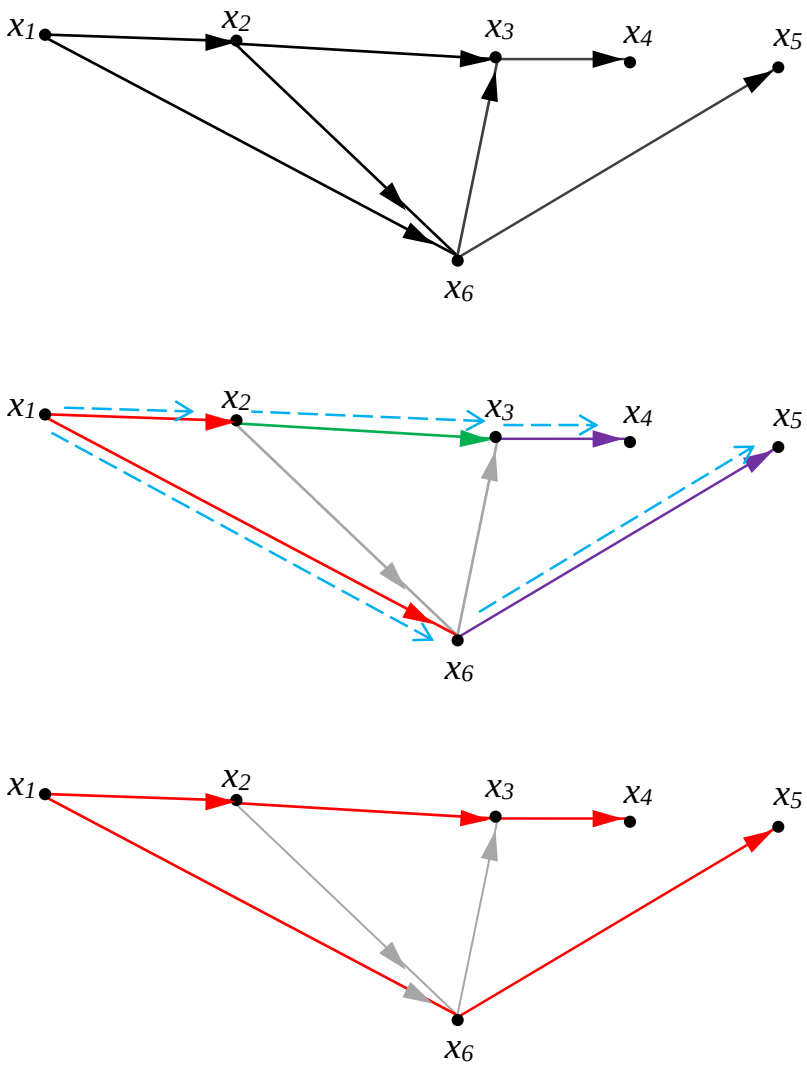
Аналогічно виконуємо перегляд усіх суміжних вершин з вершинами x_5, x_6, x_7, x_8, при цьому на кожному кроці перевіряючи, чи не була вже раніше переглянута вершина. Вершин, суміжних з вершиною x_6, які б раніше не були переглянуті, немає. Суміжною до вершин x_5, x_8 буде вершина x_7, переглянута на попередньому кроці.

Подальший перегляд виконуємо з вершини x_7, суміжними з нею будуть вершини x_9, x_{10}, x_{11}, не переглянуті раніше.

На цьому виконання алгоритму завершено.



Приклад 6: Розглянемо орієнтовний граф.



На останньому рисунку наведено результат застосування алгоритму пошуку в ширину, який дозволяє визначити найкоротший можливий шлях.

Шлях Ейлера — це маршрут у неорієнтованому графі, який проходить один раз по всіх ребрах графа. Якщо такий маршрут замкнений, то його називають циклом Ейлера. Граф, в якому є ейлерів цикл, називається ейлеровим.

Теорема. Для того, щоб скінченний зв'язний граф мав цикл Ейлера, необхідно і достатньо, щоб степені всіх його вершин були парними.

Гамільтонів шлях у графі — це маршрут, який перетинає кожну вершину графа лише один раз. Якщо шлях замкнутий, то це гамільтонів цикл. Граф, у

якому є такий цикл, називається гамільтоновим. Немає загального критерію для визначення гамільтонового графа.

Перш за все, з поняттям **гамільтонового графа** пов'язана **задача комівояжера**. Загальна **задача комівояжера** полягає в наступному: по заданій системі доріг відвідати всі пункти або міста в такій послідовності, щоб пройдений шлях був найкоротшим. Мовою теорії графів така задача ставиться так: у навантаженому зв’язному графі знайти найкоротший маршрут, який проходить через усі вершини графа. Можлива додаткова умова для закритого маршруту. Ця звичайна задача комівояжера завжди має рішення.
Є ще одна постановка попередньої задачі, в якій додатково вимагається, щоб продавець відвідував кожну точку лише **один раз**.

Це **гамільтонова задача комівояжера**. У неї не завжди є розв’язок. Ці ж задачі можна сформулювати для орієнтованих графів. Обчислювальна складність таких завдань дуже висока, вона порядку $(n-1)!$.

Точні алгоритми, які гарантують отримання маршруту продавця в будь-якому випадку, складні і застосовуються до невеликих графів. Алгоритми апроксимації застосовуються до великих графів. Однак вони можуть призвести до неоптимального маршруту.

Контрольні питання

1. Який граф називається деревом?
2. Що обчислюється за матричною теоремою про дерева?
3. Які графи називають ейлеровими чи гамільтоновими?
4. Укажіть один із способів знаходження найкоротшого дерева в графі?

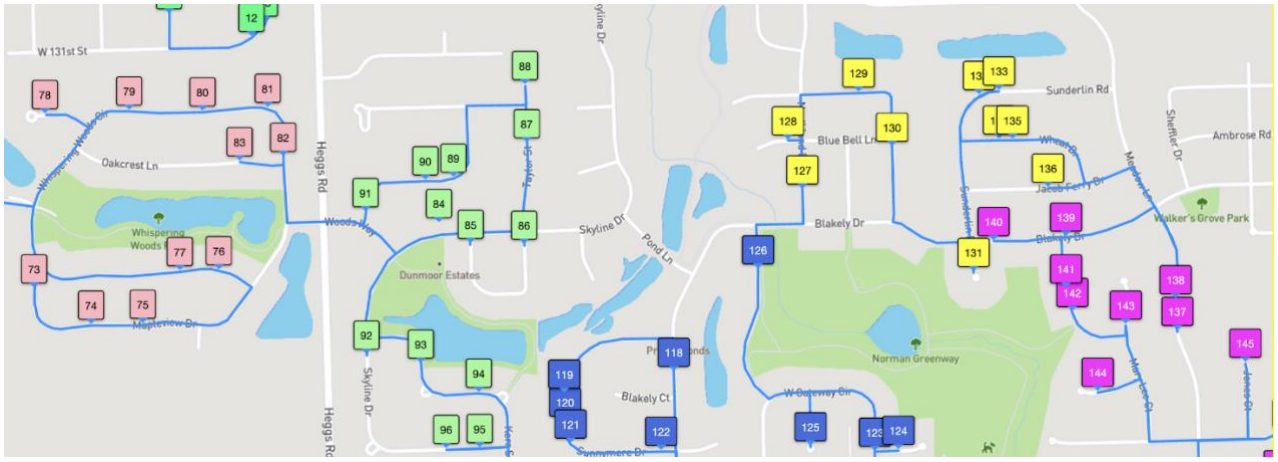
Розділ 4. Деякі застосування теорії графів

4.1. Задача комівояжера (TSP- The traveling salesman problem) це алгоритмічна задача, яка має знайти найкоротший маршрут між набором точок і місць, які необхідно відвідати. У формулюванні задачі точки — це міста, які може відвідати продавець. Мета продавця полягає в тому, щоб витрати на дорогу та пройдену відстань були якомога нижчими.

Орієнтований на оптимізацію, TSP часто використовується в інформатиці для пошуку найбільш ефективного маршруту для переміщення даних між різними вузлами. Програми включають визначення методів оптимізації мережі або обладнання. Вперше це було описано ірландським математиком **В. Р. Гамільтоном** і британським математиком **Томасом Кіркманом** у 1800-х роках шляхом створення гри, яку можна було розв’язати шляхом знаходження циклу Гамільтона, який є неперекриваючим шляхом між усіма вузлами.

TSP вивчали десятиліттями, і було запропоновано кілька рішень. Найпростіше рішення — випробувати всі можливості, але це також найбільш трудомісткий і дорогий метод. У багатьох рішеннях використовується евристика, яка забезпечує ймовірні результати. Однак результати є приблизними і не завжди оптимальними.

Замість того, щоб зосередитися на пошуку найефективнішого маршруту, TSP часто стурбований пошуком найдешевшого рішення. У TSP велика кількість змінних створює проблему під час пошуку найкоротшого маршруту, що робить приблизні, швидкі та дешеві рішення ще більш привабливими.



Маршрутизація фургона з пакетами до вашого порогу є класичним прикладом задачі комівояжера. Небагато компаній знають про це більше, ніж Amazon. Десятки тисяч їхніх водіїв щодня відправляються в дорогу, кожен з яких бере фургон Prime у тур TSP через 150 або більше зупинок для клієнтів, починаючи та закінчуючи на станції депо. А за лаштунками дослідницький підрозділ Amazon постійно шукає шляхи підвищення ефективності та безпеки своїх маршрутів.

4.2. Транспортна мережа.

Транспортна мережа — це мережа або графік у географічному просторі, що описує інфраструктуру, яка дозволяє та обмежує рух або потік. Приклади включають, але не обмежуються мережею доріг, залізницями, повітряними шляхами, трубопроводами, акведуками та лініями електропередач.

Основні визначення

Мережа є

- • зв'язним орієнтованим графом без петель.

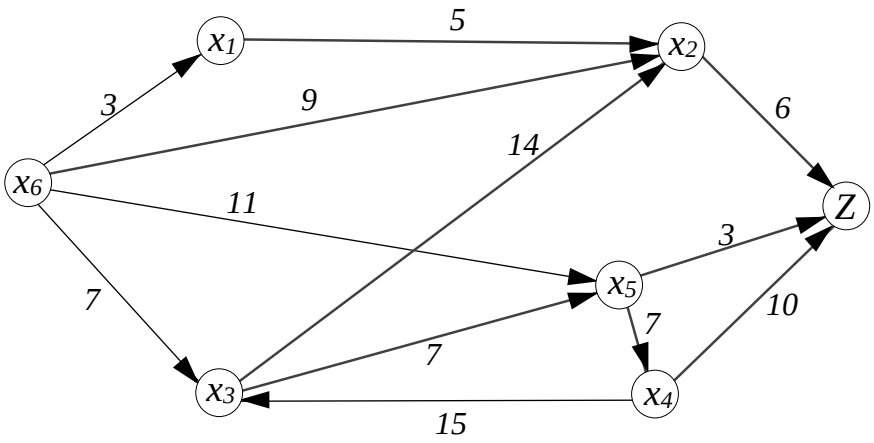
  Позначимо мережу через $G(X,U)$, де

  X – множина вершин,

  U – набір дуг.

- є лише одна вершина, в яку входить нульова кількість дуг. Ця вершина називається **джерелом**.

- є лише одна вершина z з нульовою кількістю дуг, що виходять з неї. Ця вершина називається **стоком**



Кожній дузі $u \in U$ додається невід'ємне дійсне число $C(u)$, яке називається **ємністю дуги**.

Вершина $x_i \neq x_0$; $x_i \neq z$ називаються **внутрішніми вершинами**

Течія в дузі є функцією $\varphi(u)$.

Властивості течії $\varphi(u)$:

1. $0 \leq \varphi(u) \leq C(u)$;
2. $\sum_{u \in U_x^-} \varphi(u) = \sum_{u \in U_x^+} \varphi(u)$, $x \neq x_0$; $x \neq z$;

де U_x^- множина дуг, що входять у вершину x; U_x^+ множина дуг, що виходять з вершини x.

Потік у дузі u можна розглядати як швидкість, з якою матеріал транспортується через дугу u, тому матеріал не накопичується у вершинах мережі.

Тому кількість матеріалу, який транспортується з джерела, дорівнює сумарній кількості матеріалу, який транспортується в стік.

3. $\sum_{u \in U_x^+} \varphi(u) = \sum_{u \in U_x^-} \varphi(u) = \varphi_z$,

де $\varphi(u)$ відповідає **значенню** течії або потоку в мережі.

Розглянемо завдання:

A – довільний набір вершин $x_0 \notin A$ та $z \in A$;

$\overline{A}$– доповнення набору A, $z \in A$, але $x_0 \notin \overline{A}$

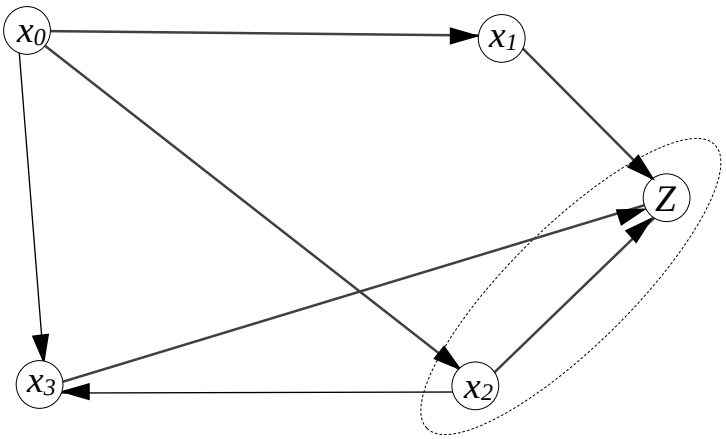
U_A^- – множина дуг, що входять до множини A;

U_A^+ – множина дуг, що йде до множини A.

Набір U_A^- називається **розрізом** мережі. Розріз містить дуги, які закінчуються множиною $\overline{A}$ або ординатою множини A.

Приклад:

$A=\{x_2, z\}$, $\overline{A}=\{x_0, x_1, x_3\}$ розріз $U_A^- = \{(x_1, z),(x_3, z),(x_0, x_2)\}$.



4. $\varphi_z = \sum_{u \in U_x^+} \varphi(u) - \sum_{u \in U_x^-} \varphi(u)$. Ємність розрізу - це сума ваг дуг, що належать цьому розрізу.

5. $\varphi_z \le C(A)$, де $C(A)$ – потужність розрізу A. Величина потоку в мережі не більше пропускної здатності будь-якого розрізу.

<u>Теорема Форда-Фолкерсона</u>

Нехай для величини мережевого потоку φ_z та потужності розрізу V виконується така рівність $\varphi_z = C(V)$: тоді φ_z є максимальний потік, який може пройти через мережу і V має мінімальну пропускну здатність з усіх розрізів у цій мережі.

Це V називається **мінімальним розрізом**.

Дуга u називається **насиченою**, якщо $\varphi(u) = C(u)$.

Дуга u називається **порожньою**, якщо $\varphi(u) = 0$.

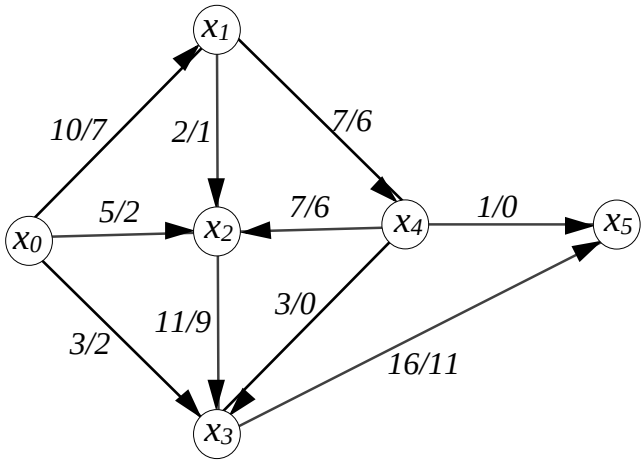
Дуга u називається **зайнятою**, якщо $\varphi(u) > 0$.

<u>Алгоритм складається з двох етапів:</u>

I - Знаходження деякого повного потоку;

II - Оцінка максимального потоку за заданими мітками до вершин мережі.

Нехай φ_z – деякий потік, розподілений по дугах мережі. Знайдіть повний потік, якщо кожен шлях від x_0 до z містить принаймні одну насичену дугу.

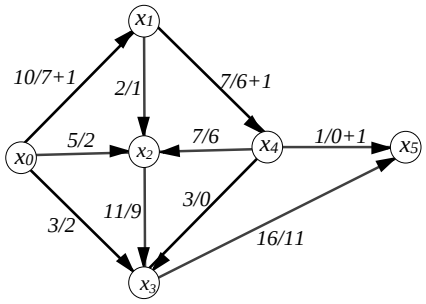


Етап I (алгоритм Форда-Фолкерсона)

Крок 1. Нехай Δ – різниця між пропускною здатністю кожної дуги та витратою в цій дузі.

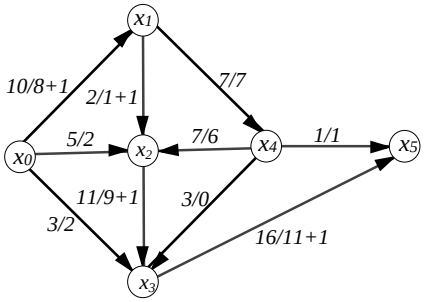
$\Delta = 1$. Дуги (x_1, x_4) і (x_4, z) насичені.

$\mu_1 : x_0, x_1, x_4, z;$ $\varphi_z = 12$



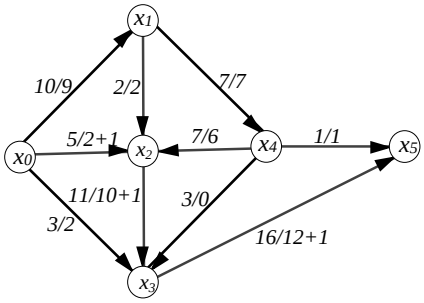
Крок 2. $\Delta = 1$. Дуга (x_1, x_2) насичена.

$\mu_2 : x_0, x_1, x_2, x_3, z;$ $\varphi_z = 13$



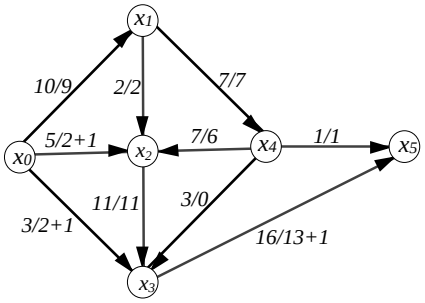
Крок 3. $\Delta = 1$. Дуга (x_2, x_3) насичена.

$\mu_3 : x_0, x_2, x_3, z;$ $\varphi_z = 14$



Крок 4. $\Delta = 1$. Дуга (x_0, x_3) насичена.

$\mu_4 : x_0, x_3, z;$ $\varphi_z = 15$ відповідає повному потоку, оскільки кожен шлях від x_0 до z містить принаймні одну насичену дугу.



Етап II (Процес маркування)

Процес збільшення потоку φ_u полягає в позначенні вершин мережі індексами, які вказують шлях, де можна змінити. Якщо така мітка може досягти стоку z, потік φ_u можна збільшити. Після цього вершини знову маркуються.

Алгоритм маркування вершин

1. $x_0[0] \leftarrow$ мітка;

2. Маркування вершини.

Якщо x_i вже має мітку, тоді мітка $[+i]$ призначається для всіх розмічених прилеглих до x_0 вершин, коли потік f_z проходить через незайняту дугу і напрямок потоку f_z збігається з напрямком розглянутої дуги. Мітка $[-i]$ призначається всім маркованим вершинам, які з'єднані з x_i зайнятою дугою з початком в x_i.

3. Якщо процес позначення вершин досягає вершини z, то переходимо до кроку 4, інакше потік, отриманий на попередньому кроці, мав максимальне значення.

4. Існує $x_0 - z$ шляхів μ із різними вершинами, позначеними (з точністю до знаку) кількістю попередніх вершин. Шлях μ будується, починаючи з вершини z.

Потік дуги φ_u відповідає:

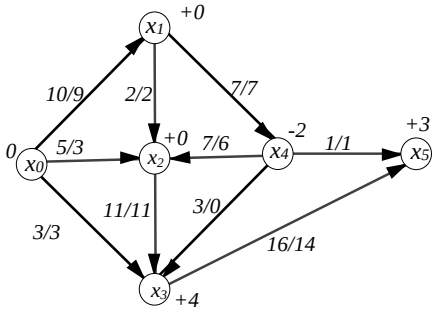
$$\varphi'_u = \begin{cases} \varphi_u, \; if \;\; u \notin \mu \\ \varphi_u + 1, \; якщо\ напрями\ потоку\ f(u)\ і\ дуги\ u\ збігаються \\ \varphi_u - 1, \; якщо\ напрями\ потоку\ f(u)\ і\ дуги\ u\ протилежні \end{cases}$$

$\varphi'_u = \varphi_u + 1$ отримано новий потік.

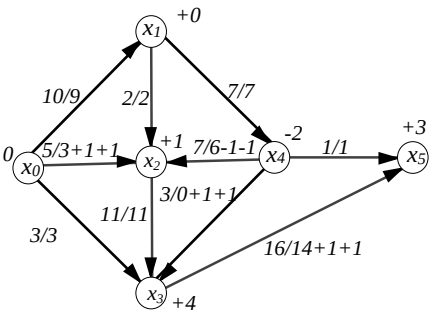
Переходимо до кроку 1.

Стік z -- маркована.

Шлях: x_0, x_2, x_3, x_4, z де потік може бути збільшений.



Збільшують на одиницю потік в таких дугах: $(x_0, x_2), (x_4, x_3), (x_3, z)$.



Зменшують на одиницю потік в дузі (x_4, x_2).

Шлях $\mu_4: x_0, x_2, x_4, x_3, z$

де потік може бути збільшений.

Дозволено потік φ_u збільшувати в дугах $(x_0, x_1), (x_1, x_2), (x_4, x_3), (x_3, z)$ і зменшувати в (x_4, x_2).

Вершина z не позначена (маркована), тоді отриманий на попередньому кроці потік був максимальним. $\varphi_z = 16$ – максимальний потік.

Список літератури

1. Балога С.І. «Дискретна математика». Навч.посібник. Ужгород: ПП «АУТДОР-ШАРК», 2021. 124 с.
2. Ліхоузова Т.А. «Дискретна математика. Практикум». Київ: КПІ ім. Ігоря Сікорського, 2020. 60 с.
3. Матвієнко М.П. Дискретна математика. Суми. 2019. 324 с.
4. Jacques Verstraete. Introduction to Graph Theory. A short course in graph theory at UCSD. February 18, 2020. Department of Mathematics University of California at San Diego California, U.S.A.

Навчальне видання

Кагадій Тетяна Станіславівна
Шпорта Анна Григорівна
Онопрієнко Олег Дмитрович

ОСНОВИ ДИСКРЕТНОЇ МАТЕМАТИКИ
(частина 2)

Методичні рекомендації
до опанування лекційних занять
з дисципліни «Дискретна математика» здобувачами ступеня бакалавра спеціальності 113 Прикладна математика

Видано в авторській редакції.

Електронний ресурс.
Підписано до видання 01.10.2024. Авт. арк. 2,0.

Національний технічний університет «Дніпровська політехніка».
49005, м. Дніпро, просп. Дмитра Яворницького,19.