Міністерство освіти і науки України

Національний університет водного господарства та природокористування

Кафедра обчислювальної техніки

**04-04-268М**

# МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни
**«Мережеві алгоритми оптимізації»**
для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Комп’ютерна інженерія» спеціальності 123 «Комп’ютерна інженерія»
денної та заочної форм навчання

Рекомендовано
науково-методичною радою
з якості ННІП
Протокол № 2 від 30.10.2023 р.

Рівне – 2023

Методичні вказівки до лабораторних робіт з навчальної дисципліни «Мережеві алгоритми оптимізації» для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Комп’ютерна інженерія» спеціальності 123 «Комп’ютерна інженерія» денної та заочної форм навчання навчання. [Електронне видання] / Соломко М. Т. – Рівне : НУВГП. – 55 с.

Укладач: Соломко М. Т., доцент кафедри обчислювальної техніки.

Відповідальний за випуск: Круліковський Б. Б., доцент, завідувач кафедри обчислювальної техніки.

Керівник групи забезпечення спеціальності
123 «Комп’ютерна інженерія» Круліковський Б. Б.

## ЗМІСТ

## Вступ

Навчальна дисципліна «Мережеві алгоритми оптимізації» є базовою для студентів спеціальності 123 "Компʼютерна інженерія" і тому вимагає ретельного вивчення на лабораторних та лекційних заняттях з впровадженням сучасних технологій навчання, максимально наближених до реальних мережевих задач.

У зв'язку зі змінами в економічних системах та зростаючою конкуренцією господарювання виникає потреба в оптимізаційному використанні результатів дослідження в галузі синтезу та аналізу об'єктів самої різної природи, що мають мережеву структуру. До таких об'єктів, зокрема, належать і численні територіальні системи: інформаційні, транспортні, енергетичні тощо. Виявилося, що математичні моделі досить адекватно описують такі системи на сильно агрегованому рівні, є в певному сенсі універсальними. Насправді хорошим описом різних комунікаційних мереж для отримання багатьох техніко-економічних характеристик слугує зважений граф, ребрам і вершинам якого приписуються «ваги», що мають зазвичай зміст відповідно пропускних здібностей і потреб (виробничих потужностей).

Для таких зважених графів формуються завдання двох типів:

1) оцінити (точно, наближено чи гарантовано) ті чи інші функціонали, задані цих графах;

2) при фіксованих вагах вершин синтезувати такі ваги на ребрах графа (при заздалегідь фіксованій структурі або при фіксації класу дозволених структур), щоб реалізовувалося рішення між витоками та стоками графа при досягненні екстремуму функціоналу, заданого на багатьох ребрах цього графа. В останньому випадку можлива постановка для зворотного завдання.

Дисципліна «Мережеві алгоритми оптимізації» відноситься до циклу оптимізаційних інструментів мережевих задач. В результаті вивчення дисципліни студенти повинні знати:

– основні поняття, що використовуються у мережевому моделюванні;

– способи зведення прикладних завдань до потокових задач;

– способи подання, зберігання та перетворення даних;

- основні мережеві алгоритми ;

- технологію вирішення оптимізаційних мережевих завдань на ЕОМ.

- пошук найкоротшого шляху; визначення максимального потоку; пропускну спроможність ділянки дороги; потік мінімальної вартості шляхів.

Представлені відомості про порядок виконання лабораторних робіт та захист звітів.

Для кожної лабораторної роботи вказано порядок підготовки, методика досліджень, інформаційний зміст, який має бути у звіті з лабораторної роботи, а також наведено орієнтовний перелік контролюючих запитань для перевірки засвоєних знань.

## 1. Загальні методичні вказівки

Перед початком першого заняття викладач проводить інструктаж з техніки безпеки у даній лабораторії і правилами протипожежної безпеки при роботі з електричними колами, приладами, й лабораторними стендами. Кожен студент повинен самостійно вивчити пераховані документи і розписом у спеціальному журналі кафедри засвідчити своє ознайомлення з правилами і заходами безпечного виконання робіт у лабораторії.

На першому занятті викладач повідомляє студентам план лабораторних занять на поточний семестр, рекомендує їм необхідну літературу, знайомить із прийнятою методикою підготовки, виконання, а також з порядком захисту звітів з виконаних лабораторних роботах.

Виконання кожної лабораторної роботи складається з трьох етапів:

1. Підготовка до лабораторної роботи, вивчення теоретичного матеріалу, підготовка заготовки звіту з описом назви, мети, досліджуваними функціями та переліком запланованих досліджень роботи, заготовками таблиць для запису експериментальних даних. Перевірку готовності до виконання лабораторної роботи студенти проходять перед її початком. При цьому перевіряється знання досліджуваної моделі, її призначення, вхідні та вихідні дані, основні характеристик. У випадку незадовільної підготовки студенти не допускаються до проведення лабораторної роботи. У процесі підготовки до лабораторної роботи студент повинен чітко усвідомити мету лабораторного дослідження, представлений об’єкт дослідження.

2. Виконання роботи після отримання допуску студентом починається із складання плану дослідження у відповідності із ходом роботи.

3. Оформлення звіту з лабораторної роботи здійснюється в домашніх умовах. Звіт, крім попередніх даних, повинен містити результати виконання лабораторної роботи: таблиці з переліком робіт за проектом, мережеві графіки проекту, аналіз і порівняння отриманих результатів з теоретичними відомостями, пояснення їх відмінностей (при наявності).

Порядок, виконання досліджень у лабораторії:

1. Студент допускається до виконання чергової лабораторної роботи при наявності підготовленої до поточного заняття заготовки та при відсутності нezданих звітів з попередніх робіт (2 і більше).

2. Після дозволу виконувати дослідження студент виконує намічені дослідження, по закінченню яких результати пред'являються викладачеві.

3. До протоколу поточної роботи заносяться результати, що отримані студентом на занятті, підписуються викладачем, якщо дослідження виконані в повному обсязі. За отриманими даними оформляється остаточний звіт з роботи.

4. До наступної лабораторної роботи остаточно оформляється протокол і пред'являється викладачеві на наступному занятті для захисту.

5. Протоколи всіх робіт зберігаються у студента до виконання всього лабораторного циклу та використовуються для підготовки до екзамену.

**2. Програмне забезпечення для проведення лабораторних робіт .**

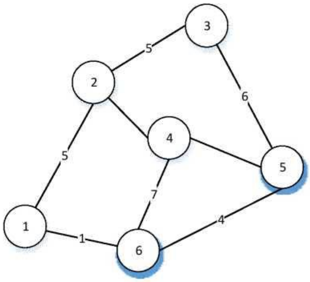
Лабораторні роботи проводяться у текстовому процесорі MS Word 16.0 або вищої версії.

# Лабораторна робота №1

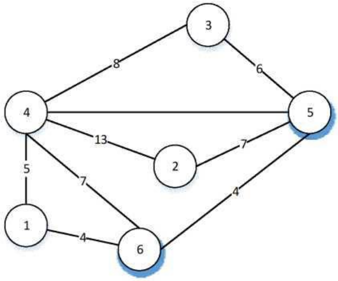
## Алгоритм Дейкстри

### Завдання та порядок виконання роботи

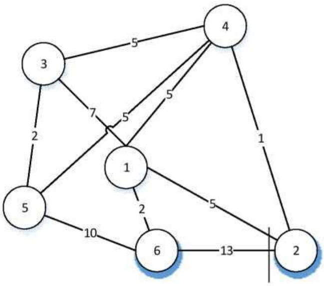
1. Використовуючи алгоритмом Дейкстри, знайти мінімальний шлях з вершини $x_1$ до вершини $x_6$ в орієнтованому графі, що заданий матрицею ваг.


3
5
2
6
4
5
5
7
1
1
6
4

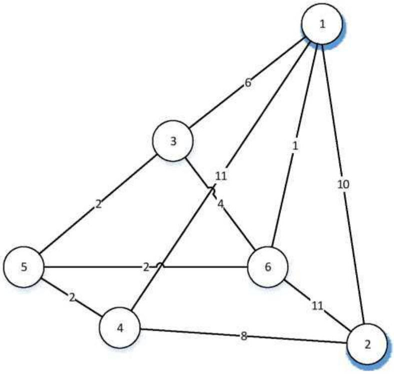
1


3
8
6
4
5
13
7
2
5
7
4
1
4
6

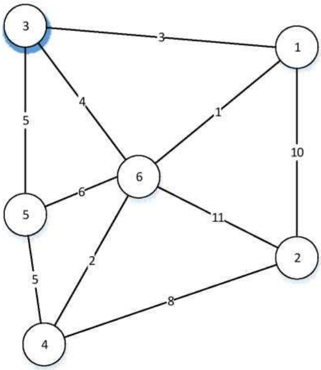
2


4
3
5
7
5
5
1
2
1
5
2
5
10
6
13
2

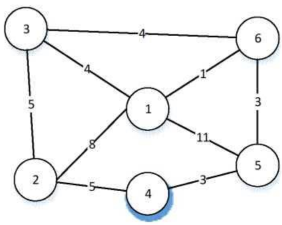
3


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

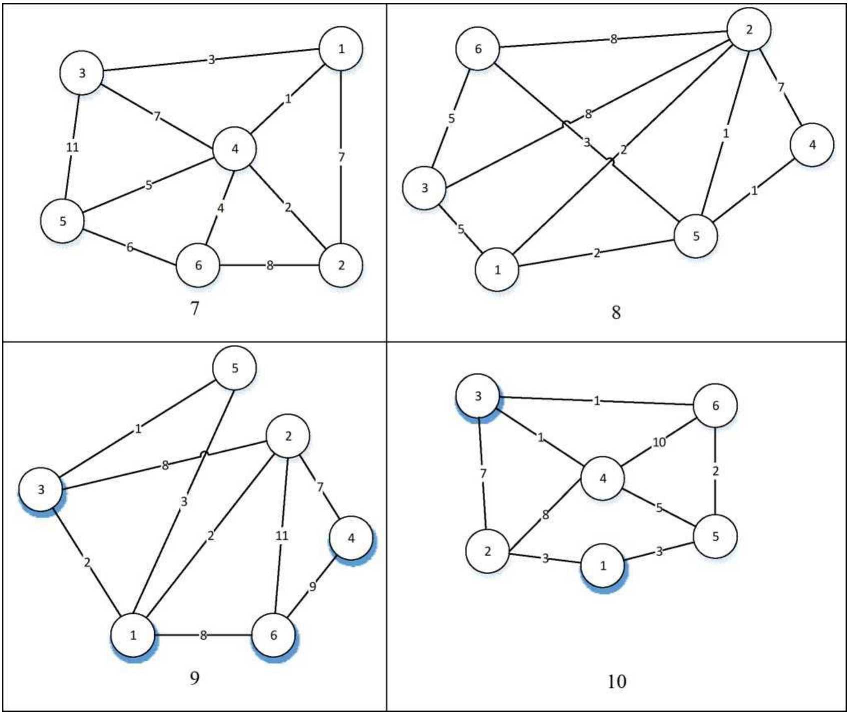
4


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

5


3
4
6
4
1
5
1
3
11
8
2
5
4
3
5

6




Контрольні питання

1. Що таке граф?

2. Які є види графів ?

3. Як можуть задаватися графи?

4. Що таке вершина у графі?

5. Що таке ребро у графі?

6. Які існують алгоритми для пошуку мінімальної відстані у графі?

7. Алгоритм Дейкстри.

8. Опишіть класичну реалізацію алгоритму Дейкстри.

Зміст звіту

1. Номер роботи, її назва та мета.
2. Короткі теоретичні відомості та відповіді на контрольні питання.
3. Згідно з варіантом знайти мінімальний шлях з вершини $x_1$ до вершини $x_6$ в орієнтованому графі.
4. Висновки.

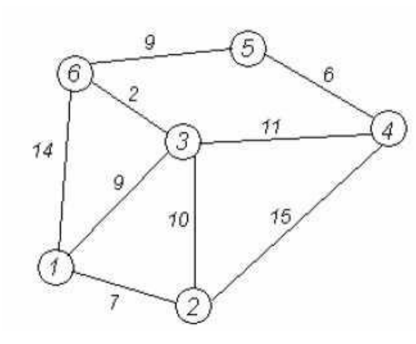
## 1. Мета роботи

Навчитися реалізовувати алгоритм Дейкстри для пошуку найкоротшого шляху від однієї вершини графа до інших.

## 2. Теоретичні відомості

Алгоритм Дейкстри – алгоритм на графах, знаходить найкоротший шлях від однієї вершини графа до всіх інших вершин. Класичний алгоритм Дейкстри застосовується тільки для графів без циклів від'ємної довжини.

**Способи задання графу:**




$$
\begin{matrix}
0 & 7 & 9 & 0 & 0 & 14 \\
7 & 0 & 10 & 17 & 0 & 0 \\
9 & 10 & 0 & 11 & 0 & 2 \\
0 & 17 & 11 & 0 & 6 & 0 \\
0 & 0 & 0 & 6 & 0 & 9 \\
14 & 0 & 2 & 0 & 9 & 0
\end{matrix}
$$

Графічно

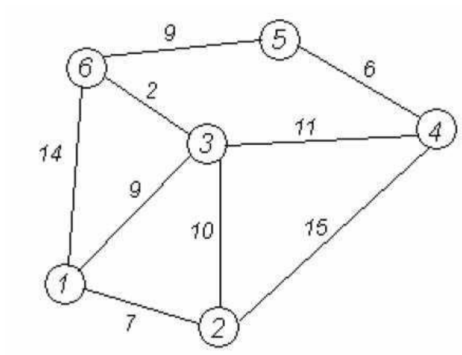
Матрично

## Приклад алгоритму Дейкстри:

Дано неорієнтований зв'язний граф **G(V, U)**. Знайти відстань від вершини **a** до всіх інших вершин **V**.

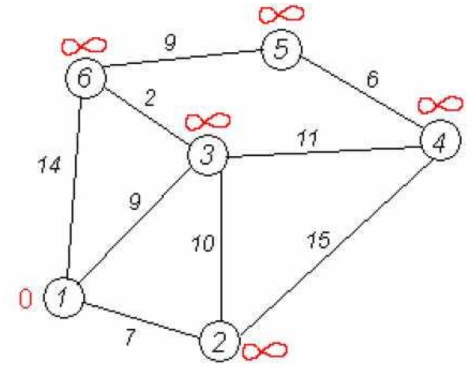
Зберігатимемо поточну мінімальну відстань до всіх вершин **V** (від даної вершини **a**) і на кожному кроці алгоритму намагатимемося зменшити цю відстань. Спочатку встановимо відстані до всіх вершин рівними нескінченості, а до вершини **a** — нулю.

Хай потрібно знайти відстані від 1-ої вершини до всіх інших. Кружечками позначені вершини, лініями — шляхи між ними («дуги»). Над дугами записана їх «ціна» — довжина шляху. Надписом над кружечком позначена поточна найкоротша відстань до вершини.



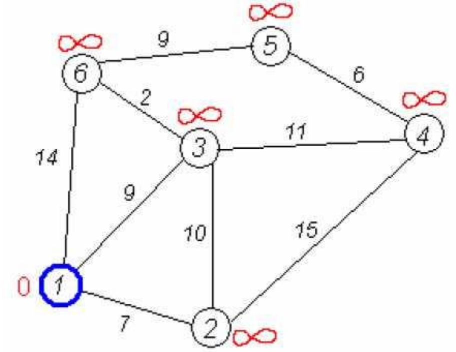
Крок 1

Ініціалізація. Відстань до всіх вершин графа **V =** Відстань до **a =0**. Жодна вершина графа ще не опрацьована.



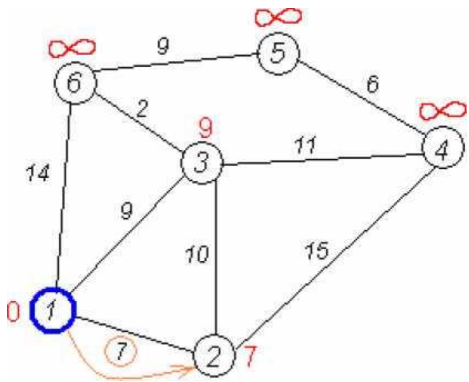

Крок 2

Находимо таку вершину (із ще не оброблених), поточна найкоротша відстань до якої мінімальна. В нашому випадку це вершина 1. Обходимо всіх її сусідів і, якщо шлях в сусідню вершину через 1 менший за поточний мінімальний шлях в цю сусідню вершину, то запам'ятовуємо цей новий, коротший шлях як поточний найкоротший шлях до сусіда



Крок 3

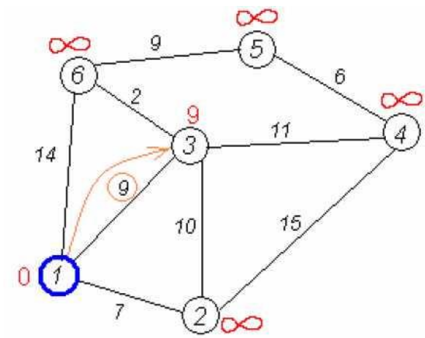
Перший по порядку сусід 1-ї вершини — 2-а вершина. Шлях

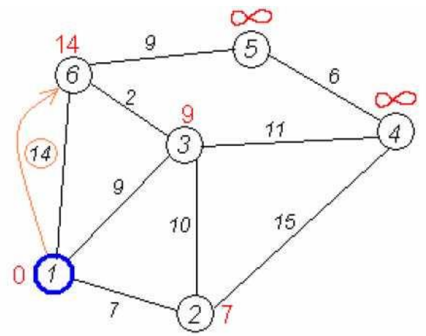



до неї через 1- у вершину дорівнює найкоротшій відстані до 1-ї вершини + довжина дуги між 1-ю та 2-ю вершиною, тобто 0 + 7 = 7. Це менше поточного найкоротшого шляху до 2-ї вершини, тому найкоротший шлях до 2-ї вершини дорівнює 7.

Кроки 4, 5

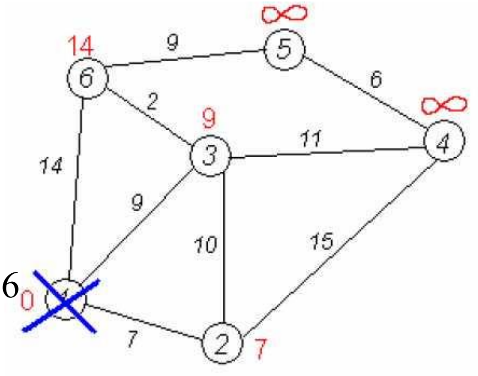
Аналогічну операцію проробляєм з двома іншими сусідами 1 -ї вершини — 3-ю та 6-ю.



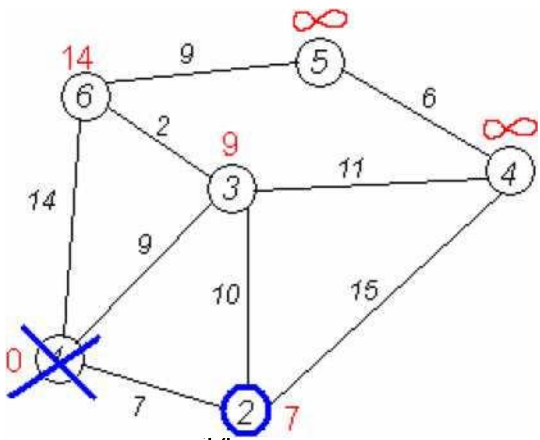
Всі сусіди вершини 1 перевірені. Поточна мінімальна відстань до вершини 1 вважається остаточною і обговоренню не підлягає (те, що це дійсно так, вперше довів Дейкстра). Тому викреслимо її з графа, щоб відмітити цей факт.

Крок 6




Крок 7

Практично відбувається повернення до кроку 2. Знову знаходимо «найближчу» необроблену (невикреслену) вершину. Це вершина 2 з поточною найкоротшою відстанню до неї = 7. І знову намагаємося зменшити відстань до всіх сусідів 2-ої вершини, намагаючись пройти в них через 2-у. Сусідами 2-ої вершини є 1, 3, 4.

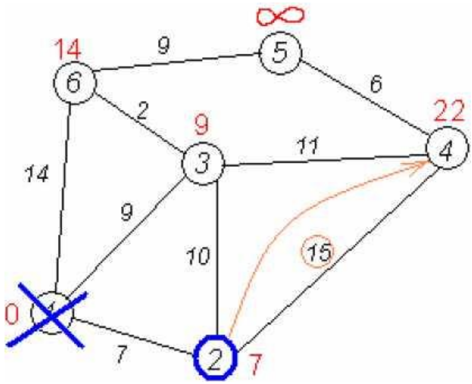


Крок 8

Перший (по порядку) сусід вершини № 2 — 1-а вершина. Але вона вже оброблена (або викреслена — див. крок 6). Тому з 1-ою вершиною нічого не робимо.

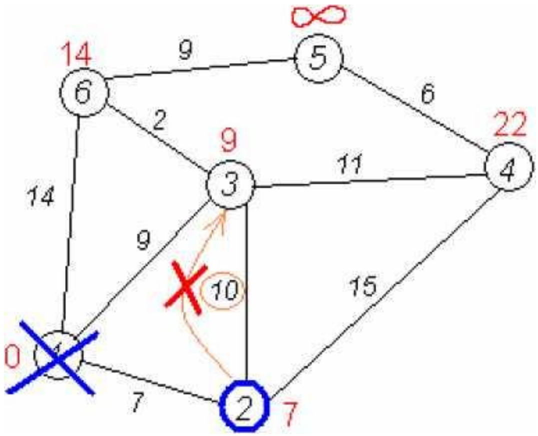
Крок 8 (з іншими вхідними даними)

Інший сусід вершини 2 — вершина 4. Якщо йти в неї через 2-у, то шлях буде = найкоротша відстань до 2-ої + відстань між 2-ою і 4-ою вершинами = 7 + 15 = 22. Оскільки 22 < да, встановлюємо відстань до вершини № 4 рівним 22.



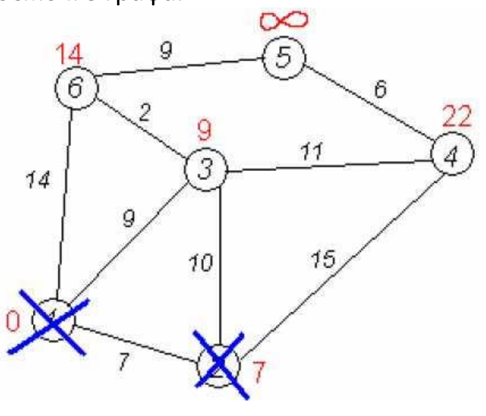

Крок 9

Ще один сусід вершини 2 — вершина 3. Якщо йти в неї через 2-у, то шлях буде = 7 + 10 = 17. Але 17 більше за відстань, що вже запам'ятали раніше до вершини № 3 і дорівнює 9, тому поточну відстань до 3-ої вершини не міняємо.



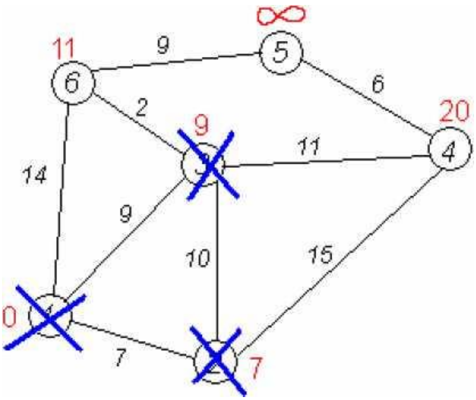

Крок 10

Всі сусіди вершини 2 переглянуті, заморожуємо відстань до неї і викреслюємо її з графа.



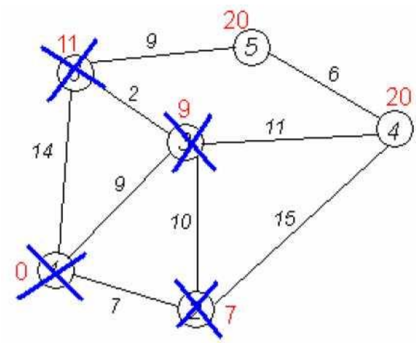

Кроки 11 — 15

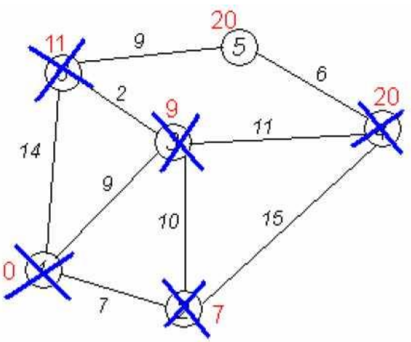
По вже «відпрацьованій» схемі повторюємо кроки 2 — 6. Тепер «найближчою» виявляється вершина № 3. Після її «обробки» отримаємо такі результати:

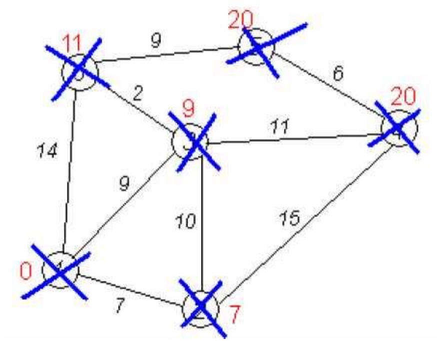



Наступні кроки

Проробляємо те саме з вершинами, що залишилися (№ по порядку: 6, 4 і 5).



Завершення виконання алгоритму

Алгоритм закінчує роботу, коли викреслені всі вершини. Результат його роботи видно на останньому малюнку: найкоротший шлях від 1-ої вершини до 2-ої становить 7, до 3-ої — 9, до 4-ої — 20, до 5-ої — 20, до 6-ої — 11 умовних одиниць.

## Література


1. Нікольський Ю. В. Дискретна математика : підручник. Львів : Магнолія Плюс, 2005. 608 с.
2. Андрійчук В. І., Комарницький М. Я. Вступ до дискретної математики : навч. посіб. К. : Центр, 2004. 254 с.
3. Бардачов Ю. М., Соколова І. А., Ходакова В. Є. Дискретна математика : підручник. К. : Вища школа, 2002. 287 с.
4. Основи дискретної математики / Капітонова Ю. В. та ін. К. : Наукова думка, 2002. 578 с.
5. Р. Хаггарти. Дискретная математика для программистов. «Техносфера» 2012. 317 с.
6. Бондаренко М. Ф., Білоус Н. В., Руткас А. Г. Комп’ютерна дискретна математика : підручник. Харків, 2004. 480 с.


# Лабораторна робота № 2

## Алгоритм Форда-Беллмана знаходження мінімального шляху у графі

Передбачається, що орієнтований граф не містить контурів від’ємної довжини.

**Завдання та порядок виконання роботи**

1. Використовуючи алгоритмом Форда-Беллмана, знайти мінімальний шлях з вершини $x_1$ до вершини $x_7$ в орієнтованому графі, що заданий матрицею ваг.

| | |
|---|---|
| 1 $\begin{pmatrix} \infty & 4 & 6 & 12 & \infty & \infty & \infty \\ \infty & \infty & \infty & 13 & 7 & \infty & \infty \\ \infty & \infty & \infty & 5 & \infty & 3 & \infty \\ \infty & \infty & \infty & \infty & 10 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 8 \\ \infty & \infty & \infty & \infty & \infty & \infty & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ | 2 $\begin{pmatrix} \infty & 1 & 3 & 9 & \infty & \infty & \infty \\ \infty & \infty & \infty & 10 & 4 & \infty & \infty \\ \infty & \infty & \infty & 2 & \infty & 1 & \infty \\ \infty & \infty & \infty & \infty & 7 & 6 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & 8 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ |
| 3 $\begin{pmatrix} \infty & 3 & 5 & 11 & \infty & \infty & \infty \\ \infty & \infty & \infty & 12 & 6 & \infty & \infty \\ \infty & \infty & \infty & 3 & \infty & 2 & \infty \\ \infty & \infty & \infty & \infty & 9 & 8 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty & 10 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ | $\begin{pmatrix} \infty & \infty & 5 & 4 & 2 & 2 & 9 \\ \infty & \infty & 1 & 1 & \infty & 1 & 1 \\ 2 & \infty & \infty & 1 & 1 & \infty & 3 \\ \infty & 2 & 1 & \infty & 1 & \infty & \infty \\ \infty & \infty & 2 & 2 & \infty & 1 & 6 \\ 1 & 5 & \infty & 1 & 1 & \infty & \infty \\ 2 & \infty & 1 & \infty & 1 & 2 & \infty \end{pmatrix}$ 4 |
| $\begin{pmatrix} \infty & 4 & \infty & \infty & 3 & 1 & \infty \\ 3 & \infty & 2 & 1 & \infty & \infty & 4 \\ 1 & 1 & \infty & \infty & \infty & \infty & 1 \\ \infty & 3 & 1 & \infty & 1 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 1 & 5 \\ \infty & 3 & \infty & 2 & 2 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 2 & \infty \end{pmatrix}$ 5 | 6 $\begin{pmatrix} \infty & \infty & 9 & \infty & \infty & 2 & 12 \\ 1 & \infty & \infty & \infty & 1 & 2 & 4 \\ 2 & 1 & \infty & \infty & 1 & \infty & 2 \\ \infty & 1 & 1 & \infty & \infty & 1 & \infty \\ 1 & 2 & \infty & 2 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 1 & \infty & 8 \\ \infty & 2 & 1 & \infty & 1 & 2 & \infty \end{pmatrix}$ |

| | |
|---|---|
| 7 $\begin{pmatrix} \infty & 3 & 4 & 9 & \infty & \infty & \infty \\ 12 & \infty & \infty & 10 & 4 & \infty & \infty \\ \infty & \infty & \infty & 2 & \infty & 1 & \infty \\ \infty & \infty & \infty & \infty & 7 & 6 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & 8 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ | 8 $\begin{pmatrix} \infty & 2 & 5 & 8 & 9 & \infty & \infty \\ \infty & \infty & \infty & 10 & 4 & \infty & \infty \\ 5 & 3 & \infty & 2 & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & 7 & 6 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ |
| 9 $\begin{pmatrix} 11 & \infty & \infty & 12 & 6 & \infty & \infty \\ \infty & \infty & \infty & 3 & \infty & 2 & \infty \\ \infty & \infty & \infty & \infty & 9 & 8 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty & 10 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ | 10 $\begin{pmatrix} \infty & \infty & 5 & 4 & 2 & 3 & 9 \\ \infty & \infty & 1 & 1 & \infty & 1 & 6 \\ 4 & \infty & \infty & 1 & 1 & \infty & 3 \\ \infty & 2 & 1 & \infty & 1 & \infty & \infty \\ \infty & \infty & 2 & 2 & \infty & 1 & 6 \\ 1 & 5 & \infty & 1 & 1 & \infty & \infty \\ 2 & \infty & 1 & \infty & 1 & 2 & \infty \end{pmatrix}$ |
| 11 $\begin{pmatrix} \infty & 4 & 9 & \infty & 3 & 1 & \infty \\ 3 & \infty & 2 & 1 & \infty & \infty & 4 \\ 1 & 1 & \infty & \infty & 10 & \infty & 1 \\ \infty & 3 & 1 & \infty & 1 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 1 & 5 \\ \infty & 3 & \infty & 1 & 2 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 2 & \infty \end{pmatrix}$ | 12 $\begin{pmatrix} \infty & \infty & 9 & \infty & 10 & 2 & 12 \\ 1 & \infty & \infty & \infty & 1 & 2 & 4 \\ 2 & 1 & \infty & \infty & 1 & \infty & 2 \\ \infty & 1 & 1 & \infty & \infty & 1 & 15 \\ 1 & 2 & \infty & 2 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 1 & \infty & 8 \\ \infty & 2 & 1 & \infty & 1 & 2 & \infty \end{pmatrix}$ |

| | |
|---|---|
| 13 $\begin{pmatrix} \infty & 5 & 6 & 15 & \infty & \infty & \infty \\ \infty & \infty & \infty & 13 & 7 & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty & 3 & \infty \\ \infty & \infty & \infty & \infty & 10 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 8 \\ \infty & \infty & \infty & \infty & \infty & \infty & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ | 14 $\begin{pmatrix} \infty & 2 & 3 & 9 & \infty & \infty & \infty \\ 12 & \infty & \infty & 10 & 4 & \infty & \infty \\ \infty & \infty & \infty & 2 & \infty & 1 & \infty \\ \infty & \infty & \infty & \infty & 7 & 6 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & 8 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ |
| 15 $\begin{pmatrix} \infty & 2 & 5 & 10 & \infty & \infty & \infty \\ \infty & \infty & \infty & 12 & 6 & \infty & \infty \\ \infty & \infty & \infty & 3 & \infty & 1 & \infty \\ \infty & \infty & \infty & \infty & 9 & 8 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty & 10 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ | 16 $\begin{pmatrix} \infty & \infty & 5 & 4 & 2 & 2 & 10 \\ \infty & \infty & 2 & 1 & \infty & 2 & 1 \\ 2 & \infty & \infty & 1 & 1 & \infty & 3 \\ \infty & 2 & 1 & \infty & 1 & \infty & \infty \\ \infty & \infty & 2 & 2 & \infty & 1 & 6 \\ 1 & 5 & \infty & 1 & 1 & \infty & \infty \\ 2 & \infty & 1 & \infty & 1 & 2 & \infty \end{pmatrix}$ |
| 17 $\begin{pmatrix} \infty & 4 & 9 & 8 & 3 & 1 & \infty \\ 3 & \infty & 2 & 1 & \infty & \infty & 4 \\ 1 & 1 & \infty & \infty & \infty & \infty & 1 \\ \infty & 3 & 1 & \infty & 1 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 1 & 5 \\ \infty & 3 & \infty & 2 & 2 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 2 & \infty \end{pmatrix}$ | 18 $\begin{pmatrix} \infty & \infty & 9 & \infty & 10 & 2 & 12 \\ 1 & \infty & \infty & \infty & 1 & 2 & 4 \\ 2 & 1 & \infty & \infty & 1 & \infty & 2 \\ \infty & 1 & 1 & \infty & \infty & 1 & \infty \\ 1 & 2 & 9 & 2 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 1 & \infty & 8 \\ \infty & 2 & 1 & \infty & 1 & 2 & \infty \end{pmatrix}$ |
| 19 $\begin{pmatrix} \infty & 3 & 5 & 12 & 20 & \infty & \infty \\ \infty & \infty & \infty & 13 & 8 & \infty & \infty \\ \infty & \infty & \infty & 5 & \infty & 3 & \infty \\ \infty & \infty & \infty & \infty & 10 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 8 \\ \infty & \infty & \infty & \infty & \infty & \infty & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ | 20 $\begin{pmatrix} \infty & 1 & 5 & 7 & 9 & \infty & \infty \\ \infty & \infty & \infty & 10 & 4 & \infty & \infty \\ 5 & 3 & \infty & \infty & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & 7 & 6 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 4 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ |

| | |
|---|---|
| 21 $\begin{pmatrix} \infty & 1 & 5 & 15 & \infty & \infty & \infty \\ \infty & \infty & 11 & 12 & 6 & \infty & \infty \\ \infty & \infty & \infty & 3 & \infty & 2 & \infty \\ \infty & \infty & \infty & \infty & 9 & 8 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty & 10 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$ | 22 $\begin{pmatrix} \infty & \infty & 3 & 4 & 1 & 5 & 9 \\ \infty & \infty & 1 & 2 & \infty & 1 & 6 \\ 4 & \infty & \infty & 2 & 1 & \infty & 3 \\ \infty & 2 & 3 & \infty & 1 & \infty & \infty \\ \infty & \infty & 2 & 2 & \infty & 1 & 6 \\ 1 & 5 & \infty & 1 & 1 & \infty & \infty \\ 2 & \infty & 1 & \infty & 1 & 2 & \infty \end{pmatrix}$ |
| 23 $\begin{pmatrix} \infty & 4 & 9 & \infty & 3 & 1 & \infty \\ 3 & \infty & 2 & 1 & \infty & \infty & 4 \\ 1 & 1 & \infty & \infty & 10 & 14 & 1 \\ \infty & 3 & 1 & \infty & 1 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 1 & 5 \\ \infty & 3 & \infty & 1 & 2 & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 2 & \infty \end{pmatrix}$ | 24 $\begin{pmatrix} \infty & \infty & 8 & \infty & 10 & 3 & 12 \\ 1 & \infty & \infty & \infty & 1 & 2 & 3 \\ 2 & 1 & \infty & \infty & 1 & \infty & 2 \\ \infty & 1 & 1 & \infty & \infty & 1 & 15 \\ 1 & 2 & \infty & 2 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 1 & \infty & 8 \\ \infty & 2 & 1 & \infty & 1 & 2 & \infty \end{pmatrix}$ |

Контрольні питання

1. Що таке граф?
2. Які є види графів ?
3. Як можуть задаватися графи?
4. Що таке вершина у графі?
5. Що таке ребро у графі?
6. Які існують алгоритми для пошуку мінімальної відстані у графі?
7. Алгоритм Форда-Беллмана.
8. Опишіть класичну реалізацію алгоритму Форда-Беллмана.

Зміст звіту

1. Номер роботи, її назва та мета.
2. Короткі теоретичні відомості та відповіді на контрольні питання.
3. Згідно з варіантом знайти мінімальний шлях з вершини $x_1$

до вершини $x_7$ в орієнтованому графі, що заданий матрицею ваг.

4. Висновки.

**1. Мета роботи.** Навчитися реалізовувати алгоритм Форда-Беллмана для пошуку найкоротшого шляху від однієї вершини графа до інших.

## 2. Теоретичні відомості

### Алгоритм Форда-Беллмана

Основними величинами цього алгоритму, що обчислюються, є величини $\lambda_j(k)$, де i = 1, 2, … , n (n – число вершин графа); k = 1, 2, … n – 1. Для фіксованих i і k величина $\lambda_j(k)$ дорівнює довжині мінімального шляху, що веде із заданої початкової вершини $x_1$ у вершину $x_i$ і містить не більше k дуг.

*Крок* 1. *Встановлення початкових умов*.

Ввести число вершин графа n та матрицю ваг $C = (c_{ij})$.

*Крок* 2. Покласти k = 0. Покласти $\lambda_i(0) = \infty$ для всіх вершин, крім $x_1$; покласти $\lambda_i(0) = 0$.

*Крок* 3. У циклі по k, k = 1,..., n – 1, кожній вершині $x_i$ на k-му кроці приписати індекс λi(k) за таким правилом:

$$\lambda_i(k) = \min_{1\le j\le n} \{\lambda_j(k-1) + c_{ji}\} \tag{2.1}$$

всім вершин, крім $x_1$, покласти $\lambda_i(k) = 0$. У результаті роботи алгоритму формується таблиця індексів $\lambda_i(k)$, i = 1, 2, … , n, k = 0, 1, 2, … , n – 1 $\lambda_i(k)$ визначає довжину мінімального шляху з першої вершини в i-у, що містить не більше k дуг.

*Крок* 4. Відновлення мінімального шляху.

Для будь-якої вершини $x_s$ попередня їй вершина $x_r$ визначається із співвідношення:

$$\lambda_r(n-2) + c_{rs} = \lambda_s(n-1), \qquad x_r \in G^{-1}(x_s), \tag{2.2}$$

де $G^{-1}(x_s)$ – прообраз вершини $x_s$.

Для знайденої вершини $x_r$ попередня їй вершина $x_q$ визначається із співвідношення:

$$\lambda_q(n-3) + c_{qr} = \lambda_r(n-2), \qquad x_q \in G^{-1}(x_r),$$

де $G^{-1}(x_r)$ - прообраз вершини $x_r$ і т.д.

Послідовно застосовуючи це співвідношення, починаючи від останньої вершини $x_i$, знайдемо мінімальний шлях.

**Приклад виконання завдання**

За допомогою алгоритму Форда-Беллмана знайдемо мінімальний шлях з вершини $x_1$ до вершини $x_3$ у графі, що зображений на рис. 1.

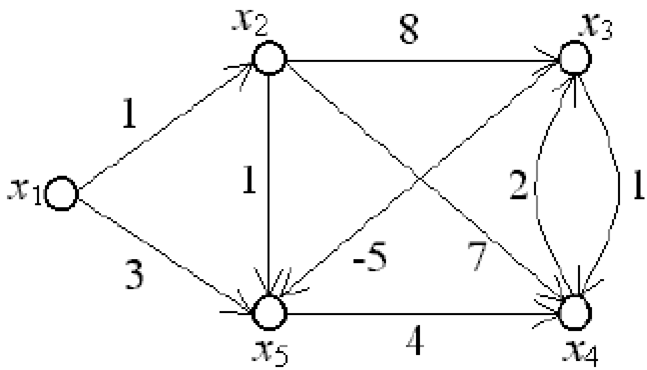



Рис. 1

Розглянемо роботу алгоритму Форда-Беллмана для цього прикладу.

Значення індексів $\lambda_i(k)$ заноситимемо до таблиці індексів (табл. 2.1).

*Крок* 1. Введемо число вершин графа n =5. Матриця терезів цього графа має вигляд:

$$C = \begin{pmatrix} \infty & 1 & \infty & \infty & 3 \\ \infty & \infty & 8 & 7 & 1 \\ \infty & \infty & \infty & 1 & -5 \\ \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty \end{pmatrix}.$$

*Крок* 2. Покладемо k = 0, λ1(0) = 0, λ2(0) = λ3(0) = λ4(0) = λ5(0) = ∞. Ці значення занесемо у перший стовпець табл. 2.1.

*Крок* 3.

<u>*k = 1.*</u>
*$\lambda_1(1) = 0$.*

Рівність (2.1) для k = 1 має вигляд:

$$\lambda_i(1) = \min_{1 \le j \le 5} \{\lambda_j(0) + c_{ji}\}.$$

$\lambda_2(1) = min\{\lambda_1(0) + c_{12};\ \lambda_2(0) + c_{22};\ \lambda_3(0) + c_{32};\ \lambda_4(0) + c_{42};\ \lambda_5(0) + c_{52};\} = min\{0 + 1; \infty + \infty; \infty + \infty; \infty + \infty; \infty + \infty\} = 1.$

$\lambda_3(1) = min\{\lambda_1(0) + c_{13};\ \lambda_2(0) + c_{23};\ \lambda_3(0) + c_{33};\ \lambda_4(0) + c_{43};\ \lambda_5(0) + c_{53};\} = min\{0 + \infty ; \infty + 8; \infty + \infty; \infty + 2; \infty + \infty\} = \infty$ .

$\lambda_4(1) = min\{\lambda_1(0) + c_{14};\ \lambda_2(0) + c_{24};\ \lambda_3(0) + c_{34};\ \lambda_4(0) + c_{44};\ \lambda_5(0) + c_{54};\} = min\{0 + \infty ; \infty + 7; \infty + 1; \infty + \infty; \infty + 4\} = \infty$ .

$\lambda_5(1) = min\{\lambda_1(0) + c_{15};\ \lambda_2(0) + c_{25};\ \lambda_3(0) + c_{35};\ \lambda_4(0) + c_{45};\ \lambda_5(0) + c_{55};\} = min\{0 + 3; \infty + 1; \infty\ - 5; \infty + \infty; \infty + \infty\} = 3.$

Отримані значення λi(1) занесемо у другий стовпець табл. 2.1. Переконуємося, що другий стовпець, починаючи з другого елемента, збігається з першим рядком матриці терезів, що легко пояснюється змістом величин λі(1), які дорівнюють довжині мінімального шляху з першої вершини в і-у, що містить не більше однієї дуги.

$\underline{k = 2.}$

$\lambda_1(2) = 0.$

Рівність (2.1) для k = 2 має видляд:

$$\lambda_i(2) = \min_{1\le j\le 5} \{\lambda_j(1) + c_{ji}\}.$$

$$\lambda_2(2) = min\{0 + 1;\ 1 + \infty; \infty + \infty; \infty + \infty; 3 + \infty\} = 1.$$

$$\lambda_3(2) = min\{0 + \infty\ ; 1 + 8; \infty + \infty; \infty + 2; 3 + \infty\} = 9\ .$$

$$\lambda_4(2) = min\{0 + \infty\ ; 1 + 7; \infty + 1; \infty + \infty; 3 + 4\} = 7\ .$$

$$\lambda_5(2) = min\{0 + 3;\ \ 1 + 1; \infty - 5; \infty + \infty; 3 + \infty\} = 2.$$

Отримані значення λі(2) занесемо у третій стовпець табл. 6.1. Величини λі(2) рівні довжині мінімального шляху з першої вершини в і-у, що містить не більше двох дуг.

$\underline{k = 3.}$

$\lambda_1(3) = 0.$

Рівність (2.1) для k = 3 має вигляд:

$$\lambda_i(3) = \min_{1 \le j \le 5} \{\lambda_j(2) + c_{ji}\}.$$

$$\lambda_2(3) = min\{0 + 1;\ \ 1 + \infty;\ 9 + \infty;\ 7 + \infty;\ 2 + \infty\} = 1.$$

$$\lambda_3(3) = min\{0 + \infty\ ;\ 1 + 8;\ 9 + \infty;\ 7 + 2;\ 2 + \infty\} = 9\ .$$

$$\lambda_4(3) = min\{0 + \infty\ ;\ 1 + 7;\ 9 + 1;\ 7 + \infty;\ 2 + 4\} = 6\ .$$

$$\lambda_5(3) = min\{0 + 3;\ \ \ 1 + 1;\ 9 - 5;\ 7 + \infty;\ 2 + \infty\} = 2.$$

Отримані значення λi(3) занесемо у четвертий стовпець табл. 2.1. Величини λi(3) рівні довжині мінімального шляху з першої вершини в і-у, що містить не більше трьох дуг.

$\underline{k = 4.}$

$\lambda_1(4) = 0.$

Рівність (2.1) для k = 4 має вигляд:

$$\lambda_i(4) = \min_{1 \le j \le 5} \{\lambda_j(3) + c_{ji}\}.$$

$$\lambda_2(4) = min\{0 + 1;\ \ 1 + \infty;\ 9 + \infty;\ 6 + \infty;\ 2 + \infty\} = 1.$$

$$\lambda_3(4) = min\{0 + \infty\ ;\ 1 + 8;\ 9 + \infty;\ 6 + 2;\ 2 + \infty\} = 8\ .$$

$$\lambda_4(4) = min\{0 + \infty\ ;\ 1 + 7;\ 9 + 1;\ 6 + \infty;\ 2 + 4\} = 6\ .$$

$$\lambda_5(4) = min\{0 + 3;\ \ \ 1 + 1;\ 9 - 5;\ 6 + \infty;\ 2 + \infty\} = 2.$$

Отримані значення λi(4) занесемо до п'ятого стовпця табл. 2.1. Величини λi(4) рівні довжині мінімального шляху з першої вершини в і-у, що містить не більше чотирьох дуг.

Таблиця 2.1

| $I$ (номер вершины) | $\lambda_i(0)$ | $\lambda_i(1)$ | $\lambda_i(2)$ | $\lambda_i(3)$ | $\lambda_i(4)$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | $\infty$ | 1 | 1 | 1 | 1 |
| 3 | $\infty$ | $\infty$ | 9 | 9 | 8 |
| 4 | $\infty$ | $\infty$ | 7 | 6 | 6 |
| 5 | $\infty$ | 3 | 2 | 2 | 2 |

*Крок* 4. Відновлення мінімального шляху. Для останньої вершини x3 попередня їй вершина xr визначається співвідношення (2.2) отриманого при s =3:

$\lambda_r(3) + c_{r3} = \lambda_3(4), \qquad x_r \in G^{-1}(x_3),$

где $G^{-1}(x_3)$ - прообраз вершины $x_3$.

$$G^{-1}(x_3) = \{x_2, x_4\}. \tag{2.3}$$

Підставимо в (2.3) послідовно r = 2 і r = 4, щоб визначити, для якого r ця рівність виконується:

$$\lambda_2(3) + c_{23} = 1 + 8 \neq \lambda_3(4) = 8,$$

$$\lambda_4(3) + c_{43} = 6 + 2 = \lambda_3(4) = 8.$$

Таким чином, вершиною, що передує вершині x3 є вершина x4.

Для вершини x4 попередня їй вершина xr визначається співвідношення (2.2) отриманого при s =4:

$$\lambda_r(2) + c_{r4} = \lambda_4(3), \qquad x_r \in G^{-1}(x_4),$$

где $G^{-1}(x_4)$ - прообраз вершины $x_4$.

$$G^{-1}(x_4) = \{x_2, x_3, x_5\}. \tag{2.4}$$

Підставимо в (2.4) послідовно r = 2, r = 3 і r = 5, щоб визначити, для якого r ця рівність виконується:

$$\lambda_2(2) + c_{24} = 1 + 7 \neq \lambda_4(3) = 6,$$

$$\lambda_3(2) + c_{34} = 1 + 1 \neq \lambda_4(3) = 6,$$

$$\lambda_5(2) + c_{54} = 2 + 4 = \lambda_4(3) = 6,$$

Таким чином, вершиною, що передує вершині x4 є вершина x5.

Для вершини x5 попередня їй вершина xr визначається співвідношення (2.2) отриманого при s =5:

$$\lambda_r(1) + c_{r5} = \lambda_5(2), \qquad x_r \in G^{-1}(x_5),$$

где $G^{-1}(x_5)$ - прообраз вершины $x_5$.

$$G^{-1}(x_5) = \{x_1, x_2\}. \tag{2.5}$$

Підставимо в (2.5) послідовно r = 1 і r = 2, щоб визначити, для якого r ця рівність виконується:

$$\lambda_1(1) + c_{15} = 0 + 3 \neq \lambda_5(2) = 2,$$

$$\lambda_2(1) + c_{25} = 1 + 1 = \lambda_5(2) = 2,$$

Таким чином, вершиною, що передує вершині x5 є вершина x2.

Для вершини x2 попередня їй вершина xr визначається із співвідношення (2.2) отриманого при s =2:

$$\lambda_r(0) + c_{r2} = \lambda_2(1), \qquad x_r \in G^{-1}(x_2),$$

где $G^{-1}(x_2)$ - прообраз вершины $x_2$.

$$G^{-1}(x_2) = \{x_1\}. \tag{2.6}$$

Підставимо в (2.6) r = 1, щоб визначити, чи виконується ця рівність:

$$\lambda_1(0) + c_{12} = 0 + 1 = \lambda_2(1) = 1.$$

Таким чином, вершиною, що передує вершині x2 є вершина x1.

Отже, знайдено мінімальний шлях - x1, x2, x5, x4, x3, його довжина дорівнює 8.

## Література


1. Мелешко Є. В., Якименко М. С., Поліщук Л .І. Алгоритми та структури даних : навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. Кропивн. : Видавець – Лисенко В. Ф., 2019. 156 с.

2. Shinichi Shirakawa, Yasushi Iwata, Youhei Akimoto Dynamic Optimization of Neural Network Structures Using Probabilistic Modeling. *The Thirty-Second AAAI Conference on Artificial Intelligence* (AAAI-18), 2018. P. 4074–4082.

3. Кононюк А. Е. Дискретно-непрерывная математика. Графы — В 12-и кн. Кн 7, ч.4. К. : "Освіта України", 2015. 494 с.

4. Devi B. R., Rao K. K., Rani M. A. Application of Modified Bellman-Ford Algorithm for Cooperative Communication URL: https://link.springer.com/article/10.1007/s11277-019-06666-7 (Published: 19 August 2019).

5. Ramadiani, Bukhori D., Azainil, Dengen N. Floyd-warshall algorithm to determine the shortest path based on android. *1st International Conference on Tropical Studies and Its Application* (ICTROPS), Series: Earth and Environmental Science 144 (2018). URL: https://www.researchgate.net/publication/325017484_Floydwarshall_algorithm_to_determine_the_shortest_path_based_on_android

# Лабораторна робота № 3

## Мурашиний алгоритм

**1. Мета роботи:**

- ознайомитись з мурашиним алгоритмом;
- виконати програмну реалізацію мурашиного алгоритму.

**Завдання та порядок виконання роботи**

1. Використовуючи завдання з лабораторної роботи №1, знайти мінімальний шлях з вершини x1 до вершини x6 в орієнтованому графі за допомогою мурашиного алгоритму, виконати програмну реалізацію мурашиного алгоритму.

.

**Контрольні питання**

1. Які є види графів ?
2. Як можуть задаватися графи?
3. Що таке вершина у графі?
4. Що таке ребро у графі?
5. Які існують алгоритми для пошуку мінімальної відстані у графі?
6. Мурашиний алгоритм.
7. Опишіть класичну реалізацію мурашиного алгоритму.

**Зміст звіту**

1. Номер роботи, її назва та мета.
2. Короткі теоретичні відомості та відповіді на контрольні питання.
3. Згідно з варіантом завдання знайти мінімальний шлях з вершини x1 до вершини x6 в орієнтованому графі за допомогою мурашиного алгоритму.
4. Висновки.

**2. Теоретичні відомості**

Ідея мурашиного алгоритму - моделювання поведінки мурашок, який пов'язаний з їх здатністю швидко знаходити найкоротший шлях мурашника до джерела їжі і адаптуватися до умов, що змінюються, знаходячи новий найкоротший шлях. При своєму русі мурашка позначає шлях феромоном, і ця інформація використовується іншими мурахами для вибору шляху. Це елементарне правило поведінки і визначає здатність мурашок знаходити новий шлях, якщо старий виявляється недоступним.

Розглянемо випадок, показаний на рисунку, коли на оптимальному досі шляху виникає перешкода. В цьому випадку потрібне визначення нового оптимального шляху. Дійшовши до перешкоди, мурашки з рівною імовірністю обходитимуть її праворуч і ліворуч. Те ж саме відбуватиметься і на зворотній стороні перешкоди. Проте, ті мурашки, які випадково оберуть найкоротший шлях, будуть швидше його проходити, і за декілька пересувань він буде більше збагачений феромоном. Оскільки рух мурашок визначається концентрацією феромону, то наступні віддаватимуть перевагу саме цьому шляху, продовжуючи збагачувати його феромоном до тих пір, поки цей шлях по якій-небудь причині не стане доступним.

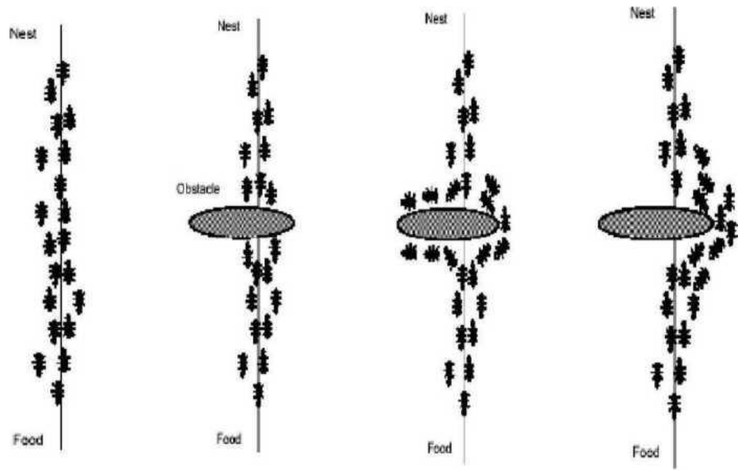



Рис. 3.1 – Зміна руху мурашок

Очевидний позитивний зворотний зв'язок швидко приведе до того, що найкоротший шлях стане єдиним маршрутом руху більшості мурашок. Моделювання випару феромона - негативного зворотного зв'язку - гарантує нам, що знайдене локально оптимальне рішення не буде єдиним - мурашки шукатимуть інші шляхи. Якщо ми моделюємо процес такої поведінки на деякому графі, ребра якого представляють собою можливі шляхи переміщення мурашок, плин визначеного часу, то найбільш збагачений феромоном шлях по ребрах цього графа і буде рішенням задачі, отриманого за допомогою мурашиного алгоритму.

Будь-який мурашиний алгоритм, незалежно від модифікацій, представимо в наступному вигляді (рис. 3.2):

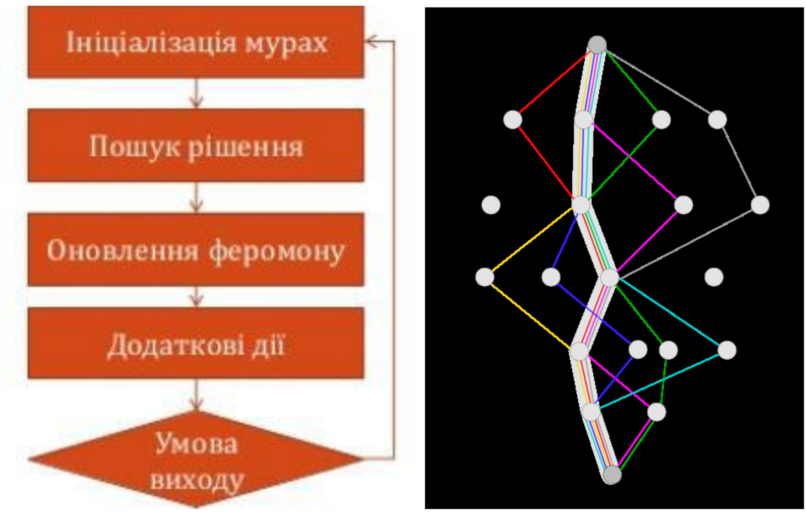



Рис. 3.2 – Метод оптимізації мурашиної колонії

Розглянемо кожен крок оптимізації мурашиної колонії.

## Мурашиний алгоритм

### 2.1. **Ініціалізація**

Стартова точка, куди поміщається мурашка, залежить від обмежень, що накладаються умовами завдання. Тому що для кожного завдання спосіб розміщення мурашок є визначальним. Або всі вони поміщаються в одну точку, або в різні з повтореннями, або без повторень. На цьому ж етапі задається початковий рівень феромону. Він іціалізується невеликим додатнім числом для того, щоб на початковому кроці ймовірність переходу до наступної вершини не була нульовою.

### 2.2. Пошук рішення

Вірогідність переходу з вершину *i* до вершини *j* визначається за формулою:

$$P_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{j \in \text{allowed nodes}} \tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}$$

де ***Тц (t)*** - рівень феромона, ***dj*** - евристична відстань, а ***а, в*** - константні параметри.

При ***а=0*** вибір ближнього міста найбільше вірогідний, тобто алгоритм стає жадібним.

При ***в=0*** вибір відбувається тільки на основі феромона, що приводить до субоптимальних рішень.

Тому необхідний компроміс між цими величинами, який знаходиться експериментально.

## 2.3. Оновлення феромону

Рівень феромону оновлюється згідно з формулою:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{\substack{k \in \text{Colony that} \\ \text{used edge } (i,j)}} \frac{Q}{L_k}$$

де ***p*** - інтенсивність випару, $L_k(t)$ - ціна поточного рі шення для k-ої мурашки, а ***Q*** - параметр, який має значення порядку ціни оптимального рішення,

тобто $\frac{Q}{L_k(t)}$ - феромон, що відкладається k-тою мурашкою, використовує ребро ***(i,j)***.

## Література


1. Dorigo M., Stutzle T. Ant Colony Optimization. MIT Press, Cambridge, MA, 2004.

4. Dorigo M. Swarm Intelligence, Ant Algorithms and Ant Colony Optimization. *Reader for CEU Summer University Course «Complex System»*, Budapest, Central European University. 2001. P. 1—38.

3. Rodrigues A. Application of Ant Colony Optimization to Data Distribution in Memory in Computer Systems. In Abstracts of 7th Annual Swarm Researchers Meeting «SwarmFest 2003». USA, Notre Dame. 2003 (повний текст на http://www. nd. edu/~arodrig6/).

4. Shmygelska A., Hoos H. An Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem. In Proc. of the Third International Workshop on Ant Algorithms «ANTS 2002». Lecture Notes in Computer Science 2463: Springer-Verlag, 2002.

# Лабораторна робота № 4

## Алгоритм простого перебору знаходження мінімального шляху у графі

**1. Мета роботи.** Навчитися реалізовувати алгоритм простого перебору для пошуку найкоротшого шляху від однієї вершини графа до іншої.

**Завдання та порядок виконання роботи**

1. Вивчити навчальний матеріал.
2. Підготовити відповіді на контрольні питання.
3. Використовуючи алгоритмом простого перебору шляхів, знайти мінімальний шлях з вершини $x_1$ до вершини $x_6$ в орієнтованому графі.

**Контрольні питання**

1. Що таке граф?
2. Які є види графів ?
3. Які існують алгоритми для пошуку мінімальної відстані у графі?
4. Алгоритм простого перебору шляхів у графі.
5. Опишіть реалізацію алгоритму простого перебору шляхів.

**Зміст звіту**

1. Номер роботи, її назва та мета.
2. Короткі теоретичні відомості та відповіді на контрольні питання.
3. Згідно з варіантом знайти мінімальний шлях з вершини $x_1$ до вершини $x_6$ в орієнтованому графі за допомогою алгоритму простого перебору шляхів.
4. Висновки.

## 2. Теоретичні відомості

### Алгоритм простого перебору

Алгоритм простого перебору відноситься до інструментарію вичерпного пошуку, який можна подати як генерування та перевірка. Вочевидь, це загальна алгоритмічна парадигма (загальна техніка вирішення будь-яких проблем), яка полягає у систематичному перерахуванні всіх можливих кандидатів на рішення та перевірці, чи задовольняє кожен кандидат твердження проблеми. В англомовних дослідженнях ця парадигма має назву - Brute Force Algorithms.

Такого роду алгоритми є прямолінійними методами вирішення проблеми, які покладаються на чисту обчислювальну потужність та випробування будь-якої можливості, а не вдосконалених методів підвищення ефективності.

Brute Force Algorithms перевіряє всі можливі сценарії, а не використовує прийом, щоб пришвидшити процес. Складність повного перебору залежить від кількості всіх можливих рішень задачі (див. рис. 2.3, який показує дію алгоритму). Якщо простір рішень дуже великий, то повний перебір може не дати результатів протягом декількох років або навіть століть.

**The Brute-Force Algorithms**

1) *A,B,C,D,E = 812*
2) *A,B,C,E,D = 777*
3) *A,B,D,C,E = 762*
4) *A,B,D,E,C = 773*
5) *A,B,E,C,D = 831*
6) *A,B,E,D,C = 877*
7) *A,C,B,D,E = 722*
8) *A,C,B,E,D = 791*
9) *A,C,D,B,E = 776*

10) $A,C,E,B,D = 741$

11) $A,D,B,C,E = 676$

12) $A,D,C,B,E = 780$

Повернемось до постановки завдання – розв’язок задачі пошуку найкоротшого шляху у представленні її моделлю графа.

Метод простого перебору у застосуванні для розв’язку зазначеної задачі буде полягати у переборі всіх можливих шляхів та обрахуванню довжини знайдених маршрутів. Вибір серед цих маршрутів того, що має мінімальну довжину і буде шуканим розв’язком поставленої задачі.

Доведено, що будь-яке завдання з класу NP може бути вирішена повним перебором. При цьому, навіть якщо обчислення цільової функції від кожного конкретного можливого рішення задачі може бути здійснено за поліноміальний час, в залежності від кількості всіх можливих рішень повний перебір може зажадати експоненціального часу роботи.

Підсумуємо сказане, та відмітимо переваги та недоліки представленого алгоритму.

*Переваги:*

- доступність та зрозумілість в виконанні;
- досить не складна комп’ютерна реалізація;
- не потрібно доводити його застосовність та підходить до виконання будь-яких завдань.

*Недоліки:*

- неоптимізований алгоритм, просто перебирає сценарії рішення;
- має погану асимптотичну оцінку трудомісткості.

## Література

1. Математичні методи дослідження операцій : підручник / Є. А. Лав-ров, Л. П. Перхун, В. В. Шендрик та ін. Суми : Сумський державний університет, 2017. 212 с.

2. Hassan M. H. Mustafa, Ayoub Al-Hamadi, Mohamed Abdulrahman, Shahinaz Mahmoud, Mohammed O Sarhan On Comparative Analogy between Ant Colony Systems and Neural Networks Considering Behavioral Learning Perfor-mance. *Journal of Computer Sciences and Applications*. 2015, Vol. 3 No. 3, 79-89.

3. Біологічні основи мурашиних колоній. [Електронний ресурс]. URL: http://posibniki.com.ua/post-prikladni-sistemi-kolektivnogo-intelektu-swarm-intelli-gence (Дата звернення 15.10.2020).

4. Korte B., Vygen J. Combinatorial Optimization: Theory and Algorithms (Algorithms and Combinatorics) 6th ed. New York, 2018. 455 p.

5. Divya M. A Comparison of Ant Colony Optimization Algorithms Applied to Distribution Network Reconfiguration. *International Journal of Engineering Re-search & Technology*, Volume 3, Issue 01, 2016. P. 1–4.

# Лабораторна робота № 5

## Складність алгоритмів

**1. Мета роботи.** Навчитися обчислювати складність алгоритмів пошуку найкоротшого шляху у графі.

**Завдання та порядок виконання роботи**

1. Вивчити навчальний матеріал.
2. Підготовити відповіді на контрольні питання.
3. Обчислити складність алгоритмів знаходження мінімальних шляхів в орієнтованому графі для алгоритмів:

- Дейкстри;
- Форда-Беллмана;

- мурашиного алгоритму;
- простого перебору.

**Контрольні питання**

1. Які є види графів ?
2. Які існують алгоритми для пошуку мінімальної відстані у графі?
3. Алгоритм простого перебору шляхів у графі.
4. Що таке обчислювальна складність алгоритмів?

**Зміст звіту**

1. Номер роботи, її назва та мета.
2. Короткі теоретичні відомості та відповіді на контрольні питання.
3. Представити обчислювальну складність алгоритмів: Дейкстри, Форда-Беллмана, мурашиного алгоритму, простого перебору.
4. Висновки.

## 2. Теоретичні відомості

Розвиток технологій різних напрямків привів до того, що для дослідження даних у більшості з них, потрібно побудувати математичну модель. Однією з таких моделей є граф, яку можна застосувати до широкого спектру даних. Такі моделі, на основі графів, використовуються в комп’ютерних мережах, проектуванні ЕОМ, файлових системах, алгоритмах та структурах даних, тощо. Найчастіше на графі виконують алгоритм пошуку оптимального шляху. Шлях між двома точками графу може означати найшвидший, найдешевший, найкоротший шлях між двома об’єктами моделі.

Першим дослідженням з теорії графів є стаття Ейлера, випущена у 1736 році. Першим алгоритмом пошуку найкоротшого шляху є алгоритм Дейкстри, створений у 1956 році.

Алгоритм знаходження найкоротшого шляху - пошук шляху, який також називають ланцюгом, між двома вершинами графу. Основна задача алгоритму полягає у знаходжені шляху з мінімальною загальною вагою ребер графу, які проходять алгоритм при досяганні кінцевої вершини. Алгоритм розпочинається з одної вершини, переходить до сусідніх вершин, порівнюючи вагу ребер, поки не дійде до кінцевої вершини.

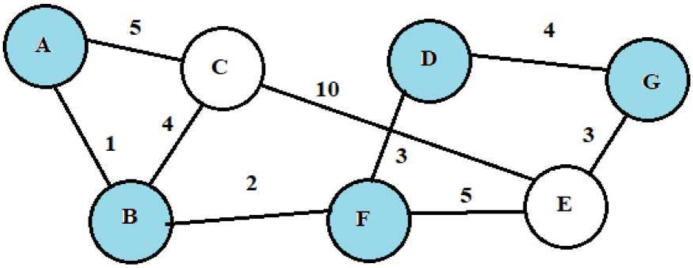



Рис 5.1. Приклад роботи алгоритму пошуку шляху

На рис. 5.1 показана робота алгоритму пошуку найкоротшого шляху між вершинами A і G, які розміщені в орієнтованому графі з вагами. Вага оптимального шляху становить 10 одиниць.

Розглянемо основні алгоритми пошуку шляхів:

***Алгоритм Дейкстри***. Найвідоміший і найпоширеніший алгоритм, який знаходить коротший маршрут від однієї вершини, до всіх інших. Алгоритм не буде працювати, якщо граф матиме ребра з вагою менше нуля. Цей алгоритм є одним із найпростіших. У випадку з графом, де кількість вершин у графі може сягати кількох тисяч використання даного алгоритму не буде оптимальним вибором. Якщо застосувати алгоритм до графа з від’ємними ребрами, алгоритм покаже неправильний (збитковий) маршрут.

Обчислювальна складність алгоритму Дейкстри - O($n^2$ + m), де n - вершини, m – ребра графу.

***Алгоритм Беллмана-Форда***. Алгоритм пошуку найкоротшого шляху у зваженому графі, який знаходить шлях від однієї вершини до решти. На відміну від алгоритму Дейкстри, алгоритм Беллмана Форда допускає наявність у графі ребер з від'ємною вагою. У випадку з графом, з великою кількістю вершин, алгоритм використає повний перебір всіх вершин графа, що призведе до великої втрати часу та займе більший обсяг пам'яті обчислювальної машини.

Обчислювальна складність алгоритму Беллмана-Форда - O(n * m) , де n - вершини, m - ребра.

**Алгоритм пошуку А** *. Даний алгоритм за суттю є розширенням алгоритму Дейкстри, але досягає вищої продуктивності за рахунок введення у роботу алгоритму евристичної функції. Алгоритм А* є алгоритмом пошуку за кращим збігом на графі, який знаходить маршрут з найменшою вагою ребер від початкової вершини до кінцевої. Даний алгоритм крок за кроком переглядає всі шляхи, що ведуть від початкової вершини в кінцеву, поки не знайде мінімальний. Порядок обходу вершин визначається евристичною функцією «відстань + вартість» та евристичною оцінкою відстані від розглянутої вершини до кінцевої.

Обчислювальна складність Алгоритм пошуку А *. - O(log h(x)), де h - евристична функція

Отже, з представлених алгоритмів найкращу продуктивність показує А*, але при великій кількості вершин, А*, як і інші представлені алгоритми, покаже незадовільний результат за витратами часу і об'єму використаної пам'яті ЕОМ.

Алгоритми пошуку найкоротшого шляху - дуже важливий розділ в теорії графів. Такі алгоритми можна використовувати для обробки або дослідження даних в різних галузях, які виходять за межі інформаційних технологій.

## Література

1. Мультиагентна система маршрутизації на основі алгоритмів пошуку найкоротшого шляху в графі. URL: https://ela.kpi.Ua/bitstream/123456789/24087/1/Tishkov magistr.pdf

2. Седжвик Р. Фундаментальные алгоритмы на С++ / К. : ДиаСофт, 2001. 688 с.

3. Оцінка складності алгоритмів, класифікація алгоритмів за складністю. URL: https://naurok.com.ua/lekciya-ocinka-skladnosti-algoritmiv-klasifikaciya-algoritmiv-za-skladnistyu-171782.html

# Лабораторна робота № 6

## Максимальний потік. Алгоритм Форда-Фалкерсона

**1. Мета роботи.** Навчитися обчислювати максимальний потік у мережі.

### Завдання та порядок виконання роботи

1. Вивчити навчальний матеріал.
2. Підготовити відповіді на контрольні питання.
3. Згідно з варіантом обчислити максимальний потік у мережі.

### Контрольні питання

1. Що таке мережа ?
2. Які існують алгоритми для обчислення максимального потоку у мережі?

3. Алгоритм обчислення максимального потоку.
4. Опишіть реалізацію алгоритму максимального потоку.

**Зміст звіту**

1. Номер роботи, її назва та мета.
2. Короткі теоретичні відомості та відповіді на контрольні питання.
3. Представити обчислений максимальний потік, згідно з варіантом.
4. Висновки.

## 2. Теоретичні відомості

Розглянемо мережу G(V, U). Нехай кожній дузі (i, j) = Uy є U, що йде з і в j, поставлене у відповідність невід'ємне число dy, яке назвемо пропускною здатністю цієї дуги (тобто максимальна кількість речовини dj, яку можна пропустити по дузі Uy за одиницю часу). Якщо вершини *i* і *j* з'єднані ребром (неорієнтованою дугою), його замінюють парою протилежно орієнтованих дуг Uy- і и]Ч з однаковою пропускною здатністю dy = djt. Кожній дузі Uj поставимо у відповідність ще одне невід'ємне число Xjj, яке назвемо потоком по дузі Uy.

Потоком по дузі називається кількість речовини Xy, що проходить через дугу Uy в одиницю часу. З фізичного сенсу вантажопотоку на мережі випливає нерівність: 0 < Xy < dy-, Uy-є U, тобто. потік по кожному ребру не може перевищувати його пропускну здатність.

Потоком на мережі з I S називається безліч X неотрицательных чисел Xy , тобто. X = { Xy-> 0 для дуг Uy є U}, що задовольняють наступним умовам: XXц-Xxy = -z; XXiS-XXSj = z; XXk-XXjj = 0,keУ,k*S,k*I. j I j I j

Ця рівність означає, що для будь-якої вершини k мережі, крім джерела і стоку, кількість речовини, що надходить у цю вершину, дорівнює кількості речовини, що виходить з неї. Цей зв'язок називається умовою збереження потоку, саме: у проміжних вершинах потоки не створюються і зникають. Отже, загальна кількість речовини, що виходить із джерела I, збігається із загальною кількістю речовини, що надходить у стік S, що і відображається в попередніх умовах. Функція z називається величиною (потужністю) потоку на мережі та показує загальну кількість речовини, яка може пройти через мережу. Необхідно знайти (побудувати) максимальний потік із джерела I у стік S таким чином, щоб величина потоку хгу по кожній дузі щ в мережі не перевищувала пропускну здатність цієї дуги і виконувалася умова збереження потоку, тобто. знайти Z^max за попередніх умов.

Це типове завдання лінійного програмування. Але так як завдання про максимальний потік має специфічну структуру, то для неї є ефективніший метод вирішення, ніж симплексний.

Поняття розрізу у мережі

Кожній дузі ставляться у відповідність два числа (dj, Xj): перше - пропускну здатність; друге – потік. Очевидно, якщо мережа є шляхом I в S, то максимальний потік буде дорівнює мінімальній з пропускних здібностей дуг, тобто. дуга з мінімальною пропускною здатністю є вузьким місцем колії. Аналогом вузького місця у мережі є розріз.

Нехай безліч вершин V в мережі G(V, U) розбито на два непересічні підмножини R, R', причому об'єднання цих підмножин дає безліч V.

Розрізом (R, R) у мережі G(V, U) називається безліч дуг щ, котрим вершини vt ER, а вершини Vj ER'. Взагалі, розріз (R,R') не збігається з розрізом (R',R). Якщо IER, а SER', то (RR) називатимемо

розрізом, що відокремлює джерело від стоку. Пропускною здатністю розрізу (R, R') називається величина C(R,R') = £dy.

Розглянемо мережу на рис. 15

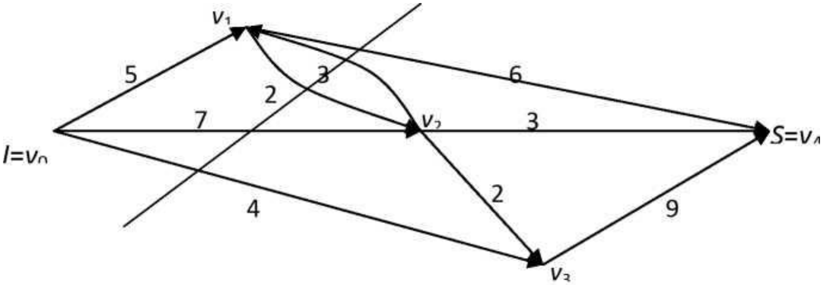



Рис. 15. Мережа G(V, U)

У цій мережі можна побудувати 7 розрізів. Розглянемо, наприклад, розріз (R, R), де R = {I, vj, Rf = {v2, v3, 5}. Тоді

(R, R') = {u14, u12, U02, U03}, C(R, Rr) = di4 + di2 + do2 + do3 = 6 + 2 + 7 + 4 = 19.

Розріз, що відокремлює джерело від стоку і має мінімальну пропускну здатність, називається мінімальним розрізом.

**Алгоритм Форда-Фалкерсона** побудови максимального потоку та знаходження мінімального розрізу полягає в наступному:

1. Знаходять збільшує шлях шляхом розміщення міток.

1.1. Нехай у мережі є допустимий потік X = {Xj} (наприклад, {Xj} = 0). Джерело I = v0 одержує мітку I+.

1.2. Проглядають всі непомічені вершини vt, сусідні з v0, і присвоюють їм мітку v+, якщо u0i - пряма дуга і x0i < d0i, і мітку v-, якщо u0i - дуга зворотна і Xi0> 0, і т.д.

1. k. Для кожної вершини vk, позначеної на попередньому

(k - 1) кроці, переглядають сусідні з нею непомічені вершини vi. Кожна така вершина vi отримує мітку vk, якщо дуга uki пряма

і xki < dk/, і мітку v-, якщо дуга зворотна uik і xik > 0.

2. Якщо на k-му кроці не виходить помітити стік S, то збільшує шляхи немає і потік в мережі є максимальним. Побудований розріз (R, R'), де R - безліч помічених вершин в мережі, а R' - безліч непомічених вершин є мінімальним розрізом, і алгоритм закінчує свою роботу. Інакше переходять до пункту 3.

3. Якщо на k-му кроці стік S виявився поміченим, то виписують шлях, що збільшує P, рухаючись від стоку S до вершини, номер якої вказаний в її мітці; потім від неї до вершини, номер якої вказаний у її мітці; і в результаті приходять до джерела.

4. Виписують безліч P+ (прямих дуг) і безліч P~ (зворотних дуг) збільшує шлях P.

5. Обчислюють для прямих дуг величину e1 = min(dij – Xj),

а для зворотних – величину $\underset{u_{ij} \in P^-}{\varepsilon_2 = \min x_{ij}.}$ .

6. Уздовж дуг збільшує шляху P змінюють потік X = { xij} на величину a = min {$\epsilon_1$, $\epsilon_2$} і отримують новий потік X = { xj }, такий що

$$x'_{ij} = \begin{cases} x_{ij}, u_{ij} \notin P; \\ x_{ij} + \varepsilon, u_{ij} \in P^+; \\ x_{ij} - \varepsilon, u_{ij} \in P^-. \end{cases}$$

**Завдання:**

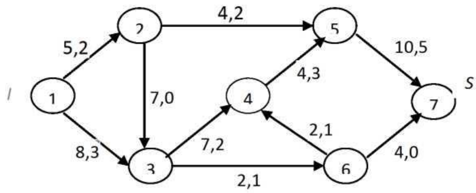
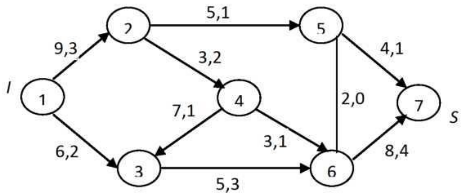
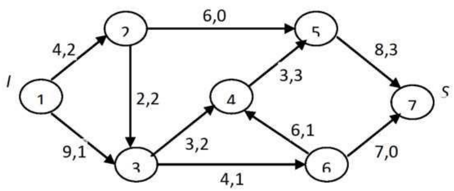
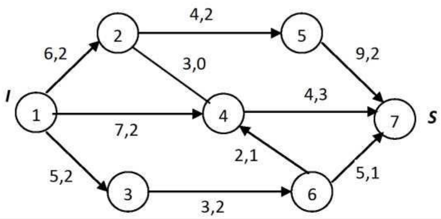
Водопровід (мережа) з'єднує джерело I зі стоком S. Є кілька шляхів, якими можна доставляти воду з джерела у стік. Вершини мережі відповідають перетинам труб, а ребра та дуги - ділянкам труб між перетинами. На мережі вказані пропускні можливості труб, тобто. максимальна кількість води в м$^3$, яку можна пропустити трубами за 1 годину. Також сформовано початковий потік із потужністю $z_0$ (м$^3$/год). Який потік води максимальної потужності можна пропустити цим трубопроводом?

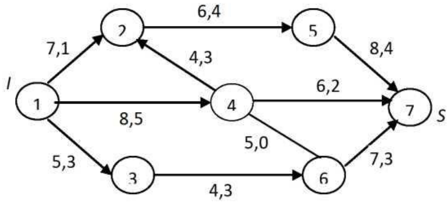
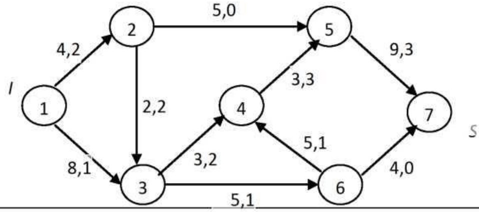
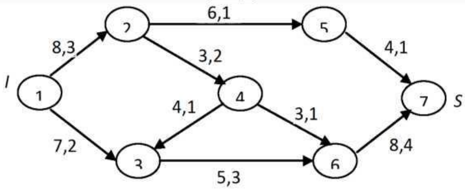
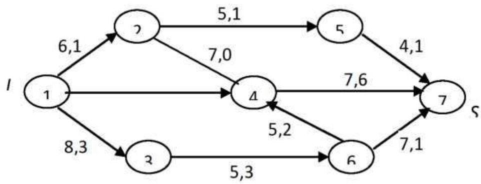
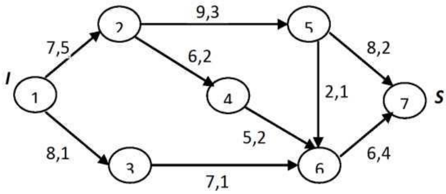
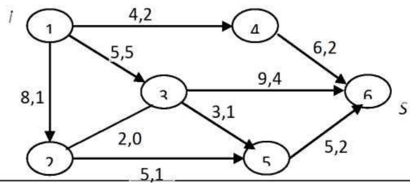
Потрібно:

1) порахувати потужність початкового потоку води $z_0$ (м$^3$/год);

2) побудувати на мережі потік води максимальної потужності $z_{max}$ (м$^3$/год), спрямований із джерела I до стоку S;

3) вказати «вузьке місце» мережі та знайти його пропускну спроможність.

<table>
<tr><th>Вариант</th><th>Граф</th></tr>
<tr><td>1</td><td>
</td></tr>
<tr><td>2</td><td>
</td></tr>
<tr><td>3</td><td>
</td></tr>
<tr><td>4</td><td>
</td></tr>
</table>

| | |
|---|---|
| 5 | 6,4 7,1 4,3 8,4 I 6,2 1 2 3 4 5 6 7 S 8,5 5,0 5,3 7,3 4,3 |
| 6 | 5,0 4,2 9,3 I 3,3 2,2 1 2 3 4 5 6 7 S 5,1 8,1 3,2 4,0 5,1 |
| 7 | 6,1 8,3 3,2 4,1 I 1 2 3 4 5 6 7 S 4,1 3,1 7,2 8,4 5,3 |
| 8 | 5,1 6,1 4,1 7,0 7,6 I 1 2 3 4 5 6 7 S 5,2 8,3 7,1 5,3 |
| 9 | 9,3 7,5 8,2 6,2 I 1 2 3 4 5 6 7 S 2,1 5,2 8,1 6,4 7,1 |
| 10 | I 4,2 1 2 3 4 5 6 5,5 6,2 9,4 8,1 S 3,1 2,0 5,2 5,1 |

## Література

1. Томас Кормен, Чарльз Лейзерсон, Рональд Рівест. Вступ до алгоритмів. MIT Press і McGraw-Hill. Глава 26. Максимальний потік.
2. Попов Ю. Д., Тюптя В. І., Шевченко В. І. Методи оптимізації : навчальний посібник для студентів спеціальностей «Прикладна математика», «Інформатика», «Соціальна інформатика». К. : Абрис, 1999. 217 с.
3. Зайченко Ю. П. Дослідження операцій : підручник. К. : Слово., 2006. 816 с.

# Лабораторна робота № 7

## Максимальний потік. Розріз у мережі

**1. Мета роботи**. Навчитися обчислювати максимальний потік за допомогою розрізу у мережі.

**Завдання та порядок виконання роботи**

1. Вивчити навчальний матеріал.
2. Підготовити відповіді на контрольні питання.
3. Згідно з варіантом у лабораторній роботі 6 обчислити максимальний потік за допомогою розрізу у мережі.

Контрольні питання

1. Що таке мережа ?
2. Що таке розріз у мережі?
2. Які існують алгоритми для обчислення максимального потоку у мережі?

3. Опишіть реалізацію алгоритму максимального потоку за допомогою розрізу у мережі.

**Зміст звіту**

1. Номер роботи, її назва та мета.
2. Короткі теоретичні відомості та відповіді на контрольні питання.
3. Згідно з варіантом представити обчислений максимальний потік за допомогою розрізу у мережі.
4. Висновки.

## 2. Теоретичні відомості

Для графа G (рис. 7.1), що має орієнтовані ребра (2,3) та (3,1),

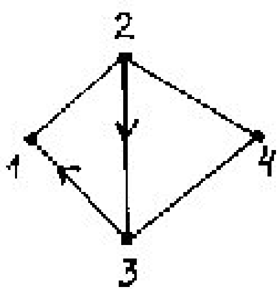



Рис. 7.1. Граф G

структурна матриця S має вигляд:

$$S = \begin{pmatrix} 1 & e_{12} & 0 & 0 \\ \overline{e_{12}} & 1 & e_{23} & e_{24} \\ e_{31} & 0 & 1 & e_{34} \\ 0 & \overline{e_{24}} & \overline{e_{34}} & 1 \end{pmatrix}.$$

Справедливим є твердження про знаходження всіх простих шляхів з вершини i у вершину j.

Твердження. Нехай S = (sj), де i, j = 1,2, ... n - структурна матриця графа G. Якщо мінор Mji цієї матриці розкрити як визначник порядку n-1, замінюючи додавання та віднімання на диз'юнкцію, а множення на кон'юнкцію і спростити, використовуючи властивості логічних функцій $x\overline{x} = 0; x \cdot 1 = x; x \vee xy = x$ , то отримаємо всі прості шляхи з i до j у формі ДНФ.

**Приклад**. Для графа G з попереднього прикладу знайдемо всі прості шляхи з вершини i = 1 у вершину j = 3, а також перерізу між вершинами i = 1 і j = 3.

Для отримання мінору M31 викреслюємо з матриці S третій рядок та перший стовпець.

$$M_{31} = \begin{vmatrix} e_{12} & 0 & 0 \\ 1 & e_{23} & e_{24} \\ \overline{e_{24}} & \overline{e_{34}} & 1 \end{vmatrix} = e_{12} \begin{vmatrix} e_{23} & e_{24} \\ \overline{e_{34}} & 1 \end{vmatrix} = e_{12}(e_{23} \vee e_{24}\overline{e_{34}}) = e_{12}e_{23} \vee e_{12}e_{24}\overline{e_{34}}$$

(Тут використана теорема про розкладання визначника за першим рядком). Знаки заперечення після отримання відповіді треба прибрати і записати прості шляхи з 1-ї вершини в 3-ю у порядку проходження ребер. Маємо: e12e23 v e12e24e43.

Для знаходження перерізів між вершинами i = 1 і j = 3 скористаємося наступним твердженням.

Твердження. Якщо всі прості шляхи з вершини і у вершину j записані як ДНФ, то сукупність всіх перерізів між цими вершинами перебуває приведенням до ДНФ заперечення всіх простих шляхів.

Для приведення до ДНФ застосовують правила де Моргана та поглинання. При записі відповіді заперечення над ребрами треба забрати.

Цю процедуру можна замінити більш простим алгоритмом. Якщо всі прості шляхи з вершини і у вершину j записані як ДНФ, то замінюючи диз'юнкцію на кон'юнкцію, а кон'юнкцію на диз'юнкцію, розкриваючи дужки і провівши спрощення, також отримаємо переріз між вершинами і та j.

У нашому випадку маємо

$$(e_{12} \vee e_{23})(e_{12} \vee e_{24} \vee e_{43}) = e_{12} \vee e_{12}e_{24} \vee e_{12}e_{43} \vee e_{23}e_{12} \vee e_{23}e_{24} \vee e_{23}e_{43} = e_{12} \vee e_{23}e_{24} \vee e_{23}e_{43}$$

При отриманні перерізів використані властивості $x \cdot x = x; x \vee xy = x$ .

## Література


1. Томас Кормен, Чарльз Лейзерсон, Рональд Рівест. Вступ до алгоритмів. MIT Press і McGraw-Hill. Глава 26. Максимальний потік.
2. Попов Ю. Д., Тюптя В. І., Шевченко В. І. Методи оптимізації : навчальний посібник для студентів спеціальностей «Прикладна математика», «Інформатика», «Соціальна інформатика». К. : Абрис, 1999. 217 с.
3. Зайченко Ю. П. Дослідження операцій : підручник. К. : Слово., 2006. 816 с.