МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Навчалньно-науковий інститут електричної інженерії та інформаційних технологій

Кафедра автоматизації та інформаційних систем

**ПРАКТИЧНА РОБОТА**

Виконав: студент групи КІ-23-1

Черниш В’ячеслав Олександрович

Перевірив:

Сидоренко Валерій Миколайович

м. Кременчук

2024 рік

**Практична робота № 6**

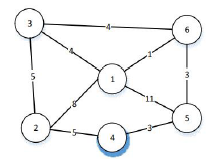
**Тема. Графи. Найкоротші шляхи**

**Мета:** набути практичних навичок розв’язання задач пошуку найкоротших шляхів у графі та оцінювання їх асимптотичної складності.

**Задачі для самостійного розв’язання**

Виконати індивідуальне завдання. Завдання полягає у знаходженні найкоротших шляхів від вершини 1 до всіх інших за допомогою алгоритму, вказаному у варіанті. Номер варіанта відповідає номеру студента у списку групи. У разі, якщо було досягнуто кінця списку задач, потрібно циклічно повернутися на його початок.

8. Задача з вар. 4, але за алгоритмом Белмена–Форда.



Кількість ітерацій k=V−1k = V - 1k=V−1, де VVV — кількість вершин у графі.

D(1)=0,

D(2)=∞,

D(3)=∞,

D(4)=∞,

D(5)=∞,

D(6)=∞

Кількість ітерацій k=V−1, де V — кількість вершин у графі.

k=6-1=5 (Кількість ітерацій)

**Перша ітерація**

Перевіряємо кожне ребро:

**Ребро 1→2 (вага 8)**:

D(2)=min(∞,D(1)+8)=min(∞,0+8)=8

**Ребро 1→3 (вага 4)**:

D(3)=min(∞,D(1)+4)=min(∞,0+4)=4

**Ребро 1→5 (вага 11)**:

D(5)=min(∞,D(1)+11)=min(∞,0+11)=11

**Ребро 1→6 (вага 1)**:

D(6)=min(∞,D(1)+1)=min(∞,0+1)=1

**Ребро 2→4 (вага 5)**:

D(4)=min(∞,D(2)+5)=min(∞,8+5)=13

**Ребро 2→3 (вага 5)**:

D(3)=min(4,D(2)+5)=min(4,8+5)=4

**Ребро 5→4 (вага 3)**:

D(4)=min(13,D(5)+3)=min(13,11+3)=14

**Ребро 6→5 (вага 3)**:

D(5)=min(11,D(6)+3)=min(11,1+3)=4

**Ребро 3→6 (вага 4)**:

D(6)=min(1,D(3)+4)=min(1,4+4)=1

D(1)=0,

D(2)=8,

D(3)=4,

D(4)=13,

D(5)=4,

D(6)=1

**Друга ітерація**

Перевіряємо кожне ребро:

**Ребро 1→2 (вага 8)**:

D(2)=min(8,D(1)+8)=min(8,0+8)=8

**Ребро 1→3 (вага 4)**:

D(3)=min(4,D(1)+4)=min(4,0+4)=4

**Ребро 1→5 (вага 11)**:

D(5)=min(4,D(1)+11)=min(4,0+11)=4

**Ребро 1→6 (вага 1)**:

D(6)=min(1,D(1)+1)=min(1,0+1)=1

**Ребро 2→4 (вага 5)**:

D(4)=min(13,D(2)+5)=min(13,8+5)=13

**Ребро 2→3 (вага 5)**:

D(3)=min(4,D(2)+5)=min(4,8+5)=4

**Ребро 5→4 (вага 3)**:

D(4)=min(13,D(5)+3)=min(13,4+3)=7

**Ребро 6→5 (вага 3)**:

D(5)=min(4,D(6)+3)=min(4,1+3)=4

**Ребро 3→6 (вага 4)**:

D(6)=min(1,D(3)+4)=min(1,4+4)=1

D(1)=0,

D(2)=8,

D(3)=4,

D(4)=7,

D(5)=4,

D(6)=1

**Третя ,четверта,пята ітерація**

У цих ітераціях відстані більше не змінюються.

Відповідь:

D(1)=0,

D(2)=8,

D(3)=4,

D(4)=7,

D(5)=4,

D(6)=1

**Контрольні питання**

1. Що таке граф і які головні складові його структури?

Граф — це математична структура, що складається з вершин (або вузлів) і ребер (або дуг), які з'єднують пари вершин. У графі можуть бути спрямовані (кожне ребро має напрямок) або неспрямовані (ребра не мають напрямку).

Головні складові структури графа:

Вершини (Nodes, Vertices) — об'єкти або точки в графі, що можуть представляти такі елементи, як міста, люди, комп'ютери тощо.

Ребра (Edges, Arcs) — зв'язки між вершинами. У спрямованих графах кожне ребро має напрямок від однієї вершини до іншої, в несуремінних графах напрямку немає.

Вага ребра — у графах з вагами (weighted graph) кожне ребро має числову вагу, яка може представляти, наприклад, відстань або час.

2. Які алгоритми використовуються для пошуку найкоротших шляхів у графах?

Для пошуку найкоротших шляхів у графах використовуються такі алгоритми:

Алгоритм Дейкстри — для графів з не від'ємними вагами.

Алгоритм Беллмана-Форда — для графів з від'ємними вагами (може обробляти графи з від'ємними вагою ребер, але не з від'ємними циклами).

Алгоритм Флойда–Форшала — для знаходження найкоротших шляхів між усіма парами вершин (повний алгоритм для всіх пар).

*Алгоритм А (A-star)*\* — для пошуку найкоротшого шляху в графах, з додатковим евристичним набором, що використовує оцінку відстані від поточної вершини до мети.

3. Як працює алгоритм Дейкстри і які його особливості?

Алгоритм Дейкстри використовується для знаходження найкоротших шляхів від однієї вершини до всіх інших у графі з невід'ємними вагами ребер.

Основні етапи роботи алгоритму:

1. Ініціалізуємо відстані до всіх вершин як нескінченність, за винятком початкової вершини, яка має відстань 0.
2. Використовуємо пріоритетну чергу (чергу з мінімальним пріоритетом), щоб вибирати з поточних вершин ту, яка має найменшу відстань.
3. Для кожної сусідньої вершини перевіряємо, чи можна покращити її відстань через поточну вершину.
4. Повторюємо кроки, поки не обробимо всі вершини.

Особливості:

Працює тільки з графами, де всі ребра мають невід'ємні ваги.

Може використовуватися для пошуку найкоротших шляхів від однієї вершини до інших.

Може працювати з пріоритетною чергою для зменшення складності.

4. Що таке алгоритм Беллмана-Форда і коли його варто застосовувати?

Алгоритм Беллмана-Форда — це алгоритм для пошуку найкоротших шляхів у графах, який працює з графами з від'ємними вагою ребер. Алгоритм за V−1 ітерацій оновлює відстані до всіх вершин, де V — кількість вершин у графі.

Основні етапи роботи:

1. Ініціалізація відстаней до всіх вершин як нескінченність, крім початкової вершини, відстань до якої дорівнює 0.
2. Для кожного ребра оновлюємо відстані до кінцевих вершин.
3. Повторюємо кроки V−1 разів для кожного ребра.

Коли використовувати:

Якщо в графі є від'ємні ваги ребер.

Якщо потрібно виявити наявність від'ємних циклів (якщо після V−1 ітерацій відстань змінюється, то це означає наявність від'ємного циклу).

Алгоритм менш ефективний для великих графів порівняно з Дейкстрою, але корисний для графів з від'ємними вагами.

5. Як працює алгоритм Флойда–Форшала і які його переваги та недоліки?

Алгоритм Флойда-Форшала використовується для знаходження найкоротших шляхів між усіма парами вершин у графі. Це так званий алгоритм для всіх пар найкоротших шляхів.

Основні етапи роботи:

1. Створюємо матрицю відстаней D, де D[i][j] — це початкова відстань між вершинами iii та jjj. Якщо немає ребра між вершинами, значення буде нескінченність.
2. Поступово для кожної вершини k перевіряємо, чи можна покращити відстань між парами вершин i та j через вершину k: D[i][j]=min(D[i][j],D[i][k]+D[k][j])
3. Повторюємо цей процес для всіх можливих проміжних вершин.

Переваги:

Може обробляти графи з від'ємними вагою ребер (але без від'ємних циклів).

Підходить для знаходження найкоротших шляхів між усіма парами вершин.

Недоліки:

Має складність , де V — кількість вершин, що робить його менш ефективним для великих графів порівняно з іншими алгоритмами (наприклад, для однієї пари вершин алгоритм Дейкстри ефективніший).