МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Навчально-науковий інститут електричної інженерії та інформаційних технологій

Кафедра автоматизації та інформаційних систем

**ПРАКТИЧНА РОБОТА**

Виконав: студент групи КІ-23-1

Бобров Євген

Перевірив:

Сидоренко Валерій Миколайович

м. Кременчук

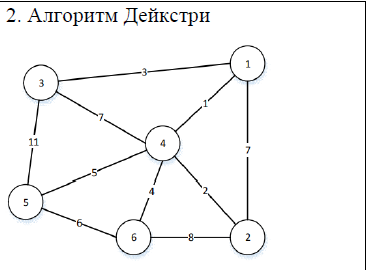
2024 рік

**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. 

Починаємо з вершини 1

d[1] = 0, d[4] = ∞, d[3] = ∞, d[2] = ∞, d[5] = ∞, d[6] = ∞

d[4]=min(∞,0+1)=1

d[2]=min(∞,0+7)=7

d[3]=min(∞,0+3)=3

d[1] = 0, d[4] = 1, d[3] = 3, d[2] = 7, d[5] = ∞, d[6] = ∞

Обираємо вершину з мінімальним d[v] (серед невідвіданих)

Вершина 4 (d[4]=1).

d[2]=min(7,1+2)=3

d[5]=min(∞,1+5)=6

d[6]=min(∞,1+4)=5

d[1] = 0, d[4] = 1, d[3] = 3, d[2] = 3, d[5] = 6, d[6] = 5

Обираємо вершину з мінімальним d[v] (серед невідвіданих)

Вершина 3 (d[3]=3).

d[5]=min(6,3+11)=6 (не змінюється)

d[1] = 0, d[4] = 1, d[3] = 3, d[2] = 3, d[5] = 6, d[6] = 5

Обираємо вершину з мінімальним d[v] (серед невідвіданих)

Вершина 2 (d[2]=3d[2] = 3d[2]=3).

d[1] = 0, d[4] = 1, d[3] = 3, d[2] = 3, d[5] = 6, d[6] = 5

Обираємо вершину з мінімальним d[v] (серед невідвіданих)

Вершина 6 (d[6]=5d[6] = 5d[6]=5).

d[1] = 0, d[4] = 1, d[3] = 3, d[2] = 3, d[5] = 6, d[6] = 5

Вершина 1: 0

Вершина 4: 1

Вершина 3: 3

Вершина 2: 3

Вершина 6: 5

Вершина 5: 6

**Контрольні питання**

**1. Що таке граф і які його основні елементи?**

Граф — це математична структура, що складається з набору вершин (вузлів) та зв’язків між ними, які називаються ребрами. Графи можуть бути спрямованими (з направленими ребрами) або неспрямованими (без напрямку).

Основні елементи графа:

* **Вершини (Nodes, Vertices):** Об’єкти або точки в графі, які можуть представляти різні елементи, такі як міста, пристрої, люди тощо.
* **Ребра (Edges, Arcs):** Зв’язки між вершинами. У спрямованих графах кожне ребро має напрямок, у неспрямованих — напрямок відсутній.
* **Вага ребра (Edge Weight):** У зважених графах кожне ребро має числову вагу, яка може означати відстань, вартість, час або інші значення.

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

Для визначення найкоротших шляхів у графах застосовуються такі алгоритми:

* **Алгоритм Дейкстри:** Призначений для графів із невід’ємними вагами.
* **Алгоритм Беллмана-Форда:** Використовується для графів із вагами, які можуть бути від’ємними. Підходить для обробки графів із від’ємними вагами, але не працює у випадках з від’ємними циклами.
* **Алгоритм Флойда-Воршелла:** Обчислює найкоротші шляхи між усіма парами вершин у графі.
* *Алгоритм A (A-star):*\* Оптимізований для пошуку шляхів у графах із використанням евристики, яка оцінює відстань до цільової вершини.

**3. Як працює алгоритм Дейкстри та які його ключові особливості?**

Алгоритм Дейкстри шукає найкоротші шляхи від заданої вершини до всіх інших у графі з невід’ємними вагами ребер.

**Основні етапи алгоритму:**

1. Встановлюємо для всіх вершин значення відстані як нескінченність, окрім початкової вершини, якій присвоюється значення 0.
2. Використовуємо пріоритетну чергу для вибору вершини з мінімальною відстанню.
3. Перевіряємо кожну сусідню вершину та оновлюємо її відстань, якщо шлях через поточну вершину є коротшим.
4. Повторюємо процедуру, доки всі вершини не будуть оброблені.

**Особливості алгоритму:**

* Працює тільки з графами, у яких всі ваги ребер невід’ємні.
* Ефективний для пошуку найкоротших шляхів від однієї вершини до інших.
* Пріоритетна черга оптимізує процес вибору вершин, що зменшує часову складність алгоритму.

**4. Що таке алгоритм Беллмана-Форда і в яких випадках він застосовується?**

Алгоритм Беллмана-Форда визначає найкоротші шляхи в графах, зокрема тих, що мають ребра з від’ємною вагою. Він виконує оновлення відстаней до всіх вершин за V−1V-1V−1 ітерацію, де VVV — кількість вершин у графі.

**Ключові етапи роботи:**

1. Ініціалізуємо відстані до всіх вершин як нескінченність, окрім початкової, для якої встановлюється значення 0.
2. Для кожного ребра перевіряємо можливість покращення відстані до кінцевої вершини.
3. Повторюємо цей процес V−1V-1V−1 разів.

**Коли застосовувати:**

* Для графів, які містять ребра з від’ємною вагою.
* Для виявлення від’ємних циклів (якщо після V−1V-1V−1 ітерацій виявляється, що відстані ще змінюються, це свідчить про наявність від’ємного циклу).

Хоча цей алгоритм менш ефективний для великих графів, ніж алгоритм Дейкстри, він є необхідним для задач, пов’язаних із від’ємними вагами.

**5. Як працює алгоритм Флойда-Воршелла, його переваги та недоліки?**

Алгоритм Флойда-Воршелла дозволяє обчислити найкоротші шляхи між усіма парами вершин у графі. Це універсальний підхід для розв’язання задачі найкоротших шляхів для всіх пар.

**Основні етапи роботи:**

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

**Переваги:**

* Працює з графами, що мають від’ємні ваги ребер (за умови відсутності від’ємних циклів).
* Забезпечує знаходження найкоротших шляхів між усіма парами вершин.

**Недоліки:**

* Має часову складність O(V3)O(V^3)O(V3), де VVV — кількість вершин, що робить алгоритм неефективним для великих графів.
* Для задачі пошуку найкоротшого шляху між однією парою вершин алгоритм Дейкстри є швидшим і ефективнішим.