МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Кафедра комп'ютерної інженерії та електроніки

ЗВІТ З ПРАКТИЧНИХ РОБІТ

з навчальної дисципліни

«Алгоритми та методи обчислень»

Тема «Графи. Найкоротші шляхи»

Студент гр. КІ-23-1 ПІБ Кобець О. О.

Кременчук 2024

Практична робота № 6

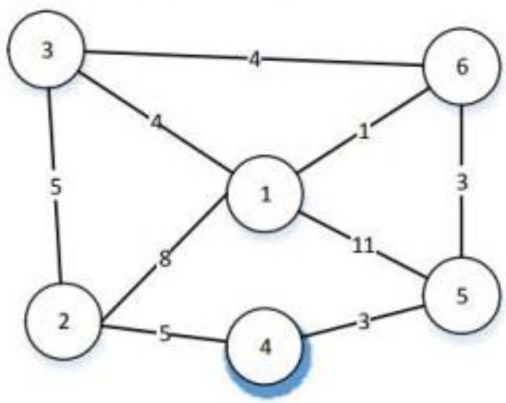
Тема. Графи. Найкоротші шляхи

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

Завдання

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

4. Алгоритм Дейкстри



Вершини: {1,2,3,4,5,6}

Ребра:

1→2 (вага: 8)

1→3 (вага: 4)

1→4 (вага: 11)

2→4 (вага: 5)

3→2 (вага: 5)

3→6 (вага: 4)

4→5 (вага: 3)

6→5 (вага: 3)

```python
from math import inf

vertices = 6

edges = [
    (1, 2, 8),
    (1, 3, 4),
    (1, 4, 11),
    (2, 4, 5),
    (3, 2, 5),
    (3, 6, 4),
    (4, 5, 3),
    (6, 5, 3),
]

distances = [inf] * (vertices + 1)
distances[1] = 0

for _ in range(vertices - 1):
    for u, v, weight in edges:
        if distances[u] != inf and distances[u] + weight < distances[v]:
            distances[v] = distances[u] + weight

negative_cycle = False
for u, v, weight in edges:
    if distances[u] != inf and distances[u] + weight < distances[v]:
        negative_cycle = True
        break

distances, negative_cycle

print("Результати роботи алгоритму Беллмана-Форда")
print("Найкоротші відстані від вершини 1 до всіх інших:")
for vertex, distance in enumerate(distances[1:], start=1):
    print(f"До вершини {vertex} : {distance}")
```

```
Результати роботи алгоритму Беллмана-Форда
Найкоротші відстані від вершини 1 до всіх інших:
До вершини 1 : 0
До вершини 2 : 8
До вершини 3 : 4
До вершини 4 : 11
До вершини 5 : 11
До вершини 6 : 8

Process finished with exit code 0
```

Контрольні питання

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

Граф – це математична структура, яка складається з набору **вершин** (або вузлів) і **ребер** (або дуг), які з'єднують ці вершини. Графи застосовуються для моделювання відносин або зв’язків між об’єктами.

Головні складові графа:

- **Вершини (Nodes, Vertices):** Об’єкти, між якими встановлюються зв’язки (наприклад, міста, сторінки в інтернеті).
- **Ребра (Edges):** Зв’язки між вершинами (наприклад, дороги, посилання). Ребра можуть бути:
  - **Неорієнтованими:** Зв’язок без напрямку.
  - **Орієнтованими:** Зв’язок має напрямок.
  - **Зваженими:** Ребра мають вагу, що визначає "вартість" або "довжину" зв’язку.

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

Основні алгоритми:

1. **Алгоритм Дейкстри:** Шукає найкоротші шляхи від однієї вершини до всіх інших у графі з додатними вагами.
2. **Алгоритм Беллмана–Форда:** Застосовується для графів з вагами, які можуть бути від’ємними.
3. **Алгоритм Флойда–Форшала:** Використовується для знаходження найкоротших шляхів між усіма парами вершин.

4. **Алгоритм Джонсона:** Ефективний для великих графів і працює з вагами, які можуть бути від'ємними.
5. **Пошук у ширину (BFS):** Застосовується для незважених графів.

3. Як працює алгоритм Дейкстри і які його особливості?

Алгоритм Дейкстри:

- Знаходить найкоротший шлях від однієї стартової вершини до всіх інших вершин у графі.
- Працює лише для графів з **додатними вагами ребер**.

Основні кроки:

1. Ініціалізувати відстань до стартової вершини як 0, а до інших вершин — як нескінченність.
2. Створити множину відвіданих вершин.
3. Для кожної невідвіданої вершини оновлювати відстані до її сусідів, якщо новий шлях коротший.
4. Вибрати вершину з найменшою відстанню та позначити її як відвідану.
5. Повторювати, поки всі вершини не будуть оброблені.

Особливості:

- Швидкість залежить від реалізації: $O(V^2)$ для матриці суміжності або $O((V+E)\log V)$ для черги з пріоритетами.
- Не працює з від'ємними вагами ребер.

4. Що таке алгоритм Белмена–Форда і коли його варто застосовувати?

Алгоритм Беллмана–Форда:

- Шукає найкоротші шляхи від однієї вершини до всіх інших.
- Підтримує графи з від'ємними вагами ребер.

Основні кроки:

1. Ініціалізувати відстань до стартової вершини як 0, а до інших вершин — як нескінченність.
2. Повторити $V-1$ разів (де V — кількість вершин): перевірити кожне ребро, чи можливо зменшити відстань до кінцевої вершини через поточну.

3. Перевірити, чи існує цикл із від’ємною вагою (за V-ю ітерацією).

Коли застосовувати?

- Коли у графі можуть бути від’ємні ваги ребер.
- Коли важлива простота реалізації (повільніше, ніж алгоритм Дейкстри для графів без від’ємних ваг).

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

Алгоритм Флойда–Форшала:

- Знаходить найкоротші шляхи між усіма парами вершин у графі.
- Підходить для графів з від’ємними вагами, але без циклів із від’ємною вагою.

Основні кроки:

1. Побудувати матрицю відстаней D, де D[i][j] — вага ребра між вершинами i та j (або нескінченність, якщо ребра немає).
2. Ітеративно оновлювати матрицю: для кожної вершини kkk, перевіряти, чи шлях i→k→j коротший, ніж i→j.
3. Після V ітерацій отримуємо матрицю найкоротших відстаней.

Переваги:

- Простота реалізації.
- Підходить для знаходження найкоротших шляхів між усіма парами вершин.

Недоліки:

- Часова складність O(V3) обмежує використання для великих графів.
- Вимагає O(V2) пам’яті для зберігання матриці.