МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Кафедра комп’ютерної інженерії та електроніки

Виконала: Групи

Пугачова Д. В. КІ-23-1

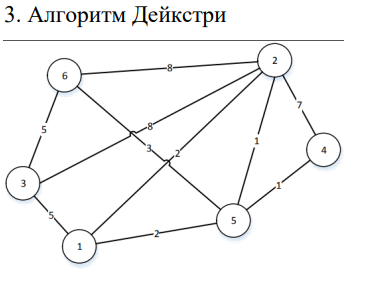
Кременчук 2024

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:



Почнемо з вершини 1.

Встановимо відстань до вершини 1 як 0 (оскільки це початкова точка) і нескінченність для всіх інших вершин.

Встановіть попередник кожної вершини на «невизначений».

Ітерація:

Вершина 1 :

Поточний найкоротший шлях: 0

Оновимо відстані до сусідів:

Вершина 3: Відстань = 5

Вершина 5: Відстань = 1

Позначимо вершину 1 як «відвідану».

Вершина 5 (наступна найближча вершина):

Поточний найкоротший шлях: 1

Оновимо відстані до сусідів:

Вершина 2: Відстань = 2 (1 + 1)

Вершина 4: Відстань = 2 (1 + 1)

Позначимо вершину 5 як «відвідану».

Вершина 2 (наступна найближча вершина):

Поточний найкоротший шлях: 2

Оновимо відстані до сусідів:

Вершина 6: Відстань = 10 (2 + 8)

Позначте вершину 2 як «відвідану».

Вершина 4 (наступна найближча вершина):

Поточний найкоротший шлях: 2

Оновлення не потрібні (лише ребро до вершини 5, яке вже відвідано).

Позначимо вершину 4 як «відвідану».

Вершина 3 (наступна найближча вершина):

Поточний найкоротший шлях: 5

Оновлення не потрібні (лише ребро до вершини 6, яка вже має меншу відстань).

Позначимо вершину 3 як «відвідану».

Вершина 6 (остання вершина):

Поточний найкоротший шлях: 10

Немає сусідів для оновлення.

Позначимо вершину 6 як «відвідану».

Останні найкоротші шляхи

Вершина 1: 0

Вершина 2: 2

Вершина 3: 5

Вершина 4: 2

Вершина 5: 1

Вершина 6: 10

Це забезпечує найкоротший шлях від вершини 1 до всіх інших *вершин у графі.*

*Відповідь :*

*Вершина 1: 0*

*Вершина 2: 2*

*Вершина 3: 5*

*Вершина 4: 2*

*Вершина 5: 1*

*Вершина 6: 10*

**Контрольні питання**

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

***Граф*** *— це математична структура, яка складається з набору вершин (або вузлів) та набору ребер, що з'єднують пари вершин. Граф може бути орієнтованим або неорієнтованим.*

***Основні складові графа****:*

* ***Вершини (вершини)****: елементи графа, які можуть представляти будь-які об'єкти (наприклад, міста, комп'ютери, люди).*
* ***Ребра (зв'язки, дуги)****: зв'язки між вершинами, що можуть бути орієнтованими або неорієнтованими. У орієнтованих графах ребра мають напрямок (з однієї вершини до іншої).*
* ***Ваги ребер (для графів з вагами)****: числові значення, що призначаються ребрам і можуть представляти, наприклад, відстань, вартість чи час, пов'язаний із переходом між вершинами.*

*Графи можуть бути* ***орієнтованими*** *(кожне ребро має напрямок) або* ***неорієнтованими*** *(ребра не мають напрямку).*

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

* ***Алгоритм Дейкстри****: знаходить найкоротші шляхи від однієї початкової вершини до всіх інших вершин графа з невід'ємними вагами ребер.*
* ***Алгоритм Беллмана-Форда****: використовується для графів з від'ємними вагами на ребрах і здатний виявляти цикли від'ємної ваги.*
* ***Алгоритм Флойда-Форшала****: знаходить найкоротші шляхи між усіма парами вершин у графі, підходить для повних графів.*
* ***Алгоритм A\*****: використовується в задачах пошуку шляху в графах з додатковими евристиками (наприклад, для оптимізації пошуку в іграх або на картах).*

**3. Як працює алгоритм Дейкстри і які його особливості?**

***Алгоритм Дейкстри*** *— це жадібний алгоритм для знаходження найкоротших шляхів від однієї вершини до всіх інших у графі з невід'ємними вагами ребер.*

*Принцип роботи:*

1. *Починаємо з початкової вершини, надаючи їй вагу 0, а всім іншим — нескінченність.*
2. *Для кожної вершини, до якої ми можемо дістатись через її сусідів, оновлюємо найкоротший шлях (якщо знайдений шлях коротший, ніж поточний).*
3. *Вибираємо вершину з мінімальною вагою з усіх неперевірених вершин і повторюємо процес, поки не будемо мати найкоротші шляхи до всіх вершин.*

***Особливості****:*

* *Працює тільки для графів з* ***невід'ємними вагами ребер****.*
* *Протягом виконання алгоритм вибирає вершини за принципом мінімуму з поточних, не змінюючи обрані шляхи.*
* *Якщо граф містить від'ємні ваги, алгоритм може працювати неправильно.*

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

***Алгоритм Беллмана-Форда*** *— це алгоритм для знаходження найкоротших шляхів від однієї вершини до всіх інших у графах з* ***від'ємними вагами ребер****.*

***Коли варто застосовувати****:*

* *Коли в графі можуть бути* ***від'ємні ваги****.*
* *Коли потрібно виявити* ***цикли з від'ємною вагою*** *(алгоритм може це зробити).*
* *Може бути менш ефективним, ніж алгоритм Дейкстри для графів з невід'ємними вагами, оскільки має складність O(VE)O(VE)O(VE).*

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

***Алгоритм Флойда-Форшала*** *— це алгоритм для знаходження найкоротших шляхів між усіма парами вершин у графі.*

*Принцип роботи:*

1. *Створюємо матрицю відстаней, де кожен елемент dist[i][j]] є відстанню від вершини i до вершини j.*
2. *Початково, якщо є ребро між вершинами i і j, то dist[i][j] дорівнює вазі ребра, інакше — нескінченність.*
3. *Для кожної вершини k, по черзі оновлюємо відстані між кожною парою вершин i і j через k як проміжну вершину, якщо шлях через k коротший.*

***Переваги****:*

* *Відмінно підходить для задач, де потрібно знайти* ***найкоротші шляхи між всіма парами вершин****.*
* *Може працювати з графами, що містять від'ємні ваги (якщо немає циклів з від'ємною вагою).*

***Недоліки****:*

* *Має складність O(V3), що робить його повільним для великих графів.*
* *Може бути неефективним для графів, де потрібно знайти найкоротші шляхи тільки від однієї вершини (для цього кращі алгоритми, такі як Дейкстри).*