МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ   
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

НАВЧАЛЬНА ДИСЦИПЛІНА  
«**АЛГОРИТМИ І СТРУКТУРИ ДАННИХ**»

ЗВІТ

З ПРАКТИЧНОЇ РОБОТИ №6

Виконав

студент групи КН-24-1

Процко П. Д.

Перевірив

доцент кафедри КІЕ

Сидоренко В. М.

Кременчук 2025

| Тема: | Графи. Найкоротші шляхи |
| --- | --- |
| Мета: | набути практичних навичок розв’язання задач пошуку найкоротших шляхів у графі та оцінювання їх асимптотичної складності. |

Хід роботи

1. **Постановка завдання**

Завдання полягає у знаходженні найкоротших шляхів від вершини 1 до всіх інших за допомогою алгоритму Белмена–Форда.

1. **Розв’язання задачі**
2. Задано граф

Зображення, що містить ряд, коло

Вміст, створений ШІ, може бути неправильним.

Рисунок 1 – Граф

1. Реалізація алгоритму Белмена–Форда

def bellman\_ford(graph, source):  
  
 # Знаходимо всі вершини графа  
 vertices = set()  
 for u, v, \_ in graph:  
 vertices.add(u)  
 vertices.add(v)  
  
 # Крок 1: Ініціалізація  
 distances = {vertex: float('inf') for vertex in vertices}  
 distances[source] = 0  
  
 # Крок 2: Релаксація ребер n-1 разів  
 n = len(vertices)  
 for \_ in range(n - 1):  
 for u, v, weight in graph:  
 if distances[u] != float('inf') and distances[u] + weight < distances[v]:  
 distances[v] = distances[u] + weight  
  
 # Крок 3: Перевірка на від'ємний цикл  
 has\_negative\_cycle = False  
 for u, v, weight in graph:  
 if distances[u] != float('inf') and distances[u] + weight < distances[v]:  
 has\_negative\_cycle = True  
 break  
  
 return distances, has\_negative\_cycle

1. Тестування коду

# Граф представлений як список ребер: (початок, кінець, вага)  
edges = [  
 (1, 2, 2),  
 (1, 3, 5),  
 (1, 5, 2),  
 (2, 3, 8),  
 (2, 4, 7),  
 (2, 5, 1),  
 (2, 6, 8),  
 (3, 6, 5),  
 (4, 5, 1),  
 (5, 6, 3)  
]  
  
# Запуск алгоритму  
source = 1  
distances, has\_negative\_cycle = bellman\_ford(edges, source)  
  
# Виведення результатів  
if has\_negative\_cycle:  
 print("У графі виявлено від'ємний цикл!")  
else:  
 print(f"Найкоротші відстані від вершини {source}:")  
 for vertex in sorted(distances.keys()):  
 print(f"До вершини {vertex}: {distances[vertex]}")

1. Результат виконання

Найкоротші відстані від вершини 1:

До вершини 1: 0

До вершини 2: 2

До вершини 3: 5

До вершини 4: 9

До вершини 5: 2

До вершини 6: 5

1. **Відповіді на контрольні питання**
2. Що таке граф і які головні складові його структури?

Граф - це математична структура, що складається з вершин (вузлів) та ребер (зв'язків між вершинами). Головні складові: вершини, ребра (можуть бути напрямленими або ненапрямленими), ваги ребер (у зважених графах).

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

* Алгоритм Дейкстри
* Алгоритм Беллмана-Форда
* Алгоритм Флойда-Воршалла
* Алгоритм A\* (А-зірочка)
* Алгоритм Джонсона

1. Як працює алгоритм Дейкстри і які його особливості?

* Працює за принципом жадібного вибору найближчої непройденої вершини
* Поступово розширює множину вершин з відомими найкоротшими шляхами
* Особливості: працює тільки для графів з невід'ємними вагами, ефективний для розріджених графів, складність O(E log V) з пріоритетною чергою

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

* Працює шляхом послідовної релаксації всіх ребер графа
* Варто застосовувати, коли в графі можуть бути ребра з від'ємною вагою
* Дозволяє виявляти від'ємні цикли
* Складність O(V×E), повільніший за Дейкстру, але більш універсальний

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

* Динамічний алгоритм для пошуку найкоротших шляхів між усіма парами вершин
* Працює шляхом послідовного покращення шляхів через проміжні вершини
* Переваги: простота реалізації, працює з від'ємними вагами (але без від'ємних циклів)
* Недоліки: складність O(V³), неефективний для великих розріджених графів