УДК 004.42

Сіклічук А. С., здобувач вищої освіти;
Сеник І. О., асистент кафедри інформаційних і прикладних технологій,
Донецький національний університет імені Василя Стуса

ЗАСТОСУВАННЯ АЛГОРИТМУ ДЕЙКСТРИ ДЛЯ ПОШУКУ ОПТИМАЛЬНОГО МАРШРУТУ

Ключові слова: алгоритм, маршрут, граф, вершини, вага.

Вступ. У сучасному світі, де мобільність стає необхідним складником нашого повсякденного життя, розробка ефективних та оптимальних маршрутів переміщення є важливою задачею. Розроблений голландським вченим Едсгером Дейкстрою в 1956 році, алгоритм Дейкстри визнаний світовою спільнотою як ефективний та надійний метод для визначення оптимальних маршрутів. Він знайшов широке застосування в різних галузях, зокрема у транспорті, логістиці, телекомунікаціях та інформаційних системах. Однією з головних переваг алгоритму Дейкстри є його спроможність працювати з графами з неорієнтованими або орієнтованими ребрами, що робить його універсальним інструментом для розв’язання різноманітних задач. Його застосування у мобільних додатках для навігації, в системах управління транспортом та електронних картографічних сервісах відзначається високою ефективністю та можливістю адаптації до різноманітних задач маршрутизації.

Актуальність. Актуальність застосування алгоритму Дейкстри для пошуку оптимальних маршрутів надзвичайно висока в контексті сучасного життя та технологій. Наприклад, цей алгоритм може працювати з числами чи координатами, він може використовуватися в додатках для навігації та знаходити найкоротші шляхи з пункту А до пункту В. Особливо зі збільшенням населення міст і розширенням транспортної інфраструктури стає важливим налаштувати ефективне управління маршрутами для запобігання заторів та забезпечення оптимального використання ресурсів.

Загалом графи у програмуванні – це зручний спосіб зберігання певних типів даних. Концепція графа була перенесена з математики та адаптована під потреби інформатики. Внаслідок цього обхід графів став звичайним завданням, що використовується в науці про дані та машинне навчання. Для цього дослідження обрано неорієнтований граф, що має 6 вершин. Деякі вершини поєднанні між собою ребрами, що мають певну вагу, і у реальних моделях можуть репрезентувати відстань від однієї точки до іншої (рис. 1).

Задача полягає у знаходженні найкоротших відстаней від певної вершини до кожної іншої вершини графа за допомогою алгоритму Дейкстри, реалізованого на мові програмування Python. Робота передбачає використання вбудованих функцій Python та бібліотек, як-от heapq.

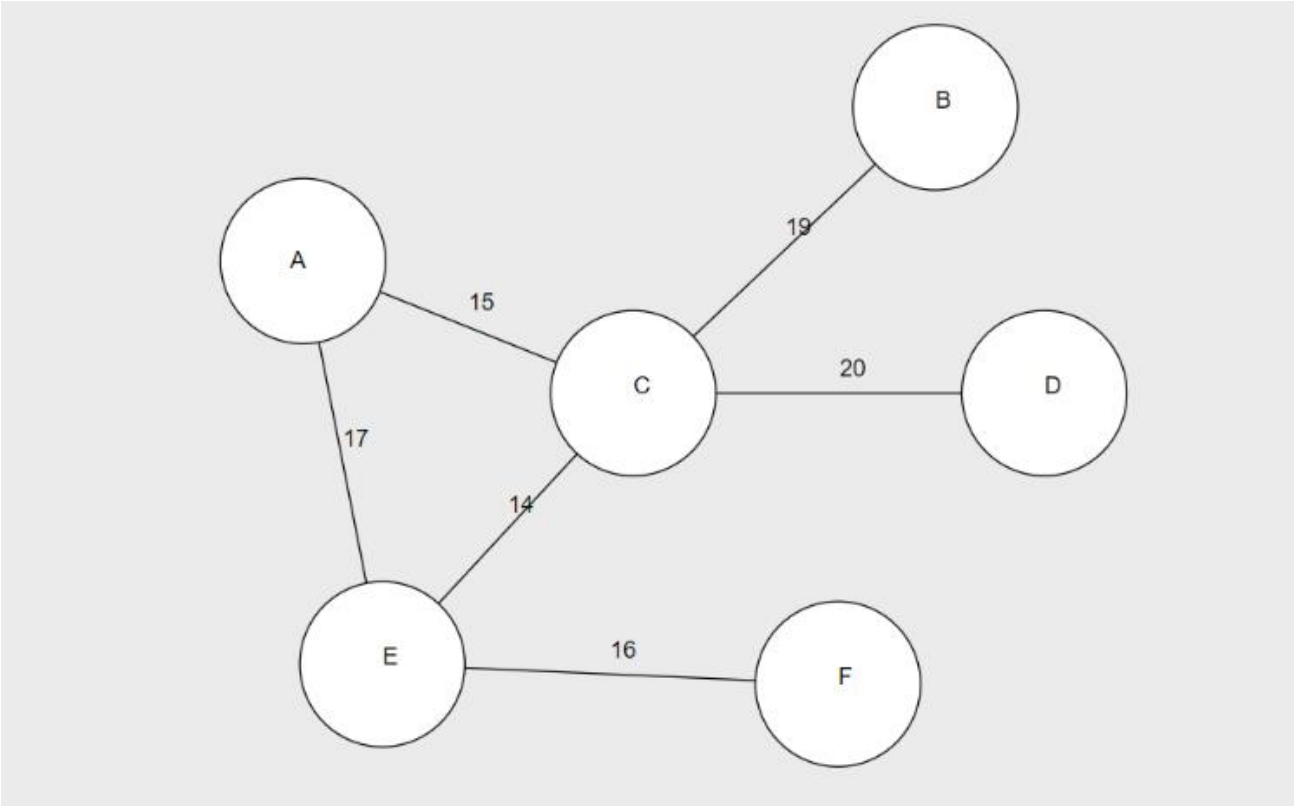

Рисунок 1 – Неорієнтований граф

Функція ‘dijkstra’ використовує алгоритм Дейкстри для знаходження найкоротших шляхів від вказаної вершини ‘start_vertex’ до всіх інших вершин у графі. Ваги ребер (відстані) задані у вигляді словника graph. Важливо враховувати, що алгоритм Дейкстри не працює коректно, якщо є ребра з від’ємними вагами. Структура програми виглядає так [Лістинг програми]:

```
import heapq

graph = {
    ‘A’: {‘C’: 15, ‘E’: 17},
    ‘B’: {‘C’: 19},
    ‘C’: {‘A’: 15, ‘E’: 14, ‘D’: 20, ‘B’: 19},
    ‘D’: {‘C’: 20},
    ‘E’: {‘A’: 17, ‘F’: 16, ‘C’: 14},
    ‘F’: {‘E’: 16}
}

def dijkstra(graph, start):
    distances = {vertex: float(‘infinity’) for vertex in graph}
    distances[start] = 0
    priority_queue = [(0, start)]

    while priority_queue:
        current_distance, current_vertex = heapq.heappop(priority_queue)
```

```
        if current_distance > distances[current_vertex]:
            continue

        for neighbor, weight in graph[current_vertex].items():
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(priority_queue, (distance, neighbor))

    return distances

start_vertex = ‘B’

shortest_distances = dijkstra(graph, start_vertex)

for vertex, distance in shortest_distances.items():
    print(f"Відстань від {start_vertex} до {vertex}: {distance}"
    )
```

Лістинг програми

Вершину, від якої ми шукаємо найкоротший шлях, можна змінювати залежно від постановки задачі.

Внаслідок виконання цієї програми ми отримаємо таку відповідь:

```
Відстань від B до A: 34
Відстань від B до B: 0
Відстань від B до C: 19
Відстань від B до D: 39
Відстань від B до E: 33
Відстань від B до F: 49
```

Така програма підходить у випадку, коли потрібно знайти найкоротший шлях від заданого міста, наприклад, до інших міст, представлених у формі графу з вагами.

Для порівняння результатів призначимо змінній ‘start_vertex’ інше значення. У цьому випадку ми будемо розглядати відстані від точки E. Внаслідок цього отримаємо:

```
Відстань від E до A: 17
Відстань від E до B: 33
Відстань від E до C: 14
Відстань від E до D: 34
Відстань від E до E: 0
Відстань від E до F: 16
```

Висновки

Проаналізувавши результати цієї програми, можна зробити висновок, що знайдені маршрути є оптимальними за довжиною, враховуючи ваги ребер. Ця

реалізація дала змогу визначити найкоротші шляхи від обраної початкової вершини до всіх інших вершин графа. Також вона може бути застосована до графів різного розміру та складності, що дає змогу використовувати алгоритм для розв'язання широкого спектру задач. Результати програми можуть мати практичне застосування у галузях маршрутизації в мережах, логістики, транспортного планування та інших, де важливо оптимізувати шляхи та витрати.

Список використаних джерел

1. Алгоритм Дейкстри. URL: http://choippo.cn.sch.in.ua/Files/downloadcenter/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%20%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B8.%20%D0%A2%D0%B5%D0%BE%D1%80%D1%96%D1%8F.pdf (дата звернення: 15.11.2023).

2. Бартіш М. Я., Дудзяний І. М. Дослідження операцій. Частина 2. Алгоритми оптимізації на графах: навч.-метод. посіб. Львів, 2007. 120 с.

3. Dijkstra's Algorithm. URL: https://www.programiz.com/dsa/dijkstra-algorithm (дата звернення: 16.11.2023).

4. Understanding Dijkstra's Algorithm in Python. URL: https://pieriantraining.com/understanding-dijkstras-algorithm-in-python/ (дата звернення: 16.11.2023).