УДК 519.17:004.021

Радченко Р. П., здобувач вищої освіти,
Сеник І. О., асистент кафедри
інформаційних технологій

ГРАФИ ТА АЛГОРИТМИ ПОШУКУ: ТЕОРІЯ, ЗАСТОСУВАННЯ ТА ОПТИМІЗАЦІЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Графи є універсальною структурою даних, яка дає змогу моделювати широкий спектр реальних об’єктів та процесів, як-от соціальні мережі, транспортні системи, ієрархії. Завдяки своїй гнучкості графи використовуються для розв’язання задач, що виникають у різних галузях комп’ютерних наук.

Основу роботи з графами становлять алгоритми пошуку. Серед них найпоширенішими є пошук у ширину (BFS) та пошук у глибину (DFS). Ці алгоритми допомагають обійти всі вершини графу, визначити його зв’язність, знайти шляхи між заданими вершинами та вирішити інші базові задачі.

Для знаходження найкоротших шляхів у графах використовуються більш складні алгоритми, як-от алгоритм Дейкстри та алгоритм Беллмана–Форда. Алгоритм Дейкстри забезпечує ефективне знаходження найкоротших шляхів у графах з невід’ємними вагами, тоді як алгоритм Беллмана–Форда здатен працювати з графами, що містять від’ємні ваги.

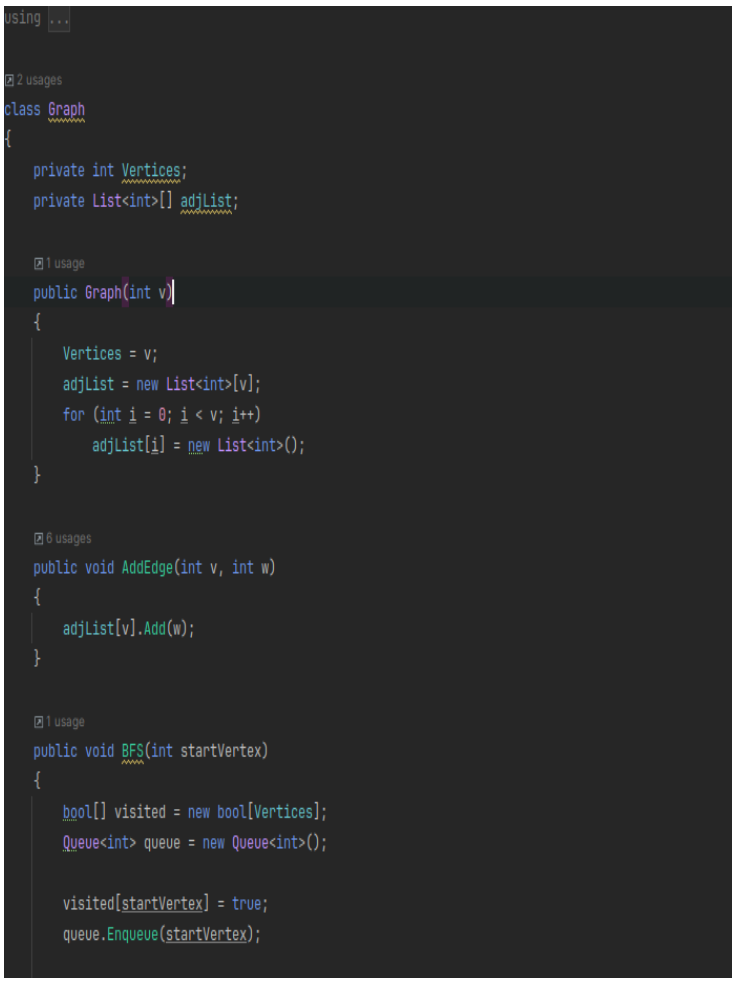
Алгоритми на графах знаходять своє застосування в реальних системах. Наприклад, вони широко використовуються для оптимізації маршрутів у транспортних мережах, де важливо мінімізувати час або вартість переміщення. У соціальних мережах графові алгоритми дають змогу аналізувати зв’язки між користувачами, що допомагає у виявленні впливових осіб або побудові рекомендаційних систем. Важливою задачею є побудова мінімального кістякового дерева. Алгоритми Пріма та Крускала допомагають знайти дерево, яке з’єднує всі вершини графу з мінімальною загальною вагою. Це має практичне значення у проєктуванні мереж, наприклад, електричних чи комунікаційних.

Аналіз складності алгоритмів на графах є ключовим аспектом їх застосування. Залежно від розміру та властивостей графу обирають той чи інший алгоритм, який забезпечить оптимальний результат за прийнятний час.

Програма нижче демонструє базову реалізацію алгоритму BFS для графу, представленого у вигляді списку суміжності. Алгоритм проходить всі вершини графу і виводить порядок їх відвідування (рис. 1 та рис. 2).

Цей код демонструє просту реалізацію алгоритму BFS, який дає змогу проходити всі вершини графу від вказаної вершини. BFS широко використовується в задачах, що потребують знаходження найкоротшого шляху у незваженому графі або перевірки зв’язності графу [4].

Графи та алгоритми пошуку продовжують відігравати ключову роль у комп’ютерних науках і технологіях, забезпечуючи основи для ефективного вирішення складних задач у реальному світі.



```
using ...

2 usages
class Graph
{
    private int Vertices;
    private List<int>[] adjList;

    1 usage
    public Graph(int v)
    {
        Vertices = v;
        adjList = new List<int>[v];
        for (int i = 0; i < v; i++)
            adjList[i] = new List<int>();
    }

    6 usages
    public void AddEdge(int v, int w)
    {
        adjList[v].Add(w);
    }

    1 usage
    public void BFS(int startVertex)
    {
        bool[] visited = new bool[Vertices];
        Queue<int> queue = new Queue<int>();

        visited[startVertex] = true;
        queue.Enqueue(startVertex);
```

Рисунок 1 – Код реалізації алгоритму BFS для графу, представленого у вигляді списку суміжності. Частина 1

```
        while (queue.Count != 0)
        {
            startVertex = queue.Dequeue();
            Console.WriteLine("Відвідано вершину: " + startVertex);

            foreach (int nextVertex in adjList[startVertex])
            {
                if (!visited[nextVertex])
                {
                    visited[nextVertex] = true;
                    queue.Enqueue(nextVertex);
                }
            }
        }
    }
}

class Program
{
    static void Main()
    {
        Graph g = new Graph(v: 5);
        g.AddEdge(v: 0, w: 1);
        g.AddEdge(v: 0, w: 2);
        g.AddEdge(v: 1, w: 2);
        g.AddEdge(v: 2, w: 0);
        g.AddEdge(v: 2, w: 3);
        g.AddEdge(v: 3, w: 3);

        Console.WriteLine("BFS починаючи з вершини 2:");
        g.BFS(startVertex: 2);
    }
```

Рисунок 2 – Код реалізації алгоритму BFS для графу, представленого у вигляді списку суміжності. Частина 2

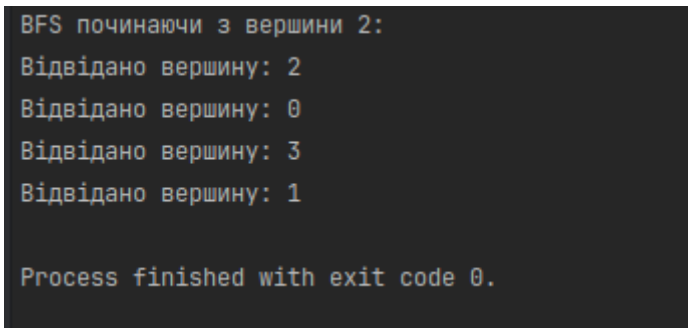


Рисунок 3 – Результат виконання коду алгоритму BFS для графу, представленого у вигляді списку суміжності

Список використаних джерел

1. BestProg. URL: https://www.bestprog.net/uk/2019/09/26/c-queue-general-concepts-ways-to-implement-the-queue-implementing-a-queue-as-a-dynamic-array-ua/ (дата звернення 30.11.2024).

2. GeeksforGeeks. URL: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/ (дата звернення 30.11.2024).

3. Vseosvita. URL: https://vseosvita.ua/lesson/rol-informatsiinykh-tekhnolohii-u-zhytti-suchasnoi-liudyny-229384.html (дата звернення 30.11.2024).

4. Програмна реалізація та дослідження алгоритмів паралельного швидкого сортування / В. О. Денисюк, Н. А. Потапова, О. В. Зелінська, М. Б. Тарасюк. *Вісник Хмельницького національного університету. Технічні науки.* 2023. № 4. С. 95–105. URL: http://journals.khnu.km.ua/vestnik/?page_id=41

УДК 004.422.5.

Ратушний О. С., здобувач вищої освіти,
Сеник І. О., асистент кафедри
інформаційних технологій

ОСОБЛИВОСТІ ВИКОРИСТАННЯ СТРУКТУРИ ДАНИХ «ГРАФ» У РОЗРОБЦІ ВІДЕОІГОР

Донецький національний університет імені Василя Стуса, м. Вінниця

Граф – це структура даних, що складається з вершин (вузлів) і зв’язків між ними (ребер). Ця структура широко застосовується у комп’ютерних науках для моделювання взаємодій між об’єктами, оскільки вона дає змогу представляти складні системи у вигляді абстрактних моделей. У розробці відеоігор графи відіграють важливу роль, забезпечуючи механізми для навігації, генерації ігрових світів, роботи зі штучним інтелектом та інших ключових функцій [1, 2].

Графи особливо корисні для моделювання ігрових світів. Наприклад, карта гри може бути представлена як граф, де вершини відповідають локаціям, а ребра – шляхам між ними. Алгоритми пошуку найкоротшого шляху, як-от A* або Дейкстри, допомагають персонажам ефективно переміщатися ігровим середовищем, що важливо для створення реалістичних ігрових сценаріїв [3]. Навіть у випадку процедурної генерації рівнів графи використовуються для зв’язування окремих кімнат або областей, створюючи цілісну ігрову карту.