УДК 004.2

Скороход О. М., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Якубич К. О., *асистент кафедри інформаційних технологій*

АЛГОРИТМИ НАЙКОРОТШОГО ШЛЯХУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Проблема пошуку найкоротшого шляху в графах є важливим складником теорії графів та має велике застосування в різних галузях. Це завдання полягає у пошуку найбільш оптимального маршруту або шляху з однієї вершини графу до іншої, такого, який має мінімальну суму ваг ребер. Відмінності у вагах ребер та типах графів визначають різні підходи до розв'язання цієї проблеми.

Граф – це математична структура, що складається з вершин (вузлів) та ребер (зв'язків), які їх з'єднують. Граф може бути орієнтованим, коли кожне ребро має напрямок, або неорієнтованим, коли напрямок не враховується.

Найкоротший шлях – це мінімальний за вагою шлях між двома вершинами у графі. Це поняття може мати різне значення залежно від конкретної мети: від мінімальної кількості ребер, яку необхідно пройти, до мінімальної суми ваг ребер на шляху.

Класифікація алгоритмів може бути проведена за ознаками:

1. За кількістю вершин:

– одноточковий: шукає найкоротші шляхи від однієї конкретної вершини до всіх інших у графі;

– багатоточковий: визначає найкоротші шляхи між усіма парами вершин.

2. За типом графу:

– неорієнтований: граф, де ребра не мають напрямку;

– орієнтований: граф, де ребра мають напрямок.

3. За характером ваги ребер:

– з додатними вагами: всі ваги ребер є додатними числами або нулем;

– з від'ємними вагами: деякі ребра можуть мати від'ємні ваги.

Розглянемо основні алгоритми пошуку найкоротшого шляху:

1. Алгоритм Дейкстри. Це один з найефективніших алгоритмів для пошуку найкоротших шляхів у графі з невід'ємними вагами. Він працює в часовому складі $O(V2)$O(V2), де VV – кількість вершин у графі.

2. Алгоритм Беллмана–Форда. Цей алгоритм може використовуватися в графах з від'ємними вагами. Він також здатний виявити від'ємні цикли. Однак його часова складність складає $O(VE)$O(VE), що робить його менш ефективним за алгоритм Дейкстри у багатьох випадках.

3. Алгоритм Флойда–Воршелла. Цей алгоритм використовується для знаходження найкоротших шляхів між усіма парами вершин у графі. Його часова складність становить $O(V3)$O(V3).

Розлянемо реалізацію алгоритму Дейкстри на мові програмування C#. Він використовує структуру даних – список суміжності – для представлення графу та пріоритетну чергу для ефективного вибору наступної вершини для дослідження.

```
using System;
using System.Collections.Generic;

class Graph
{
   private int Vertices;
   private List<Tuple<int, int>>[] adjacencyList;

   public Graph(int vertices)
   {
      Vertices = vertices;
      adjacencyList = new List<Tuple<int, int>>[vertices];
      for (int i = 0; i < vertices; i++)
      {
         adjacencyList[i] = new List<Tuple<int, int>>();
      }
   }

   public void AddEdge(int u, int v, int weight)
   {
      adjacencyList[u].Add(Tuple.Create(v, weight));
   }

   public void Dijkstra(int source)
   {
      var distance = new int[Vertices];
      var shortestPathTreeSet = new bool[Vertices];
      var priorityQueue = new SortedSet<Tuple<int, int>>(Comparer<Tuple<int, int>>.Create((x, y) =>
      x.Item1 == y.Item1 ? x.Item2.CompareTo(y.Item2) : x.Item1.CompareTo(y.Item1)));

      for (int i = 0; i < Vertices; i++)
      {
         distance[i] = int.MaxValue;
         shortestPathTreeSet[i] = false;
      }

      distance[source] = 0;
      priorityQueue.Add(Tuple.Create(0, source));

      while (priorityQueue.Count > 0)
      {
         var minNode = priorityQueue.Min;
         priorityQueue.Remove(minNode);

         int u = minNode.Item2;

         if (shortestPathTreeSet[u]) continue;

         shortestPathTreeSet[u] = true;
```

```
            foreach (var adjacent in adjacencyList[u])
            {
                int v = adjacent.Item1;
                int weight = adjacent.Item2;

                if (!shortestPathTreeSet[v] && distance[u] != int.MaxValue && distance[u] + weight < distance[v])
                {
                    distance[v] = distance[u] + weight;
                    priorityQueue.Add(Tuple.Create(distance[v], v));
                }
            }
        }

        Print(distance);
    }

    private void Print(int[] distance)
    {
        Console.WriteLine("Вершина \t Відстань від джерела");
        for (int i = 0; i < Vertices; i++)
        {
            Console.WriteLine("{0} \t\t {1}", i, distance[i]);
        }
    }
}

class Program
{
    static void Main()
    {
        Graph g = new Graph(9);

        g.AddEdge(0, 1, 4);
        g.AddEdge(0, 7, 8);
        g.AddEdge(1, 2, 8);
        g.AddEdge(1, 7, 11);
        g.AddEdge(2, 3, 7);
        g.AddEdge(2, 8, 2);
        g.AddEdge(2, 5, 4);
        g.AddEdge(3, 4, 9);
        g.AddEdge(3, 5, 14);
        g.AddEdge(4, 5, 10);
        g.AddEdge(5, 6, 2);
        g.AddEdge(6, 7, 1);
        g.AddEdge(6, 8, 6);
        g.AddEdge(7, 8, 7);

        g.Dijkstra(0);
    }
}
```

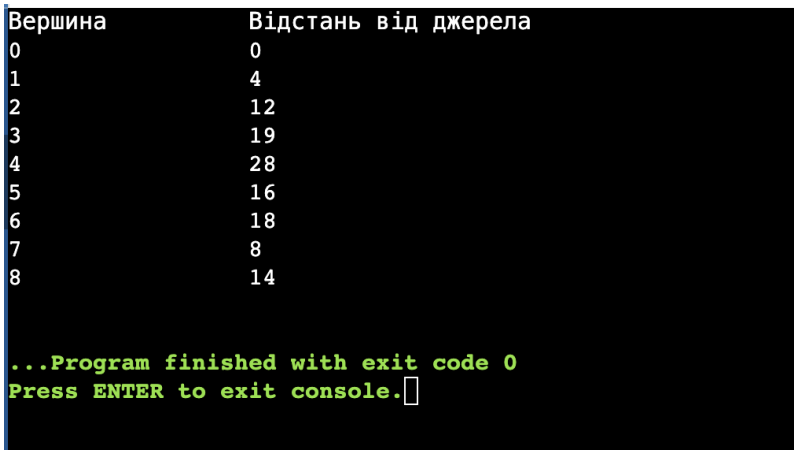

Рис. 1. Результат реалізації алгоритму Дейкстри

Отже, алгоритми найкоротшого шляху є потужними інструментами для вирішення широкого спектра задач оптимізації. Вибір конкретного алгоритму залежить від типу графу, ваг ребер та необхідної інформації (найкоротший шлях до однієї вершини чи до всіх).

Список використаних джерел

1. Ільман В. М., Іванов О. П., Панік Л. О. Алгоритми та структури даних: навчальний посібник. Дніпро. 2019. URL: https://crust.ust.edu.ua/server/api/core/bitstreams/16eada7c-c082-46f7-af81-1d6e97ac9320/content

2. Вікіпедія. Алгоритм Дейкстри. Відредаговано 4 січня 2024. URL: https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B8 (дата звернення: 20.05.2024).

3. Креневич А. Алгоритми і структури даних. Підручник. Київ: ВПЦ «Київський Університет», 2021. 200 с.

УДК 004.9

Тронь Д. В., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
науковий керівник:
Волонтир Л. О., *старший викладач кафедри інформаційних технологій*

ЧИСЕЛЬНІ МЕТОДИ РОЗВ'ЯЗАННЯ НЕЛІНІЙНИХ РІВНЯНЬ НА C#

Донецький національний університет імені Василя Стуса, м. Вінниця

Нелінійні рівняння зустрічаються у багатьох наукових та інженерних задачах. Проблема розв'язання нелінійних рівнянь полягає в тому, що багато з них не мають аналітичних розв'язків або їх знаходження є дуже складним. Це вимагає застосування чисельних методів для знаходження наближених розв'язків. Різні методи обчислень дають змогу отримувати точні та швидкі результати, роблячи їх незамінними в сучасних обчисленнях. Серед найбільш популярних методів є метод Ньютона та метод хорд, які широко застосовуються в різних галузях. Важ-