МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет ИТМО

Лабораторная работа №3

«Алгоритмы на графах»

Цель работы: Научиться реализовывать алгоритмы на графах. Реализовать средствами ООП поиск в глубину и ширину на графе. Реализовать средствами ООП алгоритм нахождение кратчайших путей (Алгоритм Дейкстры). Реализовать алгоритм Крускала.

Выполнила студентка группы №K3222

Маркозубова Анастасия Кирилловна

Задание

Задание 1. Научиться реализовывать алгоритмы на графах.

Задание 2. Реализовать средствами ООП поиск в глубину и ширину на графе.

Задание 3. Реализовать средствами ООП алгоритм нахождение кратчайших путей (Алгоритм Дейкстры). Реализовать алгоритм Крускала.

Задание 1

Реализовать поиск в глубину и в ширину. Выполнить расчет примера.

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

class Program
{
    static int n = 6;
    static int[,] g = new int[n, n];
    static bool[] visited;

    static void Main()
    {
        g[0, 3] = g[3, 0] = 1;
        g[0, 4] = g[4, 0] = 1;
        g[1, 5] = g[5, 1] = 1;
        g[2, 4] = g[2, 4] = 1;
        g[2, 5] = g[5, 2] = 1;
        g[4ё, 4] = g[4, 4] = 1;

        Console.Write("   ");
        for (int j = 0; j < n; j++)
        {
            Console.Write($"{j,3}");
        }
        Console.WriteLine("\n   " + new string('-', n * 3));

        for (int i = 0; i < n; i++)
        {
            Console.Write($"{i,2}|");
            for (int j = 0; j < n; j++)
            {
                Console.Write($"{g[i, j],3}");
            }
            Console.WriteLine();
        }

        visited = new bool[6];
        Console.WriteLine("DFS обход графа (начинаем с вершины 0):");
        DFS(0);
        Console.WriteLine();

        visited = new bool[6];
        Console.WriteLine("BFS обход графа (начинаем с вершины 0):");
        BFS(0);
        Console.WriteLine();
    }

    public static void DFS(int vertex)
    {
        visited[vertex] = true;
        Console.Write(vertex + " ");

        for (int i = 0; i < 6; i++)
        {
            if (g[vertex, i] == 1 && !visited[i])
            {
                DFS(i);
            }
        }
    }
```

```
    public static void BFS(int startVertex)
    {
        Queue<int> queue = new Queue<int>();
        visited[startVertex] = true;
        queue.Enqueue(startVertex);

        while (queue.Count > 0)
        {
            int vertex = queue.Dequeue();
            Console.Write(vertex + " ");

            for (int i = 0; i < 6; i++)
            {
                if (g[vertex, i] == 1 && !visited[i])
                {
                    visited[i] = true;
                    queue.Enqueue(i);
                }
            }
        }
    }
}
```

```
    0  1  2  3  4  5
  ------------------
0|  0  0  0  1  1  0
1|  0  0  0  0  0  1
2|  0  0  0  0  1  1
3|  1  0  0  0  0  0
4|  1  0  0  0  1  0
5|  0  1  1  0  0  0
DFS обход графа (начинаем с вершины 0):
0 3 4
BFS обход графа (начинаем с вершины 0):
0 3 4
```

Задание 2

Реализовать алгоритм нахождения кратчайших путей (Алгоритм Дейкстры). Выполнить расчет примера.

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

class Program
{
    static int n = 6;
    static int[,] g = new int[n, n];

    static void Main()
    {
        g[0, 1] = g[1, 0] = 2;
        g[0, 2] = g[2, 0] = 1;
        g[1, 3] = g[3, 1] = 5;
        g[1, 4] = g[4, 1] = 3;
        g[2, 5] = g[5, 2] = 4;
        g[3, 4] = g[4, 3] = 2;

        Dijkstra(g, 0);
    }

    public static void Dijkstra(int[,] g, int start)
    {
        int[] distances = new int[n];
        bool[] visited = new bool[n];
        int[] previous = new int[n];

        for (int i = 0; i < n; i++)
        {
            distances[i] = 100000;
            visited[i] = false;
            previous[i] = -1;
        }

        distances[start] = 0;

        for (int count = 0; count < n - 1; count++)
        {
            int minVertex = -1;
            int minDistance = 100000;

            for (int i = 0; i < n; i++)
            {
                if (!visited[i] && distances[i] <= minDistance)
                {
                    minDistance = distances[i];
                    minVertex = i;
                }
            }

            if (minVertex == -1) break;

            visited[minVertex] = true;

            for (int i = 0; i < n; i++)
            {
                int edgeWeight = g[minVertex, i];
                if (!visited[i] && edgeWeight != 0 && distances[minVertex] !=
100000)
                {
```

```
                    int newDistance = distances[minVertex] + edgeWeight;
                    if (newDistance < distances[i])
                    {
                        distances[i] = newDistance;
                        previous[i] = minVertex;
                    }
                }
            }
        }

        Console.WriteLine($"Кратчайшие расстояния от вершины {start}:");
        for (int i = 0; i < n; i++)
        {
            List<int> path = new List<int>();
            for (int a = i; a != -1; a = previous[a])
                path.Add(a);
            path.Reverse();

            Console.Write($"До вершины {i}: расстояние = {distances[i]},
путь: ");
            Console.WriteLine(string.Join(" - ", path));
        }
    }
}
```

```
Кратчайшие расстояния от вершины 0:
До вершины 0: расстояние = 0, путь: 0
До вершины 1: расстояние = 2, путь: 0 - 1
До вершины 2: расстояние = 1, путь: 0 - 2
До вершины 3: расстояние = 7, путь: 0 - 1 - 3
До вершины 4: расстояние = 5, путь: 0 - 1 - 4
До вершины 5: расстояние = 5, путь: 0 - 2 - 5
```

Задание 3

Найти минимальный остов неориентированного взвешенного графа (реализовать алгоритм Крускала).

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

public class Edge
{
    public int x { get; set; }
    public int y { get; set; }
    public int weight { get; set; }

    public Edge(int x, int y, int weight)
    {
        this.x = x;
        this.y = y;
        this.weight = weight;
    }
}

class Program
{
    static int[] parent;

    static void Main()
    {
        List<Edge> edges = new List<Edge>()
        {
            new Edge(0, 4, 7),
            new Edge(0, 2, 1),
            new Edge(1, 5, 1),
            new Edge(1, 3, 2),
            new Edge(1, 2, 6),
            new Edge(2, 3, 2),
            new Edge(2, 5, 5)
        };

        edges = edges.OrderBy(e => e.weight).ToList();
        Console.WriteLine("Отсортированные рёбра:");
        foreach (var e in edges)
            Console.WriteLine($"{e.x}-{e.y} ({e.weight})");

        parent = new int[6];
        for (int i = 0; i < 6; i++)
            parent[i] = i;

        List<Edge> mst = new List<Edge>();
        int totalWeight = 0;

        foreach (Edge edge in edges)
        {
            int rootX = Find(edge.x);
            int rootY = Find(edge.y);

            if (rootX != rootY)
            {
                mst.Add(edge);
                totalWeight += edge.weight;
                parent[rootY] = rootX;

                if (mst.Count == 5) break;
            }
```

```
        }

        Console.WriteLine("\nМинимальное остовное дерево:");
        foreach (var edge in mst)
            Console.WriteLine($"{edge.x}-{edge.y} (вес: {edge.weight})");

        Console.WriteLine($"Общий вес: {totalWeight}");
    }

    static int Find(int x)
    {
        if (parent[x] != x)
            parent[x] = Find(parent[x]);
        return parent[x];
    }
}
```

```
Отсортированные рёбра:
0-2 (1)
1-5 (1)
1-3 (2)
2-3 (2)
2-5 (5)
1-2 (6)
0-4 (7)

Минимальное остовное дерево:
0-2 (вес: 1)
1-5 (вес: 1)
1-3 (вес: 2)
2-3 (вес: 2)
0-4 (вес: 7)
Общий вес: 13
```

Заключение

В ходе выполнения лабораторной работы были изучены и реализованы на языке C# основные алгоритмы на графах – алгоритм поиска в ширину и в глубину, алгоритм Дейкстры и алгоритм Крускала.