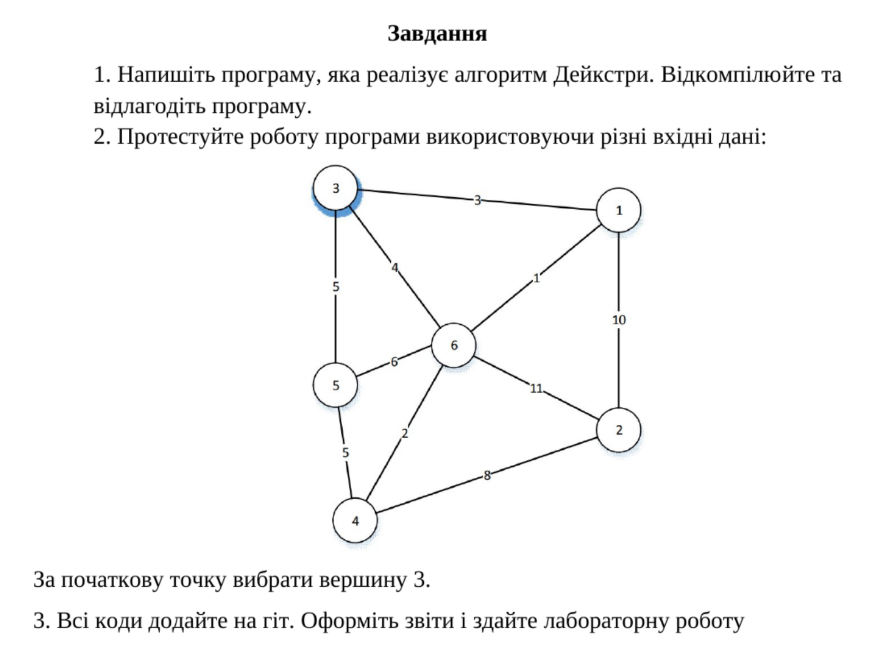
**ЗВІТ**

про виконання лабораторної роботи 3

«**Реалізація типових алгоритмів**»  
з дисципліни  
«Методика розв'язування олімпіадних задач» студента(ки) групи ІН-2327Б  
Чухрая Олександра Васильовича

**Умова завдання:**

****

**Виконання:**

using System;

using System.Collections.Generic;

class DijkstraAlgorithm

{

class Edge

{

public int To { get; set; }

public int Weight { get; set; }

public Edge(int to, int weight)

{

To = to;

Weight = weight;

}

}

class Graph

{

private readonly Dictionary<int, List<Edge>> adjacencyList = new Dictionary<int, List<Edge>>();

public void AddEdge(int from, int to, int weight)

{

if (!adjacencyList.ContainsKey(from))

adjacencyList[from] = new List<Edge>();

if (!adjacencyList.ContainsKey(to))

adjacencyList[to] = new List<Edge>();

adjacencyList[from].Add(new Edge(to, weight));

adjacencyList[to].Add(new Edge(from, weight)); // Since it's an undirected graph

}

public void Dijkstra(int start)

{

var distances = new Dictionary<int, int>();

var priorityQueue = new SortedSet<(int distance, int node)>();

var visited = new HashSet<int>();

foreach (var node in adjacencyList.Keys)

distances[node] = int.MaxValue;

distances[start] = 0;

priorityQueue.Add((0, start));

while (priorityQueue.Count > 0)

{

var minElement = priorityQueue.Min;

priorityQueue.Remove(minElement);

int currentDistance = minElement.distance;

int currentNode = minElement.node;

if (visited.Contains(currentNode)) continue;

visited.Add(currentNode);

foreach (var edge in adjacencyList[currentNode])

{

int newDist = currentDistance + edge.Weight;

if (newDist < distances[edge.To])

{

distances[edge.To] = newDist;

priorityQueue.Add((newDist, edge.To));

}

}

}

Console.WriteLine($"Shortest distances from node {start}:");

foreach (var kvp in distances)

Console.WriteLine($"To {kvp.Key} -> {kvp.Value}");

}

}

static void Main()

{

Graph graph = new Graph();

// Adding edges to the graph

graph.AddEdge(3, 1, 3);

graph.AddEdge(3, 6, 4);

graph.AddEdge(3, 5, 5);

graph.AddEdge(6, 1, 1);

graph.AddEdge(6, 2, 11);

graph.AddEdge(6, 4, 2);

graph.AddEdge(6, 5, 6);

graph.AddEdge(5, 4, 5);

graph.AddEdge(4, 2, 8);

graph.AddEdge(1, 2, 10);

graph.Dijkstra(3); // Running Dijkstra's algorithm from node 3

Console.ReadKey();

}

}

Результат:

