Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №3

Метод решения задач о наидлиннейшем (s,t) – пути в направленном графе без контуров

Выполнил:

студент гр. 953504

Кондрашов И.Д.

Проверил:

кандидат физико-математических наук

Дугинов О.И.

Минск 2022

1. **Пример работы программы**

Входные данные:

A screenshot of a computer

Description automatically generated with medium confidence

Ответ:

Graphical user interface, text

Description automatically generated

1. **Исходный код программы (C#)**

Класс GraphNode.cs – класс, представляющий один узел графа

public class GraphNode

{

public int Source { get; set; }

public int Destination { get; set; }

public int Weight { get; set; }

public string Color { get; set; }

public GraphNode(int source, string color, int dest = 0, int weight = 0)

{

Source = source;

Destination = dest;

Weight = weight;

Color = color;

}

}

Класс Program.cs – основной класс программы, где реализованы методы

1. Main(string[] args) - точка выхода в программу, здесь задаются входные данные и вызываются нужные функции для их обработки
2. DFS(GraphNode node) – метод, реализующий поиск в глубину
3. STPath(int source, int target, List<GraphNode> graphNodes) – метод, выполняющий задание лабораторной(поиск (s,t) - пути)

class Program

{

private static Stack<GraphNode> stack = new Stack<GraphNode>();

private static List<GraphNode> graph = new List<GraphNode>();

static void Main(string[] args)

{

int start = 0;

int finish = 5;

graph.Add(new GraphNode(0, "gray", 1, 1));

graph.Add(new GraphNode(0, "white", 2, 2));

graph.Add(new GraphNode(1, "white", 2, 2));

graph.Add(new GraphNode(1, "white", 4, 1));

graph.Add(new GraphNode(2, "white", 3, 1));

graph.Add(new GraphNode(3, "white", 4, 2));

graph.Add(new GraphNode(3, "white", 5, 1));

graph.Add(new GraphNode(4, "white", 5, 2));

graph.Add(new GraphNode(5, "white"));

var (opt, x) = STPath(start, finish, graph);

foreach (var item in opt)

{

Console.WriteLine($"Элемент массива opt: {item}");

}

foreach (var item in x)

{

Console.WriteLine($"Элемент массива x: {item}");

}

}

private static void DFS(GraphNode node)

{

node.Color = "gray";

foreach (var item in graph)

{

if (item.Color == "white")

{

DFS(item);

}

}

stack.Push(node);

}

private static Tuple<double[], double?[]> STPath(int source, int target, List<GraphNode> graphNodes)

{

try

{

DFS(graphNodes.ElementAt(source));

stack = new Stack<GraphNode>(stack.Reverse().ToList());

int startIndex = stack.ToList().FindIndex(x => x.Source == source);

int finishIndex = stack.ToList().FindIndex(x => x.Destination == target);

if (startIndex > finishIndex)

{

throw new Exception("Нет пути!");

}

var sortedSources = new SortedSet<int>(stack.Select(x => x.Source));

double[] opt = new double[sortedSources.Count];

Array.Fill(opt, Double.NegativeInfinity);

double?[] x = new double?[sortedSources.Count];

Array.Fill(x, 0);

opt[source] = 0;

x[source] = null;

sortedSources.ToList().ForEach((sor) =>

{

stack.ToList()

.ForEach((s) =>

{

if (s.Destination == sor)

{

var cur = sortedSources.ElementAt(s.Source);

if (s.Weight + opt[cur] > opt[sor])

{

opt[sor] = s.Weight + opt[cur];

x[sor] = s.Source;

}

}

});

});

return Tuple.Create(opt, x);

}

catch (Exception ex)

{

Console.WriteLine($"Произошла ошибка: {ex.Message}");

return Tuple.Create(new double[1], new double?[1]);

}

}

}