**Описание**

С помощью алгоритма Дейкстры вычисляется кратчайший путь между станциями. Матрица смежности, а также список станций считывается из файла.

**Листинг программы**

using metro.Properties;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Diagnostics.CodeAnalysis;

namespace metro

{

class Program

{

private class Node //объявление класса Узел

{

public string Name;

public Dictionary<Node, int> To = new Dictionary<Node, int>();

public override string ToString() //переопределение метода Имя

{

return Name; //возвращается значение типа void

}

}

private class Path //объявление класса Путь

{

public Node Prev;

public Node To;

public int Weight;

}

private class ByWeight : IComparer<Path>

{

public int Compare(Path x, Path y) //метод, который сравнивает два значения

{

var result = x.Weight.CompareTo(y.Weight);

if (result == 0)

result = x.To.Name.CompareTo(y.To.Name);

return result;

}

}

private static Node GetNodeByName(Dictionary<string, Node> nodesByName, string name)

{

if (!nodesByName.TryGetValue(name, out Node res))

nodesByName.Add(name, res = new Node() { Name = name });

return res;

}

private static List<Path> Dijkstra(Node from, Node to) // метод Дейкстры

{

var initialPath = new Path() { Weight = 0, Prev = null, To = from }; // пременная, показывающая первоначальный путь

var paths = new SortedSet<Path>(new ByWeight()) { initialPath}; // переменная, показывающая путь

var shortestPaths = new Dictionary<Node, Path>() { { from, initialPath } }; // перменная, показывающая кратчайший путь

while(paths.Count > 0) //

{

var path = paths.Min;

var node = path.To;

paths.Remove(path);

if (node == to)

break;

foreach(var edge in node.To)

{

var next = edge.Key;

var weight = path.Weight + edge.Value;

Path oldPath;

if (!shortestPaths.TryGetValue(next,out oldPath))

oldPath = null;

if(oldPath != null && oldPath.Weight > weight)

{

paths.Remove(path);

shortestPaths.Remove(next);

oldPath = null;

};

if(oldPath == null)

{

var newPath = new Path { Weight = weight, Prev = node, To = next };

shortestPaths.Add(next, newPath);

paths.Add(newPath);

}

}

}

var result = new List<Path>();

while(to != from && to != null )

{

var path = shortestPaths[to];

result.Add(path);

to = path.Prev;

}

result.Reverse();

return result;

}

static void Main(string[] args)

{

var nodesByNames = new Dictionary<string, Node>();

var edgesStrings = Resources.Edges.Split('\n');

foreach(var edgeString in edgesStrings)

{

if (string.IsNullOrEmpty(edgeString))

continue;

var NodesAndWeight = edgeString.Trim().Replace("\r", "").Split(" : "); //возвращается новая строка, в которой удалены все вхождения заданного набора символов из текущей строки

var weight = int.Parse(NodesAndWeight[1]); //преобразуется строковое представленеи числа в эквивалентное ему целое число со знаком

var nodeNames = NodesAndWeight[0].Split(" <-> "); //строка делится " <-> " на части между символами

bool undirected = nodeNames.Length > 1;

if(!undirected)

nodeNames = NodesAndWeight[0].Split(" -> ");

var node1 = GetNodeByName(nodesByNames, nodeNames[0]);

var node2 = GetNodeByName(nodesByNames, nodeNames[1]);

node1.To.Add(node2, weight);

if (undirected)

{

node2.To.Add(node1, weight);

}

}

bool onceAgain;

do

{

Node from = null; //обнуление переменой

Node to = null; //обнуление переменной

List<Node> possibleFrom = new List<Node>(); //переменная класса

List<Node> possibleTo = new List<Node>();

Console.WriteLine("Введите станцию отправления:"); //вывод на экран "Введите станцию отправления:"

var name1 = Console.ReadLine();

if (!nodesByNames.TryGetValue(name1, out from))

{

foreach(var node in nodesByNames.Values)

{

if (node.Name.Contains(name1))

possibleFrom.Add(node);

}

if (possibleFrom.Count > 10)

possibleFrom.Clear();

if(possibleFrom.Count == 0 )

Console.WriteLine("Станция отправления не распознана"); //вывод на экран "Станция отправления не распознана"

}

if (from != null || possibleFrom.Count > 0)

{

Console.WriteLine("Введите станцию назначения:"); //вывод на экран "Введите станцию назначения: "

var name2 = Console.ReadLine();

if (!nodesByNames.TryGetValue(name2, out to))

{

foreach (var node in nodesByNames.Values)

{

if (node.Name.Contains(name2))

possibleTo.Add(node);

}

if (possibleTo.Count > 10)

possibleTo.Clear();

if (possibleTo.Count == 0)

Console.WriteLine("Станция назначения не распознана"); //вывод на экран "Станция назначения не распознана"

}

if (to != null || possibleTo.Count > 0)

{

if (from != null)

possibleFrom = new List<Node> { from };

if (to != null)

possibleTo = new List<Node> { to };

List<Path> shortest = null;

if (from == null || to == null)

{

var minWeight = int.MaxValue;

foreach(var f in possibleFrom)

foreach(var t in possibleTo)

{

var path = Dijkstra(f, t);

var w = path[path.Count - 1].Weight;

if (w < minWeight)

{

minWeight = w;

from = f;

to = t;

shortest = path;

}

}

}

{

if(shortest == null) //если равно 0

shortest = Dijkstra(from, to);

foreach (var p in shortest) // перебор элементов в коллекции

{

var node1 = p.Prev;

var node2 = p.To;

int edgeWeight;

if (node1 != null && node1.To.TryGetValue(node2, out edgeWeight))

Console.WriteLine($"{node1.Name} -> {node2.Name} : {edgeWeight}");

}

var weight = shortest.Count > 0 ? shortest[shortest.Count - 1].Weight : 0;

Console.WriteLine($"Общая длина пути {weight}"); //вывод на экран значения общей длины пути

}

}

}

Console.WriteLine("Ещё раз, да?"); //повторить

onceAgain = Console.ReadLine().ToLower().Trim() == "да";

} while (onceAgain);

}

}

}

**Edges.txt**

Сенная Площадь <-> Невский проспект : 2

Невский проспект <-> Горьковская : 4

Горьковская <-> Петроградская : 2

Петроградская <-> Черная речка : 4

Черная речка <-> Пионерская : 3

Пионерская <-> Удельная : 3

Удельная <-> Озерки : 3

Озерки <-> Проспект Просвящения : 3

Проспект Просвящения <-> Парнас : 3

Сенная Площадь <-> Технологический Институт-1.2 : 2

Технологический Институт-1.2 <-> Фрунзенская : 2

Пушкинская <-> Технологический Институт-1.1 : 2

Технологический Институт-1.1 <-> Балтийская : 2

Фрунзенская <-> Технологический Институт-2.2 : 2

Технологический Институт-2.2 <-> Сенная Площадь : 2

Сенная Площадь <-> Технологический институт 2.1 : 4

Технологический институт 2.1 <-> Фрунзенская : 4

Фрунзенская <-> Московские ворота : 3

Московские ворота <-> Электросила : 2

Электросила <-> Парк Победы : 2

Парк Победы <-> Московская : 3

Московская <-> Звездная : 4

Звездная <-> Купчино : 3

Балтийская <-> Технологический Институт-2.1 : 2

Технологический Институт-2.1 <-> Пушкинская : 2

Технологический Институт-2.1 <-> Технологический Институт-2.2 : 2

Технологический Институт-1.1 <-> Технологический Институт-1.2 : 2

Технологический Институт-1.1 <-> Технологический Институт-2.1 : 4

Технологический Институт-1.1 <-> Технологический Институт-2.2 : 4

Технологический Институт-1.2 <-> Технологический Институт-2.1 : 4

Технологический Институт-1.2 <-> Технологический Институт-2.2 : 4

Пушкинская <-> Владимирская : 2

Звенигородская <-> Пушкинская : 3

Звенигородская <-> Садовая : 4

Садовая <-> Адмиралтейская : 4

Адмиралтейская <-> Спортивная : 4

Спортивная <-> Чкаловская : 2

Чкаловская <-> Крестовский остров : 3

Крестовский остров <-> Старая деревня : 2

Старая деревня <-> Комендантский проспект : 3

Звенигородская <-> Обводный канал : 3

Обводный канал <-> Волковская : 3

Волковская <-> Бухарестская : 3

Бухарестская <-> Международная : 3

Международная <-> Проспект Славы : 3

Проспект Славы <-> Дунайская : 3

Дунайская <-> Шушары : 3

Садовая <-> Спасская : 3

Спасская <-> Достоевская : 3

Достоевская <-> Лиговский проспект : 2

Лиговский проспект <-> Площадь Александра Невского-2 : 3

Площадь Александра Невского-2 <-> Площадь Александра Невского-1 : 3

Площадь Александра Невского-2 <-> Новочеркасская : 3

Новочеркасская <-> Ладожская : 2

Ладожская <-> Проспект Большевиков : 3

Проспект Большевиков <-> Улица Дыбенко : 2

Садовая <-> Сенная Площадь : 4

Гостиный двор <-> Невский проспект : 4

Достоевская <-> Владимирская : 3

Площадь Восстания <-> Маяковская : 3

Спасская <-> Сенная Площадь : 4

Площадь Александра Невского-1 <-> Елизаровская : 5

Елизаровская <-> Ломоносовская : 3

Ломоносовская <-> Пролетарская : 3

Пролетарская <-> Обухово : 3

Обухово <-> Рыбацкое : 4

Площадь Александра Невского-1 <-> Маяковская : 5

Маяковская <-> Гостиный двор : 2

Гостиный двор <-> Василеостровская : 5

Василеостровская <-> Приморская : 4

Приморская <-> Беговая : 5

Пушкинская <-> Технологический институт 1.1 : 4

Технологический институт 1.1 <-> Балтийская : 2

Балтийская <-> Нарвская : 2

Нарвская <-> Кировский завод : 3

Кировский завод <-> Автово : 2

Автово <-> Ленинский проспект : 2

Ленинский проспект <-> Проспект Ветеранов : 2

Владимирская <-> Площадь Восстания : 2

Площадь Восстания <-> Чернышевская : 3

Чернышевская <-> Площадь Ленина : 2

Площадь Ленина <-> Выборгская : 2

Выборгская <-> Лесная : 2

Лесная <-> Площадь Мужества : 3

Площадь Мужества <-> Политехническая : 2

Политехническая <-> Академическая : 2

Академическая <-> Гражданский проспект : 4

Гражданский проспект <-> Девяткино : 3

**Результат работы программы**

На рисунке 1 представлены результаты работы программы.

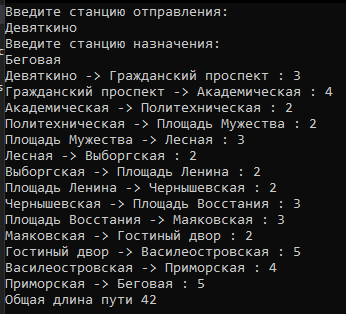


Рисунок 1 – Результат работы программы

**Вывод**

В ходе выполнения лабораторной работы были изучены теоретические материалы по теме теория графов. Также была разработана программа, написанная на языке C#, которая позволяет искать кратчайший путь между станциями А и Б Санкт-Петербургского метрополитена. Программа учитывает время пересадки, а также считает время пути.