Bài 1

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buoi7\_bai1

{

internal class Dijkstra

{

public const double INF = 1e9;

private LinkedList<Tuple<int, int, int>> edgeList;

private int e;

private int v;

private int start;

private int end;

private int[] pre;

private int[] dist;

private PriorityQueue<int, int> queue;

public Dijkstra(string path)

{

if (!File.Exists(path))

{

throw new FileNotFoundException(path);

}

string[] line = File.ReadAllLines(path);

string[] args = line[0].Split();

e = int.Parse(args[0]);

v = int.Parse(args[1]);

start = int.Parse(args[2]);

end = int.Parse(args[3]);

edgeList = new LinkedList<Tuple<int, int, int>>();

for(int i=1;i<=v; i++)

{

string[] linenumber = line[i].Split();

int \_u=int.Parse(linenumber[0]);

int \_v=int.Parse(linenumber[1]);

int \_w=int.Parse(linenumber[2]);

edgeList.AddLast(new Tuple<int,int, int>(\_u, \_v, \_w));

edgeList.AddLast(new Tuple<int,int, int>(\_v,\_u,\_w));

}

dist=new int[e];

pre=new int[e];

queue = new PriorityQueue<int, int>();

for (int i = 0; i < e; i++)

{

dist[i] = (int)INF;

}

}

public void Solve(string outpath)

{

dist[start - 1] = 0;

queue.Enqueue(start, 0);

while(queue.Count > 0)

{

int i = queue.Dequeue();

foreach(Tuple<int, int, int> edge in edgeList)

{

if(edge.Item1 != i)

{

continue;

}

if (dist[edge.Item2 - 1] > dist[i - 1] + edge.Item3)

{

dist[edge.Item2 - 1] = dist[i - 1] + edge.Item3;

pre[edge.Item2 - 1] = i;

queue.Enqueue(edge.Item2, dist[edge.Item2-1]);

}

}

}

string[] data =

{

$"{dist[end-1]}",

$"{GT.ExtractPath(pre,end)}"

};

File.WriteAllLines(outpath, data);

}

}

}

namespace Buoi7\_bai1

{

internal class Program

{

static void Main(string[] args)

{

Dijkstra dijkstra = new Dijkstra("D:\\Code\\CSharp\\Graph-Theory\\B7\\Data\\Ex1\\Disjstra.INP");

dijkstra.Solve("D:\\Code\\CSharp\\Graph-Theory\\B7\\Data\\Ex1\\Disjstra.OUT");

}

}

}

Bài 2:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buoi7\_bai2

{

internal class shortdist

{

private int INF = 9999;

private int v;

private int e;

private int s;

private int t;

private int a;

private LinkedList<Tuple<int, int, int>> edgeList;

private int[,] dist;

private int[,] truoc;

public shortdist(string path)

{

if (!File.Exists(path))

{

throw new FileNotFoundException(path);

}

string[] lines = File.ReadAllLines(path);

string[] args = lines[0].Split();

v = int.Parse(args[0]);

e = int.Parse(args[1]);

s = int.Parse(args[2]);

t = int.Parse(args[3]);

a = int.Parse(args[4]);

edgeList = new LinkedList<Tuple<int, int, int>>();

for (int i = 1; i <= e; i++)

{

string[] numbers = lines[i].Split();

edgeList.AddLast(new Tuple<int, int, int>(int.Parse(numbers[0]), int.Parse(numbers[1]), int.Parse(numbers[2])));

}

dist = new int[v, v];

truoc = new int[v, v];

for (int i = 0; i < v; i++)

{

for (int j = 0; j < v; j++)

{

dist[i, j] = INF;

}

}

foreach (Tuple<int, int, int> edge in edgeList)

{

// Đồ thị vô hướng

dist[edge.Item1 - 1, edge.Item2 - 1] = edge.Item3;

dist[edge.Item2 - 1, edge.Item1 - 1] = edge.Item3;

}

}

private string ExtractPath(int s, int t)

{

if (truoc[s - 1, t - 1] == 0)

{

return $" {t}";

}

return ExtractPath(s, truoc[s - 1, t - 1]) + $" {t}";

}

public void Solve(string outputpath)

{

// for more details

for (int y = 0; y < v; y++)

{

for (int x = 0; x < v; x++)

{

if (dist[x, y] != INF)

{

for (int j = 0; j < v; j++)

{

if (dist[y, j] != INF)

{

if (dist[x, j] == INF || (dist[x, j] > dist[x, y] + dist[y, j]))

{

dist[x, j] = dist[x, y] + dist[y, j];

truoc[x, j] = y + 1;

}

}

}

}

}

}

int sDist = dist[s - 1, a - 1] + dist[a - 1, t - 1];

string sPath = $"{s}" + ExtractPath(s, a) + ExtractPath(a, t);

string[] data =

{sDist.ToString(),sPath};

File.WriteAllLines(outputpath, data);

}

}

}

namespace Buoi7\_bai2

{

internal class Program

{

static void Main(string[] args)

{

shortdist shortdist = new shortdist("D:\\Code\\CSharp\\Graph-Theory\\B7\\Data\\Ex1\\Short.INP");

shortdist.Solve("D:\\Code\\CSharp\\Graph-Theory\\B7\\Data\\Ex1\\Short.OUT");

}

}

}