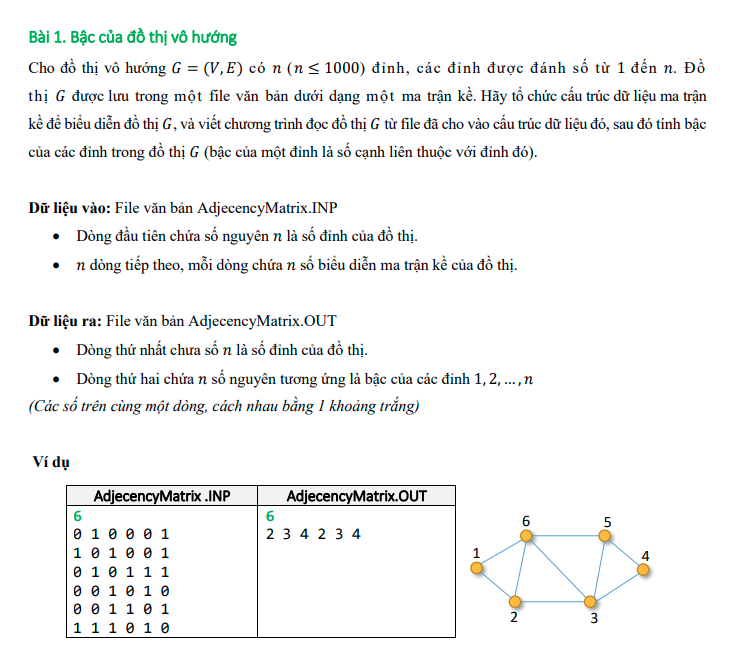
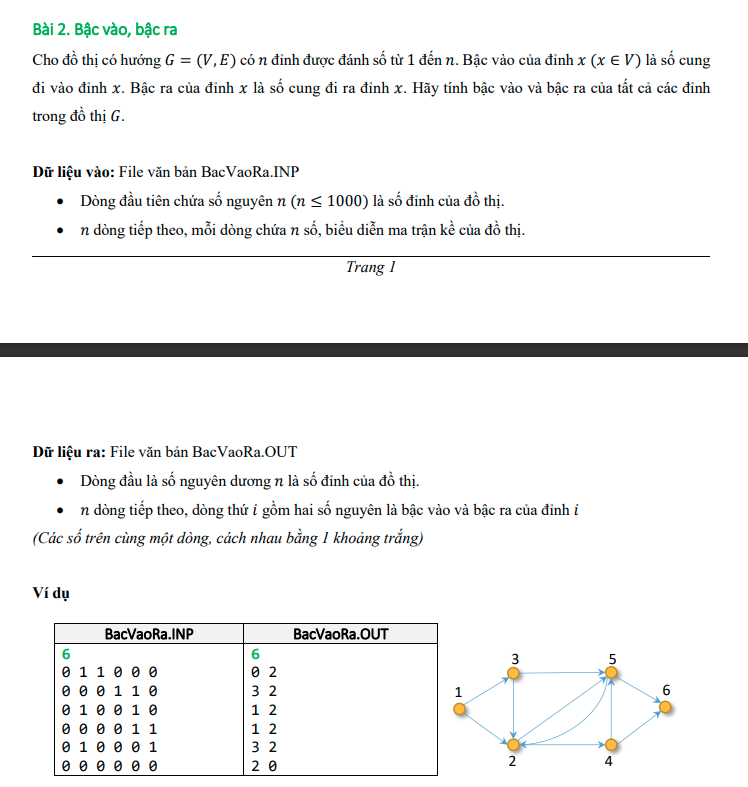
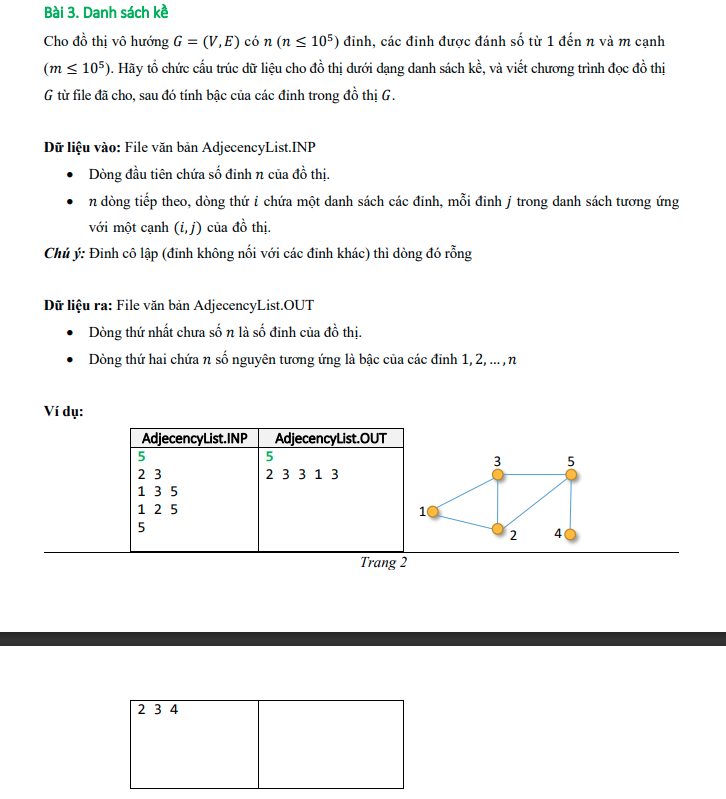
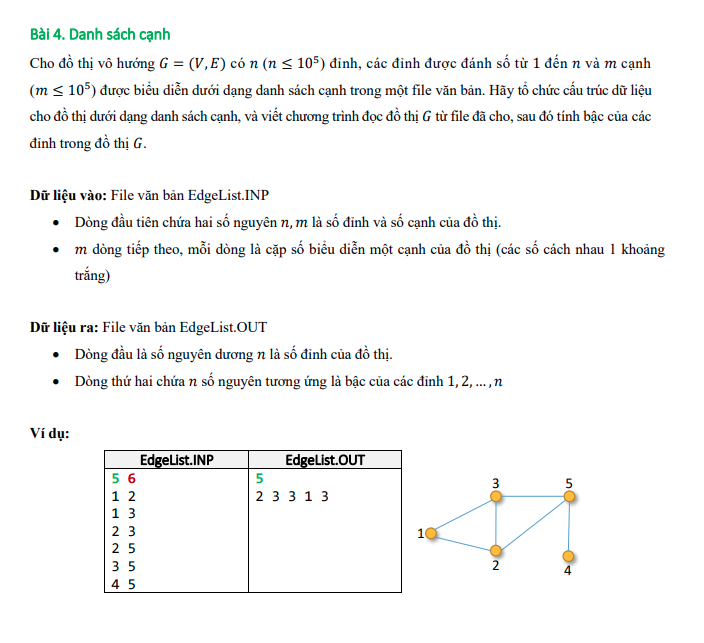
Tuần 1: Nhập, xuất và thao tác cơ bản trên đồ thị









CODE:

AdjList.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buoi1

{

internal class AdjList

{

LinkedList<int>[] v;

int n;

public LinkedList<int>[] V { get => v; set => v = value; }

public int N { get => n; set => n = value; }

public AdjList() { }

public AdjList(int k)

{

v = new LinkedList<int>[k];

}

public AdjList(LinkedList<int>[] list)

{

v = list;

}

public void FileToAdjList(string filePath)

{

StreamReader sr = new StreamReader(filePath);

n = int.Parse(sr.ReadLine());

v = new LinkedList<int>[n];

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

{

v[i] = new LinkedList<int>();

string st = sr.ReadLine();

if (st != "")

{

string[] s = st.Split();

for (int j = 0; j < s.Length; j++)

{

int x = int.Parse(s[j]);

v[i].AddLast(x);

}

}

}

sr.Close();

}

public void Output()

{

Console.WriteLine("Do thi danh sach ke - so dinh: " + n);

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

{

Console.Write(" Dinh {0} ->", i);

foreach (int x in v[i])

{

Console.Write("{0, 3}", x);

}

Console.WriteLine();

}

}

public void DegV(string filePath)

{

StreamWriter sw = new StreamWriter(filePath);

Console.WriteLine("Bậc của các đỉnh là: ");

sw.WriteLine("Bậc của các đỉnh là: ");

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

{

Console.Write("{0}: {1}", i, v[i].Count);

sw.Write("{0}: {1}", i, v[i].Count);

sw.WriteLine();

Console.WriteLine();

}

sw.Close();

}

}

}

AdjMatrix.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

namespace Buoi1

{

internal class AdjMatrix

{

private int n;

private int[,] a;

public int N { get => n; set => n = value; }

public int[,] A { get => a; set => a = value; }

public AdjMatrix() { }

public AdjMatrix(int k)

{

n = k;

a = new int[n, n];

}

public void FileToAdjMatrix(string filePath)

{

StreamReader sr = new StreamReader(filePath);

n = int.Parse(sr.ReadLine());

a = new int[n, n];

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

{

string[] s = sr.ReadLine().Split();

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

{

a[i, j] = int.Parse(s[j]);

}

}

sr.Close();

}

public void Output()

{

Console.WriteLine("Do thi ma tran ke - So dinh: " + n);

Console.WriteLine();

Console.Write("Dinh |");

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

{

Console.Write(" {0} ", i);

}

Console.WriteLine();

Console.WriteLine(" " + new string('-', 6 \* n));

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

{

Console.Write(" {0}|", i);

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

{

Console.Write(" {0,3}", a[i, j]);

}

Console.WriteLine();

}

}

public int DegVi(int level)

{

int count = 0;

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

{

if (a[level, i] == 1)

{

count += 1;

}

}

return count;

}

public void DegVi(string filePath)

{

StreamWriter sw = new StreamWriter(filePath);

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

{

int result = DegVi(i);

Console.Write("{0}: {1}", i, result);

sw.Write("{0}: {1}", i, result);

sw.WriteLine();

Console.WriteLine();

}

sw.Close();

}

public int DegOut(int i) // Bậc ra của 1 đỉnh là hàng ngang của ma trận tại chỉ số đỉnh đó

{

int count = 0;

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

{

if (a[i, j] == 1)

{

count += 1;

}

}

return count;

}

public int DegIn(int j) // Bậc vào của 1 đỉnh là hàng dọc của ma trận tại chỉ số đỉnh đó

{

int count = 0;

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

{

if (a[i, j] == 1)

{

count += 1;

}

}

return count;

}

public void DegInOut(string filePath)

{

StreamWriter sw = new StreamWriter(filePath);

Console.WriteLine(" Bậc vào, bậc ra của các đỉnh: ");

sw.WriteLine(" Bậc vào, bậc ra của các đỉnh: ");

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

{

Console.WriteLine("{0}: {1} - {2} ", i, DegIn(i), DegOut(i));

sw.WriteLine("{0}: {1} - {2} ", i, DegIn(i), DegOut(i));

}

sw.Close();

Console.WriteLine();

}

}

}

EdgeList.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buoi1

{

internal class EdgeList

{

LinkedList<Tuple<int, int>> edges;

int n;

int m;

public int N { get => n; set => n = value; }

public int M { get => m; set => m = value; }

public LinkedList<Tuple<int, int>> Edges { get => edges; set => edges = value; }

public EdgeList()

{

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

}

public void FileToEdgeList(string filePath)

{

StreamReader sr = new StreamReader(filePath);

string[] s = sr.ReadLine().Split();

n = int.Parse(s[0]);

m = int.Parse(s[1]);

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

{

s = sr.ReadLine().Split();

Tuple<int, int> e = new Tuple<int, int>(int.Parse(s[0]), int.Parse(s[1]));

edges.AddLast(e);

}

sr.Close();

}

public void Output()

{

Console.WriteLine("Danh sách cạnh của đồ thị với số đỉnh n = " + n);

foreach (Tuple<int, int> e in edges)

{

Console.WriteLine(" (" + e.Item1 + "," + e.Item2 + ")");

}

}

public void DegV(string filePath)

{

int count = 0;

StreamWriter sw = new StreamWriter(filePath);

Console.WriteLine("Bậc của các đỉnh là: ");

sw.WriteLine("Bậc của các đỉnh là: ");

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

{

foreach (Tuple<int, int> e in edges)

{

if (e.Item1 == i || e.Item2 == i)

{

count++;

}

}

Console.WriteLine("{0}: {1}", i, count);

sw.WriteLine("{0}: {1}", i, count);

count = 0;

}

sw.Close();

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buoi1

{

internal class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.Unicode;

Console.InputEncoding = Encoding.Unicode;

/\* Tạo menu \*/

Menu menu = new Menu();

string title = "NHẬP, XUẤT & CÁC THAO TÁC CƠ BẢN TRÊN ĐỒ THỊ";

// Danh sách các mục chọn

string[] ms = { "1. Bài 1: Đồ thị vô hướng ma trận kề, tính bậc của của các đỉnh",

"2. Bài 2: Đồ thị có hướng ma trận kề, bậc vào, tính bậc ra của các đỉnh",

"3. Bài 3: Đồ thị danh sách kề, tính bậc các đỉnh",

"4. Bài 4: Đồ thị danh sách cạnh, tính bậc của các đỉnh",

"0. Thoát" };

int choice;

do

{

menu.ShowMenu(title, ms);

Console.Write(" Chọn: ");

choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

{

AdjMatrix matrix = new AdjMatrix();

[matrix.FileToAdjMatrix(@"..\..\TextFile\AdjMatrix.txt](mailto:matrix.FileToAdjMatrix(@"..\..\TextFile\AdjMatrix.txt)");

matrix.Output();

[matrix.DegVi(@"..\..\TextFile\AdjMatrixOut.txt](mailto:matrix.DegVi(@"..\..\TextFile\AdjMatrixOut.txt)");

break;

}

case 2:

{

AdjMatrix matrix = new AdjMatrix();

[matrix.FileToAdjMatrix(@"..\..\TextFile\DirectedMatrix.txt](mailto:matrix.FileToAdjMatrix(@"..\..\TextFile\DirectedMatrix.txt)");

matrix.Output();

[matrix.DegInOut(@"..\..\TextFile\DirectedMatrixOut.txt](mailto:matrix.DegInOut(@"..\..\TextFile\DirectedMatrixOut.txt)");

break;

}

case 3:

{

AdjList matrix = new AdjList();

[matrix.FileToAdjList(@"..\..\TextFile\AdjList.txt](mailto:matrix.FileToAdjList(@"..\..\TextFile\AdjList.txt)");

matrix.Output();

[matrix.DegV(@"..\..\TextFile\AdjListOut.txt](mailto:matrix.DegV(@"..\..\TextFile\AdjListOut.txt)");

break;

}

case 4:

{

EdgeList matrix = new EdgeList();

[matrix.FileToEdgeList(@"..\..\TextFile\EdgeList.txt](mailto:matrix.FileToEdgeList(@"..\..\TextFile\EdgeList.txt)");

matrix.Output();

[matrix.DegV(@"..\..\TextFile\EdgeListOut.txt](mailto:matrix.DegV(@"..\..\TextFile\EdgeListOut.txt)");

break;

}

}

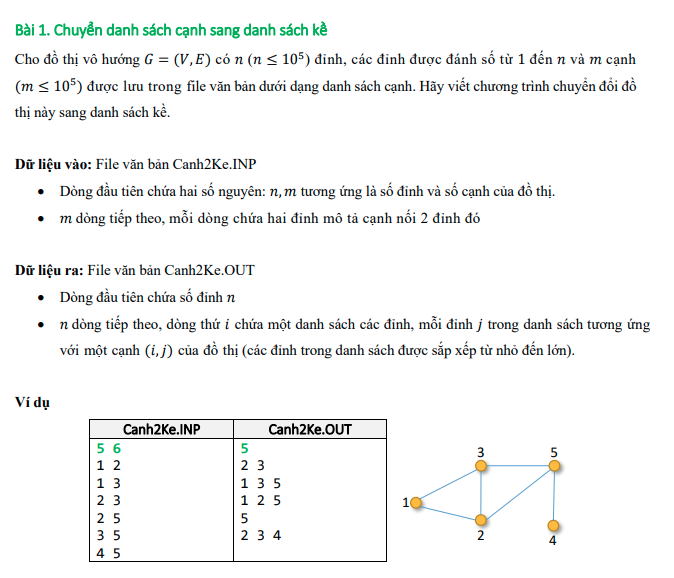
} while (choice != 0);

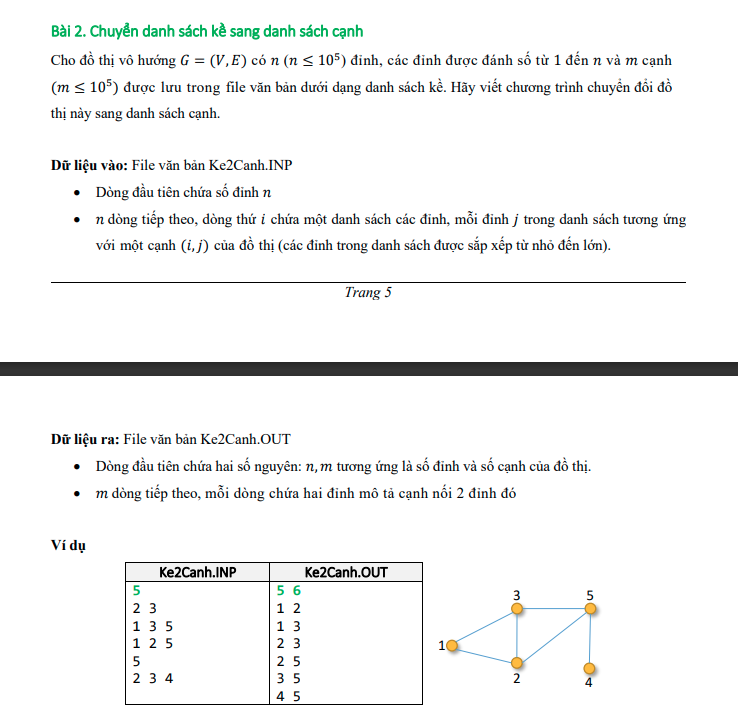
}

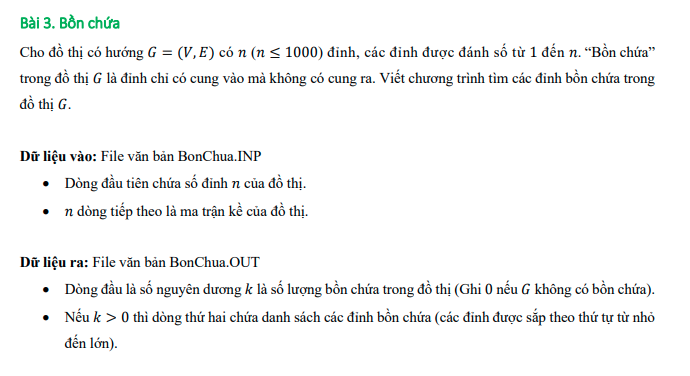
}

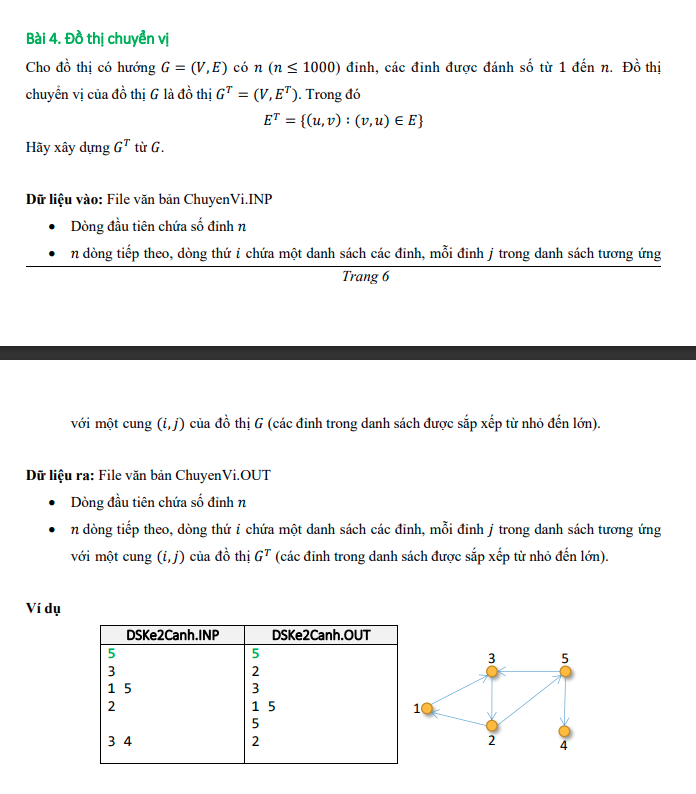
}

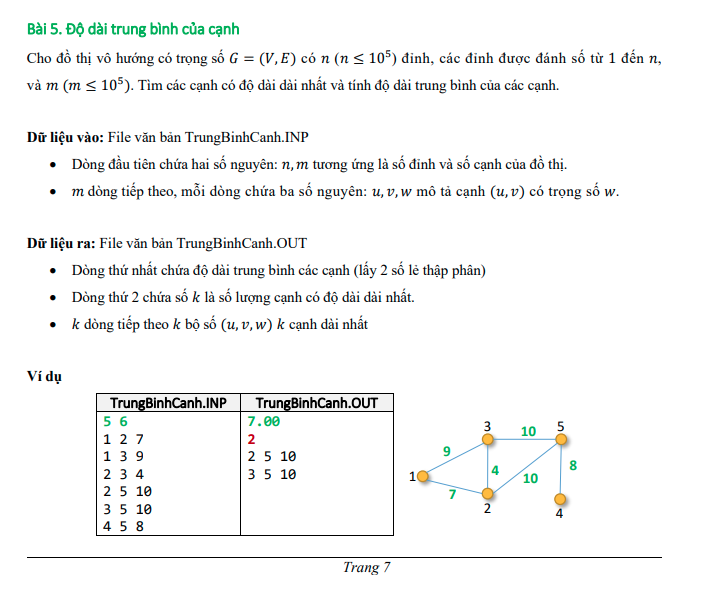
Buổi 2: Vận dụng các thao tác trên đồ thị











CODE: BÀI 1 + 2:

AdjList.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bai1

{

internal class AdjList

{

LinkedList<int>[] v;

int n;

public LinkedList<int>[] V { get => v; set => v = value; }

public int N { get => n; set => n = value; }

public AdjList() { }

public AdjList(int k)

{

v = new LinkedList<int>[k];

}

public AdjList(LinkedList<int>[] g)

{

v = g; //Lấy giá trị của nhau

}

public void FileToAdjList(string filePath)

{

StreamReader reader = new StreamReader(filePath);

n = int.Parse(reader.ReadLine()) + 1;

v = new LinkedList<int>[n];

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

{

v[i] = new LinkedList<int>();

string st = reader.ReadLine();

if (st != "") //Đặt điều kiện để chặn đỉnh cô lập

{

string[] s = st.Split();

for (int j = 0; j < s.Length; j++)

{

int value = int.Parse(s[j]);

v[i].AddLast(value);

}

}

}

reader.Close();

}

public void Output()

{

int m = n - 1;

Console.WriteLine("Đồ thị danh sách kề - số đỉnh : " + m); for (int i = 1; i < v.Length; i++)

{

Console.Write(" Đỉnh {0} ->", i);

foreach (int x in v[i])

Console.Write("{0, 3}", x);

Console.WriteLine();

}

}

public void AdjListToFile(string filePath)

{

StreamWriter sw = new StreamWriter(filePath);

int m = n - 1;

sw.WriteLine("Đồ thị danh sách kề - số đỉnh : " + m);

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

{

sw.Write(" Đỉnh {0} ->", i);

foreach (int x in v[i])

{

sw.Write("{0, 3}", x);

}

sw.WriteLine();

}

sw.Close();

}

}

}

EdgeList.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bai1

{

internal class EdgeList

{

LinkedList<Tuple<int, int>> edges;

int n;

int m;

public LinkedList<Tuple<int, int>> Edges { get => edges; set => edges = value; }

public int N { get => n; set => n = value; }

public int M { get => m; set => m = value; }

public EdgeList()

{

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

}

public void FileToEdgeList(string filePath)

{

StreamReader reader = new StreamReader(filePath);

string[] s = reader.ReadLine().Split();

n = int.Parse(s[0]);

m = int.Parse(s[1]);

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

{

s = reader.ReadLine().Split();

Tuple<int, int> newEdge = new Tuple<int, int>(int.Parse(s[0]), int.Parse(s[1]));

edges.AddLast(newEdge);

}

reader.Close();

}

public void Output()

{

Console.WriteLine("So đỉnh: n = " + (edges.Count - 1));

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

{

Console.WriteLine(" (" + edge.Item1 + "," + edge.Item2 + ")");

}

}

public void EdgeListToFile(string filePath)

{

StreamWriter sw = new StreamWriter(filePath);

int m = n - 1;

sw.WriteLine("Danh sách cạnh của đồ thị với số đỉnh n = " + m);

foreach (Tuple<int, int> e in edges)

sw.WriteLine(" (" + e.Item1 + "," + e.Item2 + ")");

sw.Close();

}

}

}

Program Bài 1 + 2:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bai1

{

internal class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.Unicode;

Console.InputEncoding = Encoding.Unicode;

EdgeList edgeList = new EdgeList();

[edgeList.FileToEdgeList(@"../../TextFile/EdgeList.txt](mailto:edgeList.FileToEdgeList(@"../../TextFile/EdgeList.txt)");

edgeList.Output();

AdjList adjList = EdgeListToAdjList(edgeList);

adjList.Output();

[adjList.AdjListToFile(@"../../TextFile/AdjListOutput.txt](mailto:adjList.AdjListToFile(@"../../TextFile/AdjListOutput.txt)");

//AdjList adjList = new AdjList();

[//adjList.FileToAdjList(@"../../TextFile/AdjList.txt](mailto://adjList.FileToAdjList(@"../../TextFile/AdjList.txt)");

//adjList.Output();

//EdgeList edgeList = AdjListToEdgeList(adjList);

[//edgeList.EdgeListToFile(@"../../TextFile/EdgeListOutput.txt](mailto://edgeList.EdgeListToFile(@"../../TextFile/EdgeListOutput.txt)");

Console.ReadKey();

}

static AdjList EdgeListToAdjList(EdgeList edgeList)

{

AdjList adjList = new AdjList();

adjList.N = edgeList.N;

adjList.V = new LinkedList<int>[adjList.N + 1];

for (int i = 1; i < adjList.N + 1; i++)

{

adjList.V[i] = new LinkedList<int>();

}

foreach (Tuple<int, int> e in edgeList.Edges)

{

adjList.V[e.Item1].AddLast(e.Item2);

adjList.V[e.Item2].AddLast(e.Item1);

}

return adjList;

}

static EdgeList AdjListToEdgeList(AdjList adjList)

{

EdgeList edgeList = new EdgeList();

edgeList.N = adjList.N;

for (int i = 1; i < adjList.N; i++)

{

foreach (var x in adjList.V[i])

{

if (edgeList.Edges.Contains(new Tuple<int, int>(x, i)) == false)

{

Tuple<int, int> e = new Tuple<int, int>(i, x);

edgeList.Edges.AddLast(e);

edgeList.M++;

}

}

}

return edgeList;

}

}

}

Bài 3:

AdjList.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bai3

{

internal class AdjMatrix

{

private int n;

private int[,] matrix;

public int N { get => n; set => n = value; }

public int[,] Matrix { get => matrix; set => matrix = value; }

public AdjMatrix() { }

public AdjMatrix(int k)

{

n = k;

matrix = new int[n, n];

}

public void FileToAdjMatrix(string filePath)

{

StreamReader sr = new StreamReader(filePath);

n = int.Parse(sr.ReadLine());

matrix = new int[n, n];

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

{

string[] s = sr.ReadLine().Split();

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

{

matrix[i, j] = int.Parse(s[j]);

}

}

sr.Close();

}

public void Output()

{

Console.WriteLine("Đồ thị ma trận kề - số đỉnh : " + n);

Console.WriteLine();

Console.Write(" Đỉnh |");

for (int i = 0; i < n; i++) Console.Write(" {0}", i);

Console.WriteLine(); Console.WriteLine(" " + new string('-', 6 \* n));

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

{

Console.Write(" {0} |", i);

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

Console.Write(" {0, 3}", matrix[i, j]);

Console.WriteLine();

}

}

public bool IsStorage(int i)

{

for (int j = 0; j < n - 1; j++)

{

if (matrix[i, j] == 1)

{

return false;

}

}

return true;

}

}

}

Program Bài 3 + 4

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bai3

{

internal class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.Unicode;

Console.InputEncoding = Encoding.Unicode;

AdjMatrix matrix = new AdjMatrix();

[matrix.FileToAdjMatrix(@"../../TextFile/DirectedMatrix.txt](mailto:matrix.FileToAdjMatrix(@"../../TextFile/DirectedMatrix.txt)");

matrix.Output();

Storage(matrix, @"../../TextFile/DirectedMatrixOutput.txt");

Console.ReadKey();

}

static void Storage(AdjMatrix g, string fileOut)

{

StreamWriter sw = new StreamWriter(fileOut);

int count = 0;

for (int i = 0; i < g.N; i++)

{

if (g.IsStorage(i) == true)

{

count++;

Console.WriteLine("Đỉnh " + i + " là đỉnh bồn chứa");

sw.WriteLine("Đỉnh " + i + " là đỉnh bồn chứa");

}

}

sw.WriteLine("Số đỉnh là bồn chứa : " + count);

Console.WriteLine("Số đỉnh là bồn chứa : " + count);

sw.Close();

}

static AdjList TransposeG(AdjList g)

{

AdjList G = new AdjList();

G.N = g.N;

G.V = new LinkedList<int>[G.N];

for (int i = 0; i < G.N; i++)

G.V[i] = new LinkedList<int>();

for (int i = 1; i < G.N; i++)

{

foreach (int x in g.V[i])

G.V[x].AddLast(i);

}

return G;

}

}

}

Bài 5:

WeightList.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bai5

{

internal class WeightEdgeList

{

LinkedList<Tuple<int, int, int>> g;

int n;

int m;

public int N { get => n; set => n = value; }

public int M { get => m; set => m = value; }

public LinkedList<Tuple<int, int, int>> G { get => g; set => g = value; }

public WeightEdgeList()

{

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

}

public void FileToWeightEdgeList(string filePath)

{

StreamReader sr = new StreamReader(filePath);

string[] s = sr.ReadLine().Split();

n = int.Parse(s[0]);

m = int.Parse(s[1]);

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

{

s = sr.ReadLine().Split();

Tuple<int, int, int> e = new Tuple<int, int, int>(int.Parse(s[0]), int.Parse(s[1]), int.Parse(s[2]));

g.AddLast(e);

}

sr.Close();

}

public void Output()

{

Console.WriteLine("Danh sách cạnh của đồ thị với số đỉnh n = " + n);

foreach (Tuple<int, int, int> e in g)

Console.WriteLine(" (" + e.Item1 + "," + e.Item2 + ")" + "=" + e.Item3);

}

}

}

Program Bài 5:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bai5

{

internal class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.Unicode;

Console.InputEncoding = Encoding.Unicode;

WeightEdgeList weightEdgeList = new WeightEdgeList();

[weightEdgeList.FileToWeightEdgeList(@"../../TextFile/WeightEdgeList.txt](mailto:weightEdgeList.FileToWeightEdgeList(@"../../TextFile/WeightEdgeList.txt)");

weightEdgeList.Output();

Console.Write("Số cạnh dài: ");

MaxEdge(weightEdgeList);

Console.WriteLine("Avarage: {0:0.00}", AverageEdge(weightEdgeList));

Console.ReadKey();

}

static void MaxEdge(WeightEdgeList g)

{

int max = -int.MaxValue;

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

foreach (Tuple<int, int, int> e in g.G)

{

if (e.Item3 > max)

{

max = e.Item3;

}

}

foreach (Tuple<int, int, int> e in g.G)

{

if (e.Item3 == max)

{

lst.AddLast(e);

}

}

Console.WriteLine("Số lượng : {0} cạnh", lst.Count);

foreach (Tuple<int, int, int> e in lst)

{

Console.WriteLine(" ({0}, {1}) = {2}", e.Item1, e.Item2, e.Item3);

}

}

static double AverageEdge(WeightEdgeList g)

{

double avg = 0;

foreach (Tuple<int, int, int> e in g.G)

{

avg = avg + e.Item3;

}

avg = avg / g.G.Count;

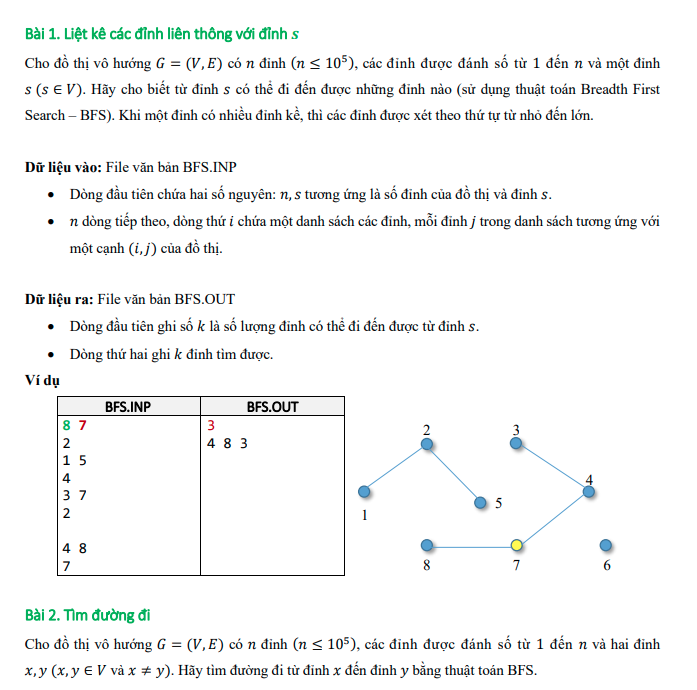
return avg;

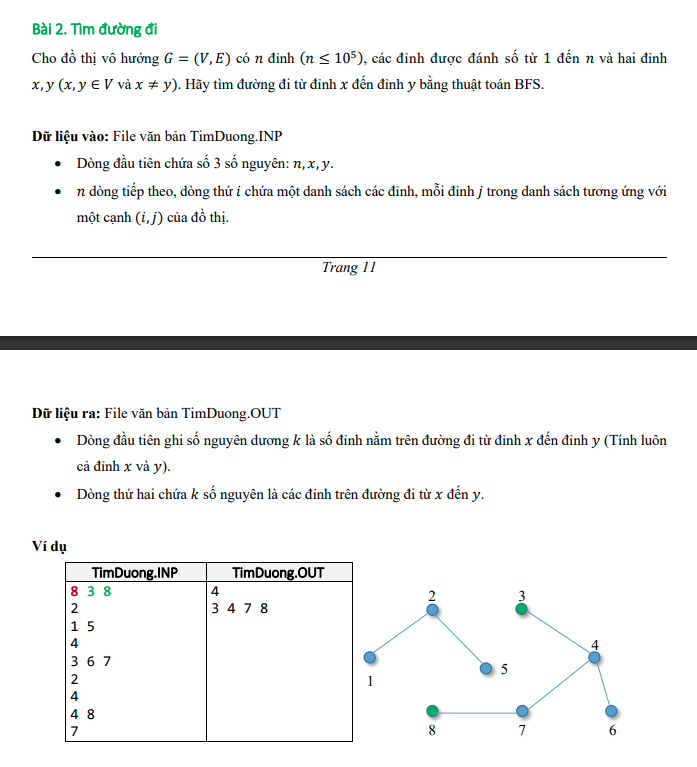
}

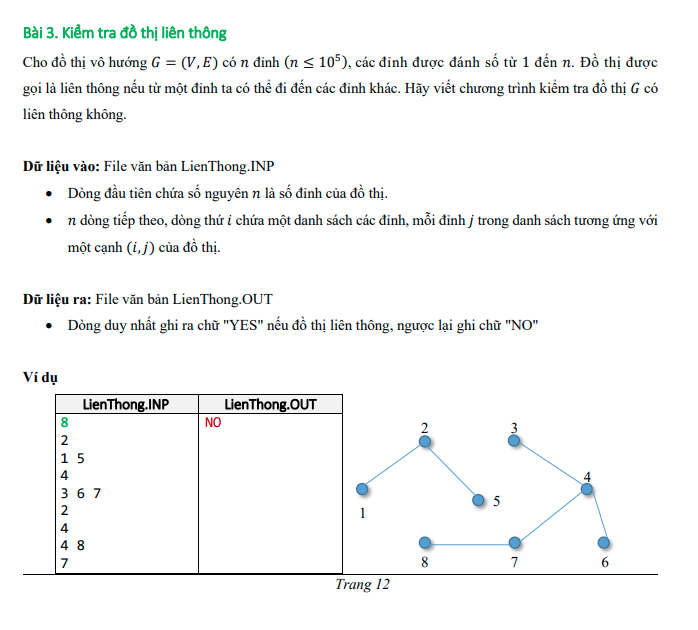
}

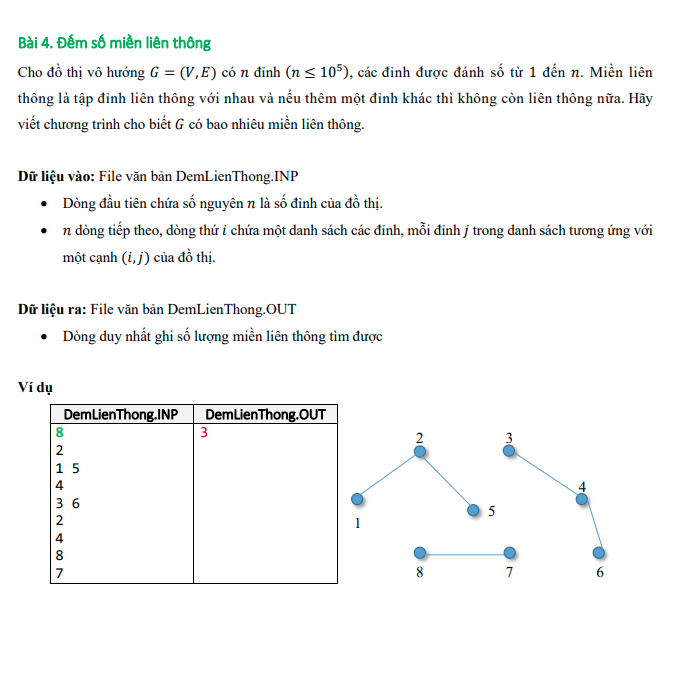
}

Buổi 3:









CODE:

AdjList.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.ComponentModel;

using System.Runtime.InteropServices;

namespace BT1

{

internal class AdjList

{

LinkedList<int>[] list;

int n;

bool[] diQua;

int[] index;

int inconnect;

public LinkedList<int>[] List { get => list; set => list = value; }

public int N { get => n; set => n = value; }

public int Inconnect { get => inconnect; set => inconnect = value; }

public AdjList() { }

public AdjList(int k)

{

list = new LinkedList<int>[k];

}

public AdjList(LinkedList<int>[] newList)

{

list = newList;

}

public void FileToAdjList(string filePath)

{

StreamReader reader = new StreamReader(filePath);

n = int.Parse(reader.ReadLine()) + 1;

list = new LinkedList<int>[n];

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

{

list[i] = new LinkedList<int>();

string value = reader.ReadLine();

if (value != "")

{

string[] s = value.Split();

for (int j = 0; j < s.Length; j++)

{

int x = int.Parse(s[j]);

list[i].AddLast(x);

}

}

}

reader.Close();

}

public void Output()

{

int m = n - 1;

for (int i = 1; i < list.Length; i++)

{

Console.Write("Dinh {0} => ", i);

foreach (int value in list[i])

{

Console.Write("{0, 3}", value);

}

Console.WriteLine();

}

}

public void BFS(int start)

{

diQua = new bool[list.Length]; // khởi tạo cho các giá trị của mảng đánh dấu

for (int i = 1; i < diQua.Length; i++)

{

diQua[i] = false; // set cho tất cả các giá trị chưa đi qua = false

}

Queue<int> queue = new Queue<int>();

diQua[start] = true; // đánh dấu là đã duyệt s

queue.Enqueue(start);

string tmp = ""; // tạo một chuỗi để lưu kết quả

while (queue.Count != 0)

{

start = queue.Dequeue();

tmp = tmp + start;

foreach (int value in list[start]) // xét các phần tử liền kề với nó

{

if (diQua[value] == true)

{

continue;

}

diQua[value] = true;

queue.Enqueue(value);

}

tmp += " ";

}

string[] tmps = tmp.TrimEnd().Split(' ');

Console.WriteLine("So luong dinh di tu {0} => {1} ", tmps[0], tmps.Length - 1);

Console.Write("Cac dinh lien thong voi {0}: ", tmps[0]);

for (int i = 1; i < tmps.Length; i++)

{

Console.Write(tmps[i] + " ");

}

Console.WriteLine();

}

public void BFS\_X\_to\_Y(int x, int y)

{

int[] dinhNamTruoc = new int[list.Length]; // Mảng này dùng để lưu các đỉnh nằm trước trên đường đi

for (int i = 1; i < dinhNamTruoc.Length; i++)

{

dinhNamTruoc[i] = -1;

}

diQua = new bool[list.Length];

for (int i = 1; i < diQua.Length; i++)

{

diQua[i] = false; // set tất cả giá trị ban đầu cho tất cả đỉnh bằng false tại chưa đi qua

}

Queue<int> queue = new Queue<int>();

diQua[x] = true; // set cho vị trí x ban đầu đã đi qua rồi tại vì bắt đầu từ x

queue.Enqueue(x); // Queue vào trước ra trước, theo BFS thì thuật toán tìm kiếm sẽ lan rộng ra ( khi cho node vào thì nó sẽ tìm các node liền kề với nó )

while (queue.Count != 0)

{

int s = queue.Dequeue(); // tạo một biến để chứa một đỉnh được dequeue ra

foreach (int value in list[s]) // check những cạnh liền kề với đỉnh đó

{

if (diQua[value] == true) // khi node này đã được đi qua rồi thì bỏ qua

{

continue;

}

diQua[value] = true; //nếu như là false thì khi đi qua sẽ set cho nó bằng true

queue.Enqueue(value); //Enqueue lại

dinhNamTruoc[value] = s; //

}

}

//Thao tác xuất đường đi từ x => y

Console.WriteLine();

int k = y;

Stack<int> stack = new Stack<int>();

while (dinhNamTruoc[k] != -1)

{

stack.Push(k);

k = dinhNamTruoc[k];

}

Console.WriteLine();

Console.Write("Duong di tu " + x + " => " + y + ": " + x);

while (stack.Count != 0)

{

k = stack.Pop();

Console.Write("-> " + k);

}

Console.WriteLine();

}

public void Connected()

{

inconnect = 0; // inconnect ở đây là số lượng các thành phần liên thông trong đồ thị nhập vào

index = new int[n]; // index dùng để lưu các đỉnh có cùng thành phần liên thông

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

{

index[i] = -1;

}

// khởi tạo các giá trị cho diQua

diQua = new bool[n];

for (int i = 0; i < diQua.Length; i++)

{

diQua[i] = false;

}

for (int i = 1; i < diQua.Length; i++)

{

if (diQua[i] == false)

{

//khởi đầu cho 1 thành phần liên thông mới

inconnect++;

BFS\_Connected(i); // function này đi tìm các đỉnh nằm trong vùng liên thông với đỉnh i

}

Console.WriteLine();

}

}

public void BFS\_Connected(int s)

{

Queue<int> queue = new Queue<int>();

diQua[s] = true;

queue.Enqueue(s);

while (queue.Count != 0)

{

s = queue.Dequeue();

index[s] = inconnect;

foreach (int value in list[s])

{

if (diQua[value] == false)

{

diQua[value] = true;

queue.Enqueue(value);

}

}

}

}

public void Output\_Connected() //hàm này dùng để xuất ra các thành phần liên thông

{

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

{

Console.WriteLine("TPLT {0} ", i);

for (int j = 0; j < index.Length; j++)

{

if (index[j] == i)

{

Console.Write(j + " ");

}

}

Console.WriteLine();

}

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BT1

{

internal class Program

{

static void Main(string[] args)

{

AdjList list = new AdjList();

//Câu 1

[//list.FileToAdjList(@"../../TextFile/BFSInput.txt](mailto://list.FileToAdjList(@"../../TextFile/BFSInput.txt)");

//Console.Write("Nhap dinh bat dau: ");

//int x = int.Parse(Console.ReadLine());

//list.BFS(x);

//Câu 2

[//list.FileToAdjList(@"../../TextFile/BFSInput.txt](mailto://list.FileToAdjList(@"../../TextFile/BFSInput.txt)");

//Console.Write("Enter x: ");

//int x = int.Parse(Console.ReadLine());

//Console.Write("Enter y: ");

//int y = int.Parse(Console.ReadLine());

//list.BFS\_X\_to\_Y(x, y);

//Câu 3

[list.FileToAdjList(@"../../TextFile/AdjList2Input.txt](mailto:list.FileToAdjList(@"../../TextFile/AdjList2Input.txt)");

list.Output();

list.Connected();

if (list.Inconnect == 1)

{

Console.Write("Do thi lien thong");

}

else

{

Console.WriteLine("Do thi khong lien thong");

Console.WriteLine("Do thi co {0} TPLT", list.Inconnect);

list.Output\_Connected();

}

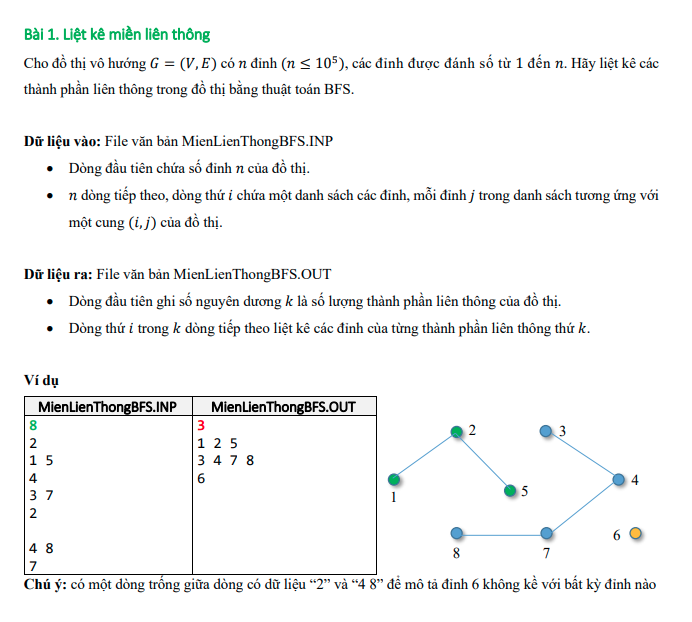
Console.ReadKey();

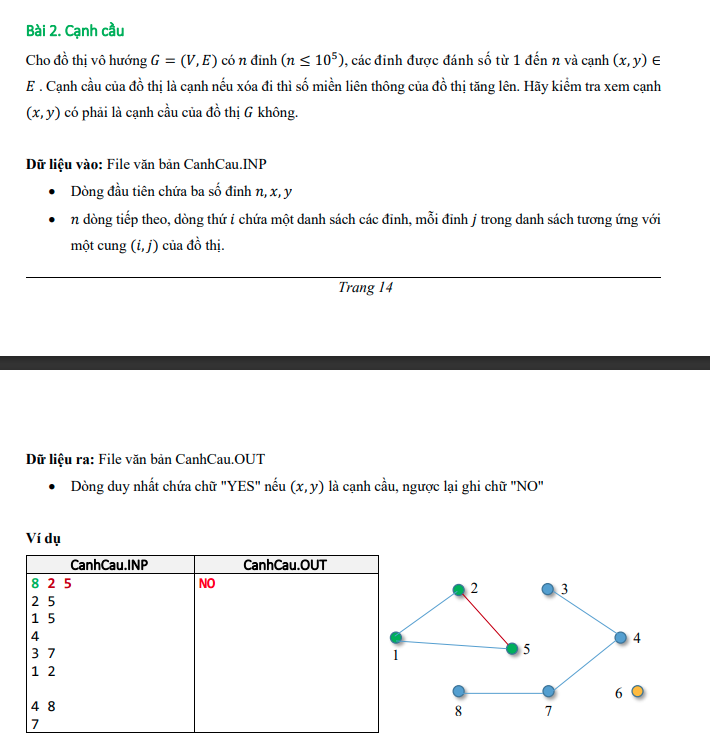
}

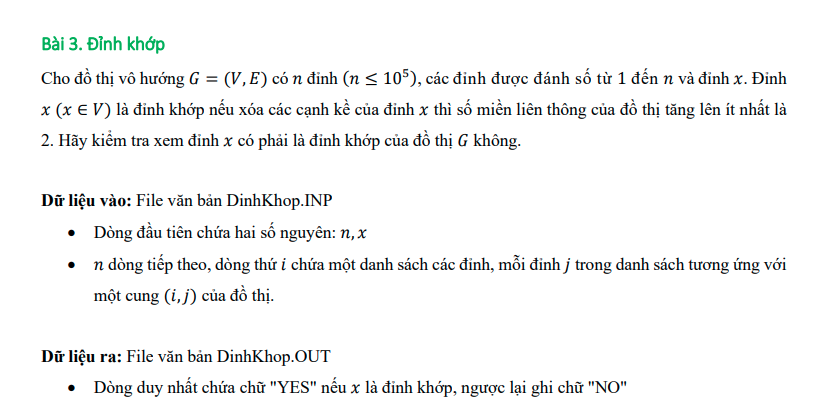
}

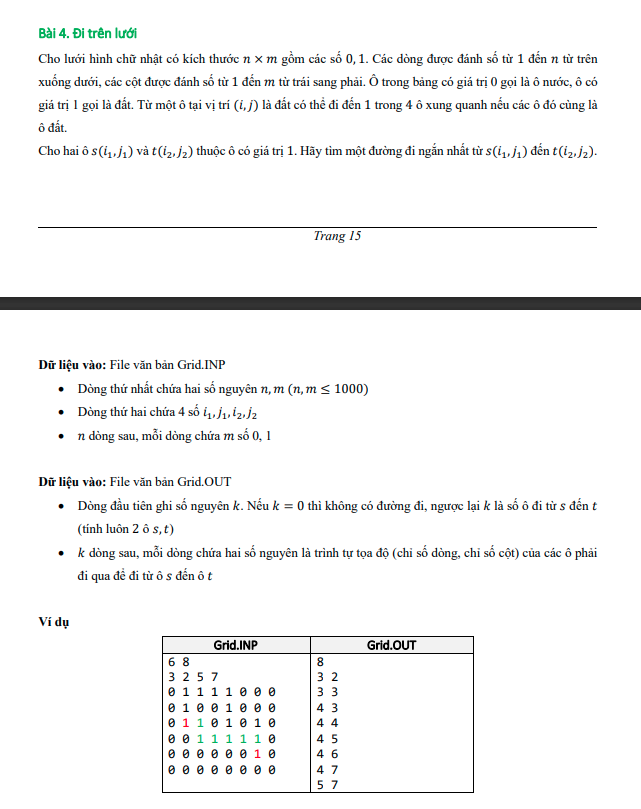
}

Buổi 4:









CODE:

AdjList.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace BT1

{

internal class AdjList

{

LinkedList<int>[] list;

int n; // số đỉnh

bool[] visited; // mảng đánh dấu trạng thái dùng để đánh dấu các thành phần đã đi qua

int[] index; //dùng để đánh dấu các TPLT

int inconnect; // dùng để đếm các TPLT

public LinkedList<int>[] List { get => list; set => list = value; }

public int N { get => n; set => n = value; }

public int Inconnect { get => inconnect; set => inconnect = value; }

public AdjList() { }

public AdjList(int k)

{

n = k;

list = new LinkedList<int>[n];

}

public AdjList(LinkedList<int>[] newList)

{

list = newList;

}

public void FileToAdjList(string filePath)

{

StreamReader reader = new StreamReader(filePath);

n = int.Parse(reader.ReadLine()) + 1;

list = new LinkedList<int>[n];

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

{

list[i] = new LinkedList<int>();

string line = reader.ReadLine();

if (line != "")

{

string[] s = line.Split();

for (int j = 0; j < s.Length; j++)

{

int x = int.Parse(s[j]);

list[i].AddLast(x);

}

}

}

reader.Close();

}

public void Output()

{

int m = n - 1;

Console.WriteLine("Do thi danh sach ke - so dinh: " + m);

for (int i = 1; i < list.Length; i++)

{

Console.Write("Dinh {0} => ", i);

foreach (var value in list[i])

{

Console.Write("{0, 3}", value);

}

Console.WriteLine();

}

}

public void RemoveEdgeX(int x) // hàm này dùng để xóa cạnh kề của một đỉnh để xét coi đỉnh đó có phải đỉnh chóp hay không

{

for (int i = 1; i < list.Length; i++)

{

if (i == x)

{

list[i].Clear();

}

else

{

list[i].Remove(x);

}

}

}

public void RemoveEdgeXY(int x, int y)

{

list[x].Remove(y);

list[y].Remove(x);

}

public void BFS\_Connected(int s)

{

Queue<int> queue = new Queue<int>();

visited[s] = true;

queue.Enqueue(s);

while (queue.Count != 0) // lặp lại cho đến khi queue không còn phần tử nào hết

{

s = queue.Dequeue(); // lấy từ trong queue ra 1 phần tử

index[s] = inconnect; //

foreach (int value in list[s])

{

if (visited[value] == false)

{

visited[value] = true;

queue.Enqueue(value);

}

}

}

}

public void Connected()

{

inconnect = 0;

index = new int[n];

for (int i = 0; i < index.Length; i++)

{

index[i] = -1;

}

visited = new bool[n];

for (int i = 0; i < visited.Length; i++)

{

visited[i] = false;

}

for (int i = 1; i < list.Length; i++)

{

if (visited[i] == false)

{

inconnect++;

BFS\_Connected(i);

}

Console.WriteLine();

}

}

public void PathOnGrid(int x, int y, int row, int col, int[,] a)

{

// mảng này dung đẻ lưu đỉnh nằm trước trên đường đi

int[] pre = new int[list.Length];

for (int i = 0; i < list.Length; i++)

{

pre[i] = -1;

}

visited = new bool[list.Length];

for (int i = 0; i < visited.Length; i++)

{

visited[i] = false;

}

Queue<int> queue = new Queue<int>();

visited[x] = true;

queue.Enqueue(x);

while (queue.Count != 0)

{

int s = queue.Dequeue();

foreach (var value in list[s])

{

if (visited[value]) continue;

visited[value] = true;

queue.Enqueue(value);

pre[value] = s;

}

}

Console.WriteLine("Tim duong di");

Console.WriteLine();

int k = y;

Stack<int> stack = new Stack<int>(); // stack thứ nhất dùng để xuất ra tọa độ

Stack<int> stack2 = new Stack<int>(); // stack thứ hai dùng để xuất tô màu

while (k != -1)

{

stack.Push(k);

stack2.Push(k);

k = pre[k];

}

Console.WriteLine();

Console.WriteLine("Duong di tu ({0},{1}) => ({2},{3}) la: ", (x / col), x % col, (y / col), y % col, (x / col), x % col);

Console.WriteLine();

while (stack.Count > 0)

{

k = stack.Pop();

Tuple<int, int> tuple = new Tuple<int, int>(k / col, k % col);

Console.Write("=> ({0},{1})", tuple.Item1, tuple.Item2);

}

Console.WriteLine();

stack2.Push(x);

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

{

Console.WriteLine();

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

{

k = j + i \* col;

if (stack2.Contains(k))

{

Console.BackgroundColor = ConsoleColor.Magenta;

Console.Write(" " + a[i, j]);

}

else if (!stack2.Contains(k) && a[i, j] == 1)

{

Console.BackgroundColor = ConsoleColor.Green;

Console.Write(" " + a[i, j]);

}

else

{

Console.BackgroundColor = ConsoleColor.Black;

Console.Write(" " + a[i, j]);

}

}

}

Console.WriteLine();

}

public void GridToAdjList(string filePath)

{

StreamReader sr = new StreamReader(filePath);

string[] s = sr.ReadLine().Split();

int row = int.Parse(s[0]);

int col = int.Parse(s[1]);

n = row \* col;

list = new LinkedList<int>[n];

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

{

list[i] = new LinkedList<int>();

}

s = sr.ReadLine().Split();

int x = int.Parse(s[1]) + int.Parse(s[0]) \* col;

int y = int.Parse(s[3]) + int.Parse(s[2]) \* col;

int[,] a = new int[row, col];

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

{

s = sr.ReadLine().Split();

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

{

a[i, j] = int.Parse(s[j]);

}

}

Console.WriteLine("Ma trận lưới a[] : ");

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

{

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

{

if (a[i, j] == 1)

{

Console.BackgroundColor = ConsoleColor.Green;

Console.Write(" " + a[i, j]);

Console.BackgroundColor = ConsoleColor.Black;

}

else

{

Console.Write(" " + a[i, j]);

}

}

Console.WriteLine();

}

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

{

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

{

if (a[i, j] == 1)

{

int k = j + i \* col;

if (i > 0 && a[i - 1, j] == 1)

{

if (list[k].Contains(k - col) == false) list[k].AddLast(k - col);

if (list[k - col].Contains(k) == false) list[k - col].AddLast(k);

}

if (j > 0 && a[i, j - 1] == 1)

{

if (list[k].Contains(k - 1) == false) list[k].AddLast(k - 1);

if (list[k - 1].Contains(k) == false) list[k - 1].AddLast(k);

}

if (j < col - 1 && a[i, j + 1] == 1)

{

if (list[k].Contains(k + 1) == false) list[k].AddLast(k + 1);

if (list[k + 1].Contains(k) == false) list[k + 1].AddLast(k);

}

if (i < row - 1 && a[i + 1, j] == 1)

{

if (list[k].Contains(k + col) == false) list[k].AddLast(k + col);

if (list[k + col].Contains(k) == false) list[k + col].AddLast(k);

}

}

}

}

PathOnGrid(x, y, row, col, a);

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BT1

{

internal class Program

{

static void Main(string[] args)

{

AdjList list = new AdjList();

// Câu 3:

[list.GridToAdjList(@"../../TextFile/GridInput.txt](mailto:list.GridToAdjList(@"../../TextFile/GridInput.txt)");

////Câu 1 + 2:

//list.Output();

//list.Connected();

//int listInconnect1 = list.Inconnect;

//Console.WriteLine("Nhap dinh can xet: ");

//int x = int.Parse(Console.ReadLine());

//list.RemoveEdgeX(x);

//list.Connected();

//int listInconnect2 = list.Inconnect;

//if (listInconnect2 > listInconnect1 + 1)

//{

// Console.Write("{0} la dinh khop", x);

//}

//else

//{

// Console.Write("{0} khong la dinh khop", x);

//}

//Console.Write("Enter edge x: ");

//int x = int.Parse(Console.ReadLine());

//Console.Write("Enter edge y: ");

//int y = int.Parse(Console.ReadLine());

//list.RemoveEdgeXY(x, y);

//list.Connected();

//int listInconnect2 = list.Inconnect;

//if (listInconnect2 > listInconnect1)

//{

// Console.Write("{0} va {1} la canh cau", x, y);

//}

//else

//{

// Console.Write("{0} va {1} khong phai la canh cau", x, y);

//}

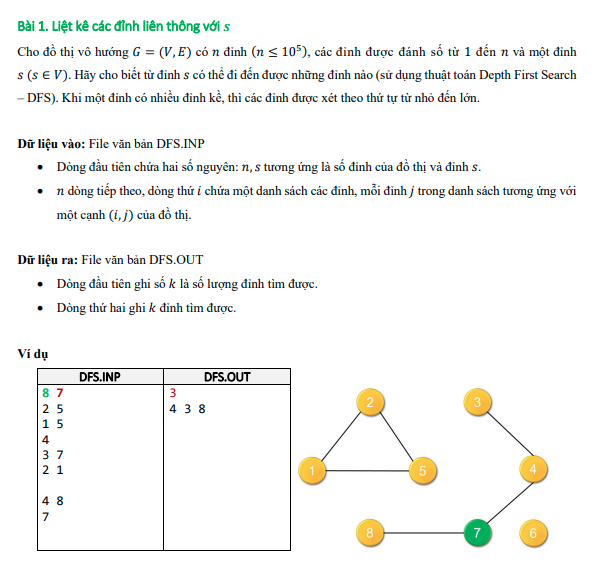
Console.ReadKey();

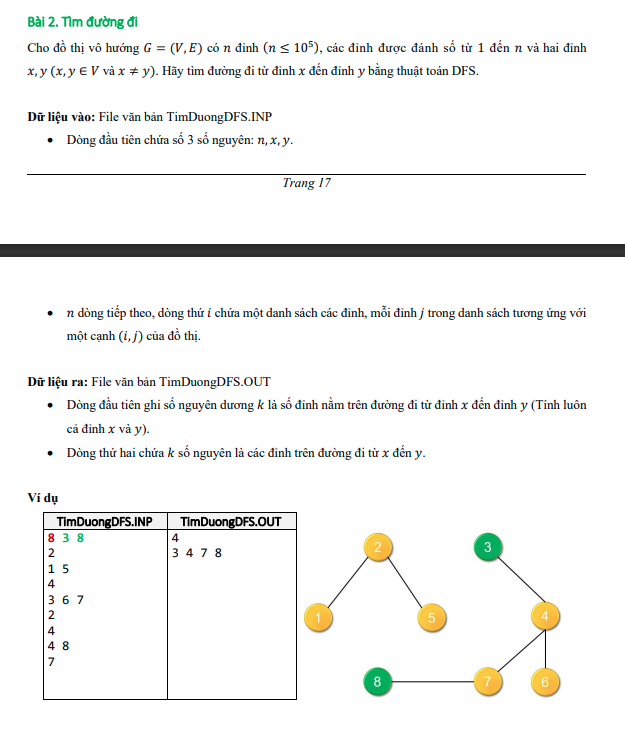
}

}

}

Buổi 5:





CODE:

AdjList.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buoi5TH

{

internal class AdjList

{

LinkedList<int>[] list;

List<int> duongdi;

int n;

bool[] diqua;

int[] index;

int inconnect; // Đếm các thành phần liên thông

int[] previous;

int[] color; // Use to color

public int N { get => n; set => n = value; }

public LinkedList<int>[] List { get => list; set => list = value; }

public int Inconnect { get => inconnect; set => inconnect = value; }

public AdjList() { }

public AdjList(int k)

{

list = new LinkedList<int>[k];

}

public AdjList(LinkedList<int>[] newLinkedList)

{

list = newLinkedList;

}

public void FileToAdjList(string filePath)

{

StreamReader sr = new StreamReader(filePath);

n = int.Parse(sr.ReadLine()) + 1;

list = new LinkedList<int>[n];

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

{

list[i] = new LinkedList<int>();

string st = sr.ReadLine();

if (st != "")

{

string[] s = st.Split();

for (int j = 0; j < s.Length; j++)

{

int x = int.Parse(s[j]);

list[i].AddLast(x);

}

}

}

sr.Close();

}

public void Xuat()

{

int count = n - 1;

Console.WriteLine("So dinh: " + count);

for (int i = 1; i < list.Length; i++)

{

Console.Write("{0} ->", i);

foreach (int x in list[i])

{

Console.Write("{0, 3}", x);

}

Console.WriteLine();

}

}

public void DegV(string filePath)

{

StreamWriter sw = new StreamWriter(filePath);

Console.WriteLine("Bậc của các đỉnh :");

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

{

int deg = 0;

deg+= list[i].Count;

Console.WriteLine("{0} : {1}", i, deg);

sw.WriteLine(i + " : " + deg);

}

sw.Close();

}

public void \_DFS(int s)

{

duongdi = new List<int>();

diqua = new bool[n];

for (int i = 1; i < n; i++) diqua[i] = false;

if (diqua[s] == false)

{

DFS(s);

}

XuatDFS(s);

}

public void DFS(int s)

{

diqua[s] = true;

duongdi.Add(s);

foreach (int u in list[s])

{

if (diqua[u] == false)

{

DFS(u);

}

}

}

public void DFSfromXtoY(int x, int y)

{

if (list[x].Count == 0 || list[y].Count == 0)

{

Console.WriteLine("Khong co duong");

return;

}

diqua = new bool[n];

for (int i = 1; i < n; i++) // Set các giá trị của visited thành false hết

{

diqua[i] = false;

}

previous = new int[n];

for (int i = 1; i < n; i++) // Set giá trị cho mảng

{

previous[i] = -1;

}

RecurseDFSXtoY(x);

if (previous[y] == -1)

{

Console.WriteLine("No path" + y);

}

else

{

XuatXtoY(x, y);

}

}

public void RecurseDFSXtoY(int s)

{

diqua[s] = true;

foreach (int u in list[s])

if (diqua[u] == false)

{

previous[u] = s;

RecurseDFSXtoY(u);

}

}

public void XuatXtoY(int x, int y)

{

Stack<int> st = new Stack<int>();

int flag = y;

while (previous[flag] != -1)

{

st.Push(flag);

flag = previous[flag];

}

Console.WriteLine();

Console.Write(x + " -> " + y + " : " + x);

while (st.Count > 0)

{

flag = st.Pop();

Console.Write(" -> " + flag);

}

Console.WriteLine();

}

public void StackDFS(int s)

{

Stack<int> stack = new Stack<int>();

diqua[s] = true;

stack.Push(s);

while (stack.Count != 0)

{

s = stack.Pop();

index[s] = inconnect;

foreach (int u in list[s])

{

if (diqua[u] == false)

{

diqua[u] = true;

stack.Push(u);

}

}

}

}

public void XuatDFS(int s)

{

foreach (var item in duongdi)

{

if (item != s)

{

Console.Write(item + " ");

}

}

}

public void OutConnected()

{

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

{

Console.Write("TPLT {0}: ", i);

for (int j = 0; j < index.Length; j++)

{

if (index[j] == i)

{

Console.Write(j + " ");

}

}

Console.WriteLine();

}

}

public void Connected()

{

inconnect = 0;

index = new int[n];

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

{

index[i] = -1;

}

diqua = new bool[n];

for (int i = 0; i < diqua.Length; i++)

{

diqua[i] = false;

}

for (int i = 1; i < diqua.Length; i++)

{

if (!diqua[i])

{

inconnect++;

StackDFS(i);

}

}

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Buoi5TH

{

internal class Program

{

static void Main(string[] args)

{

//======================================================================

// Câu 1: Tìm các đỉnh liên thông với đỉnh x

//AdjList newList = new AdjList();

[//newList.FileToAdjList(@"../../TextFile/AdjList.txt](mailto://newList.FileToAdjList(@"../../TextFile/AdjList.txt)");

//Console.Write("Enter starting vertex: ");

//int x = int.Parse(Console.ReadLine());

//Console.Write("Cac dinh lien thong voi " + x + ": ");

//newList.\_DFS(x);

//Console.ReadKey();

//======================================================================

////Câu 2: Tìm đường đi từ X tới Y

//AdjList newList = new AdjList();

[//newList.FileToAdjList(@"../../TextFile/AdjList.txt](mailto://newList.FileToAdjList(@"../../TextFile/AdjList.txt)");

//Console.Write("Enter starting vertex: ");

//int x = int.Parse(Console.ReadLine());

//Console.Write("Enter ending vertex: ");

//int y = int.Parse(Console.ReadLine());

//newList.DFSfromXtoY(x, y);

//Console.ReadKey();

//======================================================================

// Câu 3: Xét tính liên thông của đồ thị, nếu đồ thị không liên thông thì cho biết đồ thị có bao nhiêu thành phần liên thông (TPLT) và xuất các thành phần liên thông lên màn hình

//AdjList newList = new AdjList();

[//newList.FileToAdjList(@"../../TextFile/AdjList2.txt](mailto://newList.FileToAdjList(@"../../TextFile/AdjList2.txt)");

//newList.Connected();

//Console.WriteLine("Đồ thị có {0} miền liên thông", newList.Inconnect);

//newList.OutConnected();

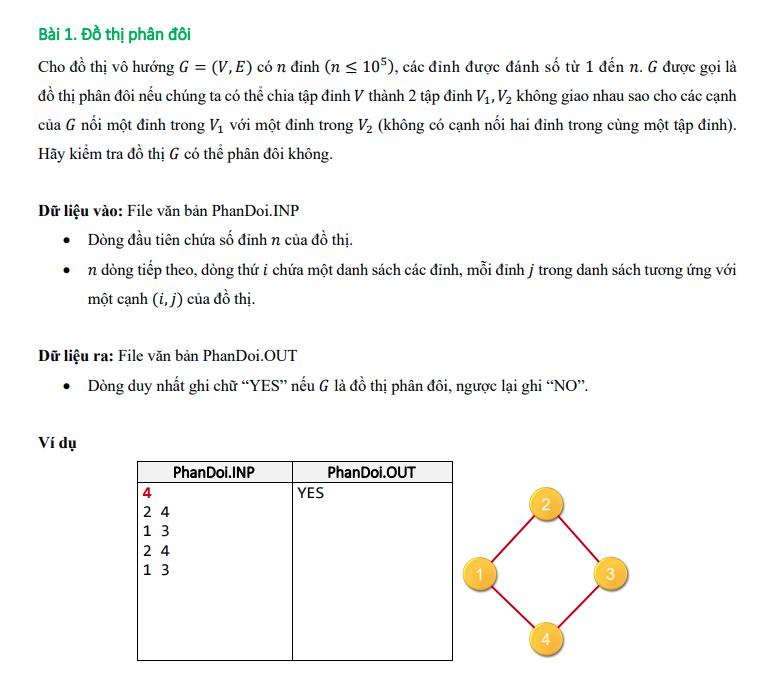
//Console.ReadKey();

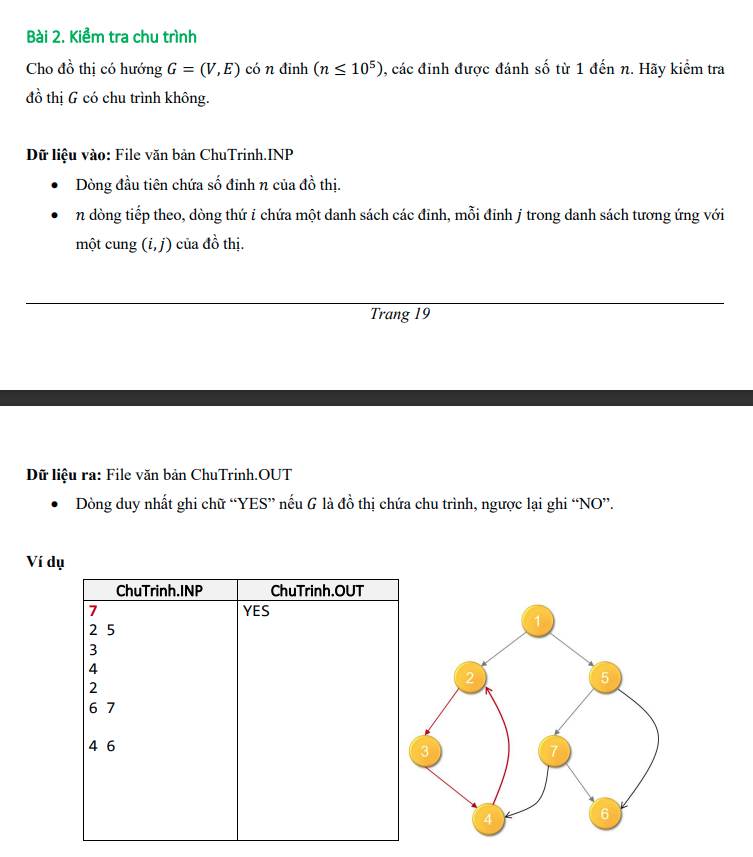
}

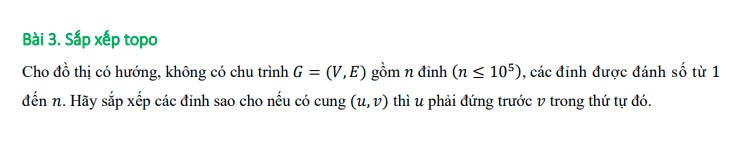
}

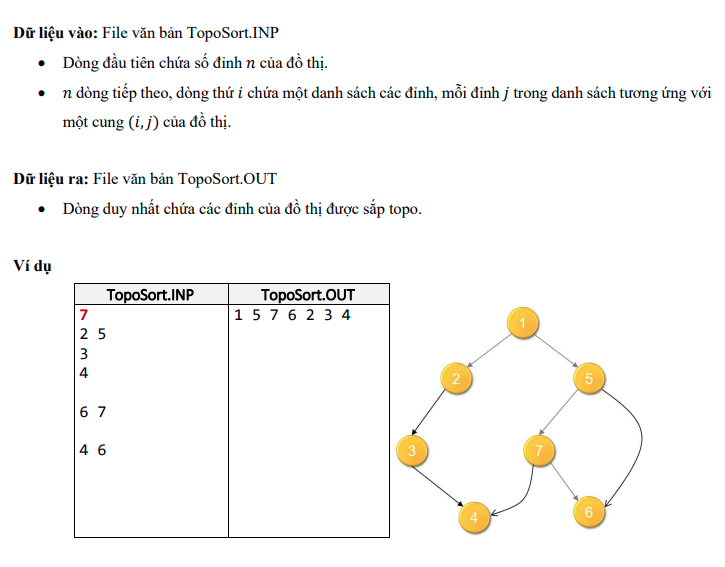
}

Buổi 6:









CODE:

AdjList.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BT6TH

{

internal class AdjList

{

LinkedList<int>[] list;

LinkedList<int> Topo;

int n;

bool[] diqua;

int[] index;

int inconnect; // Đếm các thành phần liên thông

int[] previous;

int[] color; // Use to color

public int N { get => n; set => n = value; }

public LinkedList<int>[] List { get => list; set => list = value; }

public int Inconnect { get => inconnect; set => inconnect = value; }

public AdjList() { }

public AdjList(int k)

{

list = new LinkedList<int>[k];

}

public AdjList(LinkedList<int>[] newLinkedList)

{

list = newLinkedList;

}

public void FileToAdjList(string filePath)

{

StreamReader sr = new StreamReader(filePath);

n = int.Parse(sr.ReadLine()) + 1;

list = new LinkedList<int>[n];

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

{

list[i] = new LinkedList<int>();

string st = sr.ReadLine();

if (st != "")

{

string[] s = st.Split();

for (int j = 0; j < s.Length; j++)

{

int x = int.Parse(s[j]);

list[i].AddLast(x);

}

}

}

sr.Close();

}

public void Xuat()

{

int count = n - 1;

Console.WriteLine("So dinh: " + count);

for (int i = 1; i < list.Length; i++)

{

Console.Write("{0} ->", i);

foreach (int x in list[i])

{

Console.Write("{0, 3}", x);

}

Console.WriteLine();

}

}

public void DegV(string filePath)

{

StreamWriter sw = new StreamWriter(filePath);

Console.WriteLine("Bậc của các đỉnh :");

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

{

int deg = 0;

deg += list[i].Count;

Console.WriteLine("{0} : {1}", i, deg);

sw.WriteLine(i + " : " + deg);

}

sw.Close();

}

public void Bipartite()

{

diqua = new bool[n];

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

{

diqua[i] = false;

}

color = new int[n];

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

{

color[i] = -1;

}

color[0] = 1;

if (IsBipartite(1) == true)

{

Console.WriteLine("Đồ thị 2 phía");

}

else

{

Console.WriteLine("Không phải Đồ thị 2 phía");

}

}

public bool IsBipartite(int s)

{

foreach (int u in list[s])

{

if (diqua[u] == false)

{

diqua[u] = true;

color[u] = 1 - color[s];

if (!IsBipartite(u))

{

return false;

}

}

else if (color[u] == color[s])

{

return false;

}

}

return true;

}

public void DFSTestChuTrinh()

{

color = new int[n];

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

{

color[i] = 0;

}

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

{

if (DFSCheckChuTrinh(i))

{

Console.WriteLine(" Đồ thị có chu trình");

return;

}

}

Console.WriteLine(" Đồ thị không có chu trình");

}

public bool DFSCheckChuTrinh(int s)

{

color[s] = 1;

foreach (int u in list[s])

{

if (color[u] == 0)

{

DFSCheckChuTrinh(u);

}

else if (color[u] == 1)

{

return true;

}

}

color[s] = 2;

return false;

}

public void TopoSort()

{

diqua = new bool[n];

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

{

diqua[i] = false;

}

Topo = new LinkedList<int>();

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

{

if (diqua[i] == false)

{

TopoCheck(i);

}

}

foreach (int x in Topo)

{

Console.Write(" " + x);

}

}

// Đệ qui DFS

public void TopoCheck(int s)

{

diqua[s] = true;

foreach (int u in list[s])

{

if (diqua[u] == false)

{

TopoCheck(u);

}

}

Topo.AddFirst(s);

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BT6TH

{

internal class Program

{

static void Main(string[] args)

{

//// Câu 1: Kiểm tra đồ thị phân đôi

//AdjList newList = new AdjList();

[//newList.FileToAdjList(@"../../TextFile/AdjList1.txt](mailto://newList.FileToAdjList(@"../../TextFile/AdjList1.txt)");

//newList.Xuat();

//newList.Bipartite();

//Console.ReadKey();

//=======================================================

//Câu 2: Kiểm tra đồ thị có chu trình hay không

//AdjList newList = new AdjList();

[//newList.FileToAdjList(@"../../TextFile/AdjList2.txt](mailto://newList.FileToAdjList(@"../../TextFile/AdjList2.txt)");

//newList.Xuat();

//newList.DFSTestChuTrinh();

//Console.ReadKey();

//=======================================================

AdjList newList = new AdjList();

[newList.FileToAdjList(@"../../TextFile/AdjList3.txt](mailto:newList.FileToAdjList(@"../../TextFile/AdjList3.txt)");

newList.Xuat();

Console.WriteLine("Topo sort: ");

newList.TopoSort();

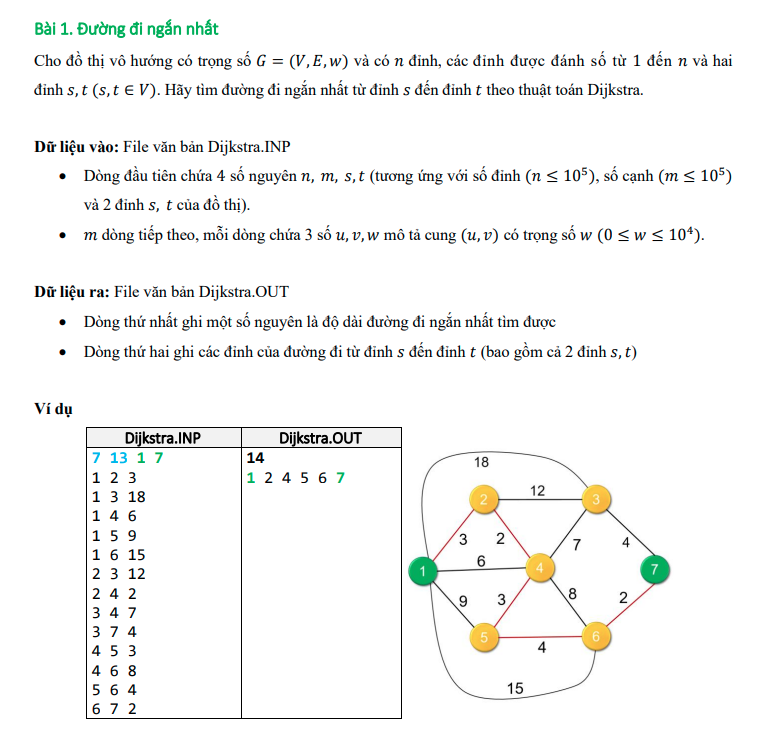
Console.ReadKey();

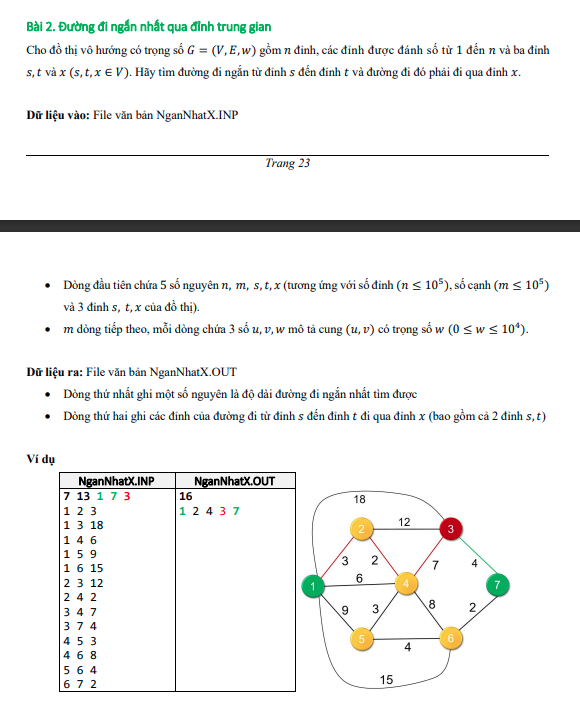
}

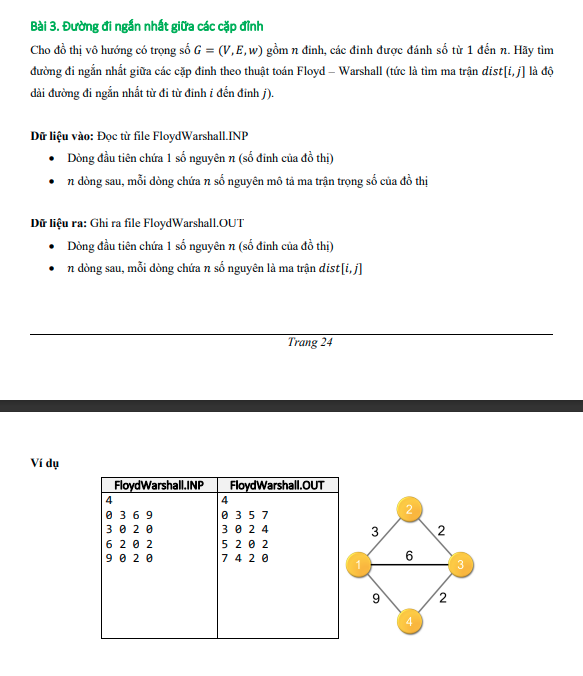
}

}

Buổi 7:







CODE:

WeightMatrix.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BTH7

{

internal class WeightMatrix

{

private int n;

private int[,] array;

int[] pre;

int[] dist;

bool[] visited;

int[,] floyPrevious;

int[,] floyDist;

public int N { get => n; set => n = value; }

public int[,] Array { get => array; set => array = value; }

public int[] Pre { get => pre; set => pre = value; }

public int[] Dist { get => dist; set => dist = value; }

public bool[] Visited { get => visited; set => visited = value; }

public int[,] FloyPrevious { get => floyPrevious; set => floyPrevious = value; }

public int[,] FloyDist { get => floyDist; set => floyDist = value; }

public WeightMatrix() { }

public WeightMatrix(int newWeightMatrix)

{

n = newWeightMatrix;

array = new int[n, n];

}

public void fileToWeightMatrix(string filePath)

{

StreamReader reader = new StreamReader(filePath);

n = int.Parse(reader.ReadLine());

array = new int[n, n];

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

{

string[] s = reader.ReadLine().Split();

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

{

if (i == j)

{

array[i, j] = 0;

}

else

{

array[i, j] = int.Parse(s[j]) > 0 ? int.Parse(s[j]) : int.MaxValue;

}

}

}

reader.Close();

}

public void Output()

{

Console.WriteLine("Đồ thị ma trận kề - số đỉnh : " + n);

Console.WriteLine();

Console.Write(" Đỉnh |");

for (int i = 0; i < n; i++) Console.Write(" {0}", i);

Console.WriteLine(); Console.WriteLine(" " + new string('-', 6 \* n));

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

{

Console.Write(" {0} |", i);

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

if (array[i, j] < int.MaxValue) Console.Write(" {0, 3}", array[i, j]);

else Console.Write(" {0, 3}", "\_");

Console.WriteLine();

}

}

public int findMinIndex()

{

int minIndex = 0;

int min = int.MaxValue;

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

{

if (visited[i] == false && dist[i] < min)

{

min = dist[i];

minIndex = i;

}

}

return minIndex;

}

public void Dijkstra(int s)

{

pre = new int[n];

dist = new int[n];

visited = new bool[n];

// hàm này dùng để xác lập các giá trị ban đầu cho các biến

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

{

dist[i] = int.MaxValue;

visited[i] = false;

pre[i] = s;

}

dist[s] = 0;

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

{

int u = findMinIndex();

visited[u] = true;

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

{

if (!visited[v] && array[u, v] != int.MaxValue)

if (dist[u] != int.MaxValue && dist[u] + array[u, v] < dist[v])

{

dist[v] = dist[u] + array[u, v];

pre[v] = u;

}

}

}

}

public void PrintDijkstra(int s)

{

Console.WriteLine("Shortest path from " + s + " to remaining verteces :");

Console.Write("Vertex : ");

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

{

Console.Write(" " + i);

}

Console.WriteLine();

Console.WriteLine(" " + new string('-', 3 \* n));

Console.Write("previous vertex : ");

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

{

Console.Write("{0, 3}", pre[i]);

}

Console.WriteLine();

Console.Write("distance : ");

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

{

if (dist[i] < int.MaxValue)

{

Console.Write("{0, 3}", dist[i]);

}

else

{

Console.Write("{0, 3}", "x");

}

}

}

public int MinRouteXY(int x, int y)

{

Dijkstra(x);

Stack<int> st = new Stack<int>();

int k = y;

st.Push(k);

while (pre[k] != x)

{

k = pre[k];

st.Push(k);

}

Console.WriteLine();

Console.Write("Shortest path from {0} to {1} : {2} ->", x, y, x);

while (st.Count > 0)

{

k = st.Pop();

Console.Write(" {0} ->", k);

}

Console.WriteLine(" độ dài : " + dist[y]);

return dist[y];

}

public void MinRouteXYZ(int x, int y, int z)

{

int distanceXY = MinRouteXY(x, y);

int distanceYZ = MinRouteXY(y, z);

int distanceXZ = distanceXY + distanceYZ;

Console.WriteLine("Distance from X to Z: " + distanceXZ);

}

public void Floyd()

{

floyDist = new int[n, n];

floyPrevious = new int[n, n];

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

{

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

{

floyDist[i, j] = array[i, j];

floyPrevious[i, j] = i;

}

}

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

{

floyDist[i, i] = 0;

}

for (int k = 0; k < n; k++)

{

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

{

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

{

if ((floyDist[i, j] > floyDist[i, k] + floyDist[k, j]) && floyDist[i, k] < int.MaxValue && floyDist[k, j] < int.MaxValue)

{

floyDist[i, j] = floyDist[i, k] + floyDist[k, j];

floyPrevious[i, j] = floyPrevious[k, j];

}

}

}

}

//Outdp();

}

public void OutDToP()

{

Console.WriteLine("distance : ");

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

{

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

{

Console.Write(" " + floyDist[i, j]);

}

Console.WriteLine();

}

Console.WriteLine();

Console.WriteLine("previous : ");

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

{

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

{

Console.Write(" " + floyPrevious[i, j]);

}

Console.WriteLine();

}

}

public void Floyd\_RouteXY(int x, int y)

{

Stack<int> st = new Stack<int>();

int k = y;

st.Push(k);

while (floyPrevious[x, k] != x)

{

k = floyPrevious[x, k];

st.Push(k);

}

Console.WriteLine();

Console.Write("Shortest path from {0} to {1}: {2}", x, y, x);

while (st.Count > 0)

{

k = st.Pop();

Console.Write(" {0} ->", k);

}

Console.WriteLine("Length : " + floyDist[x, y]);

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BTH7

{

internal class Program

{

static void Main(string[] args)

{

////Câu 1:

//WeightMatrix newMatrix = new WeightMatrix();

[//newMatrix.fileToWeightMatrix(@"../../TextFile/Graph1.txt](mailto://newMatrix.fileToWeightMatrix(@"../../TextFile/Graph1.txt)");

//newMatrix.Output();

//Console.WriteLine();

//Console.Write("Enter starting vertex: ");

//int x = int.Parse(Console.ReadLine());

//Console.Write("Enter ending vertex: ");

//int y = int.Parse(Console.ReadLine());

//newMatrix.MinRouteXY(x, y);

//Console.ReadKey();

////Câu 2:

//WeightMatrix newMatrix = new WeightMatrix();

[//newMatrix.fileToWeightMatrix(@"../../TextFile/Graph1.txt](mailto://newMatrix.fileToWeightMatrix(@"../../TextFile/Graph1.txt)");

//newMatrix.Output();

//Console.WriteLine();

//Console.Write("Enter starting vertex: ");

//int x = int.Parse(Console.ReadLine());

//Console.Write("Enter vertex: ");

//int y = int.Parse(Console.ReadLine());

//Console.Write("Enter ending vertex: ");

//int z = int.Parse(Console.ReadLine());

//newMatrix.MinRouteXYZ(x, y, z);

//Console.ReadKey();

////Câu 3:

//WeightMatrix newMatrix = new WeightMatrix();

[//newMatrix.fileToWeightMatrix(@"../../TextFile/Graph2.txt](mailto://newMatrix.fileToWeightMatrix(@"../../TextFile/Graph2.txt)");

//newMatrix.Output();

//Console.WriteLine();

//newMatrix.Floyd();

//Console.WriteLine("DP matrix: ");

//newMatrix.OutDToP();

//Console.WriteLine();

//Console.WriteLine("In details: ");

//for (int i = 0; i < newMatrix.N - 1; i++)

//{

// for (int j = 0; j < newMatrix.N; j++)

// {

// if (i != j)

// {

// newMatrix.Floyd\_RouteXY(i, j);

// }

// }

//}

//Console.WriteLine();

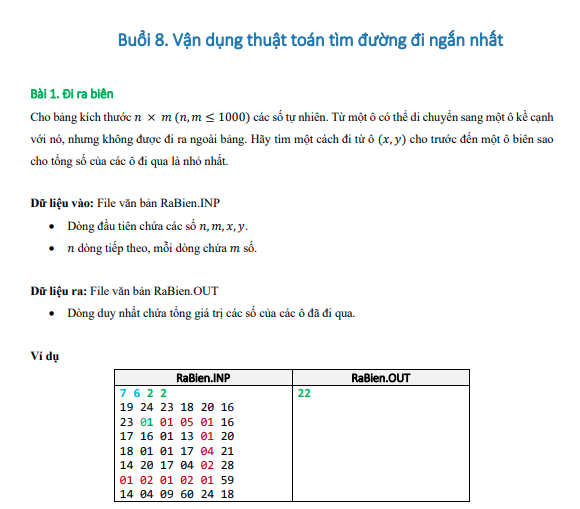
//Console.ReadKey();

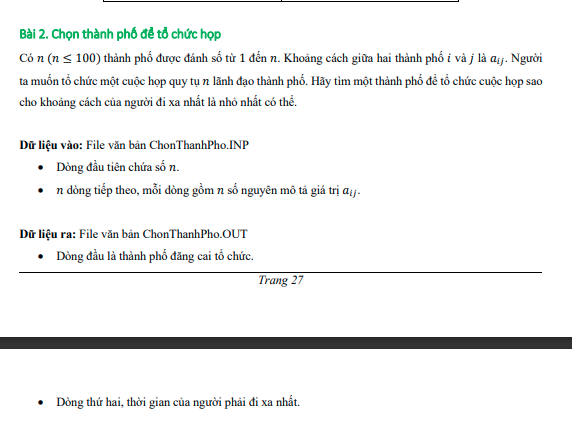
}

}

}

Buổi 8:





CODE:

WeightMatrix.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BTH7

{

internal class WeightMatrix

{

private int n;

private int[,] array;

int[] pre;

int[] dist;

bool[] visited;

int[,] floyPrevious;

int[,] floyDist;

public int N { get => n; set => n = value; }

public int[,] Array { get => array; set => array = value; }

public int[] Pre { get => pre; set => pre = value; }

public int[] Dist { get => dist; set => dist = value; }

public bool[] Visited { get => visited; set => visited = value; }

public int[,] FloyPrevious { get => floyPrevious; set => floyPrevious = value; }

public int[,] FloyDist { get => floyDist; set => floyDist = value; }

public WeightMatrix() { }

public WeightMatrix(int newWeightMatrix)

{

n = newWeightMatrix;

array = new int[n, n];

}

public void fileToWeightMatrix(string filePath)

{

StreamReader reader = new StreamReader(filePath);

n = int.Parse(reader.ReadLine());

array = new int[n, n];

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

{

string[] s = reader.ReadLine().Split();

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

{

if (i == j)

{

array[i, j] = 0;

}

else

{

array[i, j] = int.Parse(s[j]) > 0 ? int.Parse(s[j]) : int.MaxValue;

}

}

}

reader.Close();

}

public void Output()

{

Console.WriteLine("Đồ thị ma trận kề - số đỉnh : " + n);

Console.WriteLine();

Console.Write(" Đỉnh |");

for (int i = 0; i < n; i++) Console.Write(" {0}", i);

Console.WriteLine(); Console.WriteLine(" " + new string('-', 6 \* n));

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

{

Console.Write(" {0} |", i);

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

if (array[i, j] < int.MaxValue) Console.Write(" {0, 3}", array[i, j]);

else Console.Write(" {0, 3}", "\_");

Console.WriteLine();

}

}

public int findMinIndex()

{

int minIndex = 0;

int min = int.MaxValue;

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

{

if (visited[i] == false && dist[i] < min)

{

min = dist[i];

minIndex = i;

}

}

return minIndex;

}

public void Dijkstra(int s)

{

pre = new int[n];

dist = new int[n];

visited = new bool[n];

// hàm này dùng để xác lập các giá trị ban đầu cho các biến

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

{

dist[i] = int.MaxValue;

visited[i] = false;

pre[i] = s;

}

dist[s] = 0;

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

{

int u = findMinIndex();

visited[u] = true;

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

{

if (!visited[v] && array[u, v] != int.MaxValue)

if (dist[u] != int.MaxValue && dist[u] + array[u, v] < dist[v])

{

dist[v] = dist[u] + array[u, v];

pre[v] = u;

}

}

}

}

public void PrintDijkstra(int s)

{

Console.WriteLine("Shortest path from " + s + " to remaining verteces :");

Console.Write("Vertex : ");

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

{

Console.Write(" " + i);

}

Console.WriteLine();

Console.WriteLine(" " + new string('-', 3 \* n));

Console.Write("previous vertex : ");

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

{

Console.Write("{0, 3}", pre[i]);

}

Console.WriteLine();

Console.Write("distance : ");

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

{

if (dist[i] < int.MaxValue)

{

Console.Write("{0, 3}", dist[i]);

}

else

{

Console.Write("{0, 3}", "x");

}

}

}

public int MinRouteXY(int x, int y)

{

Dijkstra(x);

Stack<int> st = new Stack<int>();

int k = y;

st.Push(k);

while (pre[k] != x)

{

k = pre[k];

st.Push(k);

}

Console.WriteLine();

Console.Write("Shortest path from {0} to {1} : {2} ->", x, y, x);

while (st.Count > 0)

{

k = st.Pop();

Console.Write(" {0} ->", k);

}

Console.WriteLine(" độ dài : " + dist[y]);

return dist[y];

}

public void MinRouteXYZ(int x, int y, int z)

{

int distanceXY = MinRouteXY(x, y);

int distanceYZ = MinRouteXY(y, z);

int distanceXZ = distanceXY + distanceYZ;

Console.WriteLine("Distance from X to Z: " + distanceXZ);

}

public void Floyd()

{

floyDist = new int[n, n];

floyPrevious = new int[n, n];

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

{

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

{

floyDist[i, j] = array[i, j];

floyPrevious[i, j] = i;

}

}

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

{

floyDist[i, i] = 0;

}

for (int k = 0; k < n; k++)

{

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

{

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

{

if ((floyDist[i, j] > floyDist[i, k] + floyDist[k, j]) && floyDist[i, k] < int.MaxValue && floyDist[k, j] < int.MaxValue)

{

floyDist[i, j] = floyDist[i, k] + floyDist[k, j];

floyPrevious[i, j] = floyPrevious[k, j];

}

}

}

}

//Outdp();

}

public void OutDToP()

{

Console.WriteLine("distance : ");

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

{

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

{

Console.Write(" " + floyDist[i, j]);

}

Console.WriteLine();

}

Console.WriteLine();

Console.WriteLine("previous : ");

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

{

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

{

Console.Write(" " + floyPrevious[i, j]);

}

Console.WriteLine();

}

}

public void Floyd\_RouteXY(int x, int y)

{

Stack<int> st = new Stack<int>();

int k = y;

st.Push(k);

while (floyPrevious[x, k] != x)

{

k = floyPrevious[x, k];

st.Push(k);

}

Console.WriteLine();

Console.Write("Shortest path from {0} to {1}: {2}", x, y, x);

while (st.Count > 0)

{

k = st.Pop();

Console.Write(" {0} ->", k);

}

Console.WriteLine("Length : " + floyDist[x, y]);

}

public void Output1()

{

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

{

Console.Write(" {0}", i);

}

Console.WriteLine();

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

{

Console.Write("{0,-1}|", i);

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

{

if (array[i, j] < int.MaxValue)

{

Console.Write(" {0,1}", array[i, j]);

}

else

{

Console.Write(" {0,1}", "\_");

}

}

Console.WriteLine();

}

}

}

}

Program.cs

using BTH7;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BTH8

{

internal class Program

{

static void Main(string[] args)

{

//// Bài 1 : Bài toán đi ra biên.

//int[,] mt;

//Console.WriteLine();

//string filePath = @"..\..\TextFile\Matrix1.txt";

//mt = FileToMatrix(filePath);

//Console.WriteLine(" Ma trận a :");

//PrintMatrix(mt);

//WeightMatrix newMatrix = new WeightMatrix();

//newMatrix = MatrixToWeightMatrix(mt);

//Console.WriteLine();

////newMatrix.Output();

//Console.WriteLine(" Enter position s(x,y) :");

//Console.Write(" x = ");

//int x = int.Parse(Console.ReadLine());

//Console.Write(" y = ");

//int y = int.Parse(Console.ReadLine());

//if (x == 0 || y == 0 || x == mt.GetLength(0) - 1 || y == mt.GetLength(1) - 1)

//{

// Console.WriteLine(" ({0},{1}) là đỉnh nằm trên biên.", x, y);

//}

//else

//{

// int s = x \* mt.GetLength(1) + y;

// newMatrix.Dijkstra(s);

// ToBorderline(newMatrix, s);

//}

//Console.ReadKey();

//===============================================================

//Câu 2:

//WeightMatrix newMatrix = new WeightMatrix();

[//newMatrix.fileToWeightMatrix(@"../../TextFile/City.txt](mailto://newMatrix.fileToWeightMatrix(@"../../TextFile/City.txt)");

//newMatrix.Output();

//Console.WriteLine();

//int city = SelectCity(newMatrix);

//Console.WriteLine("Chosen city: {0}", city);

//Console.ReadKey();

}

static int[,] FileToMatrix(string fileName)

{

int[,] mt;

// Doc file Matran.txt -> Ma tran a

StreamReader sr = new StreamReader(fileName);

string[] s = sr.ReadLine().Split();

int row = int.Parse(s[0]);

int col = int.Parse(s[1]);

mt = new int[row, col];

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

{

s = sr.ReadLine().Split();

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

mt[i, j] = int.Parse(s[j]);

}

sr.Close();

return mt;

}

static void PrintMatrix(int[,] a)

{

Console.WriteLine(" 0 1 2 3 4 5");

Console.WriteLine(" -------------------------");

for (int i = 0; i < a.GetLength(0); i++)

{

Console.Write(" " + i + "|");

for (int j = 0; j < a.GetLength(1); j++)

Console.Write("{0, 4}", a[i, j]);

Console.WriteLine(" |");

Console.WriteLine(" |");

}

Console.WriteLine(" -------------------------");

}

static WeightMatrix MatrixToWeightMatrix(int[,] mt)

{

WeightMatrix g = new WeightMatrix();

int row = mt.GetLength(0);

int col = mt.GetLength(1);

g.N = row \* col;

g.Array = new int[g.N, g.N];

for (int i = 0; i < g.N; i++)

for (int j = 0; j < g.N; j++)

if (i == j) g.Array[i, j] = 0;

else g.Array[i, j] = int.MaxValue;

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

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

{

int k = i \* col + j;

if (i > 0)

g.Array[k, k - col] = mt[i - 1, j];

if (j > 0)

g.Array[k, k - 1] = mt[i, j - 1];

if (j < col - 1)

g.Array[k, k + 1] = mt[i, j + 1];

if (i < row - 1)

g.Array[k, k + col] = mt[i + 1, j];

}

// trả về g

return g;

}

static void ToBorderline(WeightMatrix g, int s)

{

Stack<int> st = new Stack<int>();

int bien = 0;

int min = int.MaxValue;

for (int i = 0; i < g.N; i++)

if (DegV(i, g) < 4)

if (g.Dist[i] < min)

{

min = g.Dist[i];

bien = i;

}

Console.WriteLine();

while (bien != s)

{

st.Push(bien);

bien = g.Pre[bien];

}

Console.Write("Đường ra biên gần nhất, xuất phát từ đỉnh s({0}) : {1}", s, s);

Console.ForegroundColor = ConsoleColor.Cyan;

while (st.Count > 0) Console.Write(" -> " + st.Pop());

Console.WriteLine(" Có độ dài = " + min);

Console.ForegroundColor = ConsoleColor.White;

}

static int DegV(int v, WeightMatrix g)

{

int deg = 0;

for (int i = 0; i < g.N; i++)

if (g.Array[v, i] < int.MaxValue && g.Array[v, i] != 0) deg++;

return deg;

} // dùng trong hàm ToBorderLine

static int SelectCity(WeightMatrix g)

{

// Chạy thuật toán Floyd -> ma trận d

g.Floyd();

// Biến cty : lưu thành phố được chọn, khởi tạo = -1

int cty = -1;

int min = int.MaxValue;

for (int i = 0; i < g.N; i++)

{

int max = -int.MaxValue;

for (int j = 0; j < g.N; j++)

if (g.FloyDist[i, j] > max) max = g.FloyDist[i, j];

if (max < min)

{

cty = i;

min = max;

}

}

return cty;

}

}

}