|  |  |
| --- | --- |
| Buổi 2 | **VẬN DỤNG CÁC THAO TÁC CƠ BẢN TRÊN ĐỒ THỊ** |

**Bài 1. Chuyển danh sách cạnh sang danh sách kề**

Cho đồ thị vô hướng G = (V, E) có n (n ≤ 105) đỉnh, các đỉnh được đánh số từ 1 đến n và m cạnh (m ≤ 105) được lưu trong file văn bản dưới dạng danh sách cạnh. Hãy viết chương trình chuyển đổi đồ thị này sang danh sách kề.

Yêu cầu :

|  |  |  |
| --- | --- | --- |
| Đồ thị | Input : EdgeList.txt | Output : AdjList.txt |
|  | **5** **6**  0 1  0 2  1 2  1 4  2 4  3 4 | 5  1 2  0 2 4  0 1 4  4  1 2 3 |

***Hướng dẫn*** :

1/. Tạo project Buoi02, thư mục TextFile và tập tin EdgeList.txt với nội dung như trên.

2/. Add vào project 3 class : AdjList.cs, EdgeList.cs và Menu.cs. Lưu ý sửa lại các namespace :

namespace Buoi02

3/. Mô hình giải quyết bài toán :

AdjList.txt

Đồ thị AdjList

Đồ thị EdgeList

EdgeList.txt

4/. Trong Program.cs, tổ chức giao diện như sau :

using System; using System.IO; using System.Text;

using System.Collections.Generic; // Thư viện cho đối tượng LinkedList

namespace Buoi02

{

class Program

{

static void Main(string[] args)

{

// Xuất text theo Unicode (có dấu tiếng Việt)

Console.OutputEncoding = Encoding.Unicode;

// Nhập text theo Unicode (có dấu tiếng Việt)

Console.InputEncoding = Encoding.Unicode;

/\* Tạo menu \*/

Menu menu = new Menu();

string title = "VẬN DỤNG CÁC THAO TÁC CƠ BẢN TRÊN ĐỒ THỊ"; // Tiêu đề menu

// Danh sách các mục chọn

string[] ms = { "1. Bài 1: Chuyển danh sách cạnh sang danh sách kề",

"2. Bài 2: Chuyển danh sách kề sang danh sách cạnh",

"3. Bài 3: Đỉnh Bồn chứa",

"4. Bài 4: Đồ thị chuyển vị",

"5. Bài 5: Độ dài trung bình của cạnh",

"0. Thoát" };

int chon;

do

{

// Xuất menu

menu.ShowMenu(title, ms);

Console.Write(" Chọn : ");

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

switch (chon)

{

case 1:

{ // Bài 1: Chuyển danh sách cạnh sang danh sách kề

// Tạo đường dẫn fileInput -> đồ thị EdgeList

string fileInput = "../../../TextFile/EdgeList.txt";

EdgeList ge = new EdgeList();

ge.FileToEdgeList(fileInput); ge.Output();

// Tạo đồ thị AdjList ga từ EdgeList ge

AdjList ga = new AdjList();

ga = EdgeListToAdjList(ge); ga.Output();

// Tạo đường dẫn fileOutput

string fileOutput = "../../../TextFile/AdjList.txt";

ga.AdjListToFile(fileOutput);

break;

}

}

Console.WriteLine(" Nhấn một phím bất kỳ");

Console.ReadKey();

Console.Clear();

} while (chon != 0);

}

// Chuyển file EdgeList sang đồ thị danh sách kề AdjList

static AdjList EdgeListToAdjList(EdgeList ge)

{

}

}

}

5/. Trong Program.cs, viết hàm EdgeListToAdjList(EdgeList ge)

// Chuyển file EdgeList sang đồ thị danh sách kề AdjList

static AdjList EdgeListToAdjList(EdgeList ge)

{

// Khởi tạo đồ thị AdjList

AdjList ga = new AdjList();

// Xác định số đỉnh n của đồ thị : ga.N = ge.N;

// Khởi tạo array v[] của đồ thị AdjList

// Khởi tạo các danh sách liên kết ga.V[i]

// Duyệt từng đỉnh i trong ga , i = 0..ga.N-1

// Khởi tạo các dslk : ga.V[i] = new LinkedList<int>();

// Xây dựng các phấn tử cho các dslk

// Duyệt từng cạnh e trong đồ thị EdgeList

{

// AddLast e.Item2 vào ga.V[e.Item1]

// AddLast e.Item1 vào ga.V[e.Item2]

}

return ga; // Đồ thị trả về

}

6/ Trong class AdjList.cs viết phương thức xuất đồ thị ra file

// Xuất đồ thị ra file fileOutput

public void AdjListToFile(string fileOutput)

{

// Khởi tạo : StreamWriter sw = new StreamWriter(fileOutput);

// Ghi vào file số đỉnh n

// Duyệt từng đỉnh i, i = 0..n-1

{

// Gọi chuổi s : string s = "";

// Duyệt các đỉnh liên kết trong v[i]

foreach (int x in v[i])

s = s + x + " ";

// Cắt khoảng trắng 2 đầu chuổi s

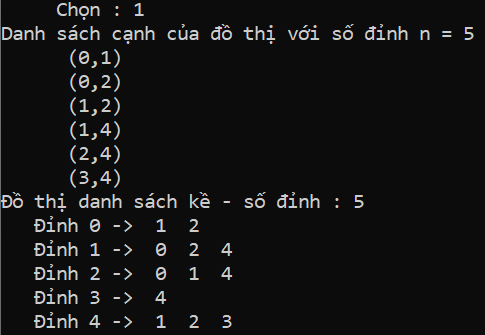
// Ghi s vào file

}

sw.Close();

}

Kết quả thực thi:



**Bài 2. Chuyển danh sách kề sang danh sách cạnh**

Cho đồ thị vô hướng G = (V, E) có n (n ≤ 105) đỉnh, các đỉnh được đánh số từ 1 đến n và m cạnh (m ≤ 105) được lưu trong file văn bản dưới dạng danh sách kề. Hãy viết chương trình chuyển đổi đồ thị này sang danh sách cạnh.

Yêu cầu :

|  |  |  |
| --- | --- | --- |
| Đồ thị | intput : AdjList.txt | Output : EdgeList.txt |
|  | 5  1 2  0 2 4  0 1 4  4  1 2 3 | **5** **6**  0 1  0 2  1 2  1 4  2 4  3 4 |

Bài 2 là bài toán ngược của bài 1 : bài tập về nhà

**Bài 3. Bồn chứa**

Cho đồ thị có hướng G = (V, E) có n (n ≤ 1000) đỉnh, các đỉnh được đánh số từ 1 đến n. “Bồn chứa”

trong đồ thị G là đỉnh chỉ có cung vào mà không có cung ra. Viết chương trình tìm các đỉnh bồn chứa trong đồ thị G.

yêu cầu :

|  |  |  |
| --- | --- | --- |
| Đồ thị | Input : DirectedMatrix.txt | Output : Bonchua.txt |
|  | 6  0 1 1 0 0 0  0 0 0 1 1 0  0 1 0 0 1 0  0 0 0 0 1 1  0 1 0 0 0 1  0 0 0 0 0 0 | Đỉnh 5 là đỉnh bồn chứa  Số đỉnh là bồn chứa : 1  (và xuất lên màn hình) |

***Hướng dẫn*** :

1/. Add class AdjMatrix.cs vào project. Trong TextFile, tạo file DirectedMatrix.txt với nội dung như trên.

2/. Trong class AdjMatrix.cs, viết phương thức kiểm tra đỉnh i là đỉnh bồn chứa hay không?

// Xác định đỉnh i là đỉnh bồn chứa hay không ?

public bool IsStorage(int i)

{

// Duyệt các cột j của ma trận a, j = 0..n-1

// Nếu tồn tại a[i, j] = 1 thì i không phải đỉnh bồn chứa ( return false)

// Kết thúc vòng lặp : return true (i là đỉnh bồn chứa)

}

3/. Trình bày case 3 trong Program.cs

case 3:

{

// Bài 3 : Bồn chứa

// Khởi tạo g là đồ thị ma trận kề : AdjMatrix g

// Tạo đường dẫn fileInput : DirectedMatrix.txt

// Tạo đồ thị g và xuất đồ thị g lên màn hình

// Tạo đường dẫn fileOutput : Storage.txt

// Gọi hàm : Storage(g, fileOutput);

break;

}

4/. Trong Program.cs, viết hàm :

static void Storage(AdjMatrix g, string fileOut)

{

// Khởi tạo : StreamWriter sw

// Khai báo biến đếm : count=0

// Duyệt các đỉnh i của g

{

// Nếu (g.IsStorage(i) == true)

{

// đếm count lên 1

// Xuất lên màn hình : ("Đỉnh " + i + " là đỉnh bồn chứa");

// Ghi file sw : ("Đỉnh " + i + " là đỉnh bồn chứa");

}

}

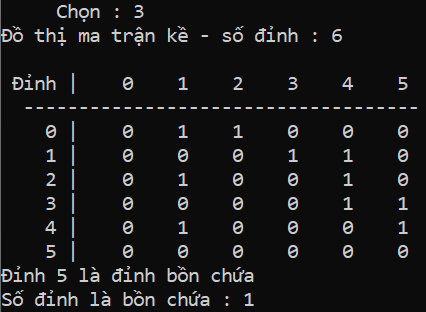
// Ghi file sw : ("Số đỉnh là bồn chứa : " + count);

// Xuất lên màn hình : ("Số đỉnh là bồn chứa : " + count);

// Đóng file sw

}

Kết quả khi thực thi :



**Bài 4. Đồ thị chuyển vị**

Cho đồ thị có hướng G = (V, E) có n (n ≤ 1000) đỉnh, các đỉnh được đánh số từ 1 đến n. Đồ thị chuyển vị của đồ thị G là đồ thị GT = (V, ET). Trong đó :

ET = {(u, v) ∶ (v, u) ∈ E}

Hãy xây dựng GT từ G.

Tìm đồ thị chuyển vị của đồ thị có hướng G được biểu diễn bằng danh sách kề.

Đồ thị chuyển vị của một [đồ thị có hướng](https://wivi.wiki/wiki/Directed_graph)*G* là một đồ thị có hướng trên cùng một tập các đỉnh với tất cả các cạnh đảo ngược so với định hướng của các cạnh tương ứng trong *G* . Nghĩa là, nếu *G* chứa một cạnh *(u, v)* thì phép đảo / chuyển vị / đảo ngược của *G* chứa một cạnh *(v, u)* và ngược lại.

Yêu cầu :

|  |  |  |
| --- | --- | --- |
| Đồ thị | Input : DirectedList.txt | Output : chuyenvi.txt |
|  | 5  2  0 4  1  2 3 | 5  1  2  0 4  4  1 |

***Hướng dẫn*** :

1/. Trong TextFile tạo file DirectedList.txt với nội dung như trên

2/. Trong Program.cs, trình bày case 4 :

case 4:

{

// Bài 4 : Đồ thị chuyển vị

// Tạo đường dẫn file Input : "../../../TextFile/DirectedList.txt";

// Khai báo đồ thị g : AdjList g = new AdjList();

// Tạo đồ thị từ fileInput và xuất đồ thị

// Khai báo G là đồ thị chuyển vị : AdjList G = new AdjList();

// Gọi hàm : G = TransposeG(g); G.Output();

// Xuất đồ thi chuyển vị G lên màn hình

// SV tự làm ghi kết quả vào file Transpose.txt

break;

}

3/. Trong Program.cs, viết hàm

// Bài 4 : Đồ thị chuyển vị, nhận vào đồ thị g, trả về đồ thị chuyển vị G

static AdjList TransposeG(AdjList g)

{

// Khai báo đồ thị G : AdjList G

// Xác định số đỉnh G.N là số đỉnh g.N

// Cấp phát vùng nhớ cho G.V : new LinkedList<int>[G.N];

// Khởi tạo các dslk G.V[i] = new LinkedList<int>() , i = 0..G.N-1

// Duyệt từng đỉnh i của G

{

// Duyệt từng đỉnh x trong G.V[i]

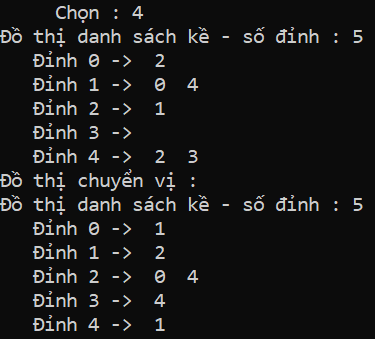
// AddLast i vào G.V[x] : G.V[x].AddLast(i);

}

// Trả về G

}

Kết quả khi thực thi :



**Bai 5. Độ dài trung bình của cạnh**

Cho đồ thị vô hướng có trọng số G = (V, E) có n (n ≤ 105) đỉnh, các đỉnh được đánh số từ 1 đến n, và m (m ≤ 105). Tìm các cạnh có độ dài dài nhất và tính độ dài trung bình của các cạnh.

Yêu cầu :

|  |  |  |
| --- | --- | --- |
| Đồ thị | Input : WeightEdgeList.txt | Output : Trungbinhcanh.txt |
|  | 5 6  1 2 7  1 3 9  2 3 4  2 5 10  3 5 10  4 5 8 | Số lượng cạnh dài nhất : 2  (2, 5) = 10  (3, 5) = 10  Chiều dài TB cạnh : 8.00 |

***Hướng dẫn*** :

1/. Trong TextFile tạo file WeightEdgeList.txtvới nội dung như trên

2/. Add class WeightEdgeList.cs vào project

3/. Trong program.cs viết các hàm :

// Độ dài nhất và các cạnh có độ dài lớn nhất

static void MaxEdge(WeightEdgeList g)

{

int max = -int.MaxValue;

// Dùng một dslk lst dùng chứa các cạnh dài nhất

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

// Tìm độ dài nhất max, trong g.G

. . .

// Tìm các cạnh dài nhất (item3 = max) và Add vào lst

. . .

// Xuất các cạnh dài nhất & số lượng

. . .

}

// Trung bình cạnh

static double AverageEdge(WeightEdgeList g)

{

. . .

return avg;

}

4/. Trong Program.cs, trình bày case 5 :

case 5:

{

// Tạo tham số fileInput

string fileInput = "../../../TextFile/WeightEdgeList.txt";

WeightEdgeList g = new WeightEdgeList();

g.FileToWeightEdgeList(fileInput); g.Output();

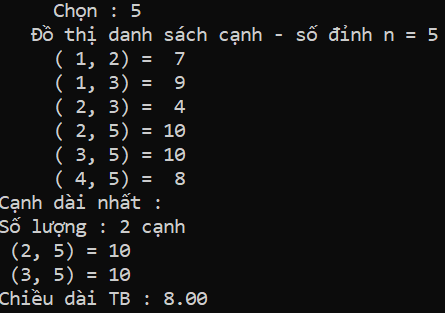
Console.WriteLine("Cạnh dài nhất :"); MaxEdge(g);

Console.WriteLine("Chiều dài TB : {0:0.00}", AverageEdge(g));

break;

}

Kết quả thực thi



**Bài tập làm thêm**

**Bài 1. Chuyển ma trận kề sang danh sách kề**

Cho đồ thị vô hướng G = (V, E) có n (n ≤ 1000) đỉnh, các đỉnh được đánh số từ 1 đến n được lưu

trong file văn bản dưới dạng ma trận kề. Hãy viết chương trình chuyển đổi đồ thị này sang danh sách kề.

**Bài 2. Chuyển ma trận kề sang danh sách cạnh**

Cho đồ thị vô hướng G = (V, E) có n (n ≤ 1000) đỉnh, các đỉnh được đánh số từ 1 đến n được lưu

trong file văn bản dưới dạng ma trận kề. Hãy viết chương trình chuyển đổi đồ thị này sang danh sách

cạnh.

Viết các hàm tiện ích cho phép chuyển đổi qua lại giữa các cách biểu diễn đồ thị khác nhau

 Chuyển từ Danh sách kề sang ma trận kề.

 Chuyển từ Danh sách cạnh sang ma trận kề.