|  |  |
| --- | --- |
| Buổi 3 | **TÌM KIẾM TRÊN ĐỒ THỊ BẰNG THUẬT TOÁN**  **Breadth First Search – BFS** |

1. **Duyệt đồ thị bằng thuật toán BFS**

**1.1** *Thuật toán (sử dụng Queue)*

1. Lấy s là một đỉnh của đồ thị
2. s 🡪 Queue
3. Lặp : nếu Queue chưa rỗng

* Lấy u 🡨 Queue
* Duyệt đỉnh u
* Đặt các đỉnh kề của u chưa được xét vào Queue

**1.2** *Cài đặt thuật toán*

* Sử dụng một array (bool) visite[] để đánh dấu các đỉnh đã xét

Đỉnh i đã xét 🡪 visited[i] = true, ngược lại 🡪 visited[i] = false

* Sử dụng một cấu trúc Queue. Gọi s là đỉnh bắt đầu

BFS(s)

{

Khởi tạo và gán giá trị ban đầu : visited[i] = false , ∀i = 0..< n

Đặt Queue = ∅ ;

s 🡪 Queue ; // bỏ s vào Queue

visited[s] = true ; // đánh dấu đã xét đỉnh s

while( Queue ≠ ∅)

{

s 🡨 Queue ; // lấy s từ trong Queue

Duyệt đỉnh s ;

Duyệt( u ∈ Kề(s))

{

if( đã xét đỉnh u : visited[u] = true) continue;

u 🡪 Queue ;

đánh dấu đã xét u : visited[u] = true ;

}

}

}

Duyệt đỉnh s là một tác vụ nào đó đối với s, ví dị như : xuất lên màn hình / vào file, tính toán, v.v…

1. **Bài tập áp dụng**

**Bài 1** : Liệt kê các đỉnh liên thông với đỉnh x bằng thuật toán BFS

|  |  |  |
| --- | --- | --- |
| Đồ thị | Input : AdjList1.txt | Output : xuất lên màn hình |
|  | 8  1  0 4  3  2 6  1  3 7  7 | Xuất đồ thị DS kề  (giả sử nhập x = 2)  Các đỉnh liên thông với 2 :  2 – 3 – 6 – 7 |

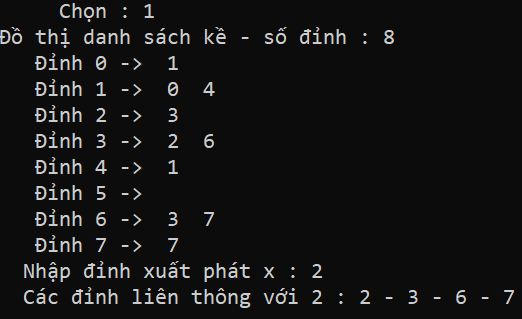
*Hướng dẫn* :

* Trong TextFile tạo file AdjList1.txt với nội dung như trên
* Add file AdjList.cs vào Project. Và để đơn giản cho việc cài đặt, thêm vào class thành phần sau :

**bool[] visited; // Dùng đánh dấu đỉnh đã đi qua**

* Viết thêm phương thức BFS(int s) : cài đặt theo thuật toán trên với phép duyệt là xuất lên màn hình

Kết quả thực thi:



**Bài 2** : Tìm đườnh đi từ đỉnh x đến đỉnh y bằng thuật toán BFS

|  |  |  |
| --- | --- | --- |
| Đồ thị | Input : AdjList2.txt | Output : xuất lên màn hình |
|  | 8  1  0 4  3  2 5 6  1  3  3 7  7 | Xuất đồ thị DS kề  (giả sử nhập x = 2, y = 7)  Đường đi từ 2 🡪 7 : 2 🡪 3 🡪 6 🡪 7 |

*Hướng dẫn* :

* Trong TextFile tạo file AdjList2.txt với nội dung như trên
* Trong class AdjList.cs viết phương thức : public void BFS\_XtoY(int x, int y) dựa theo giải thuật BFS, trong đó có sử dụng thêm array : int[] pre = new int[v.Length];  
  với pre[u] = v là đỉnh nằm ngay trước u trên đường đi từ x 🡪 y
* Giải thuật :

public void BFS\_XtoY(int x, int y)

{

Khởi tạo pre[] và gán trị ban đầu : pre[i] = -1 , ∀i = 0..<n

Khởi tạo visited[] và gán giá trị : visited[i] = false , ∀i = 0..<n

Khởi tạo queue q : Queue<int> q = new Queue<int>();

Đánh dấu duyệt x : visited[x] = true;

Bỏ x vào q : x 🡪 q;

while( q ≠ ∅)

{

Gọi s 🡨 q;

Duyệt(u ∈ Kề(s)) (int u in v[s])

{

Nếu (đã xét đỉnh u : visited[u]) continue;

Đánh dấu đã xét u : visited[u] = true;

u 🡪 q;

Đặt đỉnh s trước u : pre[u] = s;

}

}

// Xuất đường đi từ x đến y

Gọi k : int k = y;

Sử dụng một Stack stk : Stack<int> stk = new Stack<int>();

Lặp khi (pre[k] != -1)

{

Bỏ k 🡪 stk (stk.Push(k);)

Đặt lại giá trị k : k = pre[k];

}

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

Lặp khi stk ≠ ∅ (stk.Count > 0)

{

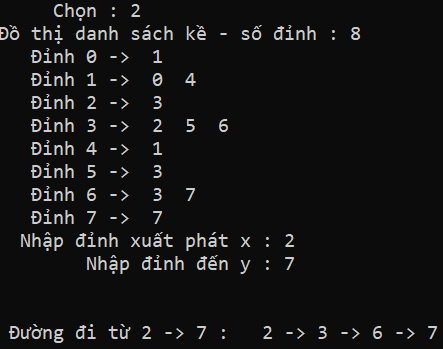
k 🡨 stk (k = stk.Pop();)

Xuất : Console.Write(" -> " + k);

}

}

Kết quả thực thi



**Bài 3** : Xét tính liên thông của đồ thị ? tính số thành phần liên thông (TPLT) của đồ thị và xuất các thành phần liên thông lên màn hình.

|  |  |  |
| --- | --- | --- |
| Đồ thị | Input : AdjList2.txt | Output : xuất lên màn hình |
|  | 8  1  0 4  3  2 5 6  1  3  3 7  7 | Đồ thị có 2 thành phần liên thông  TPLT 1 : 0 1 4  TPLT 2 : 2 3 5 6 7 |

*Hướng dẫn* :

* Trong class AdjList.cs, để đơn giản cho cài đặt 🡪 thêm 2 thành phần như sau :

**int[] index; // Dùng đánh dấu các TPLT**

**int inconnect; // Dùng đếm số TPLT, và thêm propeties**

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

* Biến inconnect để lưu số TPLT, ban đầu inconnect = 0 và sẽ tăng 1 khi gọi hàm duyệt BFS
* index[v] : lưu chỉ số thành phần liên thông chứa đỉnh v, mỗi khi gọi hàm duyệt BFS(v) thì chỉ số thành phần liên thông hiện tại đang chứa v 🡪 index[v] = inconnect
* Trong class AdjList.cs, viết phương thức : public void Connected() theo giải thuật sau :

// Xét tính liên thông và xác định giá trị cho visite[], index[]

// Xác định inconnect : số thành phần liên thông (TPLT)

public void Connected()

{

// inconnect : số TPLT 🡪 giá trị ban đầu = 0

// index : lưu các đỉnh cùng một TPLT, khởi tạ index[] n phần tử

// Khởi gán index[i] = -1, Vi = 0 .. < n

// Khởi tạo và giá trị ban đầu cho visited[i] = false, Vi = 0 .. < n

// Duyệt từng đỉnh i

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

// Nếu chưa duyệt đỉnh i (visited[i] == false)

{

// Khởi đầu cho một TPLT mới -> tăng inconnect++

// Tìm và đánh dấu các đỉnh cùng TPLT, gọi hàm

BFS\_Connected(i);

}

Console.WriteLine();

}

* Trong class AdjList.cs, viết phương thức : public void BFS\_Connected(int s) để duyệt một thành phần liên thông với đỉnh bắt đầu s theo giải thuật sau :

// Lượt duyệt mới vớt đỉnh bắt đầu: s

public void BFS\_Connected(int s)

{

// Sử dụng một queue cho giải thuật

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

// Duyệt đỉnh s (visited[s] = true)

// Đưa s vào q

// Lặp khi queue q còn phần tử

{

// Lấy từ queue q ra một phần tử -> s

// gán giá trị TPLT : index[s] = inconnect;

// Duyệt các đỉnh kề u của s (int u in v[s])

{

// Nếu u chưa duyệt (visited[u] == false)

{

// Duyệt u : visited[u] = true;

// Đưa u vào Queue q

}

}

}

}

* Trong class AdjList.cs viết phương thức : public void OutConnected() để xuất các TPLT ra màn hình :

// Xuất các thành phần liên thông

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)

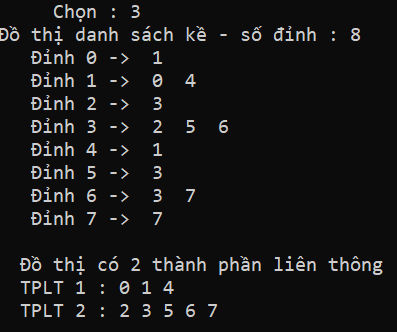
Console.Write(j + " ");

Console.WriteLine();

}

}

Kết quả thực thi



**Code :**

**Program.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace Buoi0304

{

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 = "TÌM KIẾM TRÊN ĐỒ THỊ BẰNG THUẬT TOÁN BFS (Breadth First Search)"; string[] ms = { "1. Bài 1 : Liệt kê các đỉnh liên thông với đỉnh x bằng thuật toán BFS",

"2. Bài 2 : Tìm đường đi từ đỉnh x -> y",

"3. Bài 3 : Xét tính liên thông. Số TPLT, xuất các TPLT",

"0. Thoát" };

int chon;

do

{

Console.Clear();

menu.ShowMenu(title, ms);

Console.Write(" Chọn : ");

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

switch (chon)

{

case 1:

{ // Bài 1 : duyệt đồ thị từ đỉnh x theo BFS

// Tạo đường dẫn file filePath = "../../../TextFile/AdjList1.txt";

// Khởi tạo đồ thị g : AdjList g = new AdjList();

// Đọc file ra đồ thị g; Xuất đồ thị lên màn hình

Console.Write(" Nhập đỉnh xuất phát x : ");

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

Console.Write(" Các đỉnh liên thông với {0} : ", x);

// Gọi phương thức BFS(x);

break;

}

case 2:

{ // Bài 2 : Tìm đường đi từ đỉnh x -> y

// Tạo đường dẫn file filePath = "../../../TextFile/AdjList2.txt";

// Khởi tạo đồ thị g : AdjList g = new AdjList();

// Đọc file ra đồ thị g; Xuất đồ thị lên màn hình

Console.Write(" Nhập đỉnh xuất phát x : ");

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

Console.Write(" Nhập đỉnh đến y : ");

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

// Gọi phương thức BFS\_XtoY(x, y);

break;

}

case 3:

{ // Bài 3 : Xét tính liên thông. Số TPLT, xuất các TPLT

// Tạo đường dẫn file filePath = "../../../TextFile/AdjList2.txt";

// Khởi tạo đồ thị g : AdjList g = new AdjList();

// Đọc file ra đồ thị g; Xuất đồ thị lên màn hình

g.Connected();

if (g.Inconnect == 1)

Console.WriteLine(" Đồ thị liên thông");

else

{

Console.WriteLine(" Đồ thị có {0} thành phần liên thông", g.Inconnect);

g.OutConnected(); // Xuất các TPLT

}

break;

}

}

Console.ReadKey();

} while (chon != 0);

}

**AdjList.cs**

using System;

using System.IO;

using System.Collections.Generic;

using System.Text;

namespace Buoi0304

{

class AdjList

{

LinkedList<int>[] v;

int n; // Số đỉnh

// Để đơn giản : thêm các thành phần chỉ tham gia vào giải thuật

bool[] visited; // Dùng đánh dấu đỉnh đã đi qua

int[] index; // Dùng đánh dấu các TPLT

int inconnect; // Dùng đếm số TPLT

//Propeties

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

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

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

// Contructor

public AdjList() { }

public AdjList(int k) // Khởi tạo v có k đỉnh

...

// Đọc file AdjList.txt --> danh sách kề v

public void FileToAdjList(string filePath)

...

// Xuất đồ thị

public void Output()

...

// Bài 1 : Duyệt đồ thị theo BFS với đỉnh xuất phát s

public void BFS(int s)

{

// Khởi tạo visited[]

visited = new bool[v.Length];

// Khởi gáng visited[i] = false (i=0.. <visited.Lenght)

// Khai báo Queue : Queue<int> q = new Queue<int>();

// Đánh dấu duyệt s (visited[s]= true)

// Đưa s vào queue :s -> q

// Lặp khi q còn chưa rỗng

{

// Lấy trong q ra phần tử s : s <- q

// Duyệt s, xuất s lên màn hình (Console.Write(s + " - ");)

// Xét các đỉnh kề u của s : (foreach (int u in v[s]))

{

// Nếu đã duyệt u (visited[u]=true) -> bỏ qua : continue;

// Đánh dấu duyệt u (visited[u]=true)

// Đưa u vào q : u -> q

}

}

}

// Bài 2 : Tìm đường đi từ đỉnh x đến y theo BFS

public void BFS\_XtoY(int x, int y)

{

// pre[] : lưu đỉnh nằm trước trên đường đi

// Khởi tạo pre[] với v.Lenght phần tử

// Gán các giá trị pre[i] = -1 với i = 0 ... <v.Lenght

// khởi tạo và gán các giá trị ban đầu cho visited[]

// Khai báo một queue<int> q

// Đánh dấu đã duyệt x

// Đưa x vào q

// Lặp khi q chưa rỗng

{

// Lấy trong q ra đỉnh s : int s = q.Dequeue();

// Duyệt các đỉnh kề u của s : (foreach (int u in v[s]))

{

// Nếu đã duyệt u (visited[u]=true) thì bỏ qua (continue)

// Đánh dấu đã duyệt u

// Đưa u vào q

// Đặt s là đỉnh trước u : pre[u] = s;

}

}

// Xuất đường đi từ x đến y

Console.WriteLine();

int k = y;

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

while (pre[k] != -1)

{

stk.Push(k);

k = pre[k];

}

Console.WriteLine();

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

while (stk.Count > 0)

{

k = stk.Pop();

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

}

Console.WriteLine();

}

// Xét tính liên thông và xác định giá trị cho visite[], index[]

// Xác định inconnect : số thành phần liên thông (TPLT)

public void Connected()

{

// inconnect : số TPLT 🡪 giá trị ban đầu = 0

// index : lưu các đỉnh cùng một TPLT, khởi tạo index[] n phần tử

// Khởi gán index[i] = -1, i = 0 .. < n

// Khởi tạo và giá trị ban đầu cho visited[i] = false, Vi = 0 .. < n

// Duyệt từng đỉnh i

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

// Nếu chưa duyệt đỉnh i (visited[i] == false)

{

// Khởi đầu cho một TPLT mới -> tăng inconnect++

// Tìm và đánh dấu các đỉnh cùng TPLT, gọi hàm

BFS\_Connected(i);

}

Console.WriteLine();

}

// Lượt duyệt mới vớt đỉnh bắt đầu: s

public void BFS\_Connected(int s)

{

// Sử dụng một queue cho giải thuật

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

// Duyệt đỉnh s (visited[s] = true)

// Đưa s vào q

// Lặp khi queue q còn phần tử

{

// Lấy từ queue q ra một phần tử -> s

// gán giá trị TPLT : index[s] = inconnect;

// Duyệt các đỉnh kề u của s (int u in v[s])

{

// Nếu u chưa duyệt (visited[u] == false)

{

// Duyệt u : visited[u] = true;

// Đưa u vào Queue q

}

}

}

}

// Xuất các thành phần liên thông

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();

}

}

}

}