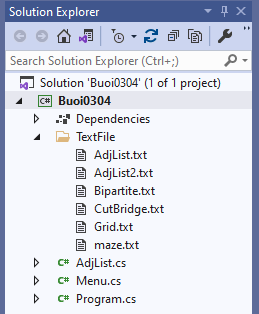
|  |  |
| --- | --- |
| Bài giải  Buổi 03 | **TÌM KIẾM TRÊN ĐỒ THỊ BẰNG THUẬT TOÁN**  **Breadth First Search – BFS** |
| Buổi 04 | **Ứng dụng thuật toán Breadth First Search – BFS** |

1. **Cấu trú chương trình**



1. **Nội dung các file văn bản (txt)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AdjList.txt | AdjList2.txt | Bipartite.txt | CutBridge.txt | Grid.txt |  |
| 8  1  0 4  3  2 6  1  3 7  7 | 8  1  0 4  3  2 5 6  1  3  3 7  7 | 6  1 3 5  0 2 4  1 3 5  0 2 4  1 3 5  0 2 4 | 8  1 4  0 4  3  2 6  0 1  3 7  6 | 6 8  2 1 4 6  0 1 1 1 1 0 0 0  0 1 0 0 1 0 0 0  0 1 1 0 1 0 1 0  0 0 1 1 1 1 1 0  0 0 0 0 0 0 1 0  0 0 0 0 0 0 0 0 | 20 20  1 0 18 19  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0  0 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0  0 0 1 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 0  0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 0 1 0  0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 0 1 0  0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0  0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0  0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0  0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0  0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0  0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0  0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0  0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1 0  0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0  0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1 0  0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 0  0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0  0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |

1. **class Program**

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();

// Tiêu đề menu

string title = "TÌM KIẾM TRÊN ĐỒ THỊ BẰNG THUẬT TOÁN BFS (Breadth First Search)";

// Danh sách các mục chọn

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",

"4. Bài 4 : Đỉnh khớp",

"5. Bài 5 : Cạnh cầu",

"6. Bài 6 : Kiểm tra đồ thị 2 phía",

"7. Bài 7 : Đường đi trên lưới",

"8. Bài 8 : Mê cung",

"0. Thoát" };

int chon;

do

{

Console.Clear();

// Xuất menu

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

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

AdjList g = new AdjList();

g.FileToAdjList(filePath); g.Output();

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

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

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

break;

}

case 2:

{

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

string filePath = "../../../TextFile/AdjList2.txt";

AdjList g = new AdjList();

g.FileToAdjList(filePath); g.Output();

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());

g.BFS\_XtoY(x, y);

break;

}

case 3:

{

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

string filePath = "../../../TextFile/AdjList2.txt";

AdjList g = new AdjList();

g.FileToAdjList(filePath); g.Output(); 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);

}

}

1. **class AdjList**

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

int[] pre; // Đánh dấu các đỉnh trước sau

int[] index; // Dùng đánh dấu các đỉnh trong các 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; }

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

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

public int[] Index { get => index; set => index = value; }

// Contructor

public AdjList() { }

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

{

v = new LinkedList<int>[k];

}

// copy g --> đồ thị hiện tại v

public AdjList(LinkedList<int>[] g)

{

v = g;

}

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

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();

// Đặt điều kiện không phải đỉnh cô lập

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();

}

// Xuất đồ thị

public void Output()

{

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

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

{

Console.Write(" Đỉnh {0} ->", i);

foreach (int x in v[i])

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

Console.WriteLine();

}

}

// Duyệt đồ thị theo BFS với đỉnh xuất phát s

public void BFS(int s)

{

// Khởi tạo các giá trị visited[i]

visited = new bool[v.Length];

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

visited[i] = false;

// Sử dụng Queue

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

visited[s] = true; // Đánh dấu duyệt s

q.Enqueue(s); // s -> q

while (q.Count != 0)

{

s = q.Dequeue(); // s <- q

Console.Write(s + " - "); // Duyệt s

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

{

if (visited[u]) continue; // Đã duyệt -> bỏ qua

visited[u] = true; // Đánh dấu duyệt u

q.Enqueue(u); // u -> q

}

}

}

// 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

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

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

pre[i] = -1;

// khởi tạo các giá trị cho visited

visited = new bool[v.Length];

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

visited[i] = false;

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

visited[x] = true;

q.Enqueue(x);

while (q.Count != 0)

{

int s = q.Dequeue();

foreach (int u in v[s])

{

if (visited[u]) continue;

visited[u] = true;

q.Enqueue(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

inconnect = 0;

// index : lưu các đỉnh cùng một TPLT

index = new int[n];

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

index[i] = -1;

// Khởi tạo giá trị cho visited[i]

visited = new bool[n];

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

visited[i] = false;

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

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

{

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

inconnect++;

// Tìm và đánh dấu các đỉnh cùng TPLT

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>();

visited[s] = true; // Duyệt đỉnh s

q.Enqueue(s); // Đưa s vào q

while (q.Count != 0) // Lặp khi queue còn phần tử

{

s = q.Dequeue(); // Lấy từ queue ra một phần tử -> s

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

foreach (int u in v[s])

{

if (visited[u] == false)

{

visited[u] = true;

q.Enqueue(u);

}

}

}

}

// 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();

}

}

}