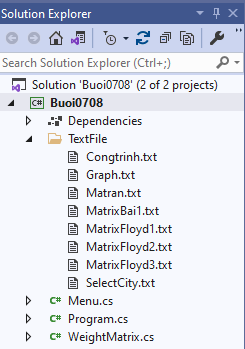
|  |  |
| --- | --- |
| Bài giải  Buổi 03 | **Tìm đường đi ngắn nhất - Giải thuật Dijkstra, Floyd** |
| Buổi 04 | **Ứng dụng giải thuật Dijkstra, Floyd** |

1. **Cấu trú chương trình**



1. **Các file txt**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Graph.txt | MatrixBai1.txt | MatrixFloyd2.txt | Matran.txt | SelectCity.txt |
| 5  0 1 0 0 7  0 0 1 4 8  0 0 0 2 4  0 0 1 0 0  0 0 4 0 0 | 7  0 3 18 6 9 15 0  3 0 12 2 0 0 0  18 12 0 7 0 0 4  6 2 7 0 3 8 0  9 0 0 3 0 4 0  15 0 0 8 4 0 2  0 0 4 0 0 2 0 | 5  0 5 0 9 1  5 0 2 0 0  0 2 0 7 0  9 0 7 0 2  1 0 0 2 0 | 7 6  19 24 23 18 20 16  19 01 01 05 01 16  17 16 01 13 01 20  18 01 01 17 04 21  14 20 17 04 02 28  01 02 01 02 01 59  14 04 09 60 24 18 | 7  0 3 18 6 9 15 0  3 0 12 2 0 0 0  18 12 0 7 0 0 4  6 2 7 0 3 8 0  9 0 0 3 0 4 0  15 0 0 8 4 0 2  0 0 4 0 0 2 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();

string title = "ĐƯỜNG ĐI NGẮN NHẤT"; // Tiêu đề menu

// Danh sách các mục chọn

string[] ms = { "1. Bài 1: Tìm đường đi ngắn nhất từ một đỉnh đến các đỉnh còn lại - Dijkstra",

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

"3. Bài 3: Tìm đường đi ngắn nhất qua đỉnh trung gian",

"4. Bài 4: Đường đi ngắn nhất giữa các cặp đỉnh - Thuật toán Floyd",

"5. Bài 5: Bài toán đi ra biên",

"6. Bài 6: Chọn thành phố để họp",

"7. Bài 7 : Công trình.",

"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:

{

// Tìm đường ngắn nhất từ một đỉnh đến các đỉnh còn lại

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

WeightMatrix g = new WeightMatrix();

g.FileToWeightMatrix(filePath); g.Output();

Console.WriteLine();

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

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

Console.WriteLine();

g.Dijkstra(s);

g.PrintDijkstra(s);

break;

}

case 2:

{

// Bài 1 : Tìm đường ngắn nhất từ x đến y

string filePath = "../../../TextFile/MatrixBai1.txt";

WeightMatrix g = new WeightMatrix();

g.FileToWeightMatrix(filePath); g.Output();

Console.WriteLine();

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

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

Console.Write(" Nhập đỉnh đích y = ");

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

g.MinRouteXY(x,y);

break;

}

case 3:

{

// Bài 2: Đường đi ngắn nhất qua đỉnh trung gian

string filePath = "../../../TextFile/MatrixBai1.txt";

WeightMatrix g = new WeightMatrix();

g.FileToWeightMatrix(filePath); g.Output();

Console.WriteLine();

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

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

Console.Write(" Nhập đỉnh trung gian y = ");

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

Console.Write(" Nhập đỉnh đích z = ");

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

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

break;

}

case 4:

{

// Bài 3: Đường đi ngắn nhất giữa các cặp đỉnh - thuật toán Floyd

string filePath = "../../../TextFile/MatrixFloyd2.txt";

WeightMatrix g = new WeightMatrix();

g.FileToWeightMatrix(filePath); g.Output();

Console.WriteLine();

g.Floyd();

Console.WriteLine(" Ma trận d, p ");

g.Outdp();

Console.WriteLine();

Console.WriteLine(" Chi tiết đường đi ngắn nhất giữa các cặp đỉnh");

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

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

if(i != j) g.Floyd\_RouteXY(i, j);

Console.WriteLine();

break;

}

case 5:

{

// Bài 5 : Bài toán đi ra biên.

// Có các hàm trình bày trong Program

int[,] mt;

Console.WriteLine();

// Đọc file --> ma trận a[,]

string fileName = "../../../TextFile/Matran.txt";

mt = FileToMatrix(fileName);

// Xuất ma trận mt

Console.WriteLine(" Ma trận a :"); PrintMatrix(mt);

// Đọc ma trận mt[,] --> đồ thị ma trận kề g

WeightMatrix g = new WeightMatrix();

g = MatrixToWeightMatrix(mt);

Console.WriteLine();

//g.Output();

// Nhập tọa độ ô xuất phát s(x,y)

Console.WriteLine(" Nhập tọa độ ô xuất phát 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

{

// Chuyển đổi tọa độ (x,y) --> đỉnh s

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

// Gọi Dijkstra từ đỉnh s

g.Dijkstra(s);

// Tìm đường đi ra biên

ToBorderline(g, s);

}

break;

}

case 6:

{

// Bài 6 : Chọn thành phố để họp

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

WeightMatrix g = new WeightMatrix();

g.FileToWeightMatrix(filePath); g.Output();

Console.WriteLine();

// city : thành phố được chọn, Select(g) : tình thành phố

int city;

city = SelectCity(g);

Console.WriteLine(" Thành phố được chọn : {0}", city);

break;

}

case 7:

{

// Bài 7 : Công trình

// Đọc file Congtrinh.txt ra đồ thị ma trận kề g, xuất g

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

WeightMatrix g = new WeightMatrix();

g.FileToWeightMatrix(filePath); g.Output();

Console.WriteLine();

// thay đổi giá trị g.A[i,j] = -g.A[i,j] nếu g.A[i,j] khác vô cùng

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

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

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

g.A[i, j] = -g.A[i, j];

// Cho g.A[7, 9] = g.A[8, 9] = 0;

g.A[7, 9] = g.A[8, 9] = 0;

// Chạy thuật toán Floyd

g.Floyd();

// Xuất kết quả : -g.D[0,9]

Console.WriteLine();

Console.WriteLine(" Thời gian sớm nhất hoàn thành công trình : {0}", -g.D[0,9]);

break;

}

}

Console.ReadKey();

} while (chon != 0);

}

// Đọc file Matran.txt -> ma trận a[,]

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;

}

// Xuất ma trận a -> màn hình

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(" -------------------------");

}

// Đọc ma trận a --> WeightList g. Hàm trả về một đồ thị danh sách kề

static WeightMatrix MatrixToWeightMatrix(int[,] mt)

{

// Khai báo đồ thị ma trận kề

WeightMatrix g = new WeightMatrix();

// Xác định số dòng (row), số cột (col)của ma trận mt

int row = mt.GetLength(0);

int col = mt.GetLength(1);

// Xác định số đỉnh (g.N) của đồ thị và khở tạo g.A

g.N = row \* col;

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

// Khởi tạo giá trị ban đầu cho g.A với ô (i,i) = 0, còn lại = int.MaxValue

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

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

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

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

// Duyệt từng ô của ma trận mt : i:0 -> <row, j:0 -> < col

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

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

{

// với mỗi ô (x,y) xác định đỉnh k tương ứng

int k = i \* col + j;

// Mỗi một ô có tối đa 4 ô kề -> đỉnk k có tối đa 4 đỉnh kề -> tìm các đỉnh kề cho k

// Có 4 trường hợp như sau : i > 0, j > 0, j < col-1, i < row-1

if (i > 0)

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

if (j > 0)

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

if (j < col - 1)

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

if (i < row - 1)

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

}

// trả về g

return g;

}

// Xuất đường đi ra biên từ ô s(x,y) ngắn nhất. Ô nằm trên biên có bậc < 4 (có dưới 4 đỉnh kề

static void ToBorderline(WeightMatrix g, int s)

{

// Sử dụng Stack<int> st -> khởi tạo st

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

// Biến bien : lưu trữ đỉnh nằm trên biên cần tìm, ban đầu = 0

int bien = 0;

// min : lưu giá trị độ dài đường đi ra biên ngắn nhất, ban đầu = int.MaxValue

int min = int.MaxValue;

// Xét tất cả các đường từ (x,y) ra biên để chọn đường ngắn nhất.

// (Đỉnh ở biên có Deg < 4 -> viết thêm hàm DegV(i) tính bậc đỉnh i)

// g.Dist : chứa các độ dài đến các đỉnh, g.Pre : chứa nội dung -> đường đi

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

if (DegV(i, g) < 4) // Chỉ xét các đỉnh i có dưới 4 đỉnh kề -> bậc < 4

if (g.Dist[i] < min)

{

min = g.Dist[i];

bien = i;

}

Console.WriteLine();

// Đến đây biến bien lưu trữ đỉnh nằm trên biên gần nhất

// Xuất đường đi ra biên : từ dinhS --> đỉnh bien và độ dài là bài toán xuất đường đi

while (bien != s)

{

st.Push(bien);

bien = g.Pre[bien];

}

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

Console.ForegroundColor = ConsoleColor.Cyan;

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

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

Console.ForegroundColor = ConsoleColor.White;

}

// Hàm Tính bậc của đỉnh v,lưu ý loại trừ các giá trị đặc biệt trong g.A

static int DegV(int v, WeightMatrix g)

{

int deg = 0;

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

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

return deg;

}

// Bài 6 : chọn thành phố

static int SelectCity(WeightMatrix g)

{

// Chạy thuật toán Floyd -> ma trận d

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

// Biến min tham gia tìm, khởi tạo = int.MaxValue;

// Duyệt (i : 0 -> < g.N)

{

//Biến max : đường dài nhất khi chọn họp thành phố i

// Duyệt (j : 0 -> g.N)

// Nếu g.D[i, j] > max thì max = g.D[i, j]

// Nếu max < min

if (max < min)

{

// cty = i và min = max;

}

}

// Trả về cty

return cty;

}

}

}

1. **class WeightMatrix**

// Đồ thị Ma trận kề có trọng số

class WeightMatrix

{

public int n;

public int[,] a;

// Các array là biến toàn cục chỉ phục vụ cho giải thuật

int[] pre; // Lưu trữ đỉnh nằm ngay trước trên đường đi

int[] dist; // Lưu độ dài ngắn nhất đến các đỉnh

bool[] visited; // Duyệt đỉnh ghé thăm

int[,] p; // Ma trận Lưu trữ đỉnh nằm ngay trước trên đường đi (Floy)

int[,] d; // Ma trận Lưu độ dài ngắn nhất đến các đỉnh (Floy)

// propeties

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

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

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

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

public int[,] D { get => d; set => d = value; }

public int[,] P { get => p; set => p = value; }

// constructor không đối số

public WeightMatrix() { }

// constructor có đối số

// Khởi tạo đồ thị có k đỉnh

public WeightMatrix(int k)

{

n = k;

a = new int[n, n];

}

// Đọc file ra ma trận kề có trọng số

public void FileToWeightMatrix(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++)

if (i == j) a[i, j] = 0;

else

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

}

sr.Close();

}

// Xuất Ma trận kề có trọng số lên màn hình

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

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

Console.WriteLine();

}

}

// Dijkstra : tìm đường đi ngắn nhất đến các đỉnh từ đỉnh s

public void Dijkstra(int s)

{

// pre : Lưu đỉnh nằm trước trên đường từ s đi qua

pre = new int[n];

// dist : Lưu độ dài ngán nhất từ s đến các đỉnh còn lại

dist = new int[n];

// visited : Đánh dấu đỉnh đã đi qua

visited = new bool[n];

// Khởi gán giá trị ban đầu

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

{

dist[i] = int.MaxValue;

visited[i] = false;

pre[i] = s;

}

dist[s] = 0;

// Duyệt các đỉnh của đồ thị

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

{

int u = dmin(); // Gọi u là đỉnh sao cho dist[u] nhỏ nhất

visited[u] = true; // Đánh dấu u

// Duyệt các đỉnh v thuộc Kề(u) (tồn tại cạnh (u, v) )

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

// Nếu (v chưa đánh dấu) và (tồn tại cạnh (u,v))

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

// Nếu (dist[u] đã xác định) và (dist[u] + a[u, v] < dist[v])

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

{

dist[v] = dist[u] + a[u, v]; // Đặt dist[v] = dist[u] + a[u, v];

pre[v] = u; // Đặt đỉnh u trước v trên đường đi

}

}

// Kết quả : xác định giá trị các phần tử của dist[] và pre[]

}

// Tìm đỉnh minIndex có dist[minIndex] là nhỏ nhất

public int dmin()

{

int minIndex = 0; // Giá trị trả về

int min = int.MaxValue;

// Tìm min tại các đỉnh chưa xét

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

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

{

min = dist[i];

minIndex = i;

}

return minIndex;

}

// Xuất kết quả dist[] và pre[]

public void PrintDijkstra(int s)

{

Console.WriteLine("Đường đi ngắn nhất từ " + s + " đến các đỉnh còn lại :");

// Xuất số hiệu các đỉnh

Console.Write(" Đỉnh : ");

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

Console.Write(" " + i);

Console.WriteLine();

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

// Xuất array pre[]

Console.Write(" pre : ");

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

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

Console.WriteLine();

// Xuất array dist[]

Console.Write(" dist : ");

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

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

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

}

// Bài 1 : Tìm đường ngắn nhất từ x đến y

public void MinRouteXY(int x, int y)

{

// Chọn x là đỉnh xuất phát

Dijkstra(x);

// Kết quả : 2 array pre[] và dist[] có giá trị

// ?

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(" Đường đi ngắn nhất từ {0} đến {1} : {2} ->", x, y, x);

while (st.Count > 0)

{

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

}

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

}

// Bài 2: Đường đi ngắn nhất từ x đến z qua đỉnh trung gian y

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

{

// Tìm đường đi ngắn nhất từ x đến y : dxy

// tìm đường đi ngắn nhất từ y đến z : dyz

// Đường đi ngắn nhất tư x đến z có đi qua y : dxz = dxy + dyz

}

// Bài 3 : Đường đi ngắn nhất giữa các cặp đỉnh - giải thuật Floyd

public void Floyd()

{

// khởi tạo giá trị cho d[,], p[,]

d = new int[n, n];

p = new int[n, n];

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

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

{

d[i, j] = a[i, j];

p[i, j] = i;

}

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

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

{

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

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

}

//Outdp();

}

//Xuất ma trận d và p

public void Outdp()

{

Console.WriteLine(" dist : ");

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

{

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

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

Console.WriteLine();

}

Console.WriteLine();

Console.WriteLine(" pre : ");

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

{

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

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

Console.WriteLine();

}

}

// Sử dụng 2 ma trận d và p để xuất đường đi ngắn nhất từ x đến y

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

{

// Độ dài ngắn nhất từ x -> y = d[x,y]

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

int k = y; st.Push(k);

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

{

k = p[x, k];

st.Push(k);

}

Console.WriteLine();

Console.Write(" Đường đi ngắn nhất từ {0} đến {1} : {2} ->", x, y, x);

while (st.Count > 0)

{

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

}

Console.WriteLine(" độ dài : " + d[x, y]);

}

}