**Buổi 8: Vận dụng thuật toán tìm đường đi ngắn nhất**

**Bài 1. Đi ra biên**

class Program

{

static void Main(string[] args)

{

string inputFileName = "RaBien.INP";

string outputFileName = "RaBien.OUT";

string inputFilePath = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName, inputFileName);

string outputFilePath = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName, outputFileName);

try

{

// Đọc dữ liệu từ file input

StreamReader reader = new StreamReader(inputFilePath);

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

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

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

int x = int.Parse(input[2]) - 1;

int y = int.Parse(input[3]) - 1;

int[,] grid = new int[n, m];

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

{

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

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

{

grid[i, j] = int.Parse(row[j]);

}

}

reader.Close();

int minSum = FindMinimumPathToBorder(grid, n, m, x, y);

// Ghi kết quả ra file output

StreamWriter writer = new StreamWriter(outputFilePath);

writer.WriteLine(minSum);

writer.Close();

Console.WriteLine("Kết quả đã được ghi vào file RaBien.OUT");

}

catch (Exception ex)

{

Console.WriteLine("Đã xảy ra lỗi: " + ex.Message);

}

}

static int FindMinimumPathToBorder(int[,] grid, int n, int m, int x, int y)

{

// Biến lưu trữ tổng giá trị các ô đã đi qua

int sum = grid[x, y];

// Biến lưu trữ các ô đã thăm

bool[,] visited = new bool[n, m];

visited[x, y] = true;

// Hàng đợi lưu trữ các ô cần kiểm tra

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

queue.Enqueue((x, y));

// Mảng lưu trữ các hướng di chuyển

int[] dx = { -1, 1, 0, 0 };

int[] dy = { 0, 0, -1, 1 };

// Duyệt qua các ô cần kiểm tra trong hàng đợi

while (queue.Count > 0)

{

var (currentX, currentY) = queue.Dequeue();

// Nếu ô hiện tại là biên, thoát vòng lặp

if (currentX == 0 || currentX == n - 1 || currentY == 0 || currentY == m - 1)

{

break;

}

// Duyệt qua các ô kề cạnh của ô hiện tại

for (int i = 0; i < 4; i++)

{

int newX = currentX + dx[i];

int newY = currentY + dy[i];

// Kiểm tra xem ô kề có hợp lệ không

if (newX >= 0 && newX < n && newY >= 0 && newY < m && !visited[newX, newY])

{

// Đánh dấu ô kề đã thăm và thêm vào hàng đợi

visited[newX, newY] = true;

queue.Enqueue((newX, newY));

// Cập nhật tổng giá trị các ô đã đi qua

sum += grid[newX, newY];

}

}

}

return sum;

}

}

**Bài 2. Chọn thành phố để tổ chức họp**

class Program

{

static void Main(string[] args)

{

string inputFileName = "ChonThanhPho.INP";

string outputFileName = "ChonThanhPho.OUT";

string inputFilePath = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName, inputFileName);

string outputFilePath = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName, outputFileName);

try

{

// Đọc dữ liệu từ file input

StreamReader reader = new StreamReader(inputFilePath);

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

int[,] distances = new int[n, n];

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

{

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

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

{

distances[i, j] = int.Parse(row[j]);

}

}

reader.Close();

// Tính toán khoảng cách trung bình từ mỗi thành phố đến tất cả các thành phố khác

int[] avgDistances = new int[n];

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

{

int sum = 0;

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

{

if (i != j)

{

sum += distances[i, j];

}

}

avgDistances[i] = sum / (n - 1);

}

// Tìm thành phố có khoảng cách trung bình nhỏ nhất

int minAvgDistanceIndex = 0;

int minAvgDistance = avgDistances[0];

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

{

if (avgDistances[i] < minAvgDistance)

{

minAvgDistance = avgDistances[i];

minAvgDistanceIndex = i;

}

}

// Ghi kết quả ra file output

StreamWriter writer = new StreamWriter(outputFilePath);

writer.WriteLine(minAvgDistanceIndex + 1); // Thành phố đánh số từ 1

writer.WriteLine(minAvgDistance);

writer.Close();

Console.WriteLine("Kết quả đã được ghi vào file ChonThanhPho.OUT");

}

catch (Exception ex)

{

Console.WriteLine("Đã xảy ra lỗi: " + ex.Message);

}

}

}

**Bài 3. Đường tròn**

class Program

{

static void Main(string[] args)

{

string inputFileName = "DuongTron.INP";

string outputFileName = "DuongTron.OUT";

string inputFilePath = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName, inputFileName);

string outputFilePath = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName, outputFileName);

try

{

// Đọc dữ liệu từ file input

StreamReader reader = new StreamReader(inputFilePath);

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

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

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

int t = int.Parse(firstLine[2]);

List<Circle> circles = new List<Circle>();

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

{

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

double x = double.Parse(circleData[0]);

double y = double.Parse(circleData[1]);

double r = double.Parse(circleData[2]);

circles.Add(new Circle(x, y, r));

}

reader.Close();

// Tạo danh sách kề

List<Edge>[] graph = new List<Edge>[n];

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

{

graph[i] = new List<Edge>();

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

{

if (i != j)

{

double distance = CalculateDistance(circles[i], circles[j]);

graph[i].Add(new Edge(j, distance));

}

}

}

// Áp dụng thuật toán Dijkstra để tìm đường đi ngắn nhất từ đường tròn xuất phát đến đích

double minCost = Dijkstra(graph, s - 1, t - 1);

// Ghi kết quả ra file output

StreamWriter writer = new StreamWriter(outputFilePath);

writer.WriteLine(minCost);

writer.Close();

Console.WriteLine("Kết quả đã được ghi vào file DuongTron.OUT");

}

catch (Exception ex)

{

Console.WriteLine("Đã xảy ra lỗi: " + ex.Message);

}

}

// Hàm tính khoảng cách giữa hai đường tròn

static double CalculateDistance(Circle c1, Circle c2)

{

double dx = c1.X - c2.X;

double dy = c1.Y - c2.Y;

return Math.Sqrt(dx \* dx + dy \* dy) - c1.R - c2.R; // Khoảng cách giữa hai đường tròn trừ đi bán kính của chúng

}

// Hàm Dijkstra

static double Dijkstra(List<Edge>[] graph, int start, int end)

{

int n = graph.Length;

double[] distance = new double[n];

bool[] visited = new bool[n];

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

{

distance[i] = double.MaxValue;

visited[i] = false;

}

distance[start] = 0;

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

{

int u = MinDistance(distance, visited);

visited[u] = true;

foreach (Edge edge in graph[u])

{

int v = edge.Vertex;

double weight = edge.Weight;

if (!visited[v] && distance[u] != double.MaxValue && distance[u] + weight < distance[v])

{

distance[v] = distance[u] + weight;

}

}

}

return distance[end];

}

// Hàm tìm đỉnh có khoảng cách ngắn nhất chưa được thăm

static int MinDistance(double[] distance, bool[] visited)

{

double minDistance = double.MaxValue;

int minIndex = -1;

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

{

if (!visited[v] && distance[v] <= minDistance)

{

minDistance = distance[v];

minIndex = v;

}

}

return minIndex;

}

}

// Lớp đại diện cho một đường tròn

class Circle

{

public double X { get; }

public double Y { get; }

public double R { get; }

public Circle(double x, double y, double r)

{

X = x;

Y = y;

R = r;

}

}

// Lớp đại diện cho một cạnh trong đồ thị

class Edge

{

public int Vertex { get; }

public double Weight { get; }

public Edge(int vertex, double weight)

{

Vertex = vertex;

Weight = weight;

}

**Bài 4. Đến trường**

class Program

{

static void Main(string[] args)

{

string inputFileName = "SCHOOL.INP";

string outputFileName = "SCHOOL.OUT";

string inputFilePath = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName, inputFileName);

string outputFilePath = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName, outputFileName);

try

{

// Đọc dữ liệu từ file input

StreamReader reader = new StreamReader(inputFilePath);

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

int N = int.Parse(firstLine[0]); // Số nút giao thông

int M = int.Parse(firstLine[1]); // Số đường đi một chiều

int K = int.Parse(firstLine[2]); // Nút giao thông - trường của Tuấn

List<Edge>[] graph = new List<Edge>[N];

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

{

graph[i] = new List<Edge>();

}

// Đọc thông tin về các đường đi một chiều

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

{

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

int u = int.Parse(line[0]) - 1;

int v = int.Parse(line[1]) - 1;

int a = int.Parse(line[2]);

int b = int.Parse(line[3]);

graph[u].Add(new Edge(v, a, b));

}

reader.Close();

// Tìm thời gian sớm nhất mà mẹ Tuấn đến được cơ quan và thời gian Tuấn đi đến trường

int motherTime = Dijkstra(graph, 0, N - 1, N, K);

// Ghi kết quả ra file output

StreamWriter writer = new StreamWriter(outputFilePath);

writer.WriteLine(motherTime);

writer.Close();

Console.WriteLine("Kết quả đã được ghi vào file SCHOOL.OUT");

}

catch (Exception ex)

{

Console.WriteLine("Đã xảy ra lỗi: " + ex.Message);

}

}

// Hàm Dijkstra

static int Dijkstra(List<Edge>[] graph, int start, int end, int N, int K)

{

int[] distance = new int[N];

bool[] visited = new bool[N];

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

{

distance[i] = int.MaxValue;

visited[i] = false;

}

distance[start] = 0;

for (int count = 0; count < N - 1; count++)

{

int u = MinDistance(distance, visited, N);

visited[u] = true;

foreach (Edge edge in graph[u])

{

int v = edge.Vertex;

int a = edge.WalkTime;

int b = edge.CarTime;

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

{

distance[v] = distance[u] + a;

}

}

}

return distance[end];

}

// Hàm tìm đỉnh có khoảng cách ngắn nhất chưa được thăm

static int MinDistance(int[] distance, bool[] visited, int N)

{

int minDistance = int.MaxValue;

int minIndex = -1;

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

{

if (!visited[v] && distance[v] <= minDistance)

{

minDistance = distance[v];

minIndex = v;

}

}

return minIndex;

}

}

class Edge

{

public int Vertex { get; }

public int WalkTime { get; }

public int CarTime { get; }

public Edge(int vertex, int walkTime, int carTime)

{

Vertex = vertex;

WalkTime = walkTime;

CarTime = carTime;

}

}