Морозова Мария Леонардовна, ИПсп-121. Запахи кода.

1. Длинный метод.

static void Main()

{

try

{

// Матрица весов дорог между вершинами

double[,] adjacencyMatrix = {

{ 0, 2.75, double.MaxValue, double.MaxValue, 2.04, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue },

{ 2.75, 0, 0.63, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue },

{ double.MaxValue, 0.63, 0, 1.4, double.MaxValue, double.MaxValue, double.MaxValue, 1.72, double.MaxValue },

{ double.MaxValue, double.MaxValue, 1.4, 0, 1.51, double.MaxValue, 1.49, double.MaxValue, double.MaxValue },

{ 2.04, double.MaxValue, double.MaxValue, 1.51, 0, 0.73, double.MaxValue, double.MaxValue, double.MaxValue },

{ double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, 0.73, 0, 1.98, double.MaxValue, 3.5 },

{ double.MaxValue, double.MaxValue, double.MaxValue, 1.49, double.MaxValue, 1.98, 0, 0.37, double.MaxValue },

{ double.MaxValue, double.MaxValue, 1.72, double.MaxValue, double.MaxValue, double.MaxValue, 0.37, 0, 1.25 },

{ double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, 3.5, double.MaxValue, 1.25, 0 }

};

int n = adjacencyMatrix.GetLength(0);

// Запрос начальной точки у пользователя

int startNode;

while (true)

{

Console.Write("Введите номер начальной точки (от 1 до 9): ");

if (int.TryParse(Console.ReadLine(), out startNode) && startNode >= 1 && startNode <= 9)

break;

Console.WriteLine("Ошибка: введите число от 1 до 9.");

}

startNode--;

// Запрос точек без контейнеров

string[] input;

while (true)

{

Console.Write("Введите номера точек без контейнеров через пробел (или нажмите Enter, если все точки задействованы): ");

input = Console.ReadLine()?.Split(' ') ?? Array.Empty<string>();

bool valid = true;

foreach (string s in input)

{

if (!string.IsNullOrWhiteSpace(s) && (!int.TryParse(s, out int node) || node < 1 || node > 9))

{

valid = false;

break;

}

}

if (valid) break;

Console.WriteLine("Ошибка: вводите только числа от 1 до 9, разделённые пробелом.");

}

foreach (string s in input)

{

if (int.TryParse(s, out int node) && node >= 1 && node <= 9)

{

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

{

adjacencyMatrix[node - 1, i] = double.MaxValue;

adjacencyMatrix[i, node - 1] = double.MaxValue;

}

}

}

// Запуск алгоритма Дейкстры

double[] shortestDistances = Dijkstra(adjacencyMatrix, startNode, n);

// Вывод результатов

Console.WriteLine("Кратчайшие расстояния от точки {0}:", startNode + 1);

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

{

if (shortestDistances[i] == double.MaxValue)

Console.WriteLine("Точка {0}: недоступна", i + 1);

else

Console.WriteLine("Точка {0}: {1} км", i + 1, shortestDistances[i]);

}

}

catch (Exception ex)

{

Console.WriteLine("Произошла ошибка: " + ex.Message);

}

}

А также метод Дейкстры, который был представлен как необходимый неизменный метод.

private static double[] Dijkstra(double[,] a, int v0, int n)

{

double[] dist = new double[n];

bool[] vis = new bool[n];

int unvis = n;

int v;

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

dist[i] = Double.MaxValue;

dist[v0] = 0.0;

while (unvis > 0)

{

v = -1;

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

{

if (vis[i])

continue;

if ((v == -1) || (dist[v] > dist[i]))

v = i;

}

vis[v] = true;

unvis--;

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

{

if (dist[i] > dist[v] + a[v, i])

dist[i] = dist[v] + a[v, i];

}

}

return dist;

}

1. Жёстко заданные данные

double[,] adjacencyMatrix = {

{ 0, 2.75, double.MaxValue, double.MaxValue, 2.04, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue },

{ 2.75, 0, 0.63, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue },

{ double.MaxValue, 0.63, 0, 1.4, double.MaxValue, double.MaxValue, double.MaxValue, 1.72, double.MaxValue },

{ double.MaxValue, double.MaxValue, 1.4, 0, 1.51, double.MaxValue, 1.49, double.MaxValue, double.MaxValue },

{ 2.04, double.MaxValue, double.MaxValue, 1.51, 0, 0.73, double.MaxValue, double.MaxValue, double.MaxValue },

{ double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, 0.73, 0, 1.98, double.MaxValue, 3.5 },

{ double.MaxValue, double.MaxValue, double.MaxValue, 1.49, double.MaxValue, 1.98, 0, 0.37, double.MaxValue },

{ double.MaxValue, double.MaxValue, 1.72, double.MaxValue, double.MaxValue, double.MaxValue, 0.37, 0, 1.25 },

{ double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, double.MaxValue, 3.5, double.MaxValue, 1.25, 0 }

};