Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Дискретная математика»

Тема: «Алгоритм Шинбелла»

Семестр 4

Выполнил работу

Студент группы ИВТ-23-2б

Меновщиков Глеб Николаевич

Проверил

старший преподаватель кафедры ИТАС

Рустамханова Гульшат Ильдаровна

Пермь 2025

**Постановка задачи**

Разработать программу, которая находит оптимальные пути между всеми парами вершин в ориентированном графе, заданном матрицей смежности, для заданного количества шагов. Программа должна поддерживать два режима работы: поиск минимальных и максимальных путей.

**Входные данные**

Граф задаётся матрицей смежности размером 10×10, хранящейся в текстовом файле с расширением .txt.

В файле числа записаны построчно через пробел (0 означает отсутствие ребра, ненулевые значения — вес ребра).

**Обработка данных:**

Загрузка матрицы смежности из файла.

Преобразование нулевых значений в INF (бесконечность) для режима min или NINF (минус бесконечность) для режима max.

Вычисление оптимальных путей между всеми парами вершин за заданное количество шагов с использованием матричного метода.

**Вывод результатов:**

Матрица оптимальных стоимостей путей.

Список всех возможных маршрутов с указанием их стоимости и пути.

Общее количество найденных маршрутов.

Средняя стоимость маршрута.

Минимальная и максимальная стоимость маршрута.

**Код программы**

using System;

using System.Collections.Generic;

using System.IO;

using System.Globalization;

using System.Linq;

class Program

{

const int N = 10;

const int INF = int.MaxValue;

const int NINF = int.MinValue;

class RouteInfo : IComparable<RouteInfo>

{

public int From { get; set; }

public int To { get; set; }

public int Cost { get; set; }

public List<int> Points { get; set; }

public int CompareTo(RouteInfo other)

{

if (From != other.From) return From.CompareTo(other.From);

if (To != other.To) return To.CompareTo(other.To);

return Cost.CompareTo(other.Cost);

}

}

static void LoadMatrix(string filename, List<List<int>> matrix, bool findMin)

{

try

{

using (var file = new StreamReader(filename))

{

matrix.Clear();

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

{

var line = file.ReadLine().Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

var row = new List<int>();

foreach (var num in line)

{

int val = int.Parse(num);

row.Add(val == 0 ? (findMin ? INF : NINF) : val);

}

matrix.Add(row);

}

}

}

catch

{

Console.WriteLine("Ошибка открытия файла");

Environment.Exit(1);

}

}

static void CalculateRoutes(

List<List<int>> A,

List<List<int>> B,

List<List<List<int>>> pathsA,

out List<List<int>> result,

out List<List<List<int>>> resultPaths,

bool findMin)

{

result = new List<List<int>>();

resultPaths = new List<List<List<int>>>();

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

{

result.Add(new List<int>());

resultPaths.Add(new List<List<int>>());

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

{

result[i].Add(findMin ? INF : NINF);

resultPaths[i].Add(new List<int>());

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

{

bool available = findMin

? (A[i][k] != INF && B[k][j] != INF)

: (A[i][k] != NINF && B[k][j] != NINF);

if (available)

{

int sum = A[i][k] + B[k][j];

bool better = findMin

? (sum < result[i][j])

: (sum > result[i][j]);

if (better)

{

result[i][j] = sum;

var newPath = new List<int>(pathsA[i][k]);

if (newPath.Count > 0 && newPath.Last() != k)

newPath.Add(k);

else if (newPath.Count == 0)

{

newPath.Add(i);

newPath.Add(k);

}

newPath.Add(j);

resultPaths[i][j] = newPath;

}

}

}

}

}

}

static void ShowNetwork(List<List<int>> matrix, bool findMin, int steps)

{

Console.WriteLine($"\nАНАЛИЗ СЕТИ (ШАГИ: {steps})");

Console.WriteLine("===========================================");

Console.WriteLine($"| Матрица связей для {(findMin ? "МИНИМАЛЬНЫХ" : "МАКСИМАЛЬНЫХ")} путей |");

Console.WriteLine("===========================================");

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

{

Console.Write($"Из {i} в: ");

foreach (var val in matrix[i])

{

if ((findMin && val == INF) || (!findMin && val == NINF))

Console.Write(" -- ");

else

Console.Write($"{val,3} ");

}

Console.WriteLine();

}

Console.WriteLine("===========================================\n");

}

static void ShowRoutes(List<List<List<int>>> paths, List<List<int>> weights, bool findMin, int steps)

{

var routes = new List<RouteInfo>();

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

{

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

{

if ((findMin && weights[i][j] != INF) || (!findMin && weights[i][j] != NINF))

{

routes.Add(new RouteInfo

{

From = i,

To = j,

Cost = weights[i][j],

Points = paths[i][j]

});

}

}

}

routes.Sort();

Console.WriteLine("\nДЕТАЛИ МАРШРУТОВ:");

Console.WriteLine("==========================================================");

foreach (var route in routes)

{

Console.Write($"Маршрут {route.From} -> {route.To} ");

Console.WriteLine($"(Стоимость: {route.Cost})\nТочки: ");

for (int k = 0; k < route.Points.Count; k++)

{

Console.Write(route.Points[k]);

if (k != route.Points.Count - 1) Console.Write(" -> ");

}

Console.WriteLine("\n----------------------------------------------------------");

}

}

static void ShowStats(List<List<int>> matrix, bool findMin, int steps)

{

int count = 0;

int minCost = INF, maxCost = NINF;

double total = 0;

foreach (var row in matrix)

{

foreach (var val in row)

{

if ((findMin && val != INF) || (!findMin && val != NINF))

{

count++;

total += val;

if (val < minCost) minCost = val;

if (val > maxCost) maxCost = val;

}

}

}

Console.WriteLine("\nСТАТИСТИКА:");

Console.WriteLine("----------------------------------------");

Console.WriteLine($"Всего маршрутов: {count}");

if (count > 0)

{

Console.WriteLine($"Средняя стоимость: {total / count:F2}");

Console.WriteLine($"Минимальная стоимость: {minCost}");

Console.WriteLine($"Максимальная стоимость: {maxCost}");

}

Console.WriteLine($"Анализ завершен для {steps} шагов");

Console.WriteLine("----------------------------------------\n");

}

static void Main()

{

CultureInfo.CurrentCulture = new CultureInfo("ru-RU");

Console.WriteLine("АНАЛИЗАТОР МАРШРУТОВ");

Console.WriteLine("====================");

Console.Write("Введите имя файла: ");

string filename = Console.ReadLine();

Console.Write("Количество шагов: ");

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

Console.Write("Режим (min/max): ");

string mode = Console.ReadLine();

Console.WriteLine("====================\n");

bool findMin = mode == "min";

var network = new List<List<int>>();

LoadMatrix(filename, network, findMin);

var currentWeights = network;

var currentPaths = new List<List<List<int>>>();

// Инициализация путей

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

{

currentPaths.Add(new List<List<int>>());

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

{

currentPaths[i].Add(new List<int>());

if ((findMin && network[i][j] != INF) || (!findMin && network[i][j] != NINF))

{

currentPaths[i][j].Add(i);

currentPaths[i][j].Add(j);

}

}

}

// Расчет маршрутов

for (int step = 1; step < steps; step++)

{

CalculateRoutes(

currentWeights,

network,

currentPaths,

out var newWeights,

out var newPaths,

findMin

);

currentWeights = newWeights;

currentPaths = newPaths;

}

ShowNetwork(currentWeights, findMin, steps);

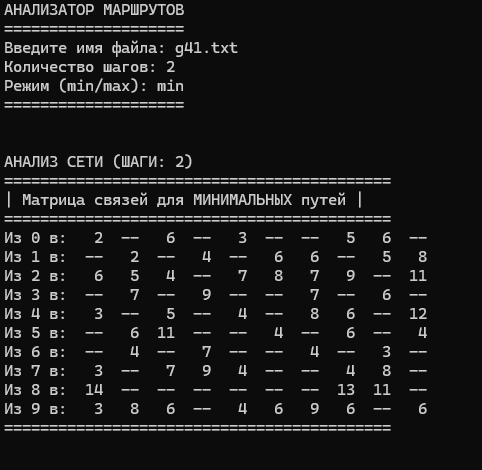
ShowRoutes(currentPaths, currentWeights, findMin, steps);

ShowStats(currentWeights, findMin, steps);

}

}

**Результаты работы программы**



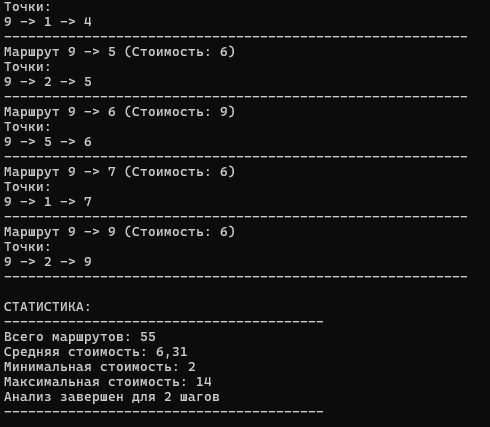
****

Рисунок 1,2 - результаты работы программы