**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра автоматизованих систем обробки інформації і управління**

Звіт

з лабораторної роботи № 4 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

**«Прикладні задачі теорії графів ч.2»**

**Виконав(ла)**

*ІП-13 Шиманська Ганна Артурівна*

(шифр, прізвище, ім'я, по батькові)

**Перевірив**

*Сопов Олексій Олександрович*

(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1. **МЕТА ЛАБОРАТОРНОЇ РОБОТИ 3**
2. **ЗАВДАННЯ 4**
3. **ВИКОНАННЯ 8**
   1. ПСЕВДОКОД АЛГОРИТМУ 8
   2. ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ 8
      1. *Вихідний код 8*

**ВИСНОВОК 10**

**КРИТЕРІЇ ОЦІНЮВАННЯ 11**

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити додаткові прикладні алгоритми на графах та способи їх імплементації.

2 ЗАВДАННЯ

Згідно варіанту (таблиця 2.1), розробити та записати алгоритм задачі на графах за допомогою псевдокоду (чи іншого способу за вибором).

Виконати програмну реалізацію алгоритму на будь-якій мові програмування для довільного графа, передбачити введення розмірності графа та введення даних графа вручну чи випадковим чином.

Для самостійно обраного графа (розмірності не менше 7 вершин) розв’язати задану за варіантом задачу вручну.

Зробити узагальнений висновок з лабораторної роботи, у якому порівняти програмне та ручне розв’язання задачі.

Таблиця 2.1 – Варіанти алгоритмів

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Задача** | **Алгоритм** | **Спосіб задання**  **мережі** |
| 1 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла  (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 2 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла  (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 3 | Пошук усіх  найкоротших шляхів | Данцига (вбудований  спосіб визначення шляхів) | Ортграф, матриця  вагів |
| 4 | Пошук усіх  найкоротших шляхів | Данцига (зовнішній спосіб  визначення шляхів) | Ортграф, матриця  вагів |
| 5 | Задача визначення  збільшуючого ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 6 | Задача про  максимальний потік | Форда - Фалкерсона | Ортграф, матриця  вагів |
| 7 | Задача про | Едмондса - Карпа | Ортграф, матриця |

|  |  |  |  |
| --- | --- | --- | --- |
|  | максимальний потік |  | вагів |
| 8 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла  (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 9 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла  (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 10 | Пошук усіх  найкоротших шляхів | Данцига (вбудований  спосіб визначення шляхів) | Ортграф, матриця  вагів |
| 11 | Пошук усіх  найкоротших шляхів | Данцига (зовнішній спосіб  визначення шляхів) | Ортграф, матриця  вагів |
| 12 | Задача визначення збільшуючого  ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 13 | Задача про  максимальний потік | Форда - Фалкерсона | Ортграф, матриця  вагів |
| 14 | Задача про  максимальний потік | Едмондса - Карпа | Ортграф, матриця  вагів |
| 15 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла  (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 16 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла  (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 17 | Пошук усіх  найкоротших шляхів | Данцига (вбудований  спосіб визначення шляхів) | Ортграф, матриця  вагів |
| 18 | Пошук усіх  найкоротших шляхів | Данцига (зовнішній спосіб  визначення шляхів) | Ортграф, матриця  вагів |
| 19 | Задача визначення | За означенням | Ортграф, матриця |

|  |  |  |  |
| --- | --- | --- | --- |
|  | збільшуючого  ланцюга |  | вагів, типи дуг |
| 20 | Задача про  максимальний потік | Форда - Фалкерсона | Ортграф, матриця  вагів |
| 21 | Задача про  максимальний потік | Едмондса - Карпа | Ортграф, матриця  вагів |
| 22 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла  (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 23 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла  (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 24 | Пошук усіх  найкоротших шляхів | Данцига (вбудований  спосіб визначення шляхів) | Ортграф, матриця  вагів |
| 25 | Пошук усіх  найкоротших шляхів | Данцига (зовнішній спосіб  визначення шляхів) | Ортграф, матриця  вагів |
| 26 | Задача визначення збільшуючого  ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 27 | Задача про  максимальний потік | Форда - Фалкерсона | Ортграф, матриця  вагів |
| 28 | Задача про максимальний потік | Едмондса - Карпа | Ортграф, матриця вагів |
| 29 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |

|  |  |  |  |
| --- | --- | --- | --- |
| 30 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 31 | Пошук усіх найкоротших шляхів | Данцига (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 32 | Пошук усіх найкоротших шляхів | Данцига (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 33 | Задача визначення збільшуючого ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 34 | Задача про максимальний потік | Форда - Фалкерсона | Ортграф, матриця вагів |
| 35 | Задача про максимальний потік | Едмондса - Карпа | Ортграф, матриця вагів |
| 36 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |

3 ВИКОНАННЯ

* 1. Псевдокод алгоритму

**Функція** GetPathMatrix(matrix)

pathMatrix[matrix.GetLength(0), matrix.GetLength(1)]

**для**  i = 0; i < pathMatrix.GetLength(0); i++ **повторити**

**для**  j = 0; i < pathMatrix.GetLength(1); j++ **повторити**

**якщо** matrix[i, j] is > 0 and < int.MaxValue/2

**то**

pathMatrix[i, j] = i

**все якщо**

**все повторити**

**все повторити**

**для** m = 0; m < matrix.GetLength(0); m++ **повторити**

**для** i = 0; i < matrix.GetLength(0); i++ **повторити**

**якщо** i == m

**то**

**продовжити**

**все якщо**

**для** j = 0; j < matrix.GetLength(1) ; j++ **повторити**

**якщо** i == j || j == m

**то**

**продовжити**

**все якщо**

**якщо** matrix[i, m] + matrix[m, j] < matrix[i, j]

**то**

pathMatrix[i, j] = m

matrix[i, j] = matrix[i, m] + matrix[m, j]

**все якщо**

**все повторити**

**все повторити**

**все повторити**

**повернути** pathMatrix

* 1. Програмна реалізація алгоритму
     1. Вихідний код

using System;  
  
namespace Lab4  
{  
 public class InputOperations  
 {  
 public static int GetDimensions()  
 {  
 while (true)  
 {  
 Console.WriteLine("Enter your graph size");  
 string input = Console.ReadLine();  
 if (int.TryParse(input, out int x))  
 {  
 return x;  
 }  
 }  
 }  
  
 public static bool IsRandomized()  
 {  
 Console.WriteLine("Do you want to fill the matrix randomly (r) or manually (m) ? ");  
 while (true)  
 {  
 string answer = Console.ReadLine();  
 if (answer == "r")  
 {  
 return true;  
 }  
  
 if (answer == "m")  
 {  
 return false;  
 }  
  
 Console.WriteLine("Incorrect input. Try again...");  
 }  
 }  
  
public static int[,] GetManualInput(int n)  
{  
 int[,] matrix = new int[n, n];  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < n; j++)  
 {  
 matrix[i, j] = int.MaxValue;  
 }  
  
 while (true)  
 {  
 Console.WriteLine($"Enter adjacent vertice's number to vertice {i + 1}:");  
 string input = Console.ReadLine();  
 if (string.IsNullOrEmpty(input)) break;  
 if (int.TryParse(input, out int adjacentVertice) && adjacentVertice <= n && adjacentVertice > 0)  
 {  
 if (adjacentVertice == i)  
 {  
 Console.WriteLine("Distance must be 0");  
 continue;  
 }  
 while (true)  
 {  
 Console.WriteLine($"Enter your distance between {i+1} and {adjacentVertice} ");  
 string distanceInput = Console.ReadLine();  
 if (int.TryParse(distanceInput, out int distance))  
 {  
 matrix[i, adjacentVertice-1] = distance;  
 break;  
 }  
 Console.WriteLine("Incorrect input");  
 }  
 }  
 else Console.WriteLine("Incorrect input");  
 }  
 }  
  
 return matrix;  
}

public static (int, int) GetEntryPointsIndexes(int verticeNumber)  
 {  
 Console.WriteLine("Enter startpoint and endpoint indexes, separated by ',' : ");  
 while (true)  
 {  
 string input = Console.ReadLine();  
 string[] points = input.Split(",");  
 if (points.Length == 2 && int.TryParse(points[0], out int startPoint) && int.TryParse(points[1], out int endPoint)   
 && startPoint > 0 && startPoint < verticeNumber && endPoint > 0 && endPoint < verticeNumber)   
 {  
 return (startPoint-1, endPoint-1);  
 }  
 Console.WriteLine("Incorrect input");  
 }  
 }  
  
 }  
}

using System;  
  
namespace Lab4  
{  
 public class MatrixRandomizer  
 {  
 public static int[,] CreateRandomMatrix(int n)  
 {  
 Random rand = new Random();  
 int[,] matrix = new int[n, n];  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < n; j++)  
 {  
 if (i == j) matrix[i, j] = 0;  
 else matrix[i, j] = rand.Next(2) == 0 ? rand.Next(10, 50) : Int32.**MaxValue**/2;  
 }  
 }  
 return matrix;  
 }  
 }  
}

using System;  
using System.IO;  
  
namespace Lab4  
{  
 public class FloydWarshallAlgorithm  
 {  
 public static int[,] GetPathMatrix(ref int[,] matrix)  
 {  
 int[,] pathMatrix = new int[matrix.GetLength(0), matrix.GetLength(1)];  
 for (int i = 0; i < pathMatrix.GetLength(0); i++)  
 {  
 for (int j = 0; j < pathMatrix.GetLength(1) ; j++)  
 {  
 if (matrix[i, j] is > 0 and < int.**MaxValue**/2) pathMatrix[i, j] = i;  
 }  
 }  
 for (int m = 0; m < matrix.GetLength(0); m++)  
 {  
 for (int i = 0; i < matrix.GetLength(0); i++)  
 {  
 if (i == m) continue;  
 for (int j = 0; j < matrix.GetLength(1) ; j++)  
 {  
 if(i == j || j == m) continue;  
 if (matrix[i, m] + matrix[m, j] < matrix[i, j])  
 {  
 pathMatrix[i, j] = m;  
 matrix[i, j] = matrix[i, m] + matrix[m, j];  
 }  
 }  
 }  
 }  
 return pathMatrix;  
 }  
 }  
}

using System;  
  
namespace Lab4  
{  
 public class DisplayOperations  
 {  
 public static void DisplayMatrix(int[,] matrix)  
 {  
 for (int i = 0; i < matrix.GetLength(0); i++)  
 {  
 for (int j = 0; j < matrix.GetLength(1); j++)  
 {  
   
 Console.Write((matrix[i,j] == int.**MaxValue**/2?"inf":$"{matrix[i,j],-3}") + " ");  
 }  
  
 Console.WriteLine();  
 }  
 }  
  
 public static string TracePath(int[,] pathMatrix, int startPoint, int endPoint)  
 {  
 if (pathMatrix[startPoint, endPoint] >= Int32.**MaxValue** / 2) return "";  
 string path = (endPoint+1).ToString();  
 int iterativeVertice = endPoint;  
 while (iterativeVertice!=startPoint)  
 {  
 iterativeVertice = pathMatrix[startPoint, iterativeVertice];  
 path = (iterativeVertice + 1) + " -> " + path;  
 }  
  
 return path;  
 }  
 }  
}

using System;  
  
namespace Lab4  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 int n = InputOperations.GetDimensions();  
 int[,] matrix;  
 if (InputOperations.IsRandomized())  
 {  
 matrix = MatrixRandomizer.CreateRandomMatrix(n);  
 }  
 else  
 {  
 matrix = InputOperations.GetManualInput(n);  
 }  
 DisplayOperations.DisplayMatrix(matrix);  
 int[,] pathMatrix = FloydWarshallAlgorithm.GetPathMatrix(ref matrix);  
 Console.WriteLine("New matrix:");  
 DisplayOperations.DisplayMatrix(matrix);  
 (int startPoint, int endPoint) = InputOperations.GetEntryPointsIndexes(n);  
 if (matrix[startPoint, endPoint] < Int32.**MaxValue** / 2)  
 {  
 Console.WriteLine($"Distance is {matrix[startPoint,endPoint]}");  
 Console.WriteLine("Path: "+DisplayOperations.TracePath(pathMatrix, startPoint, endPoint));  
 }  
 else Console.WriteLine("No path found..");  
 }  
 }  
}

* + 1. Приклад роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми для графів на 7 і

15 вершин відповідно.

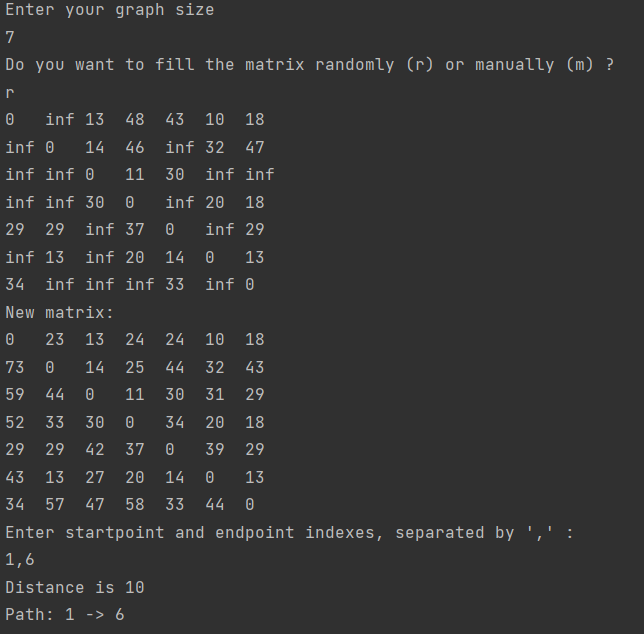
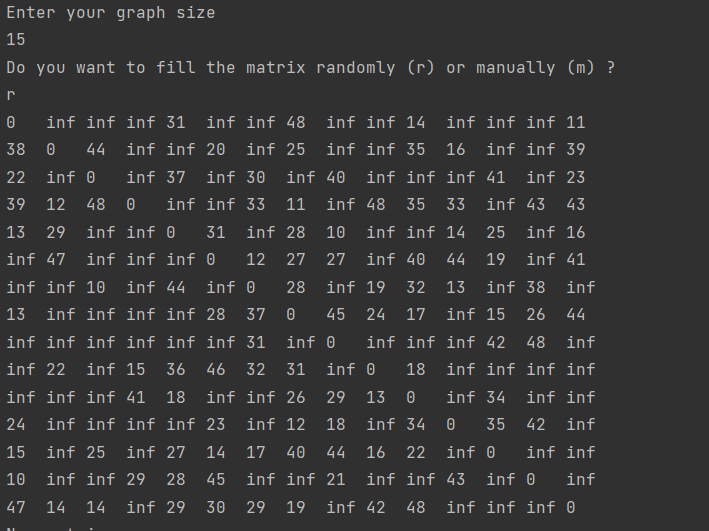


Рисунок 3.1 – Приклад роботи програми для 7 вершин



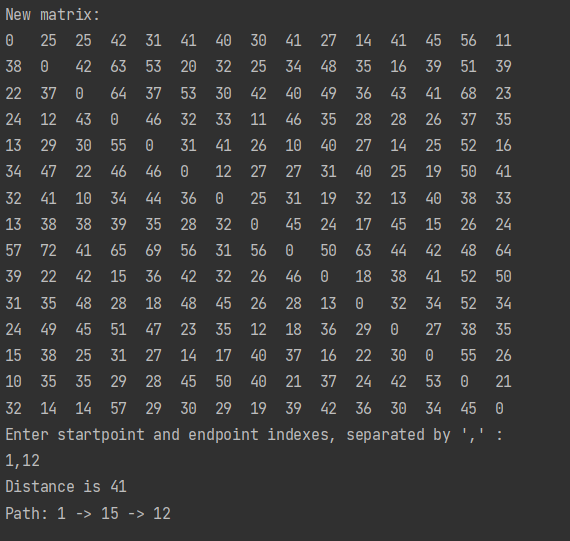


Рисунок 3.2 – Приклад роботи програми для 15 вершин

* 1. Розв’язання задачі вручну

На рисунку 3.3 наведено граф для розв’язання задачі вручну.

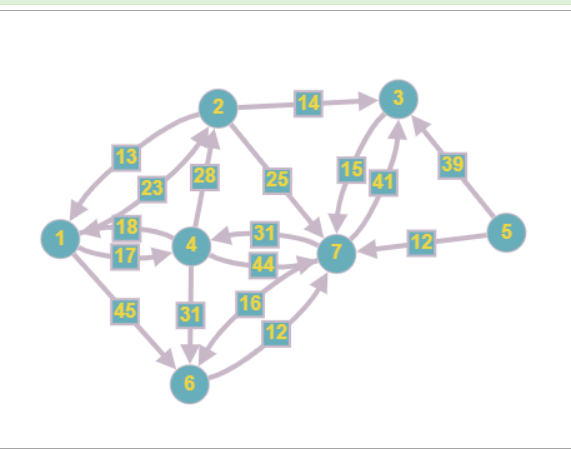


Рисунок 3.3 – Граф для розв'язання задачі алгоритмом Флойда-Уоршелла (вбудований спосіб визначення шляхів).

Початкова матриця вагів:

0 23 inf 17 inf 45 inf

13 0 14 inf inf inf 25

inf inf 0 inf inf inf 15

18 28 inf 0 inf 31 44

inf inf 39 inf 0 inf 12

inf inf inf inf inf 0 12

inf inf 41 31 inf 16 0

Початкова матриця шляхів:

0 0 0 0 0 0 0

1 0 1 0 0 0 1

0 0 0 0 0 0 2

3 3 0 0 0 3 3

0 0 4 0 0 0 4

0 0 0 0 0 0 5

0 0 6 6 0 6 0

Iteration 1. m = 1

matrix[1, 0] (13) + matrix[0, 2] (inf) >= matrix[1, 2] (14).

matrix[1, 0] (13) + matrix[0, 3] (17) < matrix[1, 3] (inf)

pathMatrix[1, 3] = 0

matrix[1, 3] = matrix[1, 0] + matrix[0, 3] = 30

matrix[1, 0] (13) + matrix[0, 4] (inf) >= matrix[1, 4] (inf).

matrix[1, 0] (13) + matrix[0, 5] (45) < matrix[1, 5] (inf)

pathMatrix[1, 5] = 0

matrix[1, 5] = matrix[1, 0] + matrix[0, 5] = 58

matrix[1, 0] (13) + matrix[0, 6] (inf) >= matrix[1, 6] (25).

matrix[2, 0] (inf) + matrix[0, 1] (23) >= matrix[2, 1] (inf).

matrix[2, 0] (inf) + matrix[0, 3] (17) >= matrix[2, 3] (inf).

matrix[2, 0] (inf) + matrix[0, 4] (inf) >= matrix[2, 4] (inf).

matrix[2, 0] (inf) + matrix[0, 5] (45) >= matrix[2, 5] (inf).

matrix[2, 0] (inf) + matrix[0, 6] (inf) >= matrix[2, 6] (15).

matrix[3, 0] (18) + matrix[0, 1] (23) >= matrix[3, 1] (28).

matrix[3, 0] (18) + matrix[0, 2] (inf) >= matrix[3, 2] (inf).

matrix[3, 0] (18) + matrix[0, 4] (inf) >= matrix[3, 4] (inf).

matrix[3, 0] (18) + matrix[0, 5] (45) >= matrix[3, 5] (31).

matrix[3, 0] (18) + matrix[0, 6] (inf) >= matrix[3, 6] (44).

matrix[4, 0] (inf) + matrix[0, 1] (23) >= matrix[4, 1] (inf).

matrix[4, 0] (inf) + matrix[0, 2] (inf) >= matrix[4, 2] (39).

matrix[4, 0] (inf) + matrix[0, 3] (17) >= matrix[4, 3] (inf).

matrix[4, 0] (inf) + matrix[0, 5] (45) >= matrix[4, 5] (inf).

matrix[4, 0] (inf) + matrix[0, 6] (inf) >= matrix[4, 6] (12).

matrix[5, 0] (inf) + matrix[0, 1] (23) >= matrix[5, 1] (inf).

matrix[5, 0] (inf) + matrix[0, 2] (inf) >= matrix[5, 2] (inf).

matrix[5, 0] (inf) + matrix[0, 3] (17) >= matrix[5, 3] (inf).

matrix[5, 0] (inf) + matrix[0, 4] (inf) >= matrix[5, 4] (inf).

matrix[5, 0] (inf) + matrix[0, 6] (inf) >= matrix[5, 6] (12).

matrix[6, 0] (inf) + matrix[0, 1] (23) >= matrix[6, 1] (inf).

matrix[6, 0] (inf) + matrix[0, 2] (inf) >= matrix[6, 2] (41).

matrix[6, 0] (inf) + matrix[0, 3] (17) >= matrix[6, 3] (31).

matrix[6, 0] (inf) + matrix[0, 4] (inf) >= matrix[6, 4] (inf).

matrix[6, 0] (inf) + matrix[0, 5] (45) >= matrix[6, 5] (16).

Поточна матриця вагів:

0 23 inf 17 inf 45 inf

13 0 14 30 inf 58 25

inf inf 0 inf inf inf 15

18 28 inf 0 inf 31 44

inf inf 39 inf 0 inf 12

inf inf inf inf inf 0 12

inf inf 41 31 inf 16 0

Поточна матриця шляхів:

0 0 0 0 0 0 0

1 0 1 0 0 0 1

0 0 0 0 0 0 2

3 3 0 0 0 3 3

0 0 4 0 0 0 4

0 0 0 0 0 0 5

0 0 6 6 0 6 0

Iteration 2. m = 2

matrix[0, 1] (23) + matrix[1, 2] (14) < matrix[0, 2] (inf)

pathMatrix[0, 2] = 1

matrix[0, 2] = matrix[0, 1] + matrix[1, 2] = 37

matrix[0, 1] (23) + matrix[1, 3] (30) >= matrix[0, 3] (17).

matrix[0, 1] (23) + matrix[1, 4] (inf) >= matrix[0, 4] (inf).

matrix[0, 1] (23) + matrix[1, 5] (58) >= matrix[0, 5] (45).

matrix[0, 1] (23) + matrix[1, 6] (25) < matrix[0, 6] (inf)

pathMatrix[0, 6] = 1

matrix[0, 6] = matrix[0, 1] + matrix[1, 6] = 48

matrix[2, 1] (inf) + matrix[1, 0] (13) >= matrix[2, 0] (inf).

matrix[2, 1] (inf) + matrix[1, 3] (30) >= matrix[2, 3] (inf).

matrix[2, 1] (inf) + matrix[1, 4] (inf) >= matrix[2, 4] (inf).

matrix[2, 1] (inf) + matrix[1, 5] (58) >= matrix[2, 5] (inf).

matrix[2, 1] (inf) + matrix[1, 6] (25) >= matrix[2, 6] (15).

matrix[3, 1] (28) + matrix[1, 0] (13) >= matrix[3, 0] (18).

matrix[3, 1] (28) + matrix[1, 2] (14) < matrix[3, 2] (inf)

pathMatrix[3, 2] = 1

matrix[3, 2] = matrix[3, 1] + matrix[1, 2] = 42

matrix[3, 1] (28) + matrix[1, 4] (inf) >= matrix[3, 4] (inf).

matrix[3, 1] (28) + matrix[1, 5] (58) >= matrix[3, 5] (31).

matrix[3, 1] (28) + matrix[1, 6] (25) >= matrix[3, 6] (44).

matrix[4, 1] (inf) + matrix[1, 0] (13) >= matrix[4, 0] (inf).

matrix[4, 1] (inf) + matrix[1, 2] (14) >= matrix[4, 2] (39).

matrix[4, 1] (inf) + matrix[1, 3] (30) >= matrix[4, 3] (inf).

matrix[4, 1] (inf) + matrix[1, 5] (58) >= matrix[4, 5] (inf).

matrix[4, 1] (inf) + matrix[1, 6] (25) >= matrix[4, 6] (12).

matrix[5, 1] (inf) + matrix[1, 0] (13) >= matrix[5, 0] (inf).

matrix[5, 1] (inf) + matrix[1, 2] (14) >= matrix[5, 2] (inf).

matrix[5, 1] (inf) + matrix[1, 3] (30) >= matrix[5, 3] (inf).

matrix[5, 1] (inf) + matrix[1, 4] (inf) >= matrix[5, 4] (inf).

matrix[5, 1] (inf) + matrix[1, 6] (25) >= matrix[5, 6] (12).

matrix[6, 1] (inf) + matrix[1, 0] (13) >= matrix[6, 0] (inf).

matrix[6, 1] (inf) + matrix[1, 2] (14) >= matrix[6, 2] (41).

matrix[6, 1] (inf) + matrix[1, 3] (30) >= matrix[6, 3] (31).

matrix[6, 1] (inf) + matrix[1, 4] (inf) >= matrix[6, 4] (inf).

matrix[6, 1] (inf) + matrix[1, 5] (58) >= matrix[6, 5] (16).

Поточна матриця вагів:

0 23 37 17 inf 45 48

13 0 14 30 inf 58 25

inf inf 0 inf inf inf 15

18 28 42 0 inf 31 44

inf inf 39 inf 0 inf 12

inf inf inf inf inf 0 12

inf inf 41 31 inf 16 0

Поточна матриця шляхів:

0 0 1 0 0 0 1

1 0 1 0 0 0 1

0 0 0 0 0 0 2

3 3 1 0 0 3 3

0 0 4 0 0 0 4

0 0 0 0 0 0 5

0 0 6 6 0 6 0

Iteration 3. m = 3

matrix[0, 2] (37) + matrix[2, 1] (inf) >= matrix[0, 1] (23).

matrix[0, 2] (37) + matrix[2, 3] (inf) >= matrix[0, 3] (17).

matrix[0, 2] (37) + matrix[2, 4] (inf) >= matrix[0, 4] (inf).

matrix[0, 2] (37) + matrix[2, 5] (inf) >= matrix[0, 5] (45).

matrix[0, 2] (37) + matrix[2, 6] (15) >= matrix[0, 6] (48).

matrix[1, 2] (14) + matrix[2, 0] (inf) >= matrix[1, 0] (13).

matrix[1, 2] (14) + matrix[2, 3] (inf) >= matrix[1, 3] (30).

matrix[1, 2] (14) + matrix[2, 4] (inf) >= matrix[1, 4] (inf).

matrix[1, 2] (14) + matrix[2, 5] (inf) >= matrix[1, 5] (58).

matrix[1, 2] (14) + matrix[2, 6] (15) >= matrix[1, 6] (25).

matrix[3, 2] (42) + matrix[2, 0] (inf) >= matrix[3, 0] (18).

matrix[3, 2] (42) + matrix[2, 1] (inf) >= matrix[3, 1] (28).

matrix[3, 2] (42) + matrix[2, 4] (inf) >= matrix[3, 4] (inf).

matrix[3, 2] (42) + matrix[2, 5] (inf) >= matrix[3, 5] (31).

matrix[3, 2] (42) + matrix[2, 6] (15) >= matrix[3, 6] (44).

matrix[4, 2] (39) + matrix[2, 0] (inf) >= matrix[4, 0] (inf).

matrix[4, 2] (39) + matrix[2, 1] (inf) >= matrix[4, 1] (inf).

matrix[4, 2] (39) + matrix[2, 3] (inf) >= matrix[4, 3] (inf).

matrix[4, 2] (39) + matrix[2, 5] (inf) >= matrix[4, 5] (inf).

matrix[4, 2] (39) + matrix[2, 6] (15) >= matrix[4, 6] (12).

matrix[5, 2] (inf) + matrix[2, 0] (inf) >= matrix[5, 0] (inf).

matrix[5, 2] (inf) + matrix[2, 1] (inf) >= matrix[5, 1] (inf).

matrix[5, 2] (inf) + matrix[2, 3] (inf) >= matrix[5, 3] (inf).

matrix[5, 2] (inf) + matrix[2, 4] (inf) >= matrix[5, 4] (inf).

matrix[5, 2] (inf) + matrix[2, 6] (15) >= matrix[5, 6] (12).

matrix[6, 2] (41) + matrix[2, 0] (inf) >= matrix[6, 0] (inf).

matrix[6, 2] (41) + matrix[2, 1] (inf) >= matrix[6, 1] (inf).

matrix[6, 2] (41) + matrix[2, 3] (inf) >= matrix[6, 3] (31).

matrix[6, 2] (41) + matrix[2, 4] (inf) >= matrix[6, 4] (inf).

matrix[6, 2] (41) + matrix[2, 5] (inf) >= matrix[6, 5] (16).

Поточна матриця вагів:

0 23 37 17 inf 45 48

13 0 14 30 inf 58 25

inf inf 0 inf inf inf 15

18 28 42 0 inf 31 44

inf inf 39 inf 0 inf 12

inf inf inf inf inf 0 12

inf inf 41 31 inf 16 0

Поточна матриця шляхів:

0 0 1 0 0 0 1

1 0 1 0 0 0 1

0 0 0 0 0 0 2

3 3 1 0 0 3 3

0 0 4 0 0 0 4

0 0 0 0 0 0 5

0 0 6 6 0 6 0

Iteration 4. m = 4

matrix[0, 3] (17) + matrix[3, 1] (28) >= matrix[0, 1] (23).

matrix[0, 3] (17) + matrix[3, 2] (42) >= matrix[0, 2] (37).

matrix[0, 3] (17) + matrix[3, 4] (inf) >= matrix[0, 4] (inf).

matrix[0, 3] (17) + matrix[3, 5] (31) >= matrix[0, 5] (45).

matrix[0, 3] (17) + matrix[3, 6] (44) >= matrix[0, 6] (48).

matrix[1, 3] (30) + matrix[3, 0] (18) >= matrix[1, 0] (13).

matrix[1, 3] (30) + matrix[3, 2] (42) >= matrix[1, 2] (14).

matrix[1, 3] (30) + matrix[3, 4] (inf) >= matrix[1, 4] (inf).

matrix[1, 3] (30) + matrix[3, 5] (31) >= matrix[1, 5] (58).

matrix[1, 3] (30) + matrix[3, 6] (44) >= matrix[1, 6] (25).

matrix[2, 3] (inf) + matrix[3, 0] (18) >= matrix[2, 0] (inf).

matrix[2, 3] (inf) + matrix[3, 1] (28) >= matrix[2, 1] (inf).

matrix[2, 3] (inf) + matrix[3, 4] (inf) >= matrix[2, 4] (inf).

matrix[2, 3] (inf) + matrix[3, 5] (31) >= matrix[2, 5] (inf).

matrix[2, 3] (inf) + matrix[3, 6] (44) >= matrix[2, 6] (15).

matrix[4, 3] (inf) + matrix[3, 0] (18) >= matrix[4, 0] (inf).

matrix[4, 3] (inf) + matrix[3, 1] (28) >= matrix[4, 1] (inf).

matrix[4, 3] (inf) + matrix[3, 2] (42) >= matrix[4, 2] (39).

matrix[4, 3] (inf) + matrix[3, 5] (31) >= matrix[4, 5] (inf).

matrix[4, 3] (inf) + matrix[3, 6] (44) >= matrix[4, 6] (12).

matrix[5, 3] (inf) + matrix[3, 0] (18) >= matrix[5, 0] (inf).

matrix[5, 3] (inf) + matrix[3, 1] (28) >= matrix[5, 1] (inf).

matrix[5, 3] (inf) + matrix[3, 2] (42) >= matrix[5, 2] (inf).

matrix[5, 3] (inf) + matrix[3, 4] (inf) >= matrix[5, 4] (inf).

matrix[5, 3] (inf) + matrix[3, 6] (44) >= matrix[5, 6] (12).

matrix[6, 3] (31) + matrix[3, 0] (18) < matrix[6, 0] (inf)

pathMatrix[6, 0] = 3

matrix[6, 0] = matrix[6, 3] + matrix[3, 0] = 49

matrix[6, 3] (31) + matrix[3, 1] (28) < matrix[6, 1] (inf)

pathMatrix[6, 1] = 3

matrix[6, 1] = matrix[6, 3] + matrix[3, 1] = 59

matrix[6, 3] (31) + matrix[3, 2] (42) >= matrix[6, 2] (41).

matrix[6, 3] (31) + matrix[3, 4] (inf) >= matrix[6, 4] (inf).

matrix[6, 3] (31) + matrix[3, 5] (31) >= matrix[6, 5] (16).

Поточна матриця вагів:

0 23 37 17 inf 45 48

13 0 14 30 inf 58 25

inf inf 0 inf inf inf 15

18 28 42 0 inf 31 44

inf inf 39 inf 0 inf 12

inf inf inf inf inf 0 12

49 59 41 31 inf 16 0

Поточна матриця шляхів:

0 0 1 0 0 0 1

1 0 1 0 0 0 1

0 0 0 0 0 0 2

3 3 1 0 0 3 3

0 0 4 0 0 0 4

0 0 0 0 0 0 5

3 3 6 6 0 6 0

Iteration 5. m = 5

matrix[0, 4] (inf) + matrix[4, 1] (inf) >= matrix[0, 1] (23).

matrix[0, 4] (inf) + matrix[4, 2] (39) >= matrix[0, 2] (37).

matrix[0, 4] (inf) + matrix[4, 3] (inf) >= matrix[0, 3] (17).

matrix[0, 4] (inf) + matrix[4, 5] (inf) >= matrix[0, 5] (45).

matrix[0, 4] (inf) + matrix[4, 6] (12) >= matrix[0, 6] (48).

matrix[1, 4] (inf) + matrix[4, 0] (inf) >= matrix[1, 0] (13).

matrix[1, 4] (inf) + matrix[4, 2] (39) >= matrix[1, 2] (14).

matrix[1, 4] (inf) + matrix[4, 3] (inf) >= matrix[1, 3] (30).

matrix[1, 4] (inf) + matrix[4, 5] (inf) >= matrix[1, 5] (58).

matrix[1, 4] (inf) + matrix[4, 6] (12) >= matrix[1, 6] (25).

matrix[2, 4] (inf) + matrix[4, 0] (inf) >= matrix[2, 0] (inf).

matrix[2, 4] (inf) + matrix[4, 1] (inf) >= matrix[2, 1] (inf).

matrix[2, 4] (inf) + matrix[4, 3] (inf) >= matrix[2, 3] (inf).

matrix[2, 4] (inf) + matrix[4, 5] (inf) >= matrix[2, 5] (inf).

matrix[2, 4] (inf) + matrix[4, 6] (12) >= matrix[2, 6] (15).

matrix[3, 4] (inf) + matrix[4, 0] (inf) >= matrix[3, 0] (18).

matrix[3, 4] (inf) + matrix[4, 1] (inf) >= matrix[3, 1] (28).

matrix[3, 4] (inf) + matrix[4, 2] (39) >= matrix[3, 2] (42).

matrix[3, 4] (inf) + matrix[4, 5] (inf) >= matrix[3, 5] (31).

matrix[3, 4] (inf) + matrix[4, 6] (12) >= matrix[3, 6] (44).

matrix[5, 4] (inf) + matrix[4, 0] (inf) >= matrix[5, 0] (inf).

matrix[5, 4] (inf) + matrix[4, 1] (inf) >= matrix[5, 1] (inf).

matrix[5, 4] (inf) + matrix[4, 2] (39) >= matrix[5, 2] (inf).

matrix[5, 4] (inf) + matrix[4, 3] (inf) >= matrix[5, 3] (inf).

matrix[5, 4] (inf) + matrix[4, 6] (12) >= matrix[5, 6] (12).

matrix[6, 4] (inf) + matrix[4, 0] (inf) >= matrix[6, 0] (49).

matrix[6, 4] (inf) + matrix[4, 1] (inf) >= matrix[6, 1] (59).

matrix[6, 4] (inf) + matrix[4, 2] (39) >= matrix[6, 2] (41).

matrix[6, 4] (inf) + matrix[4, 3] (inf) >= matrix[6, 3] (31).

matrix[6, 4] (inf) + matrix[4, 5] (inf) >= matrix[6, 5] (16).

Поточна матриця вагів:

0 23 37 17 inf 45 48

13 0 14 30 inf 58 25

inf inf 0 inf inf inf 15

18 28 42 0 inf 31 44

inf inf 39 inf 0 inf 12

inf inf inf inf inf 0 12

49 59 41 31 inf 16 0

Поточна матриця шляхів:

0 0 1 0 0 0 1

1 0 1 0 0 0 1

0 0 0 0 0 0 2

3 3 1 0 0 3 3

0 0 4 0 0 0 4

0 0 0 0 0 0 5

3 3 6 6 0 6 0

Iteration 6. m = 6

matrix[0, 5] (45) + matrix[5, 1] (inf) >= matrix[0, 1] (23).

matrix[0, 5] (45) + matrix[5, 2] (inf) >= matrix[0, 2] (37).

matrix[0, 5] (45) + matrix[5, 3] (inf) >= matrix[0, 3] (17).

matrix[0, 5] (45) + matrix[5, 4] (inf) >= matrix[0, 4] (inf).

matrix[0, 5] (45) + matrix[5, 6] (12) >= matrix[0, 6] (48).

matrix[1, 5] (58) + matrix[5, 0] (inf) >= matrix[1, 0] (13).

matrix[1, 5] (58) + matrix[5, 2] (inf) >= matrix[1, 2] (14).

matrix[1, 5] (58) + matrix[5, 3] (inf) >= matrix[1, 3] (30).

matrix[1, 5] (58) + matrix[5, 4] (inf) >= matrix[1, 4] (inf).

matrix[1, 5] (58) + matrix[5, 6] (12) >= matrix[1, 6] (25).

matrix[2, 5] (inf) + matrix[5, 0] (inf) >= matrix[2, 0] (inf).

matrix[2, 5] (inf) + matrix[5, 1] (inf) >= matrix[2, 1] (inf).

matrix[2, 5] (inf) + matrix[5, 3] (inf) >= matrix[2, 3] (inf).

matrix[2, 5] (inf) + matrix[5, 4] (inf) >= matrix[2, 4] (inf).

matrix[2, 5] (inf) + matrix[5, 6] (12) >= matrix[2, 6] (15).

matrix[3, 5] (31) + matrix[5, 0] (inf) >= matrix[3, 0] (18).

matrix[3, 5] (31) + matrix[5, 1] (inf) >= matrix[3, 1] (28).

matrix[3, 5] (31) + matrix[5, 2] (inf) >= matrix[3, 2] (42).

matrix[3, 5] (31) + matrix[5, 4] (inf) >= matrix[3, 4] (inf).

matrix[3, 5] (31) + matrix[5, 6] (12) < matrix[3, 6] (44)

pathMatrix[3, 6] = 5

matrix[3, 6] = matrix[3, 5] + matrix[5, 6] = 43

matrix[4, 5] (inf) + matrix[5, 0] (inf) >= matrix[4, 0] (inf).

matrix[4, 5] (inf) + matrix[5, 1] (inf) >= matrix[4, 1] (inf).

matrix[4, 5] (inf) + matrix[5, 2] (inf) >= matrix[4, 2] (39).

matrix[4, 5] (inf) + matrix[5, 3] (inf) >= matrix[4, 3] (inf).

matrix[4, 5] (inf) + matrix[5, 6] (12) >= matrix[4, 6] (12).

matrix[6, 5] (16) + matrix[5, 0] (inf) >= matrix[6, 0] (49).

matrix[6, 5] (16) + matrix[5, 1] (inf) >= matrix[6, 1] (59).

matrix[6, 5] (16) + matrix[5, 2] (inf) >= matrix[6, 2] (41).

matrix[6, 5] (16) + matrix[5, 3] (inf) >= matrix[6, 3] (31).

matrix[6, 5] (16) + matrix[5, 4] (inf) >= matrix[6, 4] (inf).

Поточна матриця вагів:

0 23 37 17 inf 45 48

13 0 14 30 inf 58 25

inf inf 0 inf inf inf 15

18 28 42 0 inf 31 43

inf inf 39 inf 0 inf 12

inf inf inf inf inf 0 12

49 59 41 31 inf 16 0

Поточна матриця шляхів:

0 0 1 0 0 0 1

1 0 1 0 0 0 1

0 0 0 0 0 0 2

3 3 1 0 0 3 5

0 0 4 0 0 0 4

0 0 0 0 0 0 5

3 3 6 6 0 6 0

Iteration 7. m = 7

matrix[0, 6] (48) + matrix[6, 1] (59) >= matrix[0, 1] (23).

matrix[0, 6] (48) + matrix[6, 2] (41) >= matrix[0, 2] (37).

matrix[0, 6] (48) + matrix[6, 3] (31) >= matrix[0, 3] (17).

matrix[0, 6] (48) + matrix[6, 4] (inf) >= matrix[0, 4] (inf).

matrix[0, 6] (48) + matrix[6, 5] (16) >= matrix[0, 5] (45).

matrix[1, 6] (25) + matrix[6, 0] (49) >= matrix[1, 0] (13).

matrix[1, 6] (25) + matrix[6, 2] (41) >= matrix[1, 2] (14).

matrix[1, 6] (25) + matrix[6, 3] (31) >= matrix[1, 3] (30).

matrix[1, 6] (25) + matrix[6, 4] (inf) >= matrix[1, 4] (inf).

matrix[1, 6] (25) + matrix[6, 5] (16) < matrix[1, 5] (58)

pathMatrix[1, 5] = 6

matrix[1, 5] = matrix[1, 6] + matrix[6, 5] = 41

matrix[2, 6] (15) + matrix[6, 0] (49) < matrix[2, 0] (inf)

pathMatrix[2, 0] = 6

matrix[2, 0] = matrix[2, 6] + matrix[6, 0] = 64

matrix[2, 6] (15) + matrix[6, 1] (59) < matrix[2, 1] (inf)

pathMatrix[2, 1] = 6

matrix[2, 1] = matrix[2, 6] + matrix[6, 1] = 74

matrix[2, 6] (15) + matrix[6, 3] (31) < matrix[2, 3] (inf)

pathMatrix[2, 3] = 6

matrix[2, 3] = matrix[2, 6] + matrix[6, 3] = 46

matrix[2, 6] (15) + matrix[6, 4] (inf) >= matrix[2, 4] (inf).

matrix[2, 6] (15) + matrix[6, 5] (16) < matrix[2, 5] (inf)

pathMatrix[2, 5] = 6

matrix[2, 5] = matrix[2, 6] + matrix[6, 5] = 31

matrix[3, 6] (43) + matrix[6, 0] (49) >= matrix[3, 0] (18).

matrix[3, 6] (43) + matrix[6, 1] (59) >= matrix[3, 1] (28).

matrix[3, 6] (43) + matrix[6, 2] (41) >= matrix[3, 2] (42).

matrix[3, 6] (43) + matrix[6, 4] (inf) >= matrix[3, 4] (inf).

matrix[3, 6] (43) + matrix[6, 5] (16) >= matrix[3, 5] (31).

matrix[4, 6] (12) + matrix[6, 0] (49) < matrix[4, 0] (inf)

pathMatrix[4, 0] = 6

matrix[4, 0] = matrix[4, 6] + matrix[6, 0] = 61

matrix[4, 6] (12) + matrix[6, 1] (59) < matrix[4, 1] (inf)

pathMatrix[4, 1] = 6

matrix[4, 1] = matrix[4, 6] + matrix[6, 1] = 71

matrix[4, 6] (12) + matrix[6, 2] (41) >= matrix[4, 2] (39).

matrix[4, 6] (12) + matrix[6, 3] (31) < matrix[4, 3] (inf)

pathMatrix[4, 3] = 6

matrix[4, 3] = matrix[4, 6] + matrix[6, 3] = 43

matrix[4, 6] (12) + matrix[6, 5] (16) < matrix[4, 5] (inf)

pathMatrix[4, 5] = 6

matrix[4, 5] = matrix[4, 6] + matrix[6, 5] = 28

matrix[5, 6] (12) + matrix[6, 0] (49) < matrix[5, 0] (inf)

pathMatrix[5, 0] = 6

matrix[5, 0] = matrix[5, 6] + matrix[6, 0] = 61

matrix[5, 6] (12) + matrix[6, 1] (59) < matrix[5, 1] (inf)

pathMatrix[5, 1] = 6

matrix[5, 1] = matrix[5, 6] + matrix[6, 1] = 71

matrix[5, 6] (12) + matrix[6, 2] (41) < matrix[5, 2] (inf)

pathMatrix[5, 2] = 6

matrix[5, 2] = matrix[5, 6] + matrix[6, 2] = 53

matrix[5, 6] (12) + matrix[6, 3] (31) < matrix[5, 3] (inf)

pathMatrix[5, 3] = 6

matrix[5, 3] = matrix[5, 6] + matrix[6, 3] = 43

matrix[5, 6] (12) + matrix[6, 4] (inf) >= matrix[5, 4] (inf).

Нова матриця вагів:

0 23 37 17 inf 45 48

13 0 14 30 inf 41 25

64 74 0 46 inf 31 15

18 28 42 0 inf 31 43

61 71 39 43 0 28 12

61 71 53 43 inf 0 12

49 59 41 31 inf 16 0

Поточна матриця шляхів:

0 0 1 0 0 0 1

1 0 1 0 0 6 1

6 6 0 6 0 6 2

3 3 1 0 0 3 5

6 6 4 6 0 6 4

6 6 6 6 0 0 5

3 3 6 6 0 6 0

Припустимо, що нам потрібно знайти шлях з першої в шосту точку.

Відстань 45

Шлях 1 -> 6

ВИСНОВОК

При виконанні даної лабораторної роботи я навчилася імплементувати вбудований спосіб визначення шляхів Флойда-Уоршелла та ознайомилася з додатковими прикладними алгоритмами на графах.

КРИТЕРІЇ ОЦІНЮВАННЯ

У випадку здачі лабораторної роботи до 30.03.2022 включно максимальний бал дорівнює – 5. Після 30.03.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

* псевдокод алгоритму – 10%;
* програмна реалізація алгоритму – 50%;
* розв’язання задачі вручну – 20%;
* відповідь на 3 теоретичні питання по темі роботи 15%
* висновок – 5%.

НЕ ДІЮТЬ