**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Информатики

****

**Отчет по лабораторной работе № 6**

по дисциплине «СиАОД»

на тему:

«**Сетевые алгоритмы. Динамические алгоритмы поиска путей.**»

Выполнила: студентка группы БВТ1802

Лаврухина Елена Павловна

Руководитель:

Кутейников Иван Алексеевич

Москва 2020

# **Цель работы**

Реализовать алгоритм поиска кратчайшего расстояния между двумя вершинами ориентированного взвешенного графа в соответствии с вариантом. Предусмотреть задание графа в виде матрицы смежности/инцидентности, читаемой из файла, либо графически с помощью пользовательского интерфейса. Разработать графический интерфейс пользователя с визуализацией графа и отображением кратчайшего расстояния между задаваемыми пользователем вершинами. По результатам работы проанализировать временную сложность работы заданного алгоритма в зависимости от числа узлов и ребер графа. Данные представить в виде таблицы.

Вариант № 19. Алгоритм Флойда-Уоршелла

Выполнение

Код программы

import javax.swing.\*;  
import java.awt.\*;  
import java.io.IOException;  
import java.nio.file.\*;  
import java.util.ArrayList;  
import java.util.List;  
public class lab6 extends JFrame {  
 public static final int *INF* = 999999999;  
 ArrayList<Point> point;  
 int size;  
 int n;  
 int[][] d;  
 *// рисование* lab6(int size, String path) throws IOException {  
 intil(path);  
 this.size = size;  
 setTitle("Граф");  
 setSize(size,size);  
 setVisible(true);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 }  
 private void intil(String str) throws IOException {  
 Path path = Paths.*get*(str);  
 List<String> contents = Files.*readAllLines*(path);  
 n = contents.size() - 1;  
 d = new int[n][n];  
 for(int i = 1; i < n + 1; i++){  
 String[] r = contents.get(i).split(" ");  
 for(int j = 1; j < n + 1; j++)  
 {  
 d[i- 1][j- 1 ] = *INF*;  
 int t = Integer.*parseInt*(r[j]);  
 if(i == j )  
 d[i- 1][j- 1] = 0;  
 else if(t != 0) {  
 d[i - 1][j - 1] = t;  
 }  
 }  
 }  
 }  
 public int min(int i1, int j1) {  
 int[][][] A = new int[n+1][n][n];  
 for (int i = 0; i < n; ++i)  
 for (int j = 0; j < n; ++j)  
 A[0][i][j] = d[i][j];  
 for (int k = 1; k <= n; k++)  
 for (int i = 0; i < n; i++)  
 for (int j = 0; j < n; j++)  
 A[k][i][j] = Math.*min*(A[k-1][i][j], A[k-1][i][k-1] + A[k-1][k-1][j]);  
 return A[n][i1][j1];  
 }  
 @Override  
 public void paint(Graphics g) {  
 int R = size / 2 - size / 5; *//радиус* int X = size / 2, Y = size / 2; *//координаты центра* point = new ArrayList<Point>();  
 g.setFont(new Font("TimesRoman", Font.*PLAIN*, 18));  
 double angle = 360.0 / n;  
 *//цикл создающий массив из точек* for (int i = 0; i < n; i++) {  
 int x = (int) (X + R \* Math.*cos*(Math.*toRadians*(angle \* i)));  
 int y = (int) (Y + R \* Math.*sin*(Math.*toRadians*(angle \* i)));  
 point.add(new Point(x, y));  
 }  
 for (int i = 0; i < n; i++) {  
 for(int j = i; j < n; j++) {  
 if(i != j && d[i][j] != *INF*) {  
 int x1 = point.get(i).x;  
 int y1 = point.get(i).y;  
 int x2 = point.get(j).x;  
 int y2 = point.get(j).y;  
 g.setColor(Color.*BLACK*);  
 g.drawLine(x1, y1, x2, y2);  
 }  
 }  
 }  
 int l = 0;  
 for (Point p : point) {  
 g.setColor(Color.*WHITE*);  
 g.fillOval(p.x - 25, p.y - 25, 50, 50);  
 g.setColor(Color.*BLACK*);  
 g.drawOval(p.x - 25, p.y - 25, 50, 50);  
 g.setColor(Color.*RED*);  
 g.drawString(String.*valueOf*(l + 1), p.x - 5, p.y + 5);  
 l++;  
 }  
 }  
 public static void main(String[] args) throws IOException {  
 long startTime = System.*nanoTime*();  
 lab6 st = new lab6(800,"C:\\Users\\Timelord\\Desktop\\dz\\2 k\\siaod\\lab6\\test2.txt");  
 long endTime = System.*nanoTime*()-startTime;  
 System.*out*.println("Минимальное расстояние = " + st.min(1,3)+"\nВремя выполнения = "+endTime+" нс");  
 }  
}

Результат программы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Число узлов** | **Минимальное расстояние** | **Число ребер** | **Время** | **Рисунок** |
| 6 | 9 | 9 | 154503300, нс. |  |
| 12 | 7 | 38 | 158864500,нс. |  |

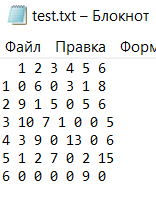


Рисунок 1 – Пример входных данных

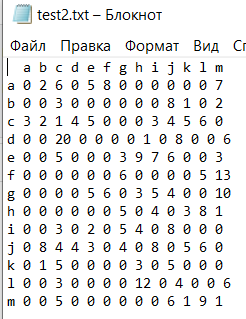


Рисунок 2 – Пример входных данных

# **Вывод**

В ходе работы мы научились реализовывать алгоритмы поиска кратчайшего пути и разрабатывать графический интерфейс пользователя. Проанализировав результаты программы, можно сделать вывод, что время поиска очень мало для такого объема вершин и ребер.