УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №7.1

по предмету «Основы алгоритмизации и программирования»

Вариант 18

Выполнил:

Машевский Д.В.

Гр. 351003

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Графе задан списками инцидентности. Разработать программу, находящую максимальный поток в сети. Граф визуализировать. Найденный поток выделить цветом

**Код программы Delphi:**

Unit Unit6Laba71;

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.Grids,

Vcl.StdCtrls,

Vcl.ExtCtrls,

Vcl.Menus;

Type

TIncidenceList = Array Of Array Of Integer;

TAdjacencyMatrix = Array Of Array Of Integer;

TAoI = Array Of Integer;

TMainForm = Class(TForm)

InputEdit: TEdit;

Button1: TButton;

Label1: TLabel;

Label2: TLabel;

StringGrid1: TStringGrid;

Button2: TButton;

Button3: TButton;

MainMenu: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

Procedure Button1Click(Sender: TObject);

Procedure Button2Click(Sender: TObject);

Procedure Button3Click(Sender: TObject);

Procedure InputEditChange(Sender: TObject);

Procedure InputEditKeyPress(Sender: TObject; Var Key: Char);

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Procedure StringGrid1KeyPress(Sender: TObject; Var Key: Char);

Procedure N2Click(Sender: TObject);

Procedure N5Click(Sender: TObject);

Procedure N4Click(Sender: TObject);

Private

{ Private declarations }

Public

IncidenceList: TIncidenceList;

End;

Var

MainForm: TMainForm;

Versh: Array Of Integer;

Matrix: Array Of Array Of Integer;

Answ: Array Of Integer;

Ans: Integer;

Ans\_vec: Array Of Integer;

N: Integer;

Const

SideLength: Integer = 40;

Implementation

{$R \*.dfm}

Uses

Unit7;

Function CheckData(): Boolean;

Begin

If MainForm.InputEdit.Text = '' Then

Begin

Result := False;

Exit;

End;

For Var I := 0 To MainForm.Stringgrid1.ColCount - 1 Do

If (MainForm.Stringgrid1.Cells[I, 0] = '') Or (MainForm.Stringgrid1.Cells[I, 1] = '') Then

Begin

Result := False;

Exit;

End;

Result := True;

End;

Function IsNumberInArray(Num: Integer): Boolean;

Var

I: Integer;

Begin

For I := 0 To High(Versh) Do

Begin

If Versh[I] = Num Then

Begin

Result := True;

Exit;

End;

End;

Result := False;

End;

Function GetIndex(Num: Integer): Integer;

Begin

For Var I := 0 To High(Versh) Do

If Versh[I] = Num Then

Begin

Result := I;

Exit;

End;

End;

Procedure Draw(Canvas: TCanvas);

Var

CenterX, CenterY, I: Integer;

Angle, Radius: Double;

Num: Integer;

Begin

Setlength(Versh, 0);

For I := 0 To 1 Do

For Var J := 0 To MainForm.Stringgrid1.ColCount - 1 Do

Begin

Num := Strtoint(MainForm.Stringgrid1.Cells[J, I]);

If Not IsNumberInArray(Num) Then

Begin

Setlength(Versh, Length(Versh) + 1);

Versh[High(Versh)] := Strtoint(MainForm.Stringgrid1.Cells[J, I]);

End;

End;

N := Length(Versh);

//Вычисляем центр окружности

CenterX := DrawForm.ClientWidth Div 2;

CenterY := DrawForm.ClientHeight Div 2;

//Вычисляем радиус окружности

Radius := N \* SideLength / (Pi);

For Var J := 0 To MainForm.Stringgrid1.ColCount - 1 Do

Begin

I := GetIndex(Strtoint(MainForm.Stringgrid1.Cells[J, 0]));

Angle := 2 \* Pi \* I / N;

Canvas.MoveTo(Round(CenterX + Radius \* Cos(Angle) - SideLength / 2) + 20,

Round(CenterY + Radius \* Sin(Angle) - SideLength / 2) + 20);

I := GetIndex(Strtoint(MainForm.Stringgrid1.Cells[J, 1]));

Angle := 2 \* Pi \* I / N;

Canvas.LineTo(Round(CenterX + Radius \* Cos(Angle) - SideLength / 2) + 20,

Round(CenterY + Radius \* Sin(Angle) - SideLength / 2) + 20);

End;

For I := 0 To N - 1 Do

Begin

Angle := 2 \* Pi \* I / N;

Canvas.Rectangle(Round(CenterX + Radius \* Cos(Angle) - SideLength / 2), Round(CenterY + Radius \* Sin(Angle) - SideLength / 2),

Round(CenterX + Radius \* Cos(Angle) + SideLength / 2), Round(CenterY + Radius \* Sin(Angle) + SideLength / 2));

Canvas.TextOut(Round(CenterX + Radius \* Cos(Angle) - SideLength / 2) + 5, Round(CenterY + Radius \* Sin(Angle) - SideLength / 2) + 5,

Inttostr(Versh[I]))

End;

End;

Procedure Draw2(Canvas: TCanvas);

Var

CenterX, CenterY: Integer;

Angle, Radius: Double;

Int: Integer;

Begin

Canvas.Pen.Color := ClRed;

CenterX := DrawForm.ClientWidth Div 2;

CenterY := DrawForm.ClientHeight Div 2;

//Вычисляем радиус окружности

Radius := N \* SideLength / (Pi);

For Var I := 0 To High(Answ) Do

BEgin

Int := GetIndex(Answ[I]);

Angle := 2 \* Pi \* Int / N;

Canvas.Rectangle(Round(CenterX + Radius \* Cos(Angle) - SideLength / 2), Round(CenterY + Radius \* Sin(Angle) - SideLength / 2),

Round(CenterX + Radius \* Cos(Angle) + SideLength / 2), Round(CenterY + Radius \* Sin(Angle) + SideLength / 2));

Canvas.TextOut(Round(CenterX + Radius \* Cos(Angle) - SideLength / 2) + 5, Round(CenterY + Radius \* Sin(Angle) - SideLength / 2) + 5,

Inttostr(Versh[Int]))

End;

Canvas.Pen.Color := ClBlack;

End;

Procedure TMainForm.Button1Click(Sender: TObject);

Begin

If InputEdit.Text <> '' Then

Begin

Stringgrid1.ColCount := Strtoint(InputEdit.Text);

Stringgrid1.RowCount := 2;

For Var I := 0 To Stringgrid1.Colcount - 1 Do

Begin

Stringgrid1.Cells[I, 0] := '';

Stringgrid1.Cells[I, 1] := '';

End;

StringGrid1.Options := StringGrid1.Options + [GoEditing];

End;

End;

Procedure TMainForm.Button2Click(Sender: TObject);

Begin

If CheckData() Then

Begin

DrawForm.Show;

Draw(DrawForm.Canvas);

End

Else

Application.MessageBox('Заполните все поля', 'Ошибка', 0);

End;

Function IsNotInclude(Arr: TAoI; Num: Integer): Boolean;

Begin

For Var I := 0 To High(Arr) Do

If Num = Arr[I] Then

Begin

Result := False;

Exit;

End;

Result := True;

End;

Procedure Recurse(Current\_vertex: Integer; Var Taken\_vertexes: TAoI);

Var

I, J: Integer;

Is\_all\_edges\_marked: Boolean;

Begin

If Length(Taken\_vertexes) < Ans Then

Begin

If Current\_vertex = N Then

Begin

Is\_all\_edges\_marked := True;

For I := 0 To N - 1 Do

Begin

For J := 0 To N - 1 Do

Begin

If Matrix[I][J] = 1 Then

Begin

If IsNotInclude(Taken\_vertexes, I) And IsNotInclude(Taken\_vertexes, J) Then

Begin

Is\_all\_edges\_marked := False;

Break;

End;

End;

End;

End;

If Is\_all\_edges\_marked Then

Begin

Ans := Length(Taken\_vertexes);

SetLength(Ans\_vec, Ans);

For I := 0 To Ans - 1 Do

Ans\_vec[I] := Taken\_vertexes[I];

End;

End

Else

Begin

Recurse(Current\_vertex + 1, Taken\_vertexes);

Setlength(Taken\_vertexes, Length(Taken\_vertexes) + 1);

Taken\_vertexes[High(Taken\_vertexes)] := Current\_vertex;

Recurse(Current\_vertex + 1, Taken\_vertexes);

Setlength(Taken\_vertexes, Length(Taken\_vertexes) - 1);

End;

End;

End;

Procedure TMainForm.Button3Click(Sender: TObject);

Var

I, J: Integer;

GO: TAoI;

Int: Integer;

Test: String;

Begin

If Not Checkdata Then

BEgin

Application.MessageBox('Заполните все поля', 'Ошибка', 0);

Exit;

End;

Setlength(Versh, 0);

For I := 0 To 1 Do

For J := 0 To MainForm.Stringgrid1.ColCount - 1 Do

Begin

Int := Strtoint(MainForm.Stringgrid1.Cells[J, I]);

If Not IsNumberInArray(Int) Then

Begin

Setlength(Versh, Length(Versh) + 1);

Versh[High(Versh)] := Strtoint(MainForm.Stringgrid1.Cells[J, I]);

End;

End;

N := Length(Versh);

Setlength(Matrix, N, N);

For I := 0 To N - 1 Do

For J := 0 To N - 1 Do

Matrix[I][J] := 0;

For J := 0 To MainForm.Stringgrid1.ColCount - 1 Do

Begin

Test := MainForm.Stringgrid1.Cells[J, 0] + MainForm.Stringgrid1.Cells[J, 1];

Matrix[GetIndex(Strtoint(MainForm.Stringgrid1.Cells[J, 0])), GetIndex(Strtoint(MainForm.Stringgrid1.Cells[J, 1]))] := 1;

Matrix[GetIndex(Strtoint(MainForm.Stringgrid1.Cells[J, 1])), GetIndex(Strtoint(MainForm.Stringgrid1.Cells[J, 0]))] := 1;

End;

Ans := N;

SetLength(Ans\_vec, Ans);

For I := 0 To Ans - 1 Do

Ans\_vec[I] := I;

Recurse(0, GO);

Setlength(Answ, Length(Ans\_vec));

For I := 0 To High(Ans\_vec) Do

Answ[I] := Versh[Ans\_vec[I]];

DrawForm.Show;

Draw(DrawForm.Canvas);

Draw2(DrawForm.Canvas);

End;

Procedure TMainForm.FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Begin

CanClose := MessageBox(MainForm.Handle, 'Хотите выйти?', 'Выход', MB\_YESNO + MB\_ICONQUESTION) = ID\_YES;

End;

Procedure TMainForm.InputEditChange(Sender: TObject);

Begin

For Var I := 0 To Stringgrid1.Colcount - 1 Do

Begin

Stringgrid1.Cells[I, 0] := '';

Stringgrid1.Cells[I, 1] := '';

End;

End;

Procedure TMainForm.InputEditKeyPress(Sender: TObject; Var Key: Char);

Begin

If InputEdit.SelStart < Length(InputEdit.Text) Then

InputEdit.Text := '';

If (Length(InputEdit.Text) = 0) And (Key = '0') Then

Key := #0;

If (InputEdit.Text = '0') And (Key = '0') Then

Key := #0;

If (Length(InputEdit.Text) > 1) And (Key <> #8) Then

Key := #0;

If Not(Key In ['0' .. '9', #8]) Then

Key := #0;

End;

Procedure TMainForm.N2Click(Sender: TObject);

Var

OpenDialog: TOpenDialog;

FileName: String;

FileReader: TextFile;

Curr: Integer;

Begin

OpenDialog := TOpenDialog.Create(Self);

Try

If OpenDialog.Execute Then

Begin

FileName := OpenDialog.FileName;

If (ExtractFileExt(FileName) = '.txt') Then

Begin

AssignFile(FileReader, FileName);

Reset(FileReader);

Read(FileReader, Curr);

InputEdit.Text := Inttostr(Curr);

Stringgrid1.ColCount := Curr;

For Var I := 0 To Curr - 1 Do

Begin

Read(FileReader, Curr);

MainForm.Stringgrid1.Cells[I, 0] := Inttostr(Curr);

Read(FileReader, Curr);

MainForm.Stringgrid1.Cells[I, 1] := Inttostr(Curr);

End;

CloseFile(FileReader);

End

Else

Application.MessageBox('Неправильный формат файла!', 'Предупреждение', 0);

End;

Finally

OpenDialog.Free;

End;

End;

Procedure TMainForm.N4Click(Sender: TObject);

Begin

Application.MessageBox('Введите количество ребер, а далее в таблицу заполните ребра, то есть 2 вершины, которые они связывают.',

'Инструкция', 0)

End;

Procedure TMainForm.N5Click(Sender: TObject);

Begin

Application.MessageBox('Машевский Даниил Витальевич, гр. 351003 Лабораторная 7.1 ', 0)

End;

Procedure TMainForm.StringGrid1KeyPress(Sender: TObject; Var Key: Char);

Var

ColIndex, RowIndex: Integer;

Begin

ColIndex := StringGrid1.Col; //Индекс выбранного столбца

RowIndex := StringGrid1.Row; //Индекс выбранной строки

If (Length(StringGrid1.Cells[ColIndex, RowIndex]) = 1) And (Key <> #8) Then

Key := #0;

If Not(Key In ['1' .. '9', #8]) Then

Key := #0;

End;

End.

**Код на Java:**

import java.util.ArrayList;

import java.util.Scanner;

public class Main {

static int ans;

static ArrayList<Integer> ansVec;

static int n;

static int[][] matrix;

public static void recurse(int currentVertex, ArrayList<Integer> takenVertices) {

if (takenVertices.size() >= ans) return;

if (currentVertex == n) {

boolean isAllEdgesMarked = true;

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

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

if (matrix[i][j] == 1) {

if (!takenVertices.contains(i) && !takenVertices.contains(j)) {

isAllEdgesMarked = false;

break;

}

}

}

}

if (isAllEdgesMarked) {

ans = takenVertices.size();

ansVec = new ArrayList<>(takenVertices);

}

} else {

recurse(currentVertex + 1, new ArrayList<>(takenVertices));

takenVertices.add(currentVertex);

recurse(currentVertex + 1, new ArrayList<>(takenVertices));

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Введите 1, если хотите считать данные с файла, 0 - если с

консоли");

int conf = scanner.nextInt();

int max = 0;

String path;

ArrayList<Integer> A = new ArrayList<>();

ArrayList<Integer> B = new ArrayList<>();

if (conf == 1) {

System.out.println("Введите путь к файлу");

scanner.nextLine(); // Consume the newline character

path = scanner.nextLine();

// Read from file and populate A and B

// ...

} else {

System.out.println("Введите количество ребер");

int len = scanner.nextInt();

for (int i = 0; i < len; ++i) {

int temp = scanner.nextInt();

if (max < temp) max = temp;

A.add(temp);

temp = scanner.nextInt();

if (max < temp) max = temp;

B.add(temp);

}

}

n = max;

matrix = new int[n][n];

for (int i = 0; i < A.size(); ++i) {

matrix[A.get(i) - 1][B.get(i) - 1] = 1;

matrix[B.get(i) - 1][A.get(i) - 1] = 1;

}

ans = n;

ansVec = new ArrayList<>();

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

ansVec.add(i);

}

ArrayList<Integer> GO = new ArrayList<>();

recurse(0, GO);

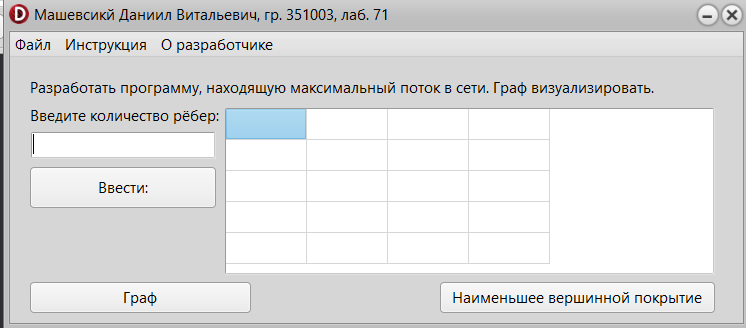
System.out.println("Помеченые вершины:");

for (int x : ansVec) System.out.print((x + 1) + " ");

}

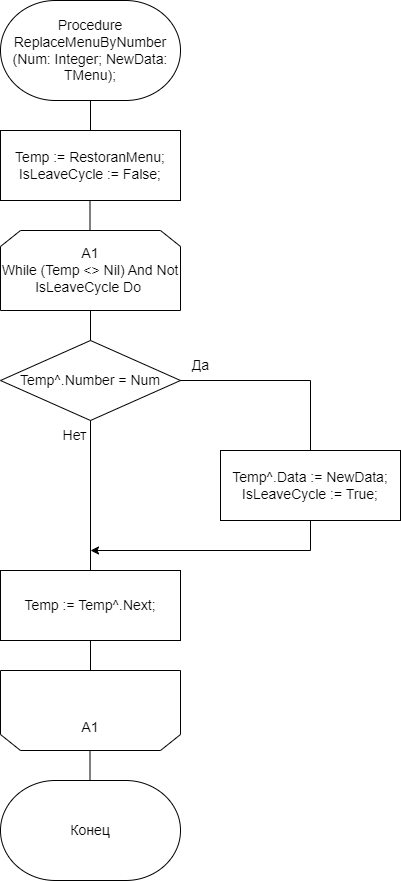
}

**Скриншоты:**

****

****

**Блок-схема:**

****

