УО «Белорусский государственный университет информатики и

радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №7.1

по предмету

Основы Алгоритмизации и Программирования

Вариант 15

**Выполнил**

**Коловайтис Н. А.**

**Проверила**

**Данилова Г.В.**

Группа:

**8**51001

Минск 2019

**Задание**

Граф задан матрицей инциденций. Разработать программу, реализующую нахождения остовного дерева в графе в графе. Граф визуализировать. Найденное дерево выделить цветом.

**Код программы**

**(Delphi)**

**unit** MainForm;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants,

FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Edit,

FMX.EditBox, FMX.SpinBox, FMX.Controls.Presentation, FMX.StdCtrls,

FMX.Objects, FMX.ScrollBox, FMX.Memo, FMX.Menus, MMSystem, FMX.Ani;

**type**

TMyPoint = **record**

X, Y: **Integer**;

**end**;

TLink = **record**

First, Second: **Integer**;

**end**;

TSpanningTree = **array** **of** TLink;

TGraphPoint = **record**

Text: **String**;

CordX: **Integer**;

CordY: **Integer**;

**end**;

TGraphLine = **record**

Text: **String**;

StartX: **Integer**;

StartY: **Integer**;

EndX: **Integer**;

EndY: **Integer** **end**;

TFormMain = **class**(TForm)PTable: TPanel;

SBPoints: TSpinBox;

SBLines: TSpinBox;

PNames: TPanel;

PBGraph: TPaintBox;

MIncidence: TMemo;

SaveDialog: TSaveDialog;

OpenDialog: TOpenDialog;

MainMenu: TMainMenu;

MIOpen: TMenuItem;

MISave: TMenuItem;

MIHelp: TMenuItem;

LPointsCount: TLabel;

LLinesCount: TLabel;

ImageBatman: TImage;

BLABatman: TBitmapListAnimation;

**procedure** SBPointsChange(Sender: **TObject**);

**procedure** FormCreate(Sender: **TObject**);

**procedure** SBLinesChange(Sender: **TObject**);

**procedure** CheckMatrixCorrect();

**procedure** CheckBoxChange(Sender: **TObject**);

**procedure** PBGraphPaint(Sender: **TObject**; Canvas: TCanvas);

**function** FindSpanningTree(): TSpanningTree;

**procedure** MIHelpClick(Sender: **TObject**);

**procedure** MIOpenClick(Sender: **TObject**);

**function** GetLinesCount(**var** InputFile: **TextFile**): **Integer**;

**procedure** MISaveClick(Sender: **TObject**);

**procedure** PartyMod();

**private**

**var**

ViewMatrix: **array** **of** **array** **of** TCheckBox;

NameLines, NamePoints: **array** **of** TLabel;

LinesList: **array** **of** TLink;

PointsCount: **Integer**;

CanDraw: **Boolean**;

**procedure** CheckFileCorrect(**var** InputFile: **TextFile**);

**const**

MaxLines: **array** [2 .. 9] **of** **Integer** = (1, 3, 6, 10, 15, 21, 28, 36);

SideOfCell = 20;

ColorGreen = $FF008000;

ColorBlack = $FF000000;

ColorWhite = $FFFFFFFF;

StandartOpacity = 1;

MaxPoint = 9;

MaxLine = 36;

StartX = 5;

**public**

**procedure** SetMatrixLength(PointsCount, LinesCount: **Integer**);

**procedure** AddCell(PointIndex, LineIndex: **Integer**);

**procedure** RemoveCell(PointIndex, LineIndex: **Integer**);

**procedure** ClearGraph();

**procedure** DrawPoint(Name: **String**; CordX, CordY: **Integer**; Color: **Cardinal**);

**procedure** DrawLine(StartX, StartY, EndX, EndY: **Integer**; Color: **Cardinal**);

**procedure** AddLine(Line: TLink);

**procedure** StartPlaySound();

**end**;

**var**

FormMain: TFormMain;

**implementation**

*{$R \*.fmx}*

**procedure** TFormMain.AddCell(PointIndex, LineIndex: **Integer**);

**begin**

**if** (ViewMatrix[PointIndex, LineIndex] = **nil**) **then**

**begin**

ViewMatrix[PointIndex, LineIndex] := TCheckBox.Create(PTable);

**with** ViewMatrix[PointIndex, LineIndex] **do**

**begin**

Parent := PTable;

Position.X := LineIndex \* SideOfCell;

Position.Y := PointIndex \* SideOfCell;

Width := SideOfCell;

Height := SideOfCell;

Visible := **true**;

OnChange := CheckBoxChange;

**end**;

**end**;

**end**;

**procedure** TFormMain.AddLine(Line: TLink);

**var**

Point1, Point2: **Integer**;

**begin**

Point1 := Line.First;

Point2 := Line.Second;

MIncidence.Lines[Point1] := MIncidence.Lines[Point1] +

IntToStr(Point2 + 1) + ', ';

MIncidence.Lines[Point2] := MIncidence.Lines[Point2] +

IntToStr(Point1 + 1) + ', ';

**end**;

**procedure** TFormMain.CheckBoxChange(Sender: **TObject**);

**begin**

CheckMatrixCorrect;

**end**;

**procedure** TFormMain.CheckFileCorrect(**var** InputFile: **TextFile**);

**begin**

**end**;

**procedure** TFormMain.CheckMatrixCorrect;

**var**

i, j: **Integer**;

IsCorrect: **Boolean**;

LineCount: **Integer**;

**begin**

IsCorrect := **true**;

PointsCount := Length(ViewMatrix);

**if** (PointsCount > 0) **then**

**begin**

LineCount := Length(ViewMatrix[0]);

SetLength(LinesList, LineCount);

**for** i := 0 **to** LineCount - 1 **do**

**begin**

j := 0;

**while** (j < PointsCount) **and** (**not** ViewMatrix[j][i].IsChecked) **do**

Inc(j);

LinesList[i].First := j;

Inc(j);

**while** (j < PointsCount) **and** (**not** ViewMatrix[j][i].IsChecked) **do**

Inc(j);

LinesList[i].Second := j;

**if** (j >= PointsCount) **then**

IsCorrect := **false**

**else**

**begin**

Inc(j);

**while** (j < PointsCount) **and** (**not** ViewMatrix[j][i].IsChecked) **do**

Inc(j);

**if** (j <> PointsCount) **then**

IsCorrect := **false**;

**end**;

**end**;

**end**;

CanDraw := IsCorrect;

PBGraph.Repaint;

MIncidence.Repaint;

**end**;

**procedure** TFormMain.ClearGraph;

**const**

Width = 281;

Height = 281;

**var**

i: **Integer**;

**begin**

PBGraph.Canvas.ClearRect(TRectF.Create(0, 0, Width, Height), ColorWhite);

**if** (CanDraw) **then**

**for** i := 0 **to** PointsCount - 1 **do**

MIncidence.Lines[i] := IntToStr(i + 1) + ': '

**else**

**for** i := 0 **to** PointsCount - 1 **do**

MIncidence.Lines[i] := '';

MISave.Enabled := **false**;

**end**;

**procedure** TFormMain.DrawLine(StartX, StartY, EndX, EndY: **Integer**;

Color: **Cardinal**);

**var**

StartPoint, EndPoint: TPointF;

**begin**

StartPoint := TPointF.Create(StartX, StartY);

EndPoint := TPointF.Create(EndX, EndY);

Canvas.Stroke.Color := Color;

**with** PBGraph.Canvas **do**

DrawLine(StartPoint, EndPoint, StandartOpacity);

**end**;

**procedure** TFormMain.DrawPoint(Name: **String**; CordX, CordY: **Integer**;

Color: **Cardinal**);

**Const**

Diametr = 20;

**var**

PlaceToDraw: TRectF;

**begin**

PlaceToDraw := TRectF.Create(TPointF.Create(CordX - Diametr / 2,

CordY - Diametr / 2), TPointF.Create(CordX + Diametr / 2,

CordY + Diametr / 2));

**with** PBGraph.Canvas **do**

**begin**

Fill.Color := ColorWhite;

FillEllipse(PlaceToDraw, StandartOpacity);

Stroke.Color := Color;

Fill.Color := Color;

DrawEllipse(PlaceToDraw, StandartOpacity);

FillText(PlaceToDraw, Name, **true**, StandartOpacity,

[TFillTextFlag.RightToLeft], TTextAlign.Center);

**end**;

**end**;

**function** TFormMain.FindSpanningTree: TSpanningTree;

**var**

SingleLine: TLink;

AnswerTree: TSpanningTree; *// В дереве хранятся ребра из списка*

WasPoints: **set** **of** **Byte**;

MinLinesCount, PointsFounded, LinesCount: **Integer**;

NotAnswer: **Boolean**;

CurrentIndex: **Integer**;

Point1, Point2: **Byte**;

WasGoWithoutChanges: **Integer**;

**begin**

LinesCount := Length(LinesList);

MinLinesCount := PointsCount - 1;

SetLength(AnswerTree, MinLinesCount);

AnswerTree[0] := LinesList[0];

WasPoints := [AnswerTree[0].First, AnswerTree[0].Second];

PointsFounded := 2;

NotAnswer := **true**;

CurrentIndex := 0;

WasGoWithoutChanges := 0;

**while** NotAnswer **do**

**begin**

Inc(CurrentIndex);

**if** (CurrentIndex = LinesCount) **then**

CurrentIndex := 0;

Point1 := LinesList[CurrentIndex].First;

Point2 := LinesList[CurrentIndex].Second;

**if** ((Point1 **in** WasPoints) **and** (**not**(Point2 **in** WasPoints))) **then**

**begin**

Include(WasPoints, Point2);

Inc(PointsFounded);

AnswerTree[PointsFounded - 2] := LinesList[CurrentIndex];

WasGoWithoutChanges := 0;

**end**

**else** **if** ((**not**(Point1 **in** WasPoints)) **and** (Point2 **in** WasPoints)) **then**

**begin**

Include(WasPoints, Point1);

Inc(PointsFounded);

AnswerTree[PointsFounded - 2] := LinesList[CurrentIndex];

WasGoWithoutChanges := 0;

**end**

**else**

**begin**

Inc(WasGoWithoutChanges);

**if** (WasGoWithoutChanges > LinesCount) **then**

**begin**

NotAnswer := **false**;

SetLength(AnswerTree, 0);

**end**;

**end**;

**if** (PointsFounded = PointsCount) **then**

NotAnswer := **false**;

**end**;

result := AnswerTree;

**end**;

**procedure** TFormMain.FormCreate(Sender: **TObject**);

**var**

i: **Integer**;

**begin**

SetLength(NameLines, MaxLine);

**for** i := 0 **to** MaxLine - 1 **do**

**begin**

NameLines[i] := TLabel.Create(PNames);

**with** NameLines[i] **do**

**begin**

Parent := PNames;

Position.X := (i + 1) \* SideOfCell;

Position.Y := 0;

Width := SideOfCell;

Height := SideOfCell;

Visible := **true**;

Text := IntToStr(i + 1);

**end**;

**end**;

SetLength(NamePoints, MaxPoint);

**for** i := 0 **to** MaxPoint - 1 **do**

**begin**

NameLines[i] := TLabel.Create(PNames);

**with** NameLines[i] **do**

**begin**

Parent := PNames;

Position.X := StartX;

Position.Y := (i + 1) \* SideOfCell;

Width := SideOfCell;

Height := SideOfCell;

Visible := **true**;

Text := IntToStr(i + 1);

**end**;

**end**;

SetMatrixLength(2, 1);

PartyMod;

**end**;

**function** TFormMain.GetLinesCount(**var** InputFile: **TextFile**): **Integer**;

**var**

LinesCount, PastLinesCount, NewPointsCount: **Integer**;

IntToRead: **Integer**;

**begin**

PastLinesCount := 0;

PointsCount := 0;

**while** **not** Eoln(InputFile) **do**

**begin**

**Read**(InputFile, IntToRead);

Inc(PastLinesCount);

**if** (**not**(IntToRead **in** [0 .. 1])) **then**

**raise** Exception.Create('Error');

**end**;

NewPointsCount := 1;

**while** **not** Eof(InputFile) **do**

**begin**

Readln(InputFile);

LinesCount := 0;

**while** **not** Eoln(InputFile) **do**

**begin**

**Read**(InputFile, IntToRead);

Inc(LinesCount);

**if** (**not**(IntToRead **in** [0 .. 1])) **then**

**raise** Exception.Create('Error');

**end**;

**if** (LinesCount <> PastLinesCount) **then**

**raise** Exception.Create('Error');

Inc(NewPointsCount);

**end**;

**if** (NewPointsCount <= MaxPoint) **and** (NewPointsCount >= 2) **and**

(MaxLines[NewPointsCount] >= PastLinesCount) **then**

**begin**

PointsCount := NewPointsCount;

result := PastLinesCount;

**end**

**else**

**raise** Exception.Create('Error');

**end**;

**procedure** TFormMain.MIHelpClick(Sender: **TObject**);

**const**

HelpMessage =

'Данная программа по матрице инциденций составляет графическое представление графа и список’+

'инцидентности. Если в графе возможно составить остовное дерево, оно будет выделено зеленым’+

‘цветом.';

**begin**

ShowMessage(HelpMessage);

**end**;

**procedure** TFormMain.MIOpenClick(Sender: **TObject**);

**const**

ErrorMessage = 'Произошла ошибка открытия файла.';

**var**

InputFile: **TextFile**;

LinesCount: **Integer**;

i: **Integer**;

j: **Integer**;

IntToRead: **Integer**;

**begin**

**if** OpenDialog.Execute **then**

**begin**

**try**

AssignFile(InputFile, OpenDialog.FileName);

Reset(InputFile);

LinesCount := GetLinesCount(InputFile);

Reset(InputFile);

SBPoints.Value := PointsCount;

SBLines.Value := LinesCount;

**for** i := 0 **to** PointsCount - 1 **do**

**for** j := 0 **to** LinesCount - 1 **do**

**begin**

**Read**(InputFile, IntToRead);

ViewMatrix[i][j].IsChecked := IntToRead = 1;

**end**;

**except**

ShowMessage(ErrorMessage);

**end**;

CloseFile(InputFile);

**end**;

**end**;

**procedure** TFormMain.MISaveClick(Sender: **TObject**);

**const**

ErrorMessage = 'Произошла ошибка сохранения в файл.';

**begin**

**if** SaveDialog.Execute **then**

**begin**

**try**

MIncidence.Lines.SaveToFile(SaveDialog.FileName);

**except**

ShowMessage(ErrorMessage);

**end**;

**end**;

**end**;

**procedure** TFormMain.PartyMod;

**begin**

StartPlaySound;

ImageBatman.Visible := **true**;

BLABatman.Enabled := **true**;

BLABatman.Start;

**end**;

**procedure** TFormMain.PBGraphPaint(Sender: **TObject**; Canvas: TCanvas);

**Const**

GraphRadious = 100;

StartX = 140;

StartY = 140;

**var**

CurrentLine: TLink;

PointsCords: **array** **of** TMyPoint;

DeltaAngle: **Extended**;

i: **Integer**;

SpanningTree: TSpanningTree;

**begin**

ClearGraph;

**If** (CanDraw) **then**

**begin**

PBGraph.BeginUpdate;

SetLength(PointsCords, PointsCount);

DeltaAngle := 2 \* Pi / PointsCount;

**for** i := 0 **to** PointsCount - 1 **do**

**begin**

PointsCords[i].X := StartX + round(GraphRadious \* cos(DeltaAngle \* i));

PointsCords[i].Y := StartY + round(GraphRadious \* sin(DeltaAngle \* i));

**end**;

**for** CurrentLine **in** LinesList **do**

**begin**

DrawLine(PointsCords[CurrentLine.First].X,

PointsCords[CurrentLine.First].Y, PointsCords[CurrentLine.Second].X,

PointsCords[CurrentLine.Second].Y, ColorBlack);

**end**;

**for** i := 0 **to** PointsCount - 1 **do**

**begin**

DrawPoint(IntToStr(i + 1), PointsCords[i].X, PointsCords[i].Y,

ColorBlack);

**end**;

SpanningTree := FindSpanningTree;

**for** CurrentLine **in** SpanningTree **do**

**begin**

DrawLine(PointsCords[CurrentLine.First].X,

PointsCords[CurrentLine.First].Y, PointsCords[CurrentLine.Second].X,

PointsCords[CurrentLine.Second].Y, ColorGreen);

DrawPoint(IntToStr(CurrentLine.First + 1),

PointsCords[CurrentLine.First].X, PointsCords[CurrentLine.First].Y,

ColorGreen);

DrawPoint(IntToStr(CurrentLine.Second + 1),

PointsCords[CurrentLine.Second].X, PointsCords[CurrentLine.Second].Y,

ColorGreen);

**end**;

PBGraph.EndUpdate;

**for** CurrentLine **in** LinesList **do**

**begin**

AddLine(CurrentLine);

**end**;

MISave.Enabled := **true**;

**end**;

**end**;

**procedure** TFormMain.RemoveCell(PointIndex, LineIndex: **Integer**);

**begin**

**if** (ViewMatrix[PointIndex, LineIndex] <> **nil**) **then**

ViewMatrix[PointIndex, LineIndex].Destroy;

**end**;

**procedure** TFormMain.SBLinesChange(Sender: **TObject**);

**var**

NewPointsCount: **Integer**;

NewLinesCount: **Integer**;

**begin**

NewPointsCount := round(SBPoints.Value);

NewLinesCount := round(SBLines.Value);

SetMatrixLength(NewPointsCount, NewLinesCount);

**end**;

**procedure** TFormMain.SBPointsChange(Sender: **TObject**);

**var**

NewPointsCount: **Integer**;

NewLinesCount: **Integer**;

**begin**

NewPointsCount := round(SBPoints.Value);

SBLines.Max := MaxLines[NewPointsCount];

NewLinesCount := round(SBLines.Value);

SetMatrixLength(NewPointsCount, NewLinesCount);

**end**;

**procedure** TFormMain.SetMatrixLength(PointsCount, LinesCount: **Integer**);

**var**

i: **Integer**;

j: **Integer**;

OldPointsCount, OldLinesCount: **Integer**;

**begin**

PTable.Width := LinesCount \* SideOfCell;

PTable.Height := PointsCount \* SideOfCell;

OldPointsCount := Length(ViewMatrix);

**if** (OldPointsCount <> 0) **then**

OldLinesCount := Length(ViewMatrix[0])

**else**

OldLinesCount := 0;

**if** (OldLinesCount < LinesCount) **then**

**begin**

SetLength(ViewMatrix, PointsCount, LinesCount);

**for** i := 0 **to** PointsCount - 1 **do**

**for** j := OldLinesCount **to** LinesCount - 1 **do**

**Self**.AddCell(i, j);

**end**

**else**

**begin**

**for** i := 0 **to** OldPointsCount - 1 **do**

**for** j := LinesCount **to** OldLinesCount - 1 **do**

**Self**.RemoveCell(i, j);

**end**;

**if** (OldPointsCount < PointsCount) **then**

**begin**

SetLength(ViewMatrix, PointsCount, LinesCount);

**for** i := OldPointsCount **to** PointsCount - 1 **do**

**for** j := 0 **to** OldLinesCount - 1 **do**

**Self**.AddCell(i, j);

**end**

**else**

**begin**

**for** i := PointsCount **to** OldPointsCount - 1 **do**

**for** j := 0 **to** OldLinesCount - 1 **do**

**Self**.RemoveCell(i, j);

**end**;

SetLength(ViewMatrix, PointsCount, LinesCount);

CheckMatrixCorrect;

**end**;

**procedure** TFormMain.StartPlaySound;

**const**

AudioResourceName = 'Audio';

**var**

hResource: **THandle**;

pData: **Pointer**;

**begin**

hResource := LoadResource(hInstance, FindResource(hInstance,

AudioResourceName, RT\_RCDATA));

pData := LockResource(hResource);

SndPlaySound(pData, SND\_MEMORY **or** SND\_ASYNC **or** SND\_LOOP);

FreeResource(hResource);

**end**;

**end**.

**Схема алгоритма**



**Скриншоты**



