**МИНИСТЕРСТВО НАУКИ и ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Коломенский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Московский политехнический университет»**

**Направление:** Информатика и вычислительная техника

**Профиль:** Программное обеспечение вычислительной техники

и автоматизированных систем

**Форма обучения:** очная

**КУРСОВОЙ ПРОЕКТ**

По дисциплине «Структуры данных и алгоритмы их обработки»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_ / М.А. Крупин

(подпись) И.О. Фамилия

Руководитель

курсового проекта

старший преподаватель

кафедры АП и ИТ \_\_\_\_\_\_\_\_\_\_\_\_\_ / И.Н.Филоненко

(подпись) И.О. Фамилия

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

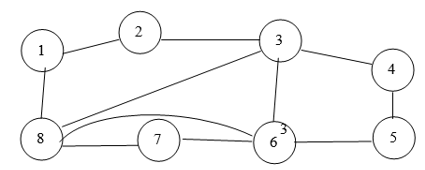
Коломна

2021

**Задача:**

Поиск каркаса минимального веса алгоритмом Прима, обход графа поиском в ширину, построение дерева поиска в ширину.

Написать программу, иллюстрирующую работу с библиотекой



*Веса ребер вводятся в матрицу смежности при запуске программы!*

**Библиотека**

unit MainLib;

interface

uses

System.SysUtils,

Vcl.Dialogs;

Type

TPItem = ^TItem;

IndexList = array[1..50] of integer;

TreeType = array[1..100] of TPItem;

TItem = record

Index: Integer;

IndexPointer: IndexList;

end;

var

PTree: TreeType;

visit: array [1..100] of boolean;

out\_arr: array of integer;

C: array[1..100] of array[1..100] of integer;

C\_BFS: array[1..100] of array[1..100] of integer;

i :integer;

procedure BFS(v:integer; n:integer; LTree: TreeType);

procedure Prim(out Way, Wght:string; n:integer);

Procedure NewNode(val: integer; List:IndexList);

implementation

procedure nullTree(LTree: TreeType);

var i: integer;

begin

for i := 1 to 100 do

LTree[i] := nil;

end;

Procedure NewNode(val: integer; List:IndexList);

var

ForNew: TPItem;

i: integer;

CheckStr: string;

begin

New(ForNew);

ForNew.Index := val;

CheckStr := inttostr(ForNew.Index) + ': ';

for i := 1 to 10 do

begin

ForNew.IndexPointer[i] := List[i];

CheckStr := CheckStr + ' ' + inttostr(ForNew.IndexPointer[i]);

end;

PTree[val] := ForNew;

end;

procedure Prim(out Way, Wght:string; n:integer);

var

visited: array[1..10] of integer;

cost: array[1..10, 1..10] of integer;

a, b, u, v, i, j, ne, min, path\_index, mincost:integer;

ti, tj:integer;

outStr : String;

path: array[1..100] of integer;

begin

outStr := '1, ';

path\_index := 1;

ne := 1;

min := maxint;

mincost := 0;

for i := 1 to 10 do

visited[i] := 0;

for i := 1 to n do

for j := 1 to n do

begin

cost[i,j] := C[i,j];

if (cost[i,j] = 0) then

cost[i, j] := maxint;

end;

visited[1] := 1;

while (ne < n) do

begin

min := maxint;

for i := 1 to n do

begin

for j := 1 to n do

if cost[i,j] < min then

if visited[i] <> 0 then

begin

min := cost[i,j];

u := i;

a := u;

v := j;

b := v;

end;

end;

if (visited[u] = 0) or (visited[v] = 0) then

begin

path[path\_index] := b;

path\_index := path\_index + 1;

ne := ne + 1;

mincost := mincost + min;

visited[b] := 1;

end;

cost[b,a] := maxint;

cost[a,b] := cost[b,a];

end;

for i := 1 to n-1 do

begin

outStr := OutStr + inttostr(path[i]);

if (i < n - 1) then

outStr := outStr + ', ';

end;

Way := outStr;

Wght := inttostr(mincost);

end;

procedure BFS(v:integer; n:integer; LTree: TreeType);

var

Q: array [1..100] of integer;

Un,Uk: integer;

j, add\_out, l\_cl: integer;

NodeVal, SaveToCheck, c1, c2: integer;

inArr: IndexList;

begin

NullTree(LTree);

Un:=0; Uk:=0;

Uk:=Uk+1; Q[Uk]:=v;

add\_out := 0;

Visit[v]:=false;

setlength(out\_arr, n);

c1:= 1;

SaveToCheck := 1;

for j := 1 to 50 do

inArr[j] := 0;

while Un < Uk do

begin

Un:=Un+1; v:=Q[Un];

inc(add\_out);

SaveToCheck := 0;

out\_arr[add\_out] := v;

for j:=0 to n do

begin

if (C\_BFS[v,j]=1) and (Visit[j]) then

begin

Uk:=Uk+1; Q[Uk]:=j;

Visit[j]:=false;

inArr[c1] := j;

c1 := c1 + 1;

SaveToCheck := v;

end;

end;

if SaveToCheck = 0 then

continue

else begin

NewNode(SaveToCheck, inArr);

for l\_cl := 1 to 50 do

inArr[l\_cl] := 0;

c1 := 1;

end;

end;

end;

end.

**Форма:**

unit Unit1;

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.ComCtrls, MainLib;

const

tab\_s = 8;

type

TForm1 = class(TForm)

StringGrid1: TStringGrid;

Label1: TLabel;

Label3: TLabel;

Button1: TButton;

TreeView1: TTreeView;

Label2: TLabel;

Label4: TLabel;

Memo1: TMemo;

Label5: TLabel;

Label6: TLabel;

procedure FormShow(Sender: TObject);

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

n\_form, v\_form: integer;

newnode : TTreeNode;

snode : TTreeNode;

StartTreeDrawVal:integer;

implementation

{$R \*.dfm}

procedure PrintTree(treenode:TTreeNode; Tree:TreeType; goVal: integer);

var

newnode, snode : TTreeNode;

PList: IndexList;

i, goSaver, si:integer;

begin

i:= 1;

if Tree[goVal] = nil then

begin

Form1.TreeView1.Items.AddChild(treenode, IntToStr(goVal));

exit

end

else begin

goSaver := goVal;

while Tree[goSaver].IndexPointer[i] <> 0 do

begin

if (i > 1) then

goVal := GoSaver

else

newnode := Form1.TreeView1.Items.AddChild(treenode, IntToStr(goVal));

snode := treenode;

treenode := newnode;

goVal := Tree[goSaver].IndexPointer[i];

PrintTree(newnode, Tree, goVal);

si := i;

inc(i);

if Tree[goSaver + 1] = nil then

begin

goVal := GoSaver;

treenode := newnode;

end;

end;

exit;

end;

end;

procedure TForm1.Button1Click(Sender: TObject);

var

i,j:integer;

way\_out, wght\_out: string;

begin

for i:=1 to 100 do

visit[i] := true;

for i := 1 to n\_form do

for j := 1 to n\_form do

C[i, j] := strtoint(stringgrid1.Cells[i, j]);

v\_form := 1;

Prim(way\_out, wght\_out, n\_form);

label4.Caption := way\_out;

label6.Caption := wght\_out;

for i := 1 to tab\_s do

for j := 1 to tab\_s do

if C[i, j] > 0 then

begin

C\_BFS[i, j] := 1;

StringGrid1.Cells[i, j] := inttostr(C\_BFS[i, j]);

end;

BFS(v\_form, n\_form, PTree);

for i := 1 to tab\_s do

Memo1.Lines.Add(inttostr(out\_arr[i]));

StartTreeDrawVal := 1;

PrintTree(nil, PTree, StartTreeDrawVal);

end;

procedure TForm1.FormShow(Sender: TObject);

var

tab\_i, tab\_j:integer;

begin

n\_form := tab\_s;

stringgrid1.RowCount := tab\_s + 1;

stringgrid1.ColCount := tab\_s + 1;

for tab\_i := 0 to tab\_s do

begin

stringgrid1.Cells[0, tab\_i] := inttostr(tab\_i);

stringgrid1.Cells[tab\_i, 0] := inttostr(tab\_i);

end;

for tab\_i := 1 to tab\_s do

for tab\_j := 1 to tab\_s do

stringgrid1.Cells[tab\_i, tab\_j] := '0';

stringgrid1.Cells[1, 2] := '2';

stringgrid1.Cells[1, 8] := '4';

stringgrid1.Cells[2, 1] := '2';

stringgrid1.Cells[2, 3] := '3';

stringgrid1.Cells[3, 2] := '3';

stringgrid1.Cells[3, 4] := '4';

stringgrid1.Cells[3, 6] := '5';

stringgrid1.Cells[3, 8] := '5';

stringgrid1.Cells[4, 3] := '4';

stringgrid1.Cells[4, 5] := '7';

stringgrid1.Cells[5, 4] := '7';

stringgrid1.Cells[5, 6] := '6';

stringgrid1.Cells[6, 5] := '6';

stringgrid1.Cells[6, 3] := '5';

stringgrid1.Cells[6, 8] := '9';

stringgrid1.Cells[6, 7] := '3';

stringgrid1.Cells[7, 8] := '7';

stringgrid1.Cells[7, 6] := '3';

stringgrid1.Cells[8, 6] := '9';

stringgrid1.Cells[8, 3] := '5';

stringgrid1.Cells[8, 1] := '4';

stringgrid1.Cells[8, 7] := '7';

end;

end.

