**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

дисциплина: Дискретная математика

Лабораторная работа №4.3

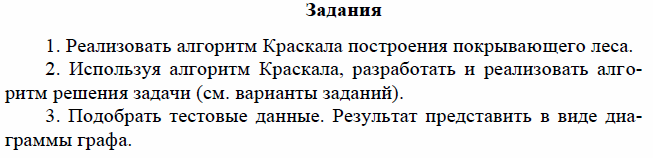
тема: «Связность»

|  |  |
| --- | --- |
|  | Выполнил: ст. группы ПВ-21  Донцов Александр Алексеевич  Проверил: Рязанов Ю.Д. |

Белгород

2018

**Цель занятия:** изучить алгоритм Краскала построения покрывающего леса, научиться использовать его при решении различных задач.





Код программы:

Модуль main.cpp

#include "Graf\_Three.h"

void matrOut(int \*\*a, int n);

int main() {

int n = 5;

int \*\*graf = (int\*\*)malloc(sizeof(int) \* n);//создаем тестовую матрицу

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

graf[i] = (int\*)malloc(sizeof(int) \* n);

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

if(i != j)

graf[i][j] = 1;

else

graf[i][j] = 0;

}

graf[1][3] = 0;

graf[3][1] = 0;

graf[2][0] = 0;

graf[0][2] = 0;

matrOut(graf, n);

int \*\*tree;

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

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

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

graf[i][j] = 0;

graf[j][i] = 0;

tree = algKracksal(graf, n, 0);

if (is\_sviaz(graf, n)) {

printf("%d->%d\n", i, j);

}

free(tree);

}

}

}

int \*\*tree = algKracksal(graf, n, 0);

matrFree(graf, n);

printf("\n");

matrOut(tree, n);

matrFree(tree, n);

getchar();

getchar();

getchar();

return 0;

}

//вывод квадратной матрицы

void matrOut(int \*\*a, int n) {

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

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

printf("%d ", a[i][j]);

printf("\n");

}

}

// Является ли граф связным

int is\_sviaz(int \*\* graf, int n) {

int \*a = (int\*)malloc(sizeof(int) \* n);

int i = 0, j, k;

a[i] = 1;

int count = 1;

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

if (a[i] == i) {

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

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

for (k = 0; k < n; k++) {

if ((k != i) && (graf[k][j] == 1) && (a[k] == 0))

a[k] = i + 1;

}

}

}

}

}

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

if (a[i] == 0) {

return 0;

}

i++;

}

return 1;

}

Заголовочный файл Graf\_Three.h

#pragma once

#define CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

int\*\* algKracksal(int\*\* graf, int n);//алгоритм Краскала

void recAlgKracksal(int \*\*graf, int \*\*tree, int n, int central, int \*was);//заполняем дерево от вершины central(рекурсия)

void arrZero(int\* a, int n);//обнуление массива a размера n

void arrMax(int\* a, int n);//INT\_MAX массива a размера n

int findCentralTop(int \*\*graf, int n);//поиск центральной вершины

int\* eccentricity(int \*\*graf, int n);//эксцентриситеты вершин графа

int distance(int \*\*graf, int n, int v1, int v2);//поиск кратчайшего пути между вершинами v1 и v2

int recDistance(int \*\*graf, int n, int \*was, int v1, int v2, int min, int now);//рекурсия для поиска расстояния между вершинами

int minIndex(int \*a, int n);//возврат индекса наименьшего значения

void matrFree(int \*\*a, int n);//освобождение памяти

void arrOut(int \*a, int n);//вывод массива a размера n

Исходный файл Graf\_Three.cpp

#include "Graf\_Three.h"

//алгоритм Краскала

int\*\* algKracksal(int\*\* graf, int n) {

int \*\*tree = (int\*\*)malloc(sizeof(int)\*n);

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

tree[i] = (int\*)malloc(sizeof(int)\*n);

arrZero(tree[i], n);

}

int centralTop = findCentralTop(graf, n);//центральная вершина

int \*was = (int\*)malloc(sizeof(int)\*n);

arrZero(was, n);

recAlgKracksal(graf, tree, n, centralTop - 1, was);//от центральной вершины построить деревья

return tree;

}

//заполняем дерево от вершины central

void recAlgKracksal(int \*\*graf, int \*\*tree, int n, int central, int \*was) {

int \*pass = (int\*)malloc(sizeof(int) \* n);

arrZero(pass, n);

was[central] = 1;

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

if (graf[central][i] && !was[i]) {

tree[central][i] = 1;

tree[i][central] = 1;

was[i] = 1;

pass[i] = 1;

}

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

if (pass[i]) {

pass[i] = 0;

recAlgKracksal(graf, tree, n, i, was);

}

free(pass);

}

//обнуление массива a размера n

void arrZero(int\* a, int n) {

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

a[i] = 0;

}

//INT\_MAX массива a размера n

void arrMax(int\* a, int n) {

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

a[i] = INT\_MAX;

}

//поиск центральной вершины

int findCentralTop(int \*\*graf, int n) {

int \*arrEccentricity;

int min;

arrEccentricity = eccentricity(graf, n);//эксцентриситеты вершин графа

min = minIndex(arrEccentricity, n);

free(arrEccentricity);

return min + 1;

}

//эксцентриситеты вершин графа

int\* eccentricity(int \*\*graf, int n) {

int max;

int \*arrEccentricity = (int\*)malloc(sizeof(int)\*n);

arrZero(arrEccentricity, n);

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

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

max = distance(graf, n, i, j);

if ((i != j) && (arrEccentricity[i] < max))//вопрос, не уверен

arrEccentricity[i] = max;

}

return arrEccentricity;

}

//поиск кратчайшего пути между вершинами v1 и v2

int distance(int \*\*graf, int n, int v1, int v2) {

int min = INT\_MAX;

int \*was = (int\*)malloc(sizeof(int) \* n);

arrZero(was, n);

min = recDistance(graf, n, was, v1, v2, min, 0);

free(was);

return min;

}

//рекурсия для поиска расстояния между вершинами

int recDistance(int \*\*graf, int n, int \*was, int v1, int v2, int min, int now) {

if (v1 == v2) {

return now;

}

else

for (int i = 0; i < n && now < min; i++)

if (graf[v1][i] && !was[i]) {

was[v1] = 1;

min = recDistance(graf, n, was, i, v2, min, now+1);

was[v1] = 0;

}

return min;

}

//возврат индекса наименьшего значения

int minIndex(int \*a, int n) {

int min = INT\_MAX;

int rez;

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

if (min > a[i]) {

min = a[i];

rez = i;

}

return rez;

}

//освобождение памяти

void matrFree(int \*\*a, int n) {

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

free(a[i]);

free(a);

}

//вывод массива a размера n

void arrOut(int \*a, int n) {

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

printf("%d ", a[i]);

printf("\n");

}

