Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Л и ОА в ИЗ»

на тему «Унарные и бинарные операции над графами»

**Выполнили:**

студенты группы 19ВВ2

Сидоров Н. Р.

Карамышев А. А.

Горбунов Д.А.

**Приняли:**

Юрова О. В.

Митрохин М. А.

Пенза 2020

**Название:**

Оценка времени выполнения программ.

**Цель работы:**

Научиться применять алгоритм оценки времени выполнения программ.

**Лабораторные задания:**

1. Сгенерируйте (используя генератор случайных чисел) две матрицы M 1 ,

М 2 смежности неориентированных помеченных графов G 1 , G 2 . Выведите

сгенерированные матрицы на экран.

2. \* Для указанных графов преобразуйте представление матриц

смежности в списки смежности. Выведите полученные списки на экран.

Задание 2

1. Для матричной формы представления графов выполните операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

2. \* Для представления графов в виде списков смежности выполните

операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

Задание 3

1. Для матричной формы представления графов выполните операцию:

а) объединения G = G 1 

G 2

б) пересечения G = G 1 

G 2

в) кольцевой суммы G = G 1  G 2

Результат выполнения операции выведите на экран.

Задание 4 \*

1. Для матричной формы представления графов выполните операцию

декартова произведения графов G = G 1 X G 2 .

Результат выполнения операции выведите на экран.

**Вывод:** Научились выполнять над графами различные операции: отождествление вершин, стягивание ребра, расщепление вершины.

Листинг:

#include <time.h>

#include <conio.h>

#include <list>

#include <iostream>

#include <cstring>

#include <windows.h>

using namespace std;

int\*\* m1;

int\*\* m2;

void mainly(void) {

srand(time(NULL)); // заполнение

setlocale(LC\_ALL, "Rus");

int N;

printf("Введите количество вершин графа: ");

scanf\_s("%d", &N);

int\*\* m1;

int\*\* m2;

m1 = new int\* [N];

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

m1[i] = new int[N];

}

m2 = new int\* [N];

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

m2[i] = new int[N];

}

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

for (int j = (1 + i); j < N; j++) {

m1[i][j] = rand() % 2;

m1[j][i] = m1[i][j];

}

}

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

m1[i][i] = 0;

}

for (int i = 0; i < N; i++) { // вывод массива

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

printf("%d ", m1[i][j]);

}

printf("\n");

}

printf("\n");

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

m2[i][i] = 0;

}

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

for (int j = (1 + i); j < N; j++) {

m2[i][j] = rand() % 2;

m2[j][i] = m2[i][j];

}

}

for (int i = 0; i < N; i++) { // вывод массива

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

printf("%d ", m2[i][j]);

}

printf("\n");

}

\_getch();

}

void zadanie1() {

int N, N2;

printf("Введите количество вершин графа: ");

scanf\_s("%d", &N);

int\*\* m2;

N2 = N;

m2 = new int\* [N+1];

for (int i = 0; i < N+1; i++) {

m2[i] = new int[N+1];

}

srand(time(NULL)); // заполнение

for (int i = 0; i < N2; i++) {

for (int j = (1 + i); j < N2; j++) {

m2[i][j] = rand() % 2;

m2[j][i] = m2[i][j];

}

}

for (int i = 0; i < N2; i++) { // обнуление главной диагонали

m2[i][i] = 0;

}

for (int i = 0; i < N2; i++) {

for (int j = 0; j < N2; j++) {

printf("%d ", m2[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

int kol1, kol2;

printf("Введите номера вершин для отождествления (через пробел): ");

scanf\_s("%d %d", &kol1, &kol2);

kol1--;

kol2--;

for (int i = 0; i < N2; i++) { // отождествление

if (m2[kol2][i]) {

m2[kol1][i] = 1;

m2[i][kol1] = 1;

}

}

//// изменение размера массива

for (int i = 0; i < kol2; i++) {

for (int j = kol2; j < (N2 - 1); j++) {

m2[i][j] = m2[i][j + 1];

m2[j][i] = m2[j + 1][i];

}

}

for (int i = kol2; i < (N2 - 1); i++) {

for (int j = kol2; j < (N2 - 1); j++) {

m2[i][j] = m2[i + 1][j + 1];

}

}

m2[kol1][kol1] = 1;

N2--;

////

for (int i = 0; i < N2; i++) { // вывод массива

for (int j = 0; j < N2; j++) {

printf("%d ", m2[i][j]);

}

printf("\n");

}

N2++;

printf("\n");

printf("\n");

printf("Введите номера вершин для стягивания ребра (через пробел): ");

scanf\_s("%d %d", &kol1, &kol2);

kol1--;

kol2--;

if (m2[kol1][kol2]) {

for (int i = 0; i < N2; i++) { // отождествление

if (m2[kol2][i]) {

m2[kol1][i] = 1;

m2[i][kol1] = 1;

}

}

//// изменение размера массива

for (int i = 0; i < kol2; i++) {

for (int j = kol2; j < (N2 - 1); j++) {

m2[i][j] = m2[i][j + 1];

m2[j][i] = m2[j + 1][i];

}

}

for (int i = kol2; i < (N2 - 1); i++) {

for (int j = kol2; j < (N2 - 1); j++) {

m2[i][j] = m2[i + 1][j + 1];

}

}

m2[kol1][kol1] = 1;

N2--;

////

for (int i = 0; i < N2; i++) { // вывод массива

for (int j = 0; j < N2; j++) {

printf("%d ", m2[i][j]);

}

printf("\n");

}

N2++;

printf("\n");

printf("Введите номер вершины для расщепления: ");

scanf\_s("%d", &kol1);

kol1--;

int sum = 0;

for (int i = 0; i < N2; i++) { // счётчик смежных вершин

if (m2[kol1][i]) {

sum++;

kol2 = i; // просто узнать 1 смежную вершину

}

}

for (int i = 0; i < 11; i++) {

m2[N2][i] = 0;

m2[i][N2] = 0;

}

if (!sum) {

m2[kol1][N2] = 1;

m2[N2][kol1] = 1;

m2[N2][N2] = 0;

}

else {

if (sum == 1) {

for (int i = 0; i < N2; i++) { // копирование значений 1 вершины в другую

m2[N2][i] = m2[kol1][i];

m2[i][N2] = m2[i][kol1];

}

m2[N2][kol1] = 1; // коррекция

m2[kol1][N2] = 1;

m2[N2][N2] = 0;

}

else { // создание новой вершины

m2[N2][kol2] = 1; // в 10 строчку под номером kol2

m2[N2][kol1] = 1; // создание связи с расщепляемой вершиной

m2[kol2][N2] = 1; // в строчку kol2 под номером 10

m2[kol1][N2] = 1; // создание связи с расщепляемой вершиной

m2[kol1][kol2] = 0;// вычёркивание связи с одной из вершин

m2[kol2][kol1] = 0;

m2[N2][N2] = 0;

}

}

}

for (int i = 0; i < N2; i++) { // вывод массива

for (int j = 0; j < N2; j++) {

printf("%d ", m2[i][j]);

}

printf("\n");

}

\_getch();

}

void zadanie3() {

int N1, N2;

printf("Введите размеры графов (через пробел): ");

scanf\_s("%d %d", &N1, &N2);

int\*\* m1;

int\*\* m2;

int\*\* c;

m1 = new int\* [N1];

for (int i = 0; i < N1; i++) {

m1[i] = new int[N1];

}

m2 = new int\* [N2];

for (int i = 0; i < N2; i++) {

m2[i] = new int[N2];

}

const unsigned int N = max(N1, N2);

int minn = min(N1, N2);

c = new int\* [N];

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

c[i] = new int[N];

}

srand(time(NULL)); // заполнение

for (int i = 0; i < N1; i++) {

for (int j = (1 + i); j < N1; j++) {

m1[i][j] = rand() % 2;

m1[j][i] = m1[i][j];

}

}

for (int i = 0; i < N1; i++) {

m1[i][i] = 0;

}

for (int i = 0; i < N1; i++) { // вывод массива

for (int j = 0; j < N1; j++) {

printf("%d ", m1[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

// заполнение

for (int i = 0; i < N2; i++) {

for (int j = (1 + i); j < N2; j++) {

m2[i][j] = rand() % 2;

m2[j][i] = m2[i][j];

}

}

for (int i = 0; i < N2; i++) {

m2[i][i] = 0;

}

for (int i = 0; i < N2; i++) { // вывод массива

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

printf("%d ", m2[i][j]);

}

printf("\n");

}

printf("\n\n\n");

printf("Результат объединения: ");

printf("\n\n\n");

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

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

c[i][j] = 0;

}

}

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

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

if (m1[i][j] || m2[i][j]) {

c[i][j] = 1;

}

else {

c[i][j] = 0;

}

}

}

for (int i = 0; i < N; i++) { // вывод массива

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

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

}

printf("\n");

}

\_getch();

}

void zadanie4() {

int N1, N2, N3;

printf("Введите количество вершин двух графов: ");

scanf\_s("%d %d", &N1, &N2);

int\*\* a;

int\*\* b;

int\*\* c;

a = new int\* [N1];

for (int i = 0; i < N1; i++) {

a[i] = new int[N1];

}

b = new int\* [N2];

for (int i = 0; i < N2; i++) {

b[i] = new int[N2];

}

N3 = N1 \* N2;

c = new int\* [N3];

for (int i = 0; i < N3; i++) {

c[i] = new int[N3];

}

srand(time(NULL)); // заполнение

for (int i = 0; i < N1; i++) {

for (int j = (1 + i); j < N1; j++) {

a[i][j] = rand() % 2;

a[j][i] = a[i][j];

}

}

for (int i = 0; i < N1; i++) {

a[i][i] = 0;

}

for (int i = 0; i < N2; i++) {

for (int j = (1 + i); j < N2; j++) {

b[i][j] = rand() % 2;

b[j][i] = b[i][j];

}

}

for (int i = 0; i < N2; i++) {

b[i][i] = 0;

}

int N;

if (N1 > N2) {

N = N1;

}

else {

N = N2;

}

for (int i = 0; i < N3; i++) {

for (int j = 0; j < N3; j++) {

c[i][j] = 0;

}

}

int x1, x2, y1, y2;

for (int i = 0; i < N3; i++) {

for (int j = 0; j < N3; j++) {

x1 = i / N;

x2 = i % N;

y1 = j / N;

y2 = j % N;

if (x1 == y1) {

if (b[x2][y2]) {

c[i][j] = 1;

}

}

else {

if (x2 == y2) {

if (a[x1][y1]) {

c[i][j] = 1;

}

}

}

}

}

printf("Декартово произведение: \n\n");

for (int i = 0; i < N1; i++) {

for (int j = 0; j < N1; j++) {

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

}

printf("\n");

}

printf("\n");

printf("\n");

for (int i = 0; i < N2; i++) {

for (int j = 0; j < N2; j++) {

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

}

printf("\n");

}

printf("\n");

printf("Результат: ");

printf("\n");

for (int i = 0; i < N3; i++) {

for (int j = 0; j < N3; j++) {

if (c[i][j] == 0) {

HANDLE hOUTPUT = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

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

}

else {

HANDLE hOUTPUT = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

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

}

}

printf("\n");

}

\_getch();

}

int Menu(void)

{

system("CLS");

int c = 0;

setlocale(LC\_ALL, "Rus");

while (c < '0' || c > '4')

{

printf(

"::=================================::\n"

"|| 1 : ЗАПОЛНЕНИЕ ||\n"

"|| 2 : ЗАДАНИЕ 1 ||\n"

"|| 3 : ЗАДАНИЕ 3 ||\n"

"|| 4 : ЗАДАНИЕ 4 ||\n"

"|| 4 : ВЫХОД ||\n"

"::=================================::\n"

"> ");

c = \_getch();

system("cls");

}

return c;

}

void main(void)

{

int Select;

while ((Select = Menu()) != '0' && Select != 27)

switch (Select)

{

case '1':

mainly();

break;

case '2':

zadanie1();

break;

case '3':

zadanie3();

break;

case '4':

zadanie4();

break;

}

}