Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по теме «Унарные и бинарные операции над графами»

Выполнили:

Студенты группы 21ВВ2

Кривенкова В.С.

Козлова К.С

Принял:

Митрохин М.А.

Юрова О.В.

Пенза, 2022

**Цель работы:**

Исследование унарных, бинарных операций над графами и приобретение практических навыков решения задач с использованием основ теории графов.

**Лабораторные работы:**

**Задание:**

**Задание 1**

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.

1. \* 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 https://lh5.googleusercontent.com/L_5gfz8876Ab2u0x_bXk2V2HkE9jl2Rw_yMDOwONwHywiAO_7hOKeD8IaEuHHpuCyI5btpgBBisHfpggXgTnrZh2FMN9vpZ1SbB1IM8RStloXJXVVCIsbcj0vmTkrSoKDaRNCkRjuUTYsk2N0WK26Aizk85gmlk0dTOOGRMeh6IZpnAMaC3LWe1mxfvAm3PEc-RAEw *G*2

б) пересечения  *G* = *G*1 https://lh3.googleusercontent.com/s8omopK4mw7F2NmG1bjJyrgjLEPWUWssAttwd3a1oAD2Kb_mdOdrfOL9BWjRgZJxjwN4iaaOgcxbq4-07fLn3KFzywp68Seks0vCaGMEHmZpLB6Ozmpusftp1W3bC2vfoteTctyaS6d15iVf_6_Ys7iycjlPLQz0D12wc8AY0D8I53ncxLRVVhHic0Med3IZAcnMTA *G*2

в) кольцевой суммы *G* = *G*1 https://lh4.googleusercontent.com/icQUnFEbZ5A5kDEair-FW6aoFkMqcsH6rldELs5wOEg4t_yjDUAP6jodBG_1Wod84NugOOVU3-saWc2zZXuuCh_coC_KHq3s7f5693UncsXwnqDW4tu2fnn_hZHSHZv70qfcdPqBcUIQYwtQhT3WTiF_IMY-1TMMOE_Yfb-zCC9ymYK-FDLppz7qnml0wx8_GlidgQ *G*2

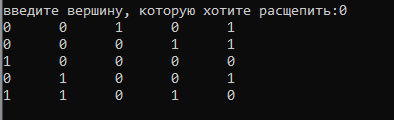
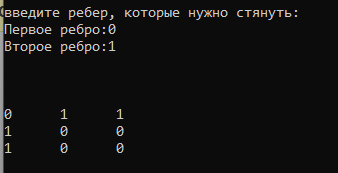
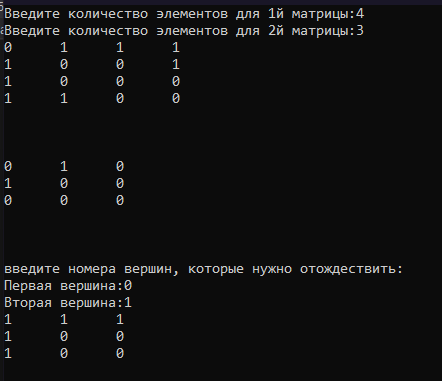
Результат выполнения операции выведите на экран.

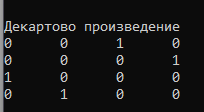
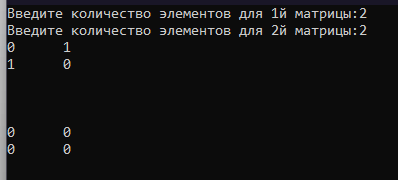
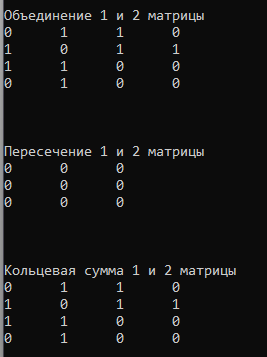
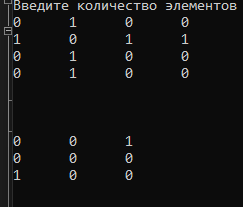
**Задание 4 \***

1. Для матричной формы представления графов выполните операцию декартова произведения графов *G = G*1X *G*2.

Результат выполнения операции выведите на экран.

**Пояснительный текст к программе:**

****

****

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <locale.h>

int main()

{

//srand(time(0));

setlocale(LC\_ALL, "rus");

int n, m, tog;

printf("Введите количество элементов для 1й матрицы:");

scanf("%d", &n);

printf("Введите количество элементов для 2й матрицы:");

scanf("%d", &m);

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

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

{

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

}

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

{

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

{

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

if (i == j)

{

a[i][j] = 0;

}

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

}

}

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

{

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

{

printf("%-5d ", a[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

printf("\n");

int\*\* b = (int\*\*)malloc(m \* sizeof(int));

for (int k = 0; k < m; k++)

{

b[k] = (int\*)malloc(m \* sizeof(int));

}

for (int k = 0; k < m; k++)

{

for (int l = k; l < m; l++)

{

b[k][l] = rand() % 2;

if (k == l)

{

b[k][l] = 0;

}

b[l][k] = b[k][l];

}

}

for (int k = 0; k < m; k++)

{

for (int l = 0; l < m; l++)

{

printf("%-5d ", b[k][l]);

}

printf("\n");

}

printf("\n");

printf("\n");

printf("\n");

int p1, p2;

printf("введите номера вершин, которые нужно отождествить:\n");

printf("Первая вершина:");

scanf("%d", &p1);

printf("Вторая вершина:");

scanf("%d", &p2);

int n1 = n - 1;

int\*\* t = (int\*\*)malloc(n1 \* sizeof(int));

for (int i = 0; i < n1; i++)

{

t[i] = (int\*)malloc(n1 \* sizeof(int));

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

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

if (i < p2)

{

if (j < p2)

{

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

}

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (i == p2 && j > p2)

t[p1][j] = a[p2][j + 1] || a[p1][j + 1];

t[p1][j] = a[p2][j] || a[p1][j];

if (j == p2 && i > p2)

t[i][p1] = a[i + 1][p2] || a[i + 1][p1];

t[i][p1] = a[i][p2] || a[i][p1];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (j >= p2 && i < p2 && i != p1)

{

t[i][j] = a[i][j + 1];

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (j >= p2 && i == p1)

t[i][j] = a[p2][j + 1] || a[p1][j + 1];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (i >= p2 && j < p2 && j != p1)

{

t[i][j] = a[i + 1][j];

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (i >= p2 && j == p1)

t[i][j] = a[i + 1][p2] || a[i + 1][p1];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (j == i && j != p1)

{

t[i][j] = 0;

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

printf("%-5d ", t[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

printf("\n");

int r1, r2;

printf("введите ребер, которые нужно стянуть:\n");

printf("Первое ребро:");

scanf("%d", &r1);

printf("Второе ребро:");

scanf("%d", &r2);

int\*\* s = (int\*\*)malloc(n1 \* sizeof(int));

for (int i = 0; i < n1; i++)

{

s[i] = (int\*)malloc(n1 \* sizeof(int));

}

if (a[r1][r2] == 1)

{

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

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

if (i < r2)

{

if (j < r2)

{

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

}

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (i == r2 && j > r2)

s[r1][j] = a[r2][j + 1] || a[r1][j + 1];

s[r1][j] = a[r2][j] || a[r1][j];

if (j == r2 && i > r2)

s[i][r1] = a[i + 1][r2] || a[i + 1][r1];

s[i][r1] = a[i][r2] || a[i][r1];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (j >= r2 && i < r2 && i != r1)

{

s[i][j] = a[i][j + 1];

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (j >= r2 && i == r1)

s[i][j] = a[r2][j + 1] || a[r1][j + 1];

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (i >= r2 && j < r2 && j != r1)

{

s[i][j] = a[i + 1][j];

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (i >= r2 && j == r1)

s[i][j] = a[i + 1][r2] || a[i + 1][r1];

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (j == i && j != p1)

{

s[i][j] = 0;

}

}

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

if (i == j)

{

s[i][j] = 0;

}

}

printf("\n");

}

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n1; j++)

{

printf("%-5d ", s[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

printf("\n");

int sh;

printf("введите вершину, которую хотите расщепить:");

scanf("%d", &sh);

int n2 = n + 1;

int\*\* R = (int\*\*)malloc(n2 \* sizeof(int));

for (int i = 0; i < n2; i++)

{

R[i] = (int\*)malloc(n2 \* sizeof(int));

}

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

R[i][j] = 0;

}

}

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

{

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

{

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

}

}

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

{

if (j % 2)

{

if (R[sh][j] == 1)

{

R[j][sh] = R[sh][j] = 0;

R[j][n] = R[n][j] = 1;

R[sh][n] = R[n][sh] = 1;

}

}

}

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

printf("%-5d ", R[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

printf("\n");

printf("Объединение 1 и 2 матрицы \n");

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

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

{

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

}

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

{

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

{

O[i][j] = a[i][j] || b[i][j];

}

}

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

{

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

{

if (j >= m)

{

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

}

if (i >= m)

{

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

}

}

}

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

{

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

{

printf("%-5d ", O[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

printf("\n");

printf("Пересечение 1 и 2 матрицы \n");

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

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

{

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

}

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

{

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

{

P[i][j] = a[i][j] & b[i][j];

}

}

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

{

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

{

if (j >= m)

{

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

}

if (i >= m)

{

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

}

}

}

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

{

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

{

printf("%-5d ", P[i][j]);

}

printf("\n");

}

printf("\n");

printf("\n");

printf("\n");

printf("Кольцевая сумма 1 и 2 матрицы \n");

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

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

{

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

}

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

{

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

{

K[i][j] = a[i][j] ^ b[i][j];

}

}

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

{

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

{

if (j >= m)

{

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

}

if (i >= m)

{

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

}

}

}

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

{

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

{

printf("%-5d ", K[i][j]);

}

printf("\n");

}

int dec = n \* m;

int x = 0;

int z = 0;

int\*\* d = (int\*\*)malloc(dec \* sizeof(int));

for (int x = 0; x < dec; x++)

{

d[x] = (int\*)malloc(dec \* sizeof(int));

}

for (int x = 0; x < dec; x++)

{

for (int z = 0; z < dec; z++)

{

d[x][z]=0;

}

printf("\n");

}

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

{

for (int k = 0; k < m; k++)

{

z = 0;

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

{

for (int l = 0; l < m; l++)

{

if (i == j)

{

d[x][z] = b[k][l];

}

if (k == l)

{

d[x][z] = a[i][j];

}

z++;

}

}

x++;

}

}

printf("\n");

printf("\n");

printf("\n");

printf("Декартово произведение \n");

for (int x = 0; x < dec; x++)

{

for (int z = 0; z < dec; z++)

{

printf("%-5d ", d[x][z]);

}

printf("\n");

}

return 0;

}

**Вывод:** Мы научились выполнять унарные, бинарные операций над графами и приобрели практические навыки решения задач с использованием основ теории графов.