Пензенский государственный университет  
Кафедра «Вычислительной техники»

## Отчет по лабораторной работе №3 по дисциплине: «Логика и основы алгоритмизации в инженерных задачах» на тему: «Унарные и бинарные операции над графами»

**Выполнили студенты группы 19ВВ1:**

Тебнев Р.  
Балалаев А.

**Приняли:**

д.т.н. профессор Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы**: изучить унарные и бинарные операции над графами и реализовать их в программе.

**Теоретическая часть:**

**Замыкание или отождествление.**

Пара вершин хi и xj в графе G замыкается (или отождествляется), если они заменяются такой новой вершиной, что все дуги в графе G, инцидентные хi и xj , становятся инцидентными новой вершине.При этом ребро, соединяющее вершины хi и xj становится петлёй. Матрица смежности графа после выполнения операции замыкания вершин хi и xj получается путем поэлементного логического сложения i - го и j – го столбцов и i -ой и j - строк в исходной матрице.

**Стягивание.**

Под стягиванием подразумевают операцию удаления дуги или ребра и отождествление его концевых вершин.

**Расщепление вершины.**

В графе G1 выделяется вершина ν. Окружение Q вершины ν разбивается на две части М, N (Mhttps://cde.osu.ru/courses2/temp4/images/und.gifN=Q, Mhttps://cde.osu.ru/courses2/temp4/images/sh.gifN=ø). После этого над графом G1 выполняются следующие преобразования:

1. удаляют вершину ν вместе с инцидентными ей ребрами (Н1=G1=ν-νzi, zihttps://cde.osu.ru/courses2/temp4/images/in.gifQ, i=1,2,3 ..);
2. добавляют новые вершины u,w и соединяющее их ребро uw(H2=H1+u+w+uw);
3. вершину u соединяют ребром с каждой вершиной из множества М(H3=H1+uzi, zihttps://cde.osu.ru/courses2/temp4/images/in.gifM);
4. вершину w соединяют ребром с каждой вершиной из множества N(G2=H3+uzi, zihttps://cde.osu.ru/courses2/temp4/images/in.gifM).

Говорят, что граф G2 получен из графа G1 расщеплением вершины ν.

**Объединение графов** G1 и G2, обозначаемое как G1 U G2, представляет собой граф G3 = (X1 U X2, A1 U A2) такой, что множество его вершин является объединением Х1 и Х2, а множество ребер – объединением A1 и A2 .

Матрица смежности результирующего графа G3 получается операцией поэлементного логического сложения матриц смежности исходных графов G1 и G2 .

**Пересечение графов** G1 и G2 , обозначаемое как G1 ∩ G2, представляет собой граф G3 = (X1 ∩ X2, A1 ∩ A2) такой, что множество его вершин и дуг состоит из вершин и дуг, присутствующих одновременно в G1 и G2.

Результирующая матрица смежности получается операцией поэлементного логического умножения матриц смежности исходных графов G1 и G2.

**Кольцевая сумма** графов G1 и G2, обозначаемая как G1 ⊕ G2, представляет собой граф, порожденный на множестве ребер A1 ⊕ A2.

Т.е. новый граф не имеет изолированных вершин и состоит только из ребер, присутствующих либо в G1 , либо в G2 , но не в обоих одновременно.

Результирующая матрица смежности получается операцией поэлементного логического сложения по mod 2 матриц смежности исходных графов G1 и G2 .

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <time.h>

#include <locale.h>

#define SIZE 5

void rand\_Zap(int k, int\* mat, int n) {

srand(time(NULL) \* k);

printf("G%d \n", k);

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

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

if (i == j) {

mat[i \* n + j] = 0;

}

if (i < j) {

mat[i \* n + j] = rand() % 2;

mat[j \* n + i] = mat[i \* n + j];

}

}

}

void print\_G(int\* mat, int n) {

printf(" ");

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

printf("%3d", i + 1);

}

printf("\n\n");

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

printf("%2d", i + 1);

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

printf("%3d", mat[i \* n + j]);

}

printf("\n");

}

printf("\n");

}

void copy(int(&mat1)[SIZE][SIZE], int(&mat2)[SIZE][SIZE]) {

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

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

mat2[i][j] = mat1[i][j];

}

void otj(int(&constant)[SIZE][SIZE], int(&work)[SIZE][SIZE], int x1, int x2, int st) {

int G[SIZE - 1][SIZE - 1], i, j, i1, j1;

copy(constant, work);

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

work[i][x1] = constant[i][x1] || constant[i][x2];

}

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

work[x1][j] = constant[x1][j] || constant[x2][j];

}

for (i = 0, i1 = x2 + 1; i < SIZE - 1; i++) {

for (j = 0, j1 = x2; j < SIZE - 1; j++) {

if (j >= x2)

j1++;

if (i < x2 && j < x2)

G[i][j] = work[i][j];

if (i < x2 && j >= x2)

G[i][j] = work[i][j1];

if (i >= x2 && j < x2)

G[i][j] = work[i1][j];

if (i >= x2 && j >= x2)

G[i][j] = work[i1][j1];

}

if (i >= x2) i1++;

}

if (constant[x1][x2] == 1 && st == 0)

G[x1][x1] = 1;

else

G[x1][x1] = 0;

printf("Результат:\n");

print\_G(&G[0][0], SIZE - 1);

}

void ras(int(&constant)[SIZE][SIZE], int x) {

int i, j, i1, mat[SIZE + 1][SIZE + 1];

x--;

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

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

mat[i][j] = constant[i][j];

}

}

for (i = 0; i < SIZE + 1; i++) {

for (j = 0; j < SIZE + 1; j++) {

if (i == 5 || j == 5) {

mat[5][j] = 0;

mat[j][5] = 0;

}

}

}

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

if (mat[x][j] == 1)

if (j % 2 == 0) {

mat[5][j] = 1;

mat[j][5] = 1;

mat[x][j] = 0;

mat[j][x] = 0;

}

}

mat[x][5] = 1;

mat[5][x] = 1;

printf("Результат расщепления\n\n");

print\_G(&mat[0][0], SIZE + 1);

}

void obed(int(&mat1)[SIZE][SIZE], int(&mat2)[SIZE][SIZE]) {

int res[SIZE][SIZE];

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

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

res[i][j] = mat1[i][j] || mat2[i][j];

}

printf("Результат объединения G1 и G2\n");

print\_G(&res[0][0], SIZE);

}

void perseh(int(&mat1)[SIZE][SIZE], int(&mat2)[SIZE][SIZE]) {

int res[SIZE][SIZE];

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

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

if (mat1[i][j] == 1 && mat2[i][j] == 1)

res[i][j] = 1;

else

res[i][j] = 0;

}

printf("Результат пересечения G1 и G2\n");

print\_G(&res[0][0], SIZE);

}

void kolc\_sum(int(&mat1)[SIZE][SIZE], int(&mat2)[SIZE][SIZE]) {

int res[SIZE][SIZE];

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

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

if ((mat1[i][j] == 1 && mat2[i][j] == 0) || (mat1[i][j] == 0 && mat2[i][j] == 1))

res[i][j] = 1;

else

res[i][j] = 0;

}

printf("Результат кольцевой суммы G1 и G2\n");

print\_G(&res[0][0], SIZE);

}

int main() {

int G1[SIZE][SIZE], G2[SIZE][SIZE], work[SIZE][SIZE], i, x1, x2, j, A[20][20];

setlocale(LC\_ALL, "Rus");

rand\_Zap(1, &G1[0][0], SIZE);

print\_G(&G1[0][0], SIZE);

rand\_Zap(2, &G2[0][0], SIZE);

print\_G(&G2[0][0], SIZE);

printf("Отождествление 1 графа - Введите 2 вершины: ");

scanf("%d%d", &x1, &x2);

if (x1 > x2) {

i = x1;

x1 = x2;

x2 = i;

}

if ((x1 >= 1) && (x2 <= SIZE)) {

x1--; x2--;

otj(G1, work, x1, x2, 0);

}

else

printf("\nНет вершин(ы)\n");

printf("Отождествление 2 графа - Введите 2 вершины: ");

scanf("%d%d", &x1, &x2);

if (x1 > x2) {

i = x1;

x1 = x2;

x2 = i;

}

if ((x1 >= 1) && (x2 <= SIZE)) {

x1--; x2--;

otj(G2, work, x1, x2, 0);

}

else

printf("\nНет вершин(ы)\n");

printf("Стягивание ребра в 1 графе - Введите 2 вершины: ");

scanf("%d%d", &x1, &x2);

if (x1 > x2) {

i = x1;

x1 = x2;

x2 = i;

}

if ((x1 >= 1) && (x2 <= SIZE)) {

if (G1[x1 - 1][x2 - 1] == 1)

{

x1--; x2--;

otj(G1, work, x1, x2, 1);

}

else printf("\nНет ребра\n");

}

else

printf("\nНет вершин(ы)\n");

printf("Стягивание ребра во 2 графе - Введите 2 вершины: ");

scanf("%d%d", &x1, &x2);

if (x1 > x2) {

i = x1;

x1 = x2;

x2 = i;

}

if ((x1 >= 1) && (x2 <= SIZE)) {

if (G2[x1 - 1][x2 - 1] == 1)

{

x1--; x2--;

otj(G2, work, x1, x2, 1);

}

else printf("\nНет ребра\n");

}

else

printf("\nНет вершин(ы)\n");

printf("Введите вершину для расщепления в 1 графе: ");

scanf("%d", &x1);

ras(G1, x1);

getchar();

printf("Введите вершину для расщепления во 2 графе: ");

scanf("%d", &x1);

ras(G2, x1);

getchar();

obed(G1, G2);

getchar();

perseh(G1, G2);

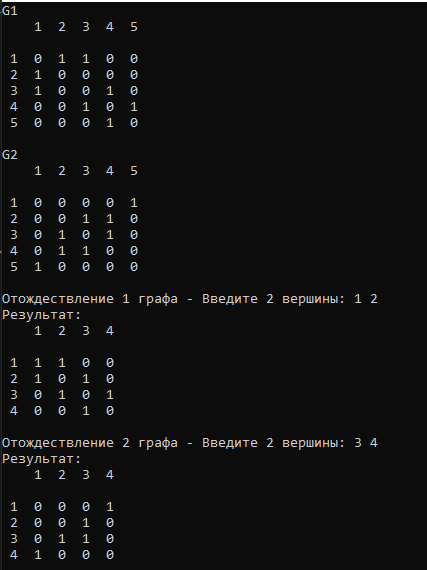
getchar();

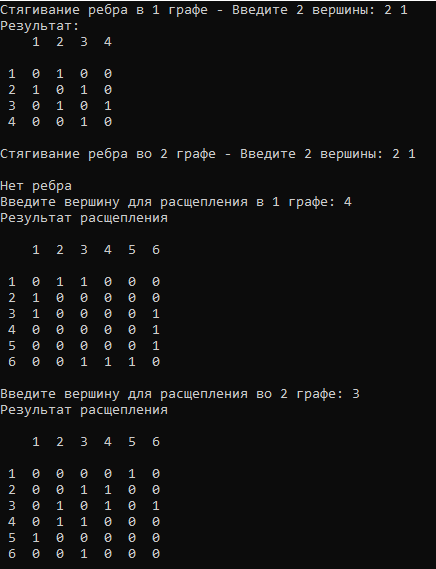
kolc\_sum(G1, G2);

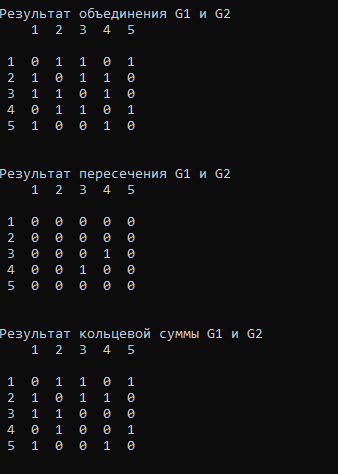
getchar();

}

**Результат работы программы:**







**Вывод:** изучил и реализовал унарные и бинарные операции над графами.