Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №3

по дисциплине: «Логика и основы оптимизации в инженерных задачах»

## на тему «Унарные и бинарные операции над графами»

Выполнили студенты группы 19ВВ2:

Муромский Д.А.

Кобзев М.И.

Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель:** научиться работать с графами в матричной и списковой форме.

**Практическая часть**

### 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.

Заполнение 2 матриц смежности и вывод на экран:

int N = 10;

int m1[10][10];

int m2[10][10];

srand(time(NULL)); // заполнение

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");

}

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");

}

Заполнение матрицы смежности и списка смежности:

int N2 = 10;

int m2[11][11];

List relist[11];

car temp;

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++) {

printf("V%d:", i + 1);

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

if (m2[i][j]) {

temp.nomer = j;

if (relist[i].GetSize() == 0) {

relist[i].push\_front(temp);

printf(" v%d", j+1);

}

else {

relist[i].push\_back(temp);

printf(" v%d", j+1);

}

}

}

printf("\n");

}

### 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.

void zad1() {

int N = 10;

int N2 = 10;

int m2[11][11];

List relist[11];

car temp;

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");

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

printf("V%d:", i + 1);

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

if (m2[i][j]) {

temp.nomer = j;

if (relist[i].GetSize() == 0) {

relist[i].push\_front(temp);

printf(" v%d", j+1);

}

else {

relist[i].push\_back(temp);

printf(" v%d", j+1);

}

}

}

printf("\n");

}

///////////////////////////////////////////////////////////////////////////////////////////

printf("\n");

printf("\n");

int kol1, kol2;

printf("Vvedite nomera vershin dla otojdectvlenia: ");

scanf("%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++;

///////////////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//////////////

bool flag(!true);

for (int i = 0; i < relist[kol2].GetSize(); i++) {

flag = false;

for (int j = 0; j < relist[kol1].GetSize(); j++) {

if (relist[kol2][i].nomer == relist[kol1][j].nomer) {

flag = true;

}

}

if (!flag) {

relist[kol1].push\_back(relist[kol2][i]);

}

}

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

for (int j = 0; j < relist[i].GetSize(); j++) {

if (relist[i][j].nomer == kol2) {

relist[i].removeAt(j);

}

/\*

if (relist[i][j].nomer > kol2) {

relist[i][j].nomer--;

}

\*/

}

}

relist[kol2].clear();

/\*

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

relist[i].clear();

for (int j = 0; j < relist[i + 1].GetSize(); j++) {

relist[i].push\_back(relist[i + 1][j]);

}

}

\*/

for (int i = 0; i < relist[kol1].GetSize(); i++){

flag = false;

for (int j = 0; j < relist[relist[kol1][i].nomer].GetSize(); j++){

if (relist[relist[kol1][i].nomer][j].nomer == kol1){

flag = true;

}

}

if (!flag){

temp.nomer = kol1;

relist[relist[kol1][i].nomer].push\_back(temp);

}

}

flag = false;

for (int i = 0; i < relist[kol1].GetSize(); i++){

if (relist[kol1][i].nomer == kol1){

flag = true;

}

}

if (!flag){

temp.nomer = kol1;

relist[kol1].push\_back(temp);

}

printf("\n");

printf("\n");

N2--;

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

for (int counter = 0; counter < relist[i].GetSize(); counter++){

for (int j = 0; j < relist[i].GetSize() - 1 ; j++){

if (relist[i][j].nomer > relist[i][j+1].nomer) {

relist[i][j].nomer = relist[i][j].nomer ^ relist[i][j+1].nomer;

relist[i][j + 1].nomer = relist[i][j+1].nomer ^ relist[i][j].nomer;

relist[i][j].nomer = relist[i][j].nomer ^ relist[i][j+1].nomer;

}

}

}

}

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

if (relist[i].GetSize()){

printf("V%d:", i + 1);

for (int j = 0; j < relist[i].GetSize(); j++) {

printf(" v%d", relist[i][j].nomer + 1);

}

}

printf("\n");

}

////////////////////////////////////////////////////////////////

printf("\n");

printf("\n");

printf("Vvedite nomera vershin dla stagivania rebra: ");

scanf("%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++;

///////////////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//////////////

bool flag(!true);

for (int i = 0; i < relist[kol2].GetSize(); i++) {

flag = false;

for (int j = 0; j < relist[kol1].GetSize(); j++) {

if (relist[kol2][i].nomer == relist[kol1][j].nomer) {

flag = true;

}

}

if (!flag) {

relist[kol1].push\_back(relist[kol2][i]);

}

}

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

for (int j = 0; j < relist[i].GetSize(); j++) {

if (relist[i][j].nomer == kol2) {

relist[i].removeAt(j);

}

/\*

if (relist[i][j].nomer > kol2) {

relist[i][j].nomer--;

}

\*/

}

}

relist[kol2].clear();

/\*

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

relist[i].clear();

for (int j = 0; j < relist[i + 1].GetSize(); j++) {

relist[i].push\_back(relist[i + 1][j]);

}

}

\*/

for (int i = 0; i < relist[kol1].GetSize(); i++){

flag = false;

for (int j = 0; j < relist[relist[kol1][i].nomer].GetSize(); j++){

if (relist[relist[kol1][i].nomer][j].nomer == kol1){

flag = true;

}

}

if (!flag){

temp.nomer = kol1;

relist[relist[kol1][i].nomer].push\_back(temp);

}

}

flag = false;

for (int i = 0; i < relist[kol1].GetSize(); i++){

if (relist[kol1][i].nomer == kol1){

flag = true;

}

}

if (!flag){

temp.nomer = kol1;

relist[kol1].push\_back(temp);

}

N2--;

}

else {

printf("Vershini ne smejnie\n");

}

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

for (int counter = 0; counter < relist[i].GetSize(); counter++){

for (int j = 0; j < relist[i].GetSize() - 1 ; j++){

if (relist[i][j].nomer > relist[i][j+1].nomer) {

relist[i][j].nomer = relist[i][j].nomer ^ relist[i][j+1].nomer;

relist[i][j + 1].nomer = relist[i][j+1].nomer ^ relist[i][j].nomer;

relist[i][j].nomer = relist[i][j].nomer ^ relist[i][j+1].nomer;

}

}

}

}

printf("\n");

printf("\n");

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

if (relist[i].GetSize()){

printf("V%d:", i + 1);

for (int j = 0; j < relist[i].GetSize(); j++) {

printf(" v%d", relist[i][j].nomer + 1);

}

}

printf("\n");

}

///////////////////////////////////////////////////////////////

printf("\n");

printf("\n");

printf("Vvedite nomer vershini dla rassheplenia: ");

scanf("%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;

}

}

////////////////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/////////////

if (!relist[kol1].GetSize()) {

temp.nomer = kol1;

relist[N].push\_back(temp);

temp.nomer = N;

relist[kol1].push\_back(temp);

}

else {

relist[N].push\_back(relist[kol1][relist[kol1].GetSize()-1]);

if (relist[kol1].GetSize() > 1) {

relist[kol1].removeAt(relist[kol1].GetSize()-1);

}

temp.nomer = kol1;

relist[N].push\_back(temp);

temp.nomer = N;

relist[kol1].push\_back(temp);

}

N2++;

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

for (int counter = 0; counter < relist[i].GetSize(); counter++){

for (int j = 0; j < relist[i].GetSize() - 1 ; j++){

if (relist[i][j].nomer > relist[i][j+1].nomer) {

relist[i][j].nomer = relist[i][j].nomer ^ relist[i][j+1].nomer;

relist[i][j + 1].nomer = relist[i][j+1].nomer ^ relist[i][j].nomer;

relist[i][j].nomer = relist[i][j].nomer ^ relist[i][j+1].nomer;

}

}

}

}

///////////////////////////////////////////////////////////////

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

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

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

}

printf("\n");

}

printf("\n");

printf("\n");

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

if (relist[i].GetSize()){

printf("V%d:", i + 1);

for (int j = 0; j < relist[i].GetSize(); j++) {

printf(" v%d", relist[i][j].nomer + 1);

}

}

printf("\n");

}

printf("\n\n");

\_getch();

}

**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.

void zad23(){

const unsigned int N1 = 5;

const unsigned int N2 = 10;

int m1[N1][N1];

int m2[N2][N2];

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

int minn = min(N1,N2);

int c[N][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");

printf("\n");

printf("Resltat ob'edinenia");

printf("\n");

printf("\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");

}

//////////////////////////////////////////////////////////////

printf("\n");

printf("\n");

printf("Resltat peresechenie");

printf("\n");

printf("\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");

}

//////////////////////////////////////////////////////////////////////

printf("\n");

printf("\n");

printf("Resltat kolcevaai summa");

printf("\n");

printf("\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])) || ((!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");

}

printf("\n\n");

\_getch();

}

**Задание 4 \***

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

Результат выполнения операции выведите на экран.

void zad4() {

const unsigned int N1 = 4;

const unsigned int N2 = 4;

const unsigned int N3 = N1\*N2;

int a[N1][N1];

int b[N2][N2];

int c[N3][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("Decartovoe proizvedenie: \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("Result: ");

printf("\n");

s

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

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

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

}

printf("\n");

}

\_getch();

}

**Вывод:** Научились выполнять над графами различные операции: отождествление вершин, стягивание ребра, расщепление вершины. Так же освоили работу с динамическими структурами данных и их применение для работы с графами.

.