Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра "Вычислительная техника"

Отчет

по лабораторной работе №6

по курсу “Л и ОА в ИЗ”

на тему “Унарные и бинарные операции над графами”

Выполнили студенты группы 22ВВС1:

Агапов И.Е.

Братчиков А.А.

Приняли:

Юрова О.В.

Акифьев И.В.

Пенза 2023

### Лабораторное 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  *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*1X *G*2.
2. Результат выполнения операции выведите на экран.

**Листинг**

Для заданий 1 и 2

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <iostream>

typedef struct Node {

int value;

struct Node\* next;

} Node;

Node\* create(int data)

{

// Выделение памяти под корень списка

Node\* tmp = (Node\*)malloc(sizeof(Node));

tmp->value = data;

// Присваивание указателю на следующий элемент значения NULL

tmp->next = NULL;

return(tmp);

}

Node\* pushStart(int data, Node\* head)

{

Node\* tmp = create(data);

// Присваивание указателю на следующий элемент значения указателя на «голову»

// первоначального списка

tmp->next = head;

return(tmp);

}

int main() {

srand(22341);

int size = 0;

int w;

int max = 0;

int min = 0;

scanf("%d", &size);

w = size;

int\*\* mas;

int\*\* nas;

mas = (int\*\*)malloc(sizeof(int\*) \* size);

nas = (int\*\*)malloc(sizeof(int\*) \* size);

Node\*\* sps;

Node\*\* last;

Node\*\* deleted;

Node\* tmp;

deleted = (Node\*\*)malloc(sizeof(Node\*) \* size);

sps = (Node\*\*)malloc(sizeof(Node\*) \* size);

last = (Node\*\*)malloc(sizeof(Node\*) \* size);

int size1 = size;

int i = 0;

int j = 0;

int s = 0;

setlocale(LC\_ALL, "");

while (size1 > i) {

j = 0;

sps[i] = create(i);

last[i] = sps[i];

mas[i] = (int\*)malloc(sizeof(int) \* w);

nas[i] = (int\*)malloc(sizeof(int) \* w);

while (w > j) {

mas[i][j] = 0;

nas[i][j] = 0;

j++;

}

i++;

}

i = 0;

j = 0;

bool fl = true;

int h = 0;

while (size1 > i) {

j = i;

while (w > j) {

if (i == j) {

mas[i][j] = 0;

nas[i][j] = 0;

}

else {

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

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

mas[j][i] = mas[i][j];

nas[j][i] = nas[i][j];

}

j++;

}

i++;

}

i = 0;

j = 0;

while (size1 > i) {

j = 0;

while (w > j) {

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

if (fl) {

last[i] = pushStart(j + 1, NULL);

fl = false;

}

else {

last[i] = pushStart(j + 1, last[i]);

}

}

j++;

}

i++;

fl = true;

}

for (int i = 0; i < size1; i++) {

sps[i] = last[i];

}

for (int i = 0; i < size1; i++) {

printf("%d\t", i + 1);

while (last[i] != NULL) {

printf("%d\t", last[i]->value);

last[i] = last[i]->next;

}

printf("\n");

}

printf("\t");

for (int o = 0; o < size; o++) {

printf("%d\t", o + 1);

}

printf("\n");

for (int o = 0; o < size; o++) {

printf("%d\t", o + 1);

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

printf("%d\t", mas[o][n]);

}

printf("\n");

}

printf("\n");

printf("\t");

for (int o = 0; o < size; o++) {

printf("%d\t", o + 1);

}

printf("\n");

for (int o = 0; o < size; o++) {

printf("%d\t", o + 1);

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

printf("%d\t", nas[o][n]);

}

printf("\n");

}

//матричное представление

printf("Отождествление вершины юху\n");

int a;

int b;

scanf("%d", &a);

scanf("%d", &b);

if (a > b) {

min = b - 1;

max = a - 1;

}

else {

min = a - 1;

max = b - 1;

}

bool flag = false;

if (mas[min][max] == 1) {

flag = true;

}

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

mas[min][n] = mas[min][n] | mas[max][n];

}

while (max + 1 < size - 1) {

for (int o = min; o < size; o++) {

mas[o][max] = mas[o][max + 1];

}

max++;

}

for (int o = 0; o < size; o++) {

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

mas[n][o] = mas[o][n];

if (n == o) {

mas[o][n] = 0;

}

}

}

if (flag) {

mas[min][min] = 1;

}

int g = 1;

printf("\t");

for (int o = 0; o < size - g; o++) {

printf("%d\t", o + 1);

}

printf("\n");

for (int o = 0; o < size - g; o++) {

printf("%d\t", o + 1);

for (int n = 0; n < w - g; n++) {

printf("%d\t", mas[o][n]);

}

printf("\n");

}

printf("Стягивание ребра юху\n");

scanf("%d", &a);

scanf("%d", &b);

if (a > b) {

min = b - 1;

max = a - 1;

}

else {

min = a - 1;

max = b - 1;

}

if (mas[min][max] == 1) {

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

mas[min][n] = mas[min][n] | mas[max][n];

}

while (max + 1 < size) {

for (int o = 0; o < size; o++) {

mas[max][o] = mas[max + 1][o];

}

max++;

}

for (int o = 0; o < size; o++) {

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

mas[o][n] = mas[n][o];

if (n == o) {

mas[o][n] = 0;

}

}

}

g++;

printf("\t");

for (int o = 0; o < size - g; o++) {

printf("%d\t", o + 1);

}

printf("\n");

for (int o = 0; o < size - g; o++) {

printf("%d\t", o + 1);

for (int n = 0; n < w - g; n++) {

printf("%d\t", mas[o][n]);

}

printf("\n");

}

}

else {

printf("Операция невозможна\n");

}

printf("Расщепление вершины юху\n");

scanf("%d", &a);

a--;

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

mas[size - g][n] = mas[a][n];

}

mas[size - g][a] = 1;

for (int o = 0; o < size; o++) {

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

mas[o][n] = mas[n][o];

}

}

mas[size - g][size - g] = 0;

printf("\t");

g--;

for (int o = 0; o < size - g; o++) {

printf("%d\t", o + 1);

}

printf("\n");

for (int o = 0; o < size - g; o++) {

printf("%d\t", o + 1);

for (int n = 0; n < w - g; n++) {

printf("%d\t", mas[o][n]);

}

printf("\n");

}

//списковое представление

printf("Отождествление вершины юху feat. spiski\n");

scanf("%d", &a);

scanf("%d", &b);

if (a > b) {

min = b - 1;

max = a - 1;

}

else {

min = a - 1;

max = b - 1;

}

int del = 0;

deleted[del] = sps[max];

last[min] = sps[min];

last[max] = sps[max];

flag = false;

while (last[min] != NULL) {

if ((last[max]->next)->value == last[min]->value) {

tmp = last[max]->next;

last[max]->next = last[max]->next->next;

free(tmp);

break;

}

last[min] = last[min]->next;

}

last[min]->next = last[max];

for (int i = 0; i < size1; i++) {

last[i] = sps[i];

}

for (int i = 0; i < size1; i++) {

if (last[i] != deleted[del]) {

printf("%d\t", i + 1);

while (last[i] != NULL) {

printf("%d\t", last[i]->value);

last[i] = last[i]->next;

}

printf("\n");

}

}

printf("Расщепление вершины юху feat. spiski\n");

scanf("%d", &a);

a--;

tmp = create(1);

last[a]=sps[a];

while (last[a]->next != NULL) {

last[a] = last[a]->next;

}

last[a]->next=tmp;

sps[a+1]=sps[a];

printf("\n");

for (int i = 0; i < size1; i++) {

last[i] = sps[i];

}

for (int i = 0; i < size1; i++) {

printf("%d\t", i + 1);

while (last[i] != NULL) {

if(last[i]->value==i+1){

printf("%d\t", last[i]->value+1);

}else{

printf("%d\t", last[i]->value);

}

last[i] = last[i]->next;

}

printf("\n");

}

}

Для заданий 3 и 4

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <iostream>

int main() {

srand(353415);

int size = 0;

int sizeO = 0;

int w;

int max = 0;

int min = 0;

scanf("%d", &size);

scanf("%d", &sizeO);

w = size;

int\*\* mas;

int\*\* nas;

int\*\* pas;

int\*\* ob;

int\*\* per;

int\*\* col;

int\*\* dec;

mas = (int\*\*)malloc(sizeof(int\*) \* size);

nas = (int\*\*)malloc(sizeof(int\*) \* size);

pas = (int\*\*)malloc(sizeof(int\*) \* sizeO);

ob = (int\*\*)malloc(sizeof(int\*) \* size);

per = (int\*\*)malloc(sizeof(int\*) \* size);

col = (int\*\*)malloc(sizeof(int\*) \* size);

dec = (int\*\*)malloc(sizeof(int\*) \* size \* sizeO);

int size1 = size;

int size2 = size\*sizeO;

int i = 0;

int j = 0;

int s = 0;

setlocale(LC\_ALL, "");

for(int i=0;size2>i;i++){

j = 0;

dec[i] = (int\*)malloc(sizeof(int) \* size2);

for(int j=0;size2>j;j++){

dec[i][j]=0;

}

}

while (sizeO > i) {

j = 0;

pas[i] = (int\*)malloc(sizeof(int) \* sizeO);

while (sizeO > j) {

pas[i][j] = 0;

j++;

}

i++;

}

i = 0;

j = 0;

int h = 0;

while (sizeO > i) {

j = i;

while (sizeO > j) {

if (i == j) {

pas[i][j] = 0;

}

else {

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

pas[j][i] = pas[i][j];

}

j++;

}

i++;

}

i=0;

while (size1 > i) {

j = 0;

mas[i] = (int\*)malloc(sizeof(int) \* w);

nas[i] = (int\*)malloc(sizeof(int) \* w);

ob[i] = (int\*)malloc(sizeof(int) \* w);

per[i] = (int\*)malloc(sizeof(int) \* w);

col[i] = (int\*)malloc(sizeof(int) \* w);

while (w > j) {

mas[i][j] = 0;

nas[i][j] = 0;

ob[i][j] = 0;

per[i][j] = 0;

col[i][j] = 0;

j++;

}

i++;

}

i = 0;

j = 0;

while (size1 > i) {

j = i;

while (w > j) {

if (i == j) {

mas[i][j] = 0;

nas[i][j] = 0;

}

else {

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

mas[j][i] = mas[i][j];

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

nas[j][i] = nas[i][j];

}

j++;

}

i++;

}

for (int i=0;size1 > i;i++) {

for (int j = 0;w > j;j++) {

if (mas[i][j] == 0 && nas[i][j] == 0) {

ob[i][j] = 0;

}

else { ob[i][j] = 1;

}

}

}

for (int i = 0;size1 > i;i++) {

for (int j = 0;w > j;j++) {

if (mas[i][j] == 1 && nas[i][j] == 1) {

per[i][j] = 1;

}

else {

per[i][j] = 0;

}

}

}

for (int i = 0;size1 > i;i++) {

for (int j = 0;w > j;j++) {

if ((mas[i][j]==1 && nas[i][j]==1)||(mas[i][j] == 0 && nas[i][j] == 0)) {

col[i][j] = 0;

}

else {

col[i][j] = 1;

}

}

}

int v1=0;

int v2=0;

for(int i=0;size>i;i++){

for(int j=0;sizeO>j;j++){

for(int k=0;size>k;k++){

for(int l=0;sizeO>l;l++){

if (i==k){

dec[v1][v2]=pas[j][l];

}

if(j==l){

dec[v1][v2]=mas[i][k];

}

v2++;

}

}

v2=0;

v1++;

}

}

printf("Первая матирца\n");

for (int o = 0; o < size; o++) {

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

printf("%d\t", mas[o][n]);

}

printf("\n");

}

printf("\n");

printf("Вторая матирца\n");

for (int o = 0; o < size; o++) {

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

printf("%d\t", nas[o][n]);

}

printf("\n");

}

printf("Третья матирца\n");

for (int o = 0; o < sizeO; o++) {

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

printf("%d\t", pas[o][n]);

}

printf("\n");

}

printf("\n");

printf("Матирца объединения \n");

for (int o = 0; o < size; o++) {

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

printf("%d\t", ob[o][n]);

}

printf("\n");

}

printf("Матирца пересечения \n");

for (int o = 0; o < size; o++) {

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

printf("%d\t", per[o][n]);

}

printf("\n");

}

printf("Матирца кольцевой суммы \n");

for (int o = 0; o < size; o++) {

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

printf("%d\t", col[o][n]);

}

printf("\n");

}

printf("\n");

printf("Матирца декартового произведения \n");

for (int o = 0; o < size2; o++) {

for (int n = 0; n < size2; n++) {

printf("%d\t", dec[o][n]);

}

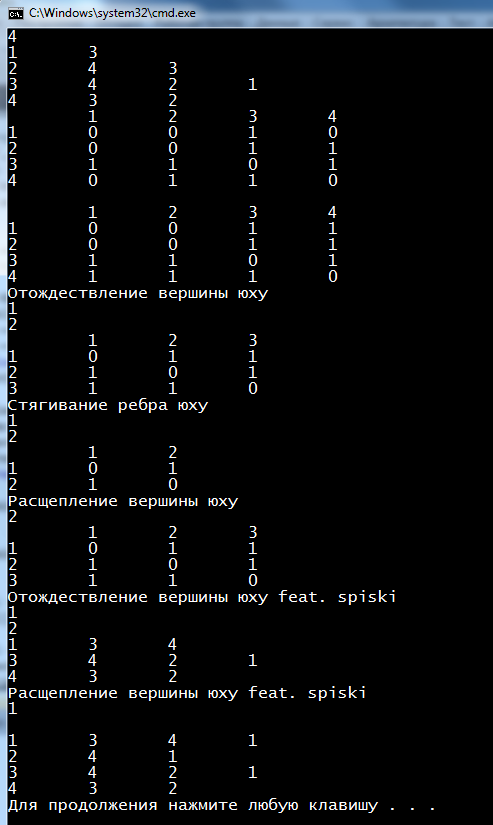
printf("\n");

}

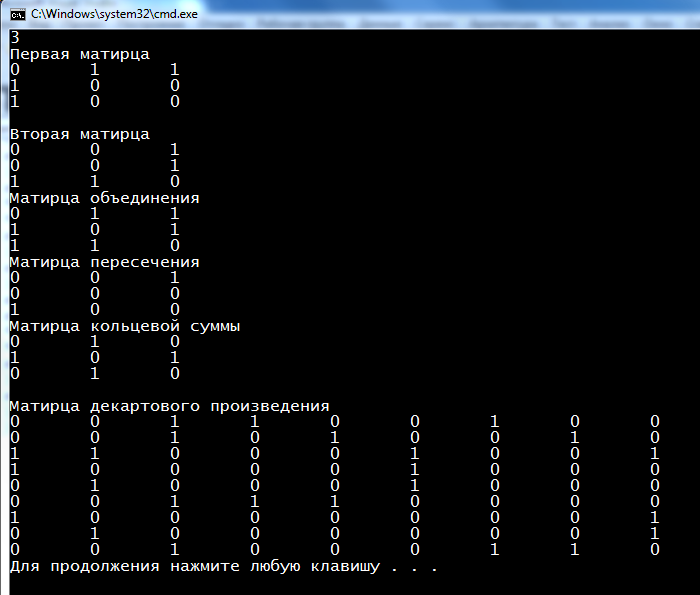
}

**Результаты работ программ**

Для заданий 1 и 2



Для заданий 3 и 4



**Вывод**

### В ходе выполнения лабораторной работы были выполнены операции отождествления вершин, стягивания ребра, расщепления вершины для матричной формы представления графов и для списка смежности.

### Были выполнены бинарные операции объединения, пересечения, кольцевой суммы, декартового произведения.