Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу “Логика и основа алгоритмизации в ИЗ”

на тему “Унарные и бинарные операции над графами”

Выполнили студенты группы 21ВВ3

Чабуев Р.

Костюков И.

Столяров А.

Рузляев Д.

Приняли:

д.т.н., профессор Митрохин М.А.

к.т.н., доцент Юрова О.В.

Пенза 2022

**Название:**

Унарные и бинарные операции над графами.

**Лабораторное задание:**

1. Сгенерировать (используя генератор случайных чисел) две матрицы M1, М2 смежности неориентированных помеченных графов G1, G2. \*Для указанных графов преобразовать представление матриц смежности в списки смежности.
2. Для матричной формы представления графов выполнить операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

\*Для представления графов в виде списков смежности выполнить операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

1. Для матричной формы представления графов выполнить операцию:

а) объединения

б) пересечения

в) кольцевой суммы

1. \*Для матричной формы представления графов выполнить операцию декартова произведения графов.

**Листинг:**

**Программа к заданию 1:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

struct Node {

int data = 0;

Node\* next = NULL;

};

int\*\* G1, \*\* G2;

int razmer = 0;

printf("Введите размер матрицы: ");

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

G1 = (int\*\*)malloc(razmer \* sizeof(int\*));

G2 = (int\*\*)malloc(razmer \* sizeof(int\*));

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

G1[i] = (int\*)malloc(razmer \* sizeof(int));

G2[i] = (int\*)malloc(razmer \* sizeof(int));

}

Node\*\* lastG1 = (Node\*\*)malloc(razmer \* sizeof(Node\*));

Node\*\* lastG2 = (Node\*\*)malloc(razmer \* sizeof(Node\*));

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

{

lastG1[i] = (Node\*)malloc(razmer \* sizeof(Node));

lastG2[i] = (Node\*)malloc(razmer \* sizeof(Node));

}

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

{

lastG1[i]->data = 0;

lastG1[i]->next = NULL;

lastG2[i]->data = 0;

lastG2[i]->next = NULL;

}

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

for (int j = i; j < razmer; j++) {

if (i == j) {

G1[i][j] = 0;

G2[i][j] = 0;

}

else {

G1[i][j] = rand() % 2;

G2[i][j] = rand() % 2;

G1[j][i] = G1[i][j];

G2[j][i] = G2[i][j];

}

}

}

printf("\nПервая матрица смежности:\n\n ");

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

printf(" %d", i);

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

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

printf(" %d | ", i);

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

printf("%d ", G1[i][j]);

}

printf("\n");

}

printf("\nПервый список смежности:\n\n ");

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

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

if (G1[i][j] == 1) {

Node\* switchCell = (Node\*)malloc(razmer \* sizeof(Node));

if (lastG1[i]->next != NULL) switchCell->next = lastG1[i]->next;

else switchCell->next = NULL;

switchCell->data = j;

lastG1[i]->next = switchCell;

}

}

}

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

Node\* sw = (Node\*)malloc(razmer \* sizeof(Node));

sw = lastG1[i];

printf("%d:", i);

while (sw->next != NULL) {

printf(" %d", sw->next->data);

sw = sw->next;

}

free(sw);

printf("\n ");

}

printf("\nВторая матрица смежности:\n\n ");

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

printf(" %d", i);

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

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

printf(" %d | ", i);

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

printf("%d ", G2[i][j]);

}

printf("\n");

}

printf("\nВторой список смежности:\n\n ");

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

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

if (G2[i][j] == 1) {

Node\* switchCell = (Node\*)malloc(razmer \* sizeof(Node));

if (lastG2[i]->next != NULL) switchCell->next = lastG2[i]->next;

else switchCell->next = NULL;

switchCell->data = j;

lastG2[i]->next = switchCell;

}

}

}

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

Node\* sw = (Node\*)malloc(razmer \* sizeof(Node));

sw = lastG2[i];

printf("%d:", i);

while (sw->next != NULL) {

printf(" %d", sw->next->data);

sw = sw->next;

}

free(sw);

printf("\n ");

}

}

**Программа к заданию 2.1 (матрица):**

#include <stdio.h>

#include <windows.h>

#include <time.h>

int flag = -1;

void otozhd(int n, int\*\* G)

{

int t, p;

printf("Введите вершину 1 для операции отождествления вершин: ");

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

printf("Введите вершину 2 для операции отождествления вершин: ");

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

printf("Итоговая матрица смежности:\n");

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

{

if ((G[t][p] == 1) && (G[p][t] == 1))

{

flag = t;

}

if ((G[i][t] == 1) || (G[i][p] == 1))

{

G[i][t] = 1;

G[t][i] = 1;

}

}

for (int i = 0; i != p; i++) {

for (int k = 0; k < (n - p - 1); k++) {

G[i][p + k] = G[i][p + k + 1];

G[p + k][i] = G[p + k + 1][i];

}

}

for (int i = p; i != n - 1; i++) {

for (int k = 0; k < (n - p - 1); k++) {

G[i][p + k] = G[i + 1][p + k + 1];

G[p + k][i] = G[p + k + 1][i + 1];

}

}

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - 1; j++)

{

if (i == j)

{

G[i][j] = 0;

}

if (flag == t)

{

G[t][t] = 1;

}

printf("%d\t", G[i][j]);

}

printf("\n");

}

}

void styagivanie(int n, int\*\* G)

{

int v1, v2;

printf("\nВведите вершину 1 для операции стягивания ребра: ");

scanf\_s("%d", &v1);

printf("Введите вершину 2 для операции стягивания ребра: ");

scanf\_s("%d", &v2);

if (G[v1][v2] != 1) return;

printf("Итоговая матрица смежности:\n");

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

{

if ((G[i][v1] == 1) || (G[i][v2] == 1))

{

G[i][v1] = 1;

G[v1][i] = 1;

}

}

for (int i = 0; i != v2; i++) {

for (int k = 0; k < (n - v2 - 2); k++) {

G[i][v2 + k] = G[i][v2 + k + 1];

G[v2 + k][i] = G[v2 + k + 1][i];

}

}

for (int i = v2; i != n - 2; i++) {

for (int k = 0; k < (n - v2 - 2); k++) {

G[i][v2 + k] = G[i + 1][v2 + k + 1];

G[v2 + k][i] = G[v2 + k + 1][i + 1];

}

}

for (int i = 0; i < n - 2; i++)

{

for (int j = 0; j < n - 2; j++)

{

if (i == j)

{

G[i][j] = 0;

}

if ((flag == v1) || (flag == v2))

{

G[v1][v1] = 1;

}

else if (flag > v2) {

G[flag - 1][flag - 1] = 1;

}

else if ((flag != -1) && (flag != v1) && (flag < v2)) {

G[flag][flag] = 1;

}

printf("%d\t", G[i][j]);

}

printf("\n");

}

}

void rashep(int n, int\*\* G)

{

int s;

for (int i = 0; i != n - 2; i++)

{

if (G[i][i] == 1) flag = i;

}

printf("\nВведите вершину для операции расщепления: ");

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

printf("Итоговая матрица смежности:\n");

for (int i = 0; i != s + 1; i++) {

for (int k = 1; k < (n - s - 2); k++) {

G[i][n - k - 1] = G[i][n - k - 2];

G[n - k - 1][i] = G[n - k - 2][i];

}

}

for (int i = s + 2; i != n - 2; i++) {

for (int k = 1; k < (n - s - 2); k++) {

G[i][n - k - 1] = G[i - 1][n - k - 2];

G[n - k - 1][i] = G[n - k - 2][i - 1];

}

}

for (int i = 0; i != n - 1; i++)

{

G[i][s + 1] = G[i][s];

G[s + 1][i] = G[i][s + 1];

}

for (int i = 0; i != n - 1; i++)

{

if (G[i][s] == 1)

{

G[s][s + 1] = 1;

G[s + 1][s] = 1;

}

}

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - 1; j++)

{

if (i == j)

{

G[i][j] = 0;

}

if ((flag <= s) && (flag != -1))

{

G[flag][flag] = 1;

}

if (flag > s)

{

G[flag + 1][flag + 1] = 1;

}

printf("%d\t", G[i][j]);

}

printf("\n");

}

}

int main(void)

{

SetConsoleOutputCP(1251);

srand(time(NULL));

int n, \*\* G;

printf("Введите n: ");

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

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

printf("Матрица смежности:\n");

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

{

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

}

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

{

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

{

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

if (i == j)

{

G[i][j] = 0;

}

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

}

}

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

{

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

{

printf("%d\t", G[i][j]);

}

printf("\n");

}

printf("\n");

otozhd(n, G);

styagivanie(n, G);

rashep(n, G);

}

**Программа к заданию 3:**

#include <stdio.h>

#include <windows.h>

#include <time.h>

void generation(int n, int\*\* Z)

{

printf("Матрица смежности:\n");

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

{

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

}

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

{

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

{

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

if (i == j)

{

Z[i][j] = 0;

}

Z[j][i] = Z[i][j];

}

}

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

{

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

{

printf("%d\t", Z[i][j]);

}

printf("\n");

}

}

void operations(int n1, int n2, int\*\* G, int\*\* M)

{

int m, b;

int\*\* L, \*\* P, \*\* J;

if (n1 <= n2) {

m = n1;

b = n2;

J = M;

}

else {

m = n2;

b = n1;

J = G;

}

printf("\nПересечение:\n");

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

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

{

L[i] = (int\*)malloc(m \* sizeof(int));

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

{

if (G[i][j] + M[i][j] == 2)

L[i][j] = 1;

else L[i][j] = 0;

}

}

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

{

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

{

printf("%d\t", L[i][j]);

}

printf("\n");

}

printf("\nКольцевая сумма:\n");

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

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

{

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

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

{

P[i][j] = J[i][j];

}

}

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

{

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

{

P[i][j] = G[i][j] ^ M[i][j];

}

}

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

{

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

{

printf("%d\t", P[i][j]);

}

printf("\n");

}

printf("\nОбъединение:\n");

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

{

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

{

if (G[i][j] + M[i][j] == 1)

J[i][j] = 1;

}

}

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

{

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

{

printf("%d\t", J[i][j]);

}

printf("\n");

}

}

int main(void)

{

SetConsoleOutputCP(1251);

srand(time(NULL));

int n1, n2, \*\* G, \*\* M;

printf("Введите n1: ");

scanf\_s("%d", &n1);

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

generation(n1, G);

printf("\nВведите n2: ");

scanf\_s("%d", &n2);

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

generation(n2, M);

operations(n1, n2, G, M);

}

**Программа к заданию 4:**

#include <stdio.h>

#include <windows.h>

#include <time.h>

void generation(int n, int\*\* Z)

{

printf("Матрица смежности:\n");

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

{

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

}

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

{

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

{

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

if (i == j)

{

Z[i][j] = 0;

}

Z[j][i] = Z[i][j];

}

}

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

{

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

{

printf("%d\t", Z[i][j]);

}

printf("\n");

}

}

void dekart(int n1, int n2, int\*\* G, int\*\* M)

{

printf("\nДекартово произведение графов:\n");

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

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

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

{

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

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

{

if (i == j)

L[i][j] = 1;

else L[i][j] = 0;

}

}

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

{

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

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

{

if (i == j)

K[i][j] = 1;

else K[i][j] = 0;

}

}

int\*\* O = (int\*\*)malloc((n1 \* n2) \* sizeof(int));

for (int i = 0; i != n1; i++) {

for (int j = 0; j != n2; j++) {

O[i \* n2 + j] = (int\*)malloc((n1 \* n2) \* sizeof(int));

for (int l = 0; l != n1; l++) {

for (int v = 0; v != n2; v++) {

if (i == l) {

O[i \* n2 + j][l \* n2 + v] = M[j][v];

}

else if (j == v) {

O[i \* n2 + j][l \* n2 + v] = G[i][l];

}

else O[i \* n2 + j][l \* n2 + v] = 0;

}

}

}

}

for (int i = 0; i != n1 \* n2; i++) {

for (int j = 0; j != n1 \* n2; j++) {

printf("%d\t", O[i][j]);

}

printf("\n");

}

}

int main(void)

{

SetConsoleOutputCP(1251);

srand(time(NULL));

int n1, n2, \*\* G, \*\* M;

printf("Введите n1: ");

scanf\_s("%d", &n1);

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

generation(n1, G);

printf("\nВведите n2: ");

scanf\_s("%d", &n2);

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

generation(n2, M);

dekart(n1, n2, G, M);

}

**Результаты выполнения программ:**

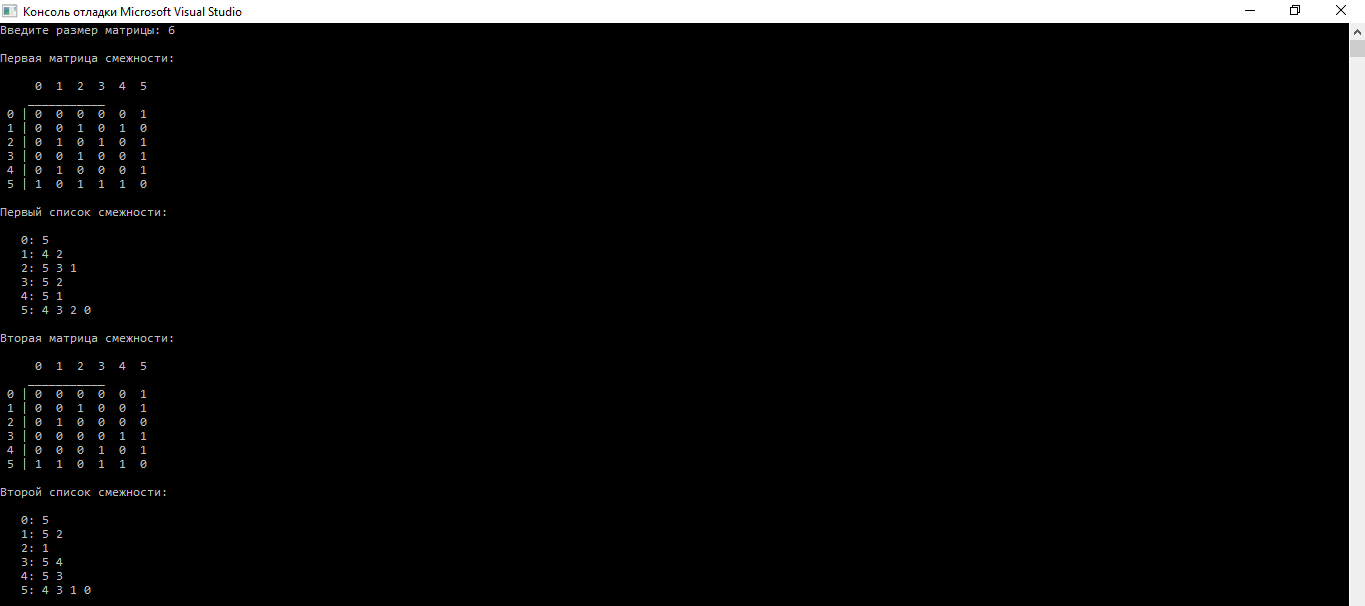


Рисунок 1 – Результат выполнения программы к заданию 1

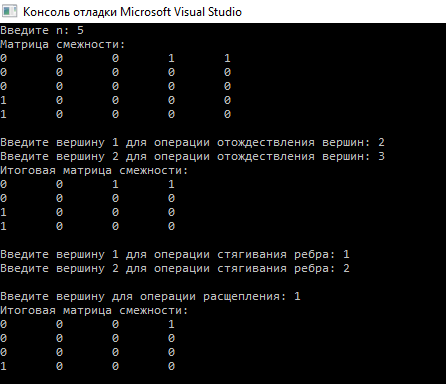


Рисунок 2 – Результат выполнения программы к заданию 2.1

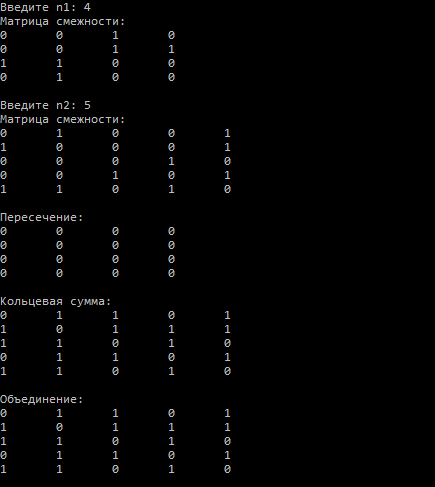


Рисунок 3 – Результат выполнения программы к заданию 3

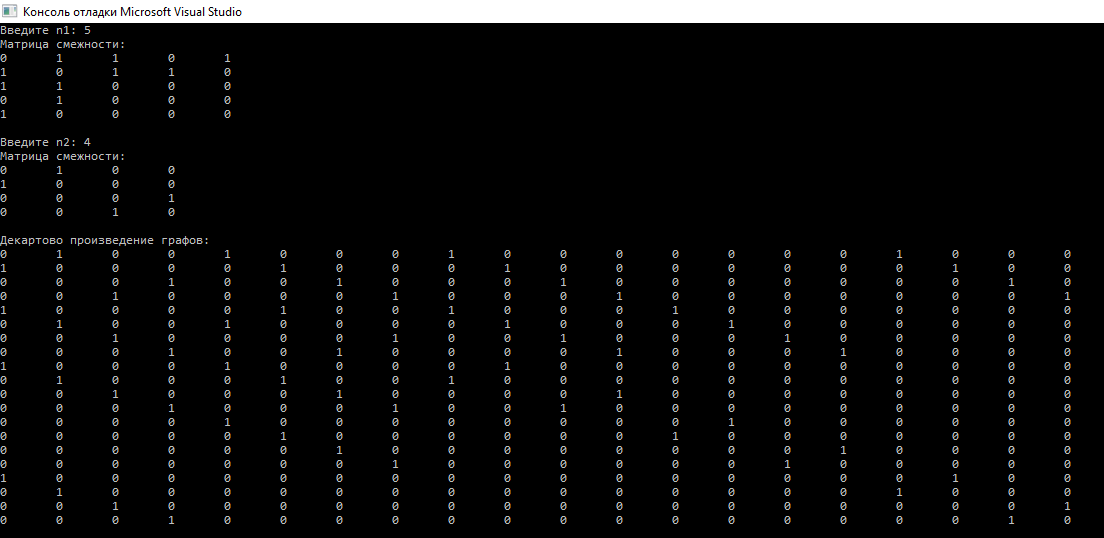


Рисунок 4 – Результат выполнения программы к заданию 4

**Вывод:** в ходе выполнения лабораторной работы было реализовано представление матриц смежности в списки смежности графов; над графом произведены операции отождествления вершин, стягивания ребра и расщепления вершины; реализованы операции пересечения, объединения, кольцевой суммы и декартова произведения графов.