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

Тюкалов В.Е.

Чинов Д.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:**

//Выполнили студенты группы 21ВВ3: Тюкалов В.Е. и Чинов Д.Д.

#include<iostream>

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<math.h>

#include<windows.h>

typedef struct Node {

int data = 0;

Node\* next = NULL;

};

void freeList(int total, Node\*\* lastG) {

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

Node\* swG1 = lastG[i];

Node\* switchCell = lastG[i];

while (swG1 != NULL) {

swG1 = switchCell->next;

free(switchCell);

switchCell = swG1;

}

}

free(lastG);

}

void freeMatrix(int total, int\*\* G) {

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

free(G[i]);

}

free(G);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

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

int total = 0, v1 = 0, v2 = 0, countI = 1, countJ = 1, num = 0, count = 0;

int countK = 0, variant = 0, countI1 = 0, countI2 = 0, countList = 0;

Node\*\* lastG1 = NULL, \*\* lastG2 = NULL, \*\* lastG = NULL;

Node\* last = NULL;

Node\* Cell = NULL;

Node\* CellLast = NULL;

printf("Введите размер матрицы: ");

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

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

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

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

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

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

}

srand(time(NULL));

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

for (int j = i; j < total; 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];

}

}

}

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

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

lastG = (Node\*\*)malloc(total \* sizeof(Node\*));

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

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

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

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

}

printf("\n Матрица смежности 1\n");

printf("\n ");

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

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < total \* 3; i++) {

printf("\_");

}

printf("\n");

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

printf(" %c | ", (i + 97));

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

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

}

printf("\n");

}

printf("\n Список смежности 1\n");

printf("\n ");

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

Cell = NULL;

lastG1[i] = NULL;

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

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

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

newCell->data = j;

newCell->next = NULL;

if (Cell != NULL) Cell->next = newCell;

else lastG1[i] = newCell;

Cell = newCell;

}

}

}

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

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

sw = lastG1[i];

printf("%c:", i + 97);

while (sw != NULL) {

printf(" %c", sw->data + 97);

sw = sw->next;

}

free(sw);

printf("\n ");

}

printf("\n Матрица смежности 2\n");

printf("\n ");

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

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < total\*3; i++) {

printf("\_");

}

printf("\n");

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

printf(" %c | ", (i + 97));

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

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

}

printf("\n");

}

printf("\n Список смежности 2\n");

printf("\n ");

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

Cell = NULL;

lastG2[i] = NULL;

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

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

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

newCell->data = j;

newCell->next = NULL;

if (Cell != NULL) Cell->next = newCell;

else lastG2[i] = newCell;

Cell = newCell;

}

}

}

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

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

sw = lastG2[i];

printf("%c:", i + 97);

while (sw != NULL) {

printf(" %c", sw->data + 97);

sw = sw->next;

}

free(sw);

printf("\n ");

}

printf("\n");

/\*-------------------------------------------------------\*/

while (variant == 0) {

printf("Введите вершину 1: ");

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

printf("Введите вершину 2: ");

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

if (v1 < total && v2 < total) variant = 1;

else printf("Выберите вершины от 0 до %d\n", (total-1));

}

int\*\* G = (int\*\*)malloc((total - 1) \* sizeof(int\*));

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

G[i] = (int\*)malloc((total - 1) \* sizeof(int));

}

if (v1 > v2) {

num = v1;

v1 = v2;

v2 = num;

}

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

if (i == v1 || i == v2) continue;

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

if (j == v1 || j == v2) continue;

G[countI][countJ] = G1[i][j];

countJ++;

}

countJ = 1;

countI++;

}

countI = 0;

countJ = 0;

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

if (i == v1) {

countI = 1;

continue;

}

if ((G1[v1][i] || G1[v2][i] == 1) && i != 0) {

G[0][i - countI] = 1;

G[i - countI][0] = 1;

}

else {

G[0][i - countI] = 0;

G[i - countI][0] = 0;

}

}

printf("\n Матрица смежности \"Задание 2\"\n");

printf("\n ");

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

if (i == 0) printf(" %c", (total + 97));

else if (i > v1) {

if ((i + 1) > v2) printf(" %c", (i + 2 + 96));

else printf(" %c", (i + 1 + 96));

}

else printf(" %c", (i + 96));

}

printf("\n ");

for (int i = 0; i < (total-1) \* 3; i++) {

printf("\_");

}

printf("\n");

count = 0;

countJ = 0;

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

if (i == 0) printf(" %c | ", (total + 97));

else if (i > v1) {

if ((i + 1) > v2) printf(" %c | ", (i + 2 + 96));

else printf(" %c | ", (i + 1 + 96));

}

else printf(" %c | ", (i + 96));

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

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

}

printf("\n");

}

/\*printf("\n Список смежности \"Задание 2\"\n");

printf("\n ");

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

Cell = NULL;

lastG[i] = NULL;

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

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

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

newCell->data = j;

newCell->next = NULL;

if (Cell != NULL) Cell->next = newCell;

else lastG[i] = newCell;

Cell = newCell;

}

}

}

int\* listZ2 = (int\*)malloc(total \* sizeof(int));

Cell = lastG1[v1];

count = 0;

while (Cell != NULL) {

listZ2[count] = Cell->data;

Cell = Cell->next;

count++;

}

Cell = lastG1[v2];

while (Cell != NULL) {

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

if (listZ2[i] != Cell->data) {

listZ2[count] = Cell->data;

Cell = Cell->next;

count++;

}

}

}

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

Cell = NULL;

CellLast = lastG1[i];

lastG[i] = NULL;

if(i == 0)

while (count - 1 > 0) {

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

newCell->data = listZ2[count];

newCell->next = NULL;

if (Cell != NULL) Cell->next = newCell;

else lastG[i] = newCell;

Cell = newCell;

}

else

while (CellLast != NULL) {

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

if (newCell->data != CellLast->data) newCell->data = CellLast->data;

else countList++;

newCell->next = NULL;

if (Cell != NULL) Cell->next = newCell;

else lastG[i] = newCell;

Cell = newCell;

CellLast = CellLast->next;

if (CellLast == NULL && countList > 0) {

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

newCell->data = CellLast->data;

newCell->next = NULL;

}

}

}

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

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

sw = lastG[i];

printf("%c:", i + 97);

while (sw != NULL) {

printf(" %c", sw->data + 97);

sw = sw->next;

}

free(sw);

printf("\n ");

}\*/

printf("\n");

/\*-------------------------------------------------------\*/

int\*\* gO = (int\*\*)malloc((total) \* sizeof(int\*));

int\*\* gP = (int\*\*)malloc((total) \* sizeof(int\*));

int\*\* gK = (int\*\*)malloc((total) \* sizeof(int\*));

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

gO[i] = (int\*)malloc((total) \* sizeof(int));

gP[i] = (int\*)malloc((total) \* sizeof(int));

gK[i] = (int\*)malloc((total) \* sizeof(int));

}

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

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

if (G1[i][j] == 1 || G2[i][j] == 1) gO[i][j] = 1;

else gO[i][j] = 0;

if (G1[i][j] == 1 && G2[i][j] == 1) gP[i][j] = 1;

else gP[i][j] = 0;

if (G1[i][j] == 1 && G2[i][j] == 0 || G1[i][j] == 0 && G2[i][j] == 1) gK[i][j] = 1;

else gK[i][j] = 0;

}

}

printf("\n Матрица объединения графов\n");

printf("\n ");

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

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < total \* 3; i++) {

printf("\_");

}

printf("\n");

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

printf(" %c | ", (i + 97));

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

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

}

printf("\n");

}

printf("\n Матрица пересечения графов\n");

printf("\n ");

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

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < total \* 3; i++) {

printf("\_");

}

printf("\n");

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

printf(" %c | ", (i + 97));

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

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

}

printf("\n");

}

printf("\n Матрица кольцевой суммы графов\n");

printf("\n ");

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

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < total \* 3; i++) {

printf("\_");

}

printf("\n");

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

printf(" %c | ", (i + 97));

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

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

}

printf("\n");

}

freeMatrix(total, gO);

freeMatrix(total, gP);

freeMatrix(total, gK);

/\*-------------------------------------------------------\*/

printf("\n Матрица смежности 1\n");

printf("\n ");

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

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < total \* 3; i++) {

printf("\_");

}

printf("\n");

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

printf(" %c | ", (i + 97));

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

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

}

printf("\n");

}

printf("\n Матрица смежности 2\n");

printf("\n ");

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

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < total \* 3; i++) {

printf("\_");

}

printf("\n");

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

printf(" %c | ", (i + 97));

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

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

}

printf("\n");

}

int\*\* G3 = (int\*\*)malloc(pow(total, 2) \* sizeof(int\*));

for (int i = 0; i < total \* total; i++) {

G3[i] = (int\*)malloc(pow(total, 2) \* sizeof(int));

}

countI = 0;

countJ = 0;

for (int k = 0; k < total \* total; k++) {

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

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

if (countJ != countI1 && countI2 == j) G3[k][countK] = G1[countI1][countJ];

else if (countI2 != j && countJ == countI1) G3[k][countK] = G2[countI2][j];

else G3[k][countK] = 0;

countK++;

}

countJ++;

}

countK = 0;

countJ = 0;

countI2++;

if ((k + 1) % total == 0) {

countI2 = 0;

countI1++;

}

}

printf("\n Матрица декартового произведения\n");

printf("\n ");

for (int i = 0; i < total\*total; i++) {

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < (total\*total) \* 3; i++) {

printf("\_");

}

printf("\n");

for (int i = 0; i < total\*total; i++) {

printf(" %c | ", (i + 97));

for (int j = 0; j < total\*total; j++) {

printf("%d ", G3[i][j]);

}

printf("\n");

}

freeMatrix(pow(total, 2), G3);

/\*-------------------------------------------------------\*/

//freeList((total - 1), lastG);

freeList(total, lastG1);

freeList(total, lastG2);

freeMatrix((total - 1), G);

freeMatrix(total, G1);

freeMatrix(total, G2);

}

**Результаты выполнения программ:**

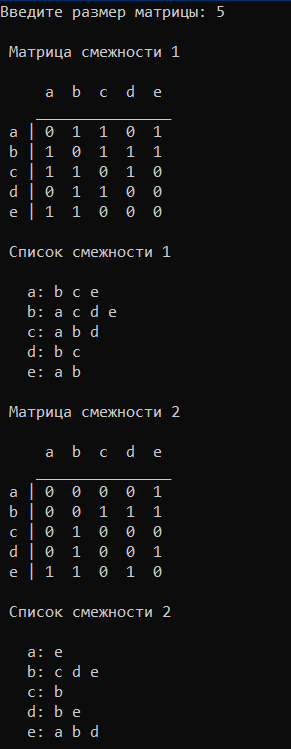


Рисунок 1 – Результат выполнения программы к заданию 1

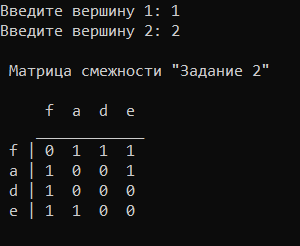


Рисунок 2 – Результат выполнения программы к заданию 2.1

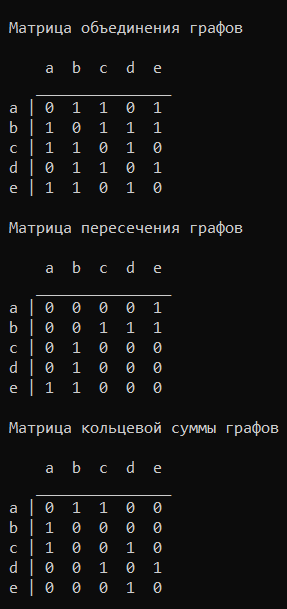


Рисунок 3 – Результат выполнения программы к заданию 3

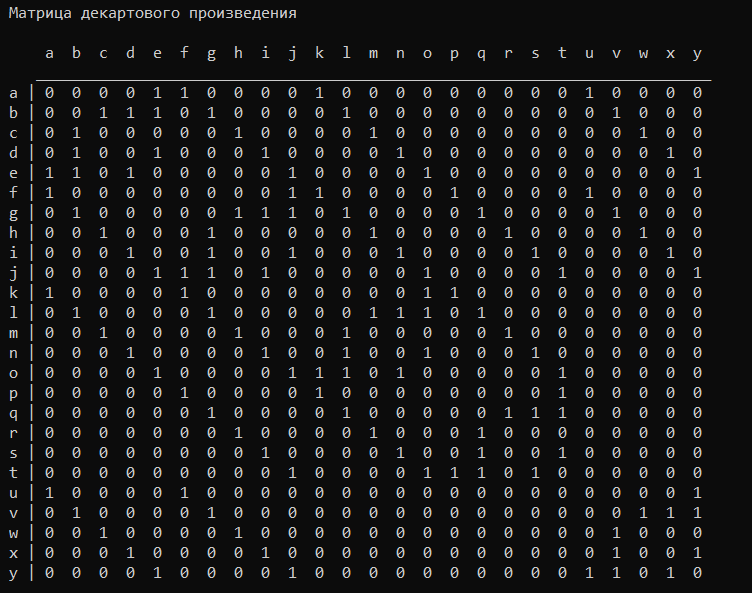


Рисунок 4 – Результат выполнения программы к заданию 4

**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.