Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Определение характеристик графов»

**Выполнили студенты группы 21вв1:**

Нечаев А.Д.

Киреев Д.А.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.

2. Определите размер графа *G*, используя матрицу смежности графа.

3. Найдите изолированные, концевые и доминирующие вершины.

**Задание 2\***

1. Постройте для графа G матрицу инцидентности.

2. Определите размер графа *G*, используя матрицу инцидентности графа.

3. Найдите изолированные, концевые и доминирующие вершины.

**Описание метода решения задачи:**

Выделили динамическую память для двумерного массива матрицы смежности. Далее выводим саму матрицу, размер матрицы и информацию о вершинах (изолированные, доминирующие или концевые). Потом выводим кол-во рёбер и выделили динамическую память для двумерного массива матрицы инцидентности. Далее выводим саму матрицу, размер матрицы и информацию о вершинах (изолированные, доминирующие или концевые).

**Листинг:**

#include <iostream>

#include <locale.h>

int main()

{

setlocale(LC\_ALL, "Rus");

int\*\* a, \*\* b, n, k = 0;

int i = 0, j = 0, w, t = 0, u = 0, q = 0, m = 0, z = 0, o = 0;

int\* arr, \* arr1;

printf("Введите кол-во строк и столбцов\n");

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

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

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

{

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

}

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

{

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

{

/\*w = rand() % 3;

if ((w == 1) || (w == 2))

{

a[i][j] = 1;

a[j][i] = 1;

}

else {

a[i][j] = 0;

a[j][i] = 0;

}

if (i == j)

{

a[i][j] = 0;

}\*/

if (i == j) { a[i][j] = 0; }

else

{

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

a[j][i] = a[i][j];

}

}

}

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

{

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

{

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

if (a[i][j] != 0)

{

m++;

}

k++;

}

printf("\n");

}

printf("Размер матрицы смежности ");

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

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

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

{

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

{

if (a[i][j] != 0)

{

t++;

}

}

arr[i] = t;

t = 0;

}

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

{

printf("%d ", arr[i]);

}

printf("\n");

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

{

if (arr[i] == 0)

{

printf("\nВершина номер %d - изолированная\n", i + 1);

}

}

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

{

if (arr[i] == 1)

{

printf("Вершина номер %d - концевая\n", i + 1);

}

}

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

{

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

{

if (a[i][j] != 0)

{

t++;

}

}

if (t == n - 1)

{

printf("Вершина номер %d - доминирующая\n", i + 1);

}

t = 0;

}

printf("\n");

q = m / 2;

printf("Кол-во рёбер");

printf(" %d\n", q);

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

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

{

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

}

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

{

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

{

if (a[i][j] != 0)

{

b[i][z] = a[i][j];

b[j][z] = a[i][j];

z++;

}

}

}

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

{

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

{

if (b[i][j] != 1)

{

b[i][j] = 0;

}

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

}

printf("\n");

}

printf("Размер матрицы инцидентности ");

printf("%d\n", n \* q);

arr1 = (int\*)malloc(n \* sizeof(int));

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

{

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

{

if (b[i][j] != 0)

{

o++;

}

}

arr1[i] = o;

o = 0;

}

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

{

printf("%d ", arr1[i]);

}

printf("\n");

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

{

if (arr1[i] == 0)

{

printf("\nВершина номер %d - изолированная\n", i + 1);

}

}

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

{

if (arr1[i] == 1)

{

printf("Вершина номер %d - концевая\n", i + 1);

}

}

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

{

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

{

if (b[i][j] != 0)

{

o++;

}

}

if (o == n - 1)

{

printf("Вершина номер %d - доминирующая\n", i + 1);

}

o = 0;

}

printf("\n");

free(a);

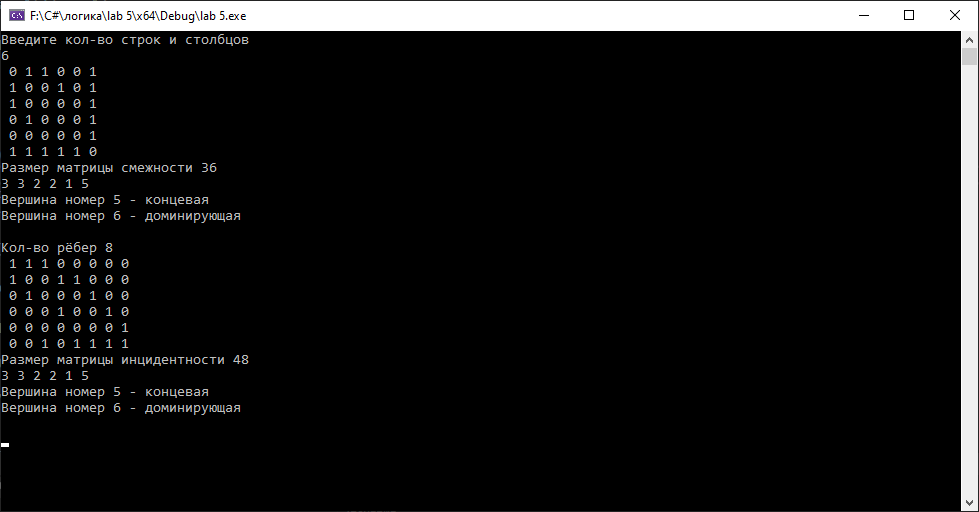
free(b);

getchar();

getchar();

}

**Результаты работы программы:**

****

**Вывод:** в ходе выполнения лабораторной работы мы научились работать с матрицей смежности и матрицей инцидентности. Также научились искать находить изолированные, концевые и доминирующие вершины. Результаты работы программ совпали с результатами трассировки, следовательно программы работают без ошибок.