Пенза 2022

Пензенский государственный университет

Кафедра «Вычислительная техника»

**Выполнил**

**студент группы 21ВВ3:**

Рузляев Д.

**Приняли:**

Митрохин М. А.

Юрова О. В.

**ОТЧЕТ**

по лабораторной работе №5

на тему «Определение характеристик графов»

### Цель работы:

### Изучение и определение характеристик графов, реализация на языке Си.

**Лабораторное задание:**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного 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 смежности графа.
3. Найдите изолированные, концевые и доминирующие вершины.

### Листинг:

**Задание 1**

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#include <locale.h>

#include <windows.h>

using namespace std;

int\*\* G;

int n;

void podschet(int n,int\*\* G) {

int gg = 0;

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

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

if (G[i][j] != 0) {

gg++;

}

}

}

printf("Размер графа = %d\n", gg);

}

void poisk(int n, int\*\* G) {

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

int count = 0; // кол-во ребер с данной вершиной

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

if (G[i][j] == 1) count++;

}

if (count == 0) cout << "Вершина " << i << " изолированная" << endl;

else if (count == 1) cout << "Вершина " << i << " концевая" << endl;

else if (count == n-1) cout << "Вершина " << i << " доминируцющая" << endl;

else cout << "Вершина " << i << " просто вершина" << endl;

}

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("Введите размер матрицы:",n);

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

srand(time(NULL));

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

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");

}

podschet(n, G);

poisk(n, G);

}

**Задание 2**

#include <Windows.h>

#include <time.h>

#include <stdio.h>

void poisk(int n, int\*\* P, int M)

{

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

{

int s = 0; // кол-во ребер с данной вершиной

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

{

if (P[j][i] == 1)

{

s++;

}

}

if (s == 0)

{

printf("Вершина %d изолированная\n", i + 1);

}

else if (s == 1)

{

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

}

else if (s == (n - 1))

{

printf("Вершина %d доминируцющая\n", i + 1);

}

else

{

printf("Вершина %d просто вершина\n", i + 1);

}

}

}

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

{

int M = 0;

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

{

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

{

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

{

M++;

}

}

}

M /= 2;

printf("Количество рёбер (Размер графа) = %d\n", M);

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

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

{

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

}

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

{

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

{

P[i][j] = 0;

}

}

int C = 0;

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

{

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

{

if (G[i][j])

{

P[C][i] = 1;

P[C][j] = 1;

C++;

}

}

}

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

{

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

{

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

}

printf("\n");

}

poisk(n, P, M);

free(P);

}

int main(void)

{

SetConsoleOutputCP(1251);

srand(time(NULL));

int n;

printf("Введите n = ");

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

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

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];

}

}

printf("Матрица смежности\n");

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

{

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

{

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

}

printf("\n");

}

printf("\n");

printf("Матрица инцидентности\n");

in(n, G);

free(G);

return 0;

}

**Результаты работы программы:**

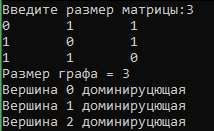


Рисунок 1 — Результаты работы программы 1

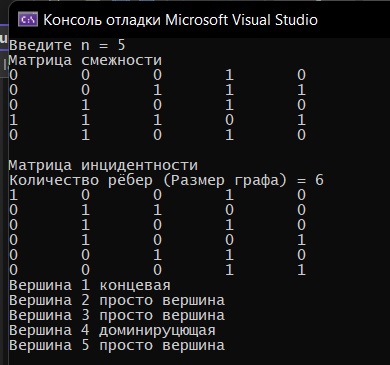


Рисунок 2 — Результаты работы программы 2

### Вывод:

В ходе выполнения лабораторной работы была сгенерирована матрица смежности для неориентированного взвешенного графа G, сделан вывод матрицы на экран, определён размер графа G, используя матрицу смежности графа, найдены изолированные, концевые и доминирующие вершины. Построена для графа G матрица инцидентности, определён размер графа G, используя матрицу смежности графа, найдены изолированные, концевые и доминирующие вершины.