Министерство науки и высшего образования РФ ФГБОУ ВПО

«Пермский государственный национальный исследовательский университет»

Кафедра прикладной математики и информатики

Отчет о выполнении индивидуального задания по теме

**«Определение числа компонент связности в графе»**

***по дисциплине «Дискретная математика»***

|  |  |
| --- | --- |
|  | **Выполнила:**  Студентка 1-го курса механико-математического факультета  Черткова Софья Михайловна,  группа ПМИ-1,2-2021 |

Пермь 2022

**Постановка задачи**

**Задание:** Определение числа компонент связности в графе.

**Входные данные:** В первой строке записано одно число *n* – количество вершин в графе. Далее располагается матрица смежности графа (*n* строк по *n* чисел в каждой).

**Выходные данные:** Одно число – количество компонент связности в графе.

**Язык программирования:** С++.

Функция DFS() рекурсивно выполняет обход в глубину.

1. Функция получает количество вершин в графе, матрицу смежности, логический массив, заранее заполненный false, вершину, от которой будет происходить обход в глубину. Логический массив используется как дополнительная характеристика вершины. В конкретном случае логический массив описывает использована ли вершина (true) или нет (false). Изначально, как и было сказано, все вершины помечены как неиспользованные.
2. Первая вершина отмечается использованной.
3. Для последней помеченной как использованная вершина выбирается смежная вершина, являющаяся первой помеченной как не использованная, и ей присваивается значение «использованная». Если таких вершин нет, то берется предыдущая помеченная вершина.
4. Шаги 2,3 повторяются до тех пор, пока все вершины не будут помечены как использованные.

**В основном блоке происходит:**

1. Чтение файла с исходными данными и выделение динамически памяти компьютера под матрицу и логический массив.
2. Выполнение обхода в глубину для каждой вершины с попутным увеличением переменной k, которая отвечает за число компонент связности.
3. Запись значения из k в выходной файл.
4. Очистка памяти, выделенной под матрицу смежности и логический массив.

**Код программы с описанием данных**

#include <iostream>

#include <fstream>

#include <windows.h>

#include <ctime>

using namespace std;

void DFS(int n, int\*\* matrix\_sm, bool\* matrix\_log, int a) { //обход в глубину

matrix\_log[a] = true; //помечаем вершину, как просмотренную

for (int i = 0; i < n; i++) //проходим по всем остальным вершинам

if (!matrix\_log[i] and matrix\_sm[a][i] != 0) //находим вершины, смежные текущей, которые еще не помечены

DFS(n, matrix\_sm, matrix\_log, i); // выполняем для них обход в глубину

}

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

unsigned int start\_time = clock(); // начало выполнения программы

ifstream f1; ofstream f2;

int i;

f1.open("input.txt");

if (f1) { //проверка читаемости файла

cout << "Файл input.txt прочтен успешно.\n";

int n, k = 0; //k - количество компонент связности, n - количество вершин в графе

f1 >> n;

int\*\* matrix\_sm = new int\* [n]; //динамическое выделение памяти под матрицу смежности

bool\* matrix\_log = new bool[n]; //динамическое выделение памяти под логический массив

for (i = 0; i < n; i++) matrix\_log[i] = false; //заполнение логического массива: вершины помечаем, как не просмотренные

for (i = 0; i < n; i++) { //заполнение матрицы смежности

matrix\_sm[i] = new int[n];

for (int j = 0; j < n; j++) f1 >> matrix\_sm[i][j];

}

for (i = 0; i < n; i++) { //перебираем все вершины графа

if (!matrix\_log[i]) { //вершина не помечена

DFS(n, matrix\_sm, matrix\_log, i); //обход в глубину

k++; //увеличиваем число найденных компонент связности

}

}

f2.open("output.txt");

if (f2) {

f2 << k;

cout << "Количество компонент связности записано в output.txt."; //запись в файл количества компонент связности

}

else cout << "Не удалось записать данные в файл output.txt";

for (i = 0; i < n; i++) delete[] matrix\_sm[i]; //очистка памяти логического массива и матрицы смежности

delete[]matrix\_sm;

delete[]matrix\_log;

}

else cout << "Не удалось прочесть файл input.txt";

f1.close(); //закрытие файлов

f2.close();

unsigned int end\_time = clock(); // конец выполнения программы

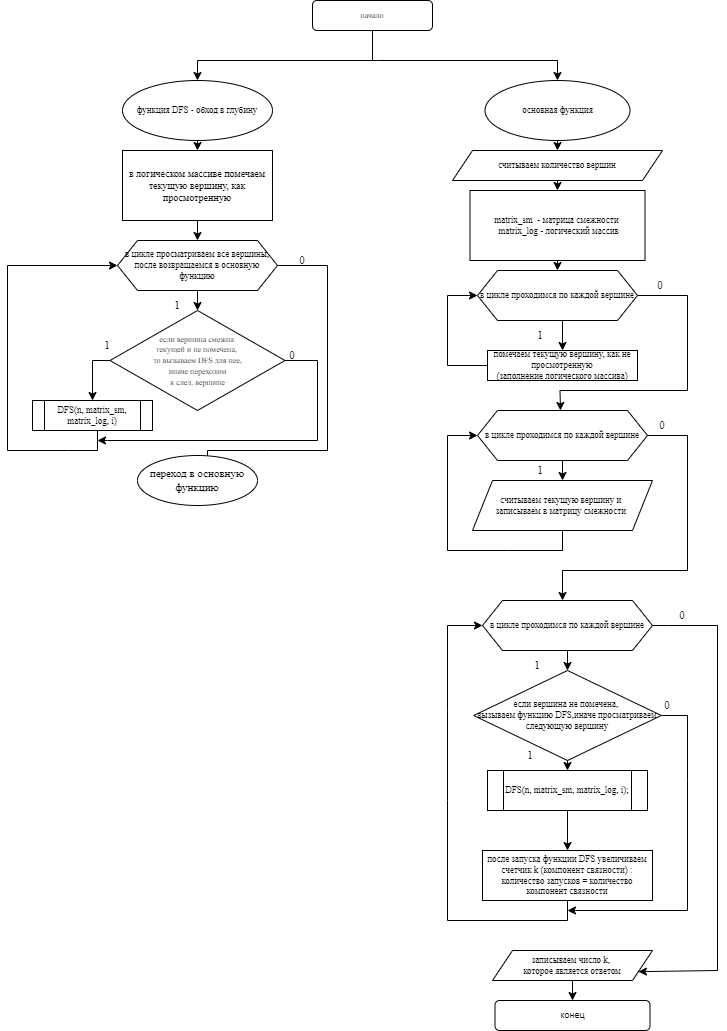
unsigned int time = end\_time - start\_time; // итоговое время выполнения программы

cout << "\nВремя выполнения программы: " << time << " мс.";

}

**Блок – схема**

<https://disk.yandex.ru/i/y9hCSxpaAy3n5w>



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Результаты тестирования**  Результаты данных тестов были проверены с помощью online-сервиса <https://graphonline.ru/> . Изображения графов взяты с этого же источника. Как уже говорилось в алгоритме, мною использовалась функция clock() для определения длительности выполнения программы. Сравнивая скорость выполнения программы на графах различной длины, можно заметить, что она работает достаточно быстро. Время выполнения заметно (по сравнению с другими тестами) увеличивается при проверке на очень больших графах (тест № 7). Стоит заметить, что при разных запусках одного и того же теста время выполнения программы менялось, в таблице представлены данные, приближенные к максимальным из проведенных. Для графов с маленьким количеством вершин время выполнения программы варьировалось в основном от 1 до 8 мс. Ознакомиться с проведенными экспериментами можно в ниже представленной таблице. | | | | | | |
| № теста | Входные данные | Изображение графа | Выходные данные, результат | | Вывод о правильности ответа | Информация о времени выполнения (мс) |
| программа | online-сервис |
| 1 | 4  0 1 0 0  1 0 0 0  0 0 0 1  0 0 1 0 |  | 2 | 2 | верный ответ | 4 |
| 2 | 3  0 1 0  1 0 1  0 1 0 |  | 1 | 1 | верный ответ | 2 |
| 3 | 12  0 0 0 0 0 0 0 0 0 1 0 0  0 0 0 0 0 0 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 0 0 0 0 0 0  0 0 0 0 1 0 0 0 0 0 0 0  0 1 0 0 0 0 0 1 0 0 0 0  0 0 0 0 0 0 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 1 0 0  1 0 0 0 0 0 0 0 1 0 0 0  0 0 0 0 0 0 0 0 0 0 0 1  0 0 0 0 0 0 0 0 0 0 1 0 |  | 6 | 6 | верный ответ | 6 |
| 4 | 40  0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0  0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  | 22 | 22 | верный ответ | 22 |
| 5 | 11  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 |  | 11 | 11 | верный ответ | 10 |
| 6 | 25  0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |  | 10 | 10 | верный ответ | 13 |
| 7 | Полный граф из 84 вершин (3486 ребер) |  | 1 | 1(т.к. все вершины связны между собой) | верный ответ | 41 |