

Министерство образования и науки РФ
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра информационных технологий и автоматизированных систем

Дискретная математика и математическая логика
Лабораторная работа № 5

Тема: «Определение компонент связности»

Выполнил: студент группы
ИВТ-23-1Б
Долганова Диана Евгеньевна
Проверил: ст. пр.
Рустамханова Г. И.

Оглавление

Оглавление	1
Цель работы	1
Задачи работы	2
Этапы выполнения	3
1. Подключение библиотек и размер матрицы	4
2. Преобразование графа в неориентированный	4
3. Алгоритм поиска в ширину	5
4. Главная функция	6
4.1 Настройка локали и открытие файла	6
4.2 Чтение матрицы смежности	6
4.3 Преобразование в неориентированный граф	6
4.4 Поиск компонент связности и построение матрицы достижимости	7
4.5 Вывод результатов	8
Тестирование программы	9
Заключение	10

Цель работы

Разработать программу, которая будет преобразовывать ориентированный граф в неориентированный и отображать матрицу достижимости и количество компонент связности.

Задачи работы

На вход программы должна подаваться матрица 10×10 из файла (возможен ориентированный граф). Требуется слабая связность, т. е. сделать неориентированный граф. Отрисовывать граф не нужно

На выходе:

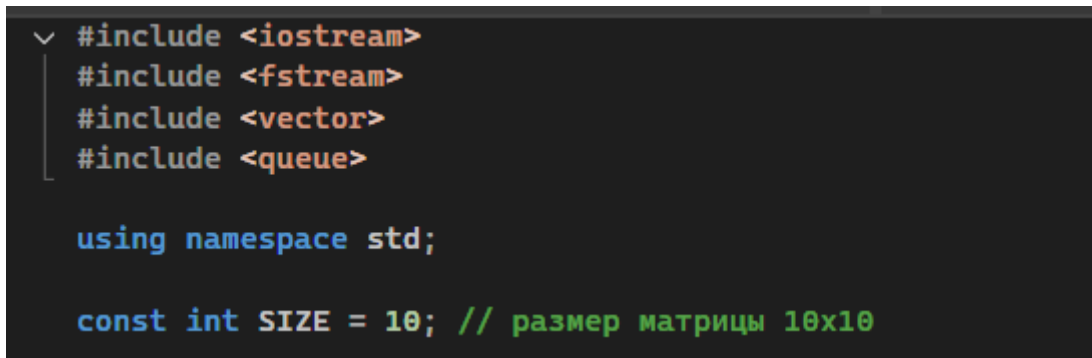
1. матрица достижимости
2. Количество компонент связности и вершины, которые относятся к каждой компоненте связности.

Этапы выполнения

Программа была разработана на языке C++ в программе среде Microsoft Visual Studio 2022.

1. Подключение библиотек и размер матрицы

Подключаем библиотеки для ввода-вывода, для работы с файлами и динамическими массивами, для работы с очередями. Задаем размер матрицы 10x10 (рис. 1).



```
✓ #include <iostream>
#include <fstream>
#include <vector>
#include <queue>

using namespace std;

const int SIZE = 10; // размер матрицы 10x10
```

Рисунок 1 - Библиотеки и размер матрицы

2. Преобразование графа в неориентированный

Функция makeUndirected (рис. 2) Преобразует ориентированный граф (представленный матрицей смежности adjMatrix) в неориентированный (матрица undirectedMatrix).

Работа функции:

- Проходит по всем элементам матрицы adjMatrix.
- Если есть ребро из вершины i в вершину j или из j в i , то в неориентированной матрице добавляются оба ребра ($\text{undirectedMatrix}[i][j] = 1$ и $\text{undirectedMatrix}[j][i] = 1$).

```

// Функция для преобразования ориентированного графа в неориентированный
void makeUndirected(const vector<vector<int>>& adjMatrix, vector<vector<int>>& undirectedMatrix) {
    for (int i = 0; i < SIZE; ++i) {
        for (int j = 0; j < SIZE; ++j) {
            if (adjMatrix[i][j] == 1 || adjMatrix[j][i] == 1) {
                undirectedMatrix[i][j] = 1;
                undirectedMatrix[j][i] = 1;
            }
        }
    }
}

```

Рисунок 2 - Функция makeUndirected

3. Алгоритм поиска в ширину

Функция bfs (рис. 3) реализует алгоритм поиска в ширину (BFS) для обхода графа и нахождения всех вершин, связанных с начальной вершиной start.

Работа функции:

- Использует очередь q для хранения вершин, которые нужно посетить.
- Начинает с вершины start, помечает ее как посещенную и добавляет в текущую компоненту связности.
- Пока очередь не пуста, извлекает вершину из очереди и проверяет всех её соседей.
- Если соседняя вершина не посещена, она добавляется в очередь, помечается как посещенная и включается в текущую компоненту.

```

// Поиск компонент связности с помощью BFS
void bfs(int start, const vector<vector<int>>& graph, vector<bool>& visited, vector<int>& component) {
    queue<int> q;
    q.push(start);
    visited[start] = true;
    component.push_back(start);

    while (!q.empty()) {
        int node = q.front();
        q.pop();

        for (int i = 0; i < SIZE; ++i) {
            if (graph[node][i] == 1 && !visited[i]) {
                visited[i] = true;
                q.push(i);
                component.push_back(i);
            }
        }
    }
}

```

Рисунок 3 - Функция bfs

4. Главная функция

4.1 Настройка локали и открытие файла

Добавляем строчку для корректного отображения русского языка и пытаемся открыть файл. Если файл не открылся, выводим сообщение об ошибке и завершаем программу с кодом 1 (рис. 4).

```
int main() {
    setlocale(LC_ALL, "RU");
    ifstream inFile("g14.txt");
    if (!inFile) {
        cerr << "Не удалось открыть файл." << endl;
        return 1;
    }
}
```

Рисунок 4 - Локаль и открытие файла

4.2 Чтение матрицы смежности

Создается двумерный вектор `adjMatrix` размером `SIZE` x `SIZE`, заполненный нулями. Построчно считываются данные из файла и сохраняются в матрицу `adjMatrix`. После чтения файл закрывается (рис. 5)

```
vector<vector<int>> adjMatrix(SIZE, vector<int>(SIZE, 0));
// Чтение матрицы смежности из файла
for (int i = 0; i < SIZE; ++i) {
    for (int j = 0; j < SIZE; ++j) {
        inFile >> adjMatrix[i][j];
    }
}
inFile.close();
```

Рисунок 5 - Чтение матрицы из файла

4.3 Преобразование в неориентированный граф

Создается матрица `undirectedMatrix`, которая будет представлять неориентированный граф. Функция `makeUndirected` преобразует ориентированный граф (`adjMatrix`) в неориентированный (`undirectedMatrix`) (рис. 6).

```
// Преобразуем ориентированный граф в неориентированный
vector<vector<int>> undirectedMatrix(SIZE, vector<int>(SIZE, 0));
makeUndirected(adjMatrix, undirectedMatrix);
```

Рисунок 6 - Вызов функции makeUndirected

4.4 Поиск компонент связности и построение матрицы достижимости

1. Инициализация:

- reachabilityMatrix: Матрица достижимости, заполненная нулями.
- visited: Массив для отслеживания посещенных вершин.
- components: Вектор для хранения всех компонент связности.

2. Поиск компонент связности:

- Для каждой непосещенной вершины запускается BFS, чтобы найти все вершины в текущей компоненте.
- Вершины компоненты сохраняются в component, а затем добавляются в components.

3. Обновление матрицы достижимости:

- Для каждой компоненты связности все вершины в ней считаются достижимыми друг из друга, поэтому в reachabilityMatrix соответствующие элементы устанавливаются в 1 (рис. 7).


```

// Матрица достижимости
vector<vector<int>> reachabilityMatrix(SIZE, vector<int>(SIZE, 0));

// Поиск компонент связности
vector<bool> visited(SIZE, false);
vector<vector<int>> components;

for (int i = 0; i < SIZE; ++i) {
    if (!visited[i]) {
        vector<int> component;
        bfs(i, undirectedMatrix, visited, component);
        components.push_back(component);

        // Обновление матрицы достижимости
        for (int node : component) {
            for (int other : component) {
                reachabilityMatrix[node][other] = 1;
            }
        }
    }
}

```

Рисунок 7 - Поиск компонент и построение матрицы

4.5 Вывод результатов

Выводим матрицу достижимости, количество компонент связности и список вершин в каждой компоненте (рис. 8).

```

// Вывод матрицы достижимости
cout << "Матрица достижимости:" << endl;
for (const auto& row : reachabilityMatrix) {
    for (int value : row) {
        cout << value << " ";
    }
    cout << endl;
}

// Вывод количества компонент связности и их вершин
cout << "Количество компонент связности: " << components.size() << endl;
for (size_t i = 0; i < components.size(); ++i) {
    cout << "Компонента " << i + 1 << ": ";
    for (int vertex : components[i]) {
        cout << vertex << " ";
    }
    cout << endl;
}

return 0;
}

```

Рисунок 8 - Вывод результатов

Тестирование программы

Для проверки программы на корректность добавим файл с матрицей и запустим программу (рис. 9 - 10).

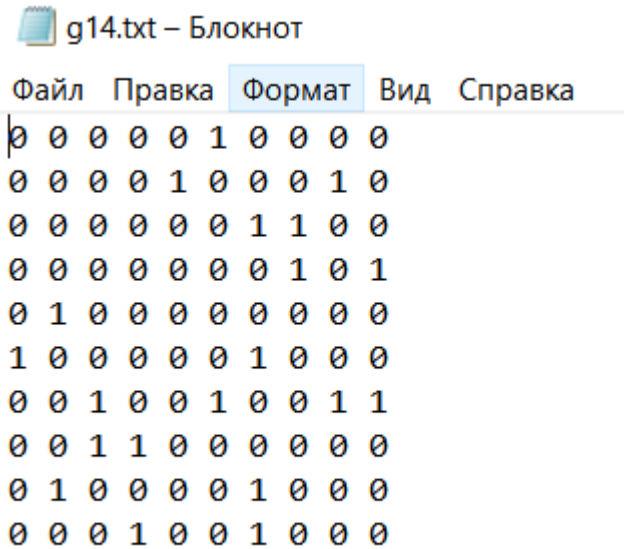


Рисунок 9 - Первая матрица

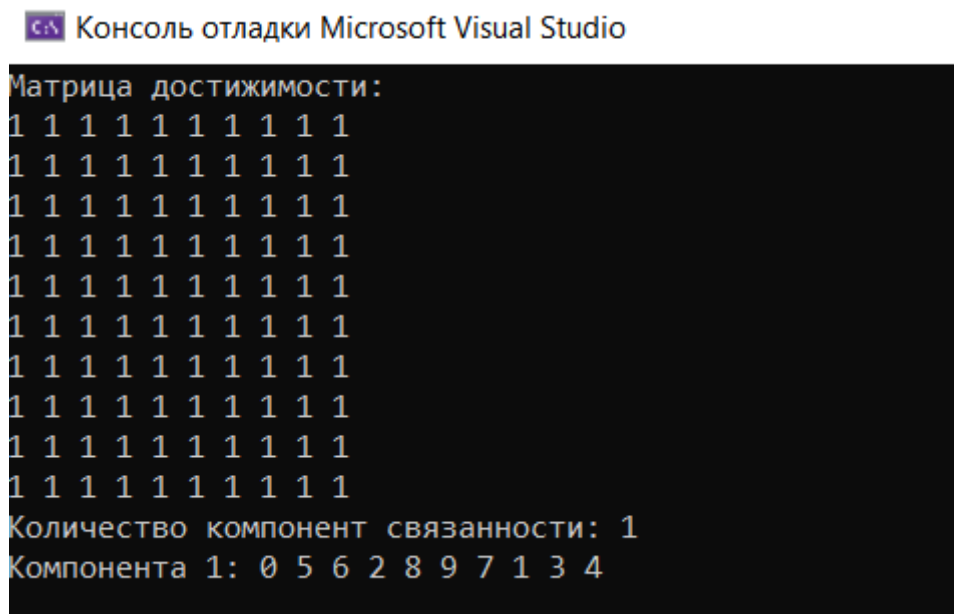


Рисунок 10 - Результат

Заключение

Была разработана программа для преобразования ориентированного графа в неориентированный, а также для вывода компонент связности и матрицы достижимости.

Список используемой литературы

1. Теория графов. Термины и определения в картинках / Хабр URL: <https://habr.com/ru/companies/otus/articles/568026/> (дата обращения: 14.02.2025).
2. Отношение связности, компоненты связности — Викиконспекты URL: https://neerc.ifmo.ru/wiki/index.php?title=Отношение_связности,_компоненты_связности (дата обращения: 14.10.2024).