Отчет по расчетной работе по предмету ПиОИвИС

Сотников Артем
(Гр.321702)

БГУИР

Цель

Ознакомиться с основами теории графов, способами представления графов, базовыми алгоритмами для работы с разными видами графов.

Условие задания

Найти сильные компоненты связности в орграфе.

Базовые понятия

- **Граф** — совокупность двух множеств - множества вершин и множества рёбер.
- **Ориентированный граф** — один из видов графа, структуры, состоящей из вершин и путей между ними.

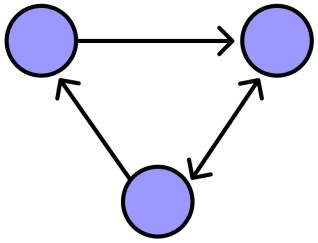

Рис. 1: Ориентированный граф

- **Сильные компоненты связанности (Strongly Connected Components, SCC)** — в ориентированном графе представляют собой группы вершин, в которых любая пара вершин может быть достигнута друг из друга по направленным ребрам. То есть, в SCC каждая вершина связана с каждой другой вершиной внутри компоненты.

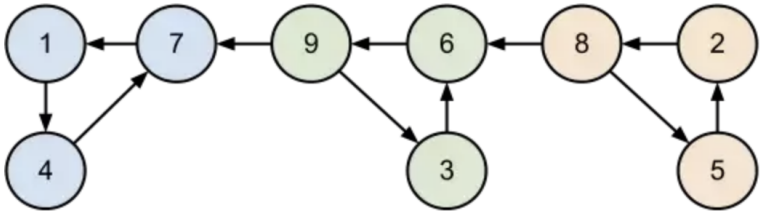

Рис. 2: Сильные компоненты связанности

- **Матрица смежности** — способ представления графа в виде матрицы.

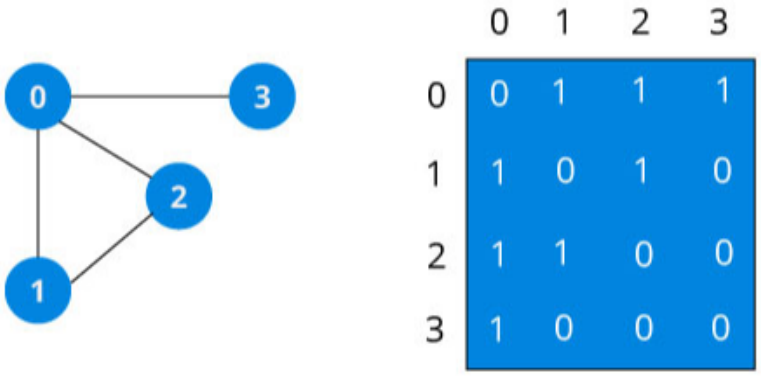

Рис. 3: Матрица смежности

- **Взвешенным** графом называется граф, вершинам и/или ребрам которого присвоены «весы» — обычно некоторые числа.
- **Depth First Search(DFS)** — поиск в глубину, позволяющий найти маршрут от точки A до точки B. Классическая реализация DFS — рекурсивная.
- **Рекурсией** называется процесс, когда какая-то функция вызывает себя же, но с другими аргументами.

Описание работы программы

- Пользователь вводит полный путь и имя файла.
- Файл открывается для чтения, и если открытие не удалось, программа выводит сообщение об ошибке и завершает выполнение.
- Считывается размер матрицы (количество вершин).
- Инициализируются графы g и gt.
- Считывается матрица смежности из файла, и рёбра добавляются в графы.
- Выводится считанная матрица смежности.
- Выполняется первый проход алгоритма DFS, формируется массив order с порядком посещения вершин.
- Выполняется второй проход алгоритма DFS, формируются компоненты сильной связности и выводятся на экран.

Алгоритм Косарайо

Алгоритм Косарайо используется для нахождения компонент сильной связности в ориентированном графе. Он основан на двух проходах по графу с использованием алгоритма обхода в глубину (DFS).

Классический пример использования алгоритма — поиск случайного пути в лабиринте. DFS начинает работу в заданной точке, на каждом шаге проходит по лабиринту до следующего поворота и выбирает направление. Если путь оказывается тупиковым, алгоритм возвращается к предыдущему повороту и пробует новое направление. В результате рано или поздно находится нужный путь.

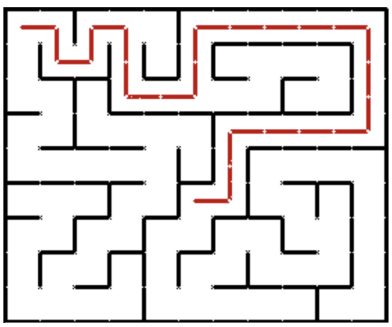

Рис. 4: Пример лабиринта

1. Инициализация:
   - g и gt представляют собой граф и его транспонированный граф соответственно. Они представлены в виде вектора векторов целых чисел.
   - used — вектор для отслеживания посещенных вершин.
   - order — вектор для хранения порядка посещения вершин при первом проходе.
   - component — вектор для хранения компонент сильной связности.
2. Первый проход алгоритма DFS (dfs1):
   - dfs1 рекурсивно обходит граф g и добавляет вершины в order в порядке, обратном времени их завершения поиска.

3. Второй проход алгоритма DFS (dfs1):
    - dfs1 рекурсивно обходит граф gt и добавляет вершины в component.
4. функции main:
    - В этой части кода происходит вызов обеих процедур DFS для поиска порядка завершения вершин (dfs1) и поиска компонент сильной связности (dfs2).

Таким образом, алгоритм Косарайю заключается в двух проходах DFS по графу и его транспонированному графу, в которых фиксируются порядок завершения вершин и находятся компоненты сильной связности.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <fstream>
#include <string>

using namespace std;

vector<vector<int>> g, gt;

vector<char> used;

vector<int> order;

vector<int> component;

void dfs1(int v) {
    used[v] = true;
    for (int u : g[v])
        if (!used[u])
            dfs1(u);
    order.push_back(v);
}

void dfs2(int v) {
    used[v] = true;
    component.push_back(v);
    for (int u : gt[v])
        if (!used[u])
            dfs2(u);
}

int main() {
    cout << "Vvedite polnyy put' i nazvaniye fayla: ";
    string filePath;
    cin >> filePath;

    ifstream inputFile(filePath);

    if (!inputFile.is_open()) {
        cout << "Ne udalos' otkryt' fayl." << endl;
        return 1;
    }

    int n;
    inputFile >> n;

    g.resize(n);
    gt.resize(n);

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            int edge;
            inputFile >> edge;
            if (edge == 1) {
```

```
                g[i].push_back(j);
                gt[j].push_back(i);
            }
        }
    }

    inputFile.close();

    cout << "Schitannaya matritsa:" << endl;
    for (int i = 0; i < n; ++i) {
        cout << "Smezhnyye vershiny dlya " << (i + 1) << ": ";
        for (int j = 0; j < g[i].size(); ++j) {
            cout << g[i][j] + 1 << ' ';
        }
        cout << endl;
    }

    used.assign(n, false);
    for (int i = 0; i < n; ++i)
        if (!used[i])
            dfs1(i);

    used.assign(n, false);
    for (int i = n - 1; i >= 0; --i) {
        int v = order[i];
        if (!used[v]) {
            dfs2(v);
            for (int u : component)
                cout << u + 1 << ' ';
            cout << endl;

            component.clear();
        }
    }

    return 0;
}
```

Пример выполнения

1. Условие

```
6
0 0 0 0 1 0
0 0 1 1 0 0
1 0 0 0 0 0
0 0 0 0 0 1
0 0 1 0 0 0
0 1 0 0 0 0
```

Рис. 5: Матрица смежности

2. Результат

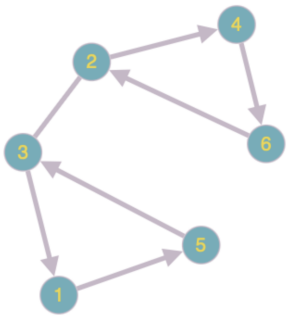

Рис. 6: Граф

```
Введите полный путь и название файла: /Users/artemsotnikov/Desktop/mat3.txt
Считанная матрица:
Смежные вершины для 1: 5
Смежные вершины для 2: 3 4
Смежные вершины для 3: 1
Смежные вершины для 4: 6
Смежные вершины для 5: 3
Смежные вершины для 6: 2
2 6 4
1 3 5
```

Рис. 7: Граф

Вывод

- изучены основы теории графов
- изучены способы представления графов
- изучены базовые алгоритмы для работы с графами