Лабораторная работа № 9 по курсу дискретного анализа: Графы

Выполнила студентка группы 08-308 МАИ *Шевлякова София.*

Условие

Задан неориентированный граф, состоящий из n вершин и m ребер. Вершины пронумерованы целыми числами от 1 до n. Необходимо вывести все компоненты связности данного графа.
Формат ввода: в первой строке заданы n и m. В следующих m строках записаны ребра. Каждая строка содержит пару чисел – номера вершин, соединенных ребром.
Формат вывода: каждую компоненту связности нужно выводить в отдельной строке, в виде списка номеров вершин через пробел. Строки при выводе должны быть отсортированы по минимальному номеру вершины в компоненте, числа в одной строке также должны быть отсортированы.

Метод решения

Понятие компоненты связности вытекает из понятия связности графа. Компонента связности - набор вершин графа, между любой парой которых существует путь. Общее понятие связности распространяется только на неориентированные графы. Для описания ориентированных графов используются понятия сильной и слабой связности.
Для поиска компонент связности используется обычный DFS (или BFS). При запуске обхода из одной вершины, он гарантированно посетит все вершины, до которых возможно добраться, то есть, всю компоненту связности, к которой принадлежит начальная вершина. Для нахождения всех компонент просто попытаемся запустить обход из каждой вершины по очереди, если мы ещё не обошли её компоненту ранее.

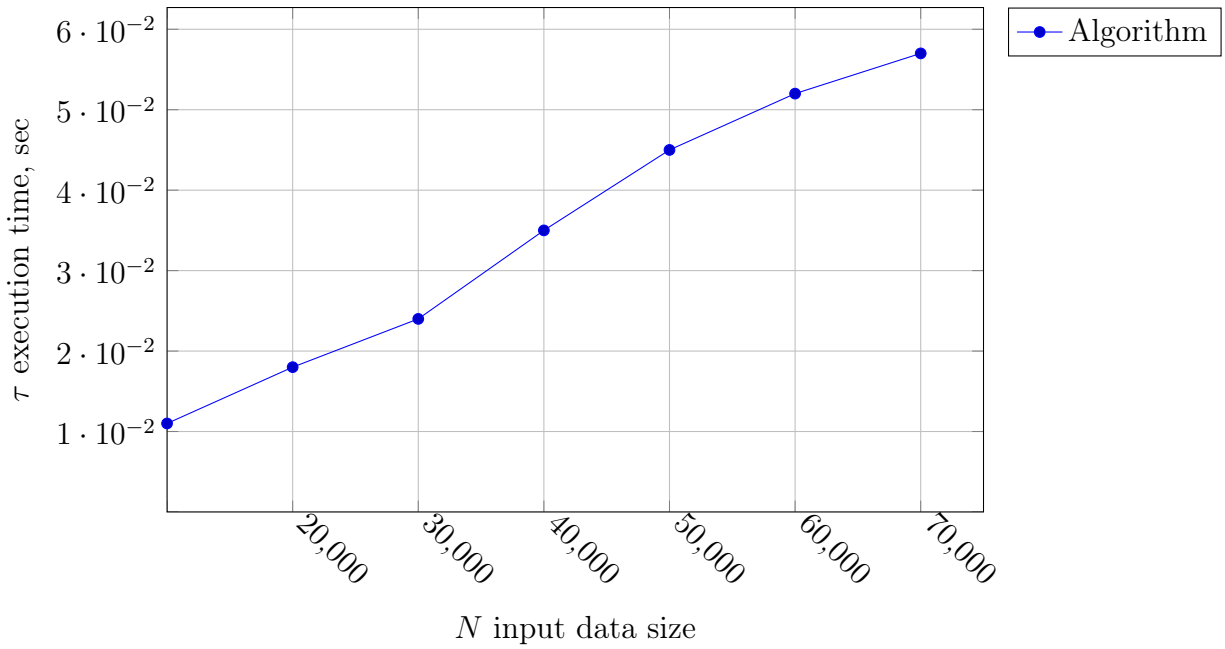
Описание программы

В данной программе содержится один файл main.cpp, в котором реализован алгоритм нахождения компонент связности.
Основные этапы работы программы:

- Сначала считаем входные данные и построим список смежностей, как один из способов хранить граф.
- В самом худшем случае у нас может быть n компонент связности, если ни одна вершина не связана ни с одной другой. Поэтому можем запустить алгоритм bfs n раз. Изначально массив check, который показывает, какие элементы мы уже рассмотрели заполнен -1, после поиска в глубину, все элементы, находящиеся в одной компоненте связности в массиве check будут иметь одинаковое значение, равное номеру компоненты связности.

- После завершения поиска компонент, запишем в result массивы, которые хранят вершин, входящих в один компонент. Далее останется лишь отсортировать сначала сами эти массивы, а потом и массив result.

Тест производительности



По результатам тестирования видно, что сложность выполнения алгоритма является линейной. Так как мы обходим только непересекающиеся компоненты связности, а значит по итогу выполнения алгоритма пройдем весь граф только 1 раз, не заходя в одну и ту же вершину несколько раз. Мы знаем, что сложность обхода графа составляет O(n + m), где n — количество вершин, а m — рёбер графа, значит, сложность алгоритма поиска компонентов связности тоже O(n + m).

Выводы

В результате проведенной лабораторной работы я вспомнила, как пишется DFS и BFS, мной была решена задача по поиску компонент связности. Сложность данного алгоритма не сложно оценить, так как мы выполняем обход графа. Как это видно из графика в предыдущем пункте временная сложность O(n + m), где n — количество вершин, а m — рёбер графа.