Лабораторная работа № 9 по курсу дискретного анализа: графы

Выполнил студент группы 08-307 МАИ *Боев Савелий*.

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.

Метод решения

Программа начинается с чтения двух целых чисел n и m, которые представляют количество объектов и количество ограничений соответственно.

Затем создается вектор **graph** с **n** векторами для представления списка смежности графа. Каждый вектор будет содержать вершины, с которыми связана соответствующая вершина.

Далее происходит ввод ограничений с помощью цикла. Для каждого ограничения, считываются два числа **a** и **b**, уменьшаются на 1 (для индексации с 0) и добавляются в список смежности графа как направленные рёбра**: graph[a].push_back(b) и graph[b].push_back(a)**.

Далее инициализируется вектор **visited** с **n** элементами, начально заполненными **false**, чтобы отслеживать посещенные вершины.

Затем запускается цикл по всем вершинам графа. Если вершина еще не посещена, то вызывается функция **DFS**, которая выполняет обход в глубину, начиная с текущей вершины. Во время обхода, вершины добавляются в вектор **component**, который представляет текущую компоненту связности. После завершения обхода, вершины внутри компоненты сортируются и выводятся на экран.

Функция **DFS** рекурсивно обходит граф, начиная с вершины **v**. Она помечает вершину **v** как посещенную, добавляет ее в текущую компоненту **component**, и затем рекурсивно вызывает **DFS** для всех непосещенных смежных вершин

u.

После завершения всех обходов, программа выводит все компоненты связности в отсортированном порядке, каждую на новой строке.

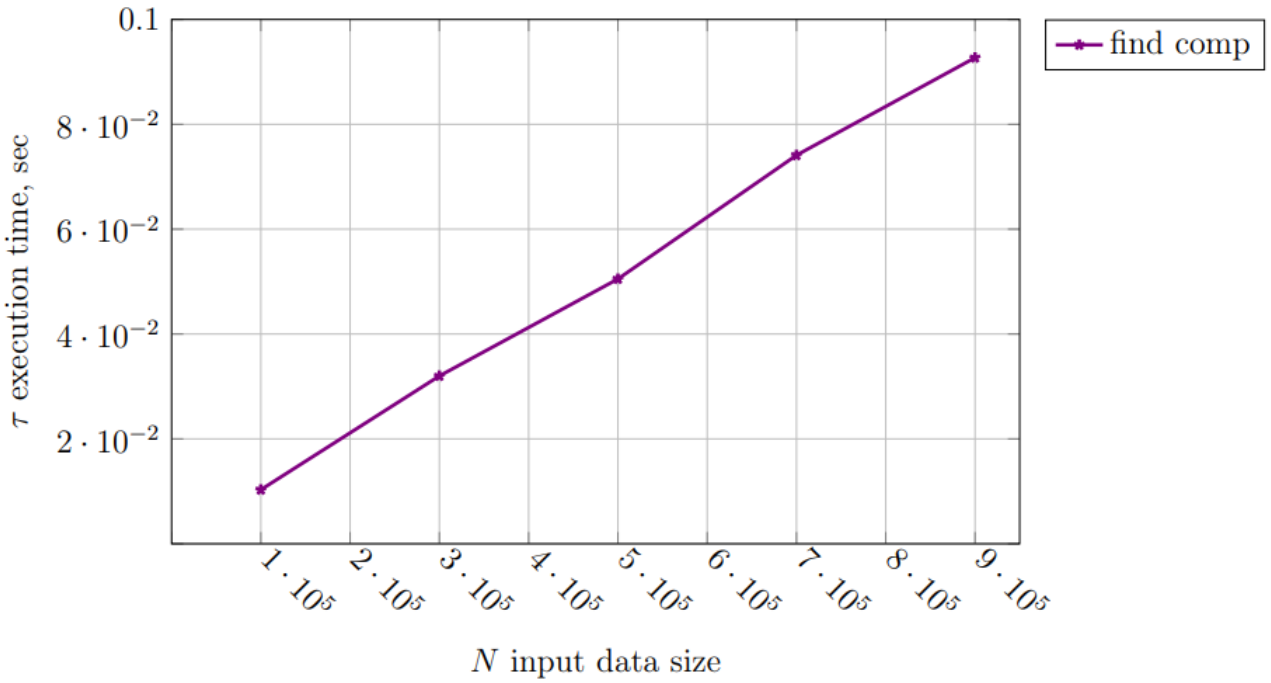
Описание программы

Выбор обхода графа: В программе реализован поиск компонент связности с использованием обхода графа в глубину (DFS). Этот выбор обусловлен простотой реализации и необходимостью обойти все вершины графа.

Оптимизация памяти: Для эффективного использования памяти выбрано представление графа в виде списка смежности. Это позволяет хранить граф в компактной форме и не требует выделения памяти под неиспользуемые элементы.

Сложность: Программа имеет временную сложность, зависящую от числа рёбер и вершин в графе: O(|E| + |V|), где |E| - количество рёбер, а |V| - количество вершин.

Тест производительности



Выводы

В ходе выполнения лабораторной работы была решена задача по поиску компонент связности графа. Делается это крайне тривиально.