Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №9 по курсу «Дискретный анализ»

Студент: А. В. Барсов
Преподаватель: Н. С. Капралов
Группа: М8О-307Б-19

Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №9

Задача: Разработать программу на языке C или C++, реализующую указанный алгоритм согласно заданию:

Вариант №3:

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 входных данных

В первой строке заданы $1 \leq n \leq 10^5$ и $1 \leq m \leq 10^5$. В следующих m строках записаны ребра. Каждая строка содержит пару чисел – номера вершин, соединенных ребром.

Формат результата

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.

1 Описание

Для решения можно воспользоваться как обходом в глубину, так и обходом в ширину.

Фактически, мы будем производить серию обходов: сначала запустим обход из первой вершины, и все вершины, которые он при этом обошёл — образуют первую компоненту связности. Затем найдём первую из оставшихся вершин, которые ещё не были посещены, и запустим обход из неё, найдя тем самым вторую компоненту связности. И так далее, пока все вершины не станут помеченными.

Итоговая асимптотика составит $O(n + m)$: в самом деле, такой алгоритм не будет запускаться от одной и той же вершины дважды, а, значит, каждое ребро будет просмотрено ровно два раза (с одного конца и с другого конца).

2 Консоль

```
[artem@IdeaPad solution]$ make
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm -c main.cpp
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm main.o -o
solution
[artem@IdeaPad solution]$ ./solution
5 4
1 2
2 3
1 3
4 5
1 2 3
4 5
```

3 Листинг программы

Я реализовал обход в глубину для перебора всех компонент связностей.

```
1 void dfs(Graph& gra, std::vector<char>& used, std::set<int>& con_comp, int u) {
2     used[u] = true;
3     con_comp.insert(u);
4     for (int v : gra[u]) {
5         if (used[v]) continue;
6         dfs(gra, used, con_comp, v);
7     }
8 }
```

4 Выводы

После выполнения данной лабораторной работы по дисциплине «Дискретный анализ» я узнал, как можно эффективно находить компоненты связности в неориентированных графах.

Данный алгоритм может пригодиться как в решении олимпиадных задач, так и в для реальных условиях, например в социальных сетях, при поиске общих возможных знакомых, ведь можно составить граф так называемых "рукопожатий и находить компании, предлагать возможных друзей и многое другое.

Список литературы

[1] Дональд Кнут *Искусство программирования.* Перевод и издатели: В. Штонда, Г. Петриковец, А. Орлович.

[2] *Справочник по алгоритмам и структурам данных.*
URL: `https://e-maxx.ru/` .