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. А. Кабанов
Преподаватель: С. А. Михайлова
Группа: М8О-301Б-22

Дата:
Оценка:
Подпись:

Москва, 2024

Лабораторная работа №9

Задача: Поиск компонент связности

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

Формат 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.

1 Описание

Компонентой связности неориентированного графа называется подмножество вершин, достижимых из какой-то заданной вершины. Как следствие неориентированности, все вершины компоненты связности достижимы друг из друга.

Дан неориентированный граф G с n вершинами и m рёбрами. Требуется найти в нём все компоненты связности, то есть разбить вершины графа на несколько групп так, что внутри одной группы можно дойти от одной вершины до любой другой, а между разными группами путей не существует.

Для решения поставленной задачи модифицируем обход в глубину.

Поиск (обход) в глубину (DFS, depth-first search) - рекурсивный алгоритм обхода корневого дерева или графа, начинающийся в корневой вершине (в случае графа - произволной вершине) и рекурсивно обходящий весь граф, посещая вершину ровно один раз.

Модификация заключается в дополнении вектора, отвечающего за хранение всех вершин какой-то из компонент, и добавлении аргумента номера компоненты для добавления вершины.

Для корректного вывода ответа воспользуемся стандартным методом сортировки sort(). Все компоненты связности выводятся в правильном порядке, так как запускается цикл для вызова DFS и берущий вершины по возрастающему порядку.

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  const uint32_t MAXN = 1e5 + 1;
5  array<bool, MAXN> visited;
6  vector<vector<uint32_t>> answer;
7  vector<vector<uint32_t>> g(MAXN, vector<uint32_t>(0));
8
9  void dfs(uint32_t v, int32_t c) {
10   visited[v] = true;
11   for (uint32_t u : g[v]) {
12     if(!visited[u]) {
13       dfs(u, c);
14     }
15   }
16   answer[c].push_back(v);
17 }
18
19 int main() {
20   ios_base::sync_with_stdio(false);
21   cin.tie(nullptr);
22   uint32_t n, m;
23   cin >> n >> m;
24   uint32_t a, b;
```

```
25    for (uint32_t i = 0; i < m; ++i) {
26      cin >> a >> b;
27      g[a].push_back(b);
28      g[b].push_back(a);
29    }
30    int32_t num = -1;
31    for (uint32_t v = 1; v <= n; ++v) {
32      if (!visited[v]) {
33        answer.push_back({});
34        dfs(v, ++num);
35      }
36    }
37    for (auto& v : answer) {
38      sort(v.begin(), v.end());
39      for (const auto& u: v) {
40        cout << u << ' ';
41      }
42      cout << '\n';
43    }
44  }
```

2 Консоль

```
[anton@home lab9-graphs]$ make
g++ -O2 -lm -fno-stack-limit -std=c++20 -x c++ solution.cpp -o executable
[anton@home lab9-graphs]$ ./executable
1 1
1 1
1
[anton@home lab9-graphs]$ ./executable
2 2
1 1
2 2
1
2
[anton@home lab9-graphs]$ ./executable
6 5
2 5
4 6
6 3
3 1
4 3
1 3 4 6
2 5
[anton@home lab9-graphs]$ ./executable
5 4
1 2
2 3
1 3
4 5
1 2 3
4 5
```

3 Тест производительности

Оценим сложность алгоритма: модифицированный DFS не будет запускаться от одной и той же вершины дважды, и каждое ребро будет просмотрено ровно два раза (с одного конца и другого), поэтому асимптотика составит $O(n + m)$.

Реализация поиска компонент связности с выводом ответа по формату может быть асимптотически больше, так как применяет сортировку sort().

Но на практике при вводе максимально разрешенного условием задачи количества вершин и ребер время выполнения не превышает 51 миллисекунд.

При выключении вывода, 10^5 вершин и 10^5 ребер время выполнения равно 4 мс, что немногим больше 2 мс при меньшем количестве вершин.

4 Выводы

Выполнив девятую лабораторную работу по курсу «Дискретный анализ», я познакомился с алгоритмами на графах. Я написал базовый модифицированный алгоритм обхода графа в глубину, поэтому это было не так сложно по сравнению с другими лабораторными работами этого курса.

Но при этом алгоритмы на графах и сами графы - очень интересная тематика для исследования и их реализации, так как с их помощью можно решать множество прикладных задач.

Список литературы

[1] *Теория графов - Викиконспекты*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Теория_графов` (дата обращения: 26.10.2024).

[2] *Поиск в глубину - Алгоритмика*
URL: `https://ru.algorithmica.org/cs/graph-traversals/dfs/` (дата обращения: 26.10.2024).

[3] *Поиск компонент связности - Алгоритмика*
URL: `https://ru.algorithmica.org/cs/graph-traversals/connectivity/` (дата обращения: 26.10.2024).