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. Семин
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б-20

Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №9

Задача: 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 $1 \leqslant n \leqslant 10^5$ и $1 \leqslant m \leqslant 10^5$.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

Для начала определим, что такое компоненты связности: понятие компоненты связности вытекает из понятия связности графа. Попросту говоря, компонента связности - часть графа (подграф), являющаяся связной. Формально, 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, но они выходят за границы материала этой лекции.

Компоненты связности можно искать как с помощью поиска в ширину, так и с помощью поиска в глубину. В моем случае, исходя из удобства, используется алгоритм обхода графа в глубину. Суть в следующем: 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.

2 Исходный код

Файл main.cpp:

```
#include <bits/stdc++.h>
#define endl '\n'


void initGraph(std::vector<std::vector<int>>& graph, int m) {
  while (m--) {
    int i, j;
    std::cin >> i >> j;
    i--; j--;
    graph[i].push_back(j);
    graph[j].push_back(i);
  }
}

void printGraph(const std::vector<std::vector<int>>& graph) {
  int n = graph.size();
  std::cout << n << endl;
  for (int i = 0; i < n; i++) {
    std::cout << i+1 << ": ";
    for (int j = 0; j < graph[i].size(); j++) {
      std::cout << graph[i][j] +1 << ' ';
    }
    std::cout << endl;
  }
}

void dfs(const std::vector<std::vector<int>>& graph, std::vector<bool>& used, std::
    vector<std::vector<int>>& res, int v) {
  used[v] = true;
  res.back().push_back(v);
  for (int i = 0; i < graph[v].size(); i++) {
    int u = graph[v][i];
    if (!used[u]) {
      dfs(graph, used, res, u);
    }
  }
}

void solve(const std::vector<std::vector<int>>& graph, int n) {
  std::vector<std::vector<int>> res;
  std::vector<bool> used(n);
  for (int i = 0; i < n; i++) {
    if (!used[i]) {
      // res.clear();
      res.push_back(std::vector<int>());
      dfs(graph, used, res, i);
```

```
46      }
47    }
48    for (std::vector<int>& comp: res) {
49      std::sort(comp.begin(), comp.end());
50      for (int k = 0; k < comp.size(); k++) {
51        std::cout << comp[k] + 1 << " ";
52      }
53      std::cout << '\n';
54    }
55  }
56
57  int main() {
58    int n, m;
59    std::cin >> n >> m;
60    std::vector<std::vector<int>> graph(n);
61    initGraph(graph, m);
62    solve(graph, n);
63    // printGraph(graph);
64    return 0;
65  }
```

3 Консоль

console input:

```
5 4
1 2
2 3
1 3
4 5
```

console output:

```
1 2 3
4 5
```

4 Тест производительности

Засечем время выполннения программы для случайного графа.
В результате работы *benchmark.cpp* видны следующий результат:

```
a@WIN-THNQL51M105:~/Desktop/DA/lab9$ ./bench <test_graph
Time: 0.002339
```

5 Выводы

Выполнив двеятую лабораторную работу по курсу «Дискретный анализ», мною были различные алгоритмы для работы с графами и применение их для решения задач. Теория графов на данный момент имеет много эффективных инструментов для решения очень широкого круга задач.

Так, в моей работе работе используется обход в глубину, сложность которого O(V + E). Существуют и другие алгоритмы, например, для поиска кратчайшего пути от одной вершины до другой. Например, алгоритм Дейкстры. Он является жадным и имеет сложность O(VlogV).

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))