Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу «Дискретный анализ»

Студент: Ю. Ю. Обыденкова
Преподаватель: А. Н. Ридли
Группа: М8О-308Б-18
Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №8

Задача: Разработать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти.
Дана последовательность длины N из целых чисел 1, 2, 3. Заданы N объектов с ограничиениями на расположение вида «A должен находиться перед B». Необходимо найти такой порядок расположения объектов, что все ограничения будут выполняться.

Формат входных данных: На первой строке два числа, N и M, за которыми следует M строк с ограничениями вида $A < B (1 \leq A, B \leq N)$ определяющими относительную последовательность объектов с номерами A и B.

1 Описание

Задача сводится к построению ориентированного графа и его топологической сортировке. Суть данного алгоритма кроется в одном свойстве обхода в глубину: Если вершина u находится перед вершиной v (в топологически отсортированном графе), то при обходе в глубину(точнее, при серии обходов, запущенных от каждой вершины графа) время выхода вершины u будет больше времени выхода вершины v вне зависимости от порядка запуска обходов. Алгоритм работает для ориентированных ациклических графов, так что перед его непосредственным запуском, нужно проверить граф на ацикличность с помощью обхода в глубину. Сложность алгоритма — $O(V + E)$.

2 Исходный код

В начале программы необходимо убедиться, что построенный граф не содержит циклов, для этого воспользуемся функцией CheckForCycle, представляющей из себя обход в глубину, красящей вершины в серый, черный или белый цвета. Суть в том, что если мы каким-либо образом добрались до серой вершины, то в графе имеется цикл. После проверки запускается топологическая сортировка, сохраняющая результат в вектор, который позже выводится.

```
1  #include <iostream>
2  #include <vector>
3  #include <unordered_map>
4  #include <unordered_set>
5
6  using Graph = std::vector<std::unordered_set<size_t>>;
7
8  enum Color {
9      White, Grey, Black
10 };
11
12 bool CheckForCycle(size_t cur, const Graph& g, std::vector<Color>& colors) {
13     colors[cur] = Grey;
14     for (size_t i : g[cur]) {
15         if ((colors[i] == White && CheckForCycle(i, g, colors)) || colors[i] == Grey) {
16             return true;
17         }
18     }
19     colors[cur] = Black;
20     return false;
21 }
22
23 void TopSort(size_t cur, const Graph& g, std::vector<bool>& used, std::vector<size_t>&
       result) {
24     if (used[cur]) {
25         return;
26     }
27     used[cur] = true;
28     for (size_t i : g[cur]) {
29         TopSort(i, g, used, result);
30     }
31     result.push_back(cur);
32 }
33
34 int main() {
35     size_t n, m;
36     std::cin >> n >> m;
37
38     Graph graph(n);
```

```
39      for (size_t i = 0; i < m; ++i) {
40          long long a,b;
41          std::cin >> a >> b;
42          graph[a - 1].insert(b - 1);
43      }
44
45      bool cycle = false;
46      std::vector<Color> colors(n, White);
47      for (size_t i = 0; i < graph.size(); ++i) {
48          cycle = cycle || CheckForCycle(i, graph, colors);
49      }
50
51      if (cycle) {
52          std::cout << -1 << "\n";
53          return 0;
54      }
55
56      std::vector<bool> used(n,false);
57      std::vector<size_t> result;
58      for (size_t i = 0; i < graph.size(); ++i) {
59          if (!used[i]) {
60              TopSort(i, graph, used, result);
61          }
62      }
63
64      for (std::vector<size_t>::const_reverse_iterator it = result.crbegin(); it !=
            result.crend(); ++it) {
65          std::cout << *it + 1 << " ";
66      }
67      std::cout << "\n";
68
69      return 0;
70  }
```

3 Консоль

```
ulia@WIN-9LNCMF0CCPQ:~$ g++ lab8.cpp -o lab8
ulia@WIN-9LNCMF0CCPQ:~$ ./lab8
3 2
1 2
2 3
1 2 3
ulia@WIN-9LNCMF0CCPQ:~$
```

4 Тест производительности

Замерим время выполенения программы для различных входных данных.

```
ulia@WIN-9LNCMF0CCPQ:~$ ./lab8 <gen_1000 >/dev/null
Time: 1 ms
ulia@WIN-9LNCMF0CCPQ:~$ ./lab8 <gen_10000 >/dev/null
Time: 14 ms
```

Как пример теста был взят граф-бамбук.

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я приобрела опыт работы с графами и узнал много нового о жадных алгоритмах, принципах их построения и анализа. Так, жадный алгоритм — это алгоритм, который на каждом шагу делает локально наилучший выбор в надежде, что итоговое решение будет оптимальным. К примеру, алгоритм Дейкстры нахождения кратчайшего пути в графе вполне себе жадный, потому что мы на каждом шагу ищем вершину с наименьшим весом, в которой мы еще не бывали, после чего обновляем значения других вершин. При этом можно доказать, что кратчайшие пути, найденные в вершинах, являются оптимальными. Известно, что если структура задачи задается матроидом, тогда применение жадного алгоритма выдаст глобальный оптимум.Если глобальная оптимальность алгоритма имеет место практически всегда, его обычно предпочитают другим методам оптимизации, таким как динамическое программирование. К слову, алгоритм Флойда, который тоже ищет кратчайшие пути в графе (правда, между всеми вершинами), не является примером жадного алгоритма. Флойд демонстрирует метод динамического программирования.

Так, в решаемой мной задаче , можно заметить, что она сводится к построению ориентированного графа и его топологической сортировке. Суть данного алгоритма кроется в одном свойстве обхода в глубину: Если вершина u находится перед вершиной v (в топологически отсортированном графе), то при обходе в глубину(точнее, при серии обходов, запущенных от каждой вершины графа) время выхода вершины u будет больше времени выхода вершины v вне зависимости от порядка запуска обходов. Алгоритм работает для ориентированных ациклических графов, так что перед его непосредственным запуском, нужно проверить граф на ацикличность с помощью обхода в глубину.

Сложность алгоритма - O(V + E).

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.))