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: М. А. Бронников
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б
Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №8

Задача: Разрабтать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти.

Реализовать программу на языке C или C++, соответствующую построенному алгоритму. Формат входных и выходных данных описан в варианте задания.

Вариант задания: Топологическая сортировка

Входные данные: 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.

Выходные данные: -1 если расположить объекты в соответствии с требованиями невозможно, последовательность номеров объектов в противном случае

1 Описание

Как сказано в [1]: «Топологическая сортировка (Topological sort) — один из основных алгоритмов на графах, который применяется для решения множества более сложных задач.

Задача топологической сортировки графа состоит в следующем: указать такой линейный порядок на его вершинах, чтобы любое ребро вело от вершины с меньшим номером к вершине с большим номером. Очевидно, что если в графе есть циклы, то такого порядка не существует. ».

«Поиск в глубину или обход в глубину (англ. Depth-first search, сокращенно DFS) — один из методов обхода графа. Алгоритм поиска описывается следующим образом: для каждой не пройденной вершины необходимо найти все не пройденные смежные вершины и повторить поиск для них.

Запускаем обход в глубину, и когда вершина обработана, заносим ее в стек. По окончании обхода в глубину вершины достаются из стека. Новые номера присваиваются в порядке вытаскивания из стека.

Цвет: во время обхода в глубину используется 3 цвета. Изначально все вершины белые. Когда вершина обнаружена, красим ее в серый цвет. Когда просмотрен список всех смежных с ней вершин, красим ее в черный цвет.
»[1]

2 Исходный код

```
#include <iostream>
#include <vector>

using namespace std;

bool cyclic(int v, vector<vector<int>>& graph, vector <int> &color ) {
    color[v] = 1;
    for(size_t i = 0; i < graph[v].size(); ++i) {
        int to = graph[v][i];
        if(color[to] == 0){
            if(cyclic (to, graph, color)){
                return true;
            }
        }
        else if(color[to] == 1){
            return true;
        }
    }
    color[v] = 2;
    return false;
}


void dfs(int v, vector <vector<int>> graph, vector<bool> &used, vector<int> &answer) {
    used[v] = true;
    for(int i=0; i < graph[v].size(); i++) {
        int to = graph[v][i];
        if(!used[to]){
            dfs(to, graph, used, answer);
        }
    }
    answer.push_back(v+1);
}

void topological_sort(int n, vector <vector<int>> graph, vector<bool> &used, vector<
    int> &answer) {
    for (int i = 0; i < n; i++)
        used[i] = false;
    for (int i = 0; i < n; i++)
        if (!used[i])
            dfs(i, graph, used, answer);
    reverse(answer.begin(), answer.end());
}

int main() {
    int N, M;
    cin >> N >> M;
```

```
47      int A, B;
48      vector<vector<int>> graph(N);
49      vector<bool> used (N);
50      vector<int> answer;
51      vector<int> color (N,0);
52      for (int i = 0; i < M; i++){
53              cin >> A >> B;
54              graph[A-1].push_back(B-1);
55          }
56      for (int i = 0; i < N; i++){
57          if (cyclic(i, graph, color)){
58              cout << "-1" << endl;
59              return 0;
60          }
61      }
62      topological_sort(N, graph, used, answer);
63      for (int i = 0; i < answer.size(); i++)
64      {
65          cout << answer[i] << ' ';
66      }
67      cout << endl;
68      return 0;
69  }
```

cyclic - процедура, возвращающая проверяющая граф на наличие циклов

topological_sort - основная процедура сортировки, вызывающая в себе функцию обхода в глубину *dfs*.

3 Консоль

```
(base) max@max-X550CC:~/DA/lab8$ g++ -std=c++17 -o run -pedantic  main.cpp
(base) max@max-X550CC:~/DA/lab8$ ./run
3 2
1 2
2 3
1 2 3
(base) max@max-X550CC:~/DA/lab8$ rm run
(base) max@max-X550CC:~/DA/lab8$ exit
```

4 Выводы

Благодаря восьмой лабораторной работе по курсу «Дискретный анализ», я, наконец, узнал что такое жадные алгоритмы на графах и где они применяются.

Топологическая сортировка применяется в самых разных ситуациях, например при распараллеливании алгоритмов, когда по некоторому описанию алгоритма нужно составить граф зависимостей его операций и, отсортировав его топологически, определить, какие из операций являются независимыми и могут выполняться параллельно (одновременно).

Я рад, что теперь в моем арсенале появился такой полезный метод, как жадные алгоритмы, ведь без знаний о них не может обойтись любой хороший программист. Я уверен, что они мне не раз пригодятся в будущем, однако задачи, которые мне придется с ними решать, будут намного труднее.

Список литературы

[1] *Топологическая сортировка - Хабр*
URL: `https://habr.com/ru/post/100953/` (дата обращения: 16.09.2019).

[2] *e-olymp 1948. Топологическая сортировка*
URL: `https://cpp.mazurok.com/tag/топологическая-сортировка/` (дата обращения: 16.09.2019).