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. В. Леухин
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б-20

Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №8

Задача:

Заданы N объектов с ограничиениями на расположение вида «A должен находиться перед B». Необходимо найти такой порядок расположения объектов, что все ограничения будут выполняться.

1 Описание

Для решения задачи обратимся к топологической сортировке, являющейся жадным алгоритмом. Топологическая сортировка используется на графах, и в результате её применения для всех вершин выполняется условие: все пути из вершины с меньшим номером ведут только в вершины с большим номером. Очевидно, что это возможно только в том случае, если граф ациклический — иначе такая нумерация вершин просто не существует.

Для реализации топологической сортировки используем алгоритм обход графа в глубину. В самом начале выберем любую вершину в качестве стартовой, откуда запустим DFS. При обходе будем красить вершины в 3 условных цвета. Если вершина имеет цвет 1, значит она ещё не была посещена. Если цвет 2 – была посещена не в текущей итерации. Наконец, если вершина имеет цвет 3, значит она была посещена в текущей итерации DFS и мы обнаружили в графе цикл — сортировка графа невозможна.

При выходе из очередной итерации DFS будем помещать вершину в вектор. Таким образом, полсе окончания обхода в глубину, в векторе будут лежать отсортированные по убыванию вершины графа. Реверсируя этот вектор, получаем ответ на поставленную задачу

```
1
2  #include <iostream>
3  #include <queue>
4  #include <vector>
5  #include <algorithm>
6
7  using namespace std;
8
9  using Matrix = vector<vector<bool>>;
10
11 bool dfs(int v, const Matrix& matrix, vector<int>& color, vector<int>& result) {
12
13     color[v] = 1;
14     for (int to = 0; to < matrix[v].size(); to++) {
15         if (matrix[v][to]) {
16             if (color[to] == 1) {
17                 return false;
18             }
19             if (color[to] == 0) {
20                 if (!dfs(to, matrix, color, result)) {
21                     return false;
22                 }
23                 color[to] = 2;
24                 result.push_back(to);
25             }
26         }
27     }
```

```
28      return true;
29
30  }
31
32  vector<int> TopologicalSort(const Matrix& matrix) {
33
34      vector<int> color(matrix.size(), 0);
35      vector<int> result;
36
37      for (int i = 0; i < matrix.size(); i++) {
38          if (!color[i]) {
39              bool isAcyclic = dfs(i, matrix, color, result);
40              if (!isAcyclic) {
41                  return {};
42              }
43              color[i] = 2;
44              result.push_back(i);
45          }
46      }
47
48      reverse(result.begin(), result.end());
49      return result;
50
51  }
52
53  int main() {
54
55      int n, m;
56      cin >> n >> m;
57      Matrix matrix(n, vector<bool>(n, false));
58
59      int a, b;
60      for (int i = 0; i < m; i++) {
61          cin >> a >> b;
62          matrix[a - 1][b - 1] = true;
63      }
64
65      vector<int> result = TopologicalSort(matrix);
66
67      if (result.empty()) {
68          cout << -1;
69      } else {
70          for (int e : result) {
71              cout << e + 1 << " ";
72          }
73      }
74
75  }
```

2 Консоль

```
C:/Users/leyhi/CLionProjects/contest/cmake-build-debug/lab8.exe
3 2
1 2
2 3
>1 2 3
```

3 Сложность алгоритма

Так как для реализации топологической сортировки используется обход в глубину, а помимо DFS никакие действия не приводят к значительному повышению сложности, их алгоритмические сложности совпадают и равны $O(n + m)$, где n — количество вершин в графе, а m – количество рёбер.

4 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я познакомился с таким понятием, как "жадные"алгоритмы и научился реализовывать топологическую сортировку. "Жадные"алгоритмы хорошо подходят для решения задач, которые можно разбить на подзадачи. Но в отличие от динамического программирования, где результат решения всех подзадач влияет на результат решения задачи, здесь на каждом локальном этапе выбирается оптимальное решение, после чего предполагается, что и решение основной задачи будет наиболее оптимальным.

Если же говорить о реализации топологической сортировки, то здесь я не испытал никаких проблем, так как она почти полностью совпадает с алгоритмом обхода в глубину, с которым я познакомился ещё на 1 курсе.

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.))

[2] *Жадные алгоритмы — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Жадные-алгоритмы` (дата обращения: 04.11.2022).

[3] *Топологическая сортировка — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Топологическая-сортировка` (дата обращения: 04.11.2022).