

引言

在计算机科学中,拓扑排序是一种对有向无环图 (DAG) 中的节点进行排序的算法。拓扑排序可以帮助我们确定一组任务的执行顺序,使得所有的依赖关系都得到满足。本文将详细介绍拓扑排序的概念、算法原理以及使用C++代码实现的步骤。

一、拓扑排序的定义

拓扑排序是对有向图中的节点进行线性排序的过程,使得图中的每条有向边的起点在排序结果中都排在终点之前。换句话说,如果图中存在一条从节点A到节点B的有向边,那么在拓扑排序中,节点A应该出现在节点B之前。

二、拓扑排序的算法原理

1.构建入度表

遍历图中的所有节点,统计每个节点的入度(即指向该节点的边的数量),并构建入度表。

2. 初始化 队列

将入度为0的节点添加到一个队列中,作为拓扑排序的起始点。

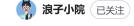
3.进行拓扑排序

- 从队列中取出一个节点,并将其输出到排序结果中。
- 遍历该节点的所有邻居节点,将其入度减1。
- 如果某个邻居节点的入度变为0,则将其加入队列中。

4.重复步骤3,直到队列为空

• 如果在排序过程中,存在入度不为0的节点,说明图中存在环,无法进行拓扑排序。

三、C++实现拓扑排序算





```
会员中心 🞁 消息
#include <queue>
    3
    4
                                                                                                        目录
    5
      using namespace std;
    6
                                                                                                        日录
    7
      vector<int> topologicalSort(vector<vector<int>& graph, vector<int>& inDegree) {
    8
          int numNodes = graph.size();
                                                                                                        引言
   9
          vector<int> result;
                                                                                                        一、拓扑排序的定义
   10
          queue<int> q;
   11
                                                                                                        二、拓扑排序的算法原理
          // 将入度为@的节点加入队列
   12
                                                                                                           1.构建入度表
   13
          for (int i = 0; i < numNodes; i++) {</pre>
              if (inDegree[i] == 0) {
   14
                                                                                                           2.初始化队列
   15
                 q.push(i);
   16
                                                                                                           3.进行拓扑排序
   17
          }
                                                                                                           4.重复步骤3,直到队列为
   18
   19
          while (!q.empty()) {
                                                                                                        三、C++实现拓扑排序算法的
   20
              int current = q.front();
   21
              q.pop();
                                                                                                        四、拓扑排序的应用举例
              result.push_back(current);
   22
                                                                                                           1.编译顺序
   23
              // 遍历当前节点的邻居节点
   24
                                                                                                           2.任务调度
   25
              for (int neighbor : graph[current]) {
   26
                 inDegree[neighbor]--;
                                                                                                           3.课程安排
   27
                 if (inDegree[neighbor] == 0) {
                                                                                                        结论
   28
                     q.push(neighbor);
   29
                                                                                                        参考资料
   30
              }
   31
          }
   32
          // 如果排序结果不包含所有节点,说明图中存在环
   33
          if (result.size() != numNodes) {
   34
              cout << "Graph contains a cycle." << endl;</pre>
   35
              result.clear();
   36
   37
          }
   38
   39
          return result;
   40
      }
   41
   42
      int main() {
   43
          int numNodes = 6;
   44
          vector<vector<int>> graph(numNodes);
   45
          vector<int> inDegree(numNodes, 0);
   46
          // 构建有向图的邻接表和入度表
   47
          graph[0].push back(1);
   48
          graph[0].push_back(2);
   49
          graph[1].push_back(3);
   50
          graph[1].push_back(4);
   51
   52
          graph[2].push_back(5);
   53
   54
          inDegree[1]++;
   55
          inDegree[2]++;
   56
          inDegree[3]++;
   57
          inDegree[4]++;
   58
          inDegree[5]++;
   59
          // 进行拓扑排序
   60
          vector<int> sortedNodes = topologicalSort(graph, inDegree);
   61
   62
   63
          if (!sortedNodes.empty()) {
   64
              cout << "Topological order: ";</pre>
   65
              for (int node : sortedNodes) {
                 cout << node << " ";</pre>
   66
   67
   68
              cout << endl;</pre>
   69
   70
                                     浪子小院(已关注)
                                                                               0
                                                                                                                    专栏目录
   71
          return 0;
```