


原创浪子小院 于 2023-08-15 21:43:03 发布 阅读量686 收藏4 点赞数1 版权

分类专栏: 基础精讲 文章标签: 数据结构 算法 c++ 开发语言 java python

 基础精讲 专栏收录该内容 7 订阅 70 篇文章 订阅专栏

目录

引言

- 一、拓扑排序的定义
- 二、拓扑排序的算法原理
 - 1.构建入度表
 - 2.初始化队列
 - 3.进行拓扑排序
 - 4.重复步骤3，直到队列为空
- 三、C++实现拓扑排序算法的代码
- 四、拓扑排序的应用举例
 - 1.编译顺序
 - 2.任务调度
 - 3.课程安排
- 结论
- 参考资料

目录

目录

引言

- 一、拓扑排序的定义
- 二、拓扑排序的算法原理
 - 1.构建入度表
 - 2.初始化队列
 - 3.进行拓扑排序
 - 4.重复步骤3，直到队列为空
- 三、C++实现拓扑排序算法的代码
- 四、拓扑排序的应用举例
 - 1.编译顺序
 - 2.任务调度
 - 3.课程安排
- 结论
- 参考资料

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 拓扑排序算法



浪子小院

已关注

1

4

0

专栏目录

```
3 #include <queue>
4
5 using namespace std;
6
7 vector<int> topologicalSort(vector<vector<int>>& graph, vector<int>& inDegree) {
8     int numNodes = graph.size();
9     vector<int> result;
10    queue<int> q;
11
12    // 将入度为0的节点加入队列
13    for (int i = 0; i < numNodes; i++) {
14        if (inDegree[i] == 0) {
15            q.push(i);
16        }
17    }
18
19    while (!q.empty()) {
20        int current = q.front();
21        q.pop();
22        result.push_back(current);
23
24        // 遍历当前节点的邻居节点
25        for (int neighbor : graph[current]) {
26            inDegree[neighbor]--;
27            if (inDegree[neighbor] == 0) {
28                q.push(neighbor);
29            }
30        }
31    }
32
33    // 如果排序结果不包含所有节点, 说明图中存在环
34    if (result.size() != numNodes) {
35        cout << "Graph contains a cycle." << endl;
36        result.clear();
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
47     // 构建有向图的邻接表和入度表
48     graph[0].push_back(1);
49     graph[0].push_back(2);
50     graph[1].push_back(3);
51     graph[1].push_back(4);
52     graph[2].push_back(5);
53
54     inDegree[1]++;
55     inDegree[2]++;
56     inDegree[3]++;
57     inDegree[4]++;
58     inDegree[5]++;
59
60     // 进行拓扑排序
61     vector<int> sortedNodes = topologicalSort(graph, inDegree);
62
63     if (!sortedNodes.empty()) {
64         cout << "Topological order: ";
65         for (int node : sortedNodes) {
66             cout << node << " ";
67         }
68         cout << endl;
69     }
70
71     return 0;
```

目录

目录

引言

一、拓扑排序的定义

二、拓扑排序的算法原理

1.构建入度表

2.初始化队列

3.进行拓扑排序

4.重复步骤3, 直到队列为空

三、C++实现拓扑排序算法

四、拓扑排序的应用举例

1.编译顺序

2.任务调度

3.课程安排

结论

参考资料