# 图【拓扑序列】【栈-深度优先】【队列-广度优先】

## 题目描述：



## 题目分析：【栈-深度优先】

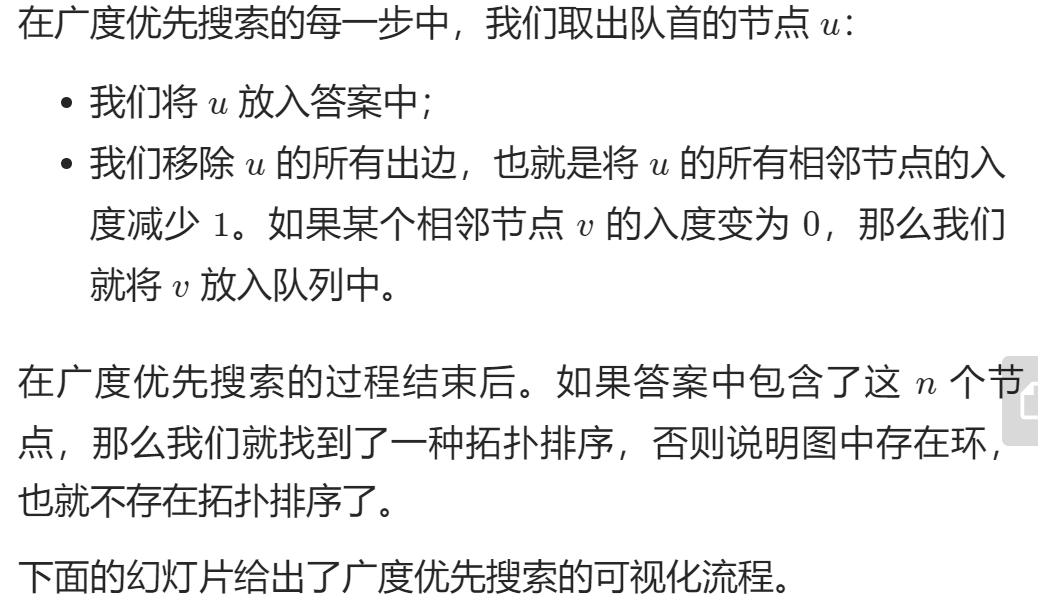
1. **u可以入栈，当且仅当u所有相邻结点都已入栈；**
2. **每个结点有三种状态：待搜索、搜索中、已完成；**
3. **利用邻接表法存储图**
4. <https://leetcode-cn.com/problems/course-schedule/solution/ke-cheng-biao-by-leetcode-solution/>

class Solution {  
private:  
 vector<vector<int>> edges;  
 vector<int> visited;  
 bool valid = true;  
  
public:  
 void dfs(int u) {  
 visited[u] = 1;  
 for (int v: edges[u]) {  
 if (visited[v] == 0) {  
 dfs(v);  
 if (!valid) {  
 return;  
 }  
 }  
 else if (visited[v] == 1) {  
 valid = false;  
 return;  
 }  
 }  
 visited[u] = 2;  
 }  
  
 bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {  
 //利用邻接表法存储表  
 edges.resize(numCourses);  
 visited.resize(numCourses);  
 for (const auto& info: prerequisites) {  
 edges[info[1]].push\_back(info[0]);  
 }  
 //visited[i]为0表示待搜索、1表示搜索中、2表示已完成  
 //通过变化visited[i]的值来表示其状态（此处由于没有要求求拓扑序列，所有灭有必要让其真入栈）  
   
 for (int i = 0; i < numCourses && valid; ++i) {  
 if (!visited[i]) {  
 dfs(i);  
 }  
 }  
 return valid;  
 }  
};  
  
作者：LeetCode-Solution  
链接：https://leetcode-cn.com/problems/course-schedule/solution/ke-cheng-biao-by-leetcode-solution/  
来源：力扣（LeetCode）  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

## 题目分析：【队列-广度优先】

1.方法一的深度优先搜索是一种「逆向思维」：最先被放入栈中的节点是在拓扑排序中最后面的节点。我们也可以使用正向思维，顺序地生成拓扑排序，这种方法也更加直观。

2.我们使用一个队列来进行广度优先搜索。初始时，所有入度为 0 的节点都被放入队列中，它们就是可以作为拓扑排序最前面的节点，并且它们之间的相对顺序是无关紧要的。

3.

class Solution {  
private:  
 vector<vector<int>> edges;  
 vector<int> indeg;  
  
public:  
 bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {  
 //构建图的邻接表的结构  
 //此处还需要求的是各个点的入度  
 edges.resize(numCourses);  
 indeg.resize(numCourses);  
 for (const auto& info: prerequisites) {  
 edges[info[1]].push\_back(info[0]);  
 ++indeg[info[0]];  
 }  
 //所有入度为0的点入队  
 queue<int> q;  
 for (int i = 0; i < numCourses; ++i) {  
 if (indeg[i] == 0) {  
 q.push(i);  
 }  
 }  
  
 int visited = 0;  
 while (!q.empty()) {  
 ++visited;  
 int u = q.front();  
 q.pop();  
 for (int v: edges[u]) {  
 --indeg[v];  
 if (indeg[v] == 0) {  
 q.push(v);  
 }  
 }  
 }  
  
 return visited == numCourses;  
 }  
};  
  
作者：LeetCode-Solution  
链接：https://leetcode-cn.com/problems/course-schedule/solution/ke-cheng-biao-by-leetcode-solution/  
来源：力扣（LeetCode）  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。