207. 课程表

> 题目：Course Schedule
> 语言: python3
> 英文版链接：https://leetcode.com/problems/course-schedule/description/
> 中文版链接：https://leetcode-cn.com/problems/course-schedule/

题目分析

这道题是一道拓扑排序的题目，但是我们并不用完成拓扑排序，只需要判断图中是否存在环即可。

拓扑排序的基础如果不了解，可以参考：
https://blog.csdn.net/lisonglisonglisong/article/details/45543451

拓扑排序用BFS和DFS两种都可以实现，BFS的实现复杂度较高，因此此处我们选用DFS来实现。

对于使用DFS实现，我们每次找到一个新的点，判断从这个点出发是否有环。

具体做法是使用一个visited数组，当visited[i]值为0，说明还没判断这个点；当visited[i]值为1，说明当前的循环正在判断这个点；当visited[i]值为2，说明已经判断过这个点，含义是从这个点往后的所有路径都没有环，认为这个点是安全的。

那么，我们对每个点出发都做这个判断，检查这个点出发的所有路径上是否有环，如果判断过程中找到了当前的正在判断的路径，说明有环；找到了已经判断正常的点，说明往后都不可能存在环，所以认为当前的节点也是安全的。如果当前点是未知状态，那么先把当前点标记成正在访问状态，然后找后续的节点，直到找到安全的节点为止。最后如果到达了无路可走的状态，说明当前节点是安全的。

答案

```python
class Solution:
    def canFinish(self, numCourses, prerequisites) -> bool:
        # 首先对传递过来的连接关系进行处理
        graphic = [[] for _ in range(numCourses)]
        for pre in prerequisites:
            graphic[pre[0]].append(pre[1])

        # 申请一个标记数组，其中为0表示未访问，1表示正在访问，2表示访问完成
        visited = [0] * numCourses
        for i in range(numCourses):
            # 判断是否有环
            if self.exist_cycle(visited, graphic, i):
                return False
        return True

    def exist_cycle(self, visited, graphic, cur_node):
        # 如果当前访问到的节点状态是1，也就是表示正在被访问，有环
        if visited[cur_node] == 1:
            return True
        # 当前访问到的节点状态是2，表示访问完成
        if visited[cur_node] == 2:
            return False

        # 否则表示是一个未访问过的节点，我们先把状态置为1
        visited[cur_node] = 1

        for next_node in graphic[cur_node]:
            # 深度遍历其指向子节点，并且从每个子节点寻找子节点后面是否存在环
            if self.exist_cycle(visited, graphic, next_node):
                return True
        # 节点访问完把状态置为访问结束
        visited[cur_node] = 2
        return False
```