1.课程表
2.课程表2
3.字符串拓扑排序

207. 课程表

难度 中等 898 收藏 分享 切换为英文 接收动态 反

你这个学期必须选修 `numCourses` 门课程，记为 `0` 到 `numCourses - 1` 。

在选修某些课程之前需要一些先修课程。 先修课程按数组 `prerequisites` 给出，其中 `prerequisites[i] = [a_i, b_i]` ，表示如果要学习课程 `a_i` 则 **必须** 先学习课程 `b_`

- 例如，先修课程对 `[0, 1]` 表示：想要学习课程 `0` ，你需要先完成课程 `1` 。

请你判断是否可能完成所有课程的学习？如果可以，返回 `true` ；否则，返回 `false` 。

示例 1：

```
输入：numCourses = 2, prerequisites = [[1,0]]
输出：true
解释：总共有 2 门课程。学习课程 1 之前，你需要完成课程 0 。这是可能的。
```

示例 2：

```
输入：numCourses = 2, prerequisites = [[1,0],[0,1]]
输出：false
解释：总共有 2 门课程。学习课程 1 之前，你需要先完成课程 0 ；并且学习课程 0
还应先完成课程 1 。这是不可能的。
```



```python
class Solution:
    def canFinish(self, numCourses: int, prerequisites: List[List[i
        def dfs(i, adjacency, flags):
            if flags[i] == -1:
                return True
            if flags[i] == 1:
                return False
            flags[i] = 1
            for j in adjacency[i]:
                if not dfs(j, adjacency, flags):
                    return False
            flags[i] = -1
            return True

        adjacency = [[] for _ in range(numCourses)]
        flags = [0 for _ in range(numCourses)]
        for cur, pre in prerequisites:
            adjacency[pre].append(cur)
        for i in range(numCourses):
            if not dfs(i, adjacency, flags):
                return False
        return True
```

210. 课程表 II

难度 中等 447 收藏 分享 切换为英文 接收动态 反

现在你总共有 *n* 门课需要选，记为 `0` 到 `n-1`。

在选修某些课程之前需要一些先修课程。 例如，想要学习课程 0 ，你需要先完成课程 1 ，我
配来表示他们: `[0,1]`

给定课程总量以及它们的先决条件，返回你为了学完所有课程所安排的学习顺序。

可能会有多个正确的顺序，你只要返回一种就可以了。如果不可能完成所有课程，返回一个空

示例 1:

```
输入: 2, [[1,0]]
输出: [0,1]
解释: 总共有 2 门课程。要学习课程 1，你需要先完成课程 0。因此，正确的课程顺序
[0,1] 。
```

示例 2:

```
输入: 4, [[1,0],[2,0],[3,1],[3,2]]
输出: [0,1,2,3] or [0,2,1,3]
解释: 总共有 4 门课程。要学习课程 3，你应该先完成课程 1 和课程 2。并且课程 1
2 都应该排在课程 0 之后。
     因此，一个正确的课程顺序是 [0,1,2,3] 。另一个正确的排序是 [0,2,1,3]
```

说明:

1. 输入的先决条件是由**边缘列表**表示的图形，而不是邻接矩阵。详情请参见图的表示法。
2. 你可以假定输入的先决条件中没有重复的边。


```python
class Solution:
    def findOrder(self, numCourses: int, prerequisites: List[List[in
        n = len(prerequisites)
        if n == 0:
            return [i for i in range(numCourses)]
        indegrees = [0 for _ in range(numCourses)]
        adj = [[] for _ in range(numCourses)]
        for cur, pre in prerequisites:
            indegrees[cur] += 1
            adj[pre].append(cur)
        res = []
        queue = []
        for i in range(numCourses):
            if indegrees[i] == 0:
                queue.append(i)
        while queue:
            pre = queue.pop(0)
            res.append(pre)
            for cur in adj[pre]:
                indegrees[cur] -= 1
                if indegrees[cur] == 0:
                    queue.append(cur)
        if len(res) != numCourses:
            return []
        return res
```

题目 1

子任务的两两之间可能存在依赖，比如[a,b]表示任务a必须要在b之前完成。现在有如下一组依赖：[b,d], [c,d], [f,b], [a,b], [g,h], [e,c]。写一个函数，输出任意一个合法的任务执行顺序，比如 fabecdgh

```python
nums = [['b', 'd'], ['c', 'd'], ['f', 'b'], ['a', 'b'], ['
def find(nums):
    indegrees = dict()
    adj = dict()
    for pre, cur in nums:
        indegrees[pre] = 0
        indegrees[cur] = 0
        adj.setdefault(pre, [])
        adj.setdefault(cur, [])

    for pre, cur in nums:
        indegrees[cur] += 1
        adj[pre].append(cur)

    res = []
    queue = []
    for str in indegrees:
        if indegrees[str] == 0:
            queue.append(str)

    while queue:
        pre = queue.pop(0)
        res.append(pre)
        for cur in adj[pre]:
            indegrees[cur] -= 1
            if indegrees[cur] == 0:
                queue.append(cur)
    return res

print(find(nums))
```