LeetCode207

作者:韩灵全

版本:2018-02-27

题目描述

There are a total of n courses you have to take, labeled from 0 to n - 1.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: [0,1]

Given the total number of courses and a list of prerequisite pairs, is it possible for you to finish all courses?

For example:

2, [[1,0]]
There are a total of 2 courses to take. To take course 1 you should have finished course 0. So it is possible.

思路报告

能不能上完0~n-1的所有课呢？什么意思呢？
我们从正面考虑的话，就是如果我从一个没有prereq的课开始上，上完它，再把那些把它设为prereq的课程上完，这样一层层的上，到最后，如果所有课都上完了，那就是可以了。
那好了，首先我们知道这里面体现了BFS思想（一层层），这样的话，我们一层层看得是什么样的层呢？ ---- 没有prereq的层，在图中怎么变现出来呢？

就是没有in-degree呗，那我就需要有一个数据结构记录每一节课的in-degree

那这些indegree怎么来的呢？
需要遍历每条edge（这些edge代表图），遍历的同时，我还是需要记录每个课下面是那些课，这是因为我需要知道，上完这节课，我需要去上什么课？

需要注意的细节:

List[] nextClasses = new List[numCourses];
数组存list，需要List[size]

代码如下

```
class Solution {
    public boolean canFinish(int numCourses, int[][] prerequisites) {
         List<Integer>[] nextClasses = new List[numCourses];
        int[] degree = new int[numCourses];

        //1.store the prerequ ralationship intp nextClasses and degree
        for (int i = 0; i < numCourses; i++) {
            nextClasses[i] = new ArrayList<Integer>();
        }
        for (int[] pair : prerequisites) {
            nextClasses[pair[1]].add(pair[0]);
            degree[pair[0]]++;
        }

        //2.if the class does not have prereq,add into queue
```

```
        Queue<Integer> queue = new ArrayDeque<>();
        for (int i = 0; i < numCourses; i++) {
            if (degree[i] == 0) {
                queue.offer(i);
            }
        }

        //3.remove the class does not have prereq, and update degree and queue
        int count = 0;
        while(!queue.isEmpty()) {
          for (int classNo : nextClasses[queue.poll()]) {
               degree[classNo]--;
              if (degree[classNo] == 0) {
                  queue.offer(classNo);
              }
          }
            count++;
        }

        return count == numCourses;
    }
}
```

套路总结

- 套路1: 思考问题可不可行的问题, 正面和反面考虑都可以, 即如果全满足了, 就可以; 和有一个不满足, 就不可以
- 套路2: 图的表示方式: Edge lists, Adjacency lists, Adjacency matrices