BFS

#G面试准备

Clone Graph

Clone Graph - LeetCode

- 要先根据起始点get所有的node
- 用bfs把所有的node放进一个set
- 建一个hashmap key是原node value是cloned node
  - value不是邻居列表
- 遍历nodeSet 初始化cloned node
- 对nodeSet里的每一个点 遍历邻居 拷到对应cloned node的邻居列表
  - 不能直接add neighbor 要add copied neighbor
- 返回hashmap中输入参数node对应的cloned node

Course Schedule

Course Schedule - LeetCode

- Topological sort
- 有向图 先从入度为0的开始 BFS地找 并把找到的点入度减一
- 如果入度减到0 就可以加入queue里了
- 需要维护一个入度数组或者hashmap
- 维护一个有向图的neighbors数组/hashmap
- 因为课号是从0到numCourses-1的数字 可以直接用数组 index作为key
- neighbors是一个List数组 每一个entry存的是一个ArrayList
- u ➔ v: v入度+1 u的邻居列表加入v
- 注意读题：[0,1]表示先上1再上0 1 ➔ 0
- while循环中遍历邻居时
  - int n = neighbors[cur].size();
  - for (int i = 0; i < n; i++)
    - int neighbor = (int) neighbors[cur].get(i);
  - 原来是 for (Integer nei : neighbors[cur]) 有Object转integer的错误 所以改
- 如何得到最后答案：检查sort完之后node数是否和总课数一致
  - 原想法：维护一个set 存sort完的node
  - 其实只需要用一个count变量 在queue.poll()的时候加一即可

Course Schedule II

Course Schedule II - LeetCode

- 注意读题：[0,1]表示先上1再上0 1 ➔ 0
- 跟上一题差不多 就是要输出排序的结果而已

- 上一题即使方向反了 还是能得到正确答案
- 这一题就要注意方向了 原来写反了

444. Sequence Reconstruction

Sequence Reconstruction

- topological sort 脑洞比较大 把他看成图 seq里的相邻数字两两有边
- 如果indegree == 0的node不止一个 就直接false 不止一种sort方法
- 建数组来存邻接表和indegree
  - index为0的地方弃用 这样后面写起来方便 不用老是减一
  - List<Integer>[] neighbors = new List[n + 1]; List[] neighbors = new List[n+1]
    - 要在前面加<Integer> 否则就默认是Object类的List
  - 问题: indegree数组都会初始化为0 如果seq里都是空的 这个图相当于连点都没有 但是下面的queue循环又会去找入度为0的点 这时候就会有bug
    - 初始化为-1 如果遇到这个点第一次出现再把他改成0
- queue做sort
  - BFS改邻居的indegree 如果indegree == 0 加入邻居
  - 如果当前queue的size大一1 说明不止有一种排序 可以false退出
  - 要存一个变量count 数当前已经排序了几个点
  - 如果count等于org的长度 才是true 不能直接return true 因为可能存在`[1][] or [1][[], []]`这种情况 queue会一直为空 不进循环 就没有机会输出里面的false
- 蠢的错误: seq是个List, for循环i < seq.length出错, 应该seq.size()
- 超级恶心的edge cases 错了6次…
  - seq里的数字越界 不在1..n范围 ➡ 建图邻接表的时候判断 不符合直接false退出

```
[5,3,2,4,1]
[[5,3,2,4],[4,1],[1],[3],[2,4],[1,1000000000]]
[1]
[[1,-9],[-9,-8],[-8,-9]]
```

  - seqs里没有seq

```
[1]
[]
```

  - seqs里都是空的seq

```
[1]
[[],[]]
```

- seqs里的seq都只有一个 ➡ 一开始算indegree的时候都是两个两个根据边算 没考虑只有点没有边的情况

```
[1]
[[1],[1],[1]]
```