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]]
```

269. Alien Dictionary

Alien Dictionary

- 拓扑排序 但并不好写 坑很多

```
[
  "wrt",
  "wrf",
  "er",
  "ett",
  "rftt"
]
```

- 两两给相同index的字加边 t ➡ f w ➡ e 等
- 一次取两个词 “wrt” “wrf”, “wrf” “er”…
- 要注意字相同的时候就不要加边了
- 如果遇到第一个mismatch 加边 马上break退出这个循环 因为这个地方有顺序 后面的字眼顺序已经没有意义了 这个很重要 忽略了
- 在queue bfs的时候要判断当前字有没有在邻接表里 再去找其neighbors 否则null pointer
  - 参考讨论的设计是degree每个字都初始为0 但是邻接表却不一定有 因为有的字不能确定顺序 不一定有边
- 最后输出的时候先判断是否size到了所有字母的个数 如果有才输出 没有就输出空字符串

490. The Maze

The Maze

- 可以用bfs做
- 题意: 球从起点开始是选择一个方向不停地滚 直到撞墙才会停下 再选择下一个方向接着滚
- 判断能不能从start滚到destination停住 如果经过但是停不住也是false

- 思路: bfs的queue存每个stop node
- hash isvisited只会存stop node有没有走过 不会把所有经过的点都存下来 不然交叉路就走不了了
- 从queue里取出一个node 上下左右进入循环
  - 选择其中一个方向 next_x next_y初始化为x y 然后一直往那个方向走到走不动为止
  - while循环跳出的时候next_x next_y是越界的点 所以要减回来那个方向 就是stop node
  - 如果这个stop node isvisited了 跳过
  - 置isvisited = true
  - 看当前stop node是不是我们要找的destination 是就直接return true
  - 不是就继续 把点加进queue里 继续bfs
- 都找不到那个点就return false

505. The Maze II

The Maze II

- 可以用Dijkstra做 把每个stop node看成图里的一个vertice 求起点到终点的最短距离
- 不要用hash 因为最短路算法的最短距离会一直改的
- 思想就是用一个Priority queue 先把起点放进去 离起点距离为0 然后找edge 更新每个stop node的距离 放进pq里
- queue去取离起点最小的node 再继续往邻居去遍历 新增距离或者把pq里有距离的node距离变小
- 最后是要得到终点离起点的距离 距离要存在一个length二维数组里 每次pq.poll的时候就可以把这个node的最短距离填进结果数组length里
- 最后看终点length[d[0]][d[1]]有没有值 有值就输出 没有值输出-1
- 自己的一个bug:
  - 不要在进while之前就把起点的length[s[0]][s[1]]改为0 否则会在`if (length[x][y] <= p.len) continue;`的时候就退出 全程只有一个点进出pq
  - 要在pq poll的时候再去更新最短距离