亚马逊 OA

Round Robin: https://www.youtube.com/watch?v=aWlQYllBZDs

一道是 Valid parentheses
不知道为什么 有三个 test case 怎么都过不去。。。。

还有一道是 bst 找 distance 的题
signiture 大概是
public static int bstDistance(int[] values, int n, int node1, int node2)

做完 coding 是 15min 的一个什么测试

1. 水果题
开始在找机经的时候发现有些人在问 水果 codelist 中间 能不能插入其他水果，这里是不允许的；另外，anything 也必须 match 一个水果，不能多个或者没有，所以不是 wildcard, 其实很好做，loop 一遍 codelist 就行了
2. movie network
输出 N 个最高评分的关联电影
BFS + min heap of N 搞定

1.买水果
public static int checkWinner (List<List<String>> codeList, List<String> shoppingCart) {}
说的是小明要帮公司买水果，给了一个 codeList， 里面装的是他买的水果，给了一个 shoppingCart 里面装的是 target 水果，目标是检查 codelist 里的水果顺序是否和 shoppingCart 里的顺序匹配。

注意的是只有 codelist 中的所有链表的 item 之后加起来小于等于 shoppingcart 里 item 之和才可能返回 1。 另外在 codelist 中有可能出现 item 时 "anything"， 它可以和任意的水果匹配。
Ex1:
codelist:
[
[apple, apple],
[orange, banana, orange]
]
shoppingCart: [orange, apple, apple, orange, banana, orange]
return 1, 因为 codelist 里的顺序和 shoppingcart 里除了第一个 orange 之后的水果顺序匹配

Ex2:
codelist:
[
[orange, banana, orange],
[apple, apple]
]
shoppingCart: [orange, apple, apple, orange, banana, orange]
return 0, 因为 codeList 里的顺序和 shoppingcart 没法匹配。

Ex3:
codelist:
[
[apple, apple],
[orange, banana, orange],
[pear, orange, grape]
]
shoppingCart: [orange, apple, apple, orange, banana, orange, pear, grape]
return 0, 因为 codelist 里最后一个[pear, orange, grape]中的 orange 没法和 shoppingcart 里的水果匹配。

Ex4:
codeList:
[
[apple, apple],
[orange, anything, orange]
]
shoppingCart:
[orange, apple, apple, orange, mango, orange]
return 1。

golf 球场修场地。
public int flatFields (int numRows, int numColumns, List<List<Integer>> fields) {}
让小明帮公司球场修场地，给一个二维的链表 fields，场地里有坑，不能走。 场地里有树要砍掉。最后目的返回是修好一层的场地的最小步数。
Ex1:
[
[1, 3, 0, 2]
[1, 1, 3, 1]
]
上图中的 1 代表平地，可以走。 0 代表坑，不能走。 大于 1 的数字代表树木，需要砍掉。规则是从上下左右四个角开始任选一个开始走，先砍数字小的树木。 比如 2 < 3，那么就得先走 2。
上图如果从右下角开始走依次经过的坐标是： （1， 3） -> (0, 3) -> (1, 3) -> (1, 2) -> (1, 1) -> (1, 0) 所以返回的最小步数是 5， 因为通过这个路径可以修平第二层的球场[1, 1, 3, 1]， 并且走到左下角终点。
Ex2:
[
[1, 0]
[3, 2]
]
上图中的最小步数返回-1 因为，没有办法修好一层， 因为从左上角 1 开始走，不能走到 0， 也不能走 3， 因为在全局中 3 比 2 大，必须先走 2。所以就没法走了。

第一题：shopping list 和 basket
给一个 shopping list 比如说 [[apple, banana], [orange, apple], [anything, orange]]

然后 checkbasket 的买东西是不是 match shopping list. 里层数组的东西和顺序必须 match，数组与数组之间可以插入任何东西， anything 表示任何东西都可以买
比如 basket [apple, banana, apple, apple, orange, apple, banana, orange, apple, orange]就是 match 的，return 1

第二题：跟这里的第二题一样 http://www.1point3acres.com/bbs/thread-288537-1-1.html

先用 pq 按顺序保存要砍的树，然后 bst 找到每棵要砍的树，注意 BST 找路径的时候 condition 是 check if height ==1，但是到了目标树它的高度其实是 > 1 的，所以 bst 在达到目标树之前就会停止，导致找不到路径。 所以要先把目标取出来，然后把树砍了（set height = 1），然后再用 BST 找。 LZ 就是把这两句写反了，没时间改了。。。。。时间都花在修我家的破网和解决 comparator generics 的 compare method 必须要 public 但是 solution 的 method 是 protected 上面了。。。哭。。。
第二题的树的高度好像没有相等的，如果要考虑树的高度相等的情况，那就很麻烦了。

Amazon warehouse。。。其实就是给你 x,y 然后算 x,y 到原点的距离，输出最小的几个，java 应该 priorityqueue 就够了，我用的 python，也还可以。
第二题，golf event 要砍树。。。每次只能砍所有树里面最矮的那颗。其实就是 maze 题的变形。2D-array. 0 不能走，1 可以走，>1 就是树，要求的输出就是从原点开始，走到每颗当前树里面最矮的那颗所需的步数+需要砍得树的高度的总和。方法我就是先找好所有的树，排好序，然后从一个点到另一个点做 BFS。 找出最小步数。
举个例子 [[1,1,0,2],[3,1,1,1]], 从（0，0）走到 （0，3）--》2 这棵树，就是 5 步+2（树高），然后从（0，3）走到 （1，0）->3 这棵树 4 步+3（树高）所以 5+2+4+3 返回 14

```
def levelFieldTime(numRows, numColumns, field):
    # WRITE YOUR CODE HERE
    import collections
    dicts = {}
    for i in range(numRows):
        for j in range(numColumns):
            if field[i][j] > 1:
                dicts[field[i][j]] = (i,j)
    lists = sorted(dicts.iterkeys())
    def findsteps(start, end, numRows, numColums, field):
        visited = [[0 for _ in range(numRows)] for _ in range(numColums)]
        direct = [(0,1),(0,-1),(1,0),(-1,0)]
        queue = collections.deque()
        queue.append(start)
        steps = 0
        while queue:
            steps += 1
            n = len(queue)
            while n > 0:
                x,y = queue.popleft()
                for kx, ky in direct:
                    i, j = x + kx, y + ky
                    if (i,j) == end:
                        return steps
```

```
                if i >= 0and i < numRows and j >= 0 and j < numColumns and field[i][j] == 1 and
visited[i][j] == 0:
                    visited[i][j] = 1
                    queue.append((i,j))
            n -= 1
        return -1
    ans = 0
    start = (0,0)
    for i in lists:
        end = dicts[i]
        x,y = end
        res = findsteps(start, end, numRows, numColumns, field)
        if res == -1:
            return -1
        ans += (res + field[x][y])
        field[x][y] = 1
        start = end
    return ans
field = [[1,1,0,12,1,13],
         [1,1,1,1,0,0],
         [0,1,0,0,0,0],
         [0,1,1,1,14,0],
         [0,0,0,0,1,0], 1point3acres.com/bbs
         [15,1,1,1,1,1]]
print levelFieldTime(6,6,field)
```

1. 水果清单
刚开始有三四个 testcase 没过
* 不要忘记写 anything 的情况
* 如果 codeList 和 shoppingCartList 都为空时，应该返回 1。
修改后就 AC 了。

codelist:
[
[apple, apple],
[orange, banana, orange]
]
shoppingCart: [orange, apple, apple, lychee, orange, banana, orange]

分别把 codelist 里的东西按顺序匹配在 shopping cart 中
在 shopping cart 中， 一个 code list 里的东西不能分开。红的里是 codelist[0], 橙色里是 codelist[1]。它们中间隔了一个 lychee, 这时允许的。 同时，在 code list 里的组合在 shopping cart 中必须依次出现， 顺序不能打乱。

思路：

1. 在 shopping cart 中找到匹配的 codelist[0], 这个类似 https://leetcode.com/problems/implement-strstr/， 假设找到的字符串在 shopping cart 中的结束坐标为 index,
2. 从 index + 1 开始找 codeList[1]
3. 重复上述过程，直到把 codeList 里面的元素都找到返回 1,
4. 如果已经搜索到 shopping cart 结尾，仍未找到 codeList 里所有元素， 就返回 0

2. 给一个无序数组，构建 bst，找出给的距离
首先查看给的两个 node 是否在数组里都能找到，如果不能则直接返回-1
接着写几个函数：
*通过每次从 root 查找建 BST,
*找 LCA，返回该 lca node,
*计算 LCA node 与 node1 距离，计算 LCA node 与 node2 距离。
*返回距离和

第一个题是求到给定点的最近的 k 个点的问题吗

第一题是 k closest point 吧 lincode 原题

1. 请问第二题可以走有数的地方么？ 2. 树被砍了之后是默认这个地方为 1 还是 0？ 3. 可能出现一个多颗高度一样的树么？

1.不可以 2.砍完是 1 3.不会

golf 球场砍树

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=291981&pid=3149745&page=1&extra=page%3D1%26filter%3Dsortid%26sortid%3D311#pid3149745

链接: https://instant.1point3acres.com/thread/278257
来源: 一亩三分地

* 你有没有在时限很短的情况下完成一个项目，你是怎么调节你的时间的，你是否因此必须牺牲什么？ * 有时候碰到 deadline 快到，但是项目遇到问题有可能做不完，怎么处理. * 你最 proud of 的一个项目是什么 * 你最成功 deliver project 的经验

你有没有没赶上 DeadLine 的项目 Could done better Describe a situation that you have done a lot on the project, but you find your method for the problem is not good, how would you deal with

that ? how you deal with a coworker who dont like you, who you dont like Build relationship with co-worker Why amazon?

链接: https://instant.1point3acres.com/thread/277830
来源: 一亩三分地

重点来了，之前的 coding 好像只有 4 选 2，我碰到了 2 个不一样的！！ 第一题是 winsum 变形，［1，2，3，4，5］ k = 2, 输入每个 window 里小的那一个， 也就是说输出［1，2，3，4］

就是给一个 list ［1，2，3，4，5］ 和一个 window size， 输出每一个 window 里较小的那一个， 不如第一个 window ［1，2］， 小的那个是 1， 所以输出的 list 里第一个数是 1，以此类推

链接: https://instant.1point3acres.com/thread/287052
来源: 一亩三分地

第二题 ITEM ASSOCIATION。 input，表示物品 A 和物品 B 相互关联。，表示物品 B 和物品 C 相互关联。 如果物品相互关联，就组成一个组。最后要求找出物品最多的那个组。 不难但是最后有个测试用例一直过不了。三天后还是收到 EMAIL，这周五 ONSITE.

我就用 list> 把组都存起来。来一个新 pair: itemA, itemB,就扫描一下看看是不是分别已经和以后的组关联。如果 AB 关联不同组，需要把组合并。否则加到已有组或者新建一个组 ssa

TestCase 2:

Input:
[0, 1, 3, 9], [2, 1, 7, 5], 2
Expected Return Value:
1.0

Explanation:
The processes run in the following time slots - P1 initially runs for 2 seconds, P2 runs for 1 second, P3 runs for 6 seconds till P4 enters the system at the 9th second - when it runs for 2 seconds, P3 then runs for 1 second followed by P4 running for 3 seconds. The waiting time of processes P1, P2, P3, P4 are 0, 1, 2, 1 respectively. The average is thus 1 second.

我是这么理解的，有不对的地方请指出
我拿贴子里面给出的例子来说明下吧。
【0，1，4】【5，2，3】q ＝ 3

第 0 秒，任务 1 进队列。我们 peek 目前队列中有的任务是任务 1，我们发现任务 1 开始时间第 0 秒，目前也是第 0 秒
这时候我们查看 3 秒的时候哪些任务达到了，发现任务 2 在第 1 秒到达。于是任务 2 进队列。
这时候我们查看任务 1 有没有执行完，发现没有执行完，于是我们 poll 任务 1，再把任务 1 add 到队列末尾。
这时候队列的顺序是任务 2，任务 1.

现在我们再次 peek 队列，于是找到任务 2.我们发现任务 2 在第 1 秒到达了，目前我们在第 3 秒。所以等待时间是 3-1=
我们重复刚刚的步骤，发现任务 2 执行时间只要 2 秒，于是我们到第 5 秒。这时候我们查找第 5 秒哪些任务到达了。我
所以目前队列顺序是任务 2，任务 1，任务 3.
我们又发现任务 2 已经执行完了，于是

我们把任务 2 poll 出队列，不再把它放进队列里了。
所以现在队列里面剩余的任务是任务 1，任务 3.

于是我们再 peek 队列。请注意，这时候的 q 被重新设置过了，不是 3－2=1 秒，而是又是 3 秒。
我们再次 peek 队列，找到任务 1，目前是在第 5 秒，我们刚刚执行过任务 1，他暂停在第 3 秒，所以任务 1 又等了 2 秒
目前任务 1 还有 2 秒。我们执行完任务 1 以后，到达第 5+2 ＝ 7 把他扔出队列。目前队列里只有任务 3 了。

然后我们再 peek，现在只有任务 3 了，目前我们在第 7 秒。任务 3 进来的时候在第 4 秒。所以任务 3 等了 7-4 ＝ 3 秒。

所以等待时间又加 3 秒。
所以最终等待时间是 2+2+3 ＝ 7 秒。平均等待时间是 7/3 ＝ 2.3333 秒。

```
public class process {
        int arriveTime;
        int excuteTime;
        process(int arr, int exc) {
```

```
            arriveTime = arr;
            excuteTime = exc;
        }
}

// Assume arrive is sorted.
public double roundRobin(int[] arrive, int[] excute, int q) {
        LinkedList<process> queue = new LinkedList<>();
        int curTime = 0;
        int waitTime = 0;
        int nextProIdx = 0;
        while (!queue.isEmpty() || nextProIdx < arrive.length) {
                if (!queue.isEmpty()) {
                        process cur = queue.poll();
                        waitTime += curTime - cur.arriveTime;
                        curTime += Math.min(cur.excuteTime, q);
                        for (int i = nextProIdx; i < arrive.length; i ++) {
                                if (arrive[i] <= curTime) {
                                        queue.offer(new process(arrive[i], excute[i]));
                                        nextProIdx = i + 1;
                                } else {
                                        break;
                                }
                        }
                        if (cur.excuteTime > q) {
                                queue.offer(new process(curTime, cur.excuteTime - q);

                        }
                } else {
                        queue.offer(new process(arrive[nextProIdx], excute[nextProIdx]));
                        curTime = arrive[nextProIdx ++];
                }
        }

        return (double)waitTime / arrive.length;
)
```