建了个讨论群，希望大家踊跃讨论新题难题，谢谢
(群已超过100人，无法扫描二维码加入了，可以加我微信MichaelWang91，还请备注下:google面经)

大家好，面经可随意地里或国人传播，但不能有商业用途。如果思路和代码有错欢迎原贴纠正，或直接doc改正，Doc每天都有人更新，会同步大家的有用修改

（踊跃提问，请用蓝色）
为了统计高频面经，请大家如果面试见过类似题目可以修改频率
如果有新题目，可以直接添加到后面的日期下面

1.自行车和人匹配问题 (高频 13次)

自行车这道题目如果有同学能提供KM算法（匈牙利算法）的实现请直接修改doc

Update time： 【2018-10-19】

2D平面上，有m个人（P），n辆自行车(B)，还有空白（O）满足以下条件
1.m < n
2.不存在两个人，到同一辆自行车距离相等, 距离用abs(x1-x2) + abs(y1-y2)定义
3.每个人尽量找离自己最近的自行车，一旦某辆自行车被占，其他人只能找别的自行车。
例：
OPOBOOP
OOOOOOO
OOOOOOO
OOOOOOO
BOOBOOB
红色的人找到第一行的自行车，距离最近。
蓝色的人离第一行自行车最近，但自行车已经被红色人占有，所以他只能找离他第二近的，右下角的自行车。

问：把人和自行车配对，输出vector<pair<int, int>>每个人对应的自行车. {i, j} 是人i对应自行车j

wtcupup 发表于 2018

大米已加， 求问自行车分配问题的思路？

可以搞几个priority_queue<距离，人，车>
然后遍历m个人，n辆车，把m*n个 距离，人，车放入其中。
每次取出距离最小的，看看这个人是否已经分配了自行车
如果没有分配，则可以把这个人和这辆车输出

FightOn 发表于 2018-3-22 04:25

谢谢楼主的解答！我有一个问题，面试官要求说所有人完成配对时候所走的总路线最小么？ lc的解法可以完成配 ...

我印象中面试官没有问 只是问如何给每个人分配一个自行车

devilnut 发表于 2018-3-22 07:48

也可能没理解题目或者楼主的做法 比如下面这个

BOOPBOOOOOOOOOOOOP

如果先给了左边第一个P最近的B 总 ...

看来确实是有反例的
我面试时候，面试官的要求是每个人都同时开始找，尽量找自己近的。假设每个人向四周的搜索速度是一样的，算是局部最优。
如果要求变成总距离最短，我的方法就不对啦。

思路：attempting 算出所有距离后放进heap里，依次pop出来记录人和车的状态
这道题具体是什么呀？有几个人？人和车配对？
查了半天相关的帖子，我的理解是：貌似就是2D grid，用0代表road，用1代表obstacles，然后B代表bike，用P代表people。然后是有m个人，n个车，m < n
Editor: 是的 车比人多 enum出所有人车匹配并排序 后从小网大排序输出 用双set去重就好 不是难题
所以这道题跟bfs/dfs没关系，就直接heap不停poll,然后用set来记录bike和people被visited 即可

如果有距离相同的情况的话怎么解呢？只能用匈牙利吗
Editor: 相同情况感觉可以跟面试官讨论

我觉得应该是类似如下情况：
0 P 0
B 0 P

0 0 B
P代表people，B代表bike，0就是过道，那么这种情况下就只需要enumerate出每一个B和每一P的abs distance，放进priorityqueue即可。然后用一个char[][]来记录visited。这是我的看法。但如果出现了“1”作为obstacles，就不能直接abs了，因为有障碍物，这个时候我能想到的就是以P为中信用bfs来计算到每一个B的距离，然后再放进heap。不晓得我的想法是否正确。
Editor: 对的

ttforsythe 发表于 2018-12-23 14:43

请问楼主，第四题匹配的时候，最后的目标是尽量多匹配还是让总的cost最小。

我面试的时候是尽量多匹配，优先安排距离最近的，并不考虑全局最优

linspiration 发表于 2018-12-21 16:55

恭喜楼主，我记得第四题地里的讨论比较开放，楼主是怎么做的?

当时没有很好的思路，就直接暴力得写了pq + map的解法，看所有匹配的可能。
写完了再跟面试官讨论如何优化，也没有特别好的思路。
最后面试官跟我提了Hungarian algorithm。

Summary:

1. 需要跟面试官讨论清楚他需要的最佳匹配是什么
2. 如果是要求全局人车距离最短
    a. 二分图的最佳匹配问题，使用匈牙利算法，参考题目 https://blog.csdn.net/u011721440/article/details/38169201
3. 如果是要求最佳匹配只是给每个人匹配到车，可以用PQ+Map

地里讨论专帖：狗家近期的高频，人车匹配，希望大家讨论一下（附我知道的followup版本）

2. 二叉树删除边 (高频 10次)

~~LC684~~，BT删除多余边
思路：
684是无向图，用union find，遍历edge，每次把parent付给做节点，若发现母节点相同则为多余
685 三种错误情况，multiple parents，cycle， both 在both的情况下仔细讨论删除第一个指向multiple parent的node
685 有很多隐含条件：比如 cycle 最多只可能有一个， multiple parents 最多两个，没有multiple parents 的话就一定有cycle, 这样就多了很多限制，分情况讨论的时候就会简单许多。

Follow up: 给一棵二叉搜索树，有一条多余边，删除它

例子：

```
   7
  / \
 5   9
/ \ /
3  8
```

对于多余边5-8，9-8此处的删除需要有选择，跟之前的题目找到多余边立马不分选择删除有区别

思路：LC 98: Validate Binary Search Tree
DFS，参数中带着左右边界，返回值为新树的根节点，如果当前节点不在当前树的范围内，返回null删除该边

参考代码

```
public void deleteEdge(TreeNode root) {
      if(root == null) return;
      root = dfs(root, Integer.MIN_VALUE, Integer.MAX_VALUE);
}
private TreeNode dfs(TreeNode root, int left, int right) {
      if(root == null) return null;
      if(root.val <= left || root.val >= right) return null;
      root.left = dfs(root.left, left, root.val);
      root.right = dfs(root.right, root.val, right);
      return root;
}
```

能否说一句无关的废话，上面BST删除多余的edge的思路，和BST insert以及delete node的思路很类似，可以放到一起总结复习。(Editor：嗯，思路差不多，但是follow up用DFS感觉好做一点，第一题UF和DFS都能做)

~~3.~~ 机器人左上到右上(高频 9次)

~~LC~~类似题目: LC 62 Unique Paths 和 LC 63 Unique Paths II
原题描述: https://www.1point3acres.com/bbs/thread-423857-1-1.html
给定一个矩形的宽和长，求所有可能的路径数量

Rules：
1. 从左上角走到右上角
2. 机器人只能走右上，右和右下

思路：

1. 按照列dp, dp[i][j] = dp[i - 1][j - 1] + dp[i][j - 1] + dp[i + 1][j - 1]， 注意i-1，i+1需要存在

followup1: 优化空间复杂度至 O(n)

思路：只保留上一列的空间，用两个数组滚动dp
参考代码

```
public int uniquePaths(int rows, int cols) {
      int[] dp = new int[rows];
      int[] tmp = new int[rows];
      dp[0] = 1;
      for(int j = 1 ; j  < cols ; j++) {
            for(int i = 0 ; i < rows ; i++) {
                  int val1 = i - 1 >= 0 ? dp[i - 1] : 0;
                  int val2 = dp[i];
                  int val3 = i + 1 < rows ? dp[i + 1] : 0;
                  tmp[i] = val1 + val2 + val3;
            }
            System.arraycopy(tmp, 0, dp, 0, tmp.length);
      }
      return dp[0];
}
```

followup2: 给定矩形里的三个点，判断是否存在遍历这三个点的路经

思路：

假设三个点坐标为(x1, y1) (x2, y2) (x3, y3)
1：从(0,0)出发，如果后一个点在前一个点展开的扇形区域内，则可以被达到
2：先对三个点按照纵坐标y排序，如果一个y上有一个以上的点，则返回false
请问这是是不是应该是如果x上有一样的点返回false啊，方向如果是→ ↗ ↘的话，x应该是单调增的 y可以不是单调吧，还是我理解的题目不太对，感谢 （Editor: 这里的Y是列坐标，即j，不是坐标系中的纵坐标）
(这个没有看太懂从（0 0）怎么走？ 是不是不只是向右 和 向下)
（Editor：这个只是follow up，题目还是原题，只能往右边，右上和右下走）
3：对于(xi, yi)，得到前一个点在该列的可达范围
len = yi - y(i-1)
upper = x(i-1) - len
lower = x(i-1) + len
如果[xi]在这个范围内，则可达
参考代码

```
public boolean canReach(int[][] points) {
      List<int[]> list = new ArrayList<>();
```

```
        list.add(new int[] {0, 0});
        for(int[] point : points) list.add(point);
        Collections.sort(list, (a, b) -> {
            return a[1] - b[1];
        });
        for(int i = 1 ; i < list.size() ; i++) {
            int[] curr = list.get(i);
            int[] prev = list.get(i-1);
            if(curr[1] == prev[1]) return false;
            int len = curr[1] - prev[1];
            int upper = prev[0] - len;
            int lower = prev[0] + len;
            if(curr[0] <= lower && curr[0] >= upper) continue;
            else return false;
        }
        return true;
    }
```

followup3: 给定矩形里的三个点，找到遍历这三个点的所有路径数量

思路：

1：还是按照follow up 1的思路用滚动数组dp，但是如果当前列有需要到达的点时，只对该点进行dp，其他格子全部置零，表示我们只care这一列上经过目标点的路径

2：如果一列上有多个需要达到的点，直接返回0；

参考代码

```
    public int uniquePaths(int rows, int cols, int[][] points) {
        int[] dp = new int[rows];
        int[] tmp = new int[rows];
        Map<Integer, Integer> map = new HashMap<>();
        for(int[] point : points) {
            if(map.containsKey(point[1])) {
                return 0;
            } else {
                map.put(point[1], point[0]);
            }
        }
        int res = 0;
        dp[0] = 1;
        for(int j = 1 ; j  < cols ; j++) {
            for(int i = 0 ; i < rows ; i++) {
                int val1 = i - 1 >= 0 ? dp[i - 1] : 0;
                int val2 = dp[i];
```

```
                        int val3 = i + 1 < rows ? dp[i + 1] : 0;
                        tmp[i] = val1 + val2 + val3;
                  }
                  System.arraycopy(tmp, 0, dp, 0, tmp.length);
                  if(map.containsKey(j)) {
                        int row = map.get(j);
                        for(int i = 0 ; i < rows ; i++) {
                              if(i != row) dp[i] = 0;
                              else res = dp[i];
                        }
                  }
            }
            return res;
      }
```

followup4: 给定一个下界 (x == H)，找到能经过给定下界的所有从左上到右上的路径数量 (x >= H)

思路：

1：先dp一遍，得到所有到右上的路径数量

2：然后在 0<=x<=H, 0<=y<=cols 这个小矩形中再DP一遍得到不经过下界的所有路径数量

3：两个结果相减得到最终路径数量

Code

重用follow up 1的代码

```
public int uniquePaths(int rows, int cols, int H) {
      return uniquePaths(rows, cols) - uniquePaths(H, cols);
}
```

followup5: 起点和终点改成从左上到左下，每一步只能↓↘↙，求所有可能的路径数量

参考代码：按照 行 dp，其他地方不变

```
      public int uniquePaths(int rows, int cols) {
            int[] dp = new int[cols];
            int[] tmp = new int[cols];
            dp[0] = 1;
            for(int i = 1 ; i  < rows ; i++) {
                  for(int j = 0 ; j < cols ; j++) {
                        int val1 = j - 1 >= 0 ? dp[j - 1] : 0;
                        int val2 = dp[j];
```

```
                        int val3 = j + 1 < cols ? dp[j + 1] : 0;
                        tmp[i] = val1 + val2 + val3;
                    }
                    System.arraycopy(tmp, 0, dp, 0, tmp.length);
                }
                return dp[0];
            }
```

补充一个该题目变种：

Given a N*N matrix with random amount of money in each cell, you start from top-left, and can only move from left to right, or top to bottom one step at a time until you hit the bottom right cell. Find the path with max amount of money on its way.
Sample data:
start
|
v
5, 15,20, ...
10, 15, 5, ...
30, 5, 5, ...
…
^end here.

思路：LC 62带上权值，思路差不多，只是求和变成求相加后的最大值

Follow up 1：要求重建从end 到 start的路径

思路：用另一个额外数组记录每一步选择的parent，dp结束后，从end依次访问它的parent重建路径

Follow up 2: 现在要求空间复杂度为O（1），dp且重建路径

空间复杂度不算返回路径时需要的空间
思路：直接修改原数组，而且带上符号，负号表示从当前cell的左边过来，正号表示从当前cell的上边过来，dp结束后从end 依次访问它的parent重建路径

参考代码

```
    public List<List<Integer>> maxMoney(int[][] moneys) {
        // assume: moneys is not null, width and length are equal
        int n = moneys.length;
        if (n == 0)
            return new ArrayList<>();
```

```
            // base case
            for (int j = 1; j < n; j++) {
                  moneys[0][j] = -(Math.abs(moneys[0][j-1]) +
moneys[0][j]);
            }
            for (int i = 1 ; i < n ; i++) {
                  moneys[i][0] = moneys[i-1][0] + moneys[i][0];
            }
            for(int i = 1; i < n ; i++) {
                  for(int j = 1; j < n ; j++) {
                        int top = Math.abs(moneys[i-1][j]) + moneys[i][j];
                        int left = Math.abs(moneys[i][j-1]) + moneys[i][j];
                        if(top >= left) moneys[i][j] = top;
                        else moneys[i][j] = -left;
                  }
            }
            System.out.println("Max path sum = " + Math.abs(moneys[n -
1][n - 1]));
            List<List<Integer>> path = new ArrayList<>();
            int curri = n-1;
            int currj = n-1;
            while (curri != 0 || currj != 0) {
                  path.add(Arrays.asList(curri, currj));
                  if(moneys[curri][currj] < 0) {
                        currj -= 1;
                  } else {
                        curri -=1;
                  }
            }
            path.add(Arrays.asList(0, 0));
            return path;
      }
```

follow up：

补充：若机器人只能走右上，右和右下。请问

1. 有三个点需要遍历
2. 如何判断三个点一个是合理的，即存在遍历三个点的路径
3. 给H，需要向下越过H界

思路：

1. 用三个点切割矩形，然后每个小块复用之前的方法，最后做乘积 什么叫做用3个点切割矩形哇？就是用起始点到最左点对角线的矩形是第一个矩形，第一点到第二点构成的矩形为第二矩形，以此类推我们就得出三个小矩形，分别计算并乘积
2. 切割后的矩形高度不能超过宽度，不然没法走到 为什么高度不能超过宽度哇，我觉得可以啊，感觉前面的点的横纵坐标比后面的切割点的横纵坐标小或者等于才可以遍历所有点，

跟矩形高度宽度有什么关系呢，不懂 因为如果高度为5，长度为2 从左上角到右下角走不过去 为啥走不过去啊？就是个矩形，往下面走就好了啊，不懂诶，这个跟矩形高度宽度有啥关系。不好意思，看followup补充。题目略有变动我忘记说了

3. 若给定H，先总的dp走一遍，再不越界走一遍（不越界走一遍是什么意思哇 就是用高度为H 宽度相等的矩形再来一遍），相减即可 给定 h是什么意思哇，是高度的意思么？就是给你横着画一条线，然后从左上到右上的所有路线 必须经过这条线或以下的区域

4. Guess Word 高频 9次

LC 题目：LC843 猜词，一个未知target，猜一个词会返回正确猜对的字母数

第一轮 白人小哥。猜词游戏，写一下如何判断player猜词的score（guess word 和 secret word中相同char的个数），然后如何根据history判断guess word是不是good guess。

思路：每次尽量二分筛选 这样比较优化
每次尽可能缩小搜索空间，除了LC上的解法如果有其他新解法欢迎修改。

5. LC890 word pattern match 高频8次

题目描述：
给一个word list和一个pattern，返回list中所有和pattern相match的单词
此处的match为能在pattern和word之间找到一个双射函数，和LC 205 Isomorphic String中的双射函数一样

思路：
1. 用两个map，用putIfAbsent存互相的对应关系，然后再查一遍对应
2. 单map把string转换成pattern array，用map.put(char, map.size())存不存在的char
这道题目本质是LC205

6. LC489 位置地形扫地机器人 高频 7次

思路：常规DFS

1. 需要用参数追踪当前机器人朝向
2. 每次backtracking时候别忘了掉头回正
3. 用string记录visit过的位置

7. LC855 考试找位子，尽量分散坐，人会离开 高频 6次

思路：
1.用优先队列：
用优先队列存slot，slot包含左右端点和长度。exclusive好算。注意如果是最左或最右时长度为right - left，若非则为(right - left) / 2，因为如果选择坐边上可以不管端点。seat时候去pq最大slot，中间切开offer两段。leave时候遍历找到左右两段整合

离开时间复杂度 O(n)
坐下时间复杂度 O(logn)

2. 用TreeSet
每次坐下时，遍历set找到最大的距离，并且记录位置，离开则直接删除目标数字

离开时间复杂度O(logn)
坐下时间复杂度O(n)

8. Key有过期时间的hashmap 高频 6次

题目：
面试官是个安卓组的小姐姐，45分钟，感觉答得一般，求过o 0
Create a map with expiring entries:
Example
12:00:00 - put(10, 25, 5000)
12:00:04 - get(10) -> 25
12:00:06 - get(10) -> null

思路：两个hash map，一个记录key，value pair，一个记录key的过期时间，get的时候检查key是否过期，如果过期了，删除key返回null
Put方法有三个参数，除了key，value还有个duration

Follow up: 采用更主动的策略删除过期的Key

思路；创建后台线程定期清理过期的Key。
用两个map，一个装<key, value>一个装<key, expiredTime>
在get中采用lazy deletion，get的时候检查key是否过期，如果过期的话两个map中都删除key，返回null。put的时候每次都更新key的expiredTime。

后台线程每过一段时间遍历所有key，调用get方法删除过期key。此处为了避免多线程冲突，Map用ConcurrentHashMap实现。

参考代码 （用后台线程主动删除）

```
class MyMap<K, V> {
	Map<K, V> map;
	Map<K, Long> time;
	private static final int DEFAULT_CAPACITY = 16;
	private static final float DEFAULT_LOAD_FACTOR = 0.75f;
	private Thread clearThread = new Thread(new Runnable() {
		@Override
		public void run() {
			while(true) {
				try {
					Thread.sleep(5000);
				}catch(Exception e) {
					e.printStackTrace();
				}
				for(K key : map.keySet()) get(key);
			}
		}

	});
	public MyMap() {
		this(DEFAULT_CAPACITY, DEFAULT_LOAD_FACTOR);
	}
	public MyMap(int capacity) {
		this(capacity, DEFAULT_LOAD_FACTOR);
	}
	public MyMap(int capacity, float loadFactor) {
		map = new ConcurrentHashMap<>(capacity, loadFactor);
		time = new ConcurrentHashMap<>(capacity, loadFactor);
		clearThread.start();
	}
	public V get(K key) {
		long now = System.currentTimeMillis();
		Long expired = time.get(key);
		if(expired == null) return null;
		if(Double.compare(now, expired) > 0) {
			map.remove(key);
			time.remove(key);
			return null;
```

```
                } else {
                        return map.get(key);
                }
        }
        public V put(K key, V value, long duration) {
                long now = System.currentTimeMillis();
                long expired = now + duration;
                time.put(key, expired);
                return map.put(key, value);
        }
}
```

9. 多个不重复的长方形内随机取点【类似LC497?】高频 6次

原帖

onsite最后一轮的面试题。其他轮都没啥发的必要就不发了。题目是这样的 给你一个list的长方形。。每个长方形面积不一样，但是你要取个点，这个点可以是任何长方形里的。但是要你每次取点的概率都是一样的。不会因为长方形大小而不同。
长方形的输入形式：左下坐标和长宽
（主要考察加权抽样和merge squares）
思路：把重复的长方形分成不重复的LC 类似题目：小块，然后用prefix sum进行二分查找
请问follow up是啥思路？用什么思路处理重叠的矩阵？以及是否需要考虑三重或者更多重的重叠区域

LC850 Rectangle Area II （如果有重复部分，思路和此题打碎矩形去掉重复部分的思路一样）
LC497 Random Point in Non-overlapping Rectangles （没有重复部分的原题）
LC528 Random Pick with Weight （类似题目）

对于有多个矩形的情况，我们可以考虑先选出一个矩形，再在该矩形内选点
要求每个点选取的概率相同，那么选取一个矩形的概率就和该矩形的面积成正比
所以我们可以把所有矩形的面积的和sum算出来，然后在rand一个数%sum，再判断落在哪个矩形里
比如说有面积为3, 2, 1, 4的矩形，总和是10，randM = rand() % 10 ，如果randM==0,1,2就选面积为3的矩形，randM == 3,4就选面积为2的矩形，其他的同理
选出了矩形，然后在该矩形内选点即可

~~10. LC853 car fleet~~问题 高频 6次

多辆车再单行路上开，给起始地点和车速，不能超车只能位置重合跟着。问最后有几次的重合车次撞线。
思路：
TreeMap
用treeMap<position, time>去存，从接近target的地方往后遍历，local variable存最大撞线时间。若currTime > maxTime，则车次++

Sort
计算每个位置的车到destination的时间，然后根据位置把车排序，从后往前scan排序后的car数组，如果cars[i]的到达时间比cars[i-1]要晚，说明可以合并cars[i-1]和cars[i]，同时跟新car[i-1]的到达时间

~~11. LC857~~ 雇工人 高频 6次

思路：用wage / quality进行sort，array同时存quality，然后遍历时记录offer和pop出的quality sum。因为是从最实惠的人开始遍历，所以是完备的。pq size为K时，更新min (prev, worker.ratio * quality_sum)

12. LC750 Corner Rectangle个数 高频 5次

思路：任选两行for for loop，若相对应列（第三个for）都有点，就count++，然后对这两行的count任取两个组合，加到result上

13. LC815 Bus Route 高频 5次

每个公交车有多个站，给一堆公交车，问A到B点最少换乘次数
思路：BFS + mem。map<stop, List<bus>>，然后用Set<bus>存visited bus（不能用stop查重，会很慢）。每次层序遍历poll出的站都是在这个step中所有能走到的站

14. LC659 Split Array into Consecutive Subsequences 高频 5次

判断一个升序含重复元素的array是否能分成多个三个数字以上构成的顺子

思路：
用freq map先过一遍存频率，再建一个map存我们能用到的tail number。再过第二遍的时候，若freq==0 continue；若能接上前面的顺子，就接；不能则新开一个顺子（记住新开时候直接要把连着的两个数字剔除，因为要保证长度为三）；都不行则为false。记住最后别忘了更新当前频率

对于每一个element，我们有两种选择
1. 把它加入之前构造好的顺子中
2. 用它新开一个顺子

此处用贪心策略，如果1能满足总是先满足1，因为新开顺子可能失败，即使新开顺子成功，当1能满足的时候，将新开顺子加入之前的顺子也能成功，所以能够选择策略1的时候没必要冒风险选择策略2

目标是用策略1或者2消耗掉所有的元素

如果两个策略都无法选择，直接返回false

用另一个map记录已经构造好的顺子中现在需要哪些尾巴，来实现将当前元素加入构造好的顺子中

15. 王位继承 高频 5次

原帖
void birth(String parent, String name) 父亲名字和孩子名字，生个娃
void death(String name) 此人要死
List<String> getOrder() 返回当前的继承顺序，string array/list

讨论得知，每个人的名字是唯一的，继承顺序符合如下规律:
假设王有大皇子二皇子三皇子，大皇子有长子次子三子，那么继承顺序是王->大皇子->大皇子长子->大皇子次子->大皇子三子->二皇子->三皇子
死掉的人不能出现在继承顺序里，但是如果上面例子中大皇子死了，只需把大皇子移除，原始继承顺序保持不变：王->大皇子长子->大皇子次子->大皇子三子->二皇子->三皇子

三个function会被反复调用，实现function细节。
思路：看起来不难的设计题，DFS只查最左枝

参考代码：

16. LC951 Tree Isomorphism Problem 树的同构问题 高频 5次

这题是Lc 100?
https://www.geeksforgeeks.org/tree-isomorphism-problem/

这题是 951. Flip Equivalent Binary Trees

思路：简单的DFS

17. N叉树，要求删一些node，返回list of roots 高频 5次

More Detail : 给一个tree有红的node有蓝的node，把红的去掉后剩下一堆零零散散的tree，返回这些tree的node，只要node，不要children，也就是说把这个node的children设置成null然后加到list里。
参数是这个树的root。找到所有的红点然后delete掉，去掉这些红点之后就会把一个tree变成散落的几个tree，然后返回这几个tree的root。直接一个recursive判断一下，如果这个node是红点的话就ignore 掉再去判断这个node的children，如果这个node是蓝点的话，要看这个蓝点的parent是不是个红点，是的话，这个蓝点就是散落的tree中其中一个tree的root。
思路：简单BFS。。

18. 可乐饮料机 高频 5次

有一系列按钮，每个按钮按下去会得到一定体积范围的可乐。先给定一个目标体积范围，问不限制按按钮次数，能否确定一定能得到目标范围内的可乐?
举例：有三个按钮，按下去得到的范围是[100, 120], [200, 240], [400, 410],
假设目标是[100, 110], 那答案是不能。因为按下一，可能得到120体积的可乐，不在目标范围里。
假设目标是[90, 120]，那答案是可以。因为按下一，一定可以得到此范围内的可乐。
假设目标是[300, 360], 那答案是可以，因为按下一再按二，一定可以得到此范围内

假设目标是[310, 360], 那答案是不能，因为按下一再按二，有可能得到300，永远没可能确定得到这个范围内的可乐。
假设目标是[1, 9999999999]，那答案是可以。随便按一个都确定满足此范围。
思路：dfs+memorization从0开始暴力解 一开始[0, 0] 通过bfs、dfs往上加直到出界

```
public static boolean dfs(List<Soda> sodas, int volumeLower, int volumeUpper,
                          int targetLower, int targetUpper, Map<String, Boolean>
memo) {

        Boolean val = memo.get(volumeLower + "-" + volumeUpper);
        if (val != null) {
            return val;
        }

        if (volumeLower >= targetLower && volumeUpper <= targetUpper) {
            return true;
        }这里反了吗？ 应该要找的范围在你当前的范围里面吧?


        if (volumeUpper > targetUpper) {
            return false;
        }

        for (Soda soda : sodas) {
            if (dfs(sodas, volumeLower + soda.lower, volumeUpper + soda.upper,
targetLower, targetUpper, memo)) {
                memo.put(volumeLower + "-" + volumeUpper, true);
                return true;
            }
        }

        memo.put(volumeLower + "-" + volumeUpper, false);
        return false;
    }
```

高频 4次

1.下围棋，判断棋盘上点是否被包围 follow up test case 各种形状
请问各种形状是指什么？是不是指棋盘的形状？如果是的话，那么是不是不同点在于不同的形状的边界情况就会不一样?
应该说的是棋子的各种摆放。手动Unit test言之成理即可。主要是要测各种edge case
思路：dfs，碰到空子返回false 没被围死

2。n层map，每层m个node，node和edge都有值，问第一层到最后的minimum cost
思路：遍历+改node值?
什么叫N层map? map是stl里面的map吗？edge存在于同层的node之间？为什么可以改node的值?

3。拿纸牌游戏， 纸牌上面有值，比如说 100， 1， -1， 2， 200， 1. 然后两个人轮流拿，直到拿完。 但是每次只能拿从左边数起的前三个，但是如果你要拿第三个，就必须前两个都拿了，你要拿第二个，就必须第一个也拿了，大家都最优策略，问最后第一个人能拿多少分。
思路：dp存当前人比另一个人能多拿的数，从后往前拿，每次看三个[谁能给个解法链接]
https://www.geeksforgeeks.org/optimal-strategy-for-a-game-dp-31/类似题目

```
  public static void main(String[] args) {


      res = new int[6];
      int[] nums = new int[] { 100, 1, -1, 2, 200, 1 };
      int[][] dp = new int[6][6];
      int n = nums.length;
      for (int l = 1; l < 6; l++) {
         for (int from = 0, to = from + l; to < n; from++, to++) {
            int takeone = from + 1 < n ? nums[from] - dp[from + 1][to] : 0;
            int taketwo = from + 2 < n ? nums[from] + nums[from + 1] - dp[from + 2][to] : 0;
            int takethree = from + 3 < n ? nums[from] + nums[from + 1] + nums[from + 2] - dp[from
+ 3][to] : 0;
            dp[from][to] = Math.max(takeone, Math.max(taketwo, takethree));
         }
      }
      System.out.println(dp[0][5]);


  }
```

我这个貌似不对 应为第二个人默认用后面的最优解了 拿 第一个1 和最后一个1， 或者可以拿1 -1 2 都是2 但是这样会影响第一个人多拿的 有人有想法么 结果是第一个人比第二个人多拿298， 但是其实第二个人如何一次直接那三个第一个人能多拿299

4。image以byte[][]储存 如果想中心镜像翻转怎么弄
思路：跟reverse words一个思路，先翻转每行的byte，再翻转自身（字节翻转可用位运算能快）

5。已知screen的高和宽，给你最小和最大的fontSize，要求给定一个string，将string用竟可能大的fontSize显示在screen里。已知两个API getHeight(int fontSize), getWidth(char c, int fontSize)

，可以得到每个character在不同fontSize下的高和宽。和面试官交流后，确认string可以拆分成几行显示在screen中
思路：先提出暴力解法，然后用二分法优化

6。LC803 打砖块
思路：最好方法从后往前补。先把砖块全都打掉，然后用贴天花板的砖块dfs+mark，然后从后往前一个一个往上加，加同时若碰上周围有mark的砖块就主动dfs，dfs出来的就是这次打掉的砖块

7。LC253 给一堆interval，问最多要定多少间会议室
思路：可以用heap常规做，也可以把开始、结束时间分别升序排序，然后2pointer往后走。复杂度一样，就是更快

8。生成随机迷宫，左上到右下，怎么设计可玩性
思路：直接暴力小人DFS瞎走

9。给一堆intervals和一个时间点，问这个时间点是不是空闲。follow up多call优化时间
思路：做一遍merge intervals再来一遍binary search

10。iterator of iterator
LC类似题目：
LC 341. Flatten Nested List Iterator
LC zigzag iterator
题目描述
Implement an Iterator of Iterators which traverses through an arbitrary number of iterators. IE, an iterator which iterates over three list iterators in the following way: L1 = a1, a2, a3 L2 = b1, b2, b3 L3 = c1, c2, c3 Then the iterator should process them in this order: a1, b1, c1, a2, b2, c2, a3, b3, c3
原题链接：https://www.1point3acres.com/bbs/thread-293238-1-1.html

思路：没什么好说的。。。

频率 3次。

1。LC676 magic dictionary各种变种 所求word再dict单词差一个字母
思路：两种解决思路。可以把words按length存起来然后每有词想search时候遍历查找相符。第二种是存的时候就开始删，记得要记录删的地方的index，最后和所求删的index是否相等
2。LC849 选座位 跟exam room
思路：注意边界条件和怎么判断距离
3。LC418 sentence screen fitting
思路：先把sentence变成空格分割的string，再一个for loop greedy往后排
4。LC68 test justification 把word list转化成等长的行对齐

思路：two pointer [left, right] greedy的把words往上放，中间spaces个数是right-left，注意首行和末尾行的判断。
5。LC844 Backspace string compare
思路：简单stack秒杀
6。 `s = "3[a]2[bc]", return "aaabcbc".`
思路：recursion call括号中间的项
7。LC334 给一个array，`arr[i] < arr[j] < arr[k] given 0 ≤ i < j < k ≤ n-1 else return false.`
思路：从左到右过array，用两个变量存第一小和第二小的数（初始最大值），更新尽量小的数，若同时遇到第三小的数，则为true
8。LC774 加油站最短距离
思路：用binary search寻找最短的距离，边search边找当前mid是否符合mid条件，注意判断边界条件和mid==target怎么走
9。LC337 二叉树House Robber
思路：dfs+dp存抢当前node和不抢当前node的最大值
10。LC340 longest substring with at most K distinct characters
思路：基本滑动窗口问题，遇到就是赚到
11。log start log finish 没太看懂题
有一个Class叫Logger，它有两个函数，一个是LogStart(int logId, int timestamp)，一个是LogFinish(int logId, int timestamp)。Log开始时LogStart会被调用，log结束时LogFinish会被调用。要求是实现这两个函数，并打印已经结束的log，打印log时要按log的开始时间排序。

```
interface Logger {
  void started(long timestamp, String requestId);
  void finished(long timestamp, String requestId);
}
```

started(100, "1")
started(101, "2")
finished(102, "2")
started(103, "3")
finished(104, "1")
finished(105, "3")

Expected Output:
$1 start at 100 end at 104
$2 start at 101 end at 102
$3 start at 103 end at 105

12。LC426 BST撸直变成双向链表 首尾相接

思路：简单DFS。想清楚思路！！！开始dummy当prev留住head，最后prev是tail。其中prev可当做class变量
13。LC215 Kth largest element in an array
思路：可以先用优先队列装个怂，再用quick select
14。LC312 扎气球游戏
思路：二维dp问题。dp[left][right]代表能在当前段内能扎出来的最高分。memorize是当dp非零则是没计算过。
15。LC769 LC768 问一个array在怎样trunk sorted之后只经过拼接就能得到升序array
思路：[0~n]的array做法为maintain一个max变量存当前max，当max==当前index则count++
无限制array时候做法为构造两个新的array存maxOfLeft和minOfRight。当一个数左看都比自己小，右看都比自己大的时候，则可以trunk。（这个更加generalize）
16。LC505 the maze II 求total steps
思路：用BestFS+PQ+memorization做，注意撞墙别忘了往回退一步
17。LC96 Unique Binary Search Trees
思路：递归+memorization
18。LC834 Sum Of distances in tree
思路：两次遍历，更新count和res。第一次post order 第二次pre order

***
给一堆坐标点，求坐标点形成的横平竖直矩形最小面积
思路：任取两点当对角线做矩形，存在set里。若已有set存在则因为另外一个对角线存在，更新面积
***
LC230: Kth smallest in BST，followup 若要改树怎么整 [改树是什么意思？有很多树call这个函数多次？]如果中间有人要insert node to BST的话，如何实现同样功能
思路：建一个TreeNodeWithCount，这样以后改的时候也是log n复杂度。改树的同时改TreeNodeWithCount

4. 面试官迟到 15 分钟。面试时间实际为 30 分钟。给定一个 picture (二维)。里面有一些有色的像素，保证所有像素是相连的（上下左右相连），且只有一个联通块。返回一个最小矩阵，这个矩阵能包含所有的有色 pixel。
这题是找上下左右的极大和极小值吗
是的 LC302原题，用二分查找做的，上下左右分别二分找边界

5. 给定一个棋盘，里面有一些棋子。你能移走这个棋子，当且仅当这个棋子的同行同列有其它棋子。要求最多能移走多少棋子。类似LC 947
思路：可以把所有棋子放到list里，每row，col存在的棋子再分别放到set of set of nodes里。用dfs思路第一轮删除任意一点，然后往后推第二次在上一次基础上清除任意满足要求的那个点，直到最后无点可清除时回溯看总共清除了多少。可mem
更新思路：用union find看能有多少组划分出来（如果同行或同列分成一组），然后最多能移走的棋子数=总棋子-组数（number of islands）
follow up：是应该用什么顺序拿，才能保证能拿最多 (这个follow up应该怎么解呢)尽量先把一个component里的都去掉?

Partition a sequence of *n* tasks into *k* days to minimize the amount of resources used per day.

- Tasks must be executed in sequence.
- You can use *k* days or fewer to complete all tasks.
- Resource usage per day does not change, regardless of the tasks that are selected.

Example:
[2, 3, 5, 2, 6, 5]

[2, 3], [5], [2], [6], [5] → 6
[2], [3], [5], [2, 6], [5] → 8

[2, 3, 5], [2, 6], [5] → 10

recurrence relation: A[n][k] = min ({
max(A*[k-1], sum(S[i+1], S[i+2], ..., S[n]))*
for i = k-1, k, ..., n-1}) 其中S表示task resource的数组，A[n][k]表示n个tasks, k days的每天最小resource值。
这题其实就是(lc410)

国人。 一个只有正整数的list， 其中插入+， * 或者（），求得到式子最大的值。 e.g. [1，2，1，2]-> (1+2)*(1+2)=9. dp解， follow up， 如果有负数该怎么办， 如果想要拿到最大的式子该怎么办。
思路：类似burst balloon dp[i][j] = max of for (k : i ~ j max(dp[i][k - 1] * dp[k][j], dp[i][k - 1] + dp[k][j]))

LC739 array of temperatures, tell me how many days have to wait till next warmer weather
思路：用stack从后往前存，每次看天气时候pop出比栈顶温度低的日子，再peek出比当前暖和的一天index

LC731 持续加intervals，问会不会出现triple booked
思路：按start end加到treemap里，start+1， end - 1，每次从小到大遍历treemap看是否存在count>2

LC736 非常难 parse lisp expression 注意边界条件和判断条件
思路：分情况 let mult add讨论，用一个parse function处理运算符以后的事宜

LC768 乱序数组 chunk出最多组使得每组sort后整个sort好
思路：从右往左扫一遍最小值，从左到右扫一遍最大值。在第二遍途中若看到左最大<=右最小时++

Jan 5

给数组照相

第二轮，国人小姐姐，进来的时候刚好听到我对三姐说have a good night, 用中文说让我别紧张。。我顿时感动的啊，我面试还没见过中国人。。后来英文面的，是面鲸有过的题，给一个数组take snapshot，每次take snapshot后存一下当时的数组状况。后面会访问某一次snapshot时数组的值。

构造一棵size为n的随机

第三轮，三哥，题目就一句话，让我先随机构造一颗size为N的树。我开始没听到“random”一词，就以为构成一颗树就行。秉着论坛经验贴说的，要提前考虑各种条件。我就开始问他，有木有其他限制，比如形状有木有要求，普通的树还是二叉树，N<=0怎么办等等。。。他就说没有，只要size是N就行了
原帖

思路1：
先构造一张全联通的顶点数目为n的图，然后随机选取n-1条边，n-1条边需要联通所有顶点且不能有loop

参考代码

```
private class Edge {
      int from;
      int to;
      public Edge(int frome, int to) {
            this.from = from;
            this.to = to;
      }
}
public Map<Integer, Set<Integer>> construct(int N) {
      Random rand = new Random();
      Map<Integer, Set<Integer>> tree = new HashMap<>();
      List<Edge> allEdges = new ArrayList<>();
      for(int i = 1 ; i<= N ; i++) {
            tree.put(i, new HashSet());
            for(int j = 1 ; j <= N ; j++) {
                  if(j != i) allEdges.add(new Edge(i, j));
```

```
            }
        }
        Set<Integer> inTree = new HashSet<>();
        while (inTree.size() < N) {
            int idx = rand.nextInt(allEdges.size());
            Edge e = allEdges.get(idx);
            if(inTree.add(e.to) || inTree.add(e.from)) {
                tree.get(e.from).add(e.to);
                tree.get(e.to).add(e.from);
            }
            allEdges.remove(idx);
        }
        return tree;
}
```

时间复杂度：worse case: O(n(n-1)!) bese case: O(n^3)

思路2：先构造随机的Prüfer sequence，然后根据sequence生成树

参考代码

N-ary Tree的最长路径

第四轮：国人小哥，n-ary tree里面的最长路径。

Jan 4, 2019

求从start能否所有路径到end

给一个state machine的图，每个节点是一个state，问某一个节点是不是唯一可以走到success state。可以有多条路，但是必须都通向唯一的success state。 dfs秒了，优化到O(N)，大哥让我找找更快的方法，我想了半天硬着头皮说 worst case还是要访问所有节点, 真的不知道还有什么方法了, 可以hint吗？大哥很开心的告诉我对的没有更快的方法了。然后还剩十分钟写了个Top K。
[原帖]

求两个subarray的差值

第三轮 白人小哥。套了个外壳，不过题目大意是，有一个数组，分成两个subarray，求所有的两个subarray的差值。注意是所有的，不是求最小差值。做题前问问题clear了一些条件，subarray可以是empty的，元素是1-10000的int，想起来了再补充。
补充：把原array分成两个之后，求两个subarray中的元素的和的差值 math.abs(sum(sub1) - sum(sub2))
原帖

删除相同字母的pair

第四轮 白人大叔（老爷爷），给一个string，相邻两个字母如果分别是同个字母的大小写，就删掉这个pair。example： aBbA 返回aA. 然后follow up是可以递归消除 aBbA 返回“”. 然后又扯了一个big integer的实现什么的没说完就到时间了。老爷爷人很好，一直笑眯眯的，后来我没写完他跟我说这是你的homework。
原帖

隔段时间统计player的得分

第五轮 白人小哥。 类似于一个设计题。有一个游戏，每个player做不同的任务有不同得分，每三十秒统计一次得分最高的人，整个游戏结束后返回在任意三十秒得分最高的<player，得分, 开始时间>的一个tuple。这轮的小哥全程低头敲代码记录，不管问什么都是好好好对对对up to you。最后我写完了以后，他问我怎么优化，然后扯了cpu cache，大哥讲兴奋了在纸上写写画画给我讲了一堆cpu 优化的事儿，想不起来一个名词非要查清楚告诉我，最后还跟我说别紧张这些都不是面试内容，就是希望我会这些就pretty cool。
原帖

Jan 3, 2019

原帖
题目是 leetcode 317 shortest distance from all buildings. 略微有些改动， 中间加了些 blocks 用2表示。我当天一面完晚上就在 leetcodes上碰到这道题了，不知道算不算不幸

给一个grid，里面有0（表示空地，可以通过），1（表示building，不能通过），2（block，不能通过），现在需要在一块空地上建造一个house，需要house到所有的1距离和最短

思路：
每遇到一个1，我们为它跑一次dijkstra算法得到每一块可以到达该building的空地距离它的最短距离

同时我们需要统计每块空地能到达的building的数量，我们只关心能到达所有building的空地

图中的边不带权值，dijkstra算法的队列可以用普通队列实现

Time Complexity: O(kn)，k为building数量，n为格子数量

Dec 25, 2018

找list中有给定prefix的所有String

给一个 String 类型的list都小写 ， 然后还有一个String 类型的 prefix也小写， 要求是返回一个list 找出在给定list中所有以这个prefix开始的所有String
Example：
list : ["word","work","apple"]
prefix "w"
结果就是返回 一个list，里面有"word" 和 "work"
然后分别用了暴力解法和trie，分析时间复杂度。
思路:

1. 暴力解法 : 直接用startsWith挨个匹配
2. DFS + Trie

Dec 13

RLEIterator，

国人大哥，上来直接做题
热身题：
What's the difference between s.equals("abc") and "abc".equals(s). 没指名任何方向，想到什么说什么。

思路: (Yunwen Zhu：好像是s为null的话，一个exception,一个false)

第二题：
类似LC 900，next只用返回下一个element就可以了，还需要实现hasNext.

热身题目:
String.equals -> compare reference -> compare length -> compare chars one by one
“abc” is in the constant pool.

RLEIterator 变种
Code

```
class RLEIterator {
```

```
	private int[] A;
	private int p;
	public RLEIterator(int[] A) {
		this.A = A;
		p = 0;
	}
	public int next() {
		while (p < A.length - 1 && A[p] <= 0) {
			p += 2;
		}
		if(p >= A.length - 1) return -1;
		A[p]--;
		return A[p - 1];
	}
	public boolean hasNext() {
		while (p < A.length - 1 && A[p] <= 0) {
			p += 2;
		}
		if(p < A.length - 1) return true;
		else return false;
	}
}
```

LC Basic Calculator II

面试官是个印度姐姐，感觉她有点笨拙（或者是认真仔细），我们bb了很多没用的，所以最后只有题目，没有follow up question

题目很简单，输出string “5*3+10” 的运算结果，我是用stack做的

我觉得我解法还是挺常规的，但是不知道为什么她好像有些细节处理的地方不是很明白，稍微浪费了一点时间

Dec 12, 2018

N行诗的所有rhyme组合

一首N行的诗的所有RHYME的组合。N=1， 韵脚A = B =C =。。。输出{A}就可以， N=2 输出{AA，AB}。。当时用N基于N-1的类似BFS的算法做的

我是这么做的，每次新加一个字母有两种选择： 1.从已经出现的字母里面随意选一个 2. 从没有出现的字母里面选一个字典序列最小的
时间复杂度 O(n * B(n))，其中 B(n)是Bell number， 这块是面试官给了提示的。

就一个字母一个字母的加，直到长度为n，每次加字母的原则就我说的那俩个，可以用一个hashset记录已经出现过的字母。实现就是通常的bakctracking。
比如n = 3，过程是这样的
A => AA => AAA,
A=> AA=>AAB,
A => AB=>ABA,
A=> AB=>ABB,
A=>AB=>ABC

Code

```
	Set<String> findRhyme(int n) {
		Set<String> res = new HashSet<>();
		findUtil(n, 0, (char)('A'-1), new StringBuilder(), res);
		return res;
	}

	void findUtil(int n, int curr, char max, StringBuilder sb,
Set<String> res) {
		if (curr >= n) {
			res.add(sb.toString());
			return;
		}
		for(int i = 0 ; (char)('A' + i) <= max ; i++) {
			sb.append((char)('A' + i));
			findUtil(n, curr + 1, max, sb, res);
			sb.deleteCharAt(sb.length() - 1);
		}
		max = (char)(max + 1);
		if (max > 'Z') return;
		sb.append(max);
		findUtil(n, curr + 1, max, sb, res);
		sb.deleteCharAt(sb.length() - 1);
	}
```

Dec 11

Morris 遍历

Code

```
void inOrder(TreeNode root) {
      if(root == null) return;
      TreeNode curr = root;
      while (curr != null) {
            if(curr.left != null) {
                  TreeNode predecessor = findPred(curr);
                  if(predecessor.right == null) {
                        predeccessor.right = curr;
                        curr = curr.left;
                  } else {
                        predecessor.right = null;
                        System.out.print(curr.key);
                        curr = curr.right;
                  }
            } else {
                  System.out.println(curr.key);
                  curr = curr.right;
            }
      }
}
TreeNode findPred(TreeNode root) {
      TreeNode curr = root.left;
      while (curr.right != null && curr.right  != root) {
            curr = curr.right;
      }
      return curr;
}
```

Dec 9

计算等式中x的值

LC 640 Solve The Equation

给定input是一个valid的str，一次方程，只有x，数字，space, +, -, =, 比如“ x + 21 - x = 12 - x”，计算x的值。
followup是加括号怎么做。

Idea:

1. use co to record the total coefficients of x
2. use sum to track the total sum of numbers
3. Recursion to solve the brackets

Code (带括号)

```
// return: res[0]: sum of numbers, res[1]: coefficients
private int[] calculate(char[] exp, int[] index) {
        int[] res = new int[2];
        int sign = 1;
        int num = 0;
        while (index[0] < exp.length) {
            if (Character.isDigit(exp[index[0]])) {
                num = num * 10 + (exp[index[0]] - '0');
                index[0]++;
            } else if (exp[index[0]] == '+' || exp[index[0]] == '-') {
                res[0] += sign * num;
                if (exp[index[0]] == '+')
                    sign = 1;
                else
                    sign = -1;
                num = 0;
                index[0]++;
            } else if (exp[index[0]] == 'x') {
                if (index[0] == 0 || !Character.isDigit(exp[index[0] - 1])) {
                    res[1] += sign * 1;
                } else {
                    res[1] += sign * num;
                }
                num = 0;
                sign = 1;
                index[0]++;
            } else if (exp[index[0]] == '(') {
                index[0]++;
                int[] tmp = calculate(exp, index);
                res[0] += sign * tmp[0];
                res[1] += sign * tmp[1];
                  num = 0;
                  sign = 1;
            } else if (exp[index[0]] == ')') {
```

```
                    index[0]++;
                    res[0] += num * sign;
                    return res;
                }
            }
            res[0] += num * sign;
            return res;
        }
```

Nov 28

Number of Island和limiter

第一面很简单，利口200，1和0换成圈圈叉叉，经典面筋。第二面先问了两个BQ，然后问了道OOD。设计一个limiter，input是event的个数和cooldown时间，每次调用来查询还有没有event在继续。

有些同学问limiter的问题，是这样的：

```
补充内容 (2018-11-30 05:45):
limiter = new MyLimiter(2, 60)
limiter.MayContinue() // true
limiter.MayContinue() // true
limiter.MayContinue() // false
sleep(60)
limiter.MayContinue() // true
sleep(30)
limiter.MayContinue()//true

补充内容 (2018-11-30 05:47):
这是面试官给的例子，用系统时间，不用实现sleep()。
希望能帮到大家～～
/*
      start
      0
*/
class MyLimiter {
      private int events;
      private int coolDown;
      private long startCoolDown;
      private int currEvents;
```

```
	public MyLimiter(int events, int coolDown) {
		this.events = events;
		this.coolDown = coolDown;
		this.currEvents = events;
	}

	public boolean MayContinue() {
		if (currEvents > 0) {
			currEvents--;
			if (currEvents <= 0) {
				startCoolDown = System.currentTimeMillis();
			}
			return true;
		} else {
			long now = System.currentTimeMillis();
			if (now - startCoolDown < (long) coolDown * 1000) {
				return false;
			} else {
				currEvents = events;
				currEvents--;
				if (currEvents <= 0) {
					startCoolDown = System.currentTimeMillis();
				}
				return true;
			}
		}
	}
}
```

Nov 25

LC 128 Longest Consecutive Sequence

题：
给一个array，全是int，求找出最长的sequence的长度，longest consecutive sequence。
例：[7，2，4，3，5]， return 4，因为最长的sequence是2345

分享一个小idea： 每次看到题目给一个list，然后答案不需要index的时候，想想能不能sort做。我第一想法就是这道题sort之后就很简单了，先sort O(nlogn) + 走一遍找最大 O(n) + space O(1)， 然后和面试官说sort可以做但是肯定还有更优解，面试官打断我说你先把这个sort写出来，然后we go from there。

写出来花了15分钟，包括一开始的一些clearification花费的时间。

然后就觉得应该用hashmap，能比nlogn更快的我当时只想到hashmap，卡住了2-5分钟，面试官说你的方向是对的，但能不能用set呢，我说能，然后准备每找到一个数，就从那个数开始，把set里的比他大的数删掉，结果发现不行，面试官提示我你可以删大的和小的都删，然后花了10分钟写出来了，最后问我你有没有什么问题要问我的，我瞎扯了一通。然后时间就用完了。

删比这个数小的和比这个数大的我分成了两个function，一个其实就够了，但我和面试官说要increase readability，故意分成两个的，不是我不会，他说可以。最后面试完我问他您觉得我这个solution 可以吗？他说他觉得不错

Nov 10

翻转字符串到目标字符串(Isomorphic Strings变种)

> 题目很多人在面经里贴过：给定一个字符串s，问能不能转化成另一个字符串p，条件是每次转换要把所有相同的字母一起变动，例子如: abca(两个a一起变) -> dbcd(变c) -> dbed(变b) -> dced。所以abca能转化成dced，return true。
> 我目前只能想到：
> 首先必须原来s中字符相等的几个位置，在p中都要被转化成同一个字符，否则 return false
> 请大家指点一下思路，或者说下证明过程，谢谢啦！

Idea:

如果有环，需要用中间tmp字符解开环后再变换
a->b->c->a需要变成a->b->c, c->k->a，这样解开字母之间的相互影响

如果tmp字符可以重复利用，一个tmp可以解开所有环，如果不能，一个环需要一个tmp字符

Code

```
private boolean isomorphic(String s, String p) {
    if(s == null) return p == null;
    if(p == null) return s == null;
    if(s.length() != p.length()) return false;
    int tmpNum = 26;
    char[] str = s.toCharArray();
    char[] pat = p.toCharArray();
    Map<Character, Character> map = new HashMap<>();
    for(int i = 0 ; i < str.length ; i++) {
        if(map.containsKey(str[i])) {
            if(map.get(str[i]) != pat[i]) return false;
```

```
                } else {
                        map.put(str[i], pat[i]);
                        tmpNum--;
                }
        }
        if(tmpNum > 0) return true;
        boolean[] visited = new boolean[26];
        for(Character key: map.keySet()) {
                if(!visited[key - 'a']) {
                        Character curr = key;
                        while (!visited[curr - 'a']) {
                                visited[curr - 'a'] = true;
                                curr = map.get(curr);
                                if(curr == null) break;
                        }
                        if(curr != null) return false;
                }
        }
        return true;
}
```

5-5
给候选人投票，票有时间点和候选人。问给定时间点选winner（不频繁调用）LC911 Online Election
Follow up：1. Top 1 2. Top k 3. 给top k求时间
思路：1. 前两个用hash table + 优先队列
2. 第三问（个人思路）用链表bucket存得票数，sort选票，遍历往链表上加。每次从后往前遍历k个查看是否符合要求（n*k）

7-25
LC60反向 给一个[2,1,3] 数字1-n，无重复，valid排序 返回他是排序中的第几个?
思路：第n位数字确定后的全排有(n-1)!个，所以题中百位2开头就说明1开头的都在前头，而1开头的数字有2个，以此类推到十位等等

7-25
1。LC20 valid parentheses
2。LC801 给两个等长array，assume互换位置后能全都递增，问最少换几次
思路：1. Stack 2. Dp记录当前节点换和不换的最少次数，注意条件别漏写

4-22
1。给double[]和target，用’+’ ’-’连接成target，输出可以得到target的所有方法
2。给string pairs [dogs, are], [cats, are], [are, cute], return “dogs cats are cute” 无环

思路：1. DFS + backtracking, 2^n 复杂度 每个空格可插加、减（memorization？怎么做）
3. LC210 拓扑排序 先构建dependency map，再DFS或BFS遍历 （DFS从头到尾各元素做DFS，途中碰到环false，BFS先扫一遍选出0dependency的种子点、每次减依赖、遍历后发现没全遍历则false）

中国/韩国小姐姐（全程期待中文脸，然而小姐姐全程英文，也许不是中国人？）
问我玩过candy crash没有，然后给我解释了一下游戏怎么玩（是消消乐吧？）
题目是candy crush游戏：一个board，每个格子里都要放随机的颜色。如果连续三个格子颜色相同，就可以消掉。可以swap相邻格子的颜色，使得出现能消掉的颜
input是int row, int col, int color，m和n是board的长宽，int color是颜色的种类。比如我假设是color = 5的时候，我们可以选0，1，2，3，4这样，小姐姐说可以哇
写一个method生成游戏的开局
要求：
随机生成，不能有地方直接消掉（不能有连续三个相同的颜色），比如:
1 2 3. more info on 1point3acres
1 3 4
1 3 4， 这样第一列就消掉了
follow up：能够走至少一步（必须存在至少一个地方，通过swap能消掉颜色）
这题还蛮好玩的，小姐姐也全程积极一起讨论，还会鼓励笑着跟你说that's prety cool!哈哈哈感谢小姐姐！

2.第二轮面试， 中国小哥， 字符串转换那道面经题，输入是String src, String target, 问你是否可以从src转换到target, 返回boolean. 比如abc -> def，转换方法就是建立映射map， a->d, b->e，c->f转换具体过程就是abc -> dbc -> dec -> def.这是最简单的情况。如果字符串map是有环的，例如, ab -> ba, 其中的环就是a->b->a,这会造成转换失败。ab -> bb ->aa ->bb .... 但是呢你可以借助第三个变量来完成转换，还是ab -> ba, 你可以建立map a->c b->a, c->b, 转换过程就是， ab ->cb -> ca ->ba。所以其实ab -> ba是可以成功转换的，返回true。 所以失败的case就是，如果你没有这样额外的变量c可以借用来解决换的问题了。假设字符集只有a-z， 你abcdef....z -> bcdef...za就不能转换，因为在26个字母中没有额外字母可以用来解环。

第一轮的小姐姐说这题她第一次出
有一个binary tree（不是平衡的） 两个人A和B轮流占领node
规定每一次占领node的时候只能占领自己已经拥有的node相连的node 如果对方无node可占领，则获胜
现在已经知道了A的第一步占领的node A，求B的第一步应该落在哪
follow up是 如果你是A 第一步应该落在哪 讲了思路写了一部分postorder的代码
思路：count subtree of all neighbors, where the max is the start point of player 2

public List<Integer> kthUser(int[][] log, int K){ }
---
input: int[][] log=new int[n][3], log代表user可能在这个interval里任意时间点访问过这个网站:
@log[0]: user id;

@log[1]: start time;

@log[2]: end time;

问第k个访问网站的用户可能有哪些，返回他们的user id，顺序随意；
eg. log=[[1,20,30],[2,0,50],[3,45,70],[4,35,55]],
if(K==1)---->return res=[1,2];
if(K==2)---->return res=[1,2,3,4];
if(K==4)---->return res=[2,3,4];
思路：用enddatesort，取第K个寻找与这个interval有intersection的就是

public int NumberOfDistinctIsland(int[][] matrix){ }

LC 711 Number of Distinct Island II
给一个只包含0、1的matrix，0为水，1为岛，岛屿为四联通区域。返回distinct的岛屿的数目。
如果一个岛屿通过上移、下移、左移、右移可以和另一个岛屿完全重合，它们被认定为一个岛屿；Eg.
if matrix=={
[1, 1, 1, 0, 0, 0],
[1, 1, 0, 0, 0, 0],
[0, 0, 0, 1, 0, 0],
[1, 1, 1, 0, 0, 0],
[1, 1, 0, 0, 1, 1]}

return 3;
左上角左下角岛屿相同，被判定为一个岛屿，matrix[2][3]为一个岛屿，右下角还有一个岛屿，共三个
思路：将岛屿用dfs遍历方式编码去重

在一个平面上有很多的矩形，有的有overlap，有的没有，现在已知有个算法可以随机产生一个在矩形覆盖范围内的点，然后让你设计算法来验证这个算法产生的点在矩形覆盖范围中确实是随机分布的，而不会集中在某个矩形中，或者只分布在矩形的一个角上。

__L_R__ 题目是一个一维的棋盘，上面有l和r两种棋子，l只能往左走，r只能往右走，不能跨过其他棋子，下划线代表空格。给初始和最终的两个state作为input，输出一个boolean，判断第二个state是否可以由第一个state通过若干操作达成

b. Follow up，棋子走到边界会消失

思路：LC777 双指针两边一起过LR，忽略空格。判断位置是否合理。follow up用subarray方法判断。

挂的是这轮，要找长度为50的bar code, bar code是由黑白两色构成，第一个和最后一个只能是白色，且最多连续三白和连续三黑，问有多少种符合条件的 bar code.

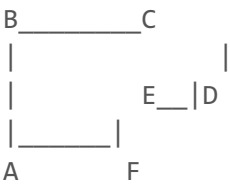
维护一个50×6的dp。50是barcode的长度，6只一共只有六种结尾的状态，即：1白， 2连白， 3连白， 1黑， 2连黑， 3连黑。. 1point3acres

```
int Solution::google(void) {
    vector<vector<int>> dp (6, vector<int> (50, 0));
    dp[0][0] = 1;

    int mod = 1e9 + 7;
    for (size_t i = 1; i < dp[0].size(); ++i) {
        // end with 1 while
        dp[0][i] = (dp[3][i - 1] + dp[4][i - 1] + dp[5][i - 1]) % mod;
        // end with 2 while
        dp[1][i] = (dp[0][i - 1]) % mod;
        // end with 3 while
        dp[2][i] = (dp[1][i - 1]) % mod;

        // end with 1 black
        dp[3][i] = (dp[0][i - 1] + dp[1][i - 1] + dp[2][i - 1]) % mod;
        // end with 2 black
        dp[4][i] = (dp[3][i - 1]) % mod;
        // end with 3 black
        dp[5][i] = (dp[4][i - 1]) % mod;
    }

    return dp[0].back() + dp[1].back() + dp[2].back();
}
```

给一个场景，比如Gmail的每个用户（如果没有自己设置头像）就会自动分配一个头像，有着名字首字母和一个特定的背景颜色。问题是给定一个List<Color> colors，给一个List<String> names，给每个名字分配一个颜色。分配规则是：相同的人必须有相同的颜色；相邻的名字要有不同的颜色。Followup：再设计一个函数，使得输入一个名字，就能输出一个Color，但除了List<Color> colors之外，不能有其他的global variables （就是说不能有个global 的map来存映射关系了）。比较两种方法的pros and cons。

思路：相邻name建无相连接图，暴力bfs着色

给一个List<int[]> 存放一大堆坐标。坐标用来描述一副画好的map。坐标依次为 最左下方的顶点，沿边顺时针各顶点。比如：



再给一个坐标（X,Y）判断坐标是不是在闭合的图内。（坐标假设valid 即永远可以构成一个闭合图形，且边要么竖直要么水平 没有斜边）
思路：可以选择一个方向 水平or 竖直 然后向一个方向做射线，如果交过两条边，那么这个点就在图外（偶数都是图外，可以自己画图试试）

给你一个array，每个数字代表工作量。再给一个K，表示最多几天做完。要minimize 每天工作量的max。比如说给你[1,2,3],要一天内做完，那么答案就是6，要两天内做完的话，可以第一天做1，第二天做5，也可以第一天做3，第二天做3。这种情况下output是min{max{1, 5}, max{3, 3}} = 3
思路：1. Binary search找res，然后遍历validate 2. 二维dp[i][j] i：用前i个工作j天内做完要多少工作量 recursion rule左大段右小段 整一个helper prefix sum

0 0 0
0 1 1
0 3 2
0 6 3

先自我介绍问了几分钟简历。面筋题。判断target字符串是否可以由给定字符串转换得到。转换的规则是：每次转换要变所有的相同字母
比如：abca -> cdec 可以 abca -> abea -> cbec -> cdec
这道题当时看面经的时候就觉得很怪，也没仔细想，加上和第一轮又很像，当时有点慌，讨论了蛮久，还好面试官super nice
最后讨论的结果是很简单，只要check对应的映射规则都满足就行了，当然长度一定要相同。
然后问什么时候需要中间变量？比如ab -> ba，必须先ab -> cb -> ca -> ac，不能直接a到b，b到a。讨论了一下lz说check有没有环，面试官满意。然后写了个dfs检查有没有环，这一轮就结束了。

01矩阵走路问题
0和1的grid，1是墙，0是路，从左上角走到右下角，最少多少步。BFS
Follow-up: 现在说能把grid中的一个1变成0，问新的最小步数是多少步
思路：Bi-BFS算左上，右下到每个1的点距离和的最小值

一个白人小哥哥，给你一个source string, 要求找长度为K的字母排序最前面的subsequence，一开始思路不明确，先随便讨论了一下brute force的解法，时间复杂度2^n * K，问能不能提高，想出了N*K的解法，写了代码。继续问对于最差的Corner case怎么提高，用上了heap，继续改了代码。follow-up问如果要top 10 subsequence
思路：1. 初始化char array，greedy每次选最靠左最小的char，同时保证后面的subsequence够长 2. 感觉只能brutal force了 recursion + backtracking + memo

LC611给一堆三角形的边长，问能构成多少个三角形
思路：确定最长边然后two sum

5. 股票，貌似是面经题，对一支股票实现价格的添加，删除，修改，查询最高价格，查询当前价格
LC679 follow up: construct result list

第四轮: 无限大的棋盘里马找从起点到底终点的路径(BFS) Follow Up: 如果有有限个Obstacles，如何判断是否可以从起点到达终点（Bi-BFS）
Follow up: 由于有可能block所有从起点到终点的路径，所以两遍一起同时搜索，若有一个queue没了则停止