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纠正，Doc每天都有人更新，会同步大家的有用修改

（已更新最近几个月的题 请踊跃用comment提问）

为了统计·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，可以按照格式添加到后面的日期下面（日期为一级标题，二级标题，题目）

1.自行车和人匹配问题 (高频 22+次)

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
Cindy Cheung: 感覺跟這題有點像 https://www.careercup.com/question?id=5687196447670272

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
Time Complexity:O(row*col + mn log(mn))
按照这个思路写了Java代码，欢迎指正，适用于仅要求车-人最近距离匹配的情况：

```
public class MyClass {
    public static void main(String[] args) {
        char[][] grid = {{'p', 'o', 'p', 'o', 'o'},
                         {'o', 'o', 'o', 'o', 'o'},
                         {'b', 'o', 'o', 'o', 'b'},
                         {'b', 'o', 'o', 'o', 'p'}
        };

        List<Location[]> results = assignBikes(grid);
        for (Location[] cur : results) {
            System.out.println(cur[0].x + " " + cur[0].y + " " + cur[1].x + " " + cur[1].y);
```

```
    }
  }

  public static class Location {
    int x;
    int y;

    public Location (int _x, int _y) {
      this.x = _x;
      this.y = _y;
    }
  }

  public static class PersonAndBike {
    public Location person;
    public Location bike;
    public int distance;


    public PersonAndBike (Location newPerson, Location newBike) {
      this.bike = newBike;
      this.person = newPerson;
      this.distance = Math.abs(person.x - bike.x) + Math.abs(person.y - bike.y);
    }
  }

  public static List<Location[]> assignBikes(char[][] grid) {
    List<Location[]> results = new ArrayList<Location[]>();
    //input check
    if (grid == null || grid.length == 0 || grid[0].length == 0) {
      return results;
    }

    List<Location> people = new ArrayList<Location>();
    List<Location> bikes = new ArrayList<Location>();
    for (int i = 0; i < grid.length; i++) {
      for (int j = 0; j < grid[0].length; j++) {
        if (grid[i][j] == 'p') {
          people.add(new Location(i, j));
        } else if (grid[i][j] == 'b') {
          bikes.add(new Location(i, j));
        }
      }
    }

    PriorityQueue<PersonAndBike> pq = new PriorityQueue<PersonAndBike>(new
Comparator<PersonAndBike>() {
      public int compare(PersonAndBike e1, PersonAndBike e2) {
```

```
                return e1.distance - e2.distance;
            }
        });

        for (Location person : people) {
            for (Location bike : bikes) {
                PersonAndBike pair = new PersonAndBike(person, bike);
                pq.offer(pair);
            }
        }

        while (!pq.isEmpty()) {
            PersonAndBike curPair = pq.poll();
            if (grid[curPair.person.x][curPair.person.y] == 'p' &&
                grid[curPair.bike.x][curPair.bike.y] == 'b') {
                Location[] result = new Location[2];
                result[0] = new Location(curPair.person.x, curPair.person.y);
                result[1] = new Location(curPair.bike.x, curPair.bike.y);
                results.add(result);

                System.out.println("find a pair!");

                grid[curPair.person.x][curPair.person.y] = '$';
                grid[curPair.bike.x][curPair.bike.y] = '$';
            }
        }

        return results;
    }
}
```

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？lz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 ...33

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 后从小往大排序输出 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

考虑例子：
PO BOOP
OOOOOO
OOOOOO
OOOOOO
BOOOOO
BOOOOO
似乎heap不行？
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 v
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为中心用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
有障碍物就不需要heap了吧？从车开始BFS就行了吧，当然前提是没有tie

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
    b. 发现这里不能使用max flow， 因为这个bipartite is a weighted graph. 参考 https://www.topcoder.com/community/competitive-programming/tutorials/assignment-problem-and-hungarian-algorithm/
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）

原题：一组坐标表示人，另一组表示车，车比人多，给每个人匹配最近的车。其中人和车的距离没有tie。
原题还比较简单，最笨的bfs也可以做，坐标数值很大的时候，时间复杂度可能会很高，稍微好一点的是用pq存所有的人车距离，每次poll最小的距离，如果这个人已经匹配到车了继续poll，直到所有人都匹配到车为止。

follow up版本1: 一组坐标表示人，另一组表示车，车比人多，其中人和车的距离有tie（距离两个人最近的车可能是同一辆），给每个人匹配一辆车，要求所有匹配的人车曼哈顿距离加起来最小（全局最优）。
这一问原题的两种方法基本全部gg，因为要求全局最优并且有tie，于是每个人不一定是匹配到距离自己最近的车子。pq方法完全失效，bfs方法无法保证全局最优（距离一个人最近的车可能有多辆，然而单凭bfs无法确定给此人匹配哪辆可以全局最优）。暴力搜索全部匹配方式，找最小总距离的匹配方式可以确保正确性，但是车和人很多的时候，时间复杂度会很高。目前个人认为这一题面试官的期待做法，应该就是二分图最小带权匹配，KM算法，但是鉴于面试的时候可能很难写出，所以在此希望大家讨论一下有没有其他稍微简单点的办法，因为和正常的二分图匹配不一样，这个已经告诉你那些节点属于哪一边了。
follow up版本2: 一组坐标表示人，另一组表示车，车比人多，其中人和车的距离有tie（距离两个人最近的车可能是同一辆），给每个人匹配一辆车，要求匹配后最大的人车距离最小。
这一问和前面的关系似乎不是很大，不过万能的暴力dfs还是能做，全部匹配方法写出来，找最长距离最小的那个，就是答案，不过和前面一样，没有非常有效的剪枝方法，复杂度很高，所以面试官也不

会满意（我同学面试答了这种方法挂掉了）。感觉可以用dp来做，但是没有想出很好的状态表示和转移方程，希望大家讨论！！！！

KM算法参考代码
Provider: null

```
public class KMAlgorithm {
	@Test
	public void test() {
//		char[][] graph = new char[][] {
//			{'O', 'P', 'O', 'B', 'P', 'O', 'P'},
//			{'O', 'O', 'O', 'O', 'O', 'O', 'O'},
//			{'B', 'O', 'O', 'B', 'O', 'O', 'B'}
//		};
		char[][] graph = new char[][] {
			{'P', 'O', 'O', 'B', 'P', 'O', 'B'}
		};
		System.out.println(match(graph));
	}

	char[][] graph;
	List<int[]> bikes;
	List<int[]> persons;
	int[][] G;
	int[] personExpect;
	int[] bikeExpect;
	boolean[] visitedPerson;
	boolean[] visitedBike;
	int[] match;
	int[] slack;
	//return : person[i, j] + bike[i, j]
	public List<List<Integer>> match(char[][] graph) {
		this.graph = graph;
		init();
		KM();
		List<List<Integer>> res = new ArrayList<>();
		for(int i = 0 ; i < match.length ; i++) {
			if(match[i] == -1) continue;
			int[] person = persons.get(match[i]);
			int[] bike = bikes.get(i);
			res.add(Arrays.asList(person[0], person[1], bike[0], bike[1]));
		}
		return res;

	}
	private void init() {
		int rows = graph.length;
		int cols = graph[0].length;
		persons = new ArrayList<>();
		bikes = new ArrayList<>();
		for(int i = 0 ; i < rows ; i++) {
			for(int j = 0 ; j < cols ; j++) {
				if(graph[i][j] == 'P') {
```

```
                                persons.add(new int[] {i, j});
                        }
                        if(graph[i][j] == 'B') {
                                bikes.add(new int[] {i, j});
                        }
                }
        }
        // build cost graph
        G = new int[persons.size()][bikes.size()];
        for(int i = 0 ; i < G.length ; i++) {
                for(int j = 0 ; j < G[0].length ; j++) {
                        int[] person = persons.get(i);
                        int[] bike = bikes.get(j);
                        int dis = Math.abs(person[0] - bike[0]) + Math.abs(person[1] - bike[1]);
                        G[i][j] = -dis;   // 求最小距离，所以这里权值用负数表示
                }
        }
        personExpect = new int[persons.size()];
        bikeExpect  = new int[bikes.size()];
        visitedPerson = new boolean[persons.size()];
        visitedBike = new boolean[bikes.size()];
        match = new int[bikes.size()];
        Arrays.fill(match, -1);
        slack = new int[bikes.size()];
        Arrays.fill(slack, Integer.MAX_VALUE);

        // init person expectation array
        for(int i = 0 ; i < persons.size() ; i++) {
                personExpect[i] = G[i][0];
                for(int j = 0 ; j < bikes.size() ; j++) {
                        personExpect[i] = Math.max(personExpect[i], G[i][j]);
                }
        }
}
boolean find(int person) {
        visitedPerson[person] = true;
        for(int bike = 0 ; bike < bikes.size() ; bike++) {
                if(visitedBike[bike]) continue;
                int gap = personExpect[person] + bikeExpect[bike] - G[person][bike];
                if(gap == 0) {
                        visitedBike[bike] = true;
                        if(match[bike] == -1 || find(match[bike])) {
                                match[bike] = person;
                                return true;
                        }
                } else {
                        slack[bike] = Math.min(slack[bike], gap);
                }
        }
        return false;
}
void KM() {
        for(int i = 0 ; i < persons.size() ; i++) {
```

```
                        Arrays.fill(slack, Integer.MAX_VALUE);
                        // assign bike for each person
                        while(true) {
                                Arrays.fill(visitedPerson, false);
                                Arrays.fill(visitedBike, false);
                                if(find(i)) break;
                                // if can not assign one bike to the person, decrease the expectation
                                int tmp = Integer.MAX_VALUE;
                                for(int bike = 0; bike < bikes.size() ; bike++) {
                                        if(!visitedBike[bike]) tmp = Math.min(tmp, slack[bike]);
                                }
                                for(int person = 0 ; person < persons.size() ; person++) {
                                        if(visitedPerson[person]) {
                                                personExpect[person] -= tmp;
                                        }
                                }
                                for(int bike = 0 ; bike < bikes.size() ; bike++) {
                                        if(visitedBike[bike]) {
                                                bikeExpect[bike] += tmp;
                                        } else {
                                                slack[bike] -= tmp;
                                        }
                                }
                        }
                }
        }
}
```

2. 二叉树删除边 (高频 13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。

3 invalid situations
case1: 2 parents no circle
case2: 2 parents with circle
case3: 1 parent with circle
2 main steps
1 check whether there exists a node with 2 parents, if yes, store the two edges.
2 if no edge yielded from step 1, apply union-find and find the edge creating cycle (same as 684); ELSE, apply union-find to ALL edges EXCEPT edges from step 1, then check: if edge 1 from step 1 creates a cycle, return edge 1; else return edge 2.

//请问有的题是binary tree删多余边，这时候输入是 root 还是 edge list？

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
provider: null

```
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)

CPP参考代码：
(Provider: Anonym)

```
void isValidBST(TreeNode* root) {
    root=isValidBST(root, nullptr, nullptr);
}
TreeNode* isValidBST(TreeNode* root, TreeNode* minNode, TreeNode* maxNode)
{
      if(!root) return nullptr;
      if(minNode && root->val <= minNode->val || maxNode && root->val >=
maxNode->val)
         return nullptr;
    root->left = isValidBST(root->left, minNode, root);
    root->right = isValidBST(root->right, root, maxNode);
```

```
    return root;
}
```

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
provider: null

```
            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];
            }
```

C++ version:
(Provider: Anonym)

```
      int uniquePath(const vector<vector<int>>& grid) {
          if(grid.empty() or grid[0].empty()) return 0;
          int rows = grid.size(), cols = grid[0].size();
          vector<int> pre(rows, 0), cur(rows, 0);
          pre[0] = 1;
          for(int j = 1 ; j < cols; j++) {//roll on by column
              for(int i = 0; i < rows; i++) {
```

```
                //simply means: cur[i]=pre[i]+pre[i-1]+pre[i+1]
                cur[i] = pre[i] + (i-1>=0? pre[i-1] : 0) + (i+1<rows?
pre[i+1]: 0) ;
            }
            cur.swap(pre);
        }
        return pre[0];
    }
```

滚动数组用位操作写起来会比较清晰，而且可以避免额外的copy

```
int uniquePath(const vector<vector<int>>& grid) {
    if(grid.empty() || grid[0].empty()) return 0;
    int rows = grid.size(), cols = grid[0].size();
    vector<vector<int>> dp(2,  vector<int>(rows, 0));
    pre[0][0] = 1;//(ze yang)应该是dp吧，不过改了还是不对
    int e = 1;
    for(int j = 1 ; j < cols; j++, e ^= 1) {//roll on by column
        for(int i = 0; i < rows; i++) {
           dp[e][i] = dp[e ^ 1][i] + (i-1>=0? dp[e ^ 1][i - 1] : 0) +
(i+1<rows? dp[e ^ 1][i + 1]: 0) ;
        }
    }
    return dp[e ^ 1][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径

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 (lower <= xi <= upper)，则可达

参考代码

provider: null

```
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;// 此处的判断不够严谨；还有一种情况是，两个点在同一列上，但这两个点其实是同一个点，则我们不应该直接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

provider: null

```
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

Provider: null

```
            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 64 最小路径和，思路差不多，只是求和变成求相加后的最大值: LC 64. Minimum Path Sum -> Maximum

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

数组全是零(或者左上角一块是零)的话就没有办法通过正负号判断来的方向了吧，这样在重构path的时候可能会index out of bound，觉得还是check左边和右边哪个更大就是从那边来的更好，然后注意第一行和第一列的特殊情况，这样不会出问题

（这个方法是面试官给的hint提示的，原数组应该都是正整数。如果全是0，用正负号表示也可以特殊处理一下第一行和第一列的情况即可，即遇到i为0时候总是往左走，j为0的时候总是往上走。）

参考代码

Provider: null

```
	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 12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
Jian Sun:这里有个knuth-mastermin-algo 值得参考 link:
https://math.stackexchange.com/questions/1192961/knuths-mastermind-algorithm
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: 见leetcode discussion

当一个单词和其他单词match number为0的次数越多，那么这个单词越不好，因为match number为0时我们减少搜索空间的速度最慢。

假如现在有无限多长度为6的单词，对于word X，和他match number为0的单词有25^6这么多个，然而和X match number为1的单词则减少到了25^5 * 6这么多个，为2时为 C(6, 2) * 25^4，以此类推，match number越大我们下一轮的搜索空间会越小，所以这里我们每一轮都挑选出当前搜索空间中和其他单词match number为0的次数最少的单词作为guess word来猜，这样minimize了每次猜词的worse case。

参考代码：
(Provider: leetcode: lee215)

```
Time Complexity: O(n ^ 2)
Space Complexity: O(n ^ 2);
class Solution {
    public void findSecretWord(String[] wordlist, Master master) {
        List<String> list = new ArrayList<>();
        for(String str: wordlist) list.add(str);
        for(int i = 0 ; i < 10 ; i++) {
```

```
            Map<String, Integer> zeroMatch = new HashMap<>();
            for(String s1: list) {
                zeroMatch.putIfAbsent(s1, 0);
                for(String s2: list) {
                    if(match(s1, s2) == 0) {
                        zeroMatch.put(s1, zeroMatch.get(s1) + 1);
                    }
                }
            }
            Pair pair = new Pair("", 101);  // list size is 100
            for(String key : zeroMatch.keySet()) {
                if(zeroMatch.get(key) < pair.freq) {
                    pair = new Pair(key, zeroMatch.get(key));
                }
            }
            int matchNum = master.guess(pair.key);
            if(matchNum == 6) return;
            List<String> tmp = new ArrayList<>();
            for(String s: list) {
                if(match(s, pair.key) == matchNum) {
                    tmp.add(s);
                }
            }
            list = tmp;
        }
    }
    private static class Pair {
        String key;
        int freq;
        public Pair(String key, int freq) {
            this.key = key;
            this.freq = freq;
        }
    }
    private int match(String s1, String s2) {
        int res = 0;
        for(int i = 0 ; i < s1.length() ; i++) {
            if(s1.charAt(i) == s2.charAt(i)) res++;
        }
        return res;
    }
}
```

Onsite: 限制条件比LC 843多一些，长度是５，都是uppercase, 没有重复字母。先实现guess function, 自己设计数据结构，返回有几个字母match（不需要位置也一样，存在于secret word里

就算）和是否猜中。这个就是基础的字符串比较，主要是注意对word进行validation。然后就是问怎么猜了。就大概说了LC 843里的方法，preprocessing dictionary， 然后根据猜的结果缩小范围。代码没时间写完。

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 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

参考代码：（disscussion 里面看到的代码量少，比较好写的代码）

```
class Solution {
    // 0: up, 1: right, 2: down , 3: left
    public void cleanRoom(Robot robot) {
        dfs(new HashSet<>(), 0, 0, 0, robot);
    }
    private void dfs(Set<String> visited, int i, int j, int currDir,
Robot r) {
        if(visited.contains(i + "," + j)) return;
        visited.add(i + "," + j);
        r.clean();
        for(int k = 0 ; k < 4; k++) {
            if(r.move()) {
                int x = i, y = j;
                switch(currDir) {
                    case 0: x -= 1; // up
                     break;
                    case 1: y += 1; // right
                        break;
                    case 2: x += 1; // down
                        break;
```

```
                    case 3: y -= 1; // left
                        break;
                }
                dfs(visited, x, y, currDir, r);
                r.turnRight();
                r.turnRight();
                r.move();
                r.turnRight();
                r.turnRight();
            }
            r.turnRight();
            currDir += 1;
            currDir %= 4;
        }
    }
}
```

7. LC855 考试找位子，尽量分散坐，人会离开 高频6次

思路：见[leetcode]
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)

1.2 在优先队列外再建一个BST可以把离开的时间优化到logn。离开的时间主要来自于一个线性的遍历，在BST里可以logn找到左右邻居，并且BST有logn时间的query/insert，不会影响坐下的时间复杂度。

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)

3. C++用Set(BST)和HashMap(Unordered_Map) 可以做到离开和坐下都是O(logn)

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）

Provider: null

```
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 synchronized 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

多辆车在单行路上开，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]，同时更新car[i-1]的到达时间
code

```
class Solution {
    public int carFleet(int target, int[] position, int[] speed) {
        int len = position.length;
        Car[] cars = new Car[len];
        for(int i = 0 ; i < len ; i++) {
            cars[i] = new Car(position[i], speed[i], target);
        }
        Arrays.sort(cars, (a, b) -> a.pos - b.pos);
        int carFleet = len;
        for(int i = len - 1 ; i > 0 ; i--) {
            if(Double.compare(cars[i].time, cars[i - 1].time) >= 0) {
                cars[i - 1].time = cars[i].time;
                carFleet--;
            }
        }
```

```
        return carFleet;
    }
    private static class Car {
        int pos;
        int speed;
        double time;
        public Car(int pos, int speed, int target) {
            this.pos = pos;
            this.speed = speed;
            time = (target - pos + 0.0) / speed;
        }
    }
}
```

11. LC857 雇工人 高频 6次

思路:
根据描述任何一个合法的pay group里面任意两个worker之间 quality1/quality2 = wage1/wage2，转化一下 wage1/quality1 = wage2 / quality2，即所有的工人他们的paid wage和他们的quality的比值都应该是一样的，根据这个性质，我们每一个合法的pay group只能取最大的wage/quality当作所有工人的pay ratio。

假如我们不取最大的，选择的ratio为w1/q1，而且存在一个工人的wagei/qualityi > w1/q1，转换一下很容易得到 wagei > quaility * w1/q1，即表示该工人的最低工资大于了他实际得到的工资，和题意不符合。

所以有了以上性质后，直接根据ratio排序，然后又需要每个group的pay最小，每个group的总工资计算方法（q1+q2+q3+...+qk）* ratio，所以就是需要quality的sum最小，那每次我们加入了一个新的ratio就把quality最大worker踢出group，这样每次group的pay可以保证是新ratio下的最小pay。遍历数组，记录所有ratio中出现的pay最小值即可。

Code
Provider: leetcode solution

```
class Solution {
    public double mincostToHireWorkers(int[] quality, int[] wage, int
K) {
        int len = quality.length;
        Worker[] worker = new Worker[len];
        for(int i = 0 ; i < len ; i++) {
            worker[i] = new Worker(quality[i], wage[i]);
        }
        Arrays.sort(worker, (a, b) -> Double.compare(a.ratio,
b.ratio));
        Queue<Worker> pq = new PriorityQueue<>((a, b) -> b.quality -
a.quality);
```

```
        int sum = 0;
        double min = Double.MAX_VALUE;
        for(int i = 0 ; i < len; i++) {
            if(pq.size() >= K) {
                Worker tmp = pq.poll();
                sum -= tmp.quality;
            }
            pq.offer(worker[i]);
            sum += worker[i].quality;
            if(pq.size() == K) {
                min = Math.min(min, sum * worker[i].ratio);
            }
        }
        return min;
    }
    private static class Worker {
        int quality;
        int wage;
        double ratio;
        public Worker(int quality, int wage) {
            this.quality = quality;
            this.wage = wage;
            ratio = (wage + 0.0) / quality;
        }
    }
}
```

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 6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 10+次

【update time：10/22/2018】
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
Provider: null

```
// key: parent, value: children
Map<String, List<String>> tree = new HashMap<>();
Set<String> dead = new HashSet<>();
String root = "king";
{
	tree.put("king", new ArrayList<>());
}

public void birth(String parent, String name) {
	if(!tree.containsKey(parent)) {
		// throw exception
	} else {
		tree.get(parent).add(name);
		tree.put(name, new ArrayList<>());
	}
}

public void death(String name) {
	dead.add(name);
}
```

Zelong Qiu: 这个方程只是lazy delete会不会导致dead越存越大？面试官会不会问如果需要实际删除发方法
根据看到的面经，还没有人被遇到过这个问题，如果有人遇到需要实际删除的问题，欢迎提供思路和代码

Provider: Yang Qi
对于动态删，提供一个思路，用一个map记录子节点到父节点的映射，如果当前节点挂掉，把自己的子节点插入的父节点自身对应的位置之后

```
public List<String> getOrder(){
	List<String> res = new ArrayList<>();
	dfs(root, res);
	return res;
}

private void dfs(String curr, List<String> res) {
	if(!dead.contains(curr)) {
		res.add(curr);
	}
	for(String child : tree.get(curr)) {
		dfs(child, res);
	}
}
```

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

https://www.geeksforgeeks.org/tree-isomorphism-problem/

951. Flip Equivalent Binary Trees

思路：简单的DFS T:O(n) if all node values are unique
参考代码：

```
public boolean flipEquiv(TreeNode root1, TreeNode root2) {
      if(root1 == null) return root2 == null;
      if(root2 == null) return root1 == null;
      if(root1.val != root2.val) return false;
      return (flipEquiv(root1.left, root2.left)&&flipEquiv(root1.right,
root2.right)) || (flipEquiv(root1.left,root2.right) &&
flipEquiv(root1.right, root2.left));
}
```

17. N叉树，要求删一些node，返回list of roots 高频8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。。不应是dfs?
想问一下这题返回的顺序重要吗？

没想到怎么做，有没有大佬写了code能发一下的?感激不尽
(Provider: fill your name)

```
   private void dfs(Node root, Node parent, List<Node> res) {
       if (root == null) {
           return;
       }
       if ((parent == null || parent.color == red) && root.color != red) {
           res.add(root);
       }
       for (Node children : root.children) {
           dfs(child, root, res);
       }
   }
public void dfsHelp (Node root, List<Node> res) {
```

```
            if (root == null) {
                  return;
            }
            if (root.isRed) {
                  for (Node child:root.childrens) {
                        if (!child.isRed) {
                              res.add(child);
                        }
                        dfsHelp(child, res);
                  }
            } else {
                  res.add(root);
                  Iterator<Node> it = root.childrens.iterator();
                  while (it.hasNext()) {
                        Node child = it.next();
                        if (child.isRed) {
                              root.childrens.remove(child);
                        }
                        dfsHelp(child, res);
                  }
            }
      }
```

```
class TreeNode(object):
    def __init__(self, val, color):
        self.val = val
        self.color = color
        self.children = []

class Solution(object):
    def listOfRoots(self, root):
        if not root: return []

        ans = []

        def dfs(node, parent):
            if not node: return
```

```
            if node.color == 'b':
                if parent.color == 'r': ans.append(node.val)
// should check if parent is None
                else: ans.append(-1)//这里加上负一是为什么？

            for child in node.children:
                dfs(child, node)

        dfs(root, None)
        return ans
```

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;
        }
        if (volumeUpper > targetUpper) {
            return false;
        }
         // if (volumeUpper - volumeLower > targetUpper - targetLower) return false;
        for (Soda soda : sodas) {
```

```
            if (dfs(sodas, volumeLower + soda.lower, volumeUpper + soda.upper,
targetLower, targetUpper, memo)) {//false的子问题会重复计算
                memo.put(volumeLower + "-" + volumeUpper, true);
                return true;
            }
        }

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

据说这题是binary search？ 这个应该bfs，dfs都能做。但是据说还可以用dp，dp怎么做啊。谁能po个解法？

区间DP的做法：(Provider: anonym)

```
public static boolean coke(List<List<Integer>> buttons, List<Integer>
target) {
    int m = target.get(0);
    int n = target.get(1);
    boolean[][] dp = new boolean[m + 1][n + 1];

    //Init
    for (int i = 0; i <= m; ++i) {
      for (int j = 0; j <= n; ++j) {
        for (List<Integer> button: buttons) {
          if (i <= button.get(0) && j >= button.get(1)) {
            dp[i][j] = true;
            break;
          }
        }
      }
    }

    for (int i = 0; i <= m; ++i) {
      for (int j = 0; j <= n; ++j) {
        for (List<Integer> button: buttons) {
          int preL = i - button.get(0);
          int preR = j - button.get(1);
          if (preL >= 0 && preR >= 0 && dp[preL][preR]) {
            dp[i][j] = true;
           break;
          }
        }
      }
    }

    return dp[m][n];
  }
```

这算是一个多重背包问题。我曾经被面到过一个类似的：给一个调色板由一堆颜色组成，每个颜色有RGB三个分量。问能否调出一个目标颜色

19. 生成随机迷宫，高频5次

左上到右下，怎么设计可玩性
maze generation. 输入是int[][] board, int[] start, int[] dest,返回一个int[][] maze. 这题题意比较复杂。简单来说就是让你随机生成一个迷宫，
条件是：
（1）你肯定要生成一些墙，这些墙宽度为1，意思就是board[0][0] - board[0][3]可以是墙，s宽度为1，长度为4。但是不能生成board[0][0] - board[1][3]这样的厚墙（2*4)
(2) 要求这个迷宫有且仅有一条路可以从start到达destination，另外对于那些不是墙的blank cell，也要有可以从start到达它的路径。也就是说不能有一些孤岛是不能到达的
(3) 后来大哥给我简化了一点，如果输入board里面
已经有一些墙，用1表示，但是这个迷宫并不是具有通路的，然后让你根据以上条件，生成迷宫。
思路：直接DFS每次走两步，避免生成没有墙的空地。
https://en.wikipedia.org/wiki/Maze_generation_algorithm
(Provider: Sean)

```
public class GenerateRandomMaze {
    public int[][] maze(int n) {
        // Assumptions: n = 2 * k + 1, where k > = 0.
        int[][] maze = new int[n][n];
        // initialize the matrix as only (0,0) is corridor,
        // other cells are all walls at the beginning.
        // later we are trying to break the walls to form corridors.
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (i == 0 && j == 0) {
                    maze[i][j] = 0;
                } else {
                    maze[i][j] = 1;
                }
            }
        }
        generate(maze, 0, 0);
        return maze;
    }
    private void generate(int[][] maze, int x, int y) {
        // get a random shuffle of all the possible directions,
        // and follow the shuffled order to do DFS & backtrack.
        Dir[] dirs = Dir.values();
        shuffle(dirs);
        for (Dir dir: dirs) {
```

```
            // advance by two steps.
            int nextX = dir.moveX(x, 2);
            int nextY = dir.moveY(y, 2);
            if (isValidWall(maze, nextX, nextY)) {
                // only if the cell is a wall(meaning we have not
visited before),
                // we break the walls through to make it corridors.
                maze[dir.moveX(x, 1)][dir.moveY(y, 1)] = 0;
                maze[nextX][nextY] = 0;
                generate(maze, nextX, nextY);
            }
        }
    }
    // Get a random order of the directions.
    private void shuffle(Dir[] dirs) {
        for (int i = 0; i < dirs.length; i++) {
            int index = (int)(Math.random() * (dirs.length - i));
            Dir tmp = dirs[i];
            dirs[i] = dirs[i + index];
            dirs[i + index] = tmp;
        }
    }
    // check if the position (x,y) is within the maze and it is a
wall.
    private boolean isValidWall(int[][] maze, int x, int y) {
        return x >= 0 && x < maze.length && y >= 0 && y <
maze[0].length && maze[x][y] ==
            1;
    }
    enum Dir {
        NORTH(-1, 0), SOUTH(1, 0), EAST(0, -1), WEST(0, 1);
        int deltaX;
        int deltaY;
        Dir(int deltaX, int deltaY) {
            this.deltaX = deltaX;
            this.deltaY = deltaY;
        }
        // move certain times of deltax.
        public int moveX(int x, int times) {
            return x + times * deltaX;
        }
        // move certain times of deltaY.
        public int moveY(int y, int times) {
            return y + times * deltaY;
```

```
            }
        }
}
```

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？
请问：这一题哪一篇帖子有比较详细的题目内容吗？感谢
请问这道题有谁做出来能po出详细的code吗？感激不尽
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
https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=434363
思路：遍历+改node值？dijkstra吧？
什么叫N层map? map是stl里面的map吗？e不改node值也行吧，反正就是DFS每一条dge存在于同层的node之间？为什么可以改node的值？可能的路径，到底了就和min cost比较就完事了？

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

这个是一个蛮经典的dp问题，lintcode上也有类似的题目
https://www.lintcode.com/problem/coins-in-a-line-ii/description

这个是LINTCODE 左数两个的解法：

```
 public boolean firstWillWin(int[] values) {

        // write your code here
        int sum = 0;
        for (int v : values) {

            sum += v;
        }
        memo = new int[sum + 1];
        Arrays.fill(memo, -1);

        int one = dfs(values, sum, 0);
        int two = sum - one;
```

```
        return one > two;
    }

    int memo[];

    int dfs(int[] values, int sum, int pos) {
        if(pos >= values.length) {
            return 0;
        }

        if(memo[sum] != -1) {
            return memo[sum];
        }

        int first = values[pos];
        int op1 = sum - dfs(values, sum - first, pos + 1);

        int op2 = 0;

        if(pos + 1 < values.length) {

            int second = values[pos + 1];
            op2 =  sum - dfs(values, sum - first - second, pos + 2);
        }

        memo[sum] = Math.max(op1, op2);
        return memo[sum];
    }
```

4。image以byte[][]储存 如果想中心镜像翻转怎么弄
https://www.1point3acres.com/bbs/thread-409626-1-1.html
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
可以详细讲一下这里怎么用位运算吗

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用尽可能大的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
Provider：Qiaoqian Lin　　Harry Yi: 给的API里面每个char的width是不同的，下述方法还需要额外计算一下

```
def fitScreen(screen_width, sucreen_height, num_chars, font_sizes):
    def ok_or_not(width, height, screen_width, screen_height, num_chars):
        num_chars_per_line = screen_width // width
        num_chars_per_column = screen_height // height
        num_chars_can_fit = num_chars_per_line * num_chars_per_column
        return num_chars_can_fit >= num_chars

    lo, hi = 0, len(font_sizes) - 1
    while lo < hi:
        mid = (lo + hi + 1) // 2
        font_size = font_sizes[mid]
        width, height = API(font_size)
        size_ok = ok_or_not(width, height, screen_width, screen_height, num_chars)
        if not size_ok:
            hi = mid -1
        else:
            lo = mid
    return lo
```

屏幕超大，如何speedup?

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

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
问：这个不merge，直接所有的往treeset里扔可以吗？

10。iterator of iterator
LC类似题目：
LC281 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：BFS 没什么好说的。。。

Provider(ccc/ddd)

```
class ZigZagIterator {
public:
    ZigZagIterator(vector<vector<int>>& v) {
        for (int i = 0; i < v.size(); ++i) {
            if (!v.empty()) {
                its.push_back({v[i].begin(), v[i].end()});
            }
        }
        current = 0;
    }

    bool HasNext() {
        return !its.empty();
    }

    int Next() {
        auto it = its.begin() + (current % its.size());
        int val = *(it->first);
        if (it->first + 1 < it->second) {
            *it = {it->first + 1, it->second};
        } else {
            its.erase(it);
        }
        ++current;
        return val;
    }

private:
    int current;
    vector<pair<vector<int>::iterator, vector<int>::iterator>> its;
```

11。 LC68 text 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。

12 **[Bash Brace Expansion]** 给出 a{d,c,b}e 得出ade ace abe。 没有nested的括号，但是可以里面的字符有括号。我是先parse再dfs。 followup是括号可以nested

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：见lc

5。LC844 Backspace string compare
思路：简单stack秒杀 注意follow up是O(1) space → two pointers
6。 LC394 `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

https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=507223&extra=page%3D1%26filter%3Dsortid%26sortid%3D311%26searchoption%5B3046%5D%5Bvalue%5D%3D1%26searchoption%5B3046%5D%5Btype%5D%3Dradio%26sortid%3D311%26orderby%3Ddateline
这个帖子里要求必须用dp，这里贴一个dp解法。
Dp[i][j]表示前i个加油站增加j个站点最小的最大距离.

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
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);
  void print();
}
```

started(100, "1")
started(101, "2")
finished(102, "2")

started(103, "3")
finished(104, "1")
finished(105, "3")
print()

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

思路：
和lru差不多， hash map + double linked list维护顺序(hanxiao yan：弱弱问一下，这个log的题为啥要用double linkedlist呀，用single linkedlist不可以么)（editor：这里可以用single list，作者因为觉得和lru题目相似，直接用的dll，这里不需要操作中间节点，感觉可以用single list）(为什么不直接用linkedHashMap 或者TreeMap)
Node {
	String id;
	long start;
	long end;
}
Map<String , Node> map
start 的时候加入map，设置好开始时间，end时间还没有设置
finish的时候设置好node的end值，然后从map中移除
打印的话直接遍历一遍dll即可

code
Provider: null
```
public class MyLogger implements Logger {
	private class Node {
		String id;
		long start;
		long end;
		Node prev;
		Node next;
		public Node(String id, long start) {
			this.id = id;
			this.start = start;
			end = -1;
		}
	}
	Node head, tail;
	Map<String, Node> map;
	public MyLogger() {
		head = new Node(“”, -1);
		tail = new Node(“”, -1);
```

```
            head.next = tail;
            tail.prev = head;
            map = new HashMap<>();
      }
      @Override
      public void started(String id, long time) {
            Node curr = new Node(id, time);
            map.put(id, curr);
            add(curr);
      }
      @Override
      public void finished(String id, long time) {
            Node curr = map.get(id);
            if(curr != null && curr.end == -1) {
                  curr.end = time;
                  map.remove(id);
            }
      }
      public void print() {
            Node curr = head.next;
            while(curr != tail) {
                  if(curr.end != -1) {
                        System.out.println(curr.id + “start at” +u
            curr.start + “ end at ” + curr.end);
                  }
                  curr = curr.next;
            }
      }
      private void add(Node curr) {
            if(curr == null) return ;
            curr.next = tail;
            curr.prev = tail.prev;
            tail.prev.next = curr;
            tail.prev = curr;
      }
}
```

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

***
LC939 && LC963 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？
优先拿掉不导致component数量增加的棋子。

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++

Google Snapshot
实现三个functions， get, set, take snapshots。其实就是一个长度为N的Array。Set可以设置Index i的值，每次take snapshot， version + 1，并且记录下当前version下 Array里面的值。然后get方法可以得到某一个Version下，每一个Index的Array的值。就是非常Naive的方法，在Chromebook上写完了。写完之后有一个变量名Typo被指出。口头跑了Test case。Follow up 时空复杂度，并且要节省空间。

举例
初始 100001
take a snapshot 返回sid 1
改变成 100201.
take a snapshot 返回sid 2
这时lookup get（3，1）（ get(index, snapshotID)）应该返回0而不是2，这是version control的原理可是关键在怎么存最省空间

可以参考视频压缩，每隔若干帧保存一份完整的图像，期间只保存delta。重建图像的时候，从最近的full snapshot开始，然后apply 每个version的delta

一开始三个数：　　0 0 0
//这时版本数为 0
set(0, 10) // 10 0 0
set(0, 12) // 12 0 0
takesnapshots() //这时版本数变成了1
set(0, 9) // 9 0 0
takesnapshots() //这时版本数变成了2
set(0, 3) // 3 0 0

get(0, 0) // 第一参数为版本数，第二参数是index。则返回12。
get(1, 0) // 返回9

看来面经还是有点用的.. 这个我也是碰到了原题. 然而并没有看到这个面经.. 哈哈
单独把每个元素处理就可以了, list<int[]> 或者treemap.

第一轮美国白人小哥，一路提醒
自己设计一个SnapShot class 和一个SnapshottableMap class，snapshot要实现get(String key) function，snapshottable map要实现put(String key, String value)，get(String key), createSnapshots(), List<Snapshot> getSnapshots()，四个function

Example：
SnapshottableMap map = new SnapshottableMap();
map.put("name", "John");
map.put("country", "UK");
Snapshot s1 = map.createSnapshot();

assert(s1.get("name").equals("John"));
assert(s1.get("country").equals("UK"));

map.put("name", "Marta");
Snapshot s2 = map.createSnapshot();-baidu 1point3acres

assert(s2.get("name").equals("Marta"));

```
assert(s2.get("country").equals("UK"));
assert(s1.get("name").equals("John"));
```

implement 4 interfaces for class Snapshot, 面之前没见过，后来好像看到是高频题，不知道怎么才是最佳。如果用数组，每次snapshot增加时复制，实现容易，get O(1), set O(n)，空间复杂度不好。估计不是面试官想要的。

```
class Snapshot {

int get(int index)  --> get current snapshot value on index

int get(int index, int snapshot)  --> get value on index on some snapshot

void set(int index, int value)   --> set value on index to be the new value, snapshot would be increased

int snapshot()  --> return current snapshot id
};
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Main {
  public static void main(String[] args) {
        SnapshottableMap map = new SnapshottableMap();
        Snapshot s0 = map.createSnapshot();

        map.put("name", "John");
        map.put("country", "UK");
        Snapshot s1 = map.createSnapshot();

        Assert(s1.get("name").equals("John"));
        Assert(s1.get("country").equals("UK"));

        map.put("name", "Marta");
        Snapshot s2 = map.createSnapshot();

        Assert(s2.get("name").equals("Marta"));
        Assert(s2.get("country").equals("UK"));
        Assert(s1.get("name").equals("John"));
```

```
        Assert(s0.get("name") == null);

        map.put("label", "AAA");
        map.put("label", "BBB");
        Snapshot s3 = map.createSnapshot();
        Assert(s3.get("label").equals("BBB"));

        System.out.println("--------\n" + s1.toDebugString() + "--------\n");
        // --------
        // country: UK
        // name: John
        // --------

        System.out.println("--------\n" + s2.toDebugString() + "--------\n");
        // --------
        // country: UK
        // name: Marta
        // --------

        System.out.println("--------\n" + s3.toDebugString() + "--------\n");
        // --------
        // country: UK
        // name: Marta
        // label: BBB
        // --------
  }

  private static void Assert(boolean value) {
        if (!value) {
        throw new RuntimeException("");
        }
  }
}

// Snapshot只存version number和主map的引用
class Snapshot {
  private int snapshotVersion;
  private SnapshottableMap map;

  public Snapshot(int version, SnapshottableMap map) {
        this.snapshotVersion = version;
        this.map = map;
  }

  public String get(String key) {
```

```
        return map.get(key, snapshotVersion);
    }

    public String toDebugString() {
        return map.toDebugString(snapshotVersion);
    }
}

// SnapshottableMap的实现
class SnapshottableMap {
    private int lastestVersion = 0;
    private List<Snapshot> snapshots = new ArrayList<>();
    private Map<String, ValueHistory> storage = new HashMap<>();

    public void put(String key, String value) {
        ValueHistory history = storage.get(key);
        if (history == null) {
        history = new ValueHistory();
        storage.put(key, history);
        }
        history.put(value, lastestVersion);
    }

    public String get(String key) {
        return get(key, lastestVersion);
    }

    // 微量级时间的create snapshot
    public Snapshot createSnapshot() {
        Snapshot snapshot = new Snapshot(lastestVersion++, this);
        snapshots.add(snapshot);
        return snapshot;
    }

    public List<Snapshot> getSnapshots() {
        return snapshots;
    }

    public String get(String key, int version) {
        ValueHistory history = storage.get(key);
        if (history == null) {
        return null;
        }
        return history.get(version);
    }
```

```
  // 打印某一个version的所有值用于调试
  public String toDebugString(int version) {
        StringBuilder sb = new StringBuilder();
        for (String key : storage.keySet()) {
        final String value = get(key, version);
        if (value != null) {
        sb.append(key + ": " + value + "\n");
        }
        }
        return sb.toString();
  }
}

class ValueHistory {
  // 代码逻辑保证slots中version号码一定是有序的
  private ArrayList<ValueSlot> slots = new ArrayList<>();

  public void put(String value, int version) {
        // 修改最后一个version或者是添加新version
        if (!slots.isEmpty()) {
        ValueSlot lastSlot = slots.get(slots.size() - 1);
        if (lastSlot.getVersion() == version) {
        lastSlot.SetValue(value);
        return;
        }
        }
        slots.add(new ValueSlot(version, value));
  }

  public String get(int version) {
        // 二分查找version
        int index = Collections.binarySearch(slots, version);
        if (index < 0) {
        if (index == -1) {
        return null;
        }
        // 不存在指定version，但是有之前的version
        index = -index - 2;
        }
        return slots.get(index).getValue();
  }
}

class ValueSlot implements Comparable<Integer> {
```

```
  private int version;
  private String value;
  public ValueSlot(int version, String value) {
        this.version = version;
        this.value = value;
  }
  public int getVersion() {
        return version;
  }
  public void SetValue(String newValue) {
        value = newValue;
  }
  public String getValue() {
        return value;
  }
  @Override
  public int compareTo(Integer other) {
        return version - other;
  }
}
```

June

[谷歌公司Youtube办公室面试经验]

一开始一个白人大叔，问我假如有一个整形数组，要分成五个五个一组，每组里面的元素都是以1为等差递增的。我又用TreeMap做的，需要O(N logN)的时间O(N)的空间，大叔指点了一下发现可以用O(1)的空间。

之后一个印度小哥，问我啥给我一个字符串和一个模式，找到字符串中符合模式的最短子字符串。LeetCode上面的原题，用两个指针滑动窗口做了。

然后中午和一个亚洲女生一起吃饭，感觉她是中国人，但是我们全程英文聊天。Youtube饭还行。

下午一个女亚裔面试行为问题，这个环节是谷歌新增的，其实都是典型的问题，比如遇到过什么困难，有没有带过别人或者团队，还有和团队里面意见不一样怎么办。面试官也是第一次做BQ的面试。

之后一轮印度小哥，问我加入一条路上面有N个街区，每个街区都有不同的场所，有人要找房子，需要去所有需要的M个场所，请找出一个街区，使得这个街区到所有的场所的最大值最小。一开始用了O(M)+O(N logN)的解法，后来小哥提示以后找到了O(M)+O(N)的解法。

最后一个国人小哥，问的是如果要去N个城市按顺序旅游，最多有M天，每个城市必须至少呆一天，而且必须那几天是晴天，已知每个城市每天的天气，请问能不能在M天之内旅游完。用DP做了，后来在小哥指点下优化了一下

G家onsite 面经
1. 给你一个width， height， 一个起始点，和终点， 写一个api来创建迷宫， 要求每一次call都会随机生成一个不一样的迷宫，而且只能有唯一solution， 每一个matrix cell可以想象成4座墙。 其实也是DFS就可以做。
我倒是做出来了， 基本就是每一步DFS要随机生成下一个你要打通的cell，但是随机生成时有点点trick，这里没有写太好，最后feedback 给的是average。 我本以为这一轮会给strong， 因为感觉题目有点难，而且当时面试官反馈还不错

2，behavior。 Feedback比较strong

3 frog jump LC403. 原题， 可惜我没做到那一题， 其实也是DFS来做，但是当时脑袋抽风，想着做了这么多DFS，来个DP试试，结果面试官以下来劲了， 不停的问为啥要用DP，啥是DP，不停打断思路，后来想了会发现不应该往这条路走，于是老老实实用DFS做，这才发现思路跟面试官想的一样了。但耽误了时间，没能完全写完，思路最后说了，所以。。。 这一轮结果不好

4. design a job schedule system 这一轮主要是问如果让你设计一个这样的系统，能跑任何程序，需要哪些东西，哪些UI给用户。 我承认这一轮回答一般，大致框架说了，但是细节说不到点上，毕竟有些东西我确实没做过，也不太熟，但是很郁闷的是面试官给了我很大的错觉，上来就是中文，还告诉我之前跟我一个公司出来的，让我倍感亲切， 每次他问一个问题，我还没想好，他就直接说了答案，面试完还聊了好久，我以为。。。这位大哥肯定会帮下我呢，结果好像这一轮feedback最差。。。哎。

5，coding （Feedback strong）
这题具体细节不说了，不难，medium， 跟LC341基本一样

May

https://www.1point3acres.com/bbs/thread-526055-1-1.html
1. 国人小哥，compress size。总共分为几个部分。先是一个无重复数字的数组，假如[1,2,50,100]，要求只要把最大的数字in place的变成最小，且比其他数字要大，这里的话就是变成[1,2,3,4]。然后followup1是变成一个二维数组，同样规则。followup2是二维数组里面要求compress时，只要同一行里满足第一问的条件，同一列里也满足就可以，也就是说，满足这个条件的话，哪怕这个数在整个二维数组里是第5大的，但是如果把它变成4也可以满足，那就把它变成4，同时只要当前的数不是数组中最大的一个，它是几都无所谓。题目有点绕，是小哥自己想的，最后做出来的解法大概是keep一个数组作为col max，再keep一个常量作为row max，然后循环，比较，只有在比当前值大的时候才更新。

2. 国人大哥。挂在这轮了。是insert number into a circular linkedlist，并update head到最小数字，据他自己说，题目不难，就是edge case很多，我做的不好，磕磕绊绊的。感觉可能如果这个题能秒他应该还准备了一个，然后我没有做到

3. 吃饭，美国小哥，很nice，带我逛campus，回答问题，饭还不错。

4. 印度tech lead。问的是给一群points，要求找到任意一组使得他们中点坐标都是整数。真的是很抓狂的一轮，我给的各种解法他都不满意，而且一定要我做到他的解法，最后还剩15分钟了才说出bit manipulation。我简直要昏倒了。第二题是profiler，给定log的inclusive时间和exclusive时间，问我怎么通过inclusive时间求出exclusive。我说build map，indegree为0的剪掉indegree为1的时间，没时间写了。

5.中国小姐姐。围棋围杀o和x，给一个当前状态的棋盘，要求更新一下被围杀掉的o的位置的棋子。就dfs没啥好说的

6. behavior。白人manager姐姐，问了各种情景问题啦，给一些自己做项目的例子啦。大概记得几个问题，有没有跟different time zone的人合作的经历，是怎么做的。有没有和很多组一起合作的经历。最看重的manager应有的素质。最看重的组员应有的素质。自己认为自己最强的素质。有没有什么是自己坚持的无法放弃的目标。有没有和经理产生冲突的时候，怎么解决。这轮真的聊的特别high，小姐姐非常善于倾听和给我一些正面的反馈，聊完我觉得我都要被自己感动哭了。最后feedback证明这轮也是strong hire了，然并卵

[刚结束的狗家西雅图上门面试]
1.国人小姐姐： 给一个句子和一个2d array. 然后通过2D array替换句子里的单词得到新的句子。例子：“alice is a good girl”, {{0, alice, bob}, {6, is, was}} 得到答案是 "bob was a good girl". 数组里的第一个是要替换的word的位置，第二位是原句子中的word，第三个是替换后的word。题目本身不难，注意各种corner case。比如word和index是否有效，index是否overlap等等。
[lc833 就是考分析corner case能力的 过于复杂的很难implement]
followup是给1000个数字，除去最小的5%和最大的5%，求中间剩下90%数字的平均值。LZ用quick select做了。又问改成数据流怎么办，怎么优化，当时时间不多了就答了priorityqueue，不过好像并不是面试官想要的，细节也没来得及深入，后面仔细想想可能是segment tree

2.白人大哥：给一个句子和一个数组，数组里是2个单词。然后统计在句子里，2个单词之后的那个单词的出现次数。例子：“alice is a good girl, she is a good student”, {"a", "good"} 得到 {"girl" => 1, "student" => 1}. 先是**corner case讨论**，句子有**标点**的话就先**全部删掉**。clarify了数组里的2个单词**顺序是不能换**的，比如换成good, a那就没有任何答案了。先brute force的解法，把2个单词组和在一起去原句子里找。后面又问句子是固定的，2个单词会变很多次怎么办，答预处理一下那个句子，放在constructor里。这里有2个办法，一个用hashmap一个用trie，都跟面试官讨论了一下利弊，最后让我实现trie的写法，这里他给了我个小hint，就是**单词放在trienode**而不是放character. 写完了还有10分钟，也没followup了，聊了会天去吃饭了

3.午饭就不说了，国人大哥很nice

4.白人小哥：系统设计：小哥也没提前准备，就看LZ简历有写爬虫的背景，就问了个爬虫的设计。设计一个系统，从各个餐厅相关网站(类似yelp这种)爬餐厅的信息和评论最后存进数据库里。
https://www.1point3acres.com/bbs/thread-208829-1-1.html 这个帖子的系统设计讲的很好，大家可以看看，我就照着上面的框架讲的，沟通也流畅。因为数据库是现成的，没让我设计数据库，主要就是讨论怎么划分service，然后还有一些实际问题怎么解决，比如餐厅换名字之类的，怎么保证一致性。

5.白人大胡子哥：问了一些背景，然后开始答题：给了一个数组，有如下特点：sorted, 有重复，有且仅有一个数字在数组里的数量超过了25%，找到这个数字。比如1,2,2,6,6,6,6,7,10 那么答案就是6. 很明显是二分的题，不过刚开始还是跟大哥讨论了一下用map，用bucket sort的解法，后面LZ卡了很久怎么二分，最后才想出来用把原数组切割成三部分，再做二分：分别是0-50%, 50%-100%, 25%-75%

，三个数组中肯定有一个中间的数字就是答案，用二分找到左右边界求出数量，再看看是不是超过了25%就行。代码的话二分那块也没来得及写完。感觉凉凉

克可蓝 🐱 家 新鲜 上门经✌🟫

1. 国人。a.热身题：给你一个句子，返回以空格为分隔的string list。不能用自带的.split() 指针过一遍，实现的时候主要注意头尾还有中间连续空格的情况就好。

```
 public static void main(String[] args) {
   String s = "   He       llo              Wo    rl       d!    ";
   System.out.println(split(s));
 }
 static List<String> split(String s) {
   List<String> res = new ArrayList<>();
   int left = 0, n = s.length();
   while (left < n && s.charAt(left) == ' ') {
      left++;
   }
   // if (left == n) return res;
   for (int right = left; right < n; right++) {
      if (s.charAt(right) == ' ') {
         res.add(s.substring(left, right));
         while (right < n && s.charAt(right) == ' ') {
            right++;
         }
         if (right == n) return res;
         left = right;
      } else if (right == n - 1) {
           res.add(s.substring(left));
      }
   }
   return res;
 }
```

b. 实现一个有dependency的class的打印，要求从dependency从深直到打到自己这层。 比如 A depends on B, C. 打印顺序是 [B,C,A] 其中B C 可能有自己的dependency 那就先打那些。用dfs 过一遍。感觉出乎意料的不难（也可能自己没理解对题目）

2. 国人 （远程视频）

a. 之我介绍

b. 给你个系统，写test case。 这轮在这花了蛮多时间，一开始没答到点子上，撤的有点多，面试官和我自己都没注意到时间，以至于后来算法题只能正好完成简单题，followup 没时间做。

c. 算法，给你一串apk 和他们支持的sdk版本号范围。返回sdk版本号分段区间。 例如 A - (3,8) B -(6,10) 返回 （3,5)(6,8)(9,10) followup 是要返回每个区间支持的apk号，比如例子里返回的3个区间分别支持 A, AB, B

这题其实也不难，但是没时间做完。这轮面的不好。所以在这提醒大家*3: 一定要注意时间，个人/project介绍都要控制好时间，要学会结束话题。

[sweep line]给的输入都是闭区间

3. 白哥 带着午饭

4. 白哥 高频的自行车题（人车匹配）

地里面经出现好多次了，居然还是问到了。这题我用pq解的，可能并不是最优的解法，但是我能解释的最好解法了。这里有个之前找到的2分法 km算法的文章，个人没有真正理解，所以没用这个方法。可能是最优解法，但是很偏的算法，没有完全理解就不敢乱用。有兴趣的同学可以看看，可能会有帮助：https://blog.csdn.net/qq_37943488/article/details/78586048

5. 白哥
a. 上来也是给个系统，求testcase。基于第二的轮的经验，我没废话，直接讲重点，很快把能想到的都说一遍。然后就开始做算法题
b. 一条路，一次只能过一个车，给你起始状态 和 终止状态，问终止状态 是可以从起始状态演变来的吗？
比如 _ _ R _ _ _ L _ ，_ _ _ R L _ _ _ --> True. L 表示 车向左开，R向右开，相遇就停，不会撞车。
我是用2个指针各自过一遍，总的来说这题不难。答完还有5分钟，和面试小哥聊了聊工作。
个人感觉这轮面的最好，test部分，算法部分 都和和面试官的互动很多。一开始他给的题目漏给了信息，过程中我帮他发现了遗漏的内容（车子们速度可以不同），补充完继续做。期间还给了其他的做法，他觉的可能是个不错的followup 表示可以回去研究研究。
判断两个条件 1) 每个车起始和终点位置和车行驶方向一致 (标记为L的车终点必须在起始点左边） 2) 两个车的相对位置没有改变 只有这两个条件都满足才算是可以
6. **高频　11。log start log finish**
设计 logger，实现2个api： start(request, time) 和 finish(request, time)
要求*finish的时候* 按照开始的时间*打印尽可能多的完成了的*request. 用一个hashmap和treemap来实现的。 好想也可以用2个map和pq来做。

followup 是问这个logger是个singleton 怎么做到thread safe。我只想到加一个lock，然后任何一个thread在读写时候先检查lock，没有的话上锁操作，然后结束解锁。有谁知道更好的方法的话，求补充。谢谢

April

https://www.1point3acres.com/bbs/thread-516897-1-1.html
1. ABC小哥，给一个数组，问里面的数字能不能组成若干个长度为5的连续数字组。比如[1,2,3,4,5,10,11,12,13,14]可以分成两组，每组连续且长度为5. Followup 是LC刘武久。我的解法有点麻烦，不过应该是正确的解法。。。

2. 白人小哥，给一个N，求对应的N*N的幻方。上来就说要brute force，不要考虑数学。Followup就是怎么优化，和一些test case。这轮有点像sudoku solver，就是纯dfs暴力解，解完验证。
Lunch是国人姐姐带去吃的拉面，食堂里面几乎全是国人，拉面味道也不错，肉很多。。。

3. 阿三小哥，上来聊了好久我的proj，感觉他还挺有兴趣的。。。题目是，给两个string，相同长度，问能不能在同一个地方切一刀，然后两个string的各半边组成一个palindrome。 比如aabac和xyzaa，可以在aab|ac和xyz|aa这里切，然后aab和aa可以组成palindrome。要求是必须切在相同的位置，而且要第一个左边和第二个右边，或者第二个左边和第一个右边组成palindrome。followup是给出所有位置。我上来有点懵，小哥给了提示才用了two pointer做的。。。当时觉得这轮可能会挂

4. 国人小哥，上来说我们热个身，N个数字随机选K个，我说那就resevior sampling呗，大概说了一下，就过了。然后给个题目是，有个产品有很多的version，每个version有相同或者不同的loading time,

假设这些loading time只会增长，不会减少, 给一个version的区间，求这个区间内，每个jump的loading time 对应的版本号。有个API可以求对应版本的loading time。比如[1,1,2,2,2,4,5,6,6]， 如果index就是版本号的话，那么要给出[2,5,6,7]. 第一个不算。我用两层二分去做，先找到下一个不同的版本，然后在这个区间内找第一个变化的版本。然后又讨论了下时间复杂度，感觉这个不是小哥要的解法，因为他对时间复杂度很疑惑。。。

[高频Bash Brace Expansion]5. 国人小哥，感觉干了一天不是很开心的样子。。。题目不难，是bash brace expansion, 给出 a{d,c,b}e 得出ade ace abe。 没有nested的括号，但是可以里面的字符有括号。我是先parse再dfs。写的可能有点乱，面试官说可读性可以再加强一下。 然后看时间还有，就给了第二题，一个数组长度为N+1,里面可能有1-N区间的整数，求里面任意重复的数，里面可以能很多重复的数。要求On O1的时间空间。想了一会，给了LC思瑶的方法，面试官说可以，就送我出门了。。。

March 18

There are two colors in the matrix, find the shortest manhattan distance of two colors
There are an array of lights. I have a list of intervals, eg [0,2) which means switch the light #0, and light #1. I want to know the final state of these lights

朋友的面经，请问这两道有什么好的方法？
kai Peng:第一题用BFS，第二题区间开头加一结尾减一遍历一遍判断奇偶数？
Ziwei Guo: 感觉第一题就是人车匹配，可以bfs也可以heap
第一题面试官说bfs太慢了。。第二题面试官问有没有比O(N) 快的方法
Kai Peng: 第一题给个例子吧，第二题如果是要确认所有的灯Output 不就O(n)了?如果是像my calendar系列那样一个function change state一个function check，用segment tree或binary index tree把change state和check state都做到log(n)

狗家西雅图丢人onsite
https://www.1point3acres.com/bbs/thread-500242-1-1.html
1.白人小哥。很抽象的题目，到现在也没理解面试官让我做什么，大意是一个matrix里每个点都是pixel，给一个predefine api fill(color)，会把一个pixel的颜色改成另一个颜色，然后逐渐扩散向四周重复相同改动，但是 1.小哥在这个matrix里随便画了个封闭曲线，说fill的时候不能超过这个曲线范围，懵逼的地方在于这个曲线真的是随意分割了每个matrix cell 2.matrix太大，机器只能一次读取一部分，让写一个recursive function 来fill 完这个封闭曲线内所有target color。到最后都不太确定写的符不符合小哥要求，小哥说了句works for me然后就走了，这轮跪

2.长发小哥，两个人向matrix放砖块游戏，写一个判断当前回合游戏还能否进行的function. 比如砖块规格2x4, matrix m*n 先暴力再用bomb enenmy 思路优化一丢丢，小哥说never seen someone done that,followup 如何作出最优一步，这轮一直在讲思路，没怎么写码，小哥很开心 猜测 positive

3. 亚裔，snapshot, 先给暴力再给map嵌套treemap优化，面试小哥不懂java，完全不知道treemap这个东西，但是他说完全相信我....这轮猜测positive

4.东南亚人，类似course schedule, 面试我的是个data scientist, 给我的感觉就是不是很懂算法，我写topo sort 的过程中他的问题全部都与我的思路无关，就给人一种not on the same track的感觉。这里他让我define class job, 每个job 知道自己的prerequisite, 我就说先建立directed graph + 记录indegree, 标准course schedule思路，但这里感觉他不同意建立graph，一直问我建立graph复杂度是多少。说真的，要么这轮他试图把我拉回right track但是我没意识到，要么他不是很懂算法，网上看了这个题记下来时间复杂度之后就来考我，反正一道简单的题各种不愉快，他觉得我复杂度太高(我后来意识到这里是我复杂度想错了其实，但是解法绝对没问题的)，给我说了应该是这样blablabla。猜测non-positive。 这里顺便问下，每个job知道自己prerequisite的前提下如何判断一个list里的job能否跑完 这道题，除了建立directed graph, 即 prerequisite 指向 下一个job以外，还有什么解法吗？

5.美国小哥 判断subarray和为0，先假装暴力再presum array做了，这里用了个set判断重复，犯蠢忘记把0放进去被面试官指出来，followup 是tree path sum,这轮猜测neutral吧谷歌室友讲bug被挑是很严重= =

Feb 26

第一轮
领我进门的中国小哥。面经上看到过，应该好好写一遍。 {a, b}c{d,e}f 需要return 所有可能的组合，acdf，acef，bcdf，bcef。followup 怎么处理 nested case: a{b{c, d}e{f}}

这个题哪位同学有例题链接求贴

Feb 11

移除石头 + string deduplication + bingo game + skip iterator

SETI Onsite，MTV, Fail
Provider: null

第一轮：白女，LC 947 remove marble，不过输入可以自己定义，我定义的输入是个二维矩阵，好写一点。

思路：没什么好说的，直接dfs，本来想用union find，但是面试官明确想让我用dfs去找connected components。写完面试官表示code写得很不错，然后没有follow up

当时写的白板代码：

```
int count;
int totalCount;
int rows;
int cols;
int[][] graph;
// 0是空格，1是marble
int removeMarble(int[][] graph) {
      // sanity check，这里只是说了下corner case，面试官说不用写
```

```
      rows = graph.length;
      cols = graph[0].length;
      this.graph = graph;
      for(int i = 0 ; i < rows;  i++) {
            for(int j = 0 ; j < cols ; j++) {
                  if(graph[i][j] == 1) {
                        dfs(i, j);
                        count++;
                  }
            }
      }
      return totalCount - count;
}
void dfs(int row, int col) {
      totalCount++;
      graph[row][col] = 0;
      for(int i = 0 ; i < rows ; i++) {
            if(graph[i][col] == 1) {
                  dfs(i, col);
            }
      }
      for(int j = 0 ; j < cols ; j++) {
            if(graph[row][j] == 1) {
                  dfs(row, j);
            }
      }
}
```

第二轮：白人小哥，先狂问了一些java基础和ood的基础，java中的多继承，什么是封装，composition（聚合）和继承什么区别，什么是多态，java中什么是checked exception，什么是runtime exception。其他的记不起来了，因为我说我主语言是java，恰好小哥也很喜欢java，所以这轮问我基础问得很多。然后是做题，string deduplication的变种，给一个string，保证全是大小写英文字母，当出现同一个字母的一个大写挨着一个小写时候，消除掉这个pair。
比如：
abBCcd -> return ad
abBCcAd -> return d

思路：双指针直接o(n)时间o(1)空间解出来，slow指针用来模拟栈的结构，每次检查slow-1字母和当前字母是否是同一个字母的大小写，是的话弹栈，不是的话当前字母进栈。很常规的字符串处理方法。

当时写的code：

```
String deduplicate(String str) {
      // sanity check, 也是和面试官讨论了下，说不用写了，直接写主函数
```

```
        char[] s = str.toCharArray();
        int slow = 0;
        int fast = 0;
        while(fast < s.length) {
              if(slow == 0) {
                    s[slow++] = s[fast];
              } else {
                    if(Character.isUpperCase(s[slow -1]) &&
        Character.isLowerCase(s[fast]) ||
        Character.isLowerCase(s[slow-1]) &&
        Character.isUpperCase(s[fast])) {
                          char c1 = Character.toLowerCase(s[slow-1]);
                          char c2 = Character.toLowerCase(s[fast]);
                          if(c1 == c2) {
                                slow--;
                          } else {
                                s[slow++] = s[fast];
                          }
                    }
                    else{
                          s[slow++] = s[fast]
                     }

              }
              fast++;

        return new String(s, 0, slow);}
}
```

第三轮：白人小哥，在google已经工作了7年，开始让我狂问他问题，然后眉飞色舞的回答了我的问题，这个阶段大概聊了超过10分钟。然后问的算法题叫bingo game。让我随机生成一张5*5的棋盘，只有两个限制：
https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=452049

1. 第一行的所有数字在1-15的范围内，第二行在16-30的范围内，以此类推
2. 每一行的数字不能有重复

给了个api：random(min, max)，可以返回[min, max]内的随机数字
follow up是生成n张这种棋盘，保证n张棋盘中的同一行没有完全相同的一行

思路：
当时看到题目是懵逼的，感觉太过简单，小心翼翼的多次确认，确实只有这两个限制，然后我说我想每一行用set存放已经生成过的数字，然后新生成的数字fill进棋盘之前检查下是否生成过就可以了，问他怎么样，他说on the right way。然后follow up，我直接开了一个长度为5的set数组Set<List<Integer>>[] sets来存放所有已经生成过的行，然后新生成的行都需要check一下是否已经生成过，生成过的话重新生成，面试官表示on the right way，因为

他就想避免bias，即如果生成两次[1,2,3,4,5]，如果只修改一个数字让它们不完全相同的话他说会产生bias，他想避免这种情况。

当时写的code：

```
int[][] generate() {
      int[][] board = new int[5][5];
      for(int i = 0 ; i < 5 ; i++) {
            Set<Integer> visited = new HashSet<>();
            for(int j = 0 ; j < 5; j++) {
                  boolean filled = false;
                  while(!filled) {
                        int val = random(i * 15 + 1, i * 15 + 15);
                        if(visited.add(val)) {
                              board[i][j] = val;
                              filled = true;
                        }
                  }
            }
      }
      return board;
}
```

follow-up
和上面code差不多，只不过每次一行生成完了后去set检查下生成过没有

第四轮：白人小哥，google工作5年，之前3个面试官好像都是swe，这个是seti，上来自我介绍了一下，然后直接开始上题目，skip iterator,定义个class如下：

```
class SkipIterator {
      public SkipIterator(Iterator itr);
      boolean hasNext();
      Integer next();
      void skip(int num);
}
```

传入的迭代器让我看作是在一个List<Integer>上面的迭代器，主要说下skip函数，输入参数是个int，表示下一个数字等于num的需要被跳过，就是当作这个list中下一个num不存在，skip可能被多次调用,skip(5),skip(5)表示后面的两个5都不要了。

思路：
当面试官说skip可以多次调用时直接想到的用map存放skip调用的频率，所以思路也很直接Map<Integer, Integer>，key是需要被跳过的数字，value是接下来多少个key需要被跳过，然后用一个Integer curr缓冲下一个integer，skip每调用一次频率加一，每跳过一次key，key的value减一。然后我设计的iterator中是next中call hasNext，hasNext负责找下一个合法元素，但是正确解法应该用另一个函数去专门找下一个合法元素。

估计是挂在了这一轮，当时代码写得有点凌乱，也没有口头跑一下保证一定work，写完面试官直接让我想test case，根本不让double check，估计是因为他是seti，比较看中这点。

正确解法应该用一个函数去专门找下一个合法元素：

```
class SkipIterator {
	private Iterator<Integer> itr;
	private boolean hasNext;
	private Integer nextElement;
	private Map<Integer, Integer> map = new HashMap<>();
	public SkipIterator(Iterator<Integer> itr) {
		this.itr = itr;
		findNext();
	}
	public boolean hasNext() {
		return hasNext;
	}
	public Integer next() {
		if(!hasNext) return null;
		Integer tmp = nextElement;
		findNext();
		return tmp;
	}
	public void skip(int num) {
		if(hasNext) {
			if(nextElement == num) {
				findNext();
			} else {
				map.put(num, map.getOrDefault(num, 0) + 1);
			}
		}
	}
	private void findNext() {
		hasNext = false;
		nextElement = null;
		while(itr.hasNext()) {
			Integer e = itr.next();
			if(map.containsKey(e)) {
				map.put(e, map.get(e) - 1);
				if(map.get(e) == 0) map.remove(e);
			} else {
				hasNext = true;
				nextElement = e;
				return;
			}
		}
```

```
        }
}
```

Feb 9

1. 给一堆职员及其老板的关系，要求实现两个query： A. 输入职员姓名按顺序返回其老板chain B. 输入老板姓名，按顺序返回职员chain

思路：
看起来像常规的建图和搜索

2. 给一个长长的String，String里面有一些char是separator，给一个isSeparator函数可以调用判断是不是separator。separator将该String分成若干部分，判断每部分是不是一个valid password。定义一个valid的password就是长度在6-10的string，包含两个以上的数字和字母。求所有valid password的数目。

思路：Two Pointers

3. google doc 有个comment 功能，每个comment是个tuple（int A，int B）。A表示是指向正文中哪一行的comment, B表示当前comment实际显示在正文对应的哪一行。要求实现用户在前一行插入一个comment后如果space不够了，后面的comment自动下移。

4. 国王住在King's Landing, 他手下有很多的信使，他要发布一项消息给所有的kingdoms，派信使去送信。问通知完所有的kingdoms所需要花费的最长时间。

5. 给两个文件，一个文件A很长，另一个文件B记录的是bad words。要求把文件A里面所有match的文件B里面的bad words替换成掉。

Feb 8

(Provider: anonymous， 匹兹堡office)
第一题：
假设一条街上有多个block，一个block上有多个建筑。求一条街上离几个最近的特定建筑的最远距离最短的block。例子： street = [
        [“store”, “school”, “museum”],
        [“hospital”, “restaurant”],
        [“school”, “restaurant”],
        [],
        [“museum”]],
requirement = [“store”, “museum”, “restaurant”].
第一个block到最近store是0，到最近museum是0，到最近restaurant是1，所以它的max是1

第二个block到最近store是1，到最近museum是1，到最近restaurant是0，所以它的max是1
第三个block到最近store是2，到最近museum是2，到最近restaurant是0，所以它的max是2
第四个block到最近store是3，到最近museum是1，到最近restaurant是1，所以它的max是3
第五个block到最近store是4，到最近museum是0，到最近restaurant是2，所以它的max是4
所以返回的block是第一个和第二个。

思路：遍历street建立map，key是建筑名字，value是list of integer representing block index。然后再遍历一次street，对requirement中的每个建筑，找当前block是否在list里，如果在就删掉之前所有的index（？？？），如果不在就取最近的两个距离中的min。时间复杂度O(m*n)，m为block个数，n为requirement个数。

```
String[][] street;
Map<String, TreeSet<Integer>> map: <name, list of block index>
int res = Integer.MAX_VALUE;
List<Integer> resBlocks;
for (int i = 0; i < street.length; i++) {
        int dist = 0;
        for (String name : requirement) {
                TreeSet<Integer> index = map.get(name);
                Integer floor = index.floor(i), ceiling = index.ceiling(i);//log(frequency)
                if (floor == null) {
                        dist = Math.max(dist, ceiling - i);
                } else if (ceiling == null) {
                        dist = Math.max(dist, i - floor);
                } else {
                        dist = Math.max(dist, Math.min(i - floor, ceiling - i));
                }
        }
        if (dist < res) {
                res = dist;
                resBlocks = new ArrayList<Integer>();
                resBlocks.add(i);
        } else if (dist == res) {
                resBlocks.add(i);
        }
}
return resBlocks;
```

（correct my code if I’m wrong 有没有更好的方法？）

用range做，相当于在一个string中找到包含所有指定字符的最短substring，在一个包含所有requirement的range中，range长度越短，max越小。

先遍历一次street，建立这样一个结构：unordered_map<string, set<int>>, 即给每种建筑建立一个有序表（二叉排序树）；然后遍历block坐标，对每个坐标，在上面的哈希表里对每种建筑对应的排序树里二分找上下界，这样就能log复杂度计算到每种建筑的最近距离

Similar:

Given s and t, find i such that s[i] = 0 and t[i] = 0, and minimize distance{i between 1 in s} + distance{i between 1 in t}

int[] s = {0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0};

int[] t = {0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0};

result: i could be 2 or 7 or 12

简化的这题可以用快慢指针O(2N)吗 虽然情况比较复杂

第二题：

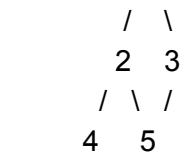
给一个路径列表，找出每个路径在列表里的parent路径。例子：exports = [“/a/b/c”, “/abc/foo”, “/a”, “/abc”, “/a/b”, “/foo/abc”]，return [[“/a/b”, “/a”], [“/abc”], [], [], [“/a”], []]

思路：建立trie，在node里有map of children和boolean isValid。然后遍历exports，到最后一个node把isValid设为true。再遍历exports，如果碰到isValid为true的parent node就把当前string加入结果。时间复杂度O(m*n)，m为路径个数，n为每个路径里“/”的个数。

第三题：

删除二叉树中多余的一个edge。假设input是root node，而且删完edge后还是root。

例子：



删除2-5或3-5都可以

follow up: 如何测试？

思路：从root开始遍历，建立set保存visited node，如果碰到有node的儿子已经在set里就删除这个edge。（lc 685的超级简化版）

第四题：

面经题，给定宽为w高为h的screen，和一句话string，还有字体的上限和下限，问能使这句话fit进screen的最大字体。有两个写好的api分别是getWidth(Character c, int font), getHeight(int font)。

思路：二分法找median font，然后parse string看能不能fit进screen。

Feb 7, 2019

刚刚面完的Onsite

new grad SETI Sunnyvale

lc 484 find permutation

第一轮：

印度小哥

第一题：给一个pattern “IIDDI”, D代表decrease, I代表increase，找出能满足这个pattern的由数字1到n组成的值最小的Permutation，n=pattern.length+1，比如上面这个例子结果就是125436

思路：
先将原数组升序排列，满足了所有的I, 然后对于连续的D，全部reverse

code:
```
class Solution {
    public int[] findPermutation(String s) {
        int len = s.length();
        int[] res = new int[len + 1];
        for(int i = 0 ; i < res.length ; i++) {
            res[i] = i + 1;
        }
        for(int i = 0 ; i < len ; i++) {
            if(s.charAt(i) == 'D') {
                int j = i;
                while(j < len && s.charAt(j) == 'D') j++;
                reverse(res, i, j);
                i = j - 1;
            }
        }
        return res;
    }
    private void reverse(int[] arr, int left, int right) {
        int l = left, r = right;
        while(l < r) {
            swap(arr, l++, r--);
        }
    }
    private void swap(int[] arr, int i, int j) {
        int tmp = arr[i];
        arr[i] = arr[j];
        arr[j] = tmp;
    }
}
```

LC622二叉树找最多siblings的层

第二题：给一个二叉树，找maximum size of level which has the largest number of siblings。这里sibling的定义就是从这一层的左端点到右端点的所有存在或者不存在的node，也就是假设这中间的Parent都有左右孩子的情况下，从左端点到右端点总共有多少个。

思路：
利用完全二叉树的index性质，按层遍历，记录每层起始node的index和末尾的index，相减后得到的距离，然后更新最大值

code：
Provider: null

```
private class MyNode {
      int idx;
      TreeNode node;
      public MyNode(TreeNode node, int idx) {
            this.node = node;
            this.idx = idx;
      }
}
public int maxSiblings(TreeNode root) {
      if(root == null) return 0;
      int res = 0;
      Queue<MyNode> queue = new LinkedList<>();
      queue.offer(new MyNode(root, 1));
      while(!queue.isEmpty()) {
            int size = queue.size();
            int first = queue.peek().idx;
            while(size-- > 0) {
                  MyNode curr = queue.poll();
                  if(curr.node.left != null) {
                        queue.offer(new MyNode(curr.node.left, curr.idx *
      2));//防止溢出的方式就是id*2变成(id - first) * 2，把绝对计算变成相对
                  }
                  if(curr.node.right != null) {
                        queue.offer(new MyNode(curr.node.right, curr.idx
            * 2 + 1));
                  }
                  res = Math.max(res, curr.idx - first + 1);
            }
      }
      return res;
}
```

棋盘上铺多米洛骨牌

lc 790 follow up poj 2446
第二轮：
中国大哥
给一个2*n的board，每个格子有两种情况，为空或者被block了。现在有大小为1*2的多米诺骨牌，问这个board最多能放多少个骨牌

思路：

动态规划
dp[i] 表示[0, i] 的最多骨牌
case 1: i列上没有block
	dp[i] = dp[i - 1] + 1;
	case 1.1: 如果dp[i-1]也没有block
		dp[i] = max(dp[i], dp[i-2] + 2)
case 2: i列上有一个block
	dp[i] = dp[i-1]
	case 2.1: 如果i-1列上没有block or i-1上有一个block在同侧
		dp[i] = max(dp[i], dp[i-2] + 1)
case 3: i列上有两个block
	dp[i] = dp[i - 1]

followup: board的size为m * n

思路：
二分图的最大匹配,匈牙利算法
对于一张二维矩阵，我们可以将所有cell分为两个点集，(x, y), x+y为奇数，x+y为偶数。
其中x+y为奇数被4个x+y为偶数包围，x+y为偶数被4个x+y为奇数包围，所以一条多米诺骨牌其实就是链接两个点集中的一条匹配边，这个题目就可以转化为删除两个点集中的一些点，求剩下点集的最大匹配，用匈牙利算法。

code
provider: null

```
    // board[i][j] = 1 -> blocked, 0 -> unblocked
    List<List<Integer>>point;
    int rows;
    int cols;
    int[][] board;
    int[] link;
    boolean[] used;
    public int chessBoard(int[][] board) {
    // init parameters
      this.board = board;
      rows = board.length;
      cols = board[0].length;
      point = new ArrayList<>();
      link = new int[rows * cols];
      used = new boolean[rows * cols];
      for(int i = 0 ; i < rows * cols ; i++) {
            point.add(new ArrayList<>());
      }
      for(int i = 0 ; i < rows ; i++) {
```

```
      for(int j = 0 ; j < cols ; j++) {
            if(i - 1 >= 0 && board[i-1][j] == 0) {
                  point.get(i * cols + j).add((i - 1) * cols + j);
            }
            if(i + 1 < rows && board[i + 1][j] == 0) {
                  point.get(i * cols + j).add((i + 1) * cols + j);
            }
            if(j + 1 < cols && board[i][j + 1] == 0) {
                  point.get(i * cols + j).add(i * cols + (j + 1));
            }
            if(j - 1 >= 0 && board[i][j - 1] == 0) {
                  point.get(i * cols + j).add(i * cols + (j - 1));
            }
      }
  }
  Arrays.fill(link, -1);
  // hangary algorithm
  return hangary();
}
boolean find(int x) {
  for(int i = 0 ; i < point.get(x).size() ; i++) {
      int vertex = point.get(x).get(i);
      if(used[vertex]) continue;
      else {
            used[vertex] = true;
            if(link[vertex] == -1 || find(link[vertex])) {
                  link[vertex] = x;
                  return true;
            }
      }
  }
  return false;
}
int hangary() {
  int ans = 0;
  for(int i = 0 ; i < rows ; i++) {
      for(int j = 0 ; j < cols ; j++) {
            if(board[i][j] == 0) {
                  Arrays.fill(used, false);
                  if(find(i * cols + j)) {
                        ans++;
                  }
            }
      }
  }
```

```
    return ans;
  }
```

判断表达式是否合法

第三轮：
中国小哥
第一题：给一个表达式算术1+2*(3-4)，判断它是否合法，可能出现的字符就是数字、运算符、括号和其他非法字符。先要求提出一些比较模糊的case比如(0)、+5然后自己定义它们是否合法，然后问如果有人写好这么一个程序，你会用哪些test case来测试它，最后implement。

思路：
感觉就是考想出来所有corner cases

第二题：有一个填满了的数独board，已知有且只有一个数字填错了，且这个数字还是在1-9之间，找出这个填错的数字的坐标。

log start log finish (频率 8)

第四轮：
中国小哥
log timestamp那道，实现start(id, ts), end(id, ts), 按start time顺序打印log
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

思路：
和lru差不多， hash map + double linked list维护顺序
Node {
	String id;
	long start;
	long end;
}
Map<String , Node> map

start 的时候加入map，设置好开始时间，end时间还没有设置
finish的时候设置好node的end值，然后从map中移除
打印的话直接遍历一遍dll即可

code
Provider: Xing

```
public class MyLogger implements Logger {
	private class Node {
		String id;
		long start;
		long end;
		Node prev;
		Node next;
		public Node(String id, long start) {
			this.id = id;
			this.start = start;
			end = -1;
		}
	}
	Node head, tail;
	Map<String, Node> map;
	public MyLogger() {
		head = new Node(“”, -1);
		tail = new Node(“”, -1);
		head.next = tail;
		tail.prev = head;
		map = new HashMap<>();
	}
	@Override
	public void started(String id, long time) {
		Node curr = new Node(id, time);
		map.put(id, curr);
		add(curr);
	}
	@Override
	public void finished(String id, long time) {
```

```
            Node curr = map.get(id);
            if(curr != null && curr.end == -1) {
                  curr.end = time;
                  map.remove(id);
            }
      }
      public void print() {
            Node curr = head.next;
            while(curr != tail) {
                  if(curr.end != -1) {
                        System.out.println(curr.id + “start at” +
            curr.start + “ end at ” + curr.end);
                  }
                  curr = curr.next;
            }
      }
      private void add(Node curr) {
            if(curr == null) return ;
            curr.next = tail;
            curr.prev = tail.prev;
            tail.prev.next = curr;
            tail.prev = curr;
      }
}
```

Feb 5

https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=478393&extra=page%3D2

第四轮 直接扔给我一张纸，一开始看到题目是懵逼的，输入是一堆线段，找所有可以形成的正方形的数量，自己设计数据结构和算法。clarify之后如下，平面无限大，线段都是水平或者竖直，不会cross，但端点可能在另一个线段上。解法大概是枚举对角线，然后check四条边，但是不能直接check每一条边是不是在input 的线段当中，因为可能正方形的一条边是由多个input线段组成的，或者完全被包含在某一个线段当中，所以需要对所有x和y值建segment tree，然后每一个查四条边是不是完全被当前坐标的segment tree cover

有没有同学可以分享一下思路,如何用线段树判断一段距离都被cover了

思路：不用线段树的操作，建立两个并查集来merge水平和垂直方向上坐标相同的点。 初始状态是一堆输入的线段就可以认为是并查集的初始状态。

Feb 3

原帖

删除node返回森林

第一轮是给一个二叉树，然后再给一个list的TreeNode，这个list里的TreeNode是肯定在二叉树里的，然后要把这些TreeNode删除，最后返回一个原二叉树删除这些TreeNode后得到的Forest
举个例子，如果二叉树如下：

```
     A
    /  \
  [B]   C
  /  \    \
 D   E    [F]
```

方括号内的TreeNode是要删除的，最后返回的Forest是[A, D, E]。
这道题我用recursion做的，输入参数一个是TreeNode，一个是它的parent是不是被删除的boolean。然后要我给出不同的test case

思路：
简单的dfs
如果当前节点需要被删除，递归进入它的左右子树得到左右子树被删除节点后的新树，如果不为null则加入list，然后由于当前节点被删除了，返回null
如果当前节点不需要被删除，左右子树递归，然后返回当前节点

code:
Provider: null

```
public TreeNode dfsUtil(TreeNode root, Set<TreeNode> deleted,
List<TreeNode> res) {
      if(root == null) return null;
      if(deleted.contains(root)) {
            TreeNode left = dfsUtil(root.left, deleted, res);
            TreeNode right = dfsUtil(root.right, deleted, res);
            if(left != null) res.add(left);
            if(right != null) res.add(right);
            return null;
      } else {
            root.left = dfsUtil(root.left, deleted, res);
            root.right = dfsUtil(root.right, deleted, res);
            return root;
      }
}
```

类似LC 981？ - Google Snap （高频）

第二轮是面经高频题，Google Snapshot
就是一个array，然后这个array有不同的版本数，然后有不同的函数，比如set(int index, int val), get(int index, int version), snapshot()。比如说我先set(1,1)，set(0,0)，set(2,2)，这样我有一个[0,1,2]，然后snapshot()之后系统里会记住version 1的array是[0,1,2]，然后set(1,0)，array就变成[0,0,2]，这个时候如果get(1,1)得到的是1，但get(1,2)得到的是0。
思路：

1. 模拟过程，用map存下snapshot即可,主要要和面试官交流得到各种异常的处理方式包括数组长度是否固定，idx是否一定合法，version不存在是返回最近的还是抛出异常等等

2. hash map嵌套treemap，牺牲了一些时间，但是如果一些idx在某些version一直没用过可以节省空间

code：
Provider: null

```
class Snapshot {
	int version = 1;
	int length;
	Map<Integer, TreeMap<Integer, Integer>> map = new HashMap<>();
	public Snapshot(int length) {
		this.length = length;
		for(int i = 0 ; i < length ; i++) {
			map.put(i, new TreeMap<>());
		}
	}
	public void set(int idx, int val) {
		// sanity check and exception process
		map.get(idx).put(version, val);
	}
	public int get(int idx, int version) {
		// sanity check and exception process
		Integer key = map.get(idx).floorKey(version);
		if(key == null) return Integer.MIN_VALUE;
		return map.get(idx).get(key);
	}
	public void snapshot() {
		version++;
	}
}
```

这里用了treeMap。题意中是否要求如果输入了不存在的版本号，那么就返回这个版本号的lower_bound？如果没有这个要求那么treeMap是能用hashMap取代的

其实treemap里面存的是每个版本改变的值，也就是说，如果版本2里面没有这个，说明版本2这个key的值没有变，还是之前版本1的值，所以必须要找到floor key。说版本好像不太对，还是snapshot比较贴切

LC Car Fleet (频率 8)

第三轮比较简单，给一个Integer，n，形成一个List<Integer>，包含1-n的所有元素，每个元素代表一辆车的车速，最后会形成车队，要返回的是每个车队的车数。举个例子，[2,4,1,3]，最后返回的是[2,2]，因为前两辆车会被第一辆车的车速2所限制，而后两辆车则会被第三辆车的车速1限制。再举个例子，[1,2,3,4,5]，最后返回[5]。[2,4,5,3,1]，返回[4,1]。
Follow-up是在原List<Integer>插入n+1，然后返回所有的车队车数可能性。最后问了算法复杂度

思路：
感觉比原题 car fleet还要简单一些，直接从前往后扫一下，如果cars[i-1] < cars[i], cars[i] = cars[i-1]，然后分别统计连续的数字个数

切矩阵

第四轮比较难，出了我不大擅长的dp，说给一个m x n的矩阵，可以横着切或者竖着切，每横着切一刀的cost为被切矩阵的列数的平方，每竖着切一刀的cost是被切矩阵的行数的平方。然后给一个desired area，问切到这个area大小最少的cost是多少。好像切好以后两个矩阵都能继续切，差点没做出来，还是弱鸡啊

输入为矩阵的长和宽和一个int代表desired area

思路：
dfs带memorization，一刀切成两半，在其中一半找面积为k的矩形，在另一半找面积为desired area - k的矩形，然后每一刀有两种切法，横着切，竖着切，分别遍历

code:
Provider: null

```
    Map<List<Integer>, Integer> memo = new HashMap<>();
    public int minCost(int rows, int cols, int area) {
      // sanity check
      return dfsUtil(rows, cols, area);
    }
    private int dfsUtil(int rows, int cols, int area) {
      if(area == 0) return 0;
      if(rows == 0) return 10000; // some big integer will mask the
result in min() function
      if(cols == 0) return 10000;
      if(rows * cols == area) return 0;
```

```
        if(rows * cols < area) return 10000;
        if(memo.containsKey(Arrays.asList(rows, cols, area))) {
            return memo.get(Arrays.asList(rows, cols, area));
        }

        int cost = Integer.MAX_VALUE;
        for(int k = 0; k <= area; k++) {
            // cut row
            int tmp = (int) Math.pow(cols, 2);
            for(int i = 1 ; i <= rows / 2 ; i++) {
                cost = Math.min(cost, tmp + dfsUtil(i, cols, k) +
dfsUtil(rows - i, cols, area - k));
            }
            // cut cols
            tmp = (int) Math.pow(rows, 2);
            for(int j = 1 ; j <= cols / 2; j++) {
                cost = Math.min(cost, tmp + dfsUtil(rows, j, k) +
dfsUtil(rows, cols - j, area - k));
            }
        }
        memo.put(Arrays.asList(rows, cols, area), cost);
        return cost;
    }
```

这题需要用DFS搞分治法吗？给定一个面积，你可以枚举出所有的长*宽组合，对原矩阵，只要对某个长宽能切割，最多两刀就能把目标矩形切出来，只是先横切还是先竖切的代价可能不同罢了。可能我理解错了题意 （editor：最后想要的面积并不一定是一个规则的矩形）

这题切的具体含义可以解释的更清楚一点么？是必须平均分，还是按照整数切还是随意的位置切呢？

Feb 2

原帖

最大化连续子串最小sum (LC 410)

有一个数组，分成K部分，要求， min of subarray sum is maximized， [1,2,4,3,3] 分成3部分，你可以分成 1| 2,4|3,3 最小数组 是1，但是你分成 1,2 |4|3,3 最小的数组sum就变成了3. 3是想要的答案。 这个我提出了DP和recursive, 面试官都不满意, 最后用的是二分查找. 比如说你给 一个数 N , 看这个数组能不能被分成K部分, 每个部分都大于N， 如果不能，就试试N/2。

思路：

如题主描述

code：

Feb 1

矩阵填字符

一个矩阵，里面有空字符和abc等character，用最近的字符填满矩阵。
A ‘’ ‘’ ‘’ ‘’
‘’ ‘’ ‘’ ‘’ ‘’
‘’ ‘’ ‘’ B ‘’
空字符填上最近字符，有tie的话随便选一个
follow up：char比空字符多

思路：
先扫一遍得到有哪些字符，然后对每个字符best-first search, 用min heap当queue，heap里面装
Node{
int i; int j;
char c; int distance;
}
根据distance来决定优先级，距离小的在前面，每次poll出来一个新node看那个位置上有字符没有，没有的话填上，有的话忽略

code
Provider: null

```
   public void fillMatrix(char[][] matrix) {
      int rows = matrix.length;
      int cols = matrix[0].length;
      Queue<Node> pq = new PriorityQueue<>((a, b) -> a.dis - b.dis);
      for(int i = 0 ; i < rows ; i++) {
            for(int j = 0 ; j < cols ; j++) {
                  if(matrix[i][j] != ' ') {
                        pq.offer(new Node(i, j, matrix[i][j], 0));
                  }
            }
      }
      int[][] dirs = new int[][] {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
      while(!pq.isEmpty()) {
            Node curr = pq.poll();
            if(matrix[curr.i][curr.j] == ' ') {
                  matrix[curr.i][curr.j] = curr.c;
            }
```

```
            // generate
            for(int[] dir : dirs) {
                  int x = curr.i + dir[0];
                  int y = curr.j + dir[1];
                  if(isValid(matrix, x, y) && matrix[x][y] == ' ') {
                        pq.offer(new Node(x, y, curr.c, curr.dis + 1));
                  }
            }
      }
    }
    private boolean isValid(char[][] m , int i, int j) {
      return i >= 0 && i < m.length && j >= 0 && j < m[0].length;
    }
    private static class Node{
      int i; int j; char c; int dis;
      public Node(int i, int j, char c, int dis) {
            this.i = i;
            this.j = j;
            this.c = c;
            this.dis = dis;
      }
    }
```

Jan 31

[原帖]

Merge K Sorted Lists + Meeting Room II

第一轮：
1. 输入一些subiterators, 合成一个superiterator, 都是升序的. 类似LC 23 Merge k Sorted Lists. heap搞定.
2. LC 253 Meeting Rooms II

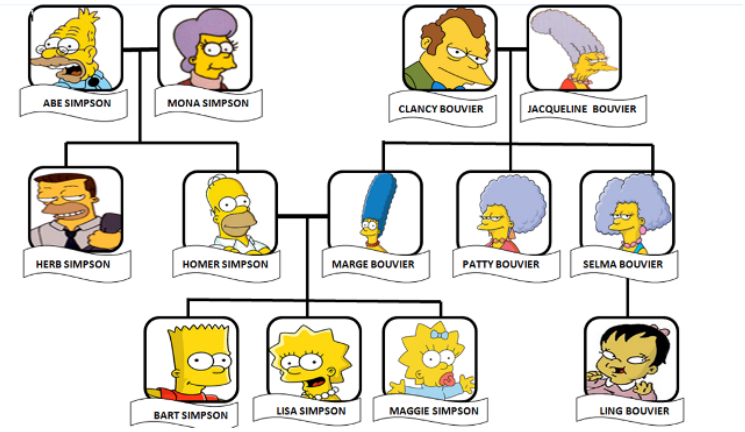
家庭树判断亲属关系

第二轮：
输入一些 parent-child pairs, 然后给一个pair, 判断他们是否genetically related. 比如(p1, c1)表示p1生了c1, c1有p1的gene. 输入(p1, c1), (p2, c1), (p1, c2), (p2, c2). 画图可知(c1, c2)是related, (p1, p2)不是related. 面试官给了一个无向图, 结点就是p1, p2, c1, c2, 根据输入把边加上, 然后我就一直按无向图来做了, 觉得从一点dfs遍历能看到另一个点就行了. 其实这应该是有向图. 如果是

无向图, p1通过c1可以到达p2, 但其实不应该. 有向图就不会有问题. 后来发现这其实很像树, 这题就是要找common ancestor, 如果有就是related. 把一个点的所有ancestor放到一个set里, 另一个点的放到另一个set里, 有交集就代表related. 这题最大的失误就是想复杂了, 想成general graph了, 其实根据遗传的知识来想要简单一些, 更像是树. 代码写出来了, 面试官说有bug, 但我觉得不是. 由于下一个面试官已经在外面等着了, 就没继续解释了. 事后想想应该说明白, 也就一两分钟的事儿, 面过试的应该都能理解. 最好还是不要有bug吧?

Family Tree
在family tree中找两个人是否是亲戚。follow up如果是亲戚的话，找lowest common ancestor。
第三题大概这样
```
class Node {
int val,
Node father;
Node mother;
}
public boolean isRelated(Node p1, Node p2);
follow up:
public Node findLowestCommonAncester(Node p1, Node p2);
```
其实这题有很多followup：
1. 假设现在人分布于各个国家，比如，中国，澳大利亚，美国，那么有什么优化可以做
    a. 这是我当年的onsite问题，估计凉在这里

Example ：



楼主做法 ： 开始我考虑是parent指向child，然后稍微一比划不对，没让我多想面试官提示我反过来，让child指向parent，这样就是求两个node到leaf的路径中有没有交集了，dfs做出来。
帖子： https://www.1point3acres.com/bbs/thread-201712-1-1.html

路人建议 ：以这个人为root节点，以父母为left和right，向上建树就是二叉树了
然后query的两个人就是两个root，对这两个二叉树进行BFS，找到最浅的公共节点n，n在tree1z和tree2的深度相加即可

这个follow up的一个关键点是找到lowest common ancestor就停了，所以用BFS比较好，不要把某一个人的ancestors都找出来了再去跟另一个人的ancestor找match.

Code:
Provider: null
```
class Node {
```

```
      int val,
      Node father;
      Node mother;
}
public boolean isRelated(Node p1, Node p2) {
      Set<Node> ancesters = new HashSet<>();
      // BFS
      Queue<Node> queue = new LinkedList<>();
      queue.offer(p1);
      while(!queue.isEmpty()) {
            Node curr = queue.poll();
            if(curr.father != null) queue.offer(curr.father);
            if(curr.mother != null) queue.offer(curr.mother);
            ancesters.add(curr);
      }
      queue = new LinkedList<>();
      queue.offer(p2);
      while(!queue.isEmpty()) {
            Node curr = queue.poll();
            if(ancesters.contains(curr)) return true;
            if(curr.father != null) queue.offer(curr.father);
            if(curr.mother != null) queue.offer(curr.mother);
      }
      return false;
}


follow up:
public Node findLowestCommonAncester(Node p1, Node p2) {
      Set<Node> ancesters = new HashSet<>();
      // BFS
      Queue<Node> queue = new LinkedList<>();
      queue.offer(p1);
      while(!queue.isEmpty()) {
            Node curr = queue.poll();
            if(curr.father != null) queue.offer(curr.father);
            if(curr.mother != null) queue.offer(curr.mother);
            ancesters.add(curr);
      }
      queue = new LinkedList<>();
      queue.offer(p2);
      while(!queue.isEmpty()) {
            Node curr = queue.poll();
            if(ancesters.contains(curr)) return curr;
            if(curr.father != null) queue.offer(curr.father);
            if(curr.mother != null) queue.offer(curr.mother);
      }
      return false;
```

}

Top k的value， 受label限制

第三轮：
value: 9， 8， 6, 8, 7.
label: A, A, B, A,B
输入一些node, node里有value和label, 找出K个value最大的, 但每个label不能超过M个. 比如K=3, M=2, 答案是 9， 8, 7. 8 > 7, 但是A的数量不能超过2.

A, A, B A B
先sort by value再从大到小scan， 复杂度O(nlogn).
面试官又写了另外三个其它可行的复杂度, A:O(n), B: O(nlogk), C: 不记得了. 问我想再试试哪个, 我选了O(nlogk), 感觉像是heap. 一开始想错了, 面试官直接说这样不行, 很多人都这样想过但就是不对. 然后就是全程带着我做, 还让我把算法的步骤写在白板上, 说是思路更清楚. 我当时都有点儿懵,没见过这样的... 再加上还在想着上一轮bug的事儿, 精力有点不太集中. 后来想出来了, 也不难. 对每个label用heap选出M个最大的, 然后再从这些用heap选出K个最大的, 最后就是O(nlogk). 中午带着吃饭的大哥说这人好像是欧洲那边的, style确实不太一样, 有点儿严厉, 但还有点儿可爱, 挺有意思. 讨论完时间不多了, 就让写了O(nlogn)的解法, 简单问了问如何测试.

思路：
如题主描述

code

```
class Solution:

    def topk(self, elements, k, m):
        '''

        :param elements: [val, label]
        :param k:
        :param m:
        :return:
        '''

        # O(nlogn)
        def _topk0(elements, k, m):
            elements = sorted(elements, reverse=True)
            ans = []
            labels = {}
            for val, lb in elements:
                if labels.get(lb, 0) >= m:
                    continue
                ans.append(val)
```

```
            labels[lb] = labels.get(lb, 0) + 1
            if len(ans) == k:
                break
        return ans

    # O(nlogk)
    # heap for each label(len == m)
    # merge heap
    def _topk1(elements, k, m):
        from collections import defaultdict
        import heapq
        labels = defaultdict(list)

        for val, label in elements:
            pq = labels[label]
            heapq.heappush(pq, val)
            if len(pq) > m:
                heapq.heappop(pq)

        que = []
        for pq in labels.values():
            for e in pq:
                heapq.heappush(que, e)
                if len(que) > k:
                    heapq.heappop(que)
        return que

    # O(n)
    # 用quick select，对每个label都求出第m大，然后再merge成一个新的数组，求第k大
    def _topk2(elements, k, m):

        def quick_select(arr, lo, hi, p):
            i, j, k = lo, lo, hi - 1
            pi = arr[p]
            while j <= k:
                if arr[j] == pi:
                    j += 1
                elif arr[j] > pi:
                    arr[i], arr[j] = arr[j], arr[i]
                    i += 1
                    j += 1
                else:
                    arr[j], arr[k] = arr[k], arr[j]
                    k -= 1
            return i

        def kth(nums,  m):
            lo, hi = 0, len(nums)
            while lo < hi:
```

```
            mid = (lo + hi) >> 1
            mid = quick_select(nums, lo, hi, mid)
            if mid == m - 1:
                break
            elif mid < m - 1:
                lo = mid + 1
            else:
                hi = mid

        from collections import defaultdict
        labels = defaultdict(list)
        for val, label in elements:
            labels[label].append(val)

        ans = []
        for nums in labels.values():
            kth(nums, m)
            ans.extend(nums[:m])

        kth(ans, k)
        return ans[:k]

    ans0 = _topk0(elements, k, m)
    ans1 = _topk1(elements, k, m)
    ans2 = _topk2(elements, k, m)
    assert( sorted(ans0) == sorted(ans1) == sorted(ans2))

import random
for _ in range(100):
    elements = []
    for i in range(30):
        elements.append([random.randint(1, 100), random.randint(0, 4)])
    k = random.randint(5, 15)
    m = random.randint(2, 5)
    so = Solution()
    so.topk(elements, k, m)
```

第四轮：

中国小哥哥。
4, 5, 6, 8
5, 5, 6, 9
6, 7, 9, 10
红色表示目的地. 问题是要找出能到所有目的地的点. 可以从其它任何点出发, 上下左右四个方向,, 只能往小于等于当前值的点挪(8可以去6, 但6不可以去8, 5可以去5). 图中绿色点就是答案. 刚开始对题目有一

些问题, 后来小哥哥给了提示, 说是目的地不多. 后来就决定反着来了, 看从目的地开始能反着到哪些点, 存成set. 再用每一个点试, 如果在所有的set里就表示从这个点出发能到所有的目的地. 假设k个目的地, m行n列, 复杂度是O(kmn). 然后问如何并行. 对每个目的地都可以单独做, 我以为是O(mn), 但其实复杂度最高的地方是最后的test部分. 把所有的set放到一台机器上复杂度还是O(kmn). 类似merge sort, set也应该两两merge. 最后能优化到O(logkmn). 谢谢小哥哥了.

第五轮：
模拟抓娃娃机。
color count
Green: 3 3个绿娃娃
Blue: 5
Red: 2
随机抓, 哪个颜色的数量多, 那个颜色被抓的概率也相应的高. 抓完要把相应颜色的数量减去1.
类似LC 528 Random Pick with Weight, 但不同点在于每次要减去1. 开始用数组, 二分查找O(logn), 更新O(n). 面试官没让写O(logn)的, 写个O(n)的就行. 写完了说想想能不能把更新的部分也优化一下. 我知道要用segment tree, 但想不起来了, 还是不太熟练. 主要是一直在想用用数组实现的方式, 卡在坐标变换上了. 面试官提示说还是用tree node吧, 好想一些. 请参考: https://leetcode.com/problems/ra ... n-with-segment-tree . 我觉得应该也把左右子树的sum放在node里. 面试官就是看看我怎么想的, 说是知道不太好写.

Jan 28

[原帖]

Tree Isomorphism Problem LC951(频率7)

1.一开始是个白人老大哥，跟我闲聊了差不多10分钟，介绍了他们组的项目，问我对于google什么项目感兴趣，然后看时间差不多了，出题
题目是check which two tree are similar, 就是一个function， 就是 1 - 3 - 2 和 1 -2 -3 就是similar ， 然后follow up问我recursion的runtime,
我太紧张了硬是没有答出来 具体题目看连接
https://www.geeksforgeeks.org/tree-isomorphism-problem/ 估计我就是挂在这里

思路：
dfs，经典题目

两个string找多出来的char

2.第二题是个东南亚老大哥 亚裔，贼严肃，也不说话，上来就说题目， 然后题目是 给你两个string s1 s2, s2会比s1多一个char,
要你找出来那个char的index 比如 today, todxay, return 3

O(1) space better than O(N) time,我想了半天，巧妙一点的二分，follow up是如果两个string 乱序怎么办，就先sort

思路：
如果没有duplicate的话，根据时间空间复杂度要求自然联想到二分查找：
在长的string上二分
mid = l + (r - l) / 2
每次比较长的mid位置上的char和短的mid位置上的char是否一样，因为最多只会被插入一个字符，一样的话，说明左边部分没被插入东西，target一定右边，否则target在左边

注意：空串需要特殊处理，不然charAt会抛异常

如果有duplicate，感觉无法better than O(N)，一个极端例子ddddddddddd, ddddddddddde,如果不check每个位置的话，无法找到多余的那个字符

code
Provider: null

```
public int findExtra(String s1, String s2) {
   if(s1.length() == 0) return 0;
   if(s2.length() == 0) return 0;
   String longer = s1.length() > s2.length() ? s1 : s2;
   String shorter = s1.length() > s2.length() ? s2 : s1;
   int l = 0, r = longer.length() - 1;
   while(l + 1 < r) {
         int mid = l + (r - l) / 2;
         if(longer.charAt(mid) == shorter.charAt(mid)) {
               l = mid;
         } else {
               r = mid;
         }
   }
   if(longer.charAt(l) != shorter.charAt(l)) return l;
   else return r;
}
```

```
def find_extra(str1, str2):
    if len(str1) > len(str2):
        str1, str2 = str2, str1
    lo, hi = 0, len(str1)
    while lo < hi:
        mid = (lo + hi) >> 1
        if str1[mid] != str2[mid]:
            hi = mid
        else:
```

```
        lo = mid + 1
    return lo
```

Binary Tree Vertical Order Traversal #LC 314 and LC 987

3.中午一个白人小姐姐带我吃了饭，下午第一轮是个中国大哥，贼亲切，一上来就用中文跟我打招呼，说接下来我们说题目用英文吧
我说好， 题目是 col order tree traversal，就是 一个tree　　1 - 2 -3 root 的col是0 left leaf是-1，right leaf 是1
没有follow up跟我中文闲聊了十分钟，真的很感谢，说是自愿来当面试官，希望为华人也多分担一些，给华人学生多一些机会

思路：

1. preorder标记每个node的vertical和level坐标，用map存，或者自定义类
2. 如果同一个position上的node也有顺序的话还得排序

代码：主要思想是map + BFS，map可以对key进行排序

```
provider: (please fill your name)
        class Solution {
        public:
            vector<vector<int>> verticalOrder(TreeNode* root) {
                vector<vector<int>> res;
                if (!root) return res;
                queue<pair<TreeNode*, int>> q;
                q.emplace(root, 0);

                map<int, vector<TreeNode*>> m;

                while(!q.empty()) {
                    int len = q.size();
                    for (int i = 0; i < len; i++) {
                        auto tmp = q.front();
                        q.pop();
                        m[tmp.second].push_back(tmp.first);
                        if (tmp.first->left) q.emplace(tmp.first->left, tmp.second -
        1);
                        if (tmp.first->right) q.emplace(tmp.first->right, tmp.second +
        1);
                    }
                }

                for (auto ele : m) {
                    res.push_back(vector<int>());
                    for (auto node : ele.second) {
```

```
                res.back().push_back(node->val);
            }
        }
        return res;
    }
};
```

Random Pick With Weight #LC 528

4. 最后一轮来了两个大哥， 有一个是面试官trainee？??, 第一题是 设计一个tree ？？？ 一天三道tree。。。 然后写个function level order traversal
第二题是 给你个rand（） return 0，1之间的随机值，要你随机从一个dataset中取数据， 就是假如说数据里面有300个美国，100个中国，100个韩国，那么
这个function output 美国的概率就是3/5 etc

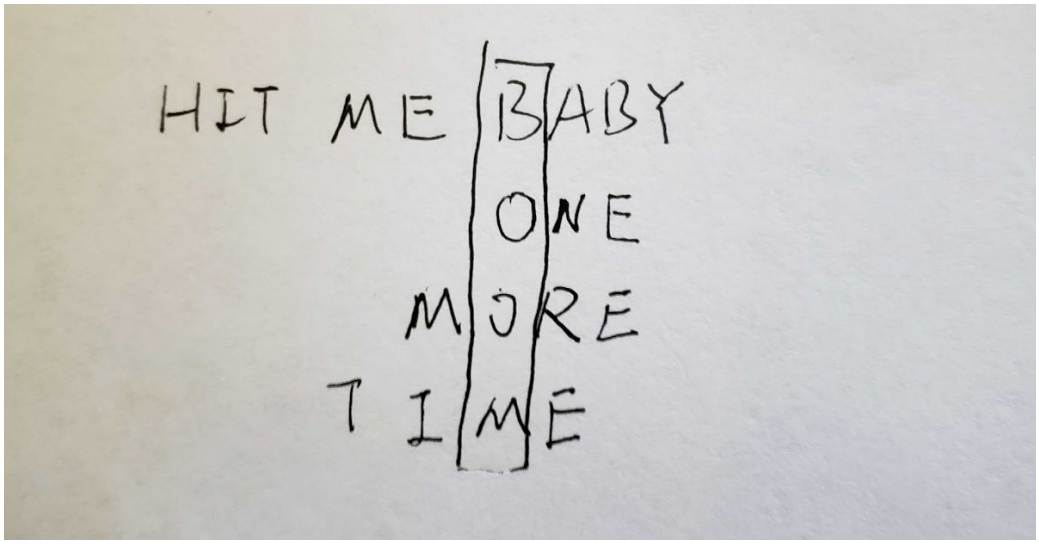
思路：
见lc

Jan 25

王位继承（10+频率）+歌词找单词+sign tree+找重复元素

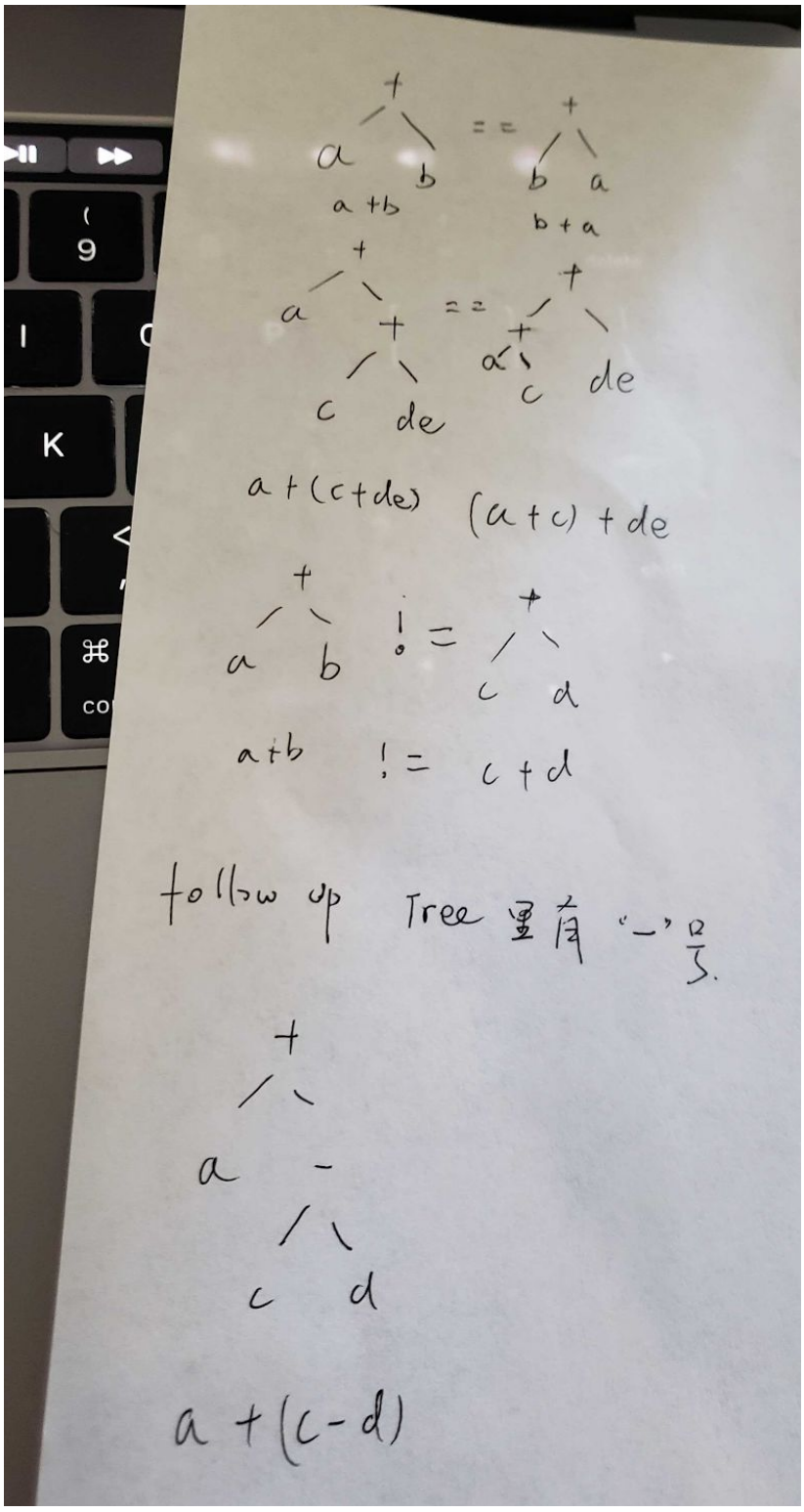
第一轮 王位继承 美国小姐姐 这轮感觉还可以

第二轮 给两个 string 分别是lyric 和 word，问能不能用lyric 构成 word
比如 lyric "HIT ME BABY ONE MORE TIME" word = "BOOM" 那么是可以的
follow up 是优化时间



第三轮 给两个 tree， root1 root2, 问计算结果等不等

tree 里面只有 加号 和 字母， 并且 input 保证valid.
加号就是hashmap统计词频
减号是dfs的时候把sign传进去？



第四轮 LC 287 find duplicate number
给一个数组长度为n+1，包含从1到n到数字，有重复元素，找出重复元素

开始没反应过来 后来才反应过来的。
follow up 就是优化空间，牺牲时间。
最后 他直接说告诉你这个 想看看我写binary search 看我的code implement ability

思路：

1. 初始解法：用set（可以先装一波没见过）
2. 如果数组可以改变，用heap sort可以做到o(1)空间 o(nlogn)时间
3. 如果数组不可以改变，可以提出最优解 - 快慢指针找环和找环的入口，不过这里可能会被问到证明找环的入口的算法正确性，略麻烦
4. 面试官期待的binary search 做法，o(1)空间，O(nlogn)时间,不改变数组
   对值域进行二分[1, n]，如果count出来小于等于mid数字个数比mid多，说明dupliate在mid左边，反之小于等于mid的数字个数等于mid或者比mid少，duplicate在mid右边

Code
Provider: 九章算法

```
public int findDuplicate(int[] nums) {
        int l = 1;
        int r = nums.length - 1;  // n

        while (l + 1 < r) {
            int mid = l + (r - l) / 2;
            if (count(nums, mid) <= mid) {
                l = mid;
            } else {
                r = mid;
            }
        }

        if (count(nums, l) <= l) {
            return r;
        }
        return l;
    }

    private int count(int[] nums, int mid) {
        int cnt = 0;
        for (int item : nums) {
            if (item <= mid) {
                cnt++;
            }
        }
        return cnt;
    }
```

Jan 24

[原帖]

二叉树删除多余边 （频率 13）

第一轮是一个比较严肃的美国小哥，题目不难但我太紧张导致脑子短路: 一个binary tree里有1个extra edge，找到并且fix。我用dfs做的，然后写完小哥让我自己找我的code能不能cover所有的情况, 然后发现并不能，改了几下之后code终于能work。反正这轮磕磕绊绊，要挂就挂在这了。

完全树数node（频率 5）

第二轮是个超酷的打耳钉的美国小姐姐。高频题LC222，先给你一个数，找这个数在不在tree里面，followup找到树的大小。这轮真是使尽各种演技。。。

用路径string找word

第三轮美国小哥，题目很长很绕但交流之后其实很简单，简洁地表述下大概就是：用户在手机屏幕上滑动打字， 给你一串string记录了你手指滑动经过的char，举个例子比如， “hrwcewelwlcfo”, 这个手指滑动可以打出一个”hello”。用户从h开始，手指划过“rwc”, 打出一个“e”, 划过”we“, 打出”l“, 划过w, 打出”l“, 再划过”cf“, 打出“o”. 这样就打出“hello”，很神奇有没有？？然后给你一个路径的string和一个dictionary of words, 问你有多少个word可以被用这个路径打出来。路径的第一个字符和最后一个字符一定是会被打出来的。这题本质很简单基本靠交流。 我说了两种方法，一种从路径入手，排列组合，一个一个放到dictionary里面查；另一种一个一个word拿出来，two pointer比较。比较了两种方法的复杂度，还算了一些数学公式，写了第二种的code。followup是如果在同一个位置可以停留多次怎么做，就是在同一个char上可以重复打好多次。比如“hwelpto”也可以打出“hello”，这里“l”重复打了两次，也不难，秒了。

lc853原题car fleet

第四轮印度哥, 是sweti，题也很简单，一列车队n辆车，速度分别是1到n，随机排列成一条，然后开始开，不能超车，问你无限时间之后return一个车队cluster array，每个数代表cluster里车的个数，要按顺序。比如input【2， 3， 1， 4】返回的是【2，2】。一开始我以为他要问cars fleet那题，没想到问了个更简单的。。。follow up是现在有一辆车speed是n+1，我们把它加入到这个n车车队的空隙中，总共n+1个空隙，然后问你每种加入，最后导致的车队cluster array，就是return一个array of array。写完followup之后讨论怎么test。

Jan 17

House Robber III (频率 5)

思路：见LC337，返回两个结果的recursion函数

Guess word改版

给一个矩阵，在里面找一个gift，不知道gift坐标，然后每次pick一个点，告诉你是更近了还是更远了还是距离不变, 距离是曼哈顿距离，要求用最少的猜测次数找到gift

思路：KD tree算法

	0	1	2	3	4
0					
1					
2					
3		gift			
4					

从（0，0）开始，先二分列坐标，看(0, 4)，如果04比00近，说明gift在j = 2的右边矩阵中，如果04比00远，在j = 2的左边矩阵中，如果相等，说明gift在j=2这条线上
一直二分到 l == r找到gift所在列

然后二分行坐标，直到找到gift

参考代码：
Provider: null

```
    public int guess(int i1, int j1, int i2, int j2) {
      int x = 3;
      int y = 1;
      int dist1 = Math.abs(i1 - x) + Math.abs(j1 - y);
      int dist2 = Math.abs(i2 - x) + Math.abs(j2 - y);
      return dist1 - dist2;
    }
    private void guessGift(int rows, int cols) {
      int left = 0, right = cols - 1;
      while (left < right) {
            int res = guess(0, left, 0, right);
```

```
            int mid = left + (right - left) / 2;
            if(res == 0) {
                  left = mid;
                  break;
            } else if(res < 0) {
                  right = mid - 1;
            } else {
                  left = mid + 1;
            }
      }
      int top = 0, bot = rows - 1;
      while (top < bot) {
            int mid = top + (bot - top) / 2;
            int res = guess(top, left, bot, left);
            if(res == 0) {
                  break;
            } else if(res < 0) {
                  bot = mid  - 1;
            } else {
                  top = mid + 1;
            }
      }
      System.out.println(top + ", " + left);
    }
```

抽搐词(lintcode twitch words)

返回重复超过三次的letter的起点和终点

Example
Given str = "whaaaaatttsup", return [[2,6],[7,9]].

Explanation:
"aaaa" and "ttt" are twitching letters, and output their starting and ending points.

follow up：给一个dic，通过删除生成合法的词语

思路：backtracking

Jan 16

[原帖]

水滴浸树

第一题，是一个台湾小姐姐，给了一颗树状图，每一个节点代表一个水塔，每一条边代表一根水管，每根水管上有一个数字，代表水流过这个水管需要多久。问题是从root开始灌水，需要多久可以灌满整个图。follow up是如果图里有环该怎么做。

思路：
把树当成带权图，然后bfs或者dfs即可

完全树求节点

第二题，是一个北欧小哥，题目是给一颗complete tree，每课树节点只有left，right这两个变量，求特定标号的节点在不在树上。follow up是给定一颗complete tree，求这棵树有多少节点。

思路：
见前文

LRU和LFU

第三题，一个中东小哥，比较简单，竟然是LRU和LFU……，都是老题了，比较幸运。

思路：
见lc

随机数组中用二分法找不到的元素

第四题，一个印度小哥，给定一个没有排序的随机数组，使用二分法找数组里的各个元素，返回有多少元素是用二分法永远找不到的。要求算法复杂度为O(n), 楼主不会。印度小哥让楼主用例子试一试，看看能不能找到什么规律。楼主试了试，没找到规律……..最后给出了nlog（n）的解法，估计最后挂在这了。

思路：
可以用类似于判断树是否为BST的思路：维护一个“值域”R=(lo, hi)（即在当前搜索下标范围[i, j]内，只有在R范围内的数才有可能被二分法找到）。求得当前搜索范围[i, j]的中点mid。如果A[mid]在当前的R中，那A[mid]就一定能被找到。然后分别递归mid的左右两侧[i, mid-1]和[mid+1, j]。递归左侧时，允许的值域是(lo, A[mid])；同理右侧的值域是(A[mid], hi)。如果任何时刻发现值域R已经不包含任何整数，那就表示[i, j]这个下标范围内没有任何能找到的数字。

code
provider: null

```
    private int find(int[] nums) {
      // do some sanity check operations
      return dfsUtil(nums, 0, nums.length - 1, Integer.MIN_VALUE,
Integer.MAX_VALUE);
    }
```

```
    private int dfsUtil(int[] nums, int left, int right, int lower, int
upper) {
      // base case: there should be no elements between (lower, upper) or
(left,right)
      if(lower >= upper) return right - left + 1;
      if(left > right) return 0;

      int mid = left + (right - left) / 2;
      int l = dfsUtil(nums, left, mid-1, lower, nums[mid]);
      int r = dfsUtil(nums, mid+1, right, nums[mid], upper);
      if(nums[mid] > lower && nums[mid] < upper) {
            return l + r;
      } else {
            return l + r + 1;
      }
    }
```

需要和面试官clarify是否包含重复值，如果包括，情况会更复杂一点。首先需要定义二分法是找左边界还是右边界，其次，不同位置同样的数是否当做同样的值。提供一个python版本代码。

```
def bs_in_unsorted_arr(unsorted):
    from collections import defaultdict
    lo, hi = -1 << 32,1 << 32
    left, right = 0, len(unsorted)
    found = defaultdict(bool)
    def bs(left, right, lo, hi):
        if left >= right:
            return
        if lo >= hi:
            return
        mid = (left + right) >> 1
        if lo <= unsorted[mid] < hi:
            found[unsorted[mid]] = True
        bs(left, mid, lo, min(unsorted[mid], hi))
        bs(mid + 1, right, max(unsorted[mid], lo), hi)
    bs(left, right, lo, hi)

    return found
```

Jan 14

Subarray包含所有set里的数

第一轮是 给一个list， 一个set， 问这个list里面有没有一个Subarray能够包含所有set里面的数

找photo id

第三轮是很迷茫的印度大哥。。input output都没说太清楚，就是给一个list of like， 每一个Like包含 photo id和timestamp，然后要我output出所有的photo id，这些photo id需满足，在当时是最大的，同时在24小时里也是最大的。。。贼迷茫

resource picker

第四轮是设计一个resource picker，有两个api getResource() 和Return resource() ,assume有1billion个resource，然后只有一台机器，1GB memory，问如何实现 getResource()和returnResource()

思路：
bitmap,

落雨滴

第五轮是 有一个0-1的线条，然后有一雨点掉下来(position)，这个雨点掉在这个线条上会散开成0.01 m 长的范围，然后设计两个api，一个是掉雨点，一个是查询当前的线条是不是都已经被雨点覆盖了

思路：
见帖子 雨滴落到线条

Jan 10

原帖

机器人左上到右上(频率 10)

第一轮是中国小哥，查了linkedin是staff engineer。。出了一个很简单的dynamic programming的题，就是给一个matrix，从一个grid 到另一个grid，只能往右上，右，右下三个方向走， 求有多少种走法。利口 六四变种

思路：
前文总结

n层楼m个房间关灯

第二轮是一个南亚人，出了一个很有意思的题：一栋楼有n层，每层有m个房间加一个过道，每条过道两侧有楼梯，只能通过两侧的楼梯上下楼。现在这栋楼里有一部分房间开着灯，每走过一个房间的cost是1，现在要把所有的房间的灯关掉，求问minimum cost是多少。 这题可以用动态规划解，也可以用dfs解，思路就是只有两种情况，爬左边的楼梯或者爬右边的楼梯，也就是一个binary tree，所以dfs可以暴力解

思路：

1. 如果不上楼，只下楼，用dp逐行扫描
   dp[i][0]代表第i层从左边下楼的最小cost dp[i][1]第i层从右边下楼的最小cost，dp[i][0] = dp[i - 1][1] + # of rooms in level i - 1 / dp[i - 1][0] + 2 * farest room with light on
   dp[i][1]同理
2. 如果可以上下楼 （按照楼主描述，应该是不考虑上下楼）

天际线

第三轮是一个烙印， 利口原题 skyline

思路：
lc原题

code：

```
Provider: Leetcode discussion
class Solution {
    public List<int[]> getSkyline(int[][] buildings) {
        List<int[]> sides = new ArrayList<>();
        for(int[] b: buildings) {
            // open side
            sides.add(new int[]{b[0], b[2]});
            // close side
            sides.add(new int[]{b[1], -b[2]});
        }

        Collections.sort(sides, (a, b) -> {
            if(a[0] != b[0]) return a[0] - b[0];
            else return b[1] - a[1];
        });
        List<int[]> res = new ArrayList<>();
        // sweeping line, the height of points on the line
```

```
        Queue<Integer> line = new
PriorityQueue<>(Collections.reverseOrder());
        line.offer(0);
        int prev = 0;
        for(int[] side: sides) {
            prev = line.peek();
            if(side[1] > 0) line.offer(side[1]);
            else line.remove(-side[1]);
            if(prev != line.peek()) {
                res.add(new int[]{side[0], line.peek()});
            }
        }
        return res;
    }
}
```

Random Number Generator

第四轮是一个黑小哥，一个设计题，设计如何design一个可以返回随机数的函数， 出了bug怎么办之类的

计算矩阵的dot product

(Provider: Stella Qiu)
电面：给定一个n*n矩阵和整数r, c，计算矩阵第r行和第c列的dot product.

solution: naive for loop，时间复杂度O(n)，空间复杂度O(1)

follow up: 如果矩阵是sparse matrix。设计数据结构来表示这个矩阵，并计算第r行和第c列的dot product.

solution: 矩阵可以用一个rowMap(HashMap<Integer, HashMap<Integer, Integer>>)和一个colMap(HashMap<Integer, HashMap<Integer, Integer>>)表示。从rowMap中找出key为r的HashMap<Integer, Integer>，这个map中的key表示column index，value表示矩阵中element的值。同理，从colMap中找出key为c的HashMap，key表示row index，value是element value。再找出两个map中相同的key，将它们的value相乘并求和。时间复杂度O(n)，空间复杂度O(n)。

矩阵中找最长线

电面：和LC 562 Longest Line of Consecutive One in Matrix 类似

思路： 二维dp
lc solution：用三维数组压缩代码

```
    public int longestLine(int[][] M) {
        int rows = M.length;
        if (rows == 0) return 0;
        int cols = M[0].length;
        int res = 0;
        // 0 horizantal, 1 vertical, 2 diagonal, 3 anti-diagonal
        int[][][] dp = new int[rows][cols][4];
        for (int i = 0 ; i < rows ; i++) {
            for (int j = 0 ; j < cols ; j++) {
                if (M[i][j] == 1) {
                    dp[i][j][0] = j > 0 ? dp[i][j-1][0] + 1 : 1;
                    dp[i][j][1] = i > 0 ? dp[i-1][j][1] + 1 : 1;
                    dp[i][j][2] = (i > 0 && j > 0) ? dp[i-1][j-1][2] +
1 : 1;
                    dp[i][j][3] = (i > 0 && j < cols - 1) ?
dp[i-1][j+1][3] + 1 : 1;
                    res = Math.max(dp[i][j][0], res);
                    res = Math.max(dp[i][j][1], res);
                    res = Math.max(dp[i][j][2], res);
                    res = Math.max(dp[i][j][3], res);
                }
            }
        }
        return res;
    }
```

LC 399 Evaluate Division (汇率题， 频率10+)

电面：给一堆钱和他们直接转换的汇率

```
A B 3   A = 3 * B
B C 7   B = C * 7
C D 1   C = D * 1
D C 1   D = C * 1
```

实现 double convert(String currency1, String currency2)这个函数， 输入两个货币，返回他们之间的汇率。
确认眼神后确定输入都是valid， 汇率都是正数，有小数
需要自己建立图，需要考虑自己换自己的情况

思路：

1. 建立图，首选dfs方法
2. 如果有多次查询，让优化查询时间可以用map cache或者用union find做

Jan 9

Valid word abbreviation(lc408) + unique word abbreviation(lc288) + word abbreviation(lc527)

一开始是就是easy那个，找abbreviation， 接着是给你一个sting[] 和一个string abbr判断是否是unique abbreviation. 最后是给你一个string[] 和一个String abbr, 找出这个string[]里abbreviation是abbr的所有string

思路：lc原题

Jan 8

拿棋子 （频率 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

思路：LC 947
follow up ：返回拿的路径，其实就是让用dfs不要用union find。每次移除棋子的时候加入路径就好
Xiaoyuan Wang: 每个连通图可以做成一个最小生成树, 每次拿叶子节点(入度,出度都为1)

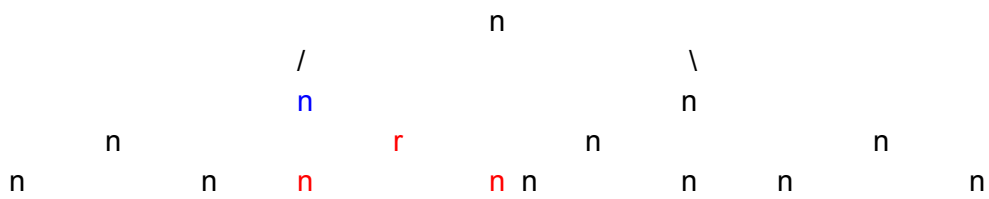
人车匹配 （见首页，频率 20+）

1. 人车匹配，说了bfs和pq，写的pq。follow up有tie怎么办，小哥说第一次出这个题让我自己想怎么处理，就随便说了

红蓝小人占领二叉树 （频率 5）

两个人红蓝，在二叉树上，每个人可以从第一个选的点开始同时往相邻的点扩展占领点，已知red选了一个点，（规则大概是两点之间的可以共同占领，但红的children只能红的占）问蓝第一个点选哪里最后能占领的最多。输入root和红的点，输出蓝色选的node

请问这题可以有哪位好心的童鞋再解释一下题目的意思吗？“规则大概是两点之间的可以共同占领，但红的children只能红的占” 这里看不懂😔
例子：(N： 空node， r: 红色点，b蓝色点)



如果蓝色小人在红色小人上方作为起点，标红点node现在对于蓝色小人来说永远不可占领，其他节点两人可以轮流扩展占领
此处b选在蓝色节点处为最佳策略，蓝色小人堵住了红色小人向上扩展的机会

[相关帖子]

思路：

这个题关键在于明白规则：

即除了第一个点，其余的点放的时候都要连接到相同颜色的点， 所以红色选定后，就把树分为了三部分，如果蓝色选的是最大的一部分并且紧挨着红色第一个点，就有可能赢。 因为最大的一部分被蓝色堵住后， 红色都到不了了

follow up是怎么更efficient

选第一个点的的时候，要选一个三部分尽量均匀的，即 任何两部分都大于第三部分的，即红色先选并选第一个点的规则， 这个是follow up

参考代码

Provider: null

```
TreeNode {
	int key;
	TreeNode left;
	TreeNode right;
	public TreeNode(int key) {this.key = key;}
}
private TreeNode redParent = null;
private TreeNode red = null;
public TreeNode findNode(TreeNode root, TreeNode red) {
	// sanity check
	this.red = red;
	int redL = countNode(red.left);
	int redR = countNode(red.right);
	int redUp = countNode(root);
	int max = Math.max(redUp, Math.max(redL, redR));
	if(max == redL) return red.left;
	else if(max == redR) return red.right;
	else return redParent;
}
private int countNode(TreeNode root) {
	if(root == null) return 0;
	if(root.left == red || root.right == red) redParent = root;
	if(root == red) return 0;
	return countNode(root.left) + 1 + countNode(root.right);
}
```

买卖股票1和4

2. 蠢口姚八巴（lc188） 先写1个transaction 然后实现k transaction 最大profit

思路：

如果k小于天数的一半，状态机

状态1: 第一次买入股票后的最大profit
状态2: 第一次卖了股票后的最大profit
状态3: 第二次买入股票后的最大profit
状态4: 第二次卖了股票后的最大profit

……一共2*k个状态
在新的一天，对于每个状态，我们可以选择停留在昨天的状态，或者通过买卖股票转移，如果状态转移后的profit比停留小，我们选择停留在昨天的状态

如果k大于天数的一半，贪心做法

Jan 7

Log Class，猜数字，矩阵最长连续路径

刚刚面完的狗家onsite，来地里汇报一下面经。题都不难。
第一轮国人大哥。给Log class, 然后给一个list里存的都是log，一个max number, 叫你找出最佳的切割点满足切割出来的log总数小于等于max(思路binary search 参考lc410)

第二轮是刷题网站伞起舞，面试官是很nice的硬老哥，一步步给提示解决了。
午饭蛮好吃的。。。。人很多就是了

第三轮是非常激情的白人大哥，聊了五分钟简历。题是输入matrix，找到最长的连续的递增路径

第四轮是我觉得最不爽的，面试官全程基本没有提示，根本不互动，低头狂敲键盘。也是一个log class。然后写一个函数，每次调用该函数的时候判断，如果在10s内的某个log数超过5，raiseAlert。raise后在某个10s内该log数小于等于3后，clearAlert。这轮我真的已经累了累了，然后写的也很乱。代码最后面试官最后说他觉得work，但我觉得其实并不一定，哈哈哈哈哈哈哈。

原帖

多支树剪支

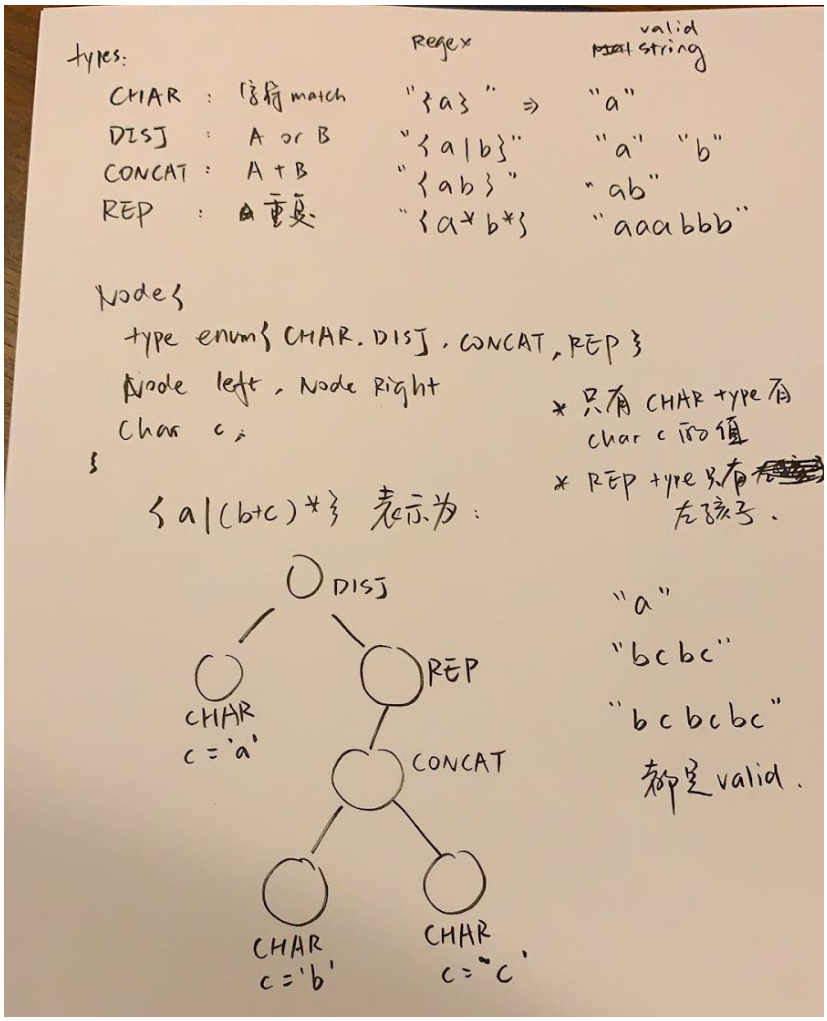
第一题 一个多支树，有的node有颜色 剪掉所有没有蓝色subnode的branch

写一个recursive的preorder就行
后面非常间接的想问iterative怎么写 我反应慢 半天才知道他是想问这个 然后提了一下stack的写法 就没时间了

思路

wild card match变种b

第二个人 类似leetcode wild card match
给个regex pattern，树的形式，给个string 问有没有march
regex pattern用tree表示
分情况讨论 recursive写



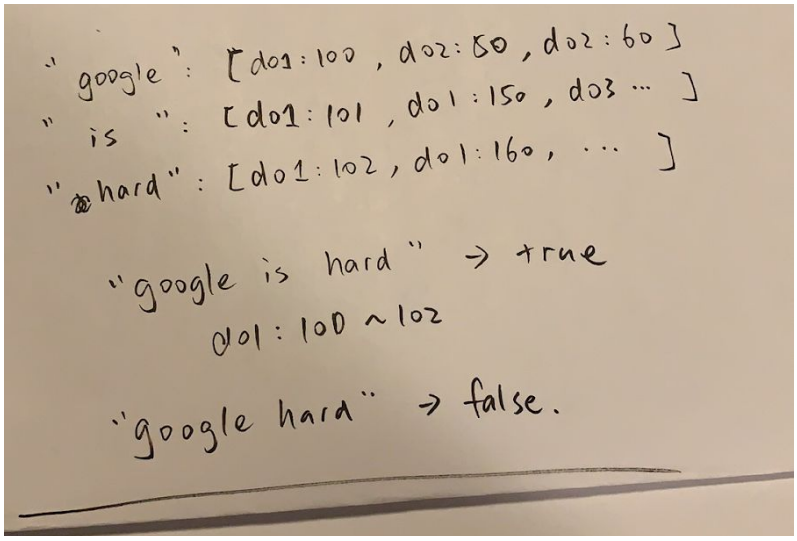
函数签名
Boolean isMatch(node regex, string s)

思路
参考代码

```
if (node.type == CHAR)
    return (s.length==0 && s.charAt(0)==node.c);
if (node.type == DISJ
    return ( match(node.left) || match(node.right));
if (CONCAT)
    for (i=0 ... length)
        if (match(node.left, substring(0,i))
        && (match(node.right, substr(i))
    {return true;}
    }
if (REP)
    for (i=0 ... length)
        if (match(node.left, substring(0,i))
            if(match(node, substring(i)):
                return true;
            }
        }
    }
return false;
```

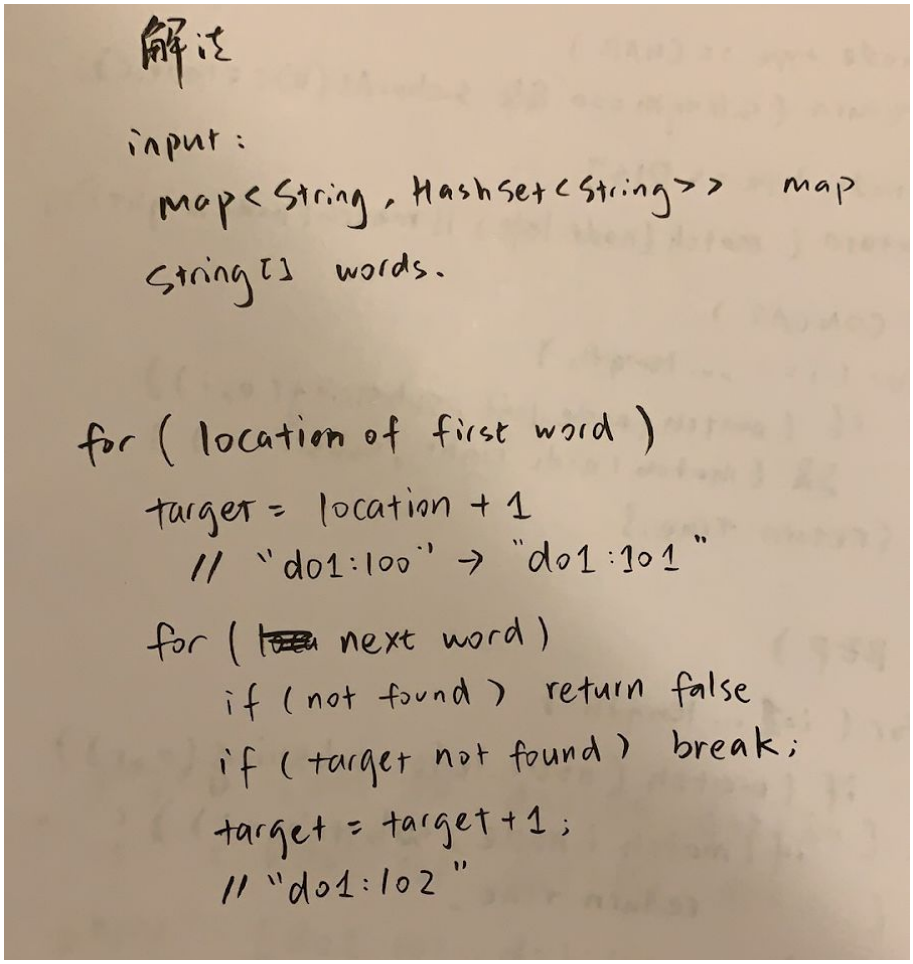
匹配search index和input phrase

第三个人 给你google的search index， 再给你一个input phrase，问有没有match input自己定义 但是存的是每个词在每个document里面出现的位置。



这题不难 不用准备哈哈 o of nk的题 k是phrase长度 因为你可以define input format
n是每个单词的index size

参考解法



构造包含所有vocab都superstring

第四个 在一个语言里面 用一个string 包含所有这个语言里的vocab
他先说了这个问题 但是没有要求用真的vocab， 而是四个digit的code等于一个string 包括所有possible 4 digit code
第四个题没要求最小 我们讨论出一个optimazation就直接做了 没要求best case
也许可以follow up 但是也许我做的慢 就没机会 哈哈
这个题 如果你们去sunnyvale还是有可能碰到的 这个哥说他问了200+人这个问题
类似 lc cracking the safe吗？（感觉像）

Cpp参考代码（Provider: Dingli Zeng）
如果词汇量未知，单词长度未知，可以考虑用trie来辅助搜索，用前一个单词在trie里寻找prefix相同的单词，找到后，从trie里移除。这个方案不一定是最优解，但肯定可以得到superstring。

下面是我的答案：

```
string getSuperString(string keys[], int n) {
    unordered_set<string> unprocessed; /* hashset for unprocessed
words */
    int maxlen = 0; /* max length of any word */
    struct TrieNode* root = getNode(); /* trie */
    for (int i=0; i<n; i++) {
        insert(root, keys[i]);
        unprocessed.insert(keys[i]);
        if(maxlen < keys[i].length()) maxlen = keys[i].length();
    }

    string res;
    while (unprocessed.size() > 0) {
        int prevLength = maxlen - 1;
        if(prevLength > res.length()) prevLength = res.length();
        while(prevLength > 0) {
            string prev = res.substr(res.length() - prevLength);
            if(startWith(root, prev)) { /* modify search method as
startWith if the node is the end of word or has children */
                string word = getWord(root, prev); /* modify search
method by adding traversal to locate the first word having the prefix
*/
                res += word.substr(prevLength);
                remove(root, word);
                unprocessed.erase(word);
                break;
            }
            prevLength--; /* reduce the size of prefix if not found a
match */
        }
        if(prevLength == 0) { /* no prefix match, just get the first
one from the hashset */
            string word = *unprocessed.begin();
            res += word;
            remove(root, word);
            unprocessed.erase(word);
        }
    }
    return res;
}
```

Jan 22

矩阵流水的问题，水会从一个点流向周围八个点钟=中最小的那个，一直流到local minimum为止，给一个矩阵，然后把每个matrix cell更新为每个点最后水流到的低点的坐标

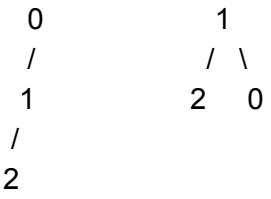
Jan 5

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。
原帖

构造一棵size为n的随机树

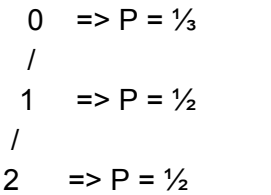
第三轮，三哥，题目就一句话，让我先随机构造一颗size为N的树。我开始没听到“random”一词，就以为构成一颗树就行。秉着论坛经验贴说的，要提前考虑各种条件。我就开始问他，有木有其他限制，比如形状有木有要求，普通的树还是二叉树，N<=0怎么办等等。。。他就说没有，只要size是N就行了
原帖



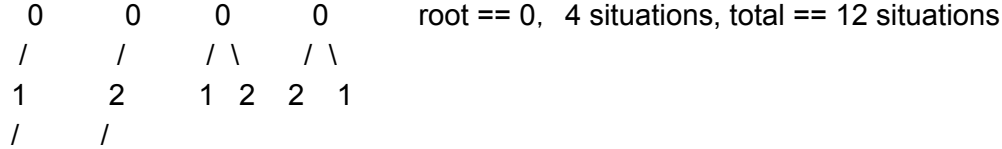
如果要求上图是不同的两棵树（虽然它们选择的边都相同，但是根不同），可以用以下思路

1. pool初始化为[1,2,3, … n]
2. 从pool中随机选择一个vertex V准备加入tree
3. 从tree中随机选择一个vertex U作为V的父亲节点
4. 将v从pool中删除，加入u的孩子中
5. 重复从2开始，直到pool为空

构造



概率：P = 1/12



2 1

参考代码：
Provider: null

```
private class TreeNode{
	int val;
	List<TreeNode> children;
	public TreeNode(int x) {
		val = x;
		children = new ArrayList<>();
	}
}

public TreeNode getTree(int N) {
	if(N == 0) return null;
	List<TreeNode> pool = new ArrayList<>();
	for(int i = 0 ; i < N; i++) {
		pool.add(new TreeNode(i));
	}
	List<TreeNode> tree = new ArrayList<>();
	Random rand = new Random();
	while (pool.size() > 0) {
		int idx = rand.nextInt(pool.size());
		TreeNode curr = pool.get(idx);
		if(tree.size() == 0) {
			tree.add(curr);
			pool.remove(idx);
		} else {
			int parent = rand.nextInt(tree.size());
			tree.get(parent).children.add(curr);
			tree.add(curr);
			pool.remove(idx);
		}
	}
	return tree.get(0);
}
```

Time complexity: o(n^2)

如果题目要求是构造一棵无根的label tree

```
 0          1
 /         / \
1         2   0
/
2
```

一张n个顶点的联通图中构造一棵随机的生成树，上图两树被看作一棵树

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

(cayley's formula中用到的树的表示方法)
Time complexity: O(n ^ 2)
参考代码
Provider: null

```
public Map<Integer, List<Integer>> getRandomTree(int n) {
	if(n <= 2) {
		Map<Integer, List<Integer>> tree = new HashMap<>();
		for(int i = 0 ; i < n ; i++) {
			tree.put(i, new ArrayList<>());
		}
		for(int i = 0 ; i < n ; i++) {
			for(int j = 0 ; j < n ; j++) {
				if(i != j) {
					tree.get(i).add(j);
					tree.get(j).add(i);
				}
			}
		}
		return tree;
	}
	int[] prufer = new int[n - 2];
	Random rand = new Random();
	for(int i = 0 ; i < prufer.length ; i++) {
		prufer[i] = rand.nextInt(n);
	}
	return getTree(prufer);
}

// 根据prufer sequence得到label tree
public Map<Integer, List<Integer>> getTree(int[] prufer) {
	int n = prufer.length + 2;
	Map<Integer, List<Integer>> tree = new HashMap<>();
	for(int i = 0 ; i < n ; i++) {
		tree.put(i, new ArrayList<>());
	}
	int[] vertices = new int[n];
	for(int x : prufer) {
		vertices[x]++;
	}
	for(int i = 0 ; i < prufer.length ; i++) {
		for(int j = 0 ; j < vertices.length ; j++) {
			if(vertices[j] == 0) {
				// connect j <--> prufer[i]
				tree.get(j).add(prufer[i]);
				tree.get(prufer[i]).add(j);
```

```
                    vertices[j] = -1; // delete from vertex set
                    vertices[prufer[i]]--; //delete from prufer set
                    break;
                }
            }
        }
        // connect the last two vertex with vertices[i] == 0
        Integer first = null;
        for(int i = 0 ; i < vertices.length ; i++) {
            if(vertices[i] == 0) {
                if(first == null) first = i;
                else {
                    tree.get(first).add(i);
                    tree.get(i).add(first);
                    break;
                }
            }
        }
        return tree;
}
```

N-ary Tree的最长路径(LC559)

第四轮：国人小哥，n-ary tree里面的最长路径。

数组偶数位大和奇数位小

[原帖]
一个已经sort 过的数组。[1,2,3,4,5,6,7] for example.
把它变成一个，偶数位的数字要大于所有这个位置之后的数字。 奇数位的数字要小于所有这个位置之后的所有数字。
上面那个例子变形之后应该是[7, 1, 6, 2, 5, 3, 4]

follow up如果数组没有sort 过呢？

思路：
如果不需要in-place，用一个额外的buffer数组双指针merge

如果需要in-place

1. [1, 2, 3, 4, 5 , 6, 7]
2. 将原数组分成两半，先reverse后半部分, [1, 2, 3, 7, 6, 5, 4]
3. 然后用shuffle string的方法将数组shuffle

参考代码：

Provider: null

```
    public void reform(int[] source) {
           int mid = source.length / 2;
           reverse(source, mid, source.length - 1);
           if(source.length % 2 == 0) shuffle(source, 0, source.length
     - 1);
           else shuffle(source, 0, source.length - 2);
    }
    private void shuffle(int[] nums, int left, int right) {
           if(left >= right) return;
           if(left + 1 == right) {
                swap(nums, left, right);
                return;
           }
           int len = right - left + 1;
           int mid = left + len / 2;
           int lMid = left + len / 4;
           int rMid = mid + len / 4;
           reverse(nums, lMid, mid - 1);
           reverse(nums, mid, rMid - 1);
           reverse(nums, lMid, rMid - 1);
           shuffle(nums, left, left + (len / 4) * 2 - 1);
           shuffle(nums, left + (len / 4) * 2, right);
    }
    private void reverse(int[] nums, int left, int right) {
           int l = left, r = right;
           while(l < r) swap(nums, l++, r--);
    }
    private void swap(int[] nums, int i, int j) {
           int tmp = nums[i];
           nums[i] = nums[j];
           nums[j] = tmp;
    }
```

看题目似乎是leetcode Wiggle sort I & II

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
https://www.geeksforgeeks.org/recursively-remove-adjacent-duplicates-given-string/

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

房子到buildings到最短路径

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 18

三个数字能否组成合法日期

1. 给三个integers, 判断这三个integers组成的（a,b,c）是否构成一个valid date.
yyyy-MM-dd也算valid， MM-dd-yyyy也算valid， 要考虑所以可能的合法的date格式。
补充：
给三个integer，然后可以让这三个integer分别作为year, month 和day，然后看他们能不能组成一个有效的日期
注意corner case
[原帖]

思路：

1. 考虑leap year和non-leap year
2. 考虑负数和0
3. 月份范围1-12
4. day范围1-31
5. year范围正整数

参考代码：
Provider: null

```
public boolean canFormDate(int[] nums) {
```

```
      return firstFrom(nums, 0);
}
int[] days = new int[]{31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31};
// yyyy-mm-dd
private boolean firstFrom(int[] nums, int curr) {
      if(curr >= nums.length) return true;
      for(int i = curr ; i < nums.length ; i++) {
            if(curr == 0) {
                  if(nums[i] > 0) {
                        swap(nums, curr, i);
                        if(firstForm(nums, curr + 1)) return true;
                        swap(nums, curr, i);
                  }
            } else if(curr == 1) {
                  if(nums[i] >= 1 && nums[i] <= 12) {
                        swap(nums, curr, i);
                        if(firstFrom(nums, curr + 1)) return true;
                        swap(nums, curr, i);
                  }
            } else {
                  int year = nums[0];
                  int month = nums[1];
                  if(month == 2) {
                        // leap year
                        if((year % 4 == 0 && year % 100 != 0) || year %
                  400 == 0) {
                              if(nums[i] >= 1 && nums[i] <= 29) return
                  true;
                        } else {
                              if(nums[i] >= 1 && nums[i] <= 28) return
                        false;
                        }
                  } else {
                        if(nums[i] >= 1 && nums[i] <= days[month-1]) {
                              return true;
                        }
                  }
            }
      }
      return false;
}
private void swap(int[] nums, int i , int j) {
      int tmp = nums[i];
      nums[i] = nums[j];
```

```
        nums[j] = tmp;
}
```

Follow up: 判断上一题的date是否是ambiguous

比如（2018,5,6）可以是2018年5月6号，也可以是2018年6月5号，这就算ambiguous，产生了歧义。

思路
看一个数字是否能满足year，month，day中的多个限制

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
Provider: LC discussion

```
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（**LC940** Distinct Subsequences II 变种)

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
Provider: null

```
	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
Provider: GeeksforGeeks

```
	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。
Code (不带括号）
Provider: (Fill your name)

```
/**
 * cntx记录x前面的系数
 * sum记录和
 */
double calcX(string equation) {
    int flag = 1, sign = 1;
    int cntx = 0, sum = 0, tmp = 0;

    for (int i = 0; i < equation.size(); i++) {
        cout << equation[i] << endl;
        if (equation[i] == '=') {
            sum += tmp * flag * sign;
            tmp = 0;
            flag = -1;
            sign = 1;
            continue;
        }
```

```
        if (isdigit(equation[i])) {
            tmp = tmp * 10 + equation[i] - '0';

        } else if (equation[i] == '+' || equation[i] == '-') {
            tmp = tmp * sign * flag;
            sum += tmp;
            tmp = 0;
            sign = equation[i] == '+' ? 1 : -1;
        } else if (equation[i] == 'x') {
            if (tmp == 0) cntx = sign == 1 ? cntx + 1 : cntx - 1;
            else {
                tmp = tmp * sign * flag;
                cntx += tmp;
                tmp = 0;
            }
        }
    }
    sum += tmp * sign * flag;

    return sum * (-1.0) / cntx;
}
```

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)

Provider: null

```
    // 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～～
Code:
Provider: null

```
/*
      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：tmp无法重复利用
Provider: null

```
	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];    // 找环
		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;
	}
```

Nov 7

还原黑白图片

帖子
给一个只有黑白pixel的图片，调用给定的两个method，把图片重新画出来。两个method分别是：int readSource(int x1, int y1, int x2, int y2)和void writeTarget(int x1, int y1, int x2, int y2);
补充

题目意思大概是要利用第一个method来判断坐标区域内是否为全黑全白或者都有，三种情况会分别返回1,-1,0。第二个method是在坐标区域内画图，但是画出来的图只有黑色。

·LC类似题目：427 construct quad tree

思路：

1. dfs解法，先检查当前square是否全黑全白，如果全黑，涂黑当前区域，返回，如果全白，返回，如果有黑有白，按照construct quad tree的的思路分成4块递归

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
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
这里的思路我个人认为是很容易举出反例的，比如有特别长跨度的interval存在的情况下，举例，k = 2, [1, 5, 10], [2, 11, 16], [3, 5, 20] 这种情况下每个人都有可能是第二个访问的，而不是只有2和3
valid(intv[x]) -> # of (intv[i].end < intv[x].start) < K && #(intv[i].start < intv[x].end) >= K

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

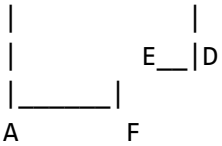
给一个场景，比如Gmail的每个用户（如果没有自己设置头像）就会自动分配一个头像，有着名字首字母和一个特定的背景颜色。问题是给定一个List<Color> colors，给一个List<String> names，给每个名字分配一个颜色。分配规则是：相同的人必须有相同的颜色；相邻的名字要有不同的颜色。

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：

```
B_________C
```



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(高频2次)
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
思路2: BFS的时候，用1来标记是否已经走过。现在拆掉了墙，可以用新的状态，如-1来标记拆了墙走过。其实本质还是记录不同的状态。

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(确定最短的两边然后binary search找最长边？)

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
Follow up: 无限棋盘 没有block 用A star heuristic.

2. 使用 dfs + memo 的概率题
给一个 input num，随机抽1~10的数字加到 num 去得到新的 sum，如果 sum < 20 继续抽数字加和，20 <= sum <= 25 算 win，25 < sum 算 lose，求 lose probability
dfs permutation 的思路加上 memo，每一层尝试用 num += 1~10，然后 recursion。应该需要算 probability 的平均值，所以 dfs 需要返回 avg prob 和一个 count 用来重新计算 avg prob

Provider: Lei Li
写了一个python版本，如果有错误请指出。

```
def win_rate(num):

    rates = {}

    def dfs(num):
        if num > 25: return 1
        if 20 <= num <= 25: return 0
        if num in rates:
            return rates[num]
        return sum(dfs(num + i) for i in range(1, 11)) / 10.0

    return dfs(num)
```