标注说明: 蓝色为高频，黄色为不确定
纯儿注：无责任整理，请不要转载.....很多都是复制的 slack 讨论的内容或者帖子上别人的代码

1. (必看论文)Google: Google File System, MapReduce, BigTable(新版：caffeine pregel dremel)
   Facebook: Cassandra
   Amazon: Dynamo
2. 有套路，如果问最短，最少，BFS
   如果问连通性，静态就是 DFS,BFS，动态就 UF
   如果问依赖性就 topo sort
   DAG 的问题就 dfs+memo
   矩阵和 Array 通常都是 DP
   问数量的通常都是 DP
   问是否可以，也很有可能 DP
   求所有解的，基本 backtracking
   排序总是可以想一想的
   再不行就数据结构头脑风暴
   对了，别忘了还有分治这个好东西
   其实从数据规模或者说要求的复杂度上也能看出解法
   最小堆好好利用，往往可以把问题复杂度从 n^2 降为 n
   如果遇到二维 dp 不会，先考虑一维情况如何解
3. (面经中看到多次)给一个 string 比如 aaabb， 重新排列这个 string 的 character 让相同的 character 不相邻 follow up 相同的 character 相距至少为 k (已解决，用 heap+queue)
   a. 另一种问法：给一个 String[] array, 和任意一个移动的 window size k， 对 array 里的元素位置进行改变，使得 window 里的元素不重复. 要 efficient 的解法。
4. 给一堆 votes(candidate, timestamp)，问当前时刻 T 得票最高的人是谁。Follow up 问得票最高的前 K 个人
   a. 并不确定题目，基本思路是 hash<candidate,count of votes>,Priorityqueue
   b. 先有那个 sorted Array 存排名好了的结果，每次 vote 的时候更新 array。然后用了三个 map：
      map Person to His Votes
      map Votes to Ranking (这里我假设的所有得票相同的人排名一样)
      map Person to Array Index

      比如当前是[a(10), b(9), c(8), d(8), e(8), f(5)]
      然后 e 加了一票
      那么 将 e 和 c 在 array 里面的位置调换
5. 平面上一堆点，找两个点使由这两个点确定的直线平分剩余所有点
   a. 思路: Brute force 法，n^2 求出所有的线，再遍历一遍所有点看是不是整好分割成相等的两部分
   b. Space: O(1), time: O(n^3)

6. 贪食蛇 就是手机里面的那个贪食蛇，不能出手机屏幕，头部不能吃到身体的任何部分，出了尾巴， 因为头部移过去的时候尾巴会移走，所以不会吃到

   输入:

   1. List of moves: [U, U, D, L, D, R….] U: up, D: down, L: left, R : right 移动的方向，有限长度的 list
   2. List of food: [(x1, y1), (x2, y2), (x3, y3)….] 食物的位置，有限长度的 list
   3. int: width 宽度 (屏幕宽度)
   4. int: height 高度 (屏幕高度)

   开始的时候，蛇在(0,0)的位置，长度是一

   蛇每次吃到一个食物，长度就会增加一，然后得分也会加一，求最后的得分 食物是一个一个出现的，第一个食物没被吃到的话，第二个食物是不会出现的。

   a. 贪食蛇那题，我当时的做法是用一个 linkedlist+hashset 来保存蛇的身体，后来跟朋友讨论，还是用 queue+hashset 比较好。每一步，如果出了边界，或者蛇下一步的头碰到了身体里除了尾巴的别的部分，游戏就结束了。
   b. 如果下一步是 valid 的，有两种情况：1)蛇头到了食物的位置，则 linkedlist 和 set 里加上新的蛇投头的位置，蛇尾巴的位置不变。2)蛇没吃到食物，则除了加上新的舌头，尾巴也会往前移一步，即之前的尾巴的位置就不是蛇的身体的一部分了，要从 linkedlist 里面和 set 里面删除掉。
   c. 为什么要用 linkedlist+set 的原因是，检查下一步的蛇头的位置是不是已经身体的一部分的时候，set 可以 O(1)做到，但是 set 不知道尾巴是哪一个；然后要删除尾巴的时候，可以从 linklist 拿到尾巴，从 list 和 set 删掉尾巴

7. 黑白棋。给一个棋盘和一个棋子的坐标，判断这个棋子是不是活的。Leetcode 也有类似的比这个难得题目。DFS/BFS 看能不能走到棋盘的边缘就好。分析复杂度。

   （DietPepsi 的翻译解读）给一个棋形，给一个已经下了的棋子，判断这个棋子是否有可能被翻转，就是终局之前是否永远不可能被翻转

   或者是下一步之后能不能活

8. 给定黑白棋的一个 Scenario，问当前黑方可以在那些位置放棋子.

   沿着上下左右两条对角线分别扫描一下。O(棋盘大小)

9. 给一堆点，问怎么画凸包，说思路就好。

   a. convex hull or convex envelope of a set X of points in the Euclidean plane or Euclidean space is the smallest convex set that contains X. (一圈包裹所有点的皮筋)
   b. 怎么判断两条线有没有交叉点
      i. 判断 orientation

```
int orientation(Point p, Point q, Point r)
{
    // See http://www.geeksforgeeks.org/orientation-3-ordered-points/
    // for details of below formula.
    int val = (q.y - p.y) * (r.x - q.x) -
              (q.x - p.x) * (r.y - q.y);

    if (val == 0) return 0;  // colinear

    return (val > 0)? 1: 2; // clock or counterclock wise
}
```

ii. ***General Case:*** *（非共线有交叉点情况）*

– (p1, q1, p2) and (p1, q1, q2) have different orientations and

– (p2, q2, p1) and (p2, q2, q1) have different orientations.

iii. ***Special Case*** *（共线有交叉点情况）*

– (p1, q1, p2), (p1, q1, q2), (p2, q2, p1), and (p2, q2, q1) are all collinear and

– the x-projections of (p1, q1) and (p2, q2) intersect

– the y-projections of (p1, q1) and (p2, q2) intersect

iv.

```
// Given three colinear points p, q, r, the function checks if
// point q lies on line segment 'pr'
bool onSegment(Point p, Point q, Point r)
{
    if (q.x <= max(p.x, r.x) && q.x >= min(p.x, r.x) &&
        q.y <= max(p.y, r.y) && q.y >= min(p.y, r.y))
       return true;

    return false;
}
```

c. The idea of Jarvis's Algorithm is simple, we start from the leftmost point (or point with minimum x coordinate value) and we keep wrapping points in counterclockwise direction.

1) Initialize p as leftmost point.

2) Do following while we don't come back to the first (or leftmost) point.

…..**a)** The next point q is the point such that the triplet (p, q, r) is counterclockwise for any other point r.

…..**b)** next[p] = q (Store q as next of p in the output convex hull).

…..**c)** p = q (Set p as q for next iteration).

d. **Time Complexity:** For every point on the hull we examine all the other points to determine the next point. Time complexity is ?(m * n) where n is number of input points and m is number of output or hull points (m <= n). In worst case, time complexity is $O(n^2)$. The worst case occurs when all the points are on the hull (m = n)

e. 方法二：

1) Find the bottom-most point by comparing y coordinate of all points. If there are two points with same y value, then the point with smaller x coordinate value is considered. Let the bottom-most point be P0. Put P0 at first position in output hull.

2) Consider the remaining n-1 points and sort them by polor angle in counterclockwise order around points[0]. If polor angle of two points is same, then put the nearest point first.

3 After sorting, check if two or more points have same angle. If two more points have same angle, then remove all same angle points except the point farthest from P0. Let the size of new array be m.

4) If m is less than 3, return (Convex Hull not possible)

5) Create an empty stack 'S' and push points[0], points[1] and points[2] to S.

6) Process remaining m-3 points one by one. Do following for every point 'points[i]'

4.1) Keep removing points from stack while orientation of following 3 points is not counterclockwise (or they don't make a left turn).

a) Point next to top in stack

b) Point at the top of stack

c) points[i]

4.2) Push points[i] to S

5) Print contents of S

f. Time Complexity: O(nlogn)

g. 缺点：不能引申到二维以上

10. (高频)写有 weight 的随机数生成器
    a. 思路：算概率，将概率变成区间的形式，然后随机生成 0-1.0 之前的一个浮点型数，落在哪个区间内就返回哪个区间对应的值
11. (高频)给一个 int array，找任意一个 popular number, popular number 就是出现次数大于等于 array length 的 1/4 的数。其实就是 Leetcode 169, 229 Mojority Element. 第一问 unsorted array 用的 hashmap。 第二问 sorted array 用的 binary search。
    a. （1）看 arr[0]和 arr[n/4]是否相等，相等则返回 arr[0]；
    （2）如果不等, binary search arr[n/4]第一次出现的位置，假设为 j，比较 arr[j] 和 arr[j+n/4],相等则返回 arr[j].
    (3) 如果上一步还好是不返回，接着 binary search arr[n/2]...
    (4) 类似的接着 binary search arr[3n/4]
12. 类似 Lint code data stream median, 写个 API，有两个方法，addURL(String url) 和 getURL()，getURL()返回的是目前为止所有 URL 长度的中位数。lz 使用两个 heap 做的。follow-up: what if we know the range of the input，比如我们知道 URL 的大小不会超过 2k，那有没有别的 implement 的方法。这个没想出来请大家指教
    a. 面试官都说了最长 2k，那就开个 2k 长度的 array 存 BIT，那每 add 一个 URL 就是加那个长度的 URL 的 frequency，取得时候就取 accumulated frequency 等于现在存的 URL 数/2 的就好

If we want to read cumulative frequency for some integer **idx**, we add to **sum tree[idx]**, substract last bit of **idx** from itself (also we can write – remove the last digit; change the last digit to zero) and repeat this while **idx** is greater than zero. We can use next function (written in C++)

```
int read(int idx){
    int sum = 0;
    while (idx > 0){
        sum += tree[idx];
        idx -= (idx & -idx);
    }
    return sum;
}
```

13. 一道常出现的面经题，拿出来讨论下自己设计接口，使得支持两个 funciton：onUpdate(timestamp, price) 和 onCorrect(temistamp, price). 可以理解为有一个时间流，每一个 timestamp 都对应一个股票的时间，每次调用一次 onUpdate 的时候，就对我们设计的那个类更新对应的 timestamp 和 price， onCorrect 就是修改之前的一个 timestamp 的 price。最后，我们的类要能返回 latest price， max price 和 min price。一开始题目描述的太模糊了我都不知道到底要干啥，墨迹半天才知道是想设计一个类，然后中途也写的乱七八糟的，用了两个 Deque 来存储一个递增和一个递减的序列，类似窗口题的方法。当 onCorrect 的时候就去看队列里面有没有对应的 timestamp，有的话移除然后重新入队。感觉这面面的也不是太好。“感觉应该是定义一个 node class {timestamp, price}, 然后放入 maxheap 和 min heap 中吧，但是如果是用 java 的话，必须要用 hash heap 才可以吧，这样的话太麻烦了啊
    a. 思路：treemap 按时间排序，maxheap(或 maxstack)存最大最小值
14. 让我设计一个 rate limiter，就是给定每秒接受 request 的最大次数，当一个新的 request 来的时候怎么判断是否接受。这题看似很简单，但你只要想到用 queue 或者其他数据结构去存储 request，你就输了。
    a. 思路：我的初始思路是存一个 queue，每当获得一个 request 的时候，获取当前 timestamp 并 offer 到 queue 中，当 queue 大于限额时，判断当前 timestamp 和第一个 timestamp 之差，从而决定是否接受（不管是不是接受，都需要将这个 timestamp push 到 queue 当中，这有可能会造成 queue 的满溢，如果 request 到达速度过快的话）为了解决括号内的问题，更好的解题思路如下图

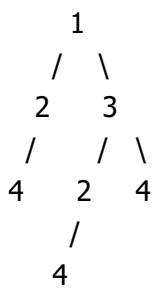
```
rate = 5.0; // unit: messages
per  = 8.0; // unit: seconds
allowance = rate; // unit: messages
last_check = now(); // floating-point, e.g. usec accuracy. Unit: seconds

when (message_received):
  current = now();
  time_passed = current - last_check;
  last_check = current;
  allowance += time_passed * (rate / per);
  if (allowance > rate):
    allowance = rate; // throttle
  if (allowance < 1.0):
    discard_message();
  else:
    forward_message();
    allowance -= 1.0;
```

15. Flow water 问题，没太看懂问题。具体题目如下：

~ ~ ~ ~ ~ ~ ~
~ 1 2 2 3 (5) *
~ 3 2 3 (4) (4) *
~ 2 4 (5) 3 1 *
~ (6) (7) 1 4 5 *
~ (5) 1 1 2 4 *
* * * * * * *

给定一个 grid，判断是否有点技能到达~,也能到达*。就用图的 DFS,BFS 做。

16. 有一堆 query<ID, start_time, end_time>，按照 start_time 排序，输出是<id, start/end, time>按照 time 排序，我最开始说全排，面试官说没有那么多的内存，后来我就改进了一下但还是用了 priorityqueue 实现。然后问我如果是 multithread，每个 thread 处理的 query 是不 overlap 的，问时间复杂度是多少
    a. follow up 多线程答案：O(nlogt), t 是 thread 的个数
    b. 如果有 t 个线程，那每个线程的工作量就是(n/t)，每个线程单独排序耗时应该是(n/t)log(n/t)吧，然后 t-merge 的效率应该是 nlogt，
17. 给一颗二叉树，返回重复的 subtree。比如：



结果应该返回[(2 -> 4), (4)] 两颗树。

也很简单，BFS 遍历，存每个 substree 到 list 里，然后用双重循坏找。其中需要写一个 helper function 判断两棵树是否相同。

follow up：time complexity， 以及如何优化。

得到 subtree list 之后提前处理，把 root value unique 的除掉。

a. 具体实现在 Excecise.java 中（bobzhang2004 的代码）
b. 也可以对每个点存 Map<preorder string+"+"+inorder string, node>然后作比较

18. 设计一个 fraction number 的 class，要求实现 equals 和 String toDecimal()。
toDecimal 是 leetcode 原题，equals 的话注意先得到 gcd，然后除了之后再比较。还有分子分母是否为 0，符号不一样等等都需要考虑。细节挺多的。
最后一个，divide float numbers, 给定一个除数被除数，以及一个精度，要求在不用除法，mod，以及右移位>>的情况下做一个除法器。不难，参考 leetcode divide two integers.
input: float a,float b, float c.
output: result
要求：abs(result - a/b)<= c
corner case 这里就不写了，基本功，要练

19. 给一个二维 boolean array， true 代表 greyed， 要找出所有可能的正方形。比如：
0 1 0
0 0 0
1 0 0
一共有 8 个正方形（边长为 1 的 7 个，为 2 的 1 个，为 3 的 0 个）。注意 matrix 的边长可能不等。用 DP 对 matrix 先预处理，方法有点类似之前地里面经出现的计算 matrix 中 rectangle 面积的题，dp[j]代表从(0, 0) 到 （i, j) 里面所有可用的 grid 的数量。
a. 说法 1：改 maximal square 那道题（leetcode 那道得出动归的结果后, 把所有 dp 相加得到的就是所有的正方形了吧. 比如说 dp[j] = 3, 表示以(i, j)作为右下角的 square 有 3 个, 因为每个正方形只有唯一的右下角, 所以相加得到总 square 数量. ）

20. (俩人玩儿硬币问题) 有一列袋子，袋子上标有内含的硬币价值，两个人轮流取，只能取头和尾的袋子，看谁最后能取到的总价值最大，然后程序是计算作为先手玩家最大能取多少。我以前只写过这题用 backtracking 的办法，给他说了之后他问我 time complexity 是多少，然后问我咋改进，我说用一个 hashtable 存下来所有遇到的情况，每次 backtracking 时去看那个 hashtable 里有没有这个情况，有的话就直接得到剩余最大的价值。他说行，我就开始写了，写完了 backtrack 的部分，我正准备加上 hashtable 时他说可以了，能写到这个也就行了，而且时间也到了。然后拍了照就走了。
思路：用 backtracing 的话其实不难，但是 dp 显然更快

dp[i, j] = max(v[i] *+ min(dp[i + 2, j], dp[i + 1, j - 1]), v[j] + min(dp[i, j - 2], dp[i + 1, j - 1]));*
because your opponent is always trying to minimize your profit.

base case: dp[i, i] = v, dp[i, i + 1] = max(v[i], v[i + 1]);

21. 迷宫该咋走能找到最短路径，我说 BFS
    follow up: 迷宫用小球走，每次确定一个方向之后小球会一直走到碰到障碍物或者边界才会停下，如果停下的地方恰好是出口就算出去了
    解法：新建一个和迷宫同样大小的 matrix，先遍历整个图两遍，把每个空位上下左右的障碍物的位置都存到那个点的 array 里，然后再用 bfs 就可以直接读到从任意一个点出发小球能停到的位置，这样可以降低时间复杂度
    //具体代码实现参考 bobzhang2004 的解法，贴在 Exercise.java 里了
22. RGB 颜色转换比如现有#2f3d13，有 16 进制的 00,33,66,99，cc, ff.要把现有的数字转成最 close to 这几个数字。比如#2f3d13 -> #333300；
    a. 二分法找最近的点……只是 16 进制看着唬人而已
23. (高频)Wall flower 问题(bomb enemy 问题), x 是花，y 是墙，在哪个位置能看到最多的花（只能横竖，不能对角）
    X 0 0 0
    X 0 Y X
    0 X 0 0
    a. 思路：存个 int[][], 每个点从左到右扫一遍，再从右到左扫一遍，从上到下扫一遍，再从下到上扫一遍
    b. 具体实现码在 Exercise.java 里了
24. 给一个 API，O(1)时间计算 slidingwindow avg, global avg, update(insert) value;
    a. 没明白题意……目测是 rangeSum 的变形
25. DP， 有几座城市，每个月在每个城市都有不同的假期，然后每个城市有飞往不同城市的航班，求最大能获取的假期和 Path. dp[i][j] 代表第 i 个月在第 j 个城市所能获得的最大假期。给 N 个 office，以及每个月 office 的假期，还有 office 之间的 adjacent matrix 问该如何变换 office 才能最大化假期；
    a. DP 方程大概是 dp[i][j] = Math.max(dp[i-1][fromCity]+map[i][j], dp[i][j])
26. restruct queue 题目意思大概是这样。一群人在一点商店的门口排好队站着，每个人的身高都不同，到中午了，大家都去吃饭，吃完饭之后回来，要求按照之前的顺序排好队，给的信息是每个人的身高和原来队伍中在其前面有多少个比他高的人。
    a. 可以 bst 吧，bst 按照这个节点的位置排序，先从(身高)高到低 sort，然后根据 f(x)插入，f(x)表示 x 前面有多少个人比他高，同时每个节点维护当前有多少个节点在其左边，这可以在递归时候返回。
    第一高的人是 root。第二高的人，f(x)值决定是在 root 的左边还是右边。第 n 高的人，f(x)最多为 n-1，由于我们已经维护了当前每个节点左边多少个节点，所以就知道插入的位置了。这里 key point 是，比 x 高的人是在他之前插入 tree 的，因为我们是按照从高到底插入 tree。
    bst 最后 in order traverse 就是返回的结果.
    但是因为没有 balance，所以结果也不一定好哪里去
    b. 这个人的 index=在他前面比他高的 + 在他前面比他矮的. 每次用完一个人的时候，把他放到 bit 里. 但是问题是，怎么 search bit. 用 treemap, 比如你用到第 5 个，那么如果是按照顺序排的话，他应该是第 5 个，在他前面的有 4 个位子。treemap 存的是在

这个位置前有多少个还 avaialble，一开始 treemap 里面存了所有 available 的位置，每次算完一个以后就从里面删掉。每次找这个身高的在 treemap 里面的 floor

c. 排队问题的变形题：给一个 array 比如[4,2,1,3,5],根据这个 array 现在我们能有了一个新的 array => 每个数是在原 array 里, 在它左边的所有比它大的 number 的个数,就是[0,1,2,1,0]. 题目是现在给了这个[0,1,2,1,0]要求原 array, 原来 array 的 range 是 1~n

   i. BST 和 SegTree 都是同样的原理，从最后一个数出发，每次在剩下的数里找第 X 大的。这个”找 X 大的”过程如果线性扫一遍就是 O(N)，总体 O(N^2)，如果用 BST 或 SegTree 可以做到 O(lnN)，所以总体 NlgN

27. 第二题就是给你一个 string,你返回 remove 最少数量的 character 之后形成的 palindrome

```
// Everay single character is a palindrom of length 1
L(i, i) = 1 for all indexes i in given sequence

// IF first and last characters are not same
If (X[i] != X[j])  L(i, j) =  max{L(i + 1, j),L(i, j - 1)}

// If there are only 2 characters and both are same
Else if (j == i + 1) L(i, j) = 2

// If there are more than two characters, and first and last
// characters are same
Else L(i, j) =  L(i + 1, j - 1) + 2
```

```
// A utility function to find minimum of two numbers
int min(int a, int b)
{  return a < b ? a : b; }
```

```
// Recursive function to find minimum number of insersions
int findMinInsertions(char str[], int l, int h)
{
    // Base Cases
    if (l > h) return INT_MAX;
    if (l == h) return 0;
    if (l == h - 1) return (str[l] == str[h])? 0 : 1;

    // Check if the first and last characters are same. On the basis of the
    // comparison result, decide which subrpoblem(s) to call
    return (str[l] == str[h])? findMinInsertions(str, l + 1, h - 1):
                               (min(findMinInsertions(str, l, h - 1),
                                    findMinInsertions(str, l + 1, h)) + 1);
}
```

a. 另一个比较 tricky 的做法是将 string 取 reverse，求 lcs，然后 string.length()-lcs(s)等于结果

b. for i = 1 to m (follow the row-major order)

   for j = 1 to n

      if xi = yj

         set c[i, j] = c[i − 1, j − 1] + 1

      else if c[i − 1, j] ≥ c[i, j − 1]

         set c[i, j] = c[i − 1, j]

      else

         set c[i, j] = c[i, j − 1]

28. 给一个双向链表和一个存着部分节点的数组，问这个数组里面的节点能划分成几个 group。
    a. 复杂度：O(n),n 是数组大小，其实意思跟 leetcode 里面 Longest Consecutive Sequence 的意思是一样的，只不过现在不是找前后连续的了，原来找前面是 cur-1，现在是 cur->prev，后面试 cur+1 变成 cur->next
    b. 代码在 Exercise.java 中

29. 题目是给一堆 people，每人有 skills，再给一堆 tasks，每个有需要的 skills，返回 bool 值是否所有任务都可完成。然后一通讨论 corner case。follow up 问 people 之间不能合作怎么做。
    补充：第一问的话没有限制，可以多个人合作完成一个任务，也可以一个人参与完成多个任务
    第二问的话一个任务只能由一个人完成，但一个人可以完成多个任务
    第二问用 bitmap
    class MyBitMap{
    byte[] bytes=new Byte[peopleList.length()/8]
    set(index, bit)
    get (index)
    }
    假设一共有 8 个人 peopleList.length()就==8
    MyBitMap 里面 0...7 每一位对应一个人
    然后建立 HashMap <Skill, MyBitMap> map. map 里面的 Entry 就是 <SkillA, 00001001> <SkillB, 11000001>...存的是每个 skill 对应的有哪些人会。 会 SkillA 的有 0, 3 号 两个人， 会 SkillB 的有 0, 6, 7 号 三个人。
    再然后遍历 task, 比如 TaskA 需要 SkillA 还有 SkillB, 那就取 00001001&11000001 !=0, 说明某个人(0 号) 可以独立完成 taskA

30. 一个 matrix，每个 matrix[i][j]有一个价格，给你一个 budget，问如何求出最大面积的子矩阵，使得子矩阵的价格之和不超过 budget
    a. 我的想法是还是先矩阵加和，dp[i][j]代表从 matrix[0][0]到 matrix[i][j]的 sum，如果 dp[i][j]不超过 budget，则返回 i*j，若超过，就用 leetcode 上 Search a 2D Matrix ii 思想来 search 分界线！！计算分界线上每个点 r,c,返回(i-r)*(j-c) 复杂度是 O((m+n)*m*n)
    b. 也可以 nlogm 或者 mlogn，但是不一定比 m+n 好 只有 m 和 n 比较悬殊的时候，取小者做 log 才有优势
    c.
```
public int findMaxLen(int []arr,int budget, int []y){
    int maxLen=0;
    int sum=0;
    int start=0;
    for(int i=0;i<arr.length;i++){
        sum+=arr[i];
        if(sum<=budget){
            if(i-start+1>maxLen){
                maxLen=i-start+1;
                y[0]=start;
```

```
                y[1]=i;
            }
        }
        else{
            while(start<=i&&sum>budget)
                sum-=arr[start++];
        }
    }
    return maxLen;
}

//int [][]sub=new int[2][2]; top-left, right-bottom corners of the sub-matrix
public int findMaxRectangle(int [][]matrix, int budget, int [][]sub){
    int maxArea=0;
    int m=matrix.length;
    int n=matrix[0].length;
    int []tmp=new int[m];
    int []y=new int[2];
    for(int left=0;left<n;left++){
        Arrays.fill(tmp, 0);
        for(int right=left;right<n;right++){
            for(int i=0;i<m;i++)
                tmp[i]+=matrix[i][right];

            int maxLen = findMaxLen(tmp,budget,y);
            int area= (right-left+1)*maxLen;
            if(area>maxArea){
                maxArea=area;
                sub[0][0]=left;
                sub[0][1]=y[0];
                sub[1][0]=right;
                sub[1][1]=y[1];
            }
        }
    }
    return maxArea;
}
```

31. 平面上有 n 个点，n>=5。每个点有 x 和 y 两个坐标，x、y 均为正整数。n 个点中可以拿掉任意 1 到 3 个点。求包含所有点的矩形的最小面积。（点可以压在线上）。

a. 首先不去掉任何点，找到最小矩形,这个矩形的外边框上最少有两个点,最多可以有 n 个点. 先 naive 就是说我所有可能都试一下,然后第二步肯定也是在新的边框上去掉一个点,第三步也是, 这样的话 worst case 是 n^3
b. 优化:一条边上如果有多于 3 个点这个边就一定会被保留下来, 所以呢，你只考察边上少于等于 3 个点的, 那么就是最多 12^3, 按照 x,y 两个方向找到最大和最小的三个点，总共也是 12 个

32. 可乐机问题：Three coke machines. Each one has two values min and max, which means if you get coke from this machine it will load you a random volume in the range [min,max]. Given a cup size n and minimum soda volume m, show if it's possible to make it from these machines.
    a. Boolean[] dp=new Boolean[n+1], 然后只要 dp[m]和 dp[n]之间任意有一个 true 就是 true

33. 穿墙问题 给一个 m*n 的 board，board 里面存 0，1，和 2，分别代表：
    0 - 可申通无阻
    1 - 有墙阻隔，人只能穿墙才能经过
    2 - 有建筑，人无法经过

    给定一个点的位置和他最多能穿的墙的数量，求到他另一个指定的点的最短路径的长度。如果路径不存在，返回-1
    用一个三为矩阵，board[k][m][n], 其中 k 表示已穿数目，m 和 n 分别是原本的 board 坐标
    //详解写在 Exercise.java 里了
34. 第一题： 问一棵二叉树能不能通过 swap 左右子树变成另一颗树，很简单，直接递归解，然后跑了跑 case,问了复杂度。
    第二题，问一二叉棵树通过 swap 操作能不能变为二分查找树，这个代码比较长，也是递归解。 swap 操作是交换左右子树的，然后子树也可以进行 swap。这跟 leetcode 有道题，verify BST 有点像，可是每步 verify 的时候，你还要考虑 swap 的可能性。

35. Q1: Assume you have a deck of cards. Each card has a number on it with no suit. We define "X of a kind" as X cards with same number on it (X >= 2). Determine if the deck can be fully divided into sets of "X of a kind".
    Example: 3, 5, 3, 5, 3 -> True
    3, 3, 5, 3, 3 -> False
36. Q2: Define "Straight" as 5 cards with consecutive numbers. Determine if the deck can be fully divided into sets of "Straight".
    Example: 1, 2, 3, 4, 4, 5, 5, 6, 7, 8 -> True
37. Define "X-Straight" as X cards with consecutive numbers (X >= 3). Determine if the deck can be fully divided into sets of "X-Straight".
    a. public boolean dfs(TreeMap<Integer,Integer> map,int len,int prev){
    if(len>=3&&map.isEmpty()){
    return true;
    }
    if(len+map.keySet().size()<3){
    return false;

```
            }
            if(len<3&&!map.containsKey(prev+1)){
                    return false;
            }
            if(len>=3&&map.containsKey(prev)){//重新开始一个顺子
                    int count=map.get(prev);
                    count--;
                    if(count==0){
                            map.remove(prev);
                    }else{
                            map.put(prev,count);
                    }
                    if(dfs(map,1,prev)){
                            return true;
                    }
                    map.put(prev,count+1);
            }
            //扩展当前顺子
            if(map.containsKey(prev+1))//类似于上面那一坨懒得写了
            //扩展不了了，同样新建一个顺子
            else if(len>=3){
                    int cur=map.higherKey(prev);
                    int count=map.get(cur);
                    count--;
                    if(count==0){
                            map.remove(cur);
                    }else{
                            map.put(cur,count);
                    }
                    if(dfs(map,1,cur))return true;
                    map.put(cur,count+1);
            }
            return false;
    }
```

38. Partition a set of numbers into two such that di"erence between their sum isminimum, and both sets have equal number of elements.
    For example: [1, 4, 9, 16] is partitioned as [1, 16] and [4, 9] with di": 17 − 13 = 4.
    需要 DFS(NP-hard)question

```
//
// Given an array of 2n integers, their sum is SUM.
// Find n elements whose sum is closest to SUM/2.
//
// A similar problem can be solved the same way:
// Given an array of numbers, sum is SUM. Divide it into 2 subgroups,
// sum of each subgroup is SUM/2.
// After getting the solution 2D array below, find i(s) where isOK[i][V] = 1.
//
```

```
// Solution here is from Beauty Of Programming, 2.18, page 204.
//
// @By: Tom Chen
// @Created on: 3/4/2013.
// @Last modified: 4/9/2013
//
#include <iostream>
#include <vector>
using namespace std;
//
// Code is from "Bian Chen Zhi Mei"
//
class Solution {
public:
    vector<int> divide_array(vector<int> & nums) {
        vector<int> ans;
        int N2 = nums.size(); // 2n.
        if (N2 & 1 == 1) {
            cout << "size of input array should be even" << endl;
            //return ans;
        }
        int N = N2 / 2;

        int sum = get_sum(nums);
        int V = sum/2;

        // isOK[i][v]: whether can find i numbers, whose sum is v.
        int isOK[N + 1][V + 1];
        int val[N+1][V+1];
        for (int i = 0; i <= N; ++ i) {
            for (int j = 0; j <= V; ++ j) {
                isOK[i][j] = 0;
            }
        }
        isOK[0][0] = 1;
        // O(N^2 * sum)
        for (int k = 1; k <= N2; ++ k) {
            for (int i = min(k, N); i > 0; -- i) {
                for (int v = 1; v <= V; ++ v) {
                    if (v >= nums[k - 1] && isOK[i - 1][v - nums[k - 1]]) {
                        isOK[i][v] = 1;
                        val[i][v] = nums[k - 1];
                    }
                }
            }
        }
        // show isOK[][] or val[][] table.
        printf("     ");
        for (int j = 0; j <= V; ++ j) printf("%3d", j);
        cout << endl;

        int x = 0, y = 0;
        for (int i = 0; i <= N; ++ i) {
            cout << "n=" << i << ": ";
            for (int j = 0; j <= V; ++ j) {
                // This shows val[][] table.
                if (isOK[i][j] && i > 0) {
```

```
                    printf("%3d", val[i][j]);
                    x = i, y = j;
                } else {
                    printf("   ");
                }
            }
            cout << endl;
        }

        // get the subarray.
        printf("for last element, i = %d, j = %d\n", x, y);
        while (y > 0) {
            ans.push_back(val[x][y]);
            y -= val[x][y];
            x -= 1;
        }

        return ans;
    }

};
/*
// The solution array isOK[i][v] with val[i][j] filled where isOK[i][v] == 1:
input (sum=44): 1 4 9 18 5 7
       0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
n=0:
n=1:       1        4  5     7     9                          18
n=2:                4  5     7  5  9  7  7  9  5     7        18        18
n=3:                            5     7  7  9  5  7  7  5     7  7
for last element, i = 3, j = 21
*/
//
// This is similar to knapsack, where w[i] = 1, v[i] = nums[i], i = 1,...,n.
// Limit W = sum_i_{v[i]} / 2, maximize sum{v[i]}
// --> However this is still different from knapsack. KS solution does not
work.
// Knapsack: n objects, value: v[i], weight: w[i], restriction: sum{w[i]} <=
W.
//           Ask for: subarray that maximize sum{v[i]}.
// EqualSumSubarray: 2n objects, value: v[i]. Restriction: divide into 2
//           subarrays (equal length or not), sum is half total sum.
//           Ask for: subarray.
//
```

39. 题目是给你一个 board，里面存储 user 的信息，user 有 id 和 socre。
    board 有 adduser(id, score)(返回 add 进去的 user 当前的 rank), findByRank(k) (这个返回 id)。
    Add 如果本身已经有 id 在 board 中，需要对这个 id 的 score 进行 update。
    adduser 操作 和 findByRank 操作都要 lgn
    用 binary search tree 即可，每个 node append 点的个数 就行
40. 关于自建 BST 的知识点：
    a. Search: 如果等于，返回
       val>node.val, 则 node=node.right
       val<node.val, 则 node=node.left

b. Successor：若有右子树，返回右子树的最左节点。
若没有，找到最近的祖先，使得当前节点位于该祖先的左子树，返回该祖先的右子树的最左节点
若没有，则说明该节点没有 successor

c. Deletion: 若该节点没有左右子树，则直接删除
若该节点有 1 个孩子，则将改孩子代替该节点接到该节点的父节点上
若该节点 v 有两个孩子，则找寻右子 u 的最左空节点的父节点 w，s.t. w.left==null. 将 v 和 w 互换位置，删除 v

d. Insertion: 首先先一路 search 下去，找到 leaf u，判断 u.val 和 val 的大小
1) 如果 u.val>val: u.left=new TreeNode(val);
2) 如果 u.val<=val: u.right=new TreeNode(val)
备注：为了实现 deletion 和 successor，最好保证 val 是 unique 的，这样的情况下我们可以给 TreeNode 增加一个字段，写明究竟有多少个相同 val 的 node

41.

```
Input: arrl[] = {1, 2, 9, 5, 5}
       exit[] = {4, 5, 12, 9, 12}
First guest in array arrives at 1 and leaves at 4,
second guest arrives at 2 and leaves at 5, and so on.

Output: 5
There are maximum 3 guests at time 5.
```

a. implement 在 exercise.java 里了

42. 大概就是，一个数组, 1112223334445556677888....当中少 2 个数字，找到就行了。
a. binarySearch(只要重复数字是连续的即可), implement 在 exercise.java 里了

43. AA 制问题：有一题是比如有一群朋友一起出去玩，然后在外面的消费可能有些人先垫付了，比如 ABCDE 五个人出去聚会，总共消费 150，然后 A 垫付了 100，B 垫付了 50，然后大家回来之后要 AA 制结清花费，问使得所有人平摊总消费的最少 transaction 次数，比如 C 回来之后给 A30 块就算一次 transaction。
a. 通过 2sum,3sum…nsum 一路算下来是面试官最推荐的解法(不现实)
b. BFS 更好
c. 最多 n-1 次 transaction

44. “还有一题是给你一些 task，然后这些 task 之间是有规定的前后执行顺序的，比如必须先执行 A 才允许执行 B，然后同时给你 5 个 processor,如何最有效率完成所有任务”
a. 好像是求连通图问题
b. 另一道相似问题的答案：最后姐姐说可以对每组边算个 hashcode,然后像 bucket sort 那样分给一个相应的 machine 处理

45. 二维 board 里面有草莓个数 从左上到右下再回来 求草莓采摘最大个数
a. 四维 dp
b. M[a][b][c][d] represents the first robot is at point(a,b) and the second robot is at point(c,d), the max value they can collect M[a][b][c][d]=Math.max(M[a-1][b][c-1][d], M[a-1][b][c][d-1]，M[a][b-1][c-1][d], M[a][b-1][c][d-1]+判断方程(grid[a][b],grid[c][d])
c. 需要判断一下 ab cd 是否重合， 重合加一个不重合加俩

46. 求 1，2... n 的最小公倍数 ie, n = 5 return 60, n=6 return 60.

a. 每个数求质数，每个质数找到最多因子
b. 该题可以引申到给 n 个任意数求最大公约或最小公倍
c.
```
int i=2;
while (i <= num/2) {
    if (num % i == 0) {
        curMap.get(i)++;
        num = num / i;
        //i = 2;
    } else {
        i++;
    }
}
```
d. nloglogn

47. 给两个大小分别为 m 和 n 的杯子，以及一个目标容量 k，判断是否能够装出目标容量的水，就用 BFS 就可以了（也有数学的解法，但是面试官说不用数学的解法做），followup 是求怎么装（这也就是为什么面试官不要用数学方法做的原因）。之后又讨论了下 DFS 和 BFS 的优劣。
    a. GCD 法：給予二整數 a、b，必存在有整數 x、y 使得 ax + by = gcd(a,b) k 只要是 gcd(m,n)的倍数即可
    b. BFS 法

48. Android 手机手势解锁的所有可能性（至少连接 4 个点，至多全部 9 个），需要考虑三点一线 invalid 的情况，如果考虑成 1-9 九宫格的话就是从 1 连到 3 这样的路径是不允许的
    a. 首先把图构建出来 adj list
    b. 考虑细节部分，比方说 1 不能到 3，不能到 7，不能到 9; 2 不能到 8; etc…
    c. Each pattern must connect at least four dots.
    d. The dots in the pattern must all be distinct.
    e. If the line segment connecting any two consecutive dots in the pattern passes through any other dots, the other dots must have previously been in the pattern.
    f. 389112 distinct patterns
    g. knight move!!!!!!

49. 给一个整数数组（可正可负可重复），返回其中任意两个元素乘积的最大值。
    a. 先 sort
    b. 全是负数，k 是奇数，取绝对值最小的 k 个
    c. 全是负数，k 是偶数，取绝对值最大的 k 个
    d. 全是正数，取最大的 k 个
    e. 有正有负，按绝对值排序，取前 k 个，如果乘积是正，then we are done. 如果乘积是负，找到最后一个正数和最后一个负数，和下一个负数和下一个正数分别调换，看乘积哪个大
    f. 复杂度 nlogn

50. 要求实现一个北美电话号码的分配系统，即 10 位的电话号码的注册、注销和查询。有如下的 API:
```
void registerNumber(String number); // 注册一个电话号码
void unregisterNumber(String number); // 注销一个电话号码
boolean isNumberRegistered(String number); // 查询一个电话号码的注册状态
```

String getAvailableNumber(); // 返回任意一个可用号码"

a. Trie 可以，在每次节点下多加一个字段，用来标记该节点下可用电话数目，用来返回随机 availableNumber

51. coding 一道 类似 24 点。 题目是给一组 integer 问可否用加减乘除使结果等于某个 target。比如｛1，50，3，6，7｝，target＝60 结果是 true 50+6/3+1+7=60
    a. 感觉类似 lc 上那道 282 Expression Add Operators
    b. 允许加括号的话解题思路是：permutation+add parentheses
    c. 不允许加括号的话应该就是 permutation+express add operations
52. 给一个 stream of characters(ascii)和一个 hot word("a-z"),
    求哪一个 character 可以和前面的连续字符组成这个 hot word,
    给出一个 char, 返回 true or false。
    要求 O(1)空间

    follow up
    是如果有多个 hot word, 如果能组成任意一个就返回 true
    不要求 O(1)空间

    Example:
    word = "try"
    stream = "abcokdeftrying....."
    output = "00000000001000....."
    0 代表 false
    1 代表 true

    a. good test case: target=abac, stream =ababacd
    b. 关键问题在于 stream 不能回溯，O(1)空间, 你只要保持最长的后缀，是 word 的前缀就可以, 然后，既然你的 window 一直都是 word 的前缀, 那么你显然不需要真的留一份儿字符串, 你只需要知道你是 word 的前缀的长度就可以 (牺牲复杂度来满足 o(1))

```
stream = ababacd
word = abac
-----
char  s  prefix
a     1  a
b     2  ab
a     3  aba
b     2  ab
a     3  aba
c     4  abac
d     0
```

53. 题目是给一个 string,找出符合条件的 substring pair 个数，条件是两个 substring 只有一个字母不同。LZ 用 dp 做的，A[i] 表示用 index 为 0 到 i 的字符能找到的 pair 个数，A[i] = A[i-1] + 第二个字符串以 i 结尾以 j 开头，第一个字符串以 k 结尾（0 < j < i, i-j <=k < i）可能的匹配个数. 算法时间复杂度是 O(N^4)，复杂度还是很高，哥哥想 follow up 的，结果没有时间了
    a. 列出所有的 substring
    b. sort 各个长度的 substring

c. 暴力解

54. 字符串压缩问题：经典的地里出现过的 String 压缩编码解码类似题, 后悔当时看到没有好好写过一遍.给一个 String 比如"abcdfffffffxyz", 写两个 methods, encode 和 decode. encode 就是比如"fffffff"变成"7xf",decode 就是要变为原字符串.我说"ff"怎么办,他说变成"2xf"你不觉得更长了吗? 我才明白了,应该是 encoded 后的 String 要比原来的短
写完以后他就问我如果原 String 本来就是"5xt"这种结构,我 encode 应该怎么处理? "1x51xx1xq"就好了..
还讨论了好多种情况, 最后一种是"1aaaaa"这种情况怎么变, 我说"1x15xa". 他说这是 6 个字符, 能不能只用 5 个? 实在想不出来,这时候第三个小哥进来了,韩国哥哥就过来告诉我说,其实看做 1a 和 aaaa 两部分 encode 就好....
    a. 特殊处理数字和 x，其他单独的字符就不压缩了
55. 题目：Given an array of Ad (profit, start_time, end_time) and a timespan [0, T], find the max profit of Ads that fit the timespan.
先说了穷举法 O(2^n), 然后说了贪心法（不是最优解），最后用 DP 解决。
    a. dp 做法 $f(i,t) = max\{ (f(i-1,t), f(i-1,start) + profit) \}$
    b.
```
for (int i = 0; i <= time; i++) {
        for (int j = 1; j <= ads.length; j++) {
            profit[i][j] = profit[i][j - 1];
            if (ads[j - 1].endTime <= i) {
                profit[i][j] = Math.max(profit[i][j], profit[ads[j - 1].startTime][j - 1] +
ads[j - 1].profit);
            }
        }
    }
    int max = 0;
    for (int i = time; i >= 0; i--) {
        max = Math.max(max, profit[i][ads.length]);
    }
    return max;
```
56. Design a algorithm to initialize the board of Candy Crush Saga. With M x N board, Q types of candies. (Rules: no 3 for run after initialization, must contain at least one valid move at the beginning)
    a. 第一步：遍历 M×N，每个点随机产生一种 candy，同时检查左边和上面不形成 3 连。
    b. 第二步：重新遍历，每个点跟左边和上边 2 个位置尝试交换，如果能形成 3 连，结束。如
    c. 果到最后不能形成，全部重新生成一遍。
    d. 整体复杂度 O(MN)，除非 Q 很大，重试概率会比较高。或者改善下第二步
    e. 第二步的改善：先随机生成 3 个连在一起的坐标，然后随机从这三个里面选一个移出去。然后完全随机地填剩下的格子，不过填的时候检查上下左右各 2 格 保证没有 3 个同色
    f. Q=3 的时候会产生特殊情况
57. 题大概是这样的，首先，一个二叉树，每条 edge 有 weight（想象成 graph），matching 代表了一组两两没有公共节点的 edges。现在让找一个 matching，满足两个条件，一是 cover 所有节点，二是 edge 之和最小。没做出来，提示下讨论了一种 recursion 的做法

a. root = max(W(左边) + (DP(右子树) + DP(左孩子的左右子树))，W(右边) + (DP(左子树) + DP(右孩子的左右子树))）
b. 参考 house robbery iii，树的自底向上问题

58. NxN 的矩阵，给一个值 M, 求一个新的矩阵，新矩阵的每一个元素是原来矩阵 MxM 中的最小值
    a. 就是 2 维的 sliding window，做两遍就可以了
    b. 先做 Nx(N-M+1)再做到(N-M+1)x(N-M+1)
    c. 就是先做 x 再做 y
    d. 参考题 28
59. 一个 n*m 方格图，有一个位置被下雨了，问最终能从哪些边界处的格子流出水
    a. dfs 吧，从数字大的到小的，如果是边界就加到 return 里面

60. 给一个正整数 n，求能被 n 整除的且只由数字 1 组成的正整数的最小值，如果找不到返回 0，否则返回这个数的长度

61. 在发邮件的时候，比如输入 ben ，下边会提示名字(FirstName, LastName)或者邮件以 ben 开头的人，设计一个类来完成这个提示功能。假设每次我们返回最多 10 个这样的结果。
    a. Follow up I，如果希望返回的结果是 alphabetic 有序的，比如输入 ben 的时候，benaa 在 benbd 前面，怎么设计。
    b. Follow up II，如果我们希望 FN 是 ben 开头的在 LN 是 ben 开头的前边，在 ben Black 在 mike Bend 前面怎么办。
    c. 用 Trie
62. ZigZag iterator Follow up: 如果这些 iterator 都有 hasPrevious(), previous()方法，意思就是后退一步，你的 class 也应该有这两个方法，来后退一步
    a. 每个 Iterator 都有一个 previous()的方法，能把 next()出去的数字弄回去
    b. 我的想法是对于每个 Iterator，用一个 int 保存之前 next 出去了多少个数字，然后找到第一个比当前大的
       比如现在结果返回了 1,4,5,2 那么就变成了
       i1 3　　2 个
       i2 4　　1 个
       i3 6　　1 个
       所以我们应该对 i1 调用 previous，并且把这个 int 减去 1
    c. 用 heap 存 iterator 和数量！返回数量最大的，如果一样大，返回 index 最后的
63. 题目大概是，每次用户会调用一个方法 double next(double v) 然后函数返回的是这个数之前的 windowSize 个数的 average. 比如 windowSize 是 3，call 了 next(10) next(11) next(3) call(1), 第一个返回 10， 第二个返回 10.5， 第三个返回 8， 第四个返回 5
    a. 因为我用了 Deque 来保存之前的数据，我以为他会问精度的问题，我记得面经里有人发过，结果没问。。。所以 Follow up 是 如果不用现成的 Deque 这个 class，你怎么办。 好像用个链表更好写
64. 判断两棵二叉树先序遍历是否一样，讨论一下，最后用 BST iterator
    a. 参考 LC 173

65. 给一个巨大的 data stream，假设都是 integer，内存和硬盘都存不下，问如何在输入完之后，找到特定的 quantile，比如 50%的，那就是求中位数。允许有一定的误差比如 10%。写一个 API，需要哪些变量和方法，这题的关键是如何不均匀滴分 bucket，使得满足误差要求。
    a. randomized online algorithm: http://stackoverflow.com/questions/1248815/percentiles-of-live-data-capture
    b. You can wait until you acquire a certain amount of raw inputs before beginning the rebinning(bucketing)
    c. average over buckets of fixed size k, with logarithmically distributed widths
    d. When a new value comes in:
        i. Determine which bin it should be stored in, based on the bin limits you've chosen.
        ii. If the bin is not full, append the value to the bin list.
        iii. If the bin is full, remove the value at the top of the bin list, and append the new value to the bottom of the bin list. This means old values are thrown away over time.
    e. 上述方法有个明显缺陷，可以用 bucket&counter 的办法来弥补 For each newly captured response time, you simply increment a counter for the bucket it falls into. To estimate the n-th percentile, all that's needed is summing up counters until the sum exceeds n percent of the total.
    f. reservoie sampling
    g. https://research.neustar.biz/tag/probabilistic-sketching/
    h. added a Python snippet: **median stream**

```
median_est = 0
for val in stream:
    if val > median_est:
        median_est += 1
    elif val < median_est:
        median_est -= 1
```

    i. added a Python snippet: **75**

```
quantile_75 = 0
for val in stream:
    r = random()
    if val > quantile_75 and r > 1 - 0.75:
        quantile_75 += 1
    elif val < quantile_75 and r > 0.75:
        quantile_75 -= 1
```

    j. You compose k empty tables and k hash functions. For each incoming element we simply hash it through each function and increment the appropriate element in the corresponding table.
    k. To find out how many times we have historically seen a particular element we simply hash our query and take the MINIMUM value that we find in the tables.
    l. To actually find the quantiles is slightly tricky, but not that hard. You basically have to perform a binary search with the range queries. So to find the first decile value, and supposing you kept around the the number of elements you have seen in the stream, you would binary search through values of x until the return count of the range query is 1/10 of the total count.

Deterministic Algorithms

The first deterministic streaming algorithm for quantiles was proposed by Manku, Rajagopalan and Lindsay [13, 14], building on the prior work by Munro and Paterson's [15]. This algorithm has space complexity $O(\frac{1}{\epsilon}\log^2 \epsilon n)$, meaning that using memory that grows poly-logarithmically in the stream size and inversely with the accuracy parameter ϵ, the quantiles can be estimated with precision ϵn. This result has since been improved by two groups: in [9], Greenwald and Khanna propose a $O(\frac{1}{\epsilon}\log \epsilon n)$ memory scheme, and in [18], Shrivastava, Buragohain, Agrawal and Suri propose a $O(\frac{1}{\epsilon}\log U)$ memory scheme, where U is the size of domain from which the input is drawn.

The Greenwald-Khanna (GK) algorithm is based on the idea that if a sorted subset $\{v_1, v_2, \cdots\}$ of the input stream S (of current size n) can be maintained such that the ranks of v_i and v_{i+1} are within $2\epsilon n$ of each other, then an arbitrary quantile query can be answered with precision ϵn. Their main contribution is to show how to maintain such a subset of values using a data structure of size $O(\frac{1}{\epsilon}\log \epsilon n)$. The Q-Digest scheme of Shrivastava et al. [18] approaches the quantile problem as a histogram problem over a universe of size U; thus $\log U$ is the number of bits needed to represent each element. Q-Digest maintains a set of buckets dynamically, merging those that are light (containing few items of the stream) and splitting those that are heavy, with an aim to keep the relative sizes of all the buckets nearly equal. Specifically, using a $O(\frac{1}{\epsilon}\log U)$ size data structure, Q-Digest ensures that the input stream is divided into $O(1/\epsilon)$ buckets, with each bucket containing $O(\epsilon n)$ items. Thus, the rank of any item can be determined with precision ϵn by locating its bucket.

In theoretical terms, the GK scheme has better performance when the input is drawn from a large universe, but the stream itself has only modest size. The Q-Digest, on the other hand, is superior when the stream size is huge but elements are drawn from a smaller universe. The GK algorithm is very clever, but requires a sophisticated analysis. The Q-Digest is simpler to understand, analyze, and implement, and it lends itself to easy extensions to distributed settings.

66. 文件读写，输入是一个 word list，讨论了半天，最后明白过来就是 string encoding, decoding. 一分钟写完 encoding，正准备写 decoding，面试官说“慢，我们换一个方式"，decoding 改为给一个原来 word list 的 index，要求返回 index 对应的单词。白人大叔很耐心的提示了半天，最后想到在文件的开始写入每个单词结束的位置，但是位置信息的位数不固定，比较难 decoding，最后又提示了半天想出来用二进制表示位置信息这样位数就固定了

67. 车牌问题：给定一个车牌号包括字母和数字，然后给定一个 dictionary，找出至少包含所有车牌里字母的最短单词。讨论了很久算法，一直没提示，后来就说干脆你就写暴力算法吧，用 hashtable 做的，感觉效率很差不过是可以 work 的

68. give a list of intervals, find min number of points which will intersect all intervals
    a. 意思应该是说用几个点组成的集合可以跟所有的 interval 相交不为空
    b. Start end 一起排序 拿到第一个 end 的时候记录 把目前开口的都扔掉。
    c. 扫描所有 interval 的时候，判断如果当前 interval 开始在当前 end 之前，就弃掉，开始在当前 end 之后，就更新当前 end 并且计数++

69. Given lots of intervals [ai, bi], find the interval that intersects the most number of intervals.
    Suppose the intervals are given as `(a1,b1), ..., (an,bn)`. Make a sorted array of length `2n` where ties are broken by
    - if `ai = aj`, then put `ai` first iff `bi < bj`
    - if `bi = bj`, then put `bi` first iff `ai < aj`
    - if `ai = bj`, then put `ai` first

    Mark each point as an `a` or a `b` (maybe keep a binary array of length `2n` to do this). Walk through the array(s) keeping track of how many intervals touch the given point (running total of `a`s minus running total of `b`s). The maximum number encountered happens at the interval with the most overlap.

This is `O(nlogn)` due to the sorting of the intervals.

70. 题是 merge interval 的变形，结合他的工作情景，是有一个 date range 的类，要实现 insert a range, delete a range, sort, check if a date is in some range。写完然后问时间空间复杂度，然后他说他实际工作中是用 Arraylist 存，没想到我用 treeset 存好像也不错~
    a. 用 Arraylist+BinarySearch? 每次插入在对的位置
    b. lc insert interval

71. 给一个二维的棋盘，棋盘上每一格上可能有镜子，方向可能是/或\，从棋盘外射进来一束光线，问射出的方向和射出的格子。Follow up: 问随机生成棋盘的方法

72. 有两个 set 的 points， set A 和 set B，又给了一个 double 类型的参数 delta，求 set A 里面每一个 point 最近的且距离小于 delta 的 set B 的 point，有点绕，就是给一个 point A，就返回一个最近的 point B，并且这个距离要小于 delta。
    lz 开始用的暴力解法，复杂度是 O(n^2)，后来想出来一个先排序再搜索，复杂度降为 O(nlogn)，三姐一直说 interesting，我也不知道是好是坏……
73. 让你自己设计数据结构和算法来解决一个具体问题，题目是如何判断两个人有血缘关系。我用图和 BFS 做的
    a. 所以是有向图，每个点指向 parents，从 2 个点开始往父母方向 bfs，经过的点记录在 2 个 hashset 里，然后每走一步 check 这个点在对方的 set 里有木有出现过这样
    b. 限制是可能有很多代人，然后可以在 node 加额外的信息
74. 给一个 infinite array 只有 0 - 9 设计一个. def getprobability(n):得到某个数出现的概率。

```
for i = 1 to k
    R[i] := S[i]
 // replace elements with gradually decreasing probability
 for i = k+1 to n
   j := random(1, i)   // important: inclusive range
   if j <= k
      R[j] := S[i]
```

75. 4 位密码串起来长度最短，用到 0000-9999 所有数字
    a. 1 位变换 答案详见 Exercise.java
    b. 一万个 node，每个 node 最多十条出边十条入边
76. （比赛夺冠概率问题）就是有一堆 player，每个人 beat 其他人的概率已知。然后已知初始的对阵表，问给定一个 player，问他最后夺冠的概率是多少
    a. 一共有 n 场比赛的话，需要进行 h=logn 轮对阵，用 dp[h+1][n]来存储，dp[k][j]表示选手 j 在第 k 轮是活着的
    b. $dp[k][j] = dp[k-1][j] + \sum_{i=start}^{i=start+len} dp[k-1][i] * matrix[j][i]$ (其中 matrix[j][i]代表选手 j 打赢选手 i 的概率)
        i. 如果当前 j 在左子树(0<(j+1)%(2^k)<=2^(k-1))，那么它下一轮将要选手从 start=(j/(2^k)*(2^k)+2^(k-1))开始，potential 对手一共有 2^(k-1)名
        ii. 如果当前 j 在右子树，那么它下一轮从 start==(j/(2^k)*(2^k))开始，potential 对手一共有 2^(k-1)名
    c. 最后返回 dp[h][target]
    d. 代码写在 Exercise.java 里了

那么这里 Memory : O(h*N) = O(NlogN)
Time Complexity: 在第 k+1 层里，对于每个 i,计算了 2^k 次，所以是 2^k *N 次
那么一共 (2^0 +2^1+...+2^(h-1))*N = (2^h-1) * N = (N-1) * N
所以是 O(N^2) time.

77. 如何用最少的 bit 来记录一副扑克牌的顺序（假设 encoder 和 decoder 都由你来设计）
    a. 因为 encoder 和 decoder 自己定，所以给每种排序赋一个数值就行，如果除去大小王一共有 52 张牌，有 52!种排序，只要 log2(52!)向上取整个 bit 就可以实现。然后问题就是怎么实现，我没被 follow up，因为这已经是我第一轮的第三题了他说知道想法就可以了。这是个 permutation 的问题，比如给每张牌赋个值，那第一个 permutation 就是 1,2,3,..50,51,52，第二个 permutation 就是 1,2,3,..50,52,51，最后一种就是 52,51,50...3,2,1。
    b. 实现在 Exercise.java 里

78. 很大的文本文件，比如 Linux 运行产生的日志文件。要在常数时间找到第 N 个日志，怎么存，怎么找。如果用到了 hash map, 假如文件最小 1b, 最大 40000b ,平均 1000b, 总大小 20T, hash map 要多大？有点蒙逼...讨论了一堆...后来讨论了存 offset 地址，问寻址方式 balabala...具体记不清了反正就是很操作系统的东西
    a. perfect hashing?

Theorem

Set the table size $m = n^2$. Assume H is a universal collection of hash functions from U to $\{0, 1, \ldots, m-1\}$. If we pick a hash function h from H uniformly at random, then

$$\Pr\left[\textit{no collision between keys in } V \textit{ w.r.t. } h\right] > 1/2$$

79. 给 N 个文本文档，找最大相同子句，以单词为单位，连续空格看作一个空格
    a. 题目不明，参考 https://www.careercup.com/question?id=5700764422897664

80. (给点或边组成三角形问题) 给 N 个点，判断能组成几个三角形。在一给定误差范围内，斜率的差小于误差范围看作一条直线不能算三角形。如果把点换成边，判断能组成几个三角形。时间复杂度是多少，怎么优化
    a. 自定义一个无向的图，然后求这个图里有多少个三角形。三角形的定义是，（a,b),(b,c)(c,a)都相连
        i. 引理 1： 以 A 为顶点的三角形个数是 A 的所有 neighbors 集合 N 里面的边的数量。
        Example：
        A 有五个 neighbor，BCDEF
        如果其中有且仅有 BC, EF, DE, CF 四条边
        则图中有 4 个三角形，以 A 为顶点
        ii. 引理 2： A 有五个 neighbors,BCDEF
        F 有五个 neighbors,ABXYZ
        则以 AF 为顶点的三角形个数是 N_A 和 N_F 的交集的大小

81. 有一个 file,里面有很多 comments (// or /*) ，然后还有个 input string, 如果这个 file 的 comment 里 contains input string 的话 return true
82. hamming distance between a and b, a, b < 2^64. 这题很快就做了出来。就是把 a^b>>i &1 64 次。 然后他就说要想办法 speed up，说给我 64G 的 ram。我想了很久最后说可以搞个 2^8 的字典，然后把 a^b 分 8 段比就好。 他就说为什么用 8， 然后就问我 2^8 的字典要用多少空间. 我没记空间大小的那些知识，所以不会做...几经提示后结论是可以用 2^32 的字典要 4G 空间，这样比两次就好。他最后又问说如果你用这个方法，但是 ram 只有 2g，那会发生什么情况。 我就说那会有 error 吧。他就说什么 error。我说不出来。他就说 “you clearly have never used win 95 swap space”. 然后差不多就结束了。我觉的最后这个哥么应该给我差评了
    a. 就是建个字典比如 1010－》2， 1111-》4，1011-》3 这样的， 每个数字对应它的 hammingdistance。 当你要查 1001 0010 这样的数字，你只要把头尾两段分别查一下加起来就好。这样只要 2 个 operation。
83. 计算二叉树两个节点的最短距离。先找 Lowest Common Ancestor，再分别求高度。
84. (雨滴点问题) you have 1 meter walkroad, and randomly generate rain, the rain is 1 cm. simulate how many rain drops to cover all the 1 meter [-0.01~1].
    a. merge interval 问题
    b. 参考 bobzhang2004 的代码，在 Exercise.java 里
85. (Valid Unicode 问题) 这次出了一个 byte encoding 的题目，大概是给你一种编码的格式，然后给你一个 byte 数组，让你判断这个数组合不合法
    这个编码的定义是，有 1-7byte 的 character，每种 character 第一个 byte 有一个固定的前缀，比如 1 byte character 第一个 byte 的前缀是 0，2 byte 是 110, 3 byte 是 1110 以此类推，8byte 是 11111111.
    有一个特殊的前缀是 10，这种 byte 是每个多 byte character 的组成部分。比如一个 2 byte character: 110XXXXX 10XXXXXX。一个 3 byte character: 1110xxxx 10xxxxxx 10xxxxxx。
    如果字符被打段或者超长都要 fail。
86. 给两个长度相等的 string，如果两个 string 对应位置的字母不相同记为一个 distance，如果现在能够交换一次其中一个 string 中任意位置的两个 char，返回能够将 distance 缩到最小的两个 char 的 index（如果有多个最优解只返回一个)
    a. source string: abcde
       target string: ebcda
    b. 我用了一个 hashtable 中套 hashtable 的方式，外层用需要的 char 作为 key，里层 hashtable 作为 value；里层 hashtable 用待换的 char 作为 key，待换的 char 所在的 index 作为 value。O(n)的时间复杂度，O(n)空间复杂度
87. 有 3n 个数围成一个环，取走其中一个的话会顺带去掉这个数相邻的两个数（这两个不计入总和），剩下的继续围成环，问取走 n 个数构成总和的最大值。
    a. 可以直接简化为求 n 个最大的不相连的数
       取任意一个不相邻的子序列，长度是 n，原序列长度是 3n
       必有至少一种可以取到此子序列的方法
       首先贪心将所有 xxOx，或者 xOxx 的 pattern 都去掉一个 xOx
       可以知道取完之后序列仍然合法，即不存在 OO 相邻
       此时由于不存在 xxOx 或者 xOxx，则必有 O 和 O 之间只有一个 x
       这时 O 的数量与 x 的数量相等
       假设我们取走了 k 次 xOx
       那么 x 的数量就是 2k + m = 2n

O 的数量是 k + m = n
推出 k = n
m = 0
Hence, proved

88. (青蛙过马路题) Design an algorithm to play a game of Frogger and then code the solution. The object of the game is to direct a frog to avoid cars while crossing a busy road. You may represent a road lane via an array. Generalize the solution for an N-lane road.
89. 一道设计题，给定一个闹钟的 class，到一定的时间会执行 dosth().

```
class alarm{
    const timer t;
    void dosth();
}
```

让你设计一个 manager 的 class，能够实现 add 一个 alarm，并且当时间到 alarm 设定到的时间的时候 trigger 它. 问怎么设计。请问大家有啥好的想法？

我想到的是

```
class manager{
   list<alarm> curlist; //sorted
   timer global_timer;
  void check_set();
   void addalarm();
}
```

addalarm 保证按 sorted 的添加到 curlist 里面，然后 check_set 可以是个 loop 一直 checktimer，从头开始找所有的到达时间的 alarm trigger 它，但是如果多线程就会遇到需要 lock 的情况

恩 如果是 java 的话可以使用 synchronize 关键字:

```
class AlarmManager extends Thread{
    private Queue<Integer> curList;
    public void addAlarm(int value){
        synchronized(Alarm.class) {
            //insert alarm to queue;
        }
    }
    private void checkAlarm(){
        synchronized(Alarm.class) {
            //check the first element in queue;
        }
    }
    public void run() {
      //checkAlarm on each second
   }
}
```

90. (Consumer&&Producer!!!!!!!)

```
class Producer implements Runnable {
   private final BlockingQueue queue;
   Producer(BlockingQueue q) { queue = q; }
   public void run() {
```

```
    try {
      while (true) { queue.put(produce()); }
    } catch (InterruptedException ex) { ... handle ...}
  }
  Object produce() { ... }
}


class Consumer implements Runnable {
  private final BlockingQueue queue;
  Consumer(BlockingQueue q) { queue = q; }
  public void run() {
    try {
      while (true) { consume(queue.take()); }
    } catch (InterruptedException ex) { ... handle ...}
  }
  void consume(Object x) { ... }
}

class Setup {
  void main() {
    BlockingQueue q = new SomeQueueImplementation();
    Producer p = new Producer(q);
    Consumer c1 = new Consumer(q);
    Consumer c2 = new Consumer(q);
    new Thread(p).start();
    new Thread(c1).start();
    new Thread(c2).start();
  }
}
```

91. (跟上面那个是一道题，如果不让用 BlockingQueue 的话)设计 blocking queue
    //重点！！！看 Exercise.java

92. 给一个词典和一个 target word,问这个 word 是不是 smashable. smashable 的定义是这个词在字典里并且它不停地任意删除一个字母得到的单词也在词典里。用了 DFS 之后提示用 hashtable 优化时间复杂度，然后问如果要提前把这个词典存好之后反复调用怎么办，按照他的提示我想到的方法是按长度 sort 词典里所有单词，按照长度 group 起来，给一个 word 之后就直接找比它长度-1 的那个 group 看有没有哪个 substring 在这个 word 里

93. given millions of electronic books and each book contains hundreds of pages. Given a function OCR to convert the electronic image of book pages to text file. That is to say, scan the books first to get photo copy and then convert the image to text file. So in the process, there is supposed to be 5% error. How do we tell if two books are identical considering the error?

    Error can come from several cases liek: "word random" -> "word ran dom"; "word random" -> "worded random"; "word random" -> "word "

    Solution: It's an open ended question, you can propose different approaches

94. Given a sparse excel file, design a data structure to effectively store all the informations.
    Sparse means most of the excel are empty。
    Write 4 functions: void set(int row, int col, int val), int get(int row, int col), vector<int> getRow(int row), vector<int> getCol(int col)
    getRow and getCol returns the corresponding row or column that is indexed at the input number which neglect empty cells.
    Solution:
    Using this one: unordered_map<int, unordered_map<int,int>> myMap1, myMap2;
    myMap1: (row -> (col -> val)) ; myMap2: (col -> (row -> val))
    Follow up: what if there is a function eraseRow or eraseCol like in excel you can erase a whole row or column.
    Hash table solution becomes unefficient because row index and column index changes and this cause erase time complexity to be O(n)
    Solution : using LinkedList, and I have no time for coding

    有关 linkedlist 的设计：
    LinkedList 的思路是 吧每行每列的元素的 cell 存下来，node 记的是与前面 cell 的差值。这样删除一行就只用改后面一个列的数据。
    比如 原来有 1 6 8 9 14
    本来删除第 6 行的时候，要变成 1 7 8 13 需要修改 6 后面的 8 9 14 的 index.
    但是如果我们存的是前后 index 的差值，我们只需要修改删除行的后面那个 index 。
    比如 一样的例子： 1 5 2 1 5
    我们还是删除一行，只需要改成 1 6 1 5 我们只需要修改一个就好了。思路是这样，没写 code。还要拓展成二维的

95. 白人，设计乌龟，一开始有两个方法，往前走一步，右转 90 度。然后说设计个方法可以执行一串命令：e.g., "FRRF"代表走、转、转、走。然后命令要支持括号和数字：e.g., "F97[RF[F]]"代表走一步，然后执行括号里的命令 9*7 次。
    a. basic calculator 题
96. 第一题是写一个 class simulate 一个病人吃药。一个 bottle 里有 half pills 或者 whole pills, 随机拿一个，如果拿出来的是 half pill 就吃掉，如果是 whole pill 就弄成两个 half pill 吃一个放回去一个。初始给的 pills 不一定全是 whole 的。一开始用了 array list，跟面试官的思路有点不同，交流了一下改用两个 integer 又写了一遍。第二题是假设一开始给的 pills 是 start state，然后给一个 end state. 求一个病人随机吃药，从 start state 吃到 end state 的概率。比如一开始 2 whole, 1 half, 求吃成 1 whole, 1 half 的概率。LZ 一开始说 build 一个 search tree 这样直观点，他表示不需要把 tree 专门建起来，LZ 就直接一边搜一边算没有用 extra memory
    a. 像是简化版的比赛冠军问题
97. 给一个 directed graph 的 start node，这个 graph 里可能有 cycle，如果 remove 一些 edge 可以使这个 graph 不含有 cycle，并且从 start node 依然能访问到所有这个 graph 里的 nodes，这些 edge 就是 back edge。要求打印出所有的 back edge。写完拍了照又 follow up 了一下，问存不存在这种 graph，你可以 remove 不同的 backedges 使得这个 graph valid。LZ 一开始觉得是不存在的，但是想不出来证明的方法。后来画了个图发现是存在的，但 LZ 自己没一下子看出来。。被提醒了下。
    a. 图的几个基本概念
    b. Tree edge, if (u, v) is an edge in the forest (and $v.\pi = u$).

c. Back edge, if v is an ancestor of u in the forest.
d. Forward edge, if v is a descendant of u in the forest.
e. Cross edge, if u and v are unrelated (either they belong to different trees, or belong to the same tree but u is not an ancestor of v and v is not an ancestor of u).

98. 给一些类和 function：
class Point{int x, y;}
class Rect{Point p1, Point p2}
class Shape{
int inside(Rect r){}; //shape 是不规则的图形，给了一个 function 判断这个 shape 和 rect 的重叠关系，
//如果 rect 完全在 shape 里 return 1. 如果 rect 和 shape 有 overlap，return 0， 如果 rect 完全在 shape 外面，return－1
}
draw(Rect r){};
以上是已经有的 function，
给一个长方形的 screen，再给一些 shapes，要求写一个 function 在 screen 上画出这些 shape。
写完后又问了一题，如果有 A 和 B 两个 shape，A 和 B 有 overlap，用给的 inside function 写一个新的 inside function，判断一个 rect 和这个 overlap area 的关系，同样返回－1，0， 1
其实就是 min(A.inside(rect), B.inside(rect));
然后问如果把 min 换成 max 的话这个算的 area 是什么样的。应该是 A 和 B 的 union.
a. Quad Tree: https://www.careercup.com/question?id=6491225129484288
b. https://www.careercup.com/question?id=5175572890124288
c. 屏幕分 4 份儿，判定 Overlap，如果不 overlap，就不继续了. 如果某一份完全在 Shape 里，把它画出来，也不继续了. 如果 overlap 但不完全，继续分 4 份儿

99. 给一个 number, 还有一个数组包含这个 number 所有 prime factors, 问这个 number 的 dividers 的个数
a. 求 n 有多少个 divider, 比如 24 就有 1 2 3 4 6 8 12 24 这些
b. $\prod_{i=0}^{primes.length-1}(primes[i]+1)$

100. music list, if shuffle is possible or not。你有一个 music 的播放列表，里面的歌曲 unique，但是播放列表的长度未知。这个音乐播放器 APP 有两个模式：random 模式和 shuffle 模式。random 模式就是每次随机播放列表里的一首歌；shuffle 模式就是 shuffle 列表里的歌，然后顺序播放，放完以后重新 shuffle，再顺序播放；现在给你一个播放历史记录，要求你写一个函数来判断用户使用的是 random 模式，还是 shuffle 模式。
a. 有关 shuffle 的算法：Fisher–Yates shuffle（Knuth shuffle）
i. -- To shuffle an array a of n elements (indices 0..n-1):
ii. for i from n−1 downto 1 do
iii. j ← random integer such that 0 ⩽ j ⩽ i
iv. exchange a[j] and a[i]

b. MineSweeper：面试官让实现一个初始化函数，给定 grid 长宽，和雷的个数，让随机的把雷放进去

101. 题目是给你一个 positive 的值 K，然后按照 fraction 的值的小到大输出所有 n/d, 其中 1<=d<=k, 0<=n<=d，还有一个要求输出的 fraction 不能有 duplicate，比如 1/2 和 2/4 是 duplicate，这种情况只要输出 1/2。我用了一个 hashtable 来处理 duplicate 的情况, fraction 的实际小数的值对应其输出 string，比如 0.5 对应 1/2，因为输出要按从小到大顺序，所以还用了个 arraylist 存实际小数的值，最后对这个 List 排了下序，然后结合 hashtable 和 list 输出最后结果。然后问时间复杂度，follow up，如果规定只想要输出某个数值区间的 fraction 怎么办，数学的一些东西，算下值范围就好了，还问了这种情况的复杂度。

a. 分数的那个复杂度可以做到 NlogN
例如 k=5，可以构造出 k-1 个 sorted arry list.
1/2<2/3<3/4<4/5
1/3<2/4<3/5
1/4<2/5
1/5
然后直接把这几个 sorted array list combine 就好了
这样不用可以 hashmap 记录 duplicate 和最后再 sort

102. 给一个 N*N grid， 上面有一些格子里面标了 1. 问总共多少方块了有 1. 所以方块包括只含 1 的那个， 还有任何含 1 的那个 所以 2*2, 3*3

a. 我们看每一个点， 数一下以这个点为最右下的方块的个数
然后看每个点， 如果是 1， 以这个点为最右下角的正方形应该有 min(x+1, y+1) 个。 （x， y） 为 index
比如
0 0
0 1
（1，1）那个点 就有 两个方块， 它自己和包括它的那个 2*2。 因为自己肯定是 1，所以每个方块里保证有 1 了，
如果是 0， 那么就要看应该少数几块， 我们需要维护一个 matrix 那个 matrix 里面记录 和这个点最近的 1 的距离。
int withzero = Math.min(radius[i-1][j], Math.min(radius[i-1][j-1], radius[j][1])) + 1;
比如
0 0 0
0 1 0
0 0 0
那么我们发现（2，2）最短距离 为它左上的 0（每次我们读 1， 往 matrix 里面放 0，表示 1 和 1 的距离为 0）
然后我们+1 因为自己是 0
然后 count 就是 min(x+1, y+1) － withzero

103. 给你一堆灯泡。可以 flip 一个范围开着变关， 关变开， 然后问这么干了 k 次以后， 随便问你一个灯泡是开着开始关着，怎么做。我还是很弱地没写完。 但是有很多 idea。主要想就是 interval， merge interval 之类

a. 假设询问有 q 个，灯泡有 n 个
对于一次 flip(l,r)，在 l 和 r+1 两点取反，即 flag[l] = flag[l] xor 1, flag[r + 1] = flag[r + 1] xor 1，灯泡 i 最后的状态就是把 j<=i 的 flag[j]xor 起来
如果全部 flip 完后才询问，可以直接按上述方法做完后算好每个灯的状态直接保存下来，询问直接输出. O(n+k+q)
如果一边 flip 一边有询问，那么可用 Binary Index Tree 将 flip 和 query 都做成 O(logn)，或者直接用 Segment Tree 或 BST 等数据结构维护区间修改 O(n+(k+q)logn)
    i. 我认为应该是 interval 的操作，将重合部分删去，增加不重合部分
    ii. 每次操作的时候查询某个点是否存在在特定的 interval 范围内即可
    iii. 每次操作仍然 logn

104. 给 2 个 STR A, B。 B 比 A 多 2 个 CHAR，可能是相同的 CHAR， 如何找出？O(1) SPACE, <= O(N) TIME。
a. char 的话就把他映射到 1-256 上。分别对两个 str 算出他们各自的乘积和相加的和： sum_a,sum_b,multiply_a,multiply_b。假设多出的两个 char 分别是 a 和 b。那么 a+b＝sumb-suma, a*b=multyplyb/multiplya; 然后计算应该就可以了，虽然我觉得如果 str 过长会有溢出。。。
b. 直接 XOR。如果为 0，a = b = (sumb-sumA)/2.

105. 求曼哈顿长度的，给一个 2D 数组，里面有 1,2,-1 若干，问所有 1 到 2 的 manhattan distance 最短的是多少。用 DP 补充：每个 1 到所有 2 的距离取最小的，然后再在这些值中取最小的
a. 317. Shortest Distance from All Buildings？
b. 286 walls and gates？multi-end bfs

106. 说有几台 server 收到 client 的请求之后发出 response，要实现一个 response ID generator，满足生成的 ID 是 global unique 和 sequencial，我当时第一反应就是 spanner，就说用个 GPS 一样的全局 timer 生成 timestamp，配上 server ID，组成一个 64bit 的 ID，然后他就问应该是 server ID 在前还是 timestamp 在前？这样做有什么好处？会有什么问题？然后又让我想另外一种设计思路，他给提示是用一个 server 专门负责生成 ID，我就想到 zookeeper
a. 设计一个系统能生成 unique ID，user 的请求速率非常高，必须用 multiple machine.
Follow up: 如何让 ID 可以粗略的按照时间排序
    i. timestamp+userID

107. 给个数组，返回一个数组，返回数组的每个元素是原数组中右边的第一个大于该位置元素的 index （好好用 Stack！！！！）
a. 只记得是用了一个数组（就是 output 的那个数组），然后是从后往前处理了一遍，记录了一些东西，反正最后的复杂度是比 N 大但是比 N 方小，然后 bestcase 复杂度是 N。当时一开始我只写了 N 方的，是面试官给了提示，就说从后往前遍历才做出来的

```
public static int[] nextGreaterElement(int[] nums) {
    if (nums == null || nums.length <= 1) return new int[];
    int n = nums.length;
    int[] indexes = new int[n];
```

```
        Stack<Integer> stack = new Stack<Integer>();
        stack.push(0);
        for (int i = 1; i < n; i++) {
            while (!stack.isEmpty() && nums[i] > nums[stack.peek()]) {
                indexes[stack.pop()] = i;
            }
            stack.push(i);
        }
        while (!stack.isEmpty()) {
            indexes[stack.pop()] = -1;
        }
        return indexes;
    }
```

108. 给你一个数组，从这个数组中挑出最少的数字，它们的和要比整个数组的和的 1% 要大。要求 Linear time 解决

a. pivot 应当是 medium，把大于 medium 的数字放在 medium 右边，如果这些数字的和比 target 大，说明我们需要继续 partition。直到 partition 至比 medium 大的数字比 target 小，说明我们需要一些在 medium 左边的数字，那么就开始 partition 左边的数字，新的 target 是 target-SumOfRightHalf。每次舍弃一半的数字，complexity 是 linear

109. （生成迷宫问题）一共有三类方法：

a. Backtracing: 存一个 boolean[][] visited, 四个 boolean[][] southWall。。。。

i. Make the initial cell the current cell and mark it as visited

ii. While there are unvisited cells

1) If the current cell has any neighbours which have not been visited

a. Choose randomly one of the unvisited neighbours

b. Push the current cell to the stack

c. Remove the wall between the current cell and the chosen cell

d. Make the chosen cell the current cell and mark it as visited

2) Else if stack is not empty

a. Pop a cell from the stack

b. Make it the current cell

b. Kruskal's algorithm:

i. Create a list of all walls, and create a set for each cell, each containing just that one cell.

ii. For each wall, in some random order:

1) If the cells divided by this wall belong to distinct sets:

a. Remove the current wall.

b. Join the sets of the formerly divided cells.

c. Randomized Prim's algorithm

i. Start with a grid full of walls.

ii. Pick a cell, mark it as part of the maze. Add the walls of the cell to the wall list.
iii. While there are walls in the list:
1) Pick a random wall from the list. We now have two cases: either there exists exactly one unvisited cell on one of the two sides of the chosen wall, or there does not. If it is the former:
a. Make the wall a passage and mark the unvisited cell as part of the maze.
b. Add the neighboring walls of the cell to the wall list.
2) Remove the wall from the list.

110. 接着 coding question 是写一个 Union Iterator，parameter 是两个 Iterator<T>，output 应该是这两个 iterator 中 elements 的 union。i.e: it1:{1,2,3,4} it2:{3,4,5} output should be:{1,2,3,4,5}
a. Follow up: 如果这两个 iterator 非常大，RAM 存放不下怎么办.
b. 反正大概意思是存在 disk 上，然后 interviewer 希望我解释一下具体怎么操作。我当时并不会呢傻楞楞的，他说没学过是不是，我说是，然后他就开始 lecture 了一番...

111. sum(n^i) 就是 1+n+n2+n3+...+n^N， 快速写了一个 O(N)之后让我优化，其实这个二分 O(lgN)很容易想的
a. 等比数列求和公式！！！！！！

112. Top K int in a large stream (This can be done in O(n))
a. 2K windows, Quick Select, throw K away.
b. 有 N/K 个 batches， 所以是 O(K * N / K) = O(N)

113. input: int n
function: 将 n 用 2 的指数表示，使得指数表达式的个数最少
output : int num（指数的最少个数）
e.g: input = 28
28 = 2^4 + 2^3 + 2^2 => num = 3
28 = 2^5 - 2^2 => num = 2
所以 output = 2
a. 感觉第二题是 greedy 的思想，从最低位的 bit 开始，只要有连续的 1 的个数超过 2 个，就写成(2^M - 2^N）

114. 游戏，猜字， 给你一个字典, player 1 脑子里想一个 5 位单词，这个单词一定来源于字典， 然后告诉你字典中每两个单词间相同单词的数量，你需要做出最好的猜测， 能最有利于接下来的猜测。。举个简单的例子， 三个单词 a, b, c. 如果告诉你 a 和 b， a 和 c 都差 3，然后 b 和 c 差 2， 你最不应该猜 a， 因为接下来这属于无效猜测。。。。先让写个类似 StrStr 的 function，然后写如何做最优猜测。。

a. 第四面：国人大叔，一直友好和蔼的交谈，整体感觉最放松的一轮。先是一个 warm up 问了一个 guess number 的游戏，用最简单的 binary search。
然后 follow up：如果给的是一列单词，A 心目中有一个目标单词，让 B 来猜。每次 B 猜一个单词，A 只会告诉他猜中了几个字母。举个栗子，如果 A 心中的是 APPLE，B 猜 ANGEL，A 告诉 B 猜中了 3 个字母 （A,L,E，位置无关）。问 B 如何猜最聪明。

再 follow up：如果 A 会作弊故意刁难 B，B 知道 A 会刁难自己，应该如何猜。

b. 我的方法是先建立 Graph，用 hashset，每个 word 是一个 key。Graph[word]是一个 array，在第 i 个位置上代表跟 word 有 i 个相同字母的单词的个数。然后根据 tree search 的理论，平均 search depth 最低的建立树的方法是 balanced tree，我们这里要找的就是 variance 最低的那个 word。

c. It is something related to making the guess that will keep decreasing the size of the result set(the set that contains all the possible string). You can think of a graph that contains strings as nodes. The edges are weighted. The weights are the common characters for two strings. Make the guess that wil result the smallest expected size of the result set after each guess.

d. (DietPepsi) 我的想法这样：邻接矩阵，matrix[i][j]表示单词 i, j 的相同字符个数, 过一遍这个矩阵，找到某一行，它的 element 的最大频次最小的那一个. 这就是最佳猜测
比如矩阵是：

1 2 3 4 5
2 1 2 3 1
3 2 3 3 5
4 3 3 5 2
5 1 5 2 1

那么第一行 1~5 的频次都是 1，这就是最佳选择, 假设这个最大频次的最小值是 m, 那么矩阵就会从 nxn 保证缩小到 mxm, 或者更小

115. design google calendar . 要求分析如何存 data, 如何 invoke user events, 如何 handle 100000events per second, 然后要写了一部分 thread safe 的 code 实现如何 invoke event.

116. 写一个类，类似 hashmap，但是有(key,value,time)这三个参数。 有 set 方法和 get 方法。
调用方法：
set(k1,v1,t1)
set(k2,v2,t2)
set(k1,v2,t2)
Get 方法有两个返回的可能(对应上面的 set 看)
get(k1,t3), if(t1<t3<t2) return v1. if(t3>= t2) return v2. key 不同的时候和 t3< t1 的时候都返回 null

Hash<k,TreeMap<t,v>>

```
Hash<Integer, TreeMap<Time, Integer>> map=new HashMap<>();
Void set(int key, int value,Time t){
        if(map.containsKey(key){
                map.get(key).put(t,value);
        }else{
                map.put(key,new TreeMap<Time,Integer>());
                map.get(key).put(t,value);

        }
}
int get(int key, Time t){
        TreeMap<Time,Integer> cur=map.get(key);
        if(cur.contains(t))return cur.get(t);
        Time t1=cur.higherKey(t);
        Time t2=cur.lowerlKey(t);
        if(Math.abs(t-t1)<Math.abs(t-t2)){
                Return cur.get(t1);
        }
        Return cur.get(t2);
}
```