实现一个蜜汁TaskScheduler. 实现getTask的function。每个task都有prerequisite。
1. MxN 01矩阵迷宫，看有没有 路徑从A点到B点。 Followup：如果矩阵超大需要机群存放，如何设计一个并行算法找到AB间的路径
基本逻辑：
1. 巨大迷宫可以划分成格子存储，每块格子能放进单机memory
2. 从起点S用DFS找到格子上下左右边界上所有可达的点集X。对每一个点，记录位置和方向（方向是指接下来的探查方向，比如说点P在当前格子的左边界上，那么下次就应该继续向左边的格子探查;如果在角上，就要探查两个方向，比如向左和向上）
3. 对于点集X中的每一个没有处理过的点，按照探查方向继续用DFS找到相邻格子中上下左右边界上的可达点，并加入X
4. 最终会hit终点e的格子的边界，再看看从边界上的点能不能达到终点e

并行化：

1.所有的迷宫格子数据存在HDFS，
2.点集X可以用一台server存储，加个RPC存取API
3.分布式的worker的逻辑就是 a)从X取几个点 b)load 相应的迷宫格子 c)找到边界点集存进server

最后又提了下failure handling 什么的，面试官说和他想的差不多，应该算是混过去了

两个list of int ranges， 求加在一起以后的ranges(重合的range合并组成一个新的range)
求点集中离给定点最近的K个点 maxheap, quickselect

1.给一个二位game board 三种cell,能走的，不能走的，打卡点。给初始位置求实现canReach函数返回能否走到任一个打卡点-BFS轻松解决
Follow up: 如果这个canReach一直被call,怎么优化 – 先扫game board，把所有打卡点找到，然后以他们为第一批开始遍历，仍旧是BFS

4a valid IP in string format and return the uint32 format，注意corner case

4b given BST and two vals(lo an hi), return sum of nodes whose val in [lo , hi]。 O(N) and O(logN) solution

```
public void inorder(TreeNode root, int[] sum) {
        if(root == null) {
                return;
        }
        inorder(root.left,sum);
        if(root.val>=lo && root.val<=hi) {
                sum[0]+=root.val;
        }
        inorder(root.right,sum);
}
 private void helper(List<Integer> list, TreeNode root, int min, int max) {
        if(root == null) {
        return;
```

```
    }
    if(root.key>min){
        helper(list,root.left,min,max);
    }
    if(root.key>=min && root.key <=max){
        list.add(root.key);
    }
    if(root.key < max){
        helper(list,root.right,min,max);
    }
  }
```

bianry tree to circular list

```
private void reconstruct(TreeNode root, TreeNode[] prev, TreeNode[] res) {
        if(root == null) {
            return;
        }
        reconstruct(root.left,prev,res);
        TreeNode r = root.right;
        if(prev[0]==null) {
            res[0]=root;
            prev[0]=root;
        }else{
            prev[0].right=root;
            root.left = prev[0];
            root.right=res[0];
            res[0].left=root;
            prev[0]=root;
        }
        reconstruct(r,prev,res);

}
```

117. Populating Next Right Pointers in Each Node II

- salbring tree，不过没有next指针， 你要用原来的left，right指针 Level BFS

```
public void connect(TreeLinkNode root) {
    if(root == null) {
        return;
    }
    Queue<TreeLinkNode> q = new LinkedList<>();
    q.offer(root);
    while(!q.isEmpty()) {
        int size = q.size();
        TreeLinkNode prev = null;
        for(int i=0; i<size; i++) {
            TreeLinkNode curr = q.poll();
```

```
            if(curr.left!=null){
                q.offer(curr.left);
            }
            if(curr.right!=null){
                q.offer(curr.right);
            }
            if(prev == null){
                prev = curr;
            }else{
                prev.next = curr;
                prev = curr;
            }
        }
        prev.next = null;
    }
```

```
postorder BST
public TreeNode reconstruct(int[] post) {
    // Write your solution here
    if(post == null || post.length == 0) {
            return null;
    }
    int[] p = new int[]{post.length-1};
    return helper(post,p,Integer.MIN_VALUE);
  }
  private TreeNode helper(int[] post, int[] p, int max) {
            if(p[0]<0 || post[p[0]]<max) {
            return null;
    }
    TreeNode root = new TreeNode(post[p[0]--]);
    root.right = helper(post,p,root.key);
    root.left = helper(post,p,max);
    return root;
  }
}
```

问一个题，就是给一个array，给定一个输入array，元素不排序 然后生成一个binary tree，还要满足任意节点左右节点都小于父节点，以及保持array是这个tree的inorder traversal结果。
这就很简单嘛，找最小值，分割数组，然后递归的获得左右子节点。follow up是用insert法再做一遍。就是新插入的节点如果大于根节点，根节点就成为其左子树。否则和根节点的右孩子比较。右孩子为空则成为右孩子。
二叉树把0元素都移到最下面，0节点非leave就是所有子节点都为0
一个多叉树，每个节点有三种状态，0，1，2. 要求如果子节点全部为2，父节点也为2. 子节点全部为0，父节点为0，其余情况都是为1.
给你一个已经合法的树，实现一个 改value 的函数（node, targetValue）,难点在怎么连带改其他需要 改值的node

设计一个数据结构实现get(key), put(key, value), delete(key)和getLast()来存放k-v对。这个getLast返回的是上一次操作的k-v对，但是删除的操作不算，因为删除的k - v已经不在数据结构里了。
链表倒着打出来。花式解呗，拿stack存着（跟recursion一样），然后翻转链表，输出， 再转回去。
follow up很神奇，假设以上是个library，多个线程的时候怎么办？加lock... 如何加最好?讨论了半天我也只做到要被翻转的两个点之间加lock然后unlock，面试官说可以只加一次，然而没时间了。反正follow up问得我一脸懵*

given 2d array pixels, how to render a circle

把html parse成DOM tree，找到target string，并且打印出来path。

有一题是phone letter combination的变形，需要实现键盘手机的输入法，1代表abcd，你输入一个1得到a输入两个1得到aa or b， 三个1： aaa ab ba c 诸如此类. 还有一题是实现postordertraversal的iterator.

```
private void helper(List<String> res, String s, int index, StringBuilder sb, String[] array){
                if(index == s.length()){
                        res.add(sb.toString());
                        return;
                }
                int len = sb.length();
                char c = s.charAt(index);
                String temp = array[c-'0'];
                int i=index;
                int count=0;
                while(i<s.length() && s.charAt(i)==c){
                        count++;
                        sb.append(temp.charAt(count-1));
                        helper(res,s,index+count,sb,array);
                        sb.setLength(len);
                        i++;
                }

        }
```

会议室，要求找出需要最多个room的时间interval

```
        public List<Interval> overlap(Interval[] intervals){
                if(intervals == null || intervals.length==0) {
                return null;
        }
              int[] start = new int[intervals.length];
                int[] end = new int[intervals.length];
                 for(int i=0; i<intervals.length; i++){
                start[i]=intervals[i].start;
```

```
            end[i]=intervals[i].end;
        }
        Arrays.sort(start);
        Arrays.sort(end);

    int count=1;
    int temps = start[0];
    int tempe = end[0];
    int j=0;
    Map<Interval,Integer> map = new HashMap<>();
    for(int i=0; i<intervals.length;i++) {
            if(i+1 < intervals.length && start[i]<end[j]) {
            count++;
            temps=Math.max(temps,start[i]);
            tempe = Math.min(tempe,end[j]);
    }else{
            map.put(new Interval(temps,tempe),count);
            count=1;
            temps=start[i];
            tempe=end[j+1];
            j++;
    }
    }
    int max = 0;
    List<Interval> res = new ArrayList<>();
    for(Map.Entry<Interval,Integer> entry: map.entrySet()) {
            if(entry.getValue()>max) {
            max=entry.getValue();
            res.clear();
            res.add(entry.getKey());
            }else if (entry.getValue()==max){
                    res.add(entry.getKey());
            }
    }

    return res;
    }
```

覆盖问题，一系列range，要求返回重合最多的range，比如 [1，5]，[2，7]，[9，11] 返回 [2，5]

s1,s2,e1,s3,s4,s5,e2,....

```
    public int overlap(Interval[] intervals){
            if(intervals == null || intervals.length==0) {
            return 0;
```

```
}
        int[] start = new int[intervals.length];
            int[] end = new int[intervals.length];
             for(int i=0; i<intervals.length; i++){
            start[i]=intervals[i].start;
            end[i]=intervals[i].end;
         }
         Arrays.sort(start);
         Arrays.sort(end);


            int i=1;
            int j= 0;
            int count = 1;
            int maxcount=1;
            int time =0;
            int time2=0;
            while(i<start.length && j<end.length) {
            if(start[i] <= end[j]) {

            count++;
            if(maxcount < count) {
                        time = i;
                        time2 = j;
                        maxcount = count;
                    }

            i++;

         }else{
            j++;
            count--;
         }
         }
```

merge two arrays of sorted intervals.
合并两个intervals list;follow up: 给定一个interval list和一个interval，生成他们的overlap list。
For example,
A: [1,5], [10,14], [16,18]
B: [2,6], [8,10], [11,20]

```
int i=0;
int j=0;
Interval prev = null;
if(A[i].start < B[j].start) {
        prev = A[i++];
```

```
}else{
            prev = B[j++];
}
while(i<A.length || j< B.length) {
            if(j==B.length || i<A.length && A[i].start < B[j].start) {
                        if(prev.end<A[i].start) {
                                    list.add(prev);
                                    prev.start = A[i].start;
                                    prev.end = A[i].end;
                        }else{
                                    prev.start =Math.min(prev.start,A[i].start);
                                    prev.end = Math.max(prev.end,A[i].end);
                        }
                        i++;
            }else if(i==A.length || j<B.length && A[i].start>B[j].start){
                  if(prev.end<B[i].start) {
                                    list.add(prev);
                                    prev.start = B[i].start;
                                    prev.end =B[i].end;
                        }else{
                                    prev.start =Math.min(prev.start,B[i].start);
                                    prev.end = Math.max(prev.end,B[i].end);
                        }
                        i++;

            }
            }
            list.add(prev);
}
```

几个user， 每个user 在下面的时间段发了信息，找一个最短的时间窗使得每个user都发了至少一条信息。应该是利口原题吧 smallest range

```
  [[1,3,4,7, 10],
   [4, 6, 8 ,10, 20],
   [7, 15, 16, 20, 25]
  ]
public int[] smallestRange(List<List<Integer>> nums) {
        if (nums == null || nums.size() == 0) {
            return new int[0];
        }
        int[] res = new int[2];
        PriorityQueue<pair> minheap = new PriorityQueue<>(11,new mycomparator());
        int max =Integer.MIN_VALUE;
        int min = Integer.MAX_VALUE;
        for(int i = 0; i < nums.size(); i++) {
            max = Math.max(max,nums.get(i).get(0));
```

```
            minheap.offer(new pair(i,0,nums.get(i).get(0)));
        }
        while(minheap.size()==nums.size()) {
            pair curr = minheap.poll();
            if(max-curr.val < min) {
                min = max-curr.val;
                res[0]=curr.val;
                res[1]=max;
            }
            if(curr.col+1 <nums.get(curr.row).size()) {
                minheap.offer(new pair(curr.row,curr.col+1,nums.get(curr.row).get(curr.col+1)));
                max=Math.max(max,nums.get(curr.row).get(curr.col+1));
            }
        }
        return res;
    }
    class pair{
        int row;
        int col;
        int val;
        public pair(int i, int j, int num){
            this.row = i;
            this.col=j;
            this.val = num;
        }
    }
    class mycomparator implements Comparator<pair>{
        @Override
        public int compare(pair one, pair two) {
            if (one.val == two.val) {
                return 0;
            }
            return one.val<two.val?-1:1;
        }
    }
```

interval [startTime, stoptime) ----integral time stamps

给这样的一串区间 I1, I2.....In

找出 一个 time stamp 出现在interval的次数最多。

startTime <= t< stopTime 代表这个数在区间里面出现过。

example： [1,3), [2, 7), [4, 8), [5, 9)

5和6各出现了三次，所以答案返回5，6。 (Hard)

- Variant: 一串start time - end time，格式是Apr 2010 - Mar 2011这种，要求计算出这些时间的总跨度，重叠的跨度不重复计算。举例：["Apr 2010 - Dec 2010", "Aug 2010 - Dec 2010", "Jan 2011 - Mar 2011"]

面经题 (a) Given a big dictionary (has ~1Mwords) that has only [A-Z] and a string of characters, named “input”, onlycontains [A-Z]. find all the words that can be formed by the characters in theinput. (b) the input is a list of words. Return a list of lists of words thateach list is formed by exactly the characters in the input list. For example: two lists {“DEBIT”, “CARD”} and{“BAD”, “CREDIT”} are formed by the same exact group of characters. 写（b）的时候出了个bug，

第一题sliding window最短包含字符串那个原题，因为做过讲了下算法就开始写完了，问了怎么优化，我说可以把哈希开小点，如果字符数只是英文字母的话。他看了表发现还剩二十多分钟，就说再来个题，实现一个线程安全的队列。我就懵逼了，这不按套路出牌啊。我一开始写了个普通版本，然后跟他说poll和offer两个函数分别加同步锁。他又说如果队列满了怎么办，我们需要把整个队列阻塞掉，又加了判断满不满的函数。然后他说我希望这个队列是个环形的，我又加了模操作重写了判断空的方法。然后又问如果队列满的同时，我还想可以poll但是不能offer怎么办，我想了想再加个变量锁判断是否可以进行poll，如果这个锁没有被持有，就可以进行offer。后来时间不够了，他说我明白你的意思了，no worry。也没照相，估计挂这轮上面了

矩阵里面包含1的最左边列数.

双向列表删除一个节点.写完还剩时间,又问单项列表怎么处理,开销多少.

给一个有正有负的递增数列，返回一个按绝对值大小排列的数列

给一个矩阵迷宫，找到两点之间最短路径，返回路径

给一个字符串比如 12345=67 可以在等号两边随便加加减号使得等式成立，求所有的方法
循环有序数组里面找一个数，话说我总是在onsite的时候被问到这道题。。。follow up是如果数组有duplicate咋办

2. 给一个无穷大网格，给一个起始和终点，还有一些格子是不能通过的，求起点到终点的最短距离

eg: {1, 2, 3, 4, 5}, target= 1239, return: "1234+5", 可以用+或者-,print all possible ways

有向图找环，有环就移除变树，如果没有环就加一条边变图，最后来句最好变出来的图的面积看起来最大

Sort a nearly sorted (or K sorted) array

给一个int array, 大致是sorted，返回是否只有两个数进行对调了。follow up 是如果array 很大怎么办
，

第二轮coding，LC341，follow up如果self referencing list 怎么办。比如
Python:
a = [1]
a.append(a)
flatten(a)

lfatten的时候会死循环，需要检测并报错。

是一个从来没见过的题。说给一堆job的dependency，比如A->B, A ->C, B->D，等等，实现一个两个interface。第一个是get_next_stage，返回下一阶段可以并行的jobs，第二个是job_done，告诉你哪个job完成了。我的第一反应是topological sort，但是紧张之下忘了怎么实现了，卡了半天。interviewer提示说不需要sort。后来反应过来原来每个stage只需要找到没有任何dependency的job。但是没有时间coding了。

写一些trie的实现，包括加一个单词和删一个单词。注意，在删除的时候，要求把不需要存在的node也跟着删掉，不是标记删除就可以了
两个BST，从小到大输出成一个list，讨论空间复杂度，最坏和最好的情况

给俩排好序的数组，找出两个数组的最小差，比如 [3 5 7] [5 9 12], 最小差是5-5=0. 双指针从头到尾扫一遍，O(m+n) 秒了。
find median of n sorted array

给一个数组, 满足以下性质, 对A[i], 有 A[j] > A[i], if j - i >= k, 然后sort这个数组. 我给的解法是heap, O(nlogk)
给 一堆 长方形 求生成随机点。

第二题给的是两个list of string 要求从第一个里面找sublist用来cover第二个list

题目是Print All Paths in a 2D Board的变种，区别在于array里存了0和1，0代表能通过，1代表不能通过。第一个follow up是能不能变成在4个方向移动，第二是输出最短的路径

首先溜贰一这一道题跟蠡口的有点不太一样，是要求有序的，比如AABC 你要先把AA执行之后才能执行B再执行C，所以用一个hashmap来记录就好了。一开始我就给的O(totalTaskTime)的解法，然后优化到了O（n），然后问我能不能优化空间。我说应该可以用queue来优化到 O(cooldown) 的空间，但是这里我想复杂了一直想着怎么只用一个queue来实现，最后没答上来怎么优化，完了后网上搜了下发现也就是额外保持一个size为cooldown的queue来保存当前需要cooldown的task，size大于cooldown的时候把最开始的task再从hashmap removed掉。

给一堆line segments，然后merge。其实也算是第一题的follow-up，有点理解错题意了，小姐姐提醒后，反应过来。只需要merge y = k * x + b，k 和 b相同的line segments，这样这题就比较简单了。我以为只要x y相交，就需要merge。

第二题是“一个编辑距离”的变体，但其实这题有点难度，输入没有给两个字符串，而是两个自定义的interface,里面只有 next()一个方法，next()返回0的时候就是 end,我的做法是再实现一个 peek()方法,然后调用这两个方法的时候有点绕晕了，最后差点没写完，不过还是写完了，面试官说是 work 的

股票题，每天都可以买，可以一起最高点卖出，求最大收益

只有三轮不知道是不是不招人了随便面一下。但是各位大佬认真的很好。两个国人大哥做题，一个白人大叔主要behavior。
第一个把下面的格式的字符串变成树：node有键，值和子树。比如第一个的键是node1，值是‘aaaa'
<node1>aaaaa<node2>bbbbb</node2><node3>cccc</node3><node4>dddd</node4></node1>
第二个白人大叔主要就是聊天最后剩了点时间做了一个把：abababcdcdee变成03ab02cd01ee的压缩。都两个两个的考虑
第三个国人大哥在k个排序list里找中位

yexiaojiaycc 发表于 2018-3-7 14:59

楼主，能不能具体说一下输出非二分图的connected component?

比如 1 - 2 - 3　　4 - 5
　　____/
输出 1 2 3，因为这部分不是bipartite， 4 5 不用输出 因为是bipartite

decode way　follow up：空间优化，backtracking 打印出所有 方式

给你一个有向图，这个图只有一个点没有入度，有多个点没有出度

要求输出所有0入度点到所有0出度点的路径。要求所有路径里面不允许有cycle. 0入度点是已知的输入，0出度点未知

第一题： 给一个只含有‘{’和‘}’的字符串，判断是否是balanced的

合法：{ { } }

不合法：{ { { { } } ；{ } } {

第二题：给两个有序整数数组，返回他们的交集

比如给{0, 1, 3, 5, 7}，{1, 4, 5, 8}，输出{1, 5}

给了一个有向图问有没有环，如果有环能不能删除边变成树，对树没有要求说是任意树。如果本来结构就是树能不能变图

2. 给你一堆飞机票排序，A--》B， B--》C 输出 A,B,C 我直接说用拓扑排序，他说太复杂了，我就蒙蔽了，跟他说怎么复杂了？你想让我优化时间复杂度还是空间复杂度，他说就是太复杂了。后来搞了半天才明白他的意思是每个城市只会在他的输入中出现一次，不用统计入度。我最后还是用dict 存a-->b 关系，用set找出start point， 然后进入dict从头走到尾。

1. LC523. Continuous Subarray Sum .
follow up:考虑negative number case

Input: [3, 5, 11] (prime numbers)
Output: [3, 5, 11, 15, 33, 55, 165] (all possible product)

给一个int[], 输出所有 A + B = C + D 的unique index pairs, A B 下标不同，CD 下标不同， A B C D 输出以后，就不能输出C D B A了 但是可以ABDC 和BACD这种

第三轮：欧洲小哥，问了两题（都是lc原题），第一题有corner case没考虑，在提示下修正，第二题又问了lc偶五三（有可能补了第二轮的短板），这回有时间写，写了用pq的最优复杂度算法，并解释了一下为什么是这个复杂度。最后有个bonus problem，是关于第二题怎么并行的做，也基本答上来了。第二题我也是先写了pq，然后给他仔细解释了一下他说应该是对的，然后问有没有其他方法，马上掏出line sweep，用一个count记录过程中遇到的最大值再返回，大叔一直重复说了好几次good，我猜这个应该就是他想要的答案吧。。。
先按开始时间的顺序排序，然后把数据按照开始时间切分成两部分，第一部分的所有开始时间在区间[S1, S2) 第二部分在[S2, S3) 那么为了计算第二份的结果还需要知道第一份中结束时间在[S2, S3)之间的数据（需要线性时间的计算量），并把在范围内的结束时间传给处理第二部分的机器（通信）。

通信完成后，两个部分就可以同时run得到结果了。

同理如果要把数据分成多份（比如三部分），第一部分会做两份结束时间数据分别传给后面两部分的机器，第二部分会把部分结束时间数据传给第三部分，第三部分接受到两部分结束时间数据后会在把他们合并成一个初始的pq。

值得注意的是如果一个会议的开始时间在第一个分割范围，结束时间在第三个分割范围，那么它是不会参与到第二部分的计算的，所以这里还有一些细节

m*n matrix，每个坐标都有一个character，找到所有左上角到右下角的路径（只能向右向下走）并打印，比如a->b->c。 第一遍 String dfs秒了。followup优化用stringbuilder，秒了。followup再次优化用char[]，稍稍卡了几下（算index算的有点儿慢），大哥给了小提。剩下时间问问题。给大家个hint，因为每个格子都是一个char (比如a b c)，所以最后总的length是固定的。那么我们是不是可以直接allocate一个fixed sized char[] 呢？

面经高频，给一个01矩阵，找最左边的1

例如. Waral 鍗氬鏈夋湁澶氫氧构绔嬶紝
000000
001111
011111
000011
最左边的1在第二列.给的矩阵1是连续右对齐的

给定一个数组，数组里每个元素是一个(x, y) pair，找出数组里的两个pairs，这两个pair满足 x1 + x2 = k and y1 + y2 = k

说给一堆job的dependency，比如A->B, A ->C, B->D，等等，实现一个两个interface。第一个是get_next_stage，返回下一阶段可以并行的jobs，第二个是job_done，告诉你哪个job完成了。我的第一反应是topological sort，但是紧张之下忘了怎么实现了，卡了半天。interviewer提示说不需要sort。后来反应过来原来每个stage只需要找到没有任何dependency的job。但是没有时间coding了。
贰凌凌的变形，问所有小岛面积的中位数

给你一个unsorted integer array（array 很大，你只有一个iterator）和 一个用double表示的百分数（比如 0.4 代表40%），输出：sort array后处于40%位置的那个数的值。例子：输入[1 3 2 5 4 6 7 9 8 10], 和 0.6， 你就返回6. 方法就是bucket sort嘛，注意一些corner case， 比如输入0.3333什么的注意四舍五入。

Coding 美国小哥。 给一个dict, 一个target， 和三种运算"+,-,*", 问最少需要一次操作。
0<=target<100k. 举个例子，dict=["1"], target="121", 需要一次操作，因为11+11 = 121.
第三面最开始有一道题让copyArray。 给个boolean[][] M, int topLeft x, int topLeft y, int bottomRight x, int bottomRight y, int dx, int dy. 让把topLeft到bottomRight的矩形的值偏移个dx, dy到M
找到包含两个字符的最小长度， order matters
一道题目不太常见，但地里也有人提到过。 求最长等差数列个数。
1， 3， 2， 5，7，4，10。 答案是4，（1，3，5，7）。 用DP 就行
trie的实现，包括加一个单词和删一个单词。注意，在删除的时候，要求把不需要存在的node也跟着

删掉，不是标记删除就可以了

给一个图，给一堆Node， 找Strongly Connected Components;
兩個向量做內積，follow up 是 sparse
sorted array 没有重复值， 也没有绝对值相等的值 返回一个绝对值 sorted array

例子 [-9, -7, -5, 0 ,1 , 2 ,3 ,4, 8]
返回 [0,1,2,3,4,-5,-7,8,-9]

方法 两个 array 分别存取 正负值， 对比 负 value 的 0 和 正 array 的 length-1，put into return array

LC 45, Jump Game II,
follow up 是找出具体的路径

```
 public int jump(int[] nums) {
    if(nums == null || nums.length==0) {
```

```
        return 0;
    }
      int sc = 0;
   int e = 0;
   int max = 0;
   for(int i=0; i<nums.length-1; i++) {
      max = Math.max(max, i+nums[i]);
      if( i == e ) {
         sc++;
         e = max;
      }
   }
   return sc;
   }
```

给一个数组，find max sum of 3 elements

Follow up: find max sum of x elements

题目是 continuous subarray sum，就是在一个非负数的数组里找到连续的一个子数组，然后加起来等于一个 target，lz 用的是两个 pointer 来解的，一个指前面，一个指后面，然后像窗口一样扫了一遍，只需要返回是不是 true；
follow up 是数组可以有负数，lz 用的是 hashSet，复杂度是线

给个 prime 数组，输出所以可能的乘积，和 subset 做法一样。follow up
是如果输入数组有重复，输出不能有重复。所有的题都问了时空复杂度

LC 227 簡化版 evaluate 一个只有数字、加号和乘号的数学表达式的字符串

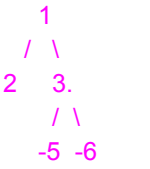
012
345
678
.9.
给定以上 2d array 作为棋盘，起始位置为 0，移动方式为走 L 型（0 可以到 5 和 7，1 可以到 6，8， 3 可以到 2 和 8）。问给定步数 n，有几种走法。

给定一个排好序的 int set 和 一个值 k 计算# of subset 满足 min(s)+max(s) < k
样例 [2 3 4 7] 7
答案 5
解释 [2] [3] [2 3] [2 4] [2 3 4]

第二轮看起来是个新题: given 2 list of interval representing users online offline timestamp e.g (already sorted), find all intervals that both of users are online.
e.g A= [(3, 5), (7, 11)] B=[(2, 4), (9, 10)] --> [(3, 4), (9, 10)].

第一题压缩 aab->2a1b

2. 给一个 tree，返回 subtree 的最大值；如：



返回的值应该为 2; (就是 node 2 组成的 subtree)

word break，要找到最少的分割次数

如何判断两个图是否同构，目测利特扣得上面没有这题（小伙伴们再 double check 下？）。
我告诉他这题是 NP Hard 的，可以用 DFS，他说对那你给我写一个。我花了大概快半个小
时才写完，中间也出了不少 bug，经过他提醒才改过来的

给了一个n*n的矩阵，每一行都是排序的，但列不是，求中位数
http://www.geeksforgeeks.org/find-median-row-wise-sorted-matrix/

implement circular array

几何算法问题。如果给你一堆的矩形， 求重合矩形重合最多的坐标位置。我上过一个算法课，大概思

路就是做一个二维的meeting room II

给定N个2D坐标（可以设想为餐厅的位置），要求输入任意坐标，可以返回方圆d距离内的所有餐厅