201 Bitwise AND of numbers range

Review II

201-1 while loop的移位条件，有时候需要看是否是最后一位了，如果是最后一位，不能进行操作以免影响结果！！！

201-2 本题的要点是 找从左往右m，n的第一个不一样的点，其后部分都归零。所以会有sol 2，先同时向右移，找到关键点后再左移回来。

202 Happy Number

ReviewII

202-1 不要猜答案，按照题目的逻辑去做。

203 Remove LinkedList Elements

204 Count Prime

Review III

204-1 请反复阅读本题答案。为什么只有在i=当前为质数时候才开启第二层循环？！为了防止重复！！

205 Isomorphic Strings

Review II

205-1 用map去比较两个string 肯定不能建立两个map：建立一个map作为两个String中字母的对应关系！

205-2 在处理1对1match的时候不只可以用containsKey 还可以用containsValue

207 Topolocial Sort for map

Review III

207-1 Tushar’s lecture about topological sort: 用set和stack可以实现topological sort.一旦有环 则做不到

207-2 方法1：建立hashMap存储dependent关系。然后遍历每个元素并且dfs她的dependent看有没有重复的元素！！！注意如果某个元素及其上游检测ok，可以标记并减少重复的查找。所以这里存储已经遍历过的元素不用hashSet，用boolean[] array 并且有-1 0 1三种可能性代表三种可能！！！

207-2 方法2： 逐个排查归类法：BFS逐渐排出每个元素！首先建立数组记录元素的dependency个数，最后结果为0的自动计入已完成的队伍。然后看谁depend on 已完成的元素，并且减去相应的值在dependency的数组中，直到该元素在数组中为0。最后看是不是所有元素都归类了

208 Implement Trie (Prefix Tree)

Review II

208-1我的解法非常好。采用randy的chaining return基本的操作单位可以连续调用，读起来非常清晰！

208-2 看到了所有的输入都是char其实可以想到用int[26] 肯定是另一种解法：所以child可以是一个TrieNode数组： TrieNode[] children = new TrieNode[26];

209 Minimum Size Subarray Sum

Review III

209-1 Slow Fast 指针对于数组中连续部分的最值

209-2 连续部分数组(start,end)的值等于数组的头到end的sum减去从数组的头到start的值。对到数组中任何一点作为end的sum值进行操作，所以想到binary search

210 Topological Sort II

Review III

210-1 需要顺序的输出没有dependency的点，想到dfs，想到queue。把又有点分成两类：保证无loop的和有loop的逐渐收敛

210-2 Topoligical Sort 实际是dfs。逻辑：dfs时候，在同一loop中发现同样点，证明有loop返回false。另外就是在其他前面遍历的loop中（不是当前loop）发现的点，已经visited不用再次visited,可以直接返回true。用stack记录点，注意：只有当没有下一层的时候 再入stack，所以stack加值要在dfs的最外层

210-2 如果是单向的只是加元素，可以用hashSet。但是如果随着程序的运行存在与否经常改变，请用boolean[] array

211 Add and Search Word – Data Structure Design

Review II

211-1本题是209 的变种。在逐层(char)寻找的过程中，如果某一层的结果数目不定（可能有多个结果需要遍历），这时候就不能采用iteration 进行dfs的方法。必须采用recursion

212 Word Search II

Review III

212-1 Trie 的好处是运用startwith和search可以迅速做出判断是否符合。所以当题目中要求判断的内容有大量重复操作时候（如本题需要单独判断每个单词）可以先建立trie based on 要判断的内容。

212-2 在涉及到判断是否含有制定的单词的时候startwith(str)是一个非常好的可用的函数。

213 House Robber II

Review II

213-1 有环和无环的区别就在于是否要取第一个、最后一个元素需分两种情况讨论

214 Shortest Palindrome

Review III

214-1 大的问题分步走。本题第一步：找到以起始点开始的最长的palindrome, 然后补齐后边的。注意：要找最长的，那么是从大到小找还是从小到大找？！

214-2 java可以accept的解法一：逐渐排除法。要找palindrome最长的，可以用i j的关系逐渐缩短其长度之间的关系

214-3 KMP 算法！

KMP算法original：参见youtube tushar视频，找的是一个String中 prefix==postfix substring的最大长度。由此找原来String从头开始最长的palindrome，变成了把原来string reverse并且接在原来String后边然后取新的combined的String的最大长度

215 Kth Largest Element In An Array

Review III

215-1 Comparator函数如果比较的是primitive type 注意一定要用其object的形式！比如public int compare(Integer i1, Integer i2){}

215-2 PriorityQueue的容积要考虑清楚。如果逆向解决问题 容积不能为逆向的值！！！

215-3 根据类似quicksort来做partition、从而可知n个元素中前T个元素小于后n-t个元素，比较T和k从而决定下一步怎么走

216 Combination Sum III

217 Contains Duplicate

217-1 是否有重复问题条件反射：Arrays.sort() ， HashSet.add()

218 Skyline Problem

Review III

218-1 当一个区间的开始和结尾都会影响高度的时候，不如把区间的线换成点起点和结束分别成为一个点，然后另外一个boolean参量来表示当前点是起点还是终点

218-2 对于compare函数不同的位置写法不一样。当new Comparator<>作为Collection.sort()的参数，其参数就是实际要比较的对象的类型，比如public int compare(Integer i1, Integer i2).如果作为PriorityQueue的参数则要写为public int compare(Object o1, Object o2). 如果在class本身中定义。名称叫compareTo，参数就是class的类型.如果放在class本身里面，class需要implements Comparator<classname>

218-3 用PriorityQueue记录当前的最大元素的时候第一件事情就是PriorityQueue写入的element是否允许有重复！熟记！Priority 允许输入重复的元素，哪怕是Integer（这个是queue的特性决定的）

218-4 public int compare(Integer i1, Integer i2) 如果return value小于0， i1“优先”

， 如果return value大于0，i2“优先”。

218-5 optimization1 : Collections.sort 中的 Comparator可以写在class Point当中

218-6 TreeMap 的几个重要函数：treemap.firstKey() treemap.lastKey() treemap.firstEntry() treemap.lastEntry()

218-7 Discussion 中的height数组：用height的正负表示是起始还是结束。

219 ContainsDuplicate ||

219-1 间隔为k则从开始i开始算最后一个符合要求的坐标为i+k-1，下一个为i+k 坐标关系搞清楚。

219-2 对于前k个特殊处理可以写在for loop里if(i>k)对于i的一些特殊值就行特殊操作。如sol 2

或者用hashMap记录位置，这样就不用管delete操作了，但是存储空间会需要多一些。

220 ContainsDuplicate III

Review III

220-1 initial想出来slow fast指针乘以一层内部遍历。为了减少时间复杂度，考虑将一层内部遍历改为HashMap或者BST处理

220-2 sol 1: TreeSet的本质是一个maintained的BST。可以做到insert, search 和delete都在log n 完成。

而我们这里对slow fast指针之间的nums的操作恰恰只有insert delete 和get a value nearby (~search)。所以treeSet是合适的DS。注意set特有的ceiling floor函数可以用来找最近点。TreeMap不能声明为Map因为这两个函数是treeMap特有

220-3 分段问题 最短间距 自然想到bucket。

首先，bucket的宽度是t+1。因为反正需要二次验证，所以宽度+1避免边界情况。第二，为了保险，bucket的number都为证 所以求bucket的number时候用以下等式： ((long)nums[i]-Integer.MIN\_VALUE)/((long)t+1)

221 Maximal Square

222. Count Complete Tree Nodes

Review III

221-1 sol 2 Feature of Complete tree: Height can be easily got from getting the left。所以在某一点分类讨论做递归！

221-2 sol 3 把递归通过变化root的值用Iteration表示出来

221-3 简化的关键是：如果有一些情况可以直接算值即可直接算。Sol2 sol3 是在每一层，一半继续递归，一半直接算结果。Sol4 是开始的层数都要递归，但是到特定层数开始左右相等了，直接算结果。

221-4 本题核心！！通过比较左右高度大小而决定是否剪枝logn

221-5 (1<<n)-1注意！！！<<左右的内容必须加括号。因为加减的优先级都要比>>强！！！请思考下列表达式的每一层的括号都是必须的 return (1<<(height-1))+ countLastLevel(root.right,height-1);

223 Rectangle Area

Review II

223-1 用Math.min / max 统一化结果

224 BasicCalculator

Review III

224-1 How to convert an integer to a string: Integer.toString(str) or String.valueOf(int)

224-2 Non-Negative number ！！！在存储到stack的时候可以通过存储数值的正负来存储其他的意思（比如符号）

224-3 while loop使用结束后 要post processing最后的操作看是否还有新的加减操作需要完成。 While loop always need post processing

224-4 对于string读数有两种处理 之一是StringBuilder() 之二是\*10+ %10 –

224-5 如果判断是不是数字可用此函数： Character.isDigit(c)

224-6 一维的Stack可以存储二维的信息，只要这两种信息可以用一种数据结构表示。如本题stack每个操作都是存两次或者pop两次。把数值和符号都用Integer表示

225 Implement Stack using Queues

225-1 一般情况下，Queue在linkedlist的实现是在list尾部加 在list头部出 想想为什么！sol 1 arrayDeque

225-2 Sol2: If only Queue API is what we can use, what we can do: 每加入一个元素的时候，先加此元素到新队列，再将原队列全部输出重新输入，这样顺序就是反的了。小递归。Queue的size可以用来forloop处理

227 Basic Calculator II

Review III

227-1 Sol 2: 如何分辨乘除和加减的不同操作：每个点我们记录两个值：preval 和curval， 如果是+-直接update两者，如果是乘除，只update preval直到加减时候再update curval。Sign在每次处理完数以后再处理

227-3 Sol 3本质上和sol 2 是一样的。关键在于如果是乘除只需要在原来的元素上加工，还不需要并入最后的结果。但是如果是加减，可以随时更新总的sum结果。

227-4 总结： stack和prevalue是很好的可以解决优先问题的方法。

228 Summary Ranges

228-1 my solution is great.遇到分类讨论的情况，如果能用一个函数表示出来，可以简化流程

229 Majority Element II

Remember this question only. I do not think the question is meaningful.

230 Kth smallest element in BST