### **Week 9 Day 2**

### **131 Palindrome Partition**

### **Review II**

### **个人分析**

### 131-1空集算不算substring：由题中示例发现：不算

### 131-2 小的palindrome一定是包括在大的palindrome中的。对于每一个点，小的palindrome会decide大的palindrome。切割的地方和大小都不定，所以本题**不能**用divide and conquer。用DFS。DFS的关键是起始点。DFS的内容是从起始点开始的下一个palindrome元素

### 131-3 DP时候一定注意 用charAt的时候对应dp矩阵的n要减一！！！

### **132 Palindrome Partition II**

### Review III

### 本题注意半个Matrix的情况

### 132-1 Tushar的解法直接在matrix上算数，稍微复杂但是很直观。Coding时候注意两个loop外层是长度l内层是start point

### 132-2 program creek解法。对于palindrome的dp : s.charAt(i)==s.charAt(j) && ((j-i)<=1 || dp[i+1][j-1])

### 可以把所有的palindrome收尾找到。然后用cut[i] 一维dp进行递归最少的层数。注意 这两部是在一个大的双层loop中同时实现。（当找到一个dp为true时立刻更新对应的cut） forloop的关键是：以当前为end的前面的序列中。

### **133 Clone Graph**

### **Review III**

### 133-1 copy必然要对原有的图进行遍历：dfs用queue

### 133-2 为了解决两个点间有多条线设计重复的情况。简历从original到copy的hashMap确定所有点只有一个。中间的连线根据所有的遍历进行！

### **134 Gas Station**

### **Review II**

### 134-1 对于这种在序列中重复操作寻找可能性的问题，最重要的是如何剪枝！ A🡪B🡪C 如果A不可以到达C那么B也不可以 本题阐述了 greedy的做法之一：从数学上剪枝

### **135 Candy：**

### **Review II**

### 135-1 关键词：相邻：所以正着顺序，反着顺序 分别遍历一遍即可

### 135-2 本题阐述了greedy的做法之二：双向greedy必然能得到一个数组 / 数列的最优解。注意在第二次（反向）greedy时候要保持第一次（正向）的最优解。

### **Week 9 Day 4**

### **136 Single Number**

### **Review III**

### Bit Manipulation 经典题

### **137 Single Number II**

### **Review III**

### 137-1 把每个bit看做他们自己分别相加（抑或），整体结果出现的次数和该数组出现次数n相同。所以如果模n，该数的影响就会取消，结果只剩下其他数的影响。请看leetcode discussion答案

### **138 Copy List with Random Pointer**

### Review III

### 138-1 sol 1 对于copy list（包括环）可以首先遍历原来的list的所有点，建立旧点和新点的hashMap一一对应的映射，然后第二次遍历再把点与点的对应关系写好。

### 138-2 sol 2 copy list的第二种方法是先把新的点放在原来的点后面组成一个原来长度二倍的list，再将其分离开来

### Week 9 Day 5

### **139 Word Break I 140 Word Break II**

### **Review III**

### 139-1 DP解法的核心是，对于到达长度为n的解能不能从0至n-1的解中加工得到

### 140-1 DFS解法注意用hashMap存出一些中间的结果从而避免对同一需求在recursion时候反复去求解（注意要带记忆）。比较sol 1 我自己写的recursion和sol3 标准答案的区别

### 140-2 DP解法我自己写的sol2 也很好。dp过程从1到n的每一步都有result。后面的result需要参考前面的result。这时候需要copy注意StringBuilder一定需要的是deep copy。每个copy元素分别new初始化再加上。注意 Collections.addAll()和 new ArrayList<>(anotherlist)都是shadow copy！

### **141 Linked List Cycle**

### **142 Linked List Cycle II**

### **143 Reorder list**

### **Review I**

143-1 Update big problems to small problems. Like reverse list, merge list .etc.

144 BT Preorder Traversal

145 BT Postorder Traversal

146 LRU Cache Design

Review III

146-1 想要get O(1)必然hashMap 同时次序发生变化。必然double LinkedList remove and add

146-2 注意本题定义inner class的format

146-3 初始化linkedlist加一个dummy的head和tail十分有用

146-4 如果delete value注意 hashMap中的内容也要delete

147 Insertion Sort List

Review II

147-1 永远注意！对于list， insertion就是建立一个新的list

147-2 不要使用sol1 去前面的一个指针，出问题的情况太多。

148 Sort List

### **149 Max Point On a Line**

Review III

149-1 从i=0 双重循环， j=i+1 🡪 end,注意duplicate和 vertical (斜率为无穷的情况)都要分别考虑。

149-2 求斜率时，注意要保证被除数和除数均为double slope = (double) y / (double) x;

149-3 最后的max是取vertical的和普通slope hashMap的和

150 Evaluate Reverse Polish Notation

Review II

150-1 要对array中的数进行及时的改变、删除操作。可以go through array 同时建立一个list 用 list的set(pos,val), remove(pos)函数

150-2 每次的操作只和之前的两个数有关🡪 stack

150-3 通过本题回忆switch的用法。可以看出switch确实比多个if else 快

151 Reverse Words In a String

Review II

151-1 用空格隔开：首先就要想到多个空格的情况！

151-2 如果不许用trim split 本题怎么做！！！ 先全部reverse再局部reverse最后去多余空格

152 Maximum Product Subarray

Review II

153 Find Minimum in Rotated Sorted Array

Review II

153-1 找变化的点，需要保证变化的点永远在left和right之间夹着！所以在binary search的时候left =mid right=mid 不要用mid+1 或者mid-1。

154 Find Minimum in Rotated Sorted Array with duplicates

Review III

154-1 对于有duplicate的情况，就是当nums[left]==nums[mid]的时候没办法判断变化是在左边还是右边，这时候需要对一遍就行go through 看看有没有低点 如果有 good 如果没有看另一边 这时候是可能o(n)的。

注意最差情况：所有元素都相等 就是O(n)

155 Min Stack

ReviewIII

155-1 My Solution: Set separate min stack to keep track the value of each level’s min value

155-2 Use one stack

156 BT Upside Down

Review III

156-1 Iteration 的解法是一种自上而下的greedy。注意的是每一轮开始需要保存下一轮的left因为指针在本轮的过程中被更新了。W

157 Read N characters Given Read4 I

Review III

157-1 For read action, it is to read chars to char[] array in the argument

158 Read N characters Given Read4 II

Review III

158-1 总体意思就是如果上次读了四个剩下两个没用，这次从这两个开始可以接着用。全局变量储存即可

160 IntersectionOf Two LinkedList

Review I