January 7 2020 Tuesday:

花花酱分类:
DP

LC 70 climbing stars:
fib数列

January 8
LC 746
LC 746 min cost climbing stairs
F[i] 表示到i位置要cost最少是多少
F [i] = min{f[i - 1] + cost[i - 1], f[i - 2] + cost[i-2]}
Follow up maybe : how to use O(1) space

LC 1137 N-th Tribonacci Number
Space vs performance optimisation

LC 303 range sum query - immutable
开数组时默认值为0， 不能开跟nums长度一样的，不然index i的位置找不到对应的值
Sum[k] is sum for nums[0...k-1]
sumRange(i , j) = sum[j + 1] - sum[i]

LC 1218 Longest Arithmetic Subsequence of given difference

- Subsequence VS subarray :
    - subsequence 可以不是连续的，subarray 和substring（这个？？）都是连续的
    - **LC 1 two sum use map to save time (eliminite second for loop) 相似题！！！**

LC 53 Maximum Subarry:
注意最好不要使用 Integer.MIN_VALUE; 因为如果nums的长度为1， 只有一个值的时候就会输出min最小值，或者就是加一个corner， when nums.length == 1 return nums[0]

January 9

LC 121 best time to buy and sell stocks

- 找到最低的，还有这个点之后的找到最高的那个位置

LC 198 house robber
LC 213 house robber || ⇒ exclude 1 and include 1, just this difference with LC198

LC 309 best time to buy and sell stock with cooldown:

- **LC 122 best time to buy and sell stock ||** as long as next is bigger than previous, add to res
- Three states: hold , sell, buy
    - Hold[0] = Integer.MIN_VALUE; sell[0] = 0; buy[0] = 0;
    -

LC 740 delete and earn

January 10 ~ January 12 nothing

January 13:
LC 740 delete and earn:
Something need to ask: are all numbers are positive?
Reduce to 198 by using a bucket, store the nums in a sorted and use this as input of house robber. Space complexity is O(n + r)

LC 790. Domino and Tromino Tiling (DP， hard to understand)

- Counting problem ⇒ dp problem
- dp[i][0] = dp[i-1][0] + dp[i-2][0] + dp[i-1][1]+dp[i-2][2]
    - dp[i-1][1] ⇒ ways to cover i cols, top row of col i is covered
    - dp[i-1][2] ⇒ ways to cover i cols, bottom row of col i is covered
- init: dp[0][0] = dp[1][0] = 1
- 看花花酱还有一个人的视频

January 14

LC 801 minimum swaps to make sequences increasing(DP)

- dfs search used as brute force (TLE) O(2**n) (my dfs can not work, why???)
- Two array to store keep and swap states

LC 139 word break

- **Use hashSet: easy and fast to check if word in dict**
- LC 140 word break ||
    - 需要好好想一想 **dfs + memo!!!!!**

January 15

LC 300 longest increasing subsequence (DP)

- **Java fill a 2D array with 1, can check Arrays.fill()**
- Very straightforward idea for one dp array method, with time complexity as O(n**2).
    - If current is bigger than previous , and dp[i] < dp[i-1] + 1, we can update dp[i]
- **Binary search** in O(nlogn) ⇒ great advance, but hard for me to understand

LC 818 race car (hard)

- **BFS ⇒ 求最小路径， 需要优化 time = O(2^D). 思路比较直接，利用queue 来做**
- **Top-down DP**
- **看评论区里的高票答案**

January 16 ~ 18

Nothing

January 19 sunday:
Nothing

January 21 Tuesday:
LC 1048 longest string chain:

- **Not easy to understand**

Lc 96 unique binary search trees:

- **LC 95 unique binary search trees ||**

January 23 thursday:
LC 1105 filling bookcase shelves: (google, dp)

- dp[i] store 0...i-1 minimum height , so this is 序列型动态规划？？？
- 两种状态：新书来了以后，要么就是新开一行，要么加到之前的一行里面,这样子就可以减少高度，因为两个可以新开一行。height = max（height1, height2）
- LC 1043 partition array for maximum sum:
    - Same idea

LC 72 Edit distance: (hard) **Delete, add, replace, same(nothing to do)**

- Brute force: time is O(3^(word.length))
- Levenshtein distance algorithm
    - dp[i][i] is an edit distance between the first i characters of word1 and first j characters of word2
    - 主要区别就在于i，j位置的字母是不是一样，一样的话
        - dp[i][j] = 1 + min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1]-1) 一样
        - dp[i][j] = 1 + min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1])
    - Levenshtein distance between two words is the minimum number of single-character edits(insert, delete, substitution) required to change one word into the other.
    - DP very straightforward idea

January 24 Friday:
LC 10 regular expression matching(hard):
看那个 easy DP java solution with detailed E。。

- dp(i,j) : does text[i:] and pattern[j:] match?
- if p.charAt(j) == s.charAt(i): dp[i][j] = dp[i-1][j-1]
- if p.charAt(j) == '.': dp[i][j] = dp[i-1][j-1]
- if p.charAt(j) == '*':
    - 1
    - 2
    - 3
- 什么时候 得到长度时要加括号什么时候不加： s.length() or s.length
- LC 44 wildcard matching

January 28 Tuesday:
DP: LC 674, 64, 361, 338
LC 64 minimum path sum: 使用滚动素组的时候要十分小心

January 30 Thursday:
LC 44 wildcard matching
和**LC 10** 很像的一道题，有类似的解决方法

- KMP
- String searching algorithm

LC 97 Interleaving String

- Straightforward 看视频

January 31 Friday
LC 115 distinct subsequences:

February 3 Monday:
LC 115 distinct subsequences:
二维状态的DP
dp[i][j] = num of subseq of s[1:j] equals T[1:i]
Init: dp[0][*] = 1
If t[i] == s[j] : dp[i][j] = dp[i-1][j-1] + dp[i][j-1]
else: dp[i][j] = dp[i][j-1] 要小心哪个是x，哪个是y。这个和前几道很相似

LC 583 delete operation for two strings:
Check out my solution
之前我的方法理解错意思了，这里只能delete，不能加或者换字符

- Use LSC , find lsc then determining the number of deletions required
- 或者use dp 找
- 空间优化， 滚动数组

LC 123. Best Time to Buy and Sell Stock III(hard) 没看太懂九章的解释， p4- 03-dp on sequence 1:03:50

LC 188 Best Time to Buy and Sell Stock IV(hard) 也没看懂

LC 300. Longest Increasing Subsequence

- **注意 O(nlogn)的方法！！！**

LC 354 russian doll envelopes

- **Sort + longest increasing subsequence**

Feb 4 Tuesday:
LC 712 Minimum ASCII delete sum for two strings

- **s1.charAt + 数字的话，可以把charAt的返回字符当作ASCII值**

LC 1187 make array strictly increasing (hard, need to see again)

- **Strictly increasing, 不能有重复**
- **LC 801 minimum swap to make sequences increasing**
- **Java, Map.Entry**

Feb 5 wednesday:
Nothing

Feb 6 thursday:
LC 1143 longest common subsequence:
这道是完全自己做的！

LC 1092 shortest common supersequence:
求string和array长度的区别， 数组不用加括号， string要加括号
先找到LCS， 然后做这个

February 10 Monday:
LC 718 maximum length of repeated subarray
dp[i][j] is longest common end at A[i] and B[j] 画个图就看出来了

Python create matrix is different with java,
Example: dp = [[5] * (3) for _ in range(4)]
print(dp) =
[[5, 5, 5], [5, 5, 5], [5, 5, 5], [5, 5, 5]], so [5] * 3 is column then in range create row
In java int [][] = new int[4][3] can create same matrix , 4 row and 3 column

LC 377 combination sum IV
Recursive one: TLE, straight forward
全局变量： 两种方法

1, private int[] dp;
2. int[] dp = null;

又又又又又又又又又忘记init 了 dp[0] = 1!!!!!

计数题， dp (nums, target) ⇒ all combinations that uses nums to sum up tp target.

Combination sum series
Fill array ‘dp’ with value “-1” Arrays.fill(dp, -1)
Sort array Arrays.sort(dp)

If create the array dp as global, we do not initial “int[] dp = new int[...]” in each public / private function otherwise we met pointNULL error in java

Bottom - up / top - down

Follow up:

- What if negative numbers are allowed in the given array?
- How does it change the problem?
- What limitation we need to add to the question to allow negative numbers ?

LC 416 partition equal subset sum
0/1 knapsack problem

If（a & 1) == 1 说明不能被2 整除
Difference between this one and LC 518 coin change 2. , this one we can not reuse, but in coin change 2 we can reuse .

February 11 Tuesday
LC 494 target sum
Has constraints
Memorized recursion is straightforward way from brute force recursion

Java HashMap. keySet return the key as a set
那个用hashmap的方法比较巧妙， 但是我没理解了暂时
一个帖子： the original problem has been : find a subset P of nums such that sum(P) = (target + sum(nums)) / 2

LC 1262 greatest sum divisible by three

Has constraints

Feb 12 Wednesday:

https://leetcode.com/problems/combination-sum/discuss/16502/A-general-approach-to-backtracking-questions-in-Java-(Subsets-Permutations-Combination-Sum-Palindrome-Partitioning)
LC 39 combination sum, (no need to sort) LC 40 combination sum || (need to sort , no duplicate)
Permutation | and ||

`Java size(), length, length()` **区别**
:https://stackoverflow.com/questions/1965500/length-and-length-in-java

`length` -- **arrays** (`int[]`, `double[]`, `String[]`) -- to know the length of the arrays

`length()` -- **String related Object** (`String`, `StringBuilder`, etc) -- to know the length of the String

`size()` -- **Collection**

Feb 14 Friday
Review LC 322 coin change , forget to init dp[0] 如何判断是0还是1？？？？ LC 377 就是1， 322就是0

刚才LC377 又又又忘记初始条件 dp[0] = 1了

Feb 20 Thursday:
dfs in tries:

LC 118 pascal's triangle:
Add value = arrayList.add(index, value), if no index, add by order
Length = arrayList.size()
arrayList.set(index, value) set value to index and replace original value

LC 208 implement trie (prefix tree) design

- **LC 211 add and search word - Data structure design**

LC 14 longest common prefix:
string.startsWith(), string.indexOf()
startsWith return a boolean value(true or false) , if str is starts with the string
indexOf return the index of value in old string
For example :
String myStr = "Hello planet earth"
myStr.indexOf("planet") = 6, which is the index of planet in myStr

LC 112 path sum || 有一个问题是为什么要remove last item？？？？？

LC 1269 number of ways to stay in the same place after some steps
static int mod = (int)Math.pow(10,9) + 7;
int MOD = (int)(1e9 + 7);
有1D的方法，可以完了康康

LC1230 Toss Strange Coins
Check dp 的长和宽为什么加2？？ 之前好像看到有一样的也是加2

LC 1220 Count Vowels Permutation
Java hashmap use "put" and "get"
Java list use "add"
滚动数组优化！！！！！ 计算顺序很重要
Convert this into directed graph

LC 1049 last stone weight ||

- LC 1046 last stone weight : priority queue
- **PriorityQueue<Integer> pq = new PriorityQueue<>((a,b) -> b-a);**
- pq.poll() will get the first item and also remove it
- pq.size() will get size

Feb 23 Sunday:

LC 1043 partition array for maximum sum

- **LC 813 largest summary of average**

Dp

Memorized + dfs 这里dfs有点不理解

LC 5 longest palindromic substring

- **L**C 516 longest palindromic subsequence: return count number
- This is return one solution

Feb 25 Tuesday

LC 1139. Largest 1-Bordered Square

- **LC 221 maximum square**
  - Brute force is also straightforward, need to review
  - dp[i,j] = min(dp[i-1, j], dp[i, j-1], dp[i-1, j-1]) + 1
    - 这里注意因为我们计算的是i-1， j-1所以可以开dp长度是多加1， 这样让i，j从1开始loop。
    - 可以先预处理第一行和第一列，但是这样的话如果是一行一列，e.g: [[1]]，要在预处理阶段给maxsqLen一个值，即在这里就更新值，不然之后返回为0
  - **优化空间，不是两行之间的关系，是和左，左上，上之间有关系**
- **这道题太难了。。。。。看不懂**

LC 688 knight probability in chessboard

- Cardinal direction
- Orthogonal direction

Mock interview Feb 25 facebook

Intersection of two arrays ||

- Arrays.copyOfRange(array, start_index, end_index)（包括start不包括end)
- Assigns zero to all the index out of range

Longest Substring with at most K distinct characters

- Two pointer
- Sliding window
    - The point is when and where to move the left pointer index
    - **Since in hashmap , value is the index, so minimum value is leftmost index**
        - **Collections.hashmap(values); get minimum value**
        - **int del_idx = Collections.min(map.values());**
        - **map.remove(s.charAt(del_idx)) // 从map里面找到value值最小的那个,然后从map里面删去那个key**
    - **两次写错！！！求 maxLen那里，忘记Math.max()！！！！！**

Find minimum value in hashmap is O(k) time ,

方法二： use dictionary: combine hashmap and Linked list（这道题的solution里面)

- In python: OrderDict
- **In java: LinkedHashMap**
    - Four operations in O(1) time
        - Insert the key
        - Get the key/ check if the key exists
        - Delete the key
        - Return the first/ the last added key/value
    - The first three operation in O(1) time are provided by the standard hashmap, the forth one , by linked list , that is why use LinkedHashMap
        - To implement LRU cache ⇒**LC 146 LRU cache**

Feb 26 Wednesday

Different between hashmap and hashset

- HashSet can not has duplicate value，HashMap can not has duplicate key, but may same values
- Hashset is slower than hashmap
- Initial
    - HashSet<String> set = new HashSet<>();
        - No constant order, just like set
    - HashMap<Integer, String> map = new HashMap<>();
- Insert item
    - set.**add**(“morning”);
        - set.contains(“morning”) ⇒ true
    - map.**put**(12, “morning”);
        - map.**containsValue**(“morning”) ⇒ true
        - map.containsKey(12) ⇒ true
        - map.**get**(12) = morning
        - map.**size**() == 1 ⇒ can change
        - map.keySet() = [12]
        - map.getOrDefault(key, 0) ⇒ 很有用的
- Hashtable and hashmap
    - Hashtable is synchronized , thread -safe can can be shared with many threads
    - Hashmap is unsynchronized and not thread
    - Hashmap allows null key(one) and multiple null values whereas hashtable doesn’t allow any null key or value

ArrayList and array 区别？ arraylist初始化不用定义长度？？？

Arrays:

- **首先，** array are used to store multiple values in a single variable , 在初始化（create）的时候需要给定值和长度，**array length can not be changed**
    - int[] intArray;
    - intArray = new int[5]; 这里需要给定长度，或者:
        - intArray = new int[]{1,2,3,4,5}
- Access array element
    - int a = intArray[0] ⇒ a == 1 access array element by index
- array length
    - int arrLen = intArray.length;
- Insert new element to index
    - intArray[0] = 100;
- **For-each** loop is used exclusively to loop through elements in arrays

- int[] dp = new int[]{1,2,2,3}
- for(int num: dp){..... num is element }

ArrayList:

- ArrayList class is a resizable array, in java.util package
- Initial
  - ArrayList<String> cars = new ArrayList<String>();
- **Add element**
  - cars.add("VW");
  - cars.add(5, "XXX"); ⇒ add XXX to index 5
  - cars.add("Tokyo")
- Access element by index
  - cars.get(0) = VW
- Change element
  - cars.set(0, "Opel"); ⇒ change index 0 value from VW to Opel
- Remove
  - cars.remove(0)
- Size
  - cars.size()
- Loop
  - For-loop by index
  - For each loop : for(String i: cars){....}
- Collections class in array
  - Collections.sort(cars)

List is an interface , need to instantiate a concrete implementation of the interface in order to use it.

- arrayList, LinkedList, Vector, Stack …..
- List listA = new ArrayList<>();
- Insert all elements from one list to another
  - List<Integer> a = new ArrayList<>();
  - a.add(a);
  - List<Integer> b = new ArrayList<>();
  - b.addAll(a);

Collections.min() 可以获得例如hashmap.value()的最小值， 一个list里的最小值等

- List<Integer> aaaa = new ArrayList<>()
- int min = Collections.min(aaaa) ⇒ return the minimum

Add element to int[] array ⇒ by index, can only change value ,not add new length, since length can not change

Length: 用于constant
Size，not constant
String use length()

Feb 26 Wednesday: 做题开始
LC 813 largest sum of averages (partition)

- **LC 139 LC 312**
- 1043. Partition Array for Maximum Sum
- 这种类似的问题有点不好理解
- 记忆化递归
- Dp[k][i] = max avg sum of first i elements (0 ~ i-1) partitioned into k group
  - Init dp[1][i] = avg{a[0] …. a[i-1]}
  - Transfer: dp[k][i] = max (dp[k-1][j] + avg(a[j+1] + … a[i]))

Feb 27 Thursday
!!!! build github.io personal web

Feb 28 Friday
LC 1278 palindrome partitioning ||| (hard)
从后往前看找palindrome的代码：

```
for(int i = s.length - 1;  i >= 0;  --i){
        for(int j = i + 1; j < s.length; ++j) {
                Do something                这样就从后往前所有index之间的值都看
        }
}
```

这题相当于两层dp， 先得到partition的点，再算里面change的最小值

LC 410 Split array largest sum

- https://www.programering.com/a/MDOzUzMwATM.html

Feb 29 nothing

March 1 Sunday:

LC 1223 dice roll simulation (hard for me , dp)

看花花那个解释,复杂度有点高

Discuss里的一个解释是：dp[i][j] = how many choice for i dices and last number is j.

init : dp[1][*] = 1   length is 1 and value is anything is 1

dp[a][1] = sum of (dp[a-1][1~6]) (if no constraint for two consecutive 1)

LC 17 letter combination of a phone number: (backtracking)

- **LC 77 combinations**
- a.substring(i, j) is to get substring of a from index (i, j) exclude j
    - For example a.substring(0,1) is to get index 0 of string a
- a.substring(i) is to get substring of a from i to end
- Use stringBuilder ====== why ??
- String 删除某一位置的是 string.deleteCharAt(i) delete string (i) value

LC 39 combination sum

- If not use start ,but use target minus to 0, will has issues

LC 69 permutation sequence

- **LC 31 next permutation**
    - Swap function
    - Reverse function
- **Factorial number system**

March 2 Monday:

LC 40 combination sum || (有重复数字)

- **Need to sort , do backtrack**

- **Need to check if duplicated**
- **有个问题是为什么当nums[i] == nums[i-1]. 是为了防止再次从最开始top-down的时候会有重复，我之前认为的是，混淆了duplicate number 和 duplicate combination， 因为其实每次选择一个i之后，要dfs进去， 如果两次选了同样的值，就会出现duplicate combination**

Three main types of interview questions about permutations:

1. Generate all permutations
2. Generate next permutation
3. Generate the permutation number k

LC 60 permutation sequence

- 要记的关于数学的太多，还要多看看

LC 46 permutations

时间复杂度怎么算

LC 78 subsets

- 几个点关于backtrack的
  - Start要不要加
  - 要不要sort , why sort ????? （通常，要是有重复数字的时候？？？）
  - 什么时候add list 去final 答案的list
  - 什么时候remove 最后加进来的 == backtrack
  - 时间复杂度

LC 90 subset || (有重复数字)

- **Need to sort**

LC 47 permutation || (有重复数字)

- **Use a boolean value to know if this number used or not**
- **怎么判断？？？**

March 3 Tuesday:

LC 784. Letter Case Permutation **(easy but important)**

- **stringBuilder**
- **有的答案是用linkedList， 这里用arrayList也可以**

Change a char array into string:

- String.valueOf(chs)
  - Example: chs = [‘a’, ‘b’, ‘c’]
  - After that chs = ‘abc’

判断一个string里的一位是不是数字：

- Character.isDigit(S.charAt(i)) true or false
- 把某一位字母变成大， 小写： chs[i] = Character.toLowerCase(chs[i])
  - chs[i] = Character.toUpperCase(chs[i])

dfs 这里用了toCharArray

ArrayList vs LinkedList

- **Insertions are easy and fast in LinkedList as compared to ArrayList, adding in LinkedList is O(1) while in ArrayList adding is O(n)**

Queue and stack in Java

LC 943 find the shortest super string **(**hard, also dp **)**

- Assume no string in A is substring of another string in A
- Travelling salesman problem

string.startWith(xx) check prefix of string if it is start with xx

Use binary number to represent the node set:

If i = 10011(binary number), node set is {0, 1, 4}

String operations in Java

- **String and stringBuilder**
    - **String does not allow append , each method you invoke on a string creates a new object and return it since sting is immutable , can not change its internal state**
    - **stringBuilder is mutable: StringBuilder sb = new StringBuilder(); sb.append(i)**

March 4
Google mock interview

March 5
看BFS和DFS youtube的一个视频， 黄浩杰

March 5 ~ march 9: nothing
March 9

LC 743 Network delay time

- **Dfs--- done**
- **Dijkstra --- need to watch video**
- **Heap**
    - **Heap.offer**

March 10
LC 743 network delay time
collections.sort() 怎么用？？？
Can sort to ascending order，这道题这种sort就是根据第一个数字的大小

LC 996 number of squareful arrays
Backtrack， graph

Graph: Hamiltonian paths of this graph (worse than dfs)
DFS from premutation || by checking each two pair are square

LC 22 generate parentheses: (very popular)
Time complexity of DFS

March 11 wednesday
LC 301 remove invalid parentheses (dfs, bfs , hard)
Closing bracket, opening bracket , 有点难想到啊！！！！

- **LC 20 valid parentheses (easy):**
    -
- Time complexity
    -

LC 37 sudoku solver (hard, backtracking)

- Hard part is how to check 3*3 block
    - Known col and row:
    - for(int i = 0; i < 9; i++){
        - if(board[3 * (row / 3) + i / 3][3 * (col / 3) + i % 3] == target) return false
- Another part is how to loop char 1 to 9!!!
    - for(char c = '1'; c <= '9'; c++)

March 12 Thursday
LC 37
LC 51 N-queues (dfs, termination condition)

- Hard point is diagonal check
    - 从左到右（top left to bottom right） row - col 一样，就是一条对角线
    - 从右到左（bottom left to top right） row + col 一样，就是一条对角线

The main idea here is to put Q in each column from left to right, and when we put Q in each column we check the validity row by row. Since we are traversing from left to right column, we only need to check whether the current position is in conflict with its left column elements. There are only three possible positions in the left column that might be in conflict with the current Q. Respectively, they are the 135 degree, horizontally left and 45 degree ones. For 135 degree, they are in a line whose slope is -1. So (y-j)/(x-i) = -1 -> y + x = i + j. For the horizontally one, x = i. And for the 45 degree one, the line slope is 1, so (y-j)/(x-i) = 1 -> y + i = x + j

注意那个 new String(board[i])

LC 52 N-queues ||

Same question, just return the # of solutions
Two method, second way use boolean to store diagonal status and col

LC 79 word search

- Use this keyword
- Int[][] dir = new int[][]{{0,1},{0, -1}, {1,0},{ -1, 0}}
- 657. Robot Return to Origin
- LC 489 robot room cleaner

March 13 Friday, 3:19 am

LC 212. Word Search II (hard, important)

Use trieNode

LC 127 word ladder (hard for me)

Loop character for(char c = 'a'; c <='z'; c++)

- **How to prove this is the shortest one ?**

Use hashSet most of part

March 14 Saturday

LC 126 word ladder ||

March 15 Sunday:

LC 126 word Ladder || (**soooooo hard**)

March 16 Monday:

LC 752 open the lock (bfs)

Arrays.asList(deadends) change string (deadends) to a arrayList

LC 818 race car: check later

LC LRU cache

March 17 Tuesday:
LC 542. 01 Matrix
Bfs, use queue
Set direction: int[][] dict = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
Queue 的操作，还不是很熟悉， 细节上很多问题， 例如 for(int[] dir: dict){ } 忘记加int[]
March 18 nothing

March 19 thursday:
Divide and Conquer VS. Backtracking explorer recursion chapter

LC 426 Convert Binary Search Tree to Sorted Doubly Linked List
Solid line 实线 dashed line 虚线
对linkedList还有tree的转换，特别是linkedList不太熟悉

Tree search

- BFS
- DFS
    - Pre-order
    - Inorder
    - Postorder

LC 84. Largest Rectangle in Histogram (hard , Monotonic Stack **)**
Stack is little hard to understand, a trick method to save time from O(n2) to O(n) is acceptable

March 22 Sunday
Google phone interview mock
LC 925. Long Pressed Name
Use pointer
if (name.charAt(i) == typed.charAt(j) && i < m) 这一步写成这样不行，应该把 i< m 写在前面，不然在检查前一个条件的时候会出现index out of range 错误

LC 1051. Height Checker
Use counting sort ,

March 24 Tuesday:
LC 675 cut off trees of golf event: (hard)
Priority queue: how to implement this in Java
Check this answer later
https://leetcode.com/problems/cut-off-trees-for-golf-event/discuss/107415/my-python-solution-inspired-by-a-algorithm

March 28 Saturday:
Contest
LC 1395. Count Number of Teams
TreeSet()
TreeSet headSet() Method in Java 这个不好理解

还是第一个那个好理解, less array and greater array, index 0 is for left, index 1 is for right.
E.g.: [1,2,3,4,5]
For number “3” ⇒ less[0] = 2, less[1] = 0, greater[0] = 0, greater[1] = 2
So the this particular solder “3” the valid count is less[0] * greater[1] + less[1] * greater[0] = 4

LC 1396 digital underground system
Design question

March 30 Monday:
LC. 934. Shortest Bridge good question
Dfs + bfs

LC 698 Partition to K Equal Sum Subsets
If there are negative number in array
Backtracking 有点像
current_num is useful when negative number inside

LC 93. Restore IP Addresses
Definition of valid IP address:
1. The length of the ip without '.' should be equal to the length of s;
2. The digit order of ip should be same as the digit order of s;
3. Each part separated by the '.' should not start with '0' except only '0';
4. Each part separared by the '.' should not larger than 255;

Each block must satisfy: `0 <= a <= 255` and the only valid block which starts at `0` is `'0'` itself. (so `'00'`, `'01'` are all **invalid**)

Tuesday, March 31

LC 131 Palindrome Partitioning

Backtrack string.substring(i,j)

注意！！！string 里面加东西是： **tempList.add(s.substring(start, i+1));**

string.length()!!!!

Recursive method is sort of easy, **iterative is hard**

LC 241 different way to add parentheses
String to integer in Java : Integer.parseInt(“222”) parse the string ‘222’ to integer 222
也可以用 Integer.valueOf(‘222’)
我觉得这个加不加hashmap影响不大

curChar 可能等于 “+” , “-” , “*” three cases

```
switch(curChar){
                    case '+':
                        curResult = left + right;
                        break;
                    case '-':
                        curResult = left - right;
                        break;
                    case '*':
                        curResult = left * right;
                        break;
```

LC 282. Expression Add Operators (hard) 花花的视频

Target sum: 类似

Use stringBuilder , but stringBuilder is not string, convert need to use (stringBuilder) sb.toString()

s.substring()使用了 O(n) 时间， 比较慢

LC 842 Split array into fibonacci sequence (backtracking)

```
// long num = Long.parseLong(s.substring(idx, i+1));   OR
// Long num = num * 10 + (s.charAt(i) - '0');
```