### 如何做笔记

- **易错点记录**，只要不是一次过的题，就存在易错点，比如语法错啦，边界条件少啦等等。很典型的递归错误就是忘了加退出条件。这样的错误记录了两三遍之后，就再也不会错了。

- **语法记录**，很多常用的语法，比如各种数据结构的初始化比如初始化一个list of list: List<List<Integer>> lol = new LinkedList<List<Integer>>();`，再比如数组排序，以及如何使用compare函数

- **知识点记录**，给每一类题都建一个文件，推荐Dropbox的paper或者Evernote。按前面第二步的方法做的时候将题目归类并记录。记录的时候，就用一句话记录这题的题号，用的关键知识点，和关键代码，这样的好处是下次复习的时候可以快速想起，比如明天就要面试FB了，今天晚上你不可能做新题了，你就可以看你记录的这些关键点，复习效率特别高。

- **好题分析**，上一点是每题必须做，这一点是选择性的，当你做到一道题卡了很久，或者学到很多知识点（有好几个follow up），你可以写一篇文章来分析，来记录你是如何一步一步优化做出最优解的。

2023年9月10日星期日：

1. [1359] Count All Valid Pickup and Delivery Options

动态编程/递归法/数学性质分析，重点在于利用数学归纳法或者排列组合的知识分析i->i+1的过程。此外，注意题目中要求【取模】。

1. [13] Roman To Integer

易错点是注意循环结构何时退出、if和else if的区别（if语句会持续执行，而else if在if为真后就不执行）。

2023年9月19日星期二

1. [234] Palindrome Linked List

检查一个单向链接的链表是否是回文的。

默认解法很好想，复制一遍即可。递归解法比较巧妙，但是在内存占用上并不占便宜。唯一的内存o（1）解法要在原链表中将后半段反转，反转的算法非常精妙，用两个节点跟踪当前节点和前一个节点（稍后会变成后一个节点），用一个临时节点存放后一个节点的地址。此外，这道题也提供了一个单向链表寻找重点的方法：两指针赛跑，慢的一次一个节点，快的一次两个节点，最终慢的节点会在中点（左）节点停下。

1. [383] Ransom Note

检查杂志字符串里面的字母够不够组成绑票条字符串。

这道题应该使用哈希图而不是哈希表，因为当我们用字母做键出现次数做值时，每个键只对应一个值。当且仅当需要每个键对应多个值时才使用哈希表。哈希表可以创建一个，根据另外一个字符串做减法，也可以创建两个做加法再对比，区别不大。

此外，注意Java的string中有一个函数indexOf(char)可以返回char对应的索引，没有则返回-1，可以用于检测并删除某个字母是否在另外一个字符串。

还有一个思路是将两个字符串按字母顺序排列后压入栈，分别对比两个栈的字母，根据字母先后顺序判断真伪。

最后，注意一些常用方法、结构的时间、空间复杂度，如快排O(nlogn)，哈希表的插入和搜索都是O（1）等。

2023年9月21日星期四

1. [704]Binary Search

二分查找的简单实现，只是要注意在这种简单的实现中没必要使用递归，因为递归会占用大量运行时堆栈的内存，而这道题只需要简单的循环就能实现。

2023年9月23日星期六

1. [3] length Of Longest Substring

这道题是要求找出输入字符串中无重复字符的最大子字符串的长度。这道题是典型的字符串/数组题，需要用到哈希图和滑窗法缩短运行时间。创新点在于用哈希图查询某字母是否出现，并将最后一次出现重复的字符的索引作为值存在图中。答案中对于索引的更新非常有技巧： i = Math.max(hashmap.get(s.charAt(j)) + 1, i);  
先将i更新为出现重复的字符对应的索引的值+1和当前i值中的最大者，再将结果更新为当前窗口大小和当前结果中的最大值。如此一来，既规避了某个字符在该滑窗之前就出现带来的影响，也巧妙地减少了滑窗的次数，使得运行时间进一步下降。