**Week 1 Day 1**

### 1: Two Sum

Done.

**2 LinkedList Addition**

**Review I**

2-1 linkedlist相加使用dummy node， 通过三值运算符把各种情况进行综合在一个while loop中

2-2 中间记得设立变量sum 避免重复计算重复相加情况

**3 Longest Substring Without Repeating Characters:**

**Review I**

**3-**1 Sliding Window First Second 指针，HashSet可以加减Character来判断重复

3-2 for looper 用于faster 指针， slower指针在里面根据while条件更新

**Week 1 Day 2**

**5 Longest Palindromic Substring**

**Review I**

5-1 对于两种回文的可能表示形式用一种method来处理 参数为string, leftstart,rightstart

5-2 用一个for loop 对于每个i 考虑两种回文序列的可能性

5-3 为了减少出错，把max leftstart rightend 用一个数组封装存入method作为最后结果。在子函数中即将结果处理完毕

### 6. ZigZag Conversion

**Review I**

6-1 对于Matrix给定了行数，但是列数不定，自然会想到一维数组，因为是字母char所以单行使用StringBuilder

6-2 StringBuilder array 需要进行初始化 注意一下函数大小写 str.substring(3); sb.toString();

6-3 对于终止可以随时发生在for loop 或者while loop中，可以另写函数单独处理返回值

6-4 非int Array如何进行初始化：

String[] ans = new String[nRows];

Arrays.fill(ans,””);

6-5 StringBuilder array 在处理结果的时候直接可以在第一个stringbuilder上继续处理

6-6 for loop 的第二个限制条件可以有多个比如：

for(int j=0; j<numRows && cur<len; j++)

### 7 Reverse Integer

### Review II

7-1 关于越界 有两种方法： 可以在容易出错的步骤反向运算看看可不可以得到原来的结果 见solution1

或者先把可以越界的变量设为long 如果发现 > Integer.MAX\_VALUE or < Integer.MIN\_VALUE 就退出。对于正常情况最后再把long变为int，见solution2. 注意：谁设为long谁保留为int脑子要清楚。题中的原始值一直在做除法不可能出界所以不用管。

7-2 注意大小写：Integer.MAX\_VALUE Integer.MIN\_VALUE

**Week 1 Day 3**

### 8 String to int

**Review I**

8-1 Integer.parseInt(String s) parse的内容是string但本题是char。 需要用charValue-‘0’

8-2 ASCII 表 0是null， 字母是先大写再小些， 数字有0到9从小到大分布

8-3 题目让写edge case 别给自己设难题。如本题：String变数字：开头都是0的情况，开头有空格的情况，结尾有其他字符的情况。别自己给自己绕进去 比如 开头有字母的情况。

8-4 如果result开始设为long，在return的时候最后转换为int即可

对于String要考虑开始和结尾有空格的情况，空格可以是一个char: ‘ ‘

8-5 预防空格的角度 string本身有trim() 函数

### 9. Palindrome Number

### Review II

9-1 Palindrome 能通过reverse Integer去证明

9-2 对于可能过界的情况，将result先设为long然后与Integer.MAX\_VALUE进行比较。结尾快返回值时候再改回int

### 10 Regular Expression

### Review III

### 方法1：DP

10-1 两个string match的题目 很多都是动态规划dp，要去抓“到当前位置的映射关系”。从一个string的第i位到第二个string的第j位可不可以match。初始化条件是string位空所以数组的每个维度的大小会是对应string的长度加1.

10-2 boolean[][] 数组默认value是false

10-3 画二位图表解决dp动态转移方程。

方法2： DFS

10-4每两个字符为一个单元，请重新阅读本题的DFS解法

**Week 1 Day 4**

### 11 Container With Most Water

### Review I

11-1 指针如何移动：不要用arr[cur] arr[cur+1] 做判断，而是直接移动cur 查看arr[cur]和temperary变量的关系。具体参见my solution和my original solution 的异同

11-2 Official 做法 比较height[l]和height[r] 谁短挪谁

### 12 Integer To Roman

### Review I

12-1 每组十个数有规律 因为只有十个，所以可以对这个十个有规律的数进行穷举

12-2 请看official solution : intToRoman\_2.

### 13 Roman To Integer

### Review I

13-1数字string转换题 一定要按照一定的顺序，比如从高映射到低映射。。。

13-2 请看solution\_2,直接数字符的数目。那IV怎么办： 等于I1 + V5 –IV组合2

### Week 2 Day 1

### 14. Longest Common Prefix (LCP)

### Review III 全部四种解法 在solution中

14-1 做题写完while loop后马上检查是不是写了cur++， 是否能够成功退出循环条件！另外在跳出循环后cur到底在哪里 有没有在non-valid的值的后边。而且有可能内部不同的break是否需要外部cur++不同，需要酌情分析

14-3 如果返回值是String 在讨论edge case时候需要问面试官是返回null还是””

14-2 在写length()或者length的时候一定要想清楚前边的变量到底是不是数组。比如本题中strs是array所以要用strs.length 但是strs[0]是string 所以要用 strs[0].length()

14-3 因为是求最长solution所以不要把prefix的初始值设为0。为了减少时间，设为strs[0],从长往短考虑

14-4 String s2 = s1; 是deep copy 不存在改了s1然后s2会变的情况。

14-5 str.startsWith(str1) 返回是否str1是str的前缀.注意！！！starts

str.indexOf(str1) 如果str1是str部分 返回str1在str的起始位置 否则返回-1

14-6 关于String length为0的情况： String s1= "";

System.***out***.println(s1.length());

System.***out***.println(s1.substring(0, 0)); //Will NOT exception!!!

System.***out***.println(s1.substring(0, 1)); //Will go to exception

14-7 本题可以用BS 是因为要在一个String数组一堆数据中找“第一个符合要求的”位置。

14-8 本题可以用divideAndConquer是因为要在数组中找到相互的“关系”，可以从少数的共性关系发展到多数以及全局的关系

### 15 3Sum

### Review III

15-1 当if while条件中有多个大于小于符号的时候 注意括号是否对称

15-2 拿到一堆数问题 首先思考： 这些数有重复么，如果有重复 会不会出现在结果中 可不可以先去重？

这些数有顺序吗 可不可以Arrays.sort()。本题中 为了防止重复结果 要排序，排序后可以从国nums[i]==nums[i-1]来判断重复。另外特别重要的是：！！！因为排序了，可以用双指针站肩的方法遍历！从而节省时间！排序去重 一定要用双指针站肩法！！！

### 16 3 Sum Closest

16-1 最接近问题 一定要求数组有序！从而采用双指针站肩。

16-2 及时剪枝 比如target==closet立刻return

**Week 2 Day 2**

**17 Letter Combinations Of Phone Number**

17-1 StringBuilder 没有remove函数， 有deleteCharAt();

17-2 DFS写在开头的终止条件不要忘了return！！！

17-3 char转数字的方法：charVal-‘0’ or Character.getNumericValue(charVal)

17-4 official solution是如何用queue的

假设x是新加字符：

while(ans.peek().length()==i) 🡪 String t = ans.remove(); for(char s: mapping[x].toCharArray) ans.add(t+s);

### 18 Four Sum

### Review II

18-1 体会 请看我自己的答案 如何去重

while(left<right && nums[left]==nums[left+1]) left++;

while(left<right && nums[right]==nums[right-1]) right--;

left++;

right--;

**19 Remove Nth Node From Last Of LinkedList**

19-1 解决删除linkedlist问题，首元素和尾元素写完以后一定要验证！

19-2 做head++时候可以配一个index计数 这样方便加速处理结果

**20 Valid Parentheses**

20-1 Stack 是否为empty为经常需要判断(包括return前) 的一个点

**Week 2 Day 3**

**21 Merge sorted LikedList**

21-1 Recursion 和 Iteration转换的过程请记住：把while loop 换成if 再用函数代替if内的部分即可

### 22 Generate Parentheses

22-1 StringBuilder 两种去掉最后一个元素的方法：sb.deleteCharAt(sb.length()-1); ---注意这里是deleteCharAt不是removeCharAt()

sb.setLength(sb.length()-1);

### 23 Merge k sorted lists

### Review III

23-1 书写遇到s想清楚 是list 还是lists， 是nums还是num

23-2 写入queue之前一定要检验这个值是否为null

23-3 PriorityQueue需要比较的元素如果没有compare函数可以在声明PriorityQueue的时候的第二个argument写comparator (第一个argument是queue的长度)具体如下：

PriorityQueue<ListNode> pqueue = new PriorityQueue<ListNode>(lists.length,

new Comparator<ListNode>()

{

@Override

public int compare(ListNode l1, ListNode l2)

{

return l1.val – l2.valp;

}

}

);

注意 1：因为这里comparator不是写在ListNode class中 所以不可以 只有一个argument。另外这里的argument type必须是ListNode 不可以是object！！！

**24 Swap Nodes In Pairs**

24-1 Iteration解法用dummy node， recursive解法不要用dummy node

24-2 如同whileloop 不要忘了i++ 指针类题目不要忘了cur=cur.next

### 25 Reverse Nodes In K Group

### Review I:

25-1 LinkedList reverse think about recursion. Read solution LC Discussion 1

**Week 2 Day 4**

**26 Reverse Duplicates from sorted array**

26-1 Array移除元素的过程就是重新靠前写一遍的过程

**27 Remove Element**

**28. Implement strStr()**

### 29 Divide Two Integers

### Review III

29-1 变量名不要搞错 别前面是sign后边是signal

29-2 dividend为Integer.MIN\_VALUE, divisor 为 Integer.MAX\_VALUE属于特殊情况

29-3 开始处理的时候直接把int变为long， 注意value<<shift 是单位时间为1的操作 直接shift++即可traverse

long longdividend = Math.abs((long)dividend);

long longdivisor = Math.abs((long) divisor);

29.4 while(dividend>divisor)还是dividend>=divisor 想清楚

### 30 Substring with Concatenation of All Words

### Review III

30-1 所有的words（pattern）长度相同 🡪可以用str.substring(cur,cur+i); cur = cur+i; traverse. 临时变量可以用candidate变量名

30-2 HashMap用于解决Words可能会有重复的情况（如果无重复可以用hashSet）

30-3 HashMap可以remove by key: hash.remove(keyValue);

30-4 比较String不能用. 请用equals

30-5 为什会有解法二：所有words candidate字数相同。可以用traverse with constant intervals with slow fast pointer for window

**Week 3 Day 1**

**31 Next permutation:**

Review II

31-1 如何调动数字（序列）的算法伪代码写好再开始编程.三个步骤请都提前写好

### 32 Longest Parentheses

### Review III

### DP做法

32-1 我的DP解法方向没有错 但是考虑问题不全面比如 ((()))((()))

32-2 DP用在String或者数组的时候通常的一个逻辑是 “到当前点截止的结果”（当前点为ending point）!

32-3 在写状态转移方程的时候注意往前回溯的任何变量 前面的dp，是否会越界！（如坐标<0 ）一定要用if条件判断，参见本题official solution 1 DP

### Stack做法

32-4 Official Solution中提供的stack解法需要revisit，不是看的特别懂

### 左右分别扫做法

32-6 为什么扫两次！！！解决((()这种一边扫没结束但也没越界的情况

### 33 Search in a sorted array

**33-1** What is the edge case of this question: [3,1] Especially when mid==start, what to do ? >= or > is important here

### Week 3 Day 2

### 34 Search For a Range

### Review II

34-1 My solution: while(start<end-1) 那么请单独考虑 只有两个元素的所有情况（都不符合要求，左边符合要求，右边符合要求，都符合要求）

34-2 可以用<=逻辑对左边界和右边界分别搜寻，注意。搜寻后如果相当，立即更新边界。

### 35 Search Insert Position

### 36 Valid Sudoku

### Review I

36-1 Revisit 从大九宫格到小九宫格的映射

int RowIndex = 3\*(i/3);

int ColIndex = 3\*(i%3);

if(board[RowIndex + j/3][ColIndex + j%3]

**37 Sudoku Server**

**First Visit On Saturday**

**38 Count And Say**

38-1 For recursion purpose StringBuilder -> String -> s.charAt()

38-2 StringBuilder 也有charAt()函数

**Week 3 Day 3**

### 39 Combination Sum

### Review III for discussion topic A general approach to backtracking questions in java.

39-1 How to remove the last element

ArrayList: cur.remove(cur.size()-1);

String: str.substring(0,str.length()-1);

StringBuilder: sb.deleteCharAt(sb.length()-1);

39-2 变量名称一致性！ candidates[i] 还是 candidate[i]

39-3 请反复阅读本题Discuss: A general approach to backtracking questions in java. 一涉及重复就需要Array.sort(array) 并且在for loop开始时候用continue去重

**40 Combination Sum II**

**Review II**

**Solution 1 My Solution**

Why my solution is wrong!!!! Still not resolved!!! Need to Revisit

Solution 2: Official Solution : The only difference is that use remain 🡪 0 for conditions check

### 41 First Missing Positive

### Review II

41-1 Trick问题请先分析：正负数0都在数组里头，肯定先要partition，这等于是第一个小的step。然后发现如果保证连续，大于right+2的数都不用考虑（一共只有right+1个正数）。所以可以用坐标来和数-1进行对应。我的答案是自己写出来的。请注意在移动swap的时候要保证，移动左作标和右坐标不能相等，并且数值也不可以相等 以免出现死循环！！！每一个swap都需要是有效的！

**Week 3 Day 4**

**42 Trapping Rain Water**

**Review I**

42-1 copy for 函数的时候一定不要忘了把全部的i 改成j或其他字母

**43 Multiply Strings**

**Review III**

43-1 乘法要一位一位的做，所以开始String要转为charArray 🡪 int[] array better

43-2 乘法看清哪边是最低位哪边是最高位。结果最多多少位就准备多少位的数组。注意一共最多m+n位最后一位是m+n-1角标。i=m-1; j=n-1;初始状态。所以对应映射为 i+j+1为结果位数。进位为i+j

43-3 如何处理进位！！！

res[p+q+1] = product%10; //REvisit !!!Why there is no + here !!! （has been included in product）

res[p+q] += product/10; //REvisit !!!Why there is + here !!!

### 44 Wild Card Matching

### Review I

Similar to question 10

**Week 4 Day 1**

**45 Jump\_2**

**Review II**

45-1 这种题的解法称为 greedy 其实是一种DP

### 46 && 47 Permutations

### Review III

46-1 有重复的时候一定要Arrays.sort()

47-1 My solution !!! Revisit the logic!!!

**48 Rotate Image**

**Review I**

Matrix problems: Please use recursion ! n = n-2 by round

### 49 Group Anagrams

Review I

49-1 hash.containsKey() 注意K大写，contain后有s

49-2 hash.keySet()注意是set不是sets，要有括号

49-3 iter.hasNext() 注意N要大写

49-4 String可以比较是否equal 所有containsKey可以判断是否已经存在（同样的内容的），但是char[] array 不可以。

**Week 4 Day 2**

### 50 My Pow(x,n)

### Review III

50-1 return n%2==0 ? myPow(x\*x,n/2) : myPow(x\*x,n/2)\*x