目录

[1. 两数相加(leetcode第2题) 2](#_Toc9604)

[2. 无重复字符的最长子串（leetcode第3题） 3](#_Toc16461)

[3. 最长回文子串（leetcode第4题） 4](#_Toc25112)

[4. Z 字形变换（leetcode第6题） 5](#_Toc13938)

[5. 整数反转（leetcode第7题） 6](#_Toc7069)

[6. 字符串转换整数 (atoi)（leetcode第8题） 7](#_Toc22307)

[7. 盛最多水的容器（leetcode第11题） 9](#_Toc28271)

[8. 整数转罗马数字 11](#_Toc27168)

[9. 罗马数字转整数 12](#_Toc5308)

1. [两数相加](https://leetcode-cn.com/problems/add-two-numbers/)(leetcode第2题)

给出两个 非空 的链表用来表示两个非负的整数。其中，它们各自的位数是按照 逆序 的方式存储的，并且它们的每个节点只能存储 一位 数字。

如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。

您可以假设除了数字 0 之外，这两个数都不会以 0 开头。

示例：

输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)

输出：7 -> 0 -> 8

原因：342 + 465 = 807

解答

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode(int x) { val = x; }

\* }

\*/

class Solution {

public ListNode addTwoNumbers(ListNode l1, ListNode l2) {

int carry = 0;

int sum = 0;

int a,b;

ListNode p= l1;

ListNode q= l2;

ListNode result = new ListNode(0);

ListNode out=result;

while (p!=null||q!=null) {

a = (p==null)? 0 : p.val;

b = (q==null)? 0 : q.val;

sum = (a+b+carry) % 10;

carry = (a+b+carry)/10;

result.next = new ListNode(sum);

result = result.next;

if (p!=null) p=p.next;

if (q!=null) q=q.next;

}

if (carry==1) {

result.next = new ListNode(1);

}

return out.next;

}

}

此题学会掌握java中链表的用法，注意进位的表示。

1. [无重复字符的最长子串](https://leetcode-cn.com/problems/longest-substring-without-repeating-characters/)（leetcode第3题）

给定一个字符串，请你找出其中不含有重复字符的 最长子串 的长度。

示例 1:

输入: "abcabcbb"

输出: 3

解释: 因为无重复字符的最长子串是 "abc"，所以其长度为 3。

示例 2:

输入: "bbbbb"

输出: 1

解释: 因为无重复字符的最长子串是 "b"，所以其长度为 1。

示例 3:

输入: "pwwkew"

输出: 3

解释: 因为无重复字符的最长子串是 "wke"，所以其长度为 3。

  请注意，你的答案必须是 子串 的长度，"pwke" 是一个子序列，不是子串。

解：class Solution {

public int lengthOfLongestSubstring(String s) {

Map<Character, Integer> map = new HashMap<>();

int start = 0;

int end = 0;

int out = 0;

for(;end<s.length();end++) {

char ch = s.charAt(end);

if(map.containsKey(ch)) {

start = Math.max(map.get(ch),start);

}

map.put(ch, end+1);

out = Math.max(out, end-start+1);

}

return out;

}

}

此题学会hash的使用方法。Hashmap存储的是字符和字符位置。通过start和end双指针确定判定范围，。

1. [最长回文子串](https://leetcode-cn.com/problems/longest-palindromic-substring/)（leetcode第4题）

给定一个字符串 s，找到 s 中最长的回文子串。你可以假设 s 的最大长度为 1000。

示例 1：

输入: "babad"

输出: "bab"

注意: "aba" 也是一个有效答案。

示例 2：

输入: "cbbd"

输出: "bb"

class Solution {

public String longestPalindrome(String s) {

int length = s.length();

int longest=0;

String longestS = "";

boolean palindrome[][] = new boolean[length][length];

for(int len = 1;len <= length; len++) {

for(int start = 0; start<length; start++) {

int end = start+len-1;

if (end>=length)

break;

if (len==1)

palindrome[start][end] = true;

else if (len==2) {

if(s.charAt(start)==s.charAt(end))

palindrome[start][end] = true;

else

palindrome[start][end] = false;

}

else {

palindrome[start][end]=palindrome[start+1][end-1]&&(s.charAt(start)==s.charAt(end));

}

if (palindrome[start][end]) {

if (len>longest) {

longest = len;

longestS = s.substring(start,end+1);

}

}

}

}

return longestS;

}

}

是动态规划问题，想办法把用到上次的计算结果为本次使用，减少复杂度。回文数要考虑的是p(i,j)=(p(i+1,j-1)&&s(i)=s(j));有时间可以学习马拉车算法。

1. [Z 字形变换](https://leetcode-cn.com/problems/zigzag-conversion/)（leetcode第6题）

将一个给定字符串根据给定的行数，以从上往下、从左到右进行 Z 字形排列。

比如输入字符串为 "LEETCODEISHIRING" 行数为 3 时，排列如下：

L C I R

E T O E S I I G

E D H N

之后，你的输出需要从左往右逐行读取，产生出一个新的字符串，比如："LCIRETOESIIGEDHN"。

请你实现这个将字符串进行指定行数变换的函数：

string convert(string s, int numRows);

示例 1:

输入: s = "LEETCODEISHIRING", numRows = 3

输出: "LCIRETOESIIGEDHN"

示例 2:

输入: s = "LEETCODEISHIRING", numRows = 4

输出: "LDREOEIIECIHNTSG"

解释:

L D R

E O E I I

E C I H N

T S G

解：class Solution {

public String convert(String s, int numRows) {

if (numRows<2) return s;

ArrayList<StringBuilder> rows = new ArrayList<StringBuilder>();

for (int i=0; i<numRows; i++) {

rows.add(new StringBuilder());

}

int flag = -1;//express direction

int i=0;

for (char ch: s.toCharArray()) { //遍历字符串的方法

rows.get(i).append(ch);

if(i==0||i==numRows-1) {

flag=-flag;

}

i+=flag;

}

StringBuilder out = new StringBuilder();

for (StringBuilder row: rows) {

out.append(row);

}

return out.toString();

}

}

此题注意巩固ArrayList的用法，注意z字型输入的规律，即从上到下到上，flag表示放向，在最上和最下的时候变换方向。

1. [整数反转](https://leetcode-cn.com/problems/reverse-integer/)（leetcode第7题）

给出一个 32 位的有符号整数，你需要将这个整数中每位上的数字进行反转。

示例 1:

输入: 123

输出: 321

 示例 2:

输入: -123

输出: -321

示例 3:

输入: 120

输出: 21

注意:

假设我们的环境只能存储得下 32 位的有符号整数，则其数值范围为 [−2^31,  2^31 − 1]。请根据这个假设，如果反转后整数溢出那么就返回 0。

解：class Solution {

public int reverse(int x) {

int out = 0;

int int\_end=0;

while(x!=0) {

int\_end = x%10;

x = x/10;

if (out>Integer.MAX\_VALUE/10||out==Integer.MAX\_VALUE/10&&int\_end>7) return 0;

if (out<Integer.MIN\_VALUE/10||out==Integer.MIN\_VALUE/10&&int\_end<-8) return 0;

out = out\*10 + int\_end;

}

return out;

}

}

回忆数字的拆分方法，注意溢出的判断方法。

1. [字符串转换整数 (atoi)](https://leetcode-cn.com/problems/string-to-integer-atoi/)（leetcode第8题）

请你来实现一个 atoi 函数，使其能将字符串转换成整数。

首先，该函数会根据需要丢弃无用的开头空格字符，直到寻找到第一个非空格的字符为止。

当我们寻找到的第一个非空字符为正或者负号时，则将该符号与之后面尽可能多的连续数字组合起来，作为该整数的正负号；假如第一个非空字符是数字，则直接将其与之后连续的数字字符组合起来，形成整数。

该字符串除了有效的整数部分之后也可能会存在多余的字符，这些字符可以被忽略，它们对于函数不应该造成影响。

注意：假如该字符串中的第一个非空格字符不是一个有效整数字符、字符串为空或字符串仅包含空白字符时，则你的函数不需要进行转换。

在任何情况下，若函数不能进行有效的转换时，请返回 0。

说明：

假设我们的环境只能存储 32 位大小的有符号整数，那么其数值范围为 [−231,  231 − 1]。如果数值超过这个范围，请返回  INT\_MAX (231 − 1) 或 INT\_MIN (−231) 。

示例 1:

输入: "42"

输出: 42

示例 2:

输入: " -42"

输出: -42

解释: 第一个非空白字符为 '-', 它是一个负号。

  我们尽可能将负号与后面所有连续出现的数字组合起来，最后得到 -42 。

示例 3:

输入: "4193 with words"

输出: 4193

解释: 转换截止于数字 '3' ，因为它的下一个字符不为数字。

示例 4:

输入: "words and 987"

输出: 0

解释: 第一个非空字符是 'w', 但它不是数字或正、负号。

因此无法执行有效的转换。

示例 5:

输入: "-91283472332"

输出: -2147483648

解释: 数字 "-91283472332" 超过 32 位有符号整数范围。

  因此返回 INT\_MIN (−231) 。

解：class Solution {

public int myAtoi(String str) {

boolean symbol = true;

int out=0;

int i=0,j;

for(i=0;i<str.length();i++) {

if (str.charAt(i)==' ')

continue;

else if (str.charAt(i)>='0'&&str.charAt(i)<='9')

break;

else if (str.charAt(i)=='+'||str.charAt(i)=='-') {

symbol = str.charAt(i)=='+';

i++;

break;

}

else

return 0;

}

for(j=i;j<str.length();j++)

if (str.charAt(j)<'0'||str.charAt(j)>'9')

break;

for(;i<j;i++) {

if(symbol) {

if(out>Integer.MAX\_VALUE/10||out==Integer.MAX\_VALUE/10&&(str.charAt(i)-'0')>7)

return Integer.MAX\_VALUE;

out=10\*out+str.charAt(i)-'0';

}

else {

if(out<Integer.MIN\_VALUE/10||out==Integer.MIN\_VALUE/10&&(str.charAt(i)-'0')>8)

return Integer.MIN\_VALUE;

out=10\*out-(str.charAt(i)-'0');

}

}

return out;

}

}

本体思路上难度不大，要考虑好各种case的解决，字符串开头的‘+’‘-’‘ ’处理到位，同样和第7题一样，要注意溢出的判断。如果用python，可以尝试用正则表达式一句话解决，但是我不会。

1. 盛最多水的容器（leetcode第11题）

给定 n 个非负整数 a1，a2，...，an，每个数代表坐标中的一个点 (i, ai) 。在坐标内画 n 条垂直线，垂直线 i 的两个端点分别为 (i, ai) 和 (i, 0)。找出其中的两条线，使得它们与 x 轴共同构成的容器可以容纳最多的水。

说明：你不能倾斜容器，且 n 的值至少为 2。



图中垂直线代表输入数组 [1,8,6,2,5,4,8,3,7]。在此情况下，容器能够容纳水（表示为蓝色部分）的最大值为 49。

示例:

输入: [1,8,6,2,5,4,8,3,7]

输出: 49

解答：class Solution {

public int maxArea(int[] height) {

int area = 0;

int max\_area=0;

for (int i=0,j=height.length-1; i<j; ) {

area = (j-i)\*Math.min(height[i], height[j]);

max\_area = Math.max(area, max\_area);

if (height[i]<height[j])

i++;

else

j--;

}

return max\_area;

}

}

此题用双指针的方法降低时间复杂度，即i，j指针分别从两头向中间靠拢，重点是短板向长板靠拢。

1. 整数转罗马数字

罗马数字包含以下七种字符： I， V， X， L，C，D 和 M。

字符 数值

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

例如， 罗马数字 2 写做 II ，即为两个并列的 1。12 写做 XII ，即为 X + II 。 27 写做  XXVII, 即为 XX + V + II 。

通常情况下，罗马数字中小的数字在大的数字的右边。但也存在特例，例如 4 不写做 IIII，而是 IV。数字 1 在数字 5 的左边，所表示的数等于大数 5 减小数 1 得到的数值 4 。同样地，数字 9 表示为 IX。这个特殊的规则只适用于以下六种情况：

I 可以放在 V (5) 和 X (10) 的左边，来表示 4 和 9。

X 可以放在 L (50) 和 C (100) 的左边，来表示 40 和 90。

C 可以放在 D (500) 和 M (1000) 的左边，来表示 400 和 900。

给定一个整数，将其转为罗马数字。输入确保在 1 到 3999 的范围内。

示例 1:

输入: 3

输出: "III"

示例 2:

输入: 4

输出: "IV"

示例 3:

输入: 9

输出: "IX"

示例 4:

输入: 58

输出: "LVIII"

解释: L = 50, V = 5, III = 3.

示例 5:

输入: 1994

输出: "MCMXCIV"

解释: M = 1000, CM = 900, XC = 90, IV = 4.

解：class Solution {

public String intToRoman(int num) {

int[] nums = {1000,900,500,400,100,90,50,40,10,9,5,4,1};

String[] roman = {"M","CM","D","CD","C","XC","L","XL","X","IX","V","IV","I"};

StringBuilder out = new StringBuilder();

int i=0;

while(i<13) {

while(num>=nums[i]) {

out.append(roman[i]);

num-=nums[i];

}

i++;

}

return out.toString();

}

}

此题即把特殊的可以用“最大”罗马字符表示的，先表示出来，减去后继续。

1. 罗马数字转整数

罗马数字包含以下七种字符: I， V， X， L，C，D 和 M。

字符 数值

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

例如， 罗马数字 2 写做 II ，即为两个并列的 1。12 写做 XII ，即为 X + II 。 27 写做  XXVII, 即为 XX + V + II 。

通常情况下，罗马数字中小的数字在大的数字的右边。但也存在特例，例如 4 不写做 IIII，而是 IV。数字 1 在数字 5 的左边，所表示的数等于大数 5 减小数 1 得到的数值 4 。同样地，数字 9 表示为 IX。这个特殊的规则只适用于以下六种情况：

I 可以放在 V (5) 和 X (10) 的左边，来表示 4 和 9。

X 可以放在 L (50) 和 C (100) 的左边，来表示 40 和 90。

C 可以放在 D (500) 和 M (1000) 的左边，来表示 400 和 900。

给定一个罗马数字，将其转换成整数。输入确保在 1 到 3999 的范围内。

示例 1:

输入: "III"

输出: 3

示例 2:

输入: "IV"

输出: 4

示例 3:

输入: "IX"

输出: 9

示例 4:

输入: "LVIII"

输出: 58

解释: L = 50, V= 5, III = 3.

示例 5:

输入: "MCMXCIV"

输出: 1994

解释: M = 1000, CM = 900, XC = 90, IV = 4.

解答：

class Solution {

public int romanToInt(String s) {

int out = 0;

HashMap<String, Integer> roman = new HashMap<>();

roman.put("M", 1000);

roman.put("CM", 900);

roman.put("D", 500);

roman.put("CD", 400);

roman.put("C", 100);

roman.put("XC", 90);

roman.put("L", 50);

roman.put("XL", 40);

roman.put("X", 10);

roman.put("IX", 9);

roman.put("V", 5);

roman.put("IV", 4);

roman.put("I", 1);

for(int i=0; i<s.length();) {

if (i+1<s.length() && roman.containsKey(s.substring(i, i+2))) {

out += roman.get(s.substring(i, i+2));

i=i+2;

}

else {

out += roman.get(s.substring(i, i+1));

i++;

}

}

return out;

}

}

哈希表的应用，注意的是两个字符的“XX”判断优先级大于“X”。