Leetcode算法题整理

1. [**两数之和**](https://leetcode-cn.com/problems/two-sum/)

**给定一个整数数组 nums 和一个目标值 target，请你在该数组中找出和为目标值的那 两个 整数，并返回他们的数组下标。**

**你可以假设每种输入只会对应一个答案。但是，你不能重复利用这个数组中同样的元素。**

**示例:**

**给定 nums = [2, 7, 11, 15], target = 9**

**因为 nums[0] + nums[1] = 2 + 7 = 9**

**所以返回 [0, 1]**

**class Solution {  
 public int[] twoSum(int[] nums, int target) {  
 for (int i = 0; i < nums.length; i++){  
 for (int j = i + 1; j <nums.length; j++){  
 if (nums[i] + nums[j] == target){  
 return new int[]{i,j};  
 }  
 }  
  
 }  
 return new int[]{};  
 }  
}**

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

**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**

**思路：整体考虑，两个链表整体进行运算，即不转化为数字直接进行运算，满十进一，余数存位。初始化一个表头，返回表头的next。**

***/\*\*  
 \* 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) {  
 ListNode root = new ListNode(0);  
 ListNode cursor = root;  
 int carry = 0;  
 while(l1 != null || l2 != null || carry != 0) {  
 int l1Val = l1 != null ? l1.val : 0;  
 int l2Val = l2 != null ? l2.val : 0;  
 int sumVal = l1Val + l2Val + carry;  
 carry = sumVal / 10;  
 ListNode sumNode = new ListNode(sumVal % 10);  
 cursor.next = sumNode;  
 cursor = sumNode;  
 if(l1 != null) l1 = l1.next;  
 if(l2 != null) l2 = l2.next;  
 }  
 return root.next;  
 }  
}**

***/\*\*  
 \* @Author Liruilong  
 \* @Description 链表反转  
 \* @Date 16:58 2019/9/2  
 \* @Param [listNode1]  
 \* @return void  
 \*\*/*public static void linkInt(ListNode listNode1){  
 *// 链表的反转,  
// 思路一: 递归反转法,在反转当前节点之前先反转后续节点,从头节点开始  
// 思路二:遍历反转法: 从前往后反转各个节点的指针域指向.*}  
*// 递归反转*public static ListNode Reverse(ListNode listNode){  
 if ( listNode == null || listNode.next == null){  
 return listNode;  
 }  
 ListNode reNode = *Reverse*(listNode.next);  
 listNode.next.next = listNode;  
 listNode.next = null;  
 return reNode;  
}  
*// 遍历反转*public static ListNode Reverses(ListNode listNode){  
 if (listNode == null){  
 return listNode;  
 }  
 *// 上一节点* ListNode prt = listNode;  
 *// 当前节点* ListNode cur = listNode.next;  
 *// 临时节点* ListNode temp ;  
 while (cur != null){  
 temp = cur.next;  
 cur.next = prt;*// 反转指针域的指向* prt = cur; *// 指针往下移动* cur = temp;  
 }  
 listNode.next = null;  
 return prt;  
}**

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

**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。**

**public static int lengthOfLongestSubstring(String s) {  
 if (s.equals(" ")){  
 return 1;  
 }  
 *// 滑窗* LinkedList<Character> linkedList = new LinkedList<>();  
 *// 最大互异连续子串大小* int size = 0;  
 char[] chars = s.toCharArray();  
 for (int i = 0; i < chars.length; i++){  
 if (!linkedList.contains(chars[i])){  
 linkedList.addLast(chars[i]);  
 size = linkedList.size() > size ? linkedList.size() : size;  
 }else {  
 size = linkedList.size() > size ? linkedList.size() : size;  
 while (linkedList.removeFirst() != chars[i]){}  
 linkedList.addLast(chars[i]);  
 }  
 }  
 return size;  
}**

**整数反转**

**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。**

**思路：**

**通过循环将数字x的每一位拆开，在计算新值时每一步都判断是否溢出。**

**溢出条件有两个，一个是大于整数最大值MAX\_VALUE，另一个是小于整数最小值MIN\_VALUE，设当前计算结果为ans，下一位为pop。**

**从ans \* 10 + pop > MAX\_VALUE这个溢出条件来看**

**当出现 ans > MAX\_VALUE / 10 且 还有pop需要添加 时，则一定溢出**

**当出现 ans == MAX\_VALUE / 10 且 pop > 7 时，则一定溢出，7是2^31 - 1的个位数**

**从ans \* 10 + pop < MIN\_VALUE这个溢出条件来看**

**当出现 ans < MIN\_VALUE / 10 且 还有pop需要添加 时，则一定溢出**

**当出现 ans == MIN\_VALUE / 10 且 pop < -8 时，则一定溢出，8是-2^31的个位数**

**public int reverse(int x) {  
 int ans = 0;  
 while (x != 0) {  
 int pop = x % 10;  
 if (ans > Integer.*MAX\_VALUE* / 10 || (ans == Integer.*MAX\_VALUE* / 10 && pop > 7))  
 return 0;  
 if (ans < Integer.*MIN\_VALUE* / 10 || (ans == Integer.*MIN\_VALUE* / 10 && pop < -8))  
 return 0;  
 ans = ans \* 10 + pop;  
 x /= 10;  
 }  
 return ans;  
}**