# LeetCode思路

## 1:两数之和

使用map存储历史的数值，key为nums[i], value 为i

## 2:两数相加

两个链表相加，有进位，进位使用 sum/10，实际值，sum%10

### 3:无重复字符串的最长子串

<https://labuladong.gitee.io/algo/2/21/60/>

使用滑动窗口，关键点

Int left, right

Map:window, need

Int length;

While(right < s.length) {

Char = s.charAt(right);

Right++

……..

While (需要收缩) {

Char = s.charAt(left);

Left++

…….

}

}

### 4:两个正序数组中的中位数

创建一个新的数组，然后合并两个数组

判断是奇数还是偶数，如果是奇数返回 nums[length/2] 否则返回（（double）nums[length/2]+（double）nums[length/2-1] ）/ 2

### 5:最长回文子串

<https://leetcode-cn.com/problems/longest-palindromic-substring/>

1:遍历字符串每个元素

2:针对当前元素，向两边扩展（相等才扩展）

3:判断左右是否相等，如果相等，那么len+=2，left--，right++

4:判断是否超过了最大的str，然后str = s.substring(left+1, right)左闭右开

### 乘最多水的容器：

<https://leetcode-cn.com/problems/container-with-most-water/>

1:两个指针指向两边，计算面积

2:判断两个指针哪边低，低的那边往中间走。

3:直到左右指针相等

4:指针运动的时候，维护最大的面积

### 三数之和：

<https://leetcode-cn.com/problems/3sum/>

1:先对数组进行排序

2:遍历每个元素

3:判断元素是否 <0，如果大于0，肯定就没有结果了（有序数组）

4:left = i+1, right = length-1

5:判断 left和right的元素和大小，移动指针（目标值就在 left 和right之间，因为有序）

6:找到后略过 left和right相等的值

7:遍历完结尾的地方，略过 nums[i]相等的地方

### 电话号码的组合：

<https://leetcode-cn.com/problems/letter-combinations-of-a-phone-number/>

<https://leetcode-cn.com/problems/letter-combinations-of-a-phone-number/solution/dian-hua-hao-ma-de-zi-mu-zu-he-by-leetcode-solutio/>

回溯中，如果路径是字符串，那么使用StringBuffer，添加StringBuffer.append/StringBuffer.deleteCharAt

就是回溯的思路。

### 删除倒数第N个节点：

<https://leetcode-cn.com/problem-list/2cktkvj/>

注意：使用一下虚拟头节点

因为不知道链表长度，所以两个指针，一个先走N+1步，然后新节点从头走，一起走，直到老节点到底。

### 有效的括号：

<https://leetcode-cn.com/problems/valid-parentheses/>

使用栈，左括号入队列，如果是右括号，出队列

### 括号生成：

<https://leetcode-cn.com/problems/generate-parentheses/>

也就是回溯，关键就是判断是否符合括号的规则

规则，left 不能大于 n/2。Right 不能大于 left

### 合并K个升序链表

<https://leetcode-cn.com/problems/merge-k-sorted-lists/submissions/>

注意使用虚拟头节点：

1:使用优先队列，然后先把那几个头节点放进去

2:取出最小的节点，拼接到move之后，判断是否有next，丢到队列中去

### 最长括号匹配：

<https://leetcode-cn.com/problems/longest-valid-parentheses/>

还是利用括号的那个 right不能大于left的特性

1:从左往右扫描

2:计算left的，或者right的个数，判断是否是 left==right，那么判断是否是最大的

3:从左往右会漏掉一些

4:从右到左遍历，然后走2的逻辑（注意，清零的条件不同，一个是right>left， 一个是left>right）