# 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）

### 组合总数：

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

回溯法，然后注意结束条件。

关键点：去重复，使用过的数字不再使用，那么就需要进行剪枝。

### 全排列：

<https://leetcode-cn.com/problems/permutations/>

同样是回溯法

关键点：每个元素只能使用一次，那么每次往下传的时候，需要把数字扣除，后者说其他手段可以判断是否使用。

### 旋转图像：

<https://leetcode-cn.com/problems/rotate-image/>

先上下变化，再对角线变化

注意上下和对角线变化的 左边变化的函数

### 字母异位：

<https://leetcode-cn.com/problems/group-anagrams/>

对于字符串排序，再作为key存储，value存储的是 排序后key一样的 value列表

### 最大子数和：

<https://leetcode-cn.com/problems/maximum-subarray/>

遍历数组，维护一个pre，判断pre是否大于0，如果大于0，那么nums[i]的最大值为 pre+nums[i]，否则就是自己，然后再维护max

### 跳跃游戏：

<https://leetcode-cn.com/problems/jump-game/>

使用队列保存当前可以跳的节点，先把第一个节点加到队列，一直去维护队列，知道把最后一个加到队列，那么说明可以到最后一个节点。

可以使用一个标记数组，来避免元素的重复添加。

队列的含义：目前可以到的节点。

### 合并区间：

<https://leetcode-cn.com/problems/merge-intervals/>

关键点：判断有序的连着区间是否可以合并的关键点，第一个区间的 right > 下一个区间的left

1:遍历所有的区间

循环遍历可以合并的区间，一直合并

### 不同路径：

<https://leetcode-cn.com/problems/unique-paths/>

最基础的动态规划

### 最小路径和：

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

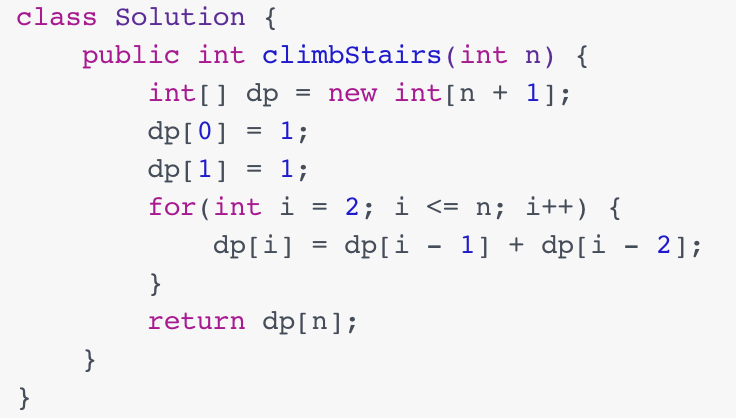
最基础的动态规划+有选择条件的赋值

### 斐波那契：

<https://leetcode-cn.com/problems/climbing-stairs/>

注意：dp[n] = dp[n-1]+dp[n-2]

注意：dp[0] = dp[1] = 1



### 颜色分类：

<https://leetcode-cn.com/problems/sort-colors/>

扫描一遍，计算数量

### 最小覆盖子串

<https://leetcode-cn.com/problems/minimum-window-substring/>

滑动窗口

Window窗口中的值

Need，需要的值

Int valid 满足的值

维护window 和valid

当valid满足（valid == need的size的时候）

维护最小的串，left右移

### 子集：

<https://leetcode-cn.com/problems/subsets/>

关键点：互不相同，那么要剪枝

回溯经典题

### 验证二叉搜索树：

<https://leetcode-cn.com/problems/validate-binary-search-tree/>

关键点：root， min，max

Root.val >=max | root.val <=min return

False，

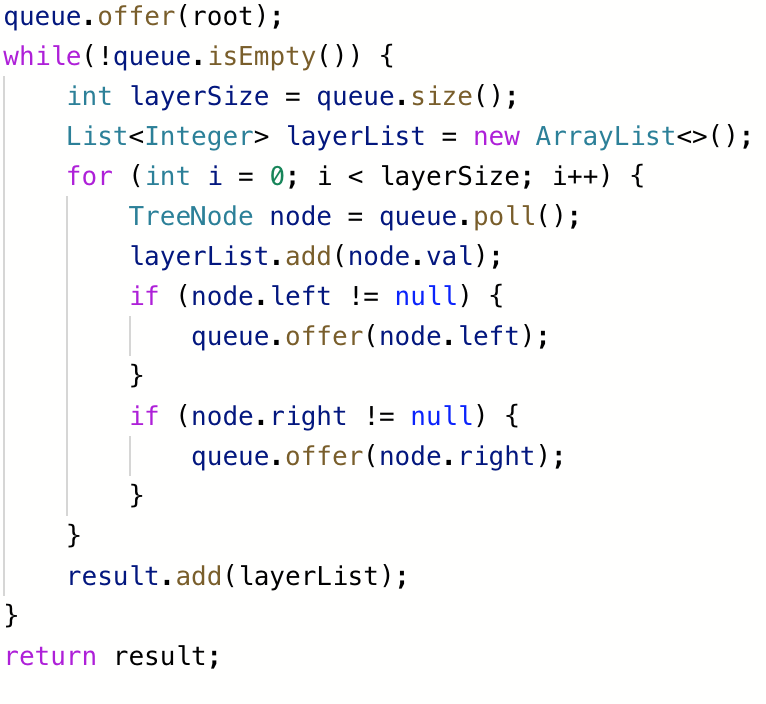
否则，针对两个子孩子来做。

### 验证对称二叉树：

<https://leetcode-cn.com/problems/symmetric-tree/>

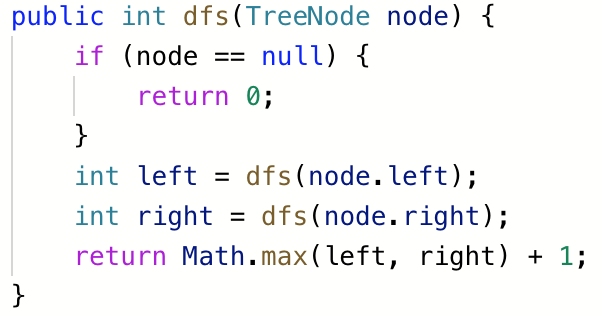
两个节点比较，比较完，比较 left.left, right.right ; left.right; right.left

### 二叉树的层序遍历：



### 二叉树的最大深度：

<https://leetcode-cn.com/problems/maximum-depth-of-binary-tree/submissions/>



### 二叉树展开为链表：

<https://leetcode-cn.com/problems/flatten-binary-tree-to-linked-list/>

使用后续遍历

### 最长连续序列：

<https://leetcode-cn.com/problems/longest-consecutive-sequence/>

使用HashSet存储，去重，然后遍历 set，如果没有比当前值 小1的，那么一直计算连续的值。如果有，那么放弃。

### 只出现一次的数字：

<https://leetcode-cn.com/problems/single-number/>

使用异或^

### 环形链表：

<https://leetcode-cn.com/problems/linked-list-cycle/>

### 环形链表2:

<https://leetcode-cn.com/problems/linked-list-cycle-ii/>

快慢指针+快指针回头

### 相交链表：

<https://leetcode-cn.com/problems/intersection-of-two-linked-lists/>

两个指针遍历，然后拼接

### 和为K的子数组：

<https://leetcode-cn.com/problems/subarray-sum-equals-k/>

1:遍历数组

2:再一个指针遍历数组，判断前缀和 相减是否满足。

### 最小公共祖先：

<https://leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-tree/>

结束条件：

1: node == null

2: node == p ｜｜ node == q

1:找左边

2:找右边

3：如果都没找到null，返回null

4：如果都找到了 != null, 返回当前

5：哪个不是null，返回哪个

### 前序创建二叉树：

