通过记录：

✓	1. 两数之和	23357	53.1%	简单
✓	3. 无重复字符的最长子串	14810	39.4%	中等
✓	7. 整数反转	7524	35.4%	中等
✓	9. 回文数	8987	55.8%	简单
✓	13. 罗马数字转整数	7268	62.3%	简单
✓	14. 最长公共前缀	7264	43.6%	简单
✓	20. 有效的括号	10389	43.9%	简单
✓	21. 合并两个有序链表	7807	66.2%	简单
✓	26. 删除有序数组中的重复项	9377	55.5%	简单
✓	27. 移除元素	9025	59.1%	简单
✓	28. 找出字符串中第一个匹配项的...	6503	43.1%	简单
✓	32. 最长有效括号	2453	37.5%	困难
✓	35. 搜索插入位置	8118	45.5%	简单
✓	50. Pow(x, n)	2252	38.1%	中等
✓	58. 最后一个单词的长度	5116	44.2%	简单
✓	66. 加一	6097	45.3%	简单

Task 1

1. 两数之和

```python
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        for i, num1 in enumerate(nums):
            for j, num2 in enumerate(nums):
                if i != j and num1 + num2 == target:
                    return [i, j]
        return []
```

思路：使用一个嵌套for循环和enumerete函数，分别遍历下标和值，根据条件返回不同的结果

9. 回文数

```python
class Solution:
    def isPalindrome(self, x: int) -> bool:
        x = str(x)
        return x == x[::-1]
```

思路：将整型转换为字符串后进行字符串的切片实现倒序

13. 罗马数字转整数

```python
class Solution:
    def romanToInt(self, s: str) -> int:
        roman = {
            'I': 1,
            'IV': 4,
            'V': 5,
            'IX': 9,
            'X': 10,
            'XL': 40,
            'L': 50,
            'XC': 90,
            'C': 100,
            'CD': 400,
            'D': 500,
            'CM': 900,
            'M': 1000
        }

        num = 0
        i = 0

        while i < len(s):
            if i + 1 < len(s) and s[i:i + 2] in roman:
                num += roman[s[i:i + 2]]
                i += 2
            else:
                num += roman[s[i]]
                i += 1

        return num
```

思路：先用字典存储各符号代表的数字，然后根据不同的键累加不同的值，其中利用字符串的切片可将两个字符看成一个整体，利用while循环确保下标不会越界

14. 最长公共前缀

```python
class Solution:
    def longestCommonPrefix(self, strs: List[str]) -> str:
        common_prefix = ""
        min_len = min(len(s) for s in strs)
        for i in range(min_len):
            current_char = strs[0][i]
            if all(s[i] == current_char for s in strs):
                common_prefix += current_char
            else:
                break

        return common_prefix
```

思路：首先用min函数找出字符串最小长度，准备遍历。然后用all函数判断得出公共前缀

20. 有效的括号

```python
class Solution:
    def isValid(self, s: str) -> bool:
        stack = []
        match = {')': '(', ']': '[', '}': '{'}

        for i in s:
            if i in match.values():
                stack.append(i)
            elif i in match.keys():
                if not stack or stack.pop() != match[i]:
                    return False

        return not stack
```

思路：创建一个字典来匹配相应的括号，利用栈来判断括号是否对应

21. 合并两个有序链表

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) -> Optional[ListNode]:
        dummy = ListNode()
        current = dummy
        while list1 and list2:
            if list1.val < list2.val:
                current.next = list1
                list1 = list1.next
            else:
                current.next = list2
                list2 = list2.next
            current = current.next
        if list1:
            current.next = list1
        elif list2:
            current.next = list2
        return dummy.next
```

思路：采用递归的方法。首先创建一个哑结点，简化链表操作。然后while循环开始递归，逐个增加结点，最后if、elif处理剩余的结点

26. 删除有序数组中的重复项

```python
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        l1 = []
        l2 = []
        for i, num in enumerate(nums):
            if num not in l1:
                l1.append(num)
            else:
                l2.append(i)
        for i in l2[::-1]:
            del nums[i]
        return len(nums)
```

思路：l1存储不重复元素，用以判断与原数组元素是否重复。l2存储重复元素的下标，用以后续删除重复元素。从后往前遍历原数组，防止改变重复元素的下标。

27. 移除元素

```python
class Solution:
    def removeElement(self, nums: List[int], val: int) -> int:
        for i in range(len(nums) - 1, -1, -1):
            if nums[i] == val:
                del nums[i]
        return len(nums)
```

思路：反向遍历数组删去目标元素，以防止改变目标元素的下标。

28. 找出字符串中第一个匹配项的下标

```python
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        i = 0
        while i <= len(haystack) - len(needle):
            if needle == haystack[i:i + len(needle)]:
                return i
            i += 1
        return -1
```

思路：利用字符串的切片找到匹配项，while循环遍历找到下标并返回

35. 搜索插入位置

```python
class Solution:
    def searchInsert(self, nums: List[int], target: int) -> int:
        if target in nums:
            return nums.index(target)
        left, right = 0, len(nums) - 1
        while left <= right:
            mid = (left + right) // 2
            if target == nums[mid]:
                return mid
            elif target < nums[mid]:
                right = mid - 1
            else:
                left = mid + 1
        return left
```

思路：由题干要求“使用时间复杂度为 O(log n) 的算法”，想到二分法。首先确定搜索范围即全部数据，然后计算中间位置并比较目标值，若不相等就更新搜索范围，重复以上过程。

58. 最后一个单词的长度

```python
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        length = 0
        s = s.strip()
        for i in s[::-1]:
            if i != " ":
                length += 1
            else:
                break
        return length
```

思路：首先s.strip()删除字符串末尾空格，然后反向遍历累加单词长度并返回

66. 加一

```python
class Solution:
    def plusOne(self, digits: List[int]) -> List[int]:
        string = ""
        l = []
        for i in digits:
            string += str(i)
        num = int(string) + 1
        for i in str(num):
            l.append(int(i))
        return l
```

思路：将数组转为字符串再转为整数后加一，然后转回数组并返回

Tsak 2

3. 无重复字符的最长子串

```python
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        string = ""
        max_length = 0
        for i in s:
            if i not in string:
                string += i
                max_length = max(max_length, len(string))
            else:
                string = string[string.index(i) + 1:] + i
        return max_length
```

思路：string用来储存无重复字符的字符串，用max_length来记录其长度。for循环遍历原字符串，用max（）得出最长子串的长度并返回。

7. 整数反转

```python
class Solution:
    def reverse(self, x: int) -> int:
        if x >= 0:
            reserve = int(str(x)[::-1])
        else:
            reserve = -int(str(-x)[::-1])
        if reserve < -2**31 or reserve > 2**31 - 1:
            return 0
        else:
            return reserve
```

思路：先将整数转换为字符串后进行切片反转，然后再将其转换为整数，最后根据条件输出对应的结果

50. Pow(x, n)

```python
class Solution:
    def myPow(self, x: float, n: int) -> float:
        return x**n
```

思路：用符号“**”求x的n次方

Task 3

32. 最长有效括号

```python
class Solution:
    def longestValidParentheses(self, s: str) -> int:
        stack = [-1]
        max_length = 0
        for i in range(len(s)):
            if s[i] == '(':
                stack.append(i)
            else:
                stack.pop()
                if not stack:
                    stack.append(i)
                else:
                    max_length = max(max_length, i - stack[-1])
        return max_length
```

思路：创建一个栈记录括号的下标，初始化为-1。用for循环遍历字符串 s，如果当前字符是左括号，将其下标 i 压入栈中；如果是右括号，弹出栈顶元素。如果栈为空，则当前右括号没有匹配的左括号，将当前右括号的下标 i 压入栈中。如果栈不为空，用“i-stack[-1]”计算当前右括号与栈顶元素之间的距离，并更新最长有效括号子串的长度 max_length。最后返回 max_length