算法分析和复杂性理论

第 1 次作业

张瀚文 2201212865

1 爬楼梯

题目描述：假设你正在爬楼梯。需要 n 阶你才能到达楼顶。每次你可以爬 1 或 2 个台阶。你有多少种不同的方法可以爬到楼顶呢？

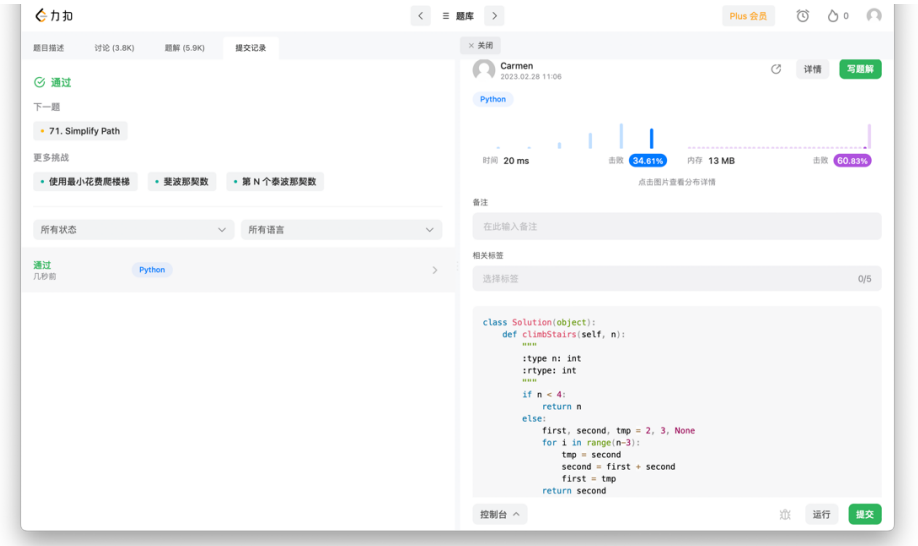
解题思路：爬第 n 阶楼梯的方法数量，等于 2 部分之和：

爬上 n−1 阶楼梯的方法数量。因为再爬 1 阶就能到第 n 阶，

爬上 n−2 阶楼梯的方法数量，因为再爬 2 阶就能到第 n 阶。

因此可以得到以下递推公式：f(n)=f(n-1)+f(n-2)，同时需要初始化。

运行结果：



测试代码：

```
class Solution(object):
    def climbStairs(self, n):
        """
        :type n: int
```

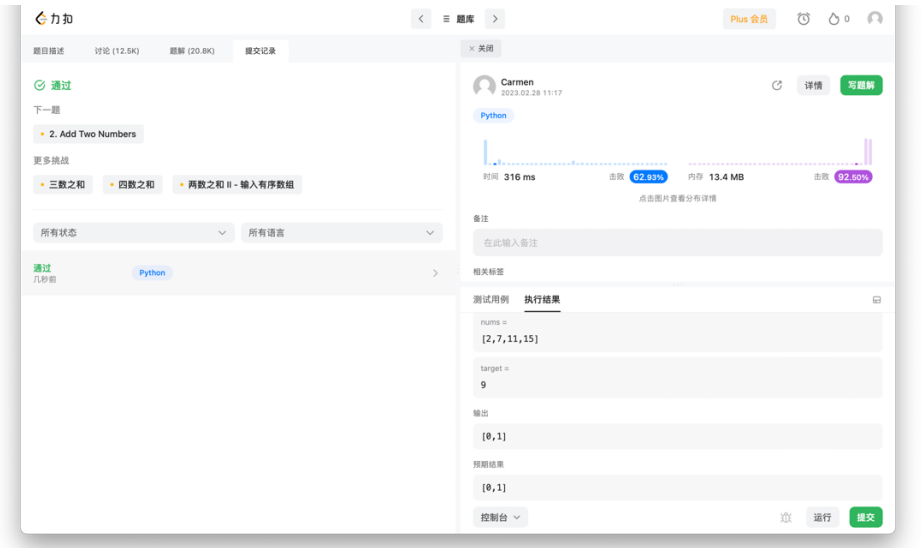
```
        :rtype: int
        """
        if n < 4:
            return n
        else:
            first, second, tmp = 2, 3, None
            for i in range(n-3):
                tmp = second
                second = first + second
                first = tmp
            return second
```

2 两数之和

题目描述：给定一个整数数组 nums 和一个整数目标值 target，请你在该数组中找出和为目标值 target 的那两个整数，并返回它们的数组下标。

解题思路：枚举数组中的每一个数 x，寻找数组中是否存在 target - x。当我们使用遍历整个数组的方式寻找 target - x 时，需要注意到每一个位于 x 之前的元素都已经和 x 匹配过，因此不需要再进行匹配。而每一个元素不能被使用两次，所以我们只需要在 x 后面的元素中寻找 target - x。

运行结果：



测试代码：

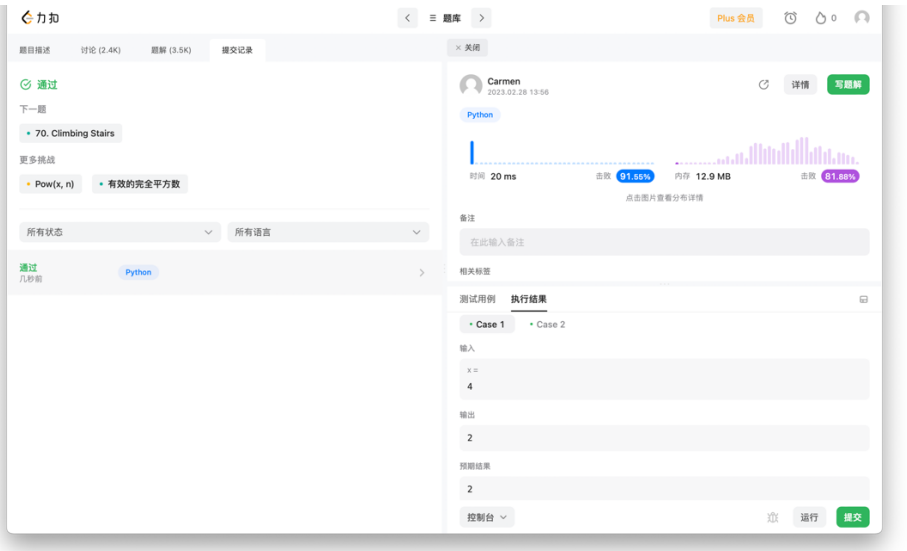
```
class Solution(object):
    def twoSum(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: List[int]
        """
        for i in range(len(nums)):
            res = target - nums[i]
            if res in nums[i+1:]:
                return [i,nums[i+1:].index(res)+i+1]
```

3　x 的平方根

题目描述：给你一个非负整数 x，计算并返回 x 的算术平方根。由于返回类型是整数，结果只保留整数部分，小数部分将被舍去。

解题思路：二分查找的下界为 0，上界可以粗略地设定为 x。在二分查找的每一步中，我们只需要比较中间元素 mid 的平方与 x 的大小关系，并通过比较的结果调整上下界的范围。

运行结果：



测试代码：

```
class Solution(object):
    def mySqrt(self, x):
```

```
"""
:type x: int
:rtype: int
"""
l, r, ans = 0, x, -1
while l <= r:
        mid = (l+r)//2
        if mid * mid <= x:
                ans = mid
                l = mid + 1
        else:
                r = mid - 1
return ans
```