算法分析和复杂性理论

第 4 次作业

张瀚文 2201212865

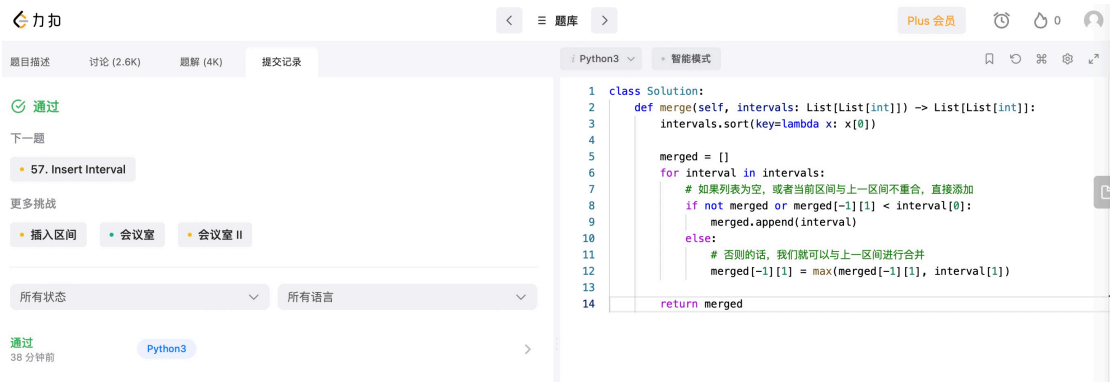
1 合并区间（056）

题目描述：以数组 intervals 表示若干个区间的集合，其中单个区间为 intervals[i] = [$start_i, end_i$] 。请你合并所有重叠的区间，并返回 一个不重叠的区间数组，该数组需恰好覆盖输入中的所有区间 。

解题思路：首先，我们将列表中的区间按照左端点升序排序。然后我们将第一个区间加入 merged 数组中，并按顺序依次考虑之后的每个区间：

- 如果当前区间的左端点在数组 merged 中最后一个区间的右端点之后，那么它们不会重合，我们可以直接将这个区间加入数组 merged 的末尾；
- 否则，它们重合，我们需要用当前区间的右端点更新数组 merged 中最后一个区间的右端点，将其置为二者的较大值。

提交记录：



测试代码：

```
class Solution:
    def merge(self, intervals: List[List[int]]) -> List[List[int]]:
        intervals.sort(key=lambda x: x[0])

        merged = []
        for interval in intervals:
            # 如果列表为空，或者当前区间与上一区间不重合，直接添加
            if not merged or merged[-1][1] < interval[0]:
                merged.append(interval)
```

```
            else:
                # 否则的话，我们就可以与上一区间进行合并
                merged[-1][1] = max(merged[-1][1], interval[1])

        return merged
```

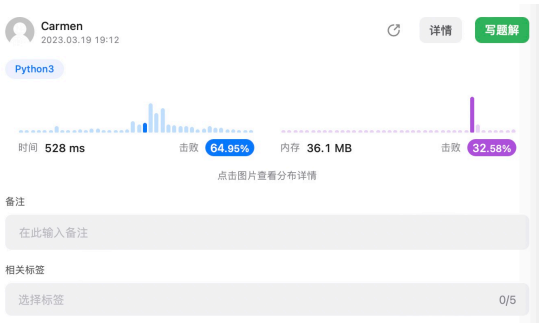
2 合并区间（148）

题目描述：给你链表的头结点 head ，请将其按升序排列并返回 排序后的链表 。

解题思路：由时间复杂度可以联想到归并排序。步骤如下：

- 通过快慢指针找到链表中点 需要确定链表的中点以进行两路归并。可以通过快慢指针的方法。快指针每次走两步，慢指针每次走一步。遍历完链表时，慢指针停留的位置就在链表的中点。
- 断链操作 split(l,n) 即切掉链表 l 的前 n 个节点，并返回后半部分的链表头。
- 合并两个有序链表(leetcode 021)
- 参考：https://leetcode.cn/problems/merge-two-sorted-lists/solutions/103891/yi-kan-jiu-hui-yi-xie-jiu-fei-xiang-jie-di-gui-by-/

提交记录：



测试代码：

```
class Solution:
    def sortList(self, head: ListNode) -> ListNode:
        def sortFunc(head: ListNode, tail: ListNode) -> ListNode:
            if not head:
                return head
            if head.next == tail:
                head.next = None
```

```
            return head
        slow = fast = head
        while fast != tail:
            slow = slow.next
            fast = fast.next
            if fast != tail:
                fast = fast.next
        mid = slow
        return merge(sortFunc(head, mid), sortFunc(mid, tail))

    def merge(head1: ListNode, head2: ListNode) -> ListNode:
        dummyHead = ListNode(0)
        temp, temp1, temp2 = dummyHead, head1, head2
        while temp1 and temp2:
            if temp1.val <= temp2.val:
                temp.next = temp1
                temp1 = temp1.next
            else:
                temp.next = temp2
                temp2 = temp2.next
            temp = temp.next
        if temp1:
            temp.next = temp1
        elif temp2:
            temp.next = temp2
        return dummyHead.next

    return sortFunc(head, None)
```

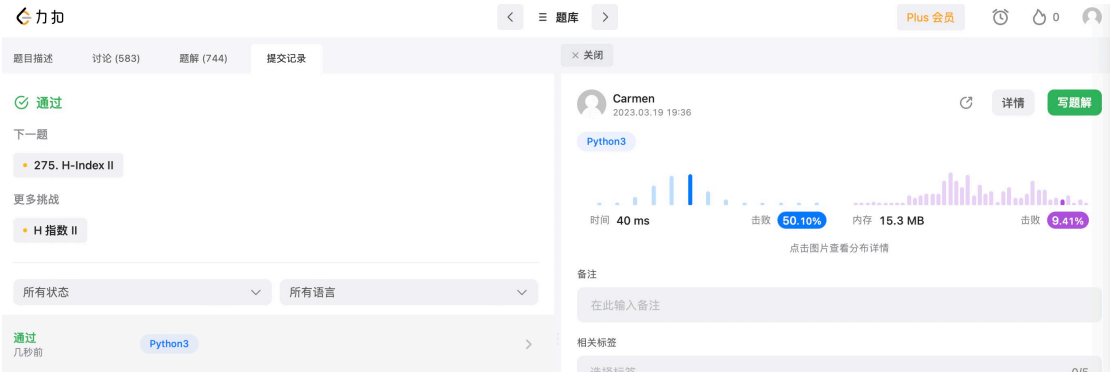
3 H 指数（274）

题目描述：给你一个整数数组 citations ，其中 citations[i] 表示研究者的第 i 篇论文被引用的次数。计算并返回该研究者的 h 指数。根据维基百科上 h 指数的定义：h 代表“高引用次数”，一名科研人员的 h 指数是指他（她）的 （n 篇论文中）总共有 h 篇论文分别被引用了至少 h 次。且其余的 n - h 篇论文每篇被引用次数不超过 h 次。如果 h 有多种可能的值，h 指数 是其中最大的那个。

解题思路：首先我们可以将初始的 H 指数 h 设为 0，然后将引用次数排序，并且对排序后的数组从大到小遍历。根据 H 指数的定义，如果当前 H 指数为 h

并且在遍历过程中找到当前值 citations[i]>h，则说明我们找到了一篇被引用了至少 h+1 次的论文，所以将现有的 h 值加 1。继续遍历直到 h 无法继续增大。最后返回 h 作为最终答案。

提交记录：



测试代码：

```
class Solution:
    def hIndex(self, citations: List[int]) -> int:
        sorted_citation = sorted(citations, reverse = True)
        h = 0; i = 0; n = len(citations)
        while i < n and sorted_citation[i] > h:
            h += 1
            i += 1
        return h
```