0023. 合并 K 个升序链表

ITCharge 大约 1 分钟

- 标签：链表、分治、堆（优先队列）、归并排序
- 难度：困难

题目链接

- 0023. 合并 K 个升序链表 - 力扣

题目大意

描述：给定一个链表数组，每个链表都已经按照升序排列。

要求：将所有链表合并到一个升序链表中，返回合并后的链表。

说明：

- $k == lists.length$。
- $0 \le k \le 10^4$。
- $0 \le lists[i].length \le 500$。
- $-10^4 \le lists[i][j] \le 10^4$。
- $lists[i]$ 按升序排列。
- $lists[i].length$ 的总和不超过 10^4。

示例：

- 示例 1：

```py
输入：lists = [[1,4,5],[1,3,4],[2,6]]
输出：[1,1,2,3,4,4,5,6]
解释：链表数组如下：
[
  1->4->5,
```

```
  1->3->4,
  2->6
]
```

将它们合并到一个有序链表中得到。

```
1->1->2->3->4->4->5->6
```

- 示例 2：

```py
输入：lists = []
输出：[]
```

解题思路

思路 1：分治算法

分而治之的思想。将链表数组不断二分，转为规模为二分之一的子问题，然后再进行归并排序。

思路 1：代码

```py
class Solution:
    def merge_sort(self, lists: List[ListNode], left: int, right: int) ->
ListNode:
        if left == right:
            return lists[left]
        mid = left + (right - left) // 2
        node_left = self.merge_sort(lists, left, mid)
        node_right = self.merge_sort(lists, mid + 1, right)
        return self.merge(node_left, node_right)

    def merge(self, a: ListNode, b: ListNode) -> ListNode:
        root = ListNode(-1)
        cur = root
        while a and b:
            if a.val < b.val:
                cur.next = a
                a = a.next
            else:
```

```
                cur.next = b
                b = b.next
            cur = cur.next
        if a:
            cur.next = a
        if b:
            cur.next = b
        return root.next

    def mergeKLists(self, lists: List[ListNode]) -> ListNode:
        if not lists:
            return None
        size = len(lists)
        return self.merge_sort(lists, 0, size - 1)
```

思路 1：复杂度分析

- **时间复杂度**：$O(k \times n \times \log_2 k)$。
- **空间复杂度**：$O(\log_2 k)$。