```
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def mergeKLists(self, lists: List[Optional[ListNode]]) ->
Optional[ListNode]:

        # 非常重要! 不需要else
        hq = [[x.val, idx] for idx, x in enumerate(lists) if x]
        heapq.heapify(hq)
        dummy = ListNode()
        node = dummy
        #lists = [x.next if x else None for x in lists ]

        while hq:
            # 1. get current minimm and update result
            cur_min, list_idx = heapq.heappop(hq)
                # append it to result
            node.next = lists[list_idx]
                # update pointer
            node = node.next

            # 2. prepare the next item to push in priority queue
            lst_nex = lists[list_idx].next

            if lst_nex is not None:
                item2push = [lst_nex.val, list_idx]
                heapq.heappush(hq, item2push)

            # 3. update linked list head
            lists[list_idx] = lists[list_idx].next

        return dummy.next

        # n*logk (n = sum(length of linkedlist in lists)```
```