42. Merge k Sorted Lists You are given an array of k linked-lists lists, each linked-list is sorted in ascending order. Merge all the linked-lists into one sorted linked-list and return it.

Example 1: Input: lists = [[1,4,5],[1,3,4],[2,6]] Output: [1,1,2,3,4,4,5,6] Explanation: The linked-lists are: [1->4->5, 1->3->4, 2->6] merging them into one sorted list: 1->1->2->3->4->4->5->6

AIM: To merger k sorted lists

PROGRAM:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next


def mergeKLists(lists):
    if not lists:
        return None

    def mergeTwoLists(l1, l2):
        dummy = ListNode(0)
        current = dummy
        while l1 and l2:
            if l1.val < l2.val:
                current.next = l1
                l1 = l1.next
            else:
                current.next = l2
                l2 = l2.next
            current = current.next
        current.next = l1 or l2
        return dummy.next

    while len(lists) > 1:
        merged_lists = []
        for i in range(0, len(lists), 2):
```

```
            if i+1 < len(lists):
                merged_lists.append(mergeTwoLists(lists[i], lists[i+1]))
            else:
                merged_lists.append(lists[i])
        lists = merged_lists

    return lists[0] if lists else None

lists = [[1,4,5],[1,3,4],[2,6]]
linked_lists = []
for lst in lists:
    dummy = ListNode(0)
    current = dummy
    for val in lst:
        current.next = ListNode(val)
        current = current.next
    linked_lists.append(dummy.next)

merged_list = mergeKLists(linked_lists)

while merged_list:
    print(merged_list.val, end=" -> ")
    merged_list = merged_list.next
print("None")
```

OUTPUT:

```
1 -> 1 -> 2 -> 3 -> 4 -> 4 -> 5 -> 6 -> None
```

TIME COMPLEXITY: O(n log k)