21. Merge Two Sorted Lists

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def mergeTwoLists(self, l1: ListNode, l2: ListNode) -> ListNode:
        dummy = ListNode(0)
        mem_dummy = dummy
        while l1 and l2:
            if l1.val < l2.val:
                dummy.next = l1
                l1 = l1.next
            else:
                dummy.next = l2
                l2 = l2.next
            dummy = dummy.next
        dummy.next = l1 or l2
        return mem_dummy.next
```

23. Merge k Sorted Lists

```
from heapq import *
class Solution:
    def mergeKLists(self, lists: List[ListNode]) -> ListNode:
        hq = []
        dummy = ListNode(0)
        dummy_mem = dummy
        for idx, node in enumerate(lists):
            if node:
                heappush(hq, (node.val, idx))
        while hq:
            _, idx = heappop(hq)
            dummy.next = lists[idx]
            if lists[idx].next:
                lists[idx] = lists[idx].next
                heappush(hq, (lists[idx].val, idx))
            dummy = dummy.next
        return dummy_mem.next
```

56. Merge Intervals

```
from heapq import *
```

```
class Solution:
    def minMeetingRooms(self, intervals: List[List[int]]) -> int:
        rooms = 0
        endings = []
        intervals.sort(key = lambda a: a[0])
        for s, e in intervals:
            if endings and endings[0] <= s:
                heappop(endings)
            else:
                rooms += 1
            heappush(endings, e)
        return rooms
```

252. Meeting Rooms
We will check whether there is any interval.

```
class Solution:
    def canAttendMeetings(self, intervals: List[List[int]]) -> bool:
        endings = []
        intervals.sort(key = lambda a: a[0])
        for idx, e in enumerate(intervals):
            if idx > 0 and e[0] < intervals[idx - 1][1]:
                return False
    return True
```

56. Merge Intervals
Nlogn + n

```
class Solution:
    def merge(self, intervals: List[List[int]]) -> List[List[int]]:
        intervals.sort()
        result = []
        for s, e in intervals:
            if result and result[-1][1] >= s:
                arr = result.pop()
                arr[1] = max(arr[1], e)
                result.append(arr)
            else:
                result.append([s, e])
        return result
```