23. Merge k Sorted Lists

Problem Statement:

1. 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
   ```

   Example 2:

   ```
   Input: lists = []
   Output: []
   ```

   Example 3:

   ```
   Input: lists = [[]]
   Output: []
   ```

Solution:

1. The idea behind this solution is to use Merge Sort which takes time complexity of O(nlogk)

2. Lets see merge sort

Practical Example:

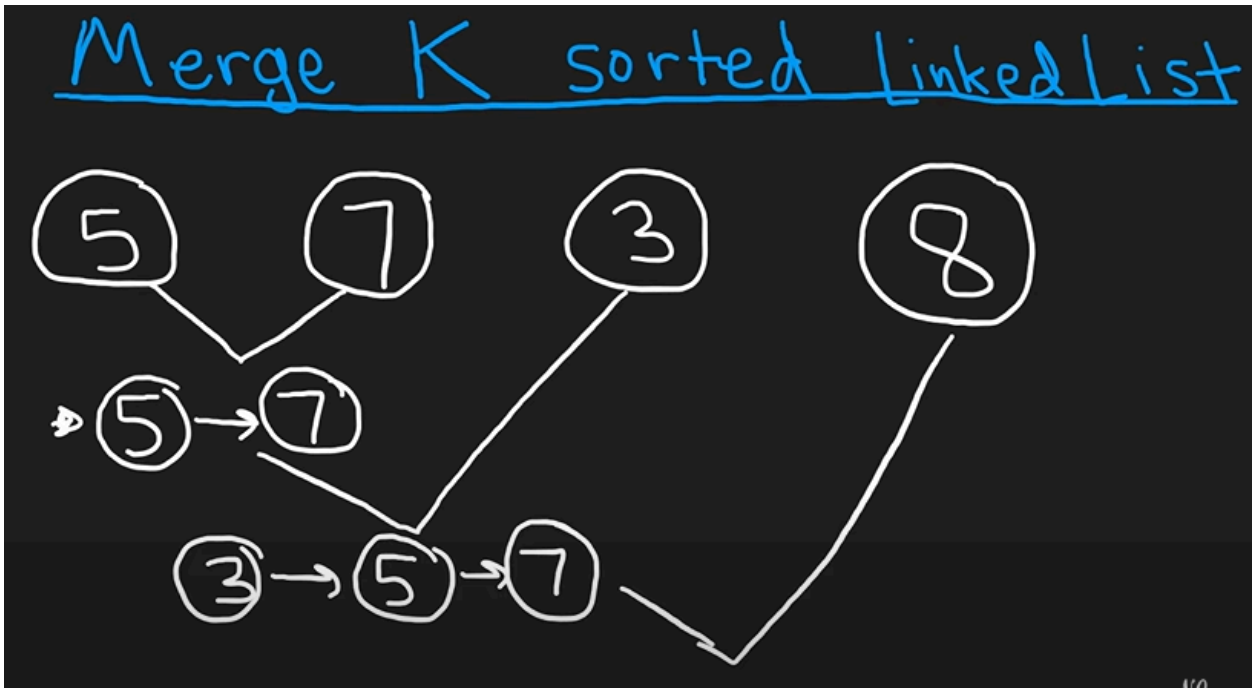
Let's sort the list of numbers: [8, 31, 25, 12] using merge sort.

1. **Divide:** Split the list into two halves: [8, 12] and [25, 31]. Now, divide each half further: [8] and [12], [25] and [31]. We are left with single-element sublists: [8], [12], [25], and [31].
2. **Conquer:** Since these sublists each have one element, they are considered sorted.
3. **Merge:**
   - Merge [8] and [12]: Since 8 is smaller, add it to the final list. Then, add 12. Now the list is [8, 12].
   - Merge [25] and [31]: Similar process, add 25 first, then 31. Now the list is [8, 12, 25, 31].

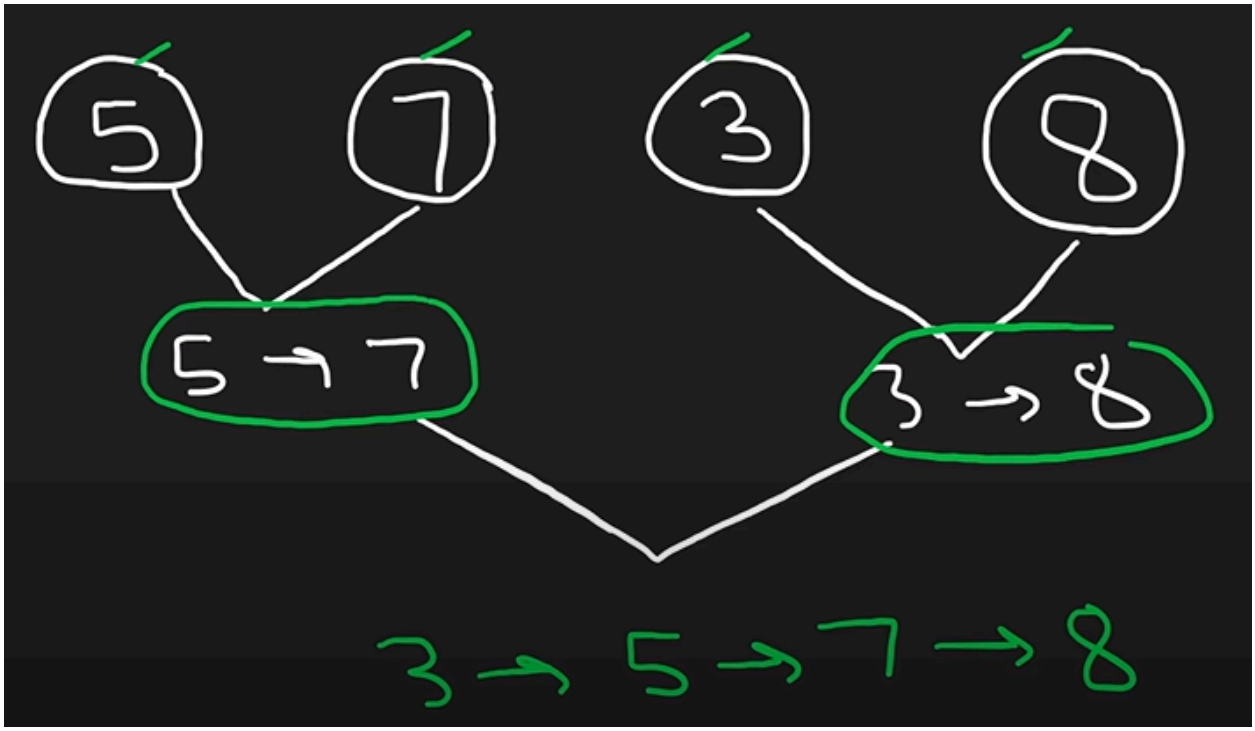
Voila! Our original list is now sorted: [8, 12, 25, 31].

3. Here lets consider we have 4 lists

4. Using the brute force…like comparing each node with remaining nodes..time complexity will be O(k.n)



5. Using merge sort we divide , compare and merge and sort



Python Code :

```
class Solution:
    def mergeKLists(self, lists: List[ListNode]) -> ListNode:
        if not lists or len(lists) == 0:
            return None

        while len(lists) > 1:
            mergedLists = []

            for i in range(0, len(lists), 2):
                l1 = lists[i]
                l2 = lists[i + 1] if (i + 1) < len(lists) else None
                mergedLists.append(self.mergeList(l1, l2))
            lists = mergedLists
        return lists[0]
```

1. First here we have dealt the edge cases
2. Now if the len(list) > 1 then we'll take one empty list and pick a pair from the original list and merge them and append it to our empty list which is mergedLists
3. Similarly we'll do that for all the pairs in our original list…until our length of lists becomes "1" means it merged all the lists
4. Finally we return lists[0]..which gives Merged K sorted lists

```
def mergeList(self, l1, l2):
    #todo
    dummy = ListNode()
    tail = dummy

    while l1 and l2:
        if l1.val < l2.val:
            tail.next = l1
            l1 = l1.next
        else:
            tail.next = l2
            l2 = l2.next
        tail = tail.next
    if l1:
        tail.next = l1
    if l2:
        tail.next = l2
    return dummy.next
```

5. And for merging two lists we used ..where we use dummy and tail pointer
6. Where we compare the values of two lists and update the tail pointer(see code and understand)