Merge K Sorted Lists(Day 24)

Prepared By Azan Imtiaz

Merge K Sorted Lists(Leatcode)

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: []

Constraints:

`k == lists.length`
$0 <= k <= 10^4$
`0 <= lists[i].length <= 500`
$-10^4 <= lists[i][j] <= 10^4$
`lists[i]` **is sorted in ascending order.**
The sum of `lists[i].length` **will not exceed** 10^4**.**

Problem Description

You are given an array of `k` **linked lists, where each linked list is sorted in ascending order. The task is to merge all these linked lists into one single sorted linked list and return it.**

Intuition to Solve

To efficiently merge multiple sorted linked lists, follow these steps:

Approach (Steps)

Initialize:

- Check if the input array `lists` is empty. If so, return null.
- Initialize `mergedList` with the first list in the array.

Iterative Merging:

- Iterate over the remaining lists in the array.
- Merge the current `mergedList` with the next list using the `mergeTwoLists` helper function.

Return Result:

- After merging all lists, return the final `mergedList`.

Merge Two Lists:

- Use a dummy node to simplify the merging process.
- Traverse both lists, appending the smaller node to the result list until one of the lists is exhausted.
- Append any remaining nodes from the non-exhausted list.

Code

```
class ListNode {
    int val;
    ListNode next;
    ListNode() {}
    ListNode(int val) { this.val = val; }
    ListNode(int val, ListNode next) { this.val = val; this.next = next; }
}

class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        if (lists == null || lists.length == 0) {
            return null;
        }

        // Start with the first list
        ListNode mergedList = lists[0];

        // Merge the remaining lists one by one
        for (int i = 1; i < lists.length; i++) {
            mergedList = mergeTwoLists(mergedList, lists[i]);
        }

        return mergedList;
    }

    // Helper function to merge two sorted linked lists
    private ListNode mergeTwoLists(ListNode l1, ListNode l2) {
        ListNode dummy = new ListNode(0);
        ListNode current = dummy;

        while (l1 != null && l2 != null) {
            if (l1.val < l2.val) {
                current.next = l1;
                l1 = l1.next;
```

```
            } else {
                current.next = l2;
                l2 = l2.next;
            }
            current = current.next;
        }

        // If there are remaining nodes in either l1 or l2
        if (l1 != null) {
            current.next = l1;
        } else {
            current.next = l2;
        }

        return dummy.next;
    }
}
```

Code Description

- **MergeKLists Method:**
  - Initializes `mergedList` with the first list.
  - Iterates over the remaining lists and merges each one with the current `mergedList` using the `mergeTwoLists` function.
- **MergeTwoLists Method:**
  - Uses a dummy node to facilitate the merging process.
  - Iterates through both input lists (`l1` and `l2`), comparing their values and appending the smaller node to the result list.
  - After one list is exhausted, appends the remaining nodes from the other list to the result list.

Time Complexity

- **Merging Two Lists:** For two lists with total nodes `n`, merging takes `O(n)` time.
- **Iterative Merging:** Merging `k` lists involves `k-1` merge operations. Each operation involves handling all nodes, leading to a time complexity of `O(N×k)`, where `N` is the total number of nodes across all lists.

Space Complexity

- **Space Complexity:** `O(1)` additional space, excluding the space required for input and output linked lists. Only a few pointers are used, and no extra data structures are needed.

Dry Runs

Test Case 1

- Input: lists = [[1,4,5],[1,3,4],[2,6]]
- Initial Merge: Merge [1,4,5] with [1,3,4]:
  - Result: [1,1,3,4,4,5].
- Final Merge: Merge [1,1,3,4,4,5] with [2,6]:
  - Result: [1,1,2,3,4,4,5,6].
- Output: [1,1,2,3,4,4,5,6]

Test Case 2

- Input: lists = []
- No lists to process, so return null.
- Output: []

Test Case 3

- Input: lists = [[]]
- The single list is empty, so return null.
- Output: []