Merge K Sorted Linked List [LeetCode](https://leetcode.com/problems/merge-k-sorted-lists/description/)

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]

**Approach 1: Merge K sorted linked lists using a brute force approach**

* **Functionality:**
  + Merges k sorted linked lists using a brute force approach.
* **Explanation:**
  + Flatten all linked lists into a single vector.
  + Sort the values in the vector.
  + Create a new linked list from the sorted values.
* **Time Complexity:**
  + **mergeKListsBruteForce**:
    - **Time complexity for flattening and sorting: O(N log N), where N is the total number of elements in all linked lists.**
    - **Time complexity for creating a new linked list: O(N).**
    - **Overall Time Complexity: O(N log N).**
* **Space Complexity:**
  + **O(N), where N is the total number of elements in all linked lists.**

**Approach 2: Merge K sorted linked lists using a min-heap approach**

* **Functionality:**
  + Merges k sorted linked lists using a min-heap approach.
* **Explanation:**
  + Create a min-heap using **Node** objects representing linked list nodes.
  + Add the head nodes of each list to the min-heap.
  + Merge the lists by repeatedly taking the smallest node from the heap.
* **Time Complexity:**
  + **mergeKListsUsingHeap**:
    - **Time complexity for initializing the min-heap: O(K log K), where K is the number of linked lists.**
    - **Each insertion and extraction from the heap: O(N log K), where N is the total number of elements in all linked lists.**
    - **Overall Time Complexity: O(N log K).**
* **Space Complexity:**
  + **O(K), where K is the number of linked lists.**

**Approach 3: Merge K sorted linked lists using a divide and conquer approach**

* **Functionality:**
  + Merges two sorted linked lists, recursively merges halves of k linked lists, and merges k sorted linked lists using divide and conquer.
* **Explanation:**
  + Utilizes **mergeTwoSortedList** to merge two sorted linked lists.
  + Recursively merges left and right halves using **mergeKListHelper**.
  + Merges the two resulting halves.
* **Time Complexity:**
  + **mergeTwoSortedList**: O(M + N), where M and N are the sizes of the two linked lists being merged.
  + **mergeKListHelper**:
    - **Time complexity for each level of recursion: O(N).**
    - **Number of levels in the recursion: O(log K).**
    - **Overall Time Complexity: O(N log K), where N is the total number of elements in all linked lists, and K is the number of linked lists.**
  + **mergeKListsDivideAndConquer: O(N log K).**
* **Space Complexity:**
  + **O(N), where N is the total number of elements in all linked lists.**

**Conclusion**

* **Brute Force:** Simple, but less efficient due to sorting.
* **Min Heap:** Efficient for large datasets, particularly when K is significantly smaller than the total number of elements.
* **Divide and Conquer:** Balances efficiency and simplicity, suitable for various scenarios.