Solution: Merge K Sorted Lists

Let's solve the Merge K Sorted Lists problem with the K-way Merge pattern.

We'll cover the following

Statement

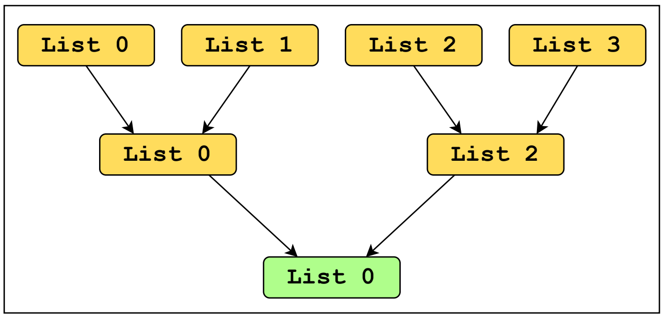
Given an array of k sorted linked lists, your task is to merge them into a single sorted list.

Constraints:

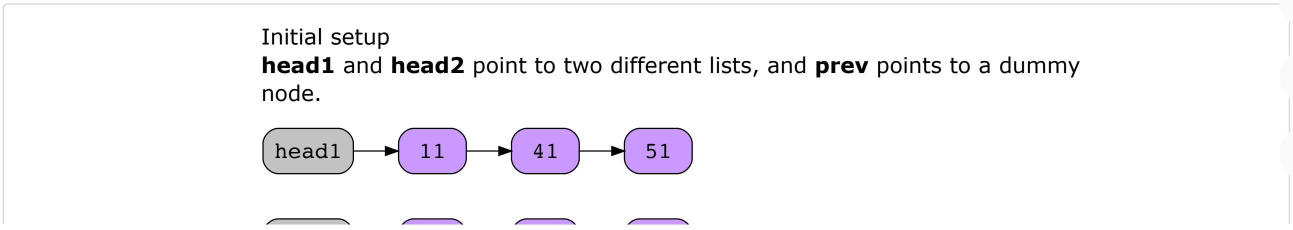
- $k =$ `lists.length`
- $0 \leq k \leq 10^3$
- $0 \leq$ `lists[i].length` ≤ 500
- $-10^3 \leq$ `lists[i][j]` $\leq 10^3$
- Each `lists[i]` is sorted in ascending order.
- The sum of all `lists[i].length` will not exceed 10^3.

Solution

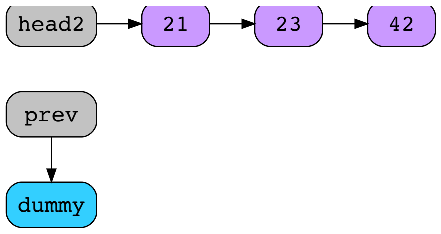
Since our task involves multiple lists, we can use the divide and conquer technique, starting with pairing the lists and then merging each pair. We repeat this until all the given lists are merged. This way, after the first pairing, we're left with $\frac{k}{2}$ lists, then $\frac{k}{4}$, $\frac{k}{8}$ and so on.



We have access to two pointers, each pointing to a different list. Let's call them head1 for the first list and head2 for the second list. If the node value of head1 is less than or equal to the node value of head2, add the head1 node to the new merged list. Otherwise, add the head2 node.



Initial setup
head1 and **head2** point to two different lists, and **prev** points to a dummy node.



1 of 8

Let’s look at the code for the solution below:

Java

main.java

LinkedListNode.java

LinkedList.java

PrintList.java

```java
class MergeSortList {
    // helper function
    public static LinkedListNode merge2Lists(LinkedListNode head1, LinkedList
        LinkedListNode dummy = new LinkedListNode(-1);
        LinkedListNode prev = dummy; // set prev pointer to dummy node
        // traverse over the lists until both or one of them becomes null
        while (head1 != null && head2 != null) {
            // if l1 value is<=  l2 value, add l1 node to the list
            if (head1.data<= head2.data) {
```

```java
            } else {
                // if l1 value is greater than l2 value, add l2 node to the l
                prev.next = head2;
                head2 = head2.next;
            }
            prev = prev.next;
        }

        if (head1 == null)
            prev.next = head2;
        else
            prev.next = head1;

        return dummy.next;
    }

    // Main function
```

Merge K sorted lists

Time complexity

The time complexity is $O(n \log k)$, where k is the number of the lists and n is the maximum length of a single list.

Space complexity

The space complexity is $O(1)$, since constant space was utilized.

Back

Merge K Sorted Lists

Next

Kth Smallest Element ...

✓ Mark as Completed