23. Merge k Sorted Lists

Hard 18.4K 660

Companies

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:

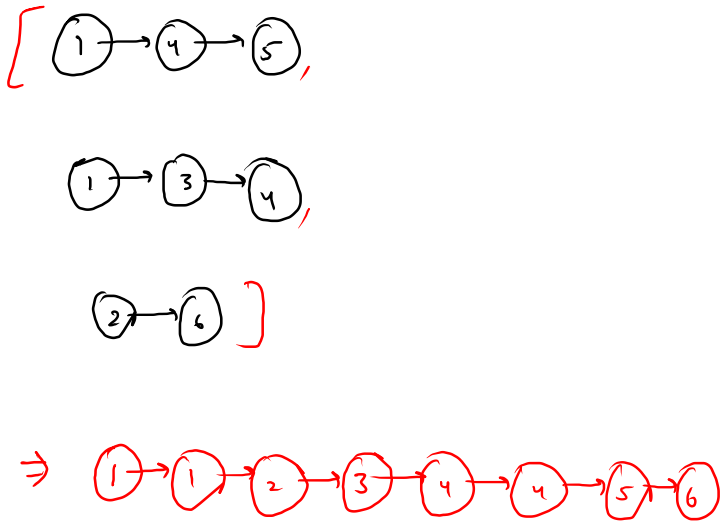
```
Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6
```

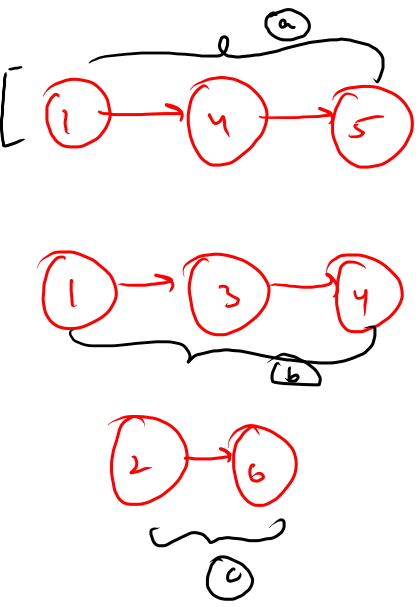
Example 2:

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

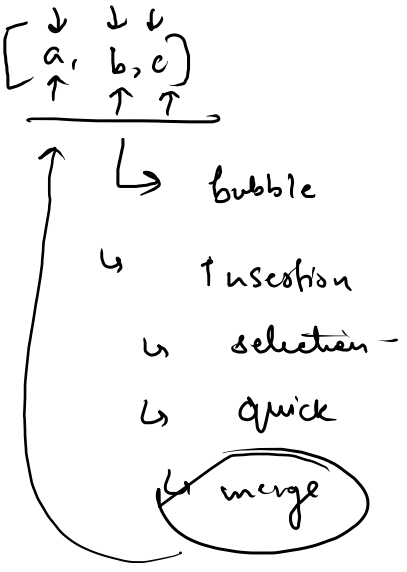
Example 3:

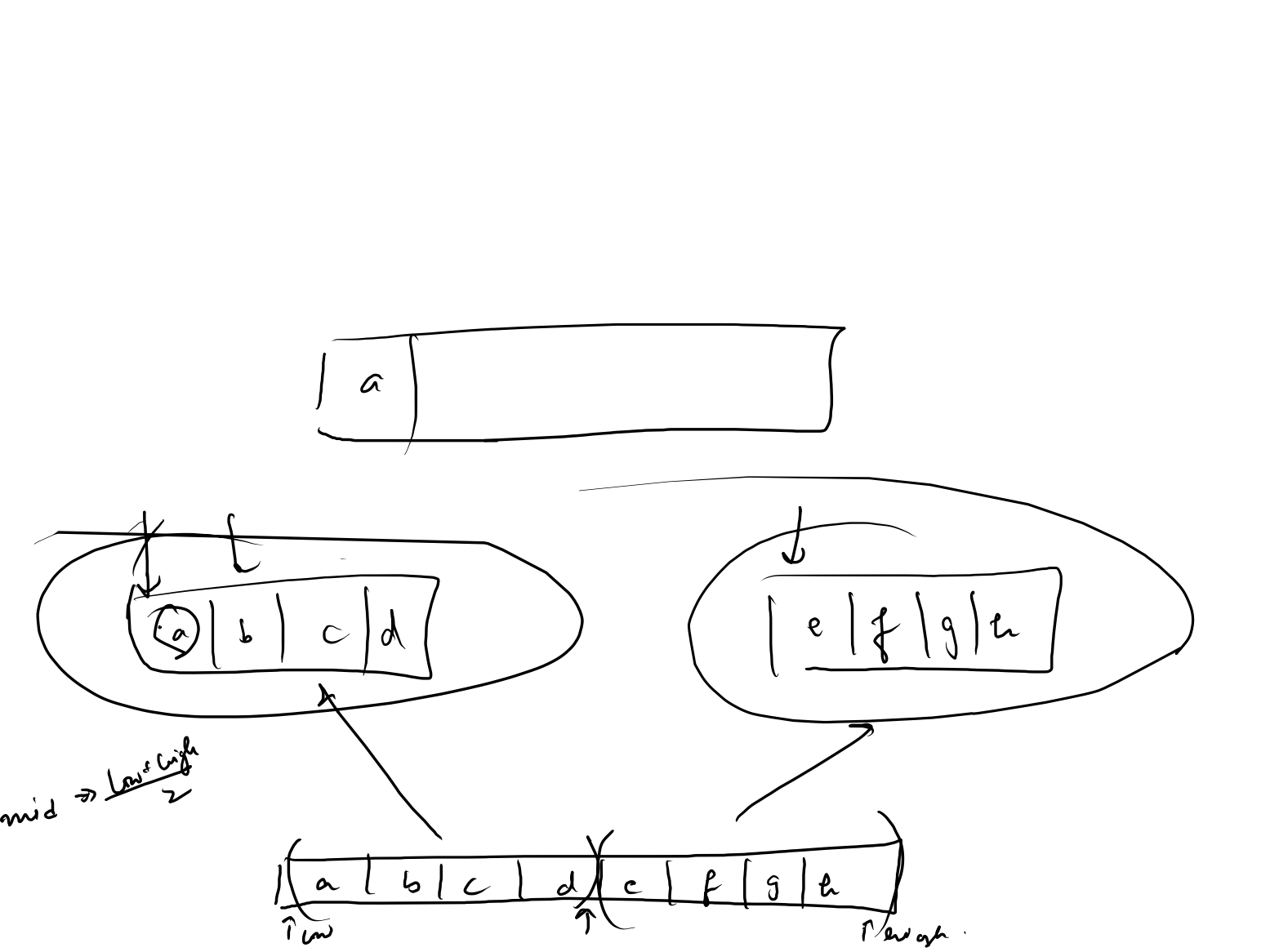
```
Input: lists = [[]]
Output: []
```

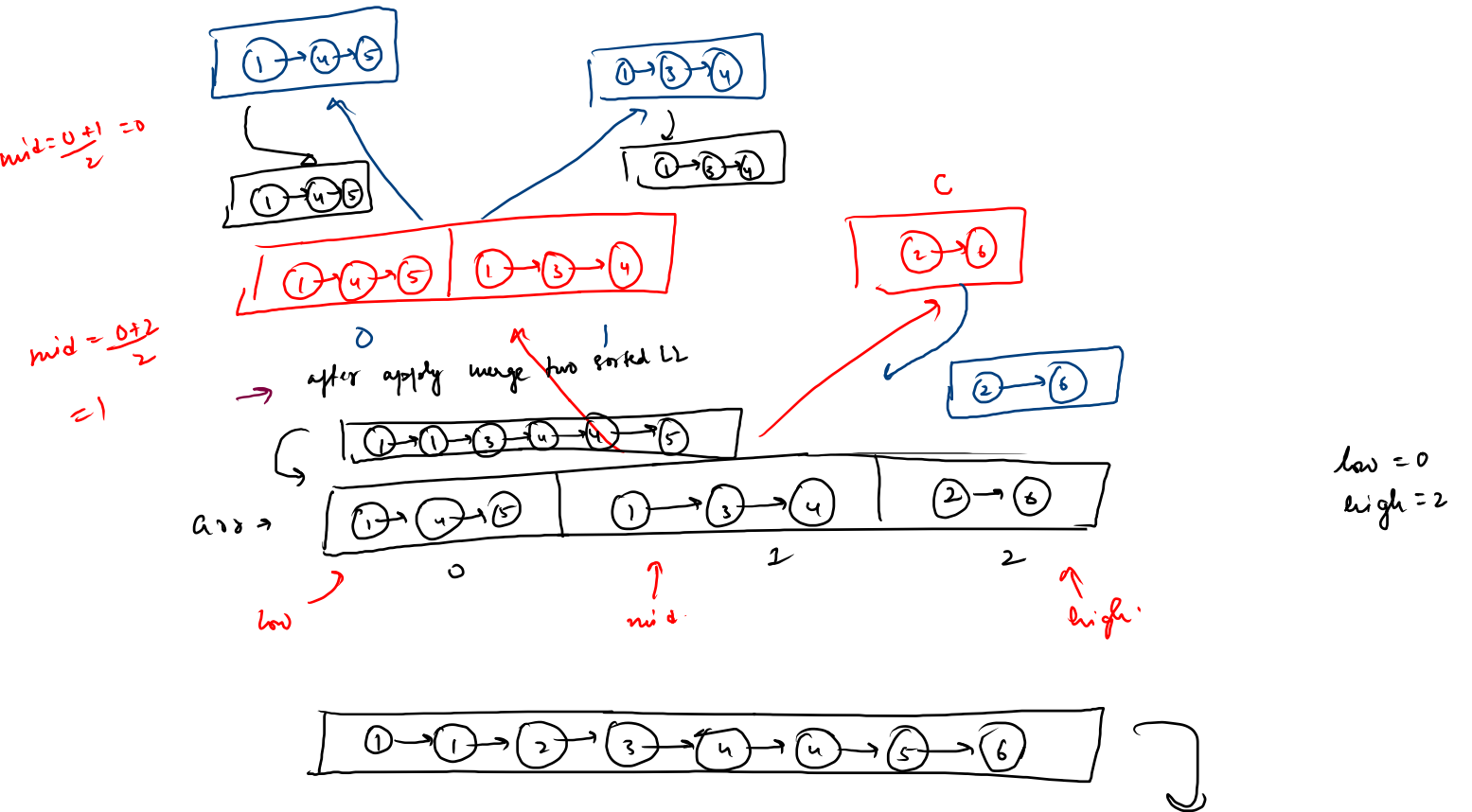


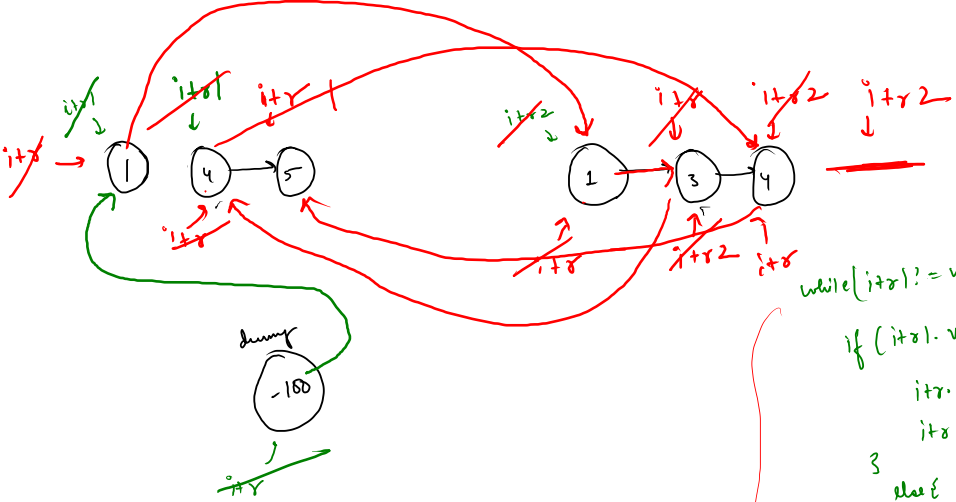

a
1
4
5
1
3
4
b
2
6
c

⇒


a, b, c
bubble
insertion
selection
quick
merge


a
a b c d
e f g h
mid ⇒ low + high / 2
a b c d e f g h
low
high

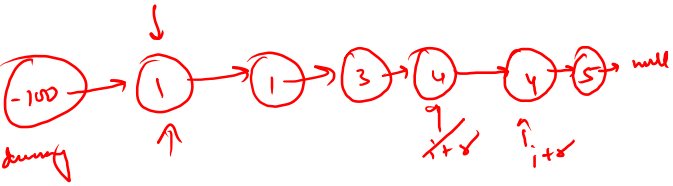

mid = 0+1/2 = 0
mid = 0+2/2 = 1
0
1
C
after apply merge two sorted LL
low = 0
high = 2
arr →
low
mid
high
0
1
2




```
while(itr1 != null && itr2 != null) {
    if (itr1.val <= itr2.val) {
        itr.next = itr1
        itr1 = itr1.next;
    }
    else {
        itr.next = itr2;
        itr2 = itr2.next;
    }
    itr = itr.next;
}
```

itr.next = (itr1 != null)? itr1 : itr2;

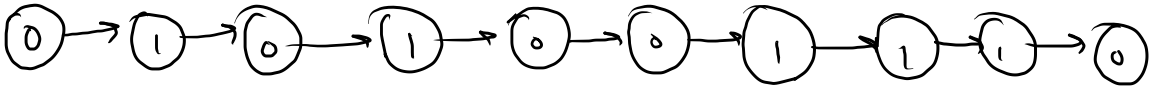
return dummy.next;



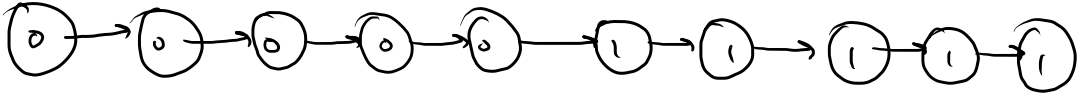
```java
public ListNode mergeKLists(ListNode[] lists, int low, int high) {
    if (low == high) {
        return lists[low];
    }

    int mid = (low+high)/2;

    ListNode firstHalf = mergeKLists(lists, low,mid);
    ListNode secondHalf = mergeKLists(lists,mid+1,high);

    return mergeTwoSortedLists(firstHalf,secondHalf);
}


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

    int low = 0, high = lists.length-1;
    return mergeKLists(lists,low,high);
}
```

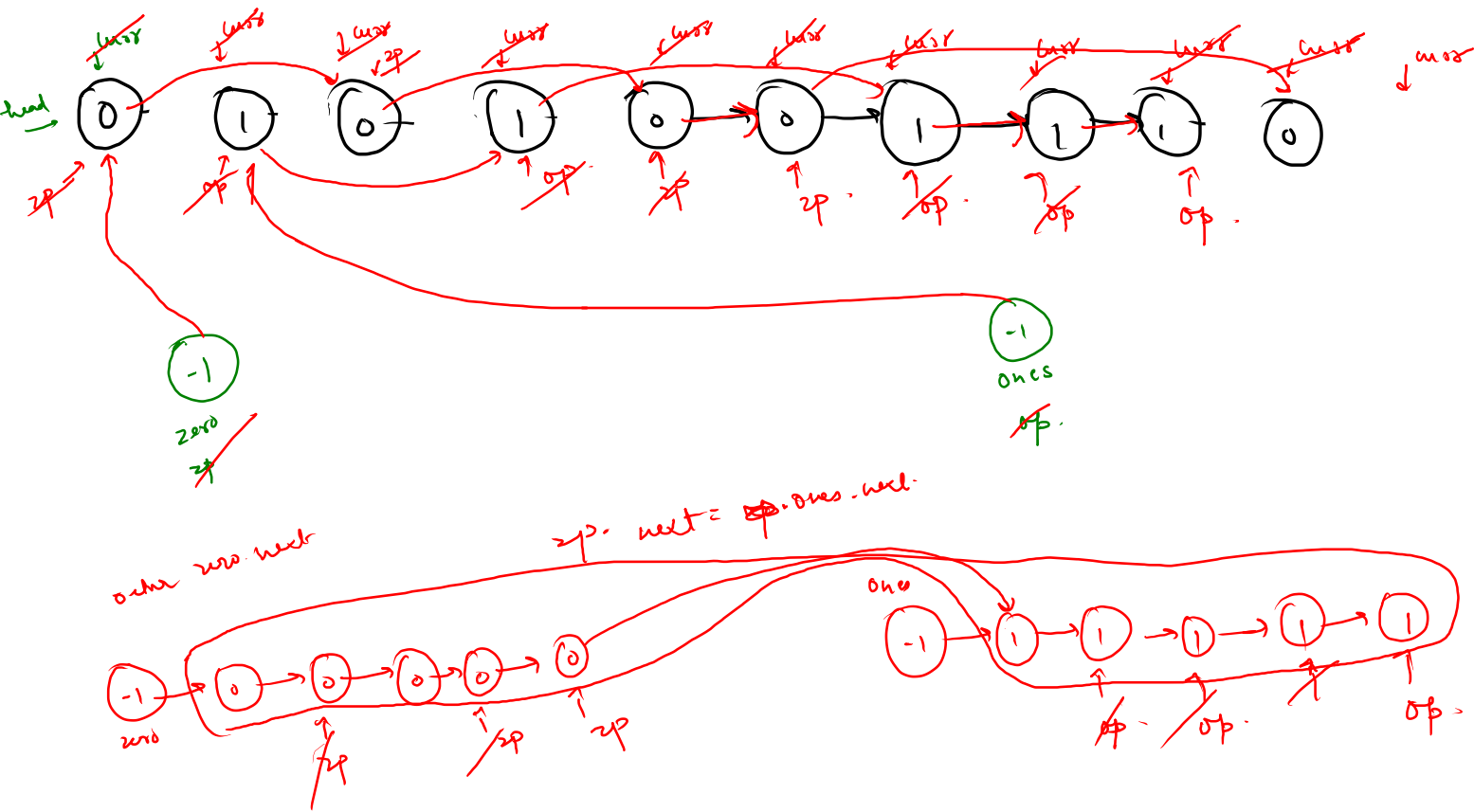
```java
public ListNode mergeTwoSortedLists(ListNode a, ListNode b) {
    if(a == null || b== null) {
        return a!= null? a:b;
    }

    ListNode itr1 = a;
    ListNode itr2 = b;

    ListNode dummyNode = new ListNode(-1000);
    ListNode itr = dummyNode;

    while(itr1 != null && itr2 != null) {
        if (itr1.val <= itr2.val) {
            itr.next = itr1;
            itr1 = itr1.next;
        } else {
            itr.next = itr2;
            itr2 = itr2.next;
        }

        itr = itr.next;
    }

    itr.next = (itr1 != null)? itr1 : itr2;

    return dummyNode.next;
}
```

I.P.



O.P.



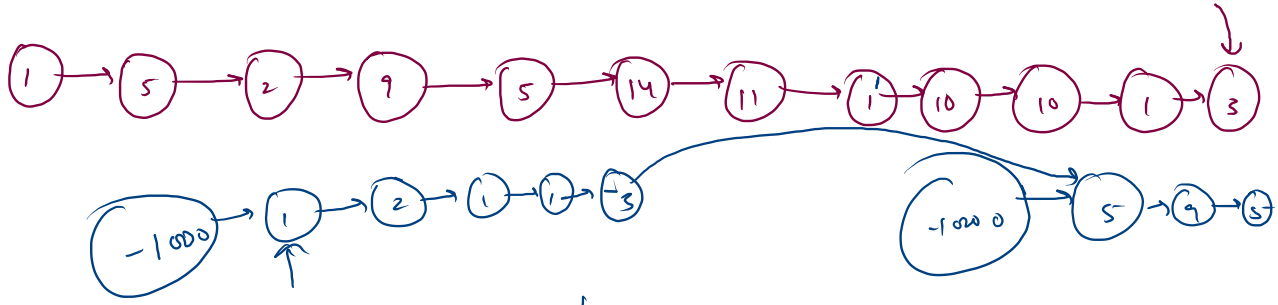

head
0
1
0
1
0
0
1
1
1
0
-1
zeros
-1
ones
curr
zp
op

other zero next
zp.next = op.ones.next
-1
zero
one
zp
op

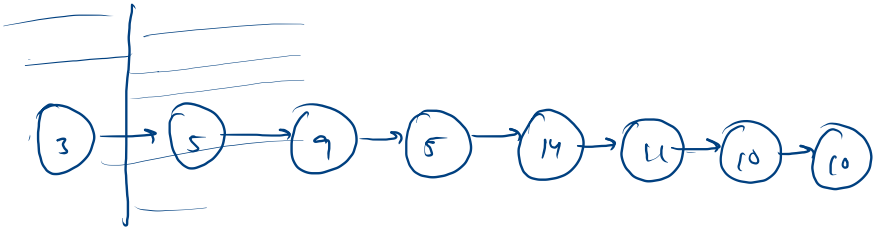
```java
public static ListNode segregate01(ListNode head) {
    if(head == null || head.next == null) {
        return head;
    }

    ListNode zeros = new ListNode(-1000);
    ListNode zp = zeros;

    ListNode ones = new ListNode(-1000);
    ListNode op = ones;


    ListNode curr = head;

    while(curr != null) {
        if(curr.val == 0) {
            zp.next = curr;
            zp = zp.next;
        } else {
            op.next = curr;
            op = op.next;
        }
        curr = curr.next;
    }

    zp.next = ones.next;

    return zeros.next;
}
```

12 1 5 2 9 5 14 11 1 10 10 1 3

val = 3



O P



```java
public static ListNode segregateOnLastIndex(ListNode head) {
    if (head == null || head.next == null) {
        return head;
    }

    ListNode curr = head;
    ListNode temp = head;

    while(temp.next != null) {
        temp = temp.next;
    }

    int pivot = temp.val;

    ListNode larger = new ListNode(pivot);
    ListNode lp = larger;

    while(curr != null) {
        if(curr.val > pivot) {
            lp.next = curr;
            lp = lp.next;
        }

        curr = curr.next;
    }

    lp.next = null;
    return larger;
}
```

Saturday timmings:
8:30 pm to 11:30 pm