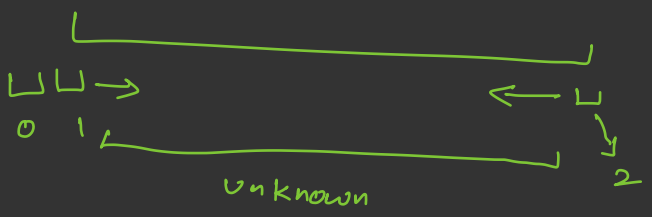
Sort → O(nlogn)

0, i, j, k



i = 0
j = 0
k = n-1

0 → (i-1) ⇒ 0's
i → (j-1) ⇒ 1's
j → k ⇒ unknown's
k+1 → n-1 ⇒ 2's

i
↓
1, 2, 1, 1, 0, 1, 2, 1, 2, 0, 1
↑ i ↑ k

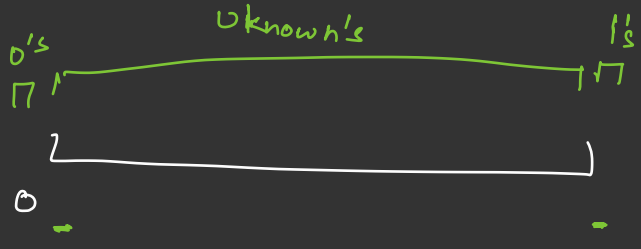
Loop

```
if (arr[j] == 0)
    swap(arr, i, j)
    i++
    j++
```

0 0 0 1 1 0 1 2 1 0 2 1 2 2 2
↑ 0 ↑ i ↑ j ↑ k

```
if (arr[j] == 1)
    j++
```

```
if(arr[i] == 2)
    swap(arr, j, k);
    k--;
```



$0 \to (i-1) \Rightarrow 0$'s

$i \to j \Rightarrow$ unknown

$j+1 \to n \Rightarrow 1$'s

$i = 0$

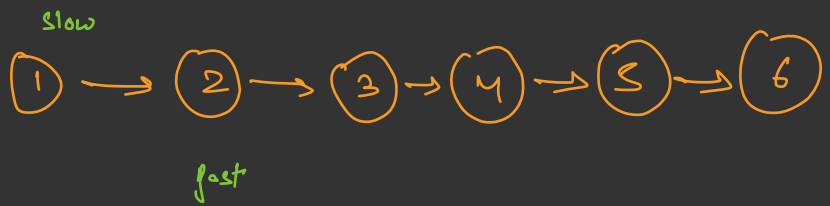
$j = n-1$

```
if( arr[i] == 0)
    i++;
else {
    swap(arr, i, j)
    j--
```

Ques Mid of linked list

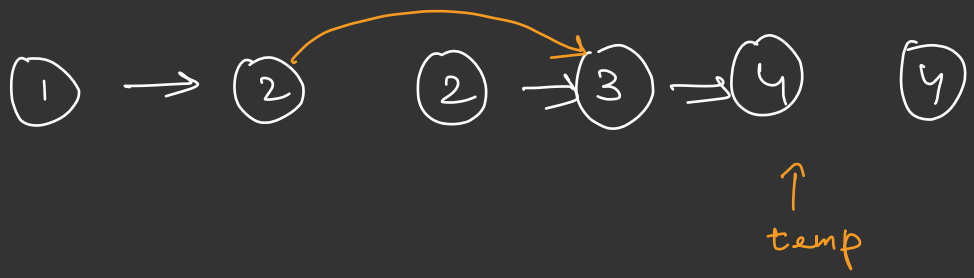
① Size is given → go till size/2
⇒ getNode((size/2) - 1)

② Slow Fast



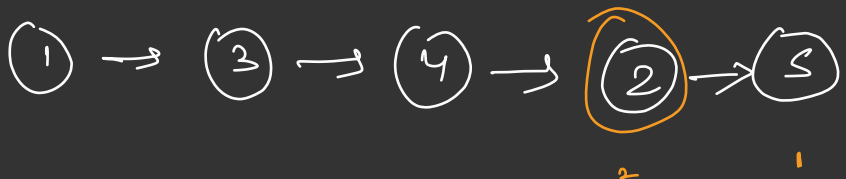

```
while(fast != null && fast.next != null){
    slow = slow.next
    fast = fast.next.next
}
return slow;
```

Ques Remove duplicates in Sorted LL



Ques kth node from end

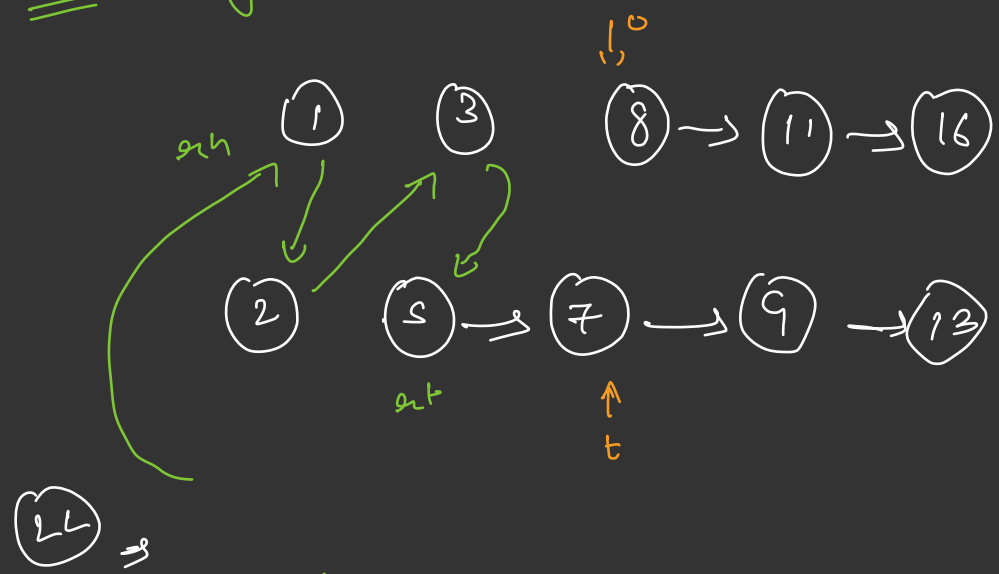
k = 2



2 1

getNode (Size - K + 1)

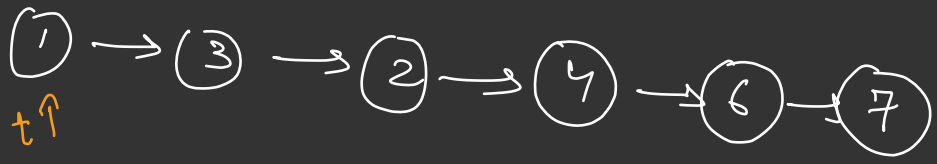
Ques Merge 2 sorted LL

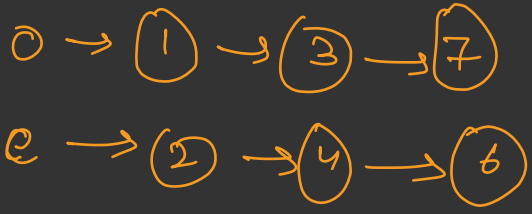


comparing

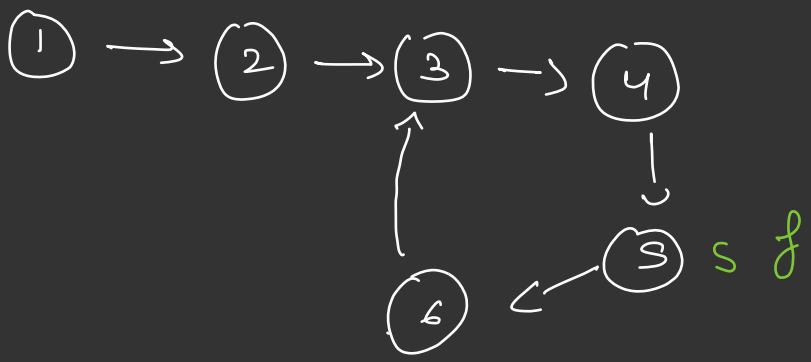
```
if (o.data < t.data) {
    rt.next = o;
    o = o.next;
}   rt = rt.next
else {
    rt.next = t;
    t = t.next;
}   rt = rt.next;
```

Ques Odd even





Ques Detect cycle/loop



Ques length of cycle