→ Partition list

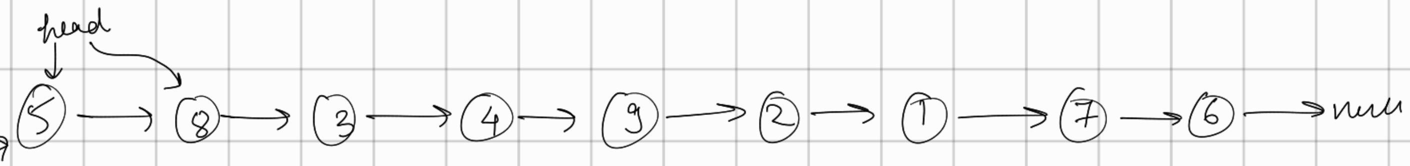
→ Sort 01, Sort 012

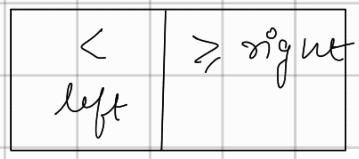
→ Quick Sort on LL

→ Clone LL ⟶ with & without extra space.

→ Delete w/o Head

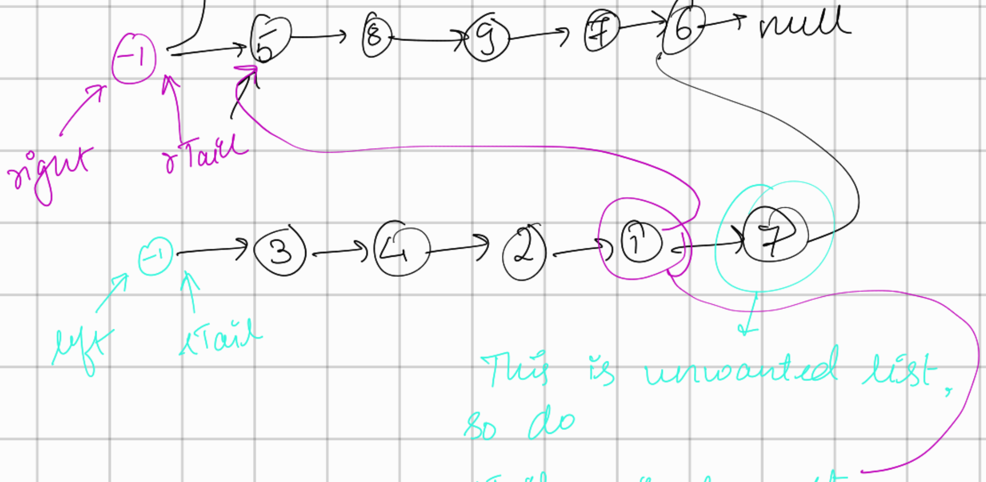
① Partition list.





pivot = head (5)

Partition observation: Pivot element reaches its correct posⁿ



This is unwanted list, so do

lTail = right.next

∴ We get the correct result

∵ 5 ≥ 5

right.next = head

rTail = rTail.next

head = head.next

∵ 8 ≥ 5

right.next = head

rTail = rTail.next

head = head.next

∵ 3 < 5

left.next = head

lTail = lTail.next

head = head.next.

① Partition on head node for correct result

② Partition on linked list is always stable

③ In Partitioning, the pivot element arrives at its correct posⁿ

CODE:

```
ListNode rDummy = new ListNode(-1), lDummy = new ListNode(-1);
ListNode rTail = rDummy, lTail = lDummy;
while (head != null) {
    if( head.val <x){
        lTail.next = head;
        lTail = head;
    } else {
        rTail.next = head;
        rTail = head;
    }

    head = head.next;
}
lTail.next = rDummy.next;
rTail.next = null;
return lDummy.next;
```

② Sort 01,

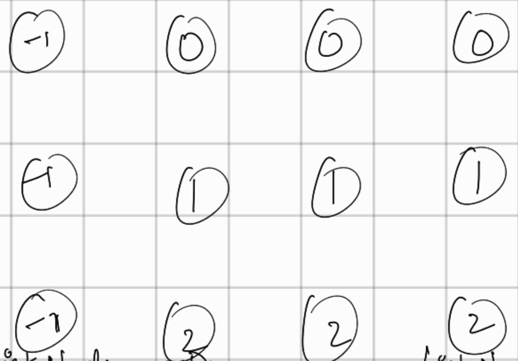
Same as above but only change condition, if(head.val ==0), else

```
ListNode rDummy = new ListNode(-1), lDummy = new ListNode(-1);
ListNode rTail = rDummy, lTail = lDummy;
while (head != null) {
    if( head.val == 0) {
        lTail.next = head;        } → all zeroes
        lTail = head;
    } else {
        rTail.next = head;        } → all ones
        rTail = head;
    }
```

```
        head = head.next;
    }
    lTail.next = rDummy.next;        → link 0 end & 1 begin
    rTail.next = null;               → last link null
    return lDummy.next;
```

③ Sort 012

$$\begin{tikzcd}
2 \arrow[r] & 1 \arrow[r] & 1 \arrow[r] & 0 \arrow[r] & 2 \arrow[r] & 1 \arrow[r] & 0 \arrow[r] & 0 \arrow[r] & 2 \arrow[r] & 1 \arrow[r] & \text{null}
\end{tikzcd}$$



```
ListNode rDummy = new ListNode(-1), lDummy = new ListNode(-1), cDummy = new
                                                                   ListNode(-1);
ListNode rTail = rDummy, lTail = lDummy, cTail = cDummy;
while (head != null) {
    if( head.val ==0){
        lTail.next = head;
        lTail = head;
    } else if( head.val ==1){
        rTail.next = head;
        rTail = head;
    } else{
        cTail.next = head,
        cTail = head;
    }
    head = head.next;
}
cTail.next = null;
rTail.next = cDummy.next;
lTail.next = rDummy.next;
return lDummy.next;
```

④ Quick Sort on LL.

$$\begin{tikzcd} 5 \arrow[r] & 8 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & 9 \arrow[r] & 2 \arrow[r] & 1 \arrow[r] & 7 \arrow[r] & 5 \arrow[r] & null \end{tikzcd}$$

Partition around head

$$\begin{tikzcd} left = & 3 \arrow[r] & 4 \arrow[r] & 2 \arrow[r] & 1 \arrow[dlll, bend left] & & \\ right = & 5 \arrow[r] & 8 \arrow[r] & 9 \arrow[r] & 7 \arrow[r] & 6 \arrow[r] & null \end{tikzcd}$$

5 node is at correct point.

CODE:

```
if(head == null || head.next == null)
      return null;

ListNode left = sortList(partition(head, head.val));
ListNode right = sortList(head.next);
head.next = right;

if(left == null) return head;
ListNode leftTail = getTail(left);
leftTail.next = head;
return left;
```

To optimize this function, use randomize quicksort