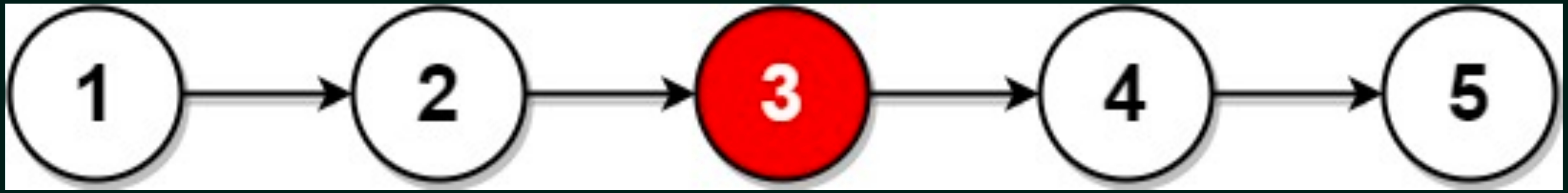
Linked List - 2

In This Lecture

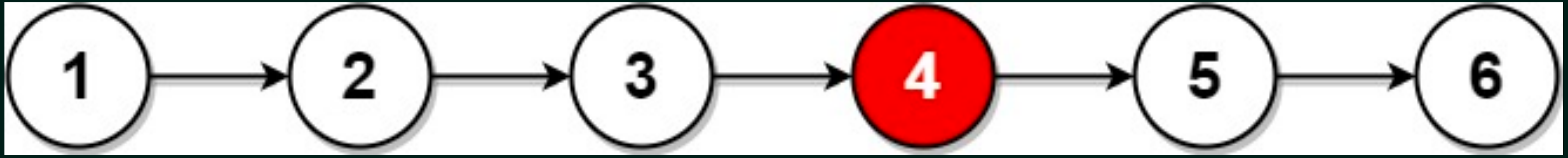
1. Find the Middle Node In A LinkedList
2. Remove Duplicates - I

Find the Middle Node In A LinkedList

Given the head of a singly linked list, return the middle node of the linked list.
If there are two middle nodes, return **the second middle** node.



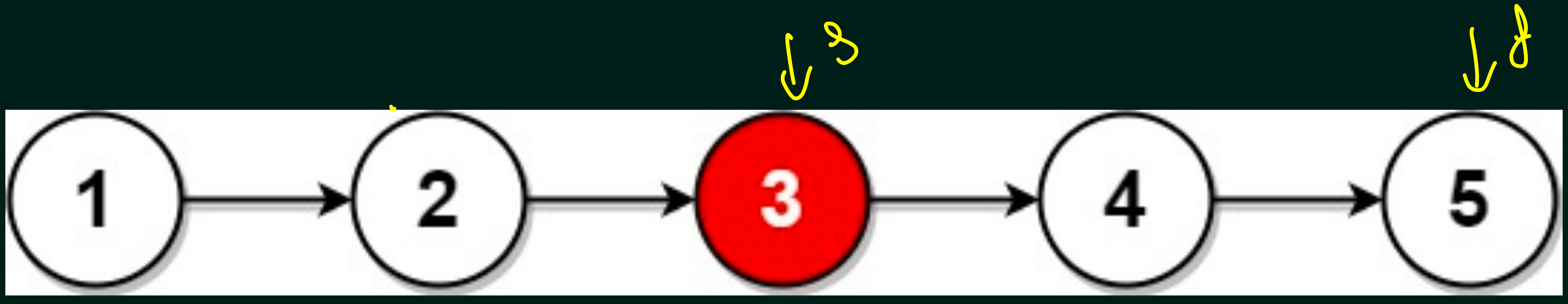
length = 5
jumps = length/2

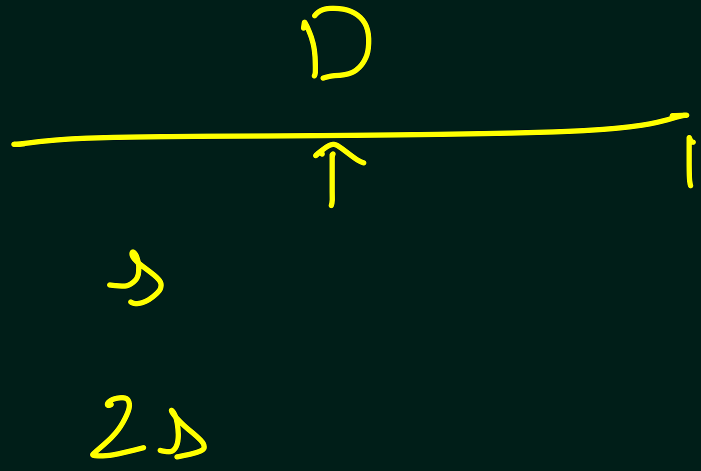


$\uparrow$ s

$\uparrow$ f

Find the Middle Node In A LinkedList





temp

count = ~~1~~ 5

jumps = 2

$\boxed{O(N)}$

Two times iterate

i) to calculate the size

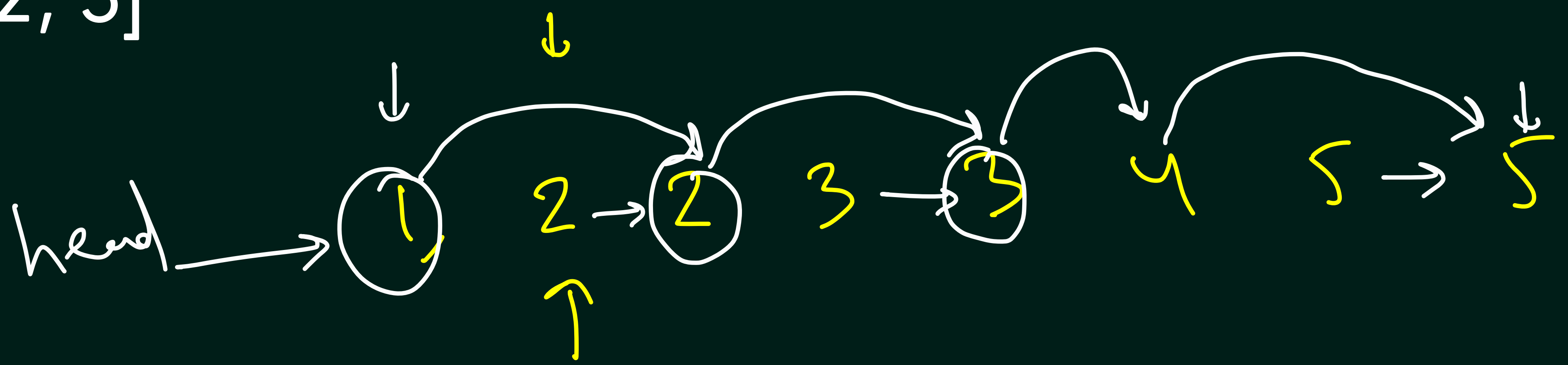
ii) to jump to the middle.

Remove Duplicates - I

Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well.

Input: l1 = [-1, -1, 2, 2, 3, 3, 3]

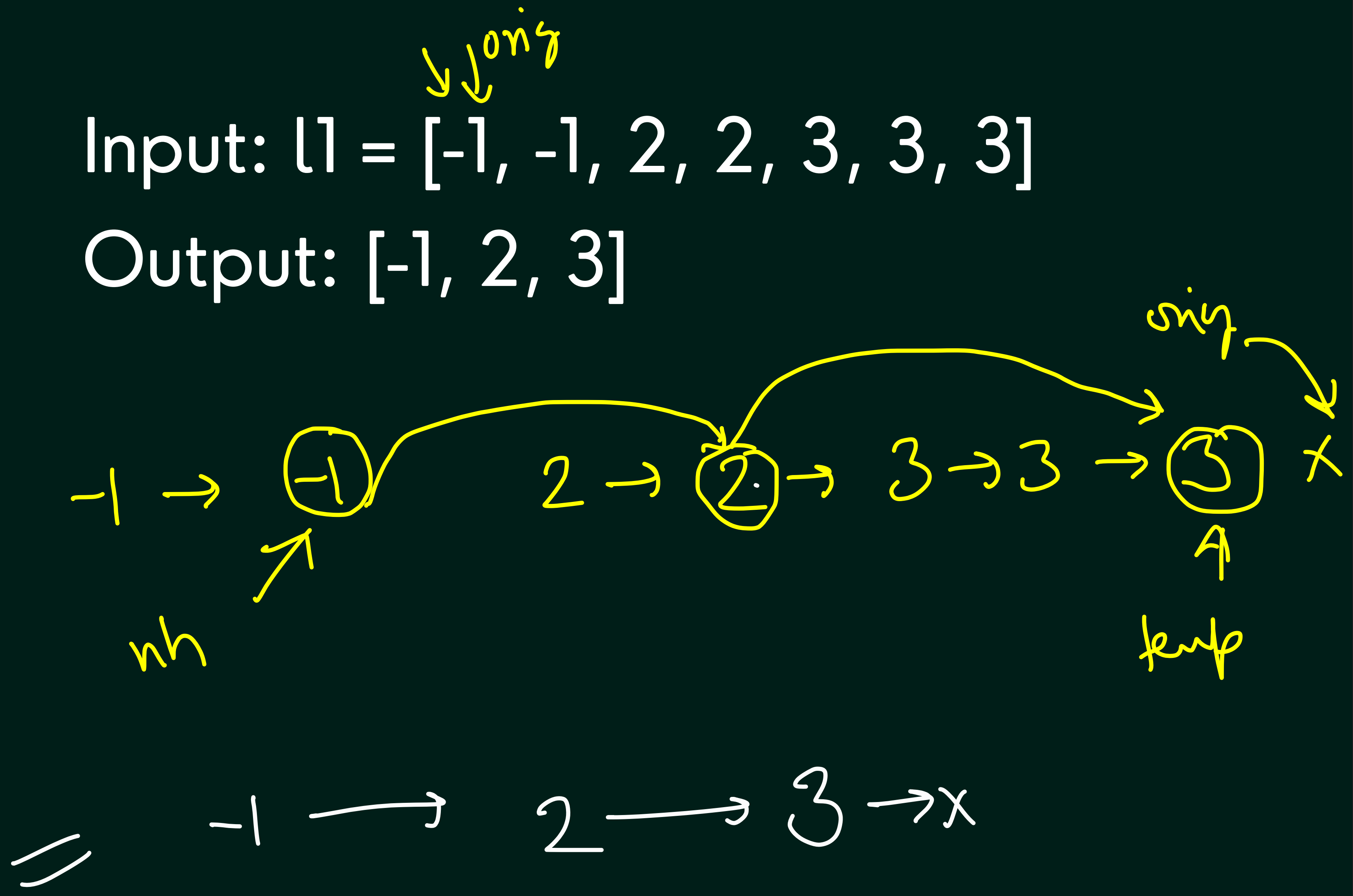
Output: [-1, 2, 3]



Remove Duplicates - I

Input: l1 = [-1, -1, 2, 2, 3, 3, 3]

Output: [-1, 2, 3]



```java
static Node removeDuplicateElements(Node head) {
    Node orig = head;
    Node newHead = null;
    Node temp = head;

    while(orig != null) {
        while(orig.next != null && orig.data == orig.next.data) {
            orig = orig.next;
        }
        if(newHead == null) {
            newHead = temp = orig;
        } else {
            temp.next = orig;
            temp = orig;
        }
        orig = orig.next; //
    }
    return newHead;
}
```