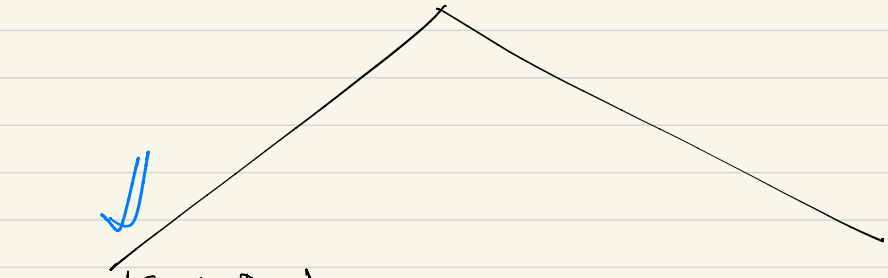
Linked List Notes (Rec-03)

2 Strategies



Slow / Fast Pointer

(Try this first)

→ Helps get to middle

Traverse and swap.

(Last option)

→ Helps move around things.

Linked List Are confusing!

General IDEA: Think only

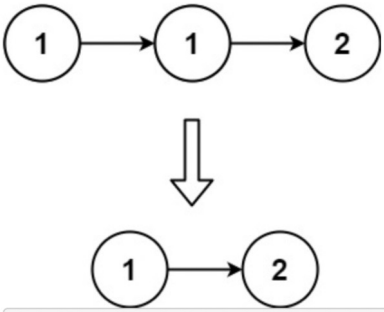
1. Elements ← You point @ these
2. Arrows ← You modify them

Q1]

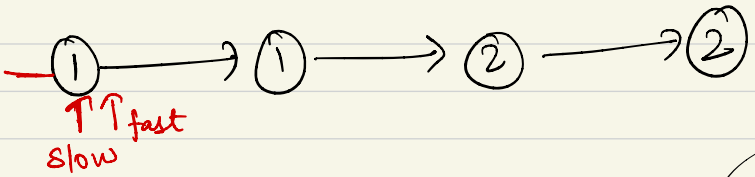
Fast and Slow Pointers

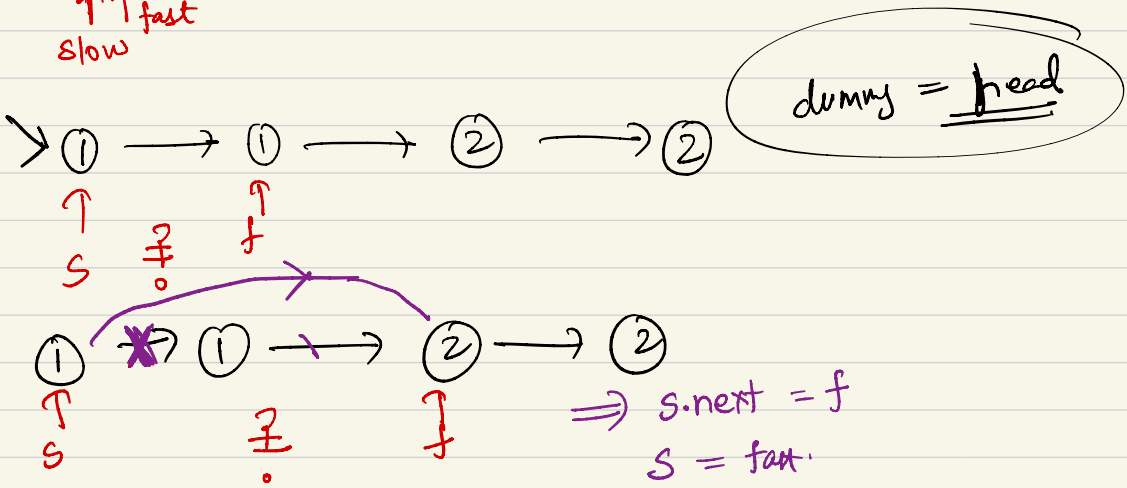
A. Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well.

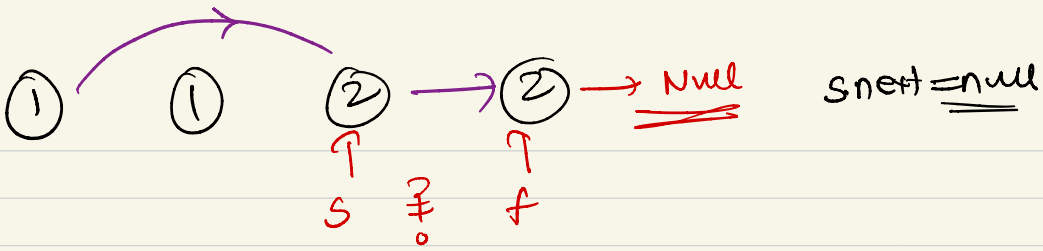
Example 1:




```
Input: head = [1,1,2]
Output: [1,2]
```






1
1
2
2
Null
snext=null
s
f

```java
public void deleteDuplicates() {
    if (head == null || head.getNext() == null) {
        return;
    }

    Node<E> slow = head;
    Node<E> fast = head.getNext();

    while (fast != null) {
        if (!slow.getElement().equals(fast.getElement())) {
            slow.setNext(fast);
            slow = slow.getNext();
        }
        fast = fast.getNext();
    }
    slow.setNext(null); // Disconnect any remaining duplicates
}
```

changing arrow

slow → fast

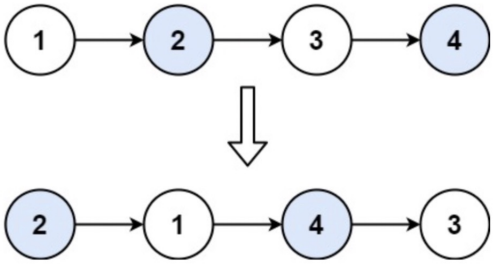
Moving ahead

list.deleteDuplicates()

Q 2

A. Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifying the values in the list's nodes (i.e., only nodes themselves may be changed.)

Example 1:




```
Input: head = [1,2,3,4]
Output: [2,1,4,3]
```

Example 2:

```
Input: head = []
Output: []
```

Example 3:

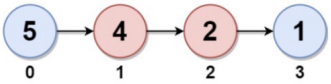
```
Input: head = [1]
Output: [1]
```

```java
public Node<E> swapPairs(Node<E> head) {
        // Dummy node acts as the starting point of the swapped list
        Node<E> dummy = new Node<>(null, head);
        Node<E> prev = dummy;

        while (prev.next != null && prev.next.next != null) {
            Node<E> curr = prev.next; // First node of the pair
            Node<E> next = curr.next; // Second node of the pair

            // Swapping
            curr.next = next.next;
            next.next = curr;
            prev.next = next;

            // Moving prev two nodes ahead
            prev = curr;
        }

        return dummy.next;
    }
```

Q3

B. In a linked list of size n, where n is even, the ith node (0-indexed) of the linked list is known as the twin of the (n-1-i) th node, if 0 <= i <= (n / 2) - 1. For example, if n = 4, then node 0 is the twin of node 3, and node 1 is the twin of node 2. These are the only nodes with twins for n = 4. The twin sum is defined as the sum of a node and its twin. Given the he ad of a linked list with even length, return the maximum twin sum of the linked list.

Example 1:




```
Input: head = [5,4,2,1]
Output: 6
Explanation:
Nodes 0 and 1 are the twins of nodes 3 and 2, respectively. All have twin sum = 6.
There are no other nodes with twins in the linked list.
Thus, the maximum twin sum of the linked list is 6.
```