Task 5: Removing Duplicates from a Sorted Linked List. A sorted linked list has been constructed with repeated elements. Describe an algorithm to remove all duplicates from the linked list efficiently.

To remove duplicates in a single pass through the list, with a time complexity of

$O(n)$, where n is the number of nodes in the list. Here is a step-by-step algorithm:

1. Algorithm

Initialize Pointers:

- Start with a pointer current at the head of the linked list.

Traverse the List:

- While current is not null and current.next is not null:
- Compare current node's value with current.next node's value.
- If they are equal, it means there's a duplicate. Remove the duplicate by changing current.next to current.next.next.
- If they are not equal, move the current pointer to the next node.

End Condition:

- The loop terminates when current or current.next becomes null.

Example in Java

```
class ListNode {
   int val;
   ListNode next;
   ListNode(int val) { this.val = val; }
}
public class RemoveDuplicates {
   public static ListNode removeDuplicates(ListNode head) {
      if (head == null) return null;
      ListNode current = head;
      while (current != null && current.next != null) {
        if (current.val == current.next.val) {
            current.next = current.next.next; // Remove the duplicate node
        } else {
            current = current.next; // Move to the next node
        }
    }
    return head;
}
public static void printList(ListNode head) {
    ListNode current = head;
    while (current != null) {
        System.out.print(current.val + " ");
        current = current.next; }
    System.out.println();    }
public static void main(String[] args) {
    // Creating a sorted linked list: 1 -> 1 -> 2 -> 3 -> 3 -> 4 -> 4 -> 5
    ListNode head = new ListNode(1);
    head.next = new ListNode(1);
    head.next.next = new ListNode(2);
    head.next.next.next = new ListNode(3);
    head.next.next.next.next = new ListNode(3);
    head.next.next.next.next.next = new ListNode(4);
    head.next.next.next.next.next.next = new ListNode(4);
    head.next.next.next.next.next.next.next = new ListNode(5);
    System.out.println("Original List:");
    printList(head);
    head = removeDuplicates(head);
    System.out.println("List after removing duplicates:");
    printList(head); }}
```

Explanation of the Code

List Node Class:

- A basic definition of a singly linked list node with an integer value and a next pointer.

Remove Duplicates Method:

- Checks if the list is empty (head == null). If so, returns null.
- Uses a current pointer to traverse the list.
- In each iteration of the loop, checks if the current node's value is equal to the next node's value.
- If equal, sets current.next to current.next.next, effectively removing the duplicate node.
- If not equal, moves current to the next node.

Print List Method:

- Helper function to print the values of the linked list nodes.

main Method:

- Creates a sample sorted linked list with duplicates.
- Prints the original list.
- Calls removeDuplicates to remove duplicates.
- Prints the modified list.