

Solution to the Problem of Removing Nodes from a Linked List

★ Understanding the Problem

The task is to remove all nodes from a linked list that contain a specific value (key) and return the updated head of the list.

★ Solution

This problem can be solved using pointers to traverse the linked list and remove nodes that match the given value. Special cases, such as removing the first node or consecutive nodes containing the target value, must be handled carefully.

★ Code

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def removeElements(head: ListNode, key: int) -> ListNode:
    # Create a dummy node to simplify handling the head of the list
    dummy = ListNode(0)
    dummy.next = head
    current = dummy

    while current.next:
        if current.next.val == key:
            current.next = current.next.next
        else:
            current = current.next

    return dummy.next
```

★ Explanation

1. **Dummy Node**: A dummy node is used to simplify operations, especially when the head node itself contains the value to be removed.
2. **Traversal**: A pointer `current` is used to traverse the list. At each step, we check if the next node contains the target value.
3. **Removal**: If the next node contains the target value, it is skipped by linking the current node to the node that follows the next one.
4. **Moving to the Next Node**: If the next node does not contain the target value, the pointer is moved to the next node.

★ Time and Space Complexity

- **Time**: $O(n)$, where n is the number of nodes in the linked list, as each node is visited once.
- **Space**: $O(1)$, as only a constant amount of additional space is used, regardless of the list size.