



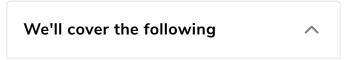






Solution Review: Remove Duplicates from Linked List

This review provides an analysis of the solution to the Remove Duplicates from a Linked List challenge.



- Solution: Using a Set/Hash Table
 - Time Complexity

Solution: Using a Set/Hash Table

```
main.py
LinkedList.py
Node.py
    from LinkedList import LinkedList
  2
     from Node import Node
  3
  4
     def remove_duplicates(lst):
         current_node = lst.get_head()
  7
         prev_node = lst.get_head()
  8
         # To store values of nodes which we already visited
  9
         visited nodes = set()
         # If List is not empty and there is more than 1 element in List
 10
 11
         if not lst.is_empty() and current_node.next_element:
 12
             while current_node:
 13
                 value = current node.data
```

This is, perhaps, the most efficient way of removing duplicates from a linked list. We've seen this approach before in Challenge 10 when we detected a loop in our linked list.

Every node we traverse is added to the visited_nodes set. If we reach a
node that already exists in the set, it must be a duplicate.

prev_node is used to keep track of the preceding node. This allows us to
easily manipulate the previous and next nodes during the deletion of our
current_node.

Time Complexity#

This is a linear algorithm, hence, the time complexity is O(n).

Next, we'll learn how to apply **union** and **intersection** operations on linked lists.



Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. See how ①







← Back

Challenge 11: Remove Duplicates fro...



Challenge 12: Union & Intersection of ...



Completed



Report an Issue

