Solution Review: Insertion at Tail
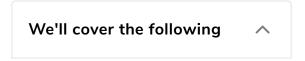
This review provides a detailed analysis of the solution to the Insertion at Tail challenge.

> **We'll cover the following** ∧
>
> - Solution: List Traversal
>   - Time Complexity

Solution: List Traversal

```
main.py

LinkedList.py

Node.py
```

```python
1   from LinkedList import LinkedList
2   from Node import Node
3   # Access HeadNode => list.getHead()
4   # Check if list is empty => list.isEmpty()
5   # Node class  { int data ; Node nextElement;}
6
7   # Inserts a value at the end of the list
8
9
10  def insert_at_tail(lst, value):
11      # Creating a new node
12      new_node = Node(value)
13
14      # Check if the list is empty, if it is simply point head to new n
15      if lst.get_head() is None:
16          lst.head_node = new_node
17          return
```

```
18
19        # if list not empty, traverse the list to the last
20        temp = lst.get_head()
21
22        while temp.next_element:
23            temp = temp.next_element
24
25        # Set the nextElement of the previous node to new node
26        temp.next_element = new_node
27        return
28
```

Insertion At End

If you grasped the logic behind insertion at the head of a linked list, this shouldn't be much of a challenge. If the list is empty, the situation is exactly like insertion at head. Otherwise, we can simply use a loop to reach the tail of the list and set our new node as the `next_element` of the last node (**line 26**).

Time Complexity

This algorithm traverses the entire linked list and hence, works in *O(n)* time.

At this point, we have covered the first two types of insertions. The last one, **Insertion at the kth Position**, is just an extension of these two. If you need to insert a node at a specific index in the list, simply iterate to that position and appropriately switch pointers. Try it out on your own.

In the next lesson, we will discuss the search algorithm for linked lists.

Interviewing soon? We've partnered with Hired so that

companies apply to you instead of you applying to them.

[how](#) ⓘ

← Back

Challenge 1: Insertion at Tail

Next →

Challenge 2: Search in a Singly Linked …

✔ Completed

⚠ Report an Issue