



# Solution Review: Find the Length of a Linked List

This review provides a detailed analysis of the solution to the Find the Length of a Linked List challenge.

## We'll cover the following ^

- Solution: Linear Iteration
- Time Complexity

## Solution: Linear Iteration #

main.py

LinkedList.py

Node.py

```
1 from Node import Node
2 from LinkedList import LinkedList
3
4
5 def length(lst):
6     # start from the first element
7     curr = lst.get_head()
8     length = 0
9
10    # Traverse the list and count the number of nodes
11    while curr:
12        length += 1
13        curr = curr.next_element
```

```
14     return length
15
16
17     lst = LinkedList()
18     lst.insert_at_head(4)
19     lst.insert_at_head(3)
20     lst.insert_at_head(2)
21     lst.insert_at_head(1)
22     lst.insert_at_head(0)
23     print(length(lst))
24
```



The logic is very similar to that of the `search` function. The trick is to iterate through the list and keep count of how many nodes you've visited. This count can be kept in the `length` variable.

## Time Complexity #

Since this is a linear algorithm, the time complexity will be  $O(n)$ .

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#) ⓘ



← Back

Next →

Challenge 4: Find the Length of a Link...

Challenge 5: Reverse a Linked List



Complete



