

Basic Linked List Operations

This lesson lists the various operations that can be performed on linked lists

We'll cover the following ^

- get_head()
 - Time complexity
- is_empty()
 - Explanation
 - · Time complexity

The primary operations which are generally a part of the LinkedList class are listed below:

- get_head() returns the head of the list
- insert_at_tail(data) inserts an element at the end of the linked list
- insert_at_head(data) inserts an element at the start/head of the linked list
- delete(data) deletes an element with your specified value from the linked list
- delete_at_head() deletes the first element of the list
- search(data) searches for an element with the specified value in the linked list
- is_empty() returns true if the linked list is empty

If you observe the list of functions mentioned above, get_head() and is_empty() are helper functions that will prove useful in all the others.

So let's define them first.

get_head()

This method simply returns the head node of our linked list:

```
from Node import Node
                                1
LinkedList.py
                                2
                                3
                                4 class LinkedList:
Node.py
                                5
                                       def __init__(self):
                                6
                                           self.head_node = None
                                7
                                8
                                       def get head(self):
                                9
                                           return self.head_node
                               10
                               11
                               12 lst = LinkedList() # Linked List created
                               13
                                   print(lst.get_head()) # Returns None since headNode does not contain
                               14
```



Time complexity

The time complexity for get_head() is O(1) as we simply return the head.

is_empty()

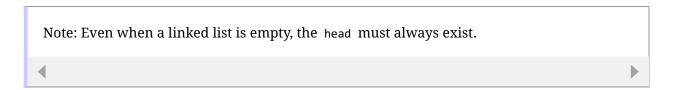
The basic condition for our list to be considered empty is that there are no nodes in the list. This implies that head points to None.

With that in mind, let's write down the simple implementation for is_empty():

```
from Node import Node
LinkedList.py
                                 3
                                 4
                                    class LinkedList:
Node.py
                                 5
                                         def __init__(self):
                                             self.head_node = None
                                 6
                                 7
                                 8
                                        def get_head(self):
                                 9
                                             return self.head_node
                                10
                                11
                                         def is_empty(self):
                                12
                                             if self.head node is None: # Check whether the head is None
                                13
                                                 return True
                                14
                                             else:
                                                 return False
                                15
                                16
                                18  lst = LinkedList() # Linked List created
                                19
                                    print(lst.is_empty()) # Returns true
                                20
\triangleright
                                                                                                          []
                                                                                            \leftarrow
```

Explanation

Nothing tricky going on here. The crux of the code lies in the if condition on **line 12**. We merely check if the head points to None.



Time complexity

It will be in O(1) as all we need to do is check whether the head node points to None or not.

This is just the tip of the iceberg. We'll tackle each of the remaining methods in the following lessons and apply them in relevant problems.

In the next lesson, we'll begin our discussion on linked list insertion functions.

