Data Structures & Algorithms – Lab #4

Aim: Getting Familiar with Linked Lists

Topics:

1. Implementing of a Node
2. Implementing of a Linked List
3. Linked List Operations

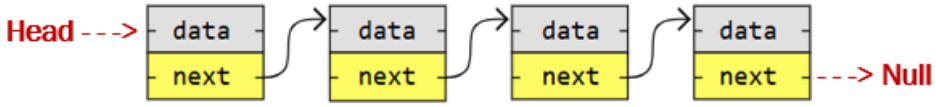
Lab Question

Q1 – Implement the **Linked List** data structure.

- First create a **Node** class.
- Then create a **Linked List** class.
- Consider the following operations for the linked list:
    - **isEmpty( )** → This method returns True if the linked list is empty, otherwise it returns False.
    - **append(item)** → This method adds a node with value=**item** to the end of the linked list.
    - **preappend(item)** → This method adds a node with value=**item** to the beginning of the linked list.
    - **remove(item)** → This method removes a node with value=**item** from the linked list.
    - **display( )** → This method displays all nodes values in the linked list.
    - **search(item)** → This method searches for a value in the list and returns **True** if the value is found, otherwise it returns **False**.
    - **size( )** → This method returns number of nodes(items) in the linked list.

Before creating the linked list, let's have a review on linked list components:

- Each linked list is **a collection of nodes** (So, first we will create a class for a node.)
  - Each node has two parts: **data** and **next**. Next part is a link to the next node.
  - The last node's next is Null.
- Each linked list has a pointer called **Head**. Head is a pointer that contains the address of the first element in the Linked List.



Creating the Node Class

```
class Node:
    def __init__(self,data):
        self.data = data
        self.next = None
```

Creating the Linked List Class (with methods)

```
class linkedList:
    def __init__(self):
        self.head = None
```

```
    def isEmpty(self):
        return self.head == None
```

```
    def append(self, item):
        new = Node(item)
        if self.head is None:
            self.head = new
        else:
            last_node = self.head
            while last_node.next:
                last_node = last_node.next
            last_node.next = new
```

```
    def preappend(self,item):
        new = Node(item)
        new.next = self.head
        self.head = new
```

```
    def remove(self,item):
        current = self.head
        previous = None
        found = False
        while not found:
            if current.data == item:
                found = True
            else:
                previous = current
                current = current.next

        if previous==None:
            self.head = current.next
        else:
            previous.next = current.next
```

```
    def display(self):

        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
```

```
    def size(self):
        current = self.head
        count = 0
        while current != None:
            count = count + 1
            current = current.next
        return count
```

```
    def search(self,item):
        current = self.head
        found = False
        while current!=None and not found:
            if current.data == item:
                found = True
            else:
                current = current.next
        return found
```

Now, it's time to create an empty object of a linked list, and call all linked list methods to add nodes to the list, remove elements from it, search for an item, and so on.

```
print("Displaying linked list after using append() and preappend() methods:")
mylist.display()

# Printing number of elements in the linked list
total = mylist.size()
print("Total number of elements in the linked list:", total)

# Searching for an element in the linked list
item = 30
if mylist.search(item)==True:
    print(item, "was found in the linked list!")
else:
    print(item, "was not found in the linked list!")

# Removing an element from the linked list
item = 20
print("Displaying linked list after removing", item,"from it:")
mylist.remove(item)
mylist.display()
```

```
Displaying linked list after using append() and preappend() methods:
10 -> 20 -> 30 -> 40 ->

Total number of elements in the linked list: 4


30 was found in the linked list!


Displaying linked list after removing 20 from it:
10 -> 30 -> 40 ->
```