LinkedList in Java

Some simple function to control a LinkedList

-by Swarnendu

Definition of Node

```
public static class Node{
    int data;
    Node next;//points towards next node
    public Node(int data){
        this.data=data;
        this.next=null;
    }
//declaration of head and tail
    public static Node head;
    public static Node tail;
    public static int size;
```

Function to add at oth index

Intuition:

Step 1 => create a new node, which will be added.

Step 2 => make newNode's next to head.

Step 3 => now shift the head to newNode.

Edge Condition => if the LinkedList is empty then then head will be null. For that we are adding a if condition head=tail=newNode

```
public void addFirst(int data){
 //step 1 => create a new node
 Node newNode = new Node(data);
 if(head==null){
     head=tail=newNode;
     return;
 }
 //step 2 => newNode.next=head
 newNode.next=head;
 //step 3 => head =newNode
 head=newNode;
}
```

Time Complexity =>O(1)

Space Complexity =>O(1)

Function to add at Last index

Intuition:

Step 1 => create a new node, which will be added.

Step 2 => make tail's next to newNode.

Step 3 => now shift the tail to newNode.

Edge Condition => if the LinkedList is empty then then head will be null. For that we are adding an if condition head=tail=newNode

```
public void addLast(int data){
   Node newNode=new Node(data);
   if(head==null){
    head=tail=newNode;
   }
   tail.next=newNode;
   tail=newNode;
}
```

Time Complexity =>O(1)

Space Complexity =>O(1)

Function to remove Node of 0^{th} index

Intuition:

Step 1 =>. First, we will store the value of data at 0^{th} index.

Step 2 => head = head's next.

Step 3 => we will return the value.

Edge Condition 1 => if the LinkedList is empty then then head will be null. For that, we will print LinkedList is empty and we are not returning a valid value, we are returning Integer.MIN_VALUE.

Edge Condition 2 => if the LinkedList have one element then size will be 1. For that head=tail=null.

```
public int removeFirst(){
    if(size==0){
        System.out.println("LinkedList is empty");
        return Integer.MIN_VALUE;
    }
    else if(size==1){
        int value=head.data;
        head=tail=null;
        size--;
        return value;
    }
    int value=head.data;
    head=head.next;
    size--;
    return value;
}
```

Time Complexity =>O(1)

Space Complexity =>O(1)

Function to remove Last Node

Intuition:

Step 1 =>. First we will iterate in the loop to find the node which we will be deleting. We will take a temp Node for that and we will update it. When we will reach the particular index, we will store the data.

Step 2 => tail =temp.next=null.

Step 3 => we will return the value.

Edge Condition 1 => if the LinkedList is empty then then head will be null. For that, we will print LinkedList is empty and we are not returning a valid value, we are returning Integer.MIN_VALUE.

Edge Condition 2 => if the LinkedList have one element then size will be 1. For that head=tail=null.

```
public int removeLast(){
     if(size==0){
         System.out.println("LinkedList is empty");
         return Integer.MIN_VALUE;
     }
     else if(size==1){
         int val=head.data;
         head=tail=null;
         size--;
         return val;
     }
     int i=0;
     Node temp=head;
     int val=0;
     while(temp.next.next!=null){
         i++;
         temp=temp.next;
         val=temp.next.data;
     }
     tail=temp.next=null;
     size--;
     return val;
 }
```

Time Complexity =>O(n)

Space Complexity =>O(1)

Function to Print LinkedList

Intuition:

Step 1 =>. We will take a temporary Node temp.

Step 2 => we will keep updating it while temp is not equal to null.

Step 3 => and we will be printing the all the data of a LinkedList

```
public void Print(){
  Node temp = head;
  while(temp!=null){
      System.out.print(temp.data + "->");
      temp=temp.next;
  }
  System.out.println("null");
 }
```

Time Complexity =>O(n)

Space Complexity =>O(1)

Function to Iterative Search node in a LinkedList

Intuition:

Step 1 => first we will pass the key element to the function.

Step 2 => then simply we will go to each node and check the data is matching with the key or not.

Step 3 => if 'yes' then we will return the index otherwise we will return -1 as the key not found.

```
public int search(int key){
    Node temp=head;
    for(int i=0;i<size;i++){
        if(key==temp.data){
            return i;
        }
        temp=temp.next;
    }
    return -1;
}
```

Time Complexity =>O(n)

Space Complexity =>O(1)

Function to Recursive Search node in a LinkedList

Intuition:

Step 1 => we will keep updating the head value by recurtion.

Step 2 => simply we will check the data is matching with the key or not. Also we will keep updating index by +1.

Step 3 => if 'yes' then we will return the index otherwise we will return -1 as the key not found.

```
public int helper(Node head,int key){
    if(head==null){
      return -1;
    }
    if(head.data==key){
        return 0;
    }
    int index = helper(head.next, key);
    if(index==-1){
        return -1;
    }
    return index+1;
    }
```

Time Complexity =>O(n)

Space Complexity =>O(1)

Function to Remove nth node form last in a LinkedList

Intuition:

Step 1 => we will calculate the size of the LinkedList. And iterate (size-n-1) time in a loop, thus we will get the node.

Step 2 => simply we will store the value in a variable.

Step 3 => and we will update temp.next=temp.next.next.

Step 4 =>we will return the value.

Edge Condition 1 => if the LinkedList is empty then then head will be null. For that, we will print LinkedList is empty and we are not returning a valid value, we are returning Integer.MIN_VALUE.

Edge Condition 2 => if the n==size the we will simply call the removeLast() which we defined earlier and return.

```
public int removeNthFromLast(int n){
      if(n==size){
         return removeFirst();
      }
      Node temp=head;
      for(int i=0;i<size-n-1;i++){
          temp=temp.next;
      }
      size--;
      int val =temp.next.data;
      temp.next=temp.next.next;
      return val;
  }
```

Time Complexity =>O(n)

Space Complexity =>O(1)

Thank You 

Follow me on:

LinkedIn