Friday, January 21, 2022 2:17 PM

Delete nodes having greater value on right

Medium Accuracy: 37.92% Submissions: 52419 Points: 4

Given a singly linked list, remove all the nodes which have a greater value on its following nodes.

Example 1:

Input:
LinkedList = 12->15->10->11->5->6->2->3
Output: 15 11 6 3
Explanation: Since, 12, 10, 5 and 2 are
the elements which have greater elements
on the following nodes. So, after deleting
them, the linked list would like be 15,
11, 6, 3.
Example 2:

Input:
LinkedList = 10->20->30->40->50->60
Output: 60
Your Task:

The task is to complete the function **compute**() which should modify the list as required and return the head of the modified linked list. The **printing** is done by the **driver** code,

Expected Time Complexity: O(N)

Expected Auxiliary Space: O(1)

Constraints:

1 ≤ size of linked list ≤ 1000

1 ≤ element of linked list ≤ 1000

Note: Try to solve the problem without using any extra space.

From <https://practice.geeksforgeeks.org/problems/delete-nodes-having-greater-value-on-right/1#>

```java
import java.util.LinkedList;
public class DeleteNodeHavingSmallerValyeThanFollowing {

    class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
        }
    }

    Node head;
    int size = 0;

    Node compute(Node head)
    {
        // your code here
        head=reverse(head);

         Node curr=head;
            while (curr!=null && curr.next!=null) {
                Node next=curr.next;

            if (curr.data>next.data) {

               curr.next=next.next;

               next.next=null;
            }
            else
            {
                curr=next;
            }
            }
             head=reverse(head);

        return head ;


    }
    public Node reverse(Node head)
    {
        if(head==null || head.next==null)
        {

            // System.out.println(" Rev fun call : return ");


            return head;
        }
        Node prev=head;
        Node curr=head.next;

        while(curr!=null)
        {
            Node next=curr.next;

            curr.next=prev;
            prev=curr;
            curr=next;

        }
        head.next=null;
        head=prev;
        // System.out.println(" Rev fun call :: success ! ");


        return head ;

    }

    public Node reverse() {
        if (head == null || head.next == null)
            return head;
        Node prev = head;
        Node curr = prev.next;
        while (curr != null) {

            Node next = curr.next;

            curr.next = prev;

            prev = curr;

            curr = next;
        }
        head.next = null;

        head = prev;

        return head;
    }
    public void print() {
        if (head == null) {
            System.out.println("Null");
            return;
        }
        Node temp = head;
        while (temp != null) {

            System.out.print(temp.data + "-->");

            temp = temp.next;
        }
        System.out.print("Null");
    }
    public void addData(int data) {
        if (head == null) {

            head = new Node(data);
            return;
        }

        Node node = new Node(data);

        node.next = head;

        head = node;

        this.size++;
    }
    public int size() {
        return this.size;
    }
    public static void main(String[] args) {

        // LinkedList <Integer> ll=new LinkedList<>();

        int arr[] = { 12, 15, 10, 11, 5, 6, 2, 3 };
        // for (int i = 0; i < arr.length; i++) {
        // ll.add(arr[i]);
        // }
        DeleteNodeHavingSmallerValyeThanFollowing ll = new
DeleteNodeHavingSmallerValyeThanFollowing();

        for (int i = 0; i < arr.length; i++) {
            ll.addData(arr[i]);
        }
        int size = ll.size();
        System.out.println(" size : " + size);

        // System.out.println(ll);
        System.out.println();

        ll.print();

        ll.reverse();
        System.out.println();

        ll.print();

        ll.reverse();
        System.out.println();

        ll.print();
    }
}
```