Flattening a Linked List

Medium Accuracy: 33.91% Submissions: 74040 Points: 4

Given a Linked List of size N, where every node represents a sub-linked-list and contains two pointers:

- (i) a **next** pointer to the next node,
- (ii) a **bottom** pointer to a linked list where this node is head.

Each of the sub-linked-list is in sorted order.

Flatten the Link List such that all the nodes appear in a single level while maintaining the sorted order.

**Note:** The flattened list will be printed using the bottom pointer instead of next pointer.

### Example 1:

### Input:

**Output:** 5-> 7-> 8- > 10 -> 19-> 20-> 22-> 28-> 30-> 35-> 40-> 45-> 50.

# **Explanation**:

The resultant linked lists has every node in a single level.

(Note: | represents the bottom pointer.)

## Example 2:

## Input:

8 50 | 30

**Output:** 5->7->8->10->19->20->22->30->50

**Explanation:** 

The resultant linked lists has every

node in a single level.

(Note: | represents the bottom pointer.)

#### Your Task:

You do not need to read input or print anything. Complete the function **flatten()** that takes the **head** of the linked list as input parameter and returns the head of flattened link list.

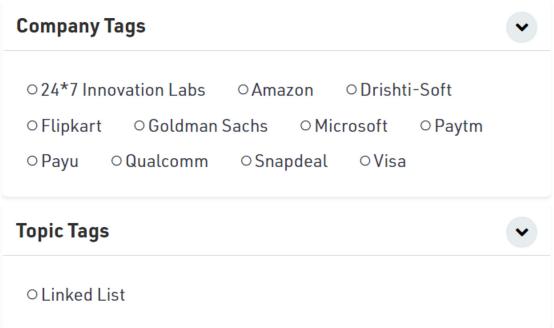
**Expected Time Complexity:** O(N\*M)

**Expected Auxiliary Space:** O(1)

#### **Constraints:**

0 <= N <= 50

1 <= **M**i <= 20





### **Related Courses**



- O Amazon SDE Preparation Test Series
- O Microsoft SDE Preparation Test Series
- Google SDE Preparation Test Series
- O Must Do Coding Questions Self Paced

# **Related Interview Experiences**

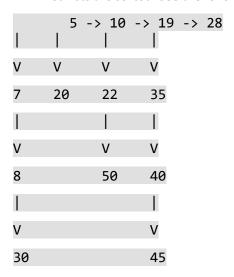


O Paytm interview experience set 8 hiring drive for backend engineer

Given a linked list where every node represents a linked list and contains two pointers of its type:

- (i) Pointer to next node in the main list (we call it 'right' pointer in the code below)
- (ii) Pointer to a linked list where this node is headed (we call it the 'down' pointer in the code below).

All linked lists are sorted. See the following example



Write a function flatten() to flatten the lists into a single linked list. The flattened linked list should also be sorted. For example, for the above input list, output list should be 5->7->8->10->19->20->22->28->30->35->40->45->50.

The idea is to use the Merge() process of <u>merge sort for linked lists</u>. We use merge() to merge lists one by one. We recursively merge() the current list with the already flattened list.

The down pointer is used to link nodes of the flattened list.

Below is the implementation of the above approach:

From < https://practice.geeksforgeeks.org/problems/flattening-a-linked-list/1>

```
// Java program for flattening a Linked List
class LinkedList
  Node head; // head of list
  /* Linked list Node*/
  class Node
    int data;
    Node right, down;
    Node(int data)
      this.data = data;
      right = null;
      down = null;
    }
  }
  // An utility function to merge two sorted linked lists
  Node merge(Node a, Node b)
  {
    // if first linked list is empty then second
    // is the answer
    if (a == null) return b;
    // if second linked list is empty then first
    // is the result
    if (b == null) return a;
    // compare the data members of the two linked lists
    // and put the larger one in the result
    Node result:
    if (a.data < b.data)
    {
```

```
result = a;
    result.down = merge(a.down, b);
  }
  else
    result = b;
    result.down = merge(a, b.down);
  }
  result.right = null;
  return result;
}
Node flatten(Node root)
  // Base Cases
  if (root == null | | root.right == null)
    return root;
  // recur for list on right
  root.right = flatten(root.right);
  // now merge
  root = merge(root, root.right);
  // return the root
  // it will be in turn merged with its left
  return root;
}
/* Utility function to insert a node at beginning of the
 linked list */
Node push(Node head_ref, int data)
  /* 1 & 2: Allocate the Node &
       Put in the data*/
  Node new_node = new Node(data);
  /* 3. Make next of new Node as head */
  new_node.down = head_ref;
  /* 4. Move the head to point to new Node */
  head_ref = new_node;
  /*5. return to link it back */
  return head_ref;
}
void printList()
  Node temp = head;
  while (temp != null)
```

```
System.out.print(temp.data + " ");
      temp = temp.down;
    }
    System.out.println();
  }
  /* Driver program to test above functions */
  public static void main(String args[])
    LinkedList L = new LinkedList();
    /* Let us create the following linked list
      5 -> 10 -> 19 -> 28
      I I I I
      V V V
      7 20 22 35
             1 1
      ٧
             V V
      8
             50 40
      ٧
                ٧
      30
                 45
    */
    L.head = L.push(L.head, 30);
    L.head = L.push(L.head, 8);
    L.head = L.push(L.head, 7);
    L.head = L.push(L.head, 5);
    L.head.right = L.push(L.head.right, 20);
    L.head.right = L.push(L.head.right, 10);
    L.head.right.right = L.push(L.head.right.right, 50);
    L.head.right.right = L.push(L.head.right.right, 22);
    L.head.right.right = L.push(L.head.right.right, 19);
    L.head.right.right = L.push(L.head.right.right.right, 45);
    L.head.right.right.right = L.push(L.head.right.right.right, 40);
    L.head.right.right = L.push(L.head.right.right.right, 35);
    L.head.right.right.right = L.push(L.head.right.right.right, 20);
    // flatten the list
    L.head = L.flatten(L.head);
    L.printList();
}/* This code is contributed by Rajat Mishra */
Output:
5 7 8 10 19 20 20 22 30 35 40 45 50
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.
Useful links to help you understand the concepts of this problem:
From < https://practice.geeksforgeeks.org/problems/flattening-a-linked-list/1>