Assignment-13

17 June 2023 14:04

Question 1

Given two linked list of the same size, the task is to create a new linked list using those linked lists. The condition is that the greater node among both linked list will be added to the new linked list.

Examples:

Input: list1 = 5->2->3->8

list2 = 1->7->4->5

Output: New list = 5->7->4->8

Input:list1 = 2->8->9->3

list2 = 5->3->6->4

Output: New list = 5->8->9->4

```
def newList(root1, root2):

  ptr1 = root1
  ptr2 = root2

  root = None
  while (ptr1 != None) :
    temp = Node(0)
    temp.next = None

    # Compare for greater node
    if (ptr1.data < ptr2.data):
      temp.data = ptr2.data
    else:
      temp.data = ptr1.data

    if (root == None):
      root = temp
    else :
      ptr = root
      while (ptr.next != None):
        ptr = ptr.next

      ptr.next = temp

    ptr1 = ptr1.next
    ptr2 = ptr2.next

  return root
```

Question 2

Write a function that takes a list sorted in non-decreasing order and deletes any duplicate nodes from the list. The list should only be traversed once.

For example if the linked list is 11->11->11->21->43->43->60 then removeDuplicates() should convert the list to 11->21->43->60.

Example 1:

Input:

LinkedList:

11->11->11->21->43->43->60

Output:

11->21->43->60

Example 2:

Input:

LinkedList:

11->11->11->21->43->43->60

Output:

11->21->43->60

```
def removeDuplicates(head):
```

```
  #code here
  prev=head
  curr=head
  while curr is not None:



    if(prev.data==curr.data):
      curr=curr.next
      prev.next=curr
    else:
      prev=curr
      curr = curr.next
  return head
```

Question 3

Given a linked list of size **N**. The task is to reverse every **k** nodes (where k is an input to the function) in the linked list. If the number of nodes is not a multiple of *k* then left-out nodes, in the end, should be considered as a group and must be reversed (See Example 2 for clarification).

Example 1:

Input:

LinkedList: 1->2->2->4->5->6->7->8

K = 4

Output:4 2 2 1 8 7 6 5

Explanation:

The first 4 elements 1,2,2,4 are reversed first

and then the next 4 elements 5,6,7,8. Hence, the

resultant linked list is 4->2->2->1->8->7->6->5.

Example 2:

Input:

LinkedList: 1->2->3->4->5

K = 3

Output:3 2 1 5 4

Explanation:

The first 3 elements are 1,2,3 are reversed

first and then elements 4,5 are reversed.Hence,

the resultant linked list is 3->2->1->5->4.

```
class Solution:
  def reverse(self,head, k):
    curr = head
    prev = None

    if k<=1 or curr is None:
      return head
    c=k
    while curr and c>0:
      next = curr.next
      curr.next = prev
      prev = curr
      curr = next
      c-=1
    head.next = self.reverse(curr,k)
    return prev
```

Question 4

Given a linked list, write a function to reverse every alternate k nodes (where k is an input to the function) in an efficient way. Give the complexity of your algorithm.

Example:

Inputs: 1->2->3->4->5->6->7->8->9->NULL and k = 3
Output: 3->2->1->4->5->6->9->8->7->NULL.

```
def kAltReverse(head, k) :
  current = head
  next = None
  prev = None
```

```
    count = 0

    while (current != None and count < k) :
        next = current.next
        current.next = prev
        prev = current
        current = next
        count = count + 1;


    if(head != None):
        head.next = current

    count = 0
    while(count < k - 1 and current != None ):
        current = current.next
        count = count + 1

    if(current != None):
        current.next = kAltReverse(current.next, k)

    return prev
```

Question 5

Given a linked list and a key to be deleted. Delete last occurrence of key from linked. The list may have duplicates.

Examples:

Input: 1->2->3->5->2->10, key = 2
Output: 1->2->3->5->10def deleteLast(head, x):

```
    temp = head
    ptr = None

    while (temp != None):

        if (temp.data == x):
            ptr = temp

        temp = temp.next

    if (ptr != None and ptr.next == None):
        temp = head
        while (temp.next != ptr):
            temp = temp.next

        temp.next = None

    if (ptr != None and ptr.next != None):
        ptr.data = ptr.next.data
        temp = ptr.next
        ptr.next = ptr.next.next

    return head
```

Question 6

Given two sorted linked lists consisting of **N** and **M** nodes respectively. The task is to merge both of the lists (in place) and return the head of the merged list.

Examples:

Input: a: 5->10->15, b: 2->3->20

Output: 2->3->5->10->15->20

Input: a: 1->1, b: 2->4

Output: 1->1->2->4

```
def sortedMerge(head1, head2):
    c1, c2 = head1, head2
```

```
    if c2.data<c1.data:
        temp = c1
        c1 = c2
        c2=c2.next
        c1.next = temp
        head1=c1

    while c1.next and c2:
        if c1.next.data<=c2.data:
            c1=c1.next
        else:
            temp = c1.next
            c1.next = c2
            c2 = c2.next
            c1.next.next = temp
            c1=c1.next

    if not c1.next and c2:
        c1.next = c2

    return head1
```

Question 7

Given a **Doubly Linked List**, the task is to reverse the given Doubly Linked List.

Example:

Original Linked list 10 8 4 2
Reversed Linked list 2 4 8 10

```
def reverseDLL(head):
    curr=head
    while(curr):
        next_node=curr.next
        curr.next=curr.prev
        curr.prev=next_node
        prev_node=curr
        # print(curr.prev,curr,curr.next)
        curr=next_node
    return prev_node
```

Question 8

Given a doubly linked list and a position. The task is to delete a node from given position in a doubly linked list.

Example 1:

Input:

LinkedList = 1 <--> 3 <--> 4

x = 3

Output:1 3

Explanation:After deleting the node at

position 3 (position starts from 1),

the linked list will be now as 1->3.

Example 2:
Input:

LinkedList = 1 <--> 5 <--> 2 <--> 9

x = 1

Output:5 2 9

```
def deleteNode(head_ref, del_):

    if (head_ref == None or del_ == None):
        return

    if (head_ref == del_):
        head_ref = del_.next
```

```
    if (del_.next != None):
        del_.next.prev = del_.prev

    if (del_.prev != None):
        del_.prev.next = del_.next

    return head_ref

def deleteNodeAtGivenPos(head_ref,n):

    if (head_ref == None or n <= 0):
        return

    current = head_ref
    i = 1


    while ( current != None and i < n ):
        current = current.next
        i = i + 1


    if (current == None):
        return

    deleteNode(head_ref, current)

    return head_ref
```