



Linked List Cycle

Top Coding Questions for Beginners

Given a linked list, determine if it has a cycle in it.

CREATE BY - ATUL KUMAR (LINKEDIN)

  @atulkumarx

Follow up: Can you solve it without using extra space?

URL: <https://leetcode.com/problems/linked-list-cycle/>

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def hasCycle(self, head):
        """
        :type head: ListNode
        :rtype: bool
        """
        if head == None:
            return False
        else:
            fast = head
            slow = head

            while fast != None and fast.next != None:
                slow = slow.next
                fast = fast.next.next
                if fast == slow:
                    break

            if fast == None or fast.next == None:
                return False
            elif fast == slow:
                return True

            return False
```

Reverse Linked List

Reverse a singly linked list.

URL: <https://leetcode.com/problems/reverse-linked-list/>

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def reverseList(self, head):
        """
        :type head: ListNode
        :rtype: ListNode
        """
        if head == None:
            return None
        elif head != None and head.next == None:
            return head
        else:
            temp = None
            next_node = None
            while head != None:
                next_node = head.next
                head.next = temp
                temp = head
                head = next_node

            return temp
```

Delete Node in a Linked List

Write a function to delete a node (except the tail) in a singly linked list, given only access to that node.

Supposed the linked list is 1 -> 2 -> 3 -> 4 and you are given the third node with value 3, the linked list should become 1 -> 2 -> 4 after calling your function.

URL: <https://leetcode.com/problems/delete-node-in-a-linked-list/>

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def deleteNode(self, node):
        """
        :type node: ListNode
        :rtype: void Do not return anything, modify node in-place instead.
        """
        if node == None:
            pass
        else:
            next_node = node.next
            node.val = next_node.val
            node.next = next_node.next
```

Merge Two Sorted Lists

Merge two sorted linked lists and return it as a new list. The new list should be made by splicing together the nodes of the first two lists.

URL: <https://leetcode.com/problems/merge-two-sorted-lists/>

```

# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def mergeTwoLists(self, l1, l2):
        """
        :type l1: ListNode
        :type l2: ListNode
        :rtype: ListNode
        """
        if l1 == None and l2 == None:
            return None
        elif l1 != None and l2 == None:
            return l1
        elif l2 != None and l1 == None:
            return l2
        else:
            dummy = ListNode(0)
            p = dummy

            while l1 != None and l2 != None:
                if l1.val < l2.val:
                    p.next = l1
                    l1 = l1.next
                else:
                    p.next = l2
                    l2 = l2.next
                p = p.next

            if l1 != None:
                p.next = l1

            if l2 != None:
                p.next = l2

            return dummy.next

```

Intersection of Two Linked Lists

Write a program to find the node at which the intersection of two singly linked lists begins.

For example, the following two linked lists:

A: $a1 \rightarrow a2 \searrow c1 \rightarrow c2 \rightarrow c3 \nearrow$

B: $b1 \rightarrow b2 \rightarrow b3$ begin to intersect at node c1.

Notes:

If the two linked lists have no intersection at all, return null. The linked lists must retain their original structure after the function returns. You may assume there are no cycles anywhere in the entire linked structure. Your code should preferably run in $O(n)$ time and use only $O(1)$ memory.

URL: <https://leetcode.com/problems/intersection-of-two-linked-lists/>

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def getIntersectionNode(self, headA, headB):
        """
        :type head1, head1: ListNode
        :rtype: ListNode
        """
        if headA == None and headB == None:
            return None
        elif headA == None and headB != None:
            return None
        elif headA != None and headB == None:
            return None
        else:
            len_a = 0
```

```
len_b = 0

current = headA
while current != None:
    current = current.next
    len_a += 1

current = headB
while current != None:
    current = current.next
    len_b += 1

diff = 0
current = None
if len_a > len_b:
    diff = len_a - len_b
    currentA = headA
    currentB = headB
else:
    diff = len_b - len_a
    currentA = headB
    currentB = headA

count = 0
while count < diff:
    currentA = currentA.next
    count += 1

while currentA != None and currentB != None:
    if currentA == currentB:
        return currentA
    else:
        currentA = currentA.next
        currentB = currentB.next
```

Linked List Cycle II

Given a linked list, return the node where the cycle begins. If there is no cycle, return null.

Note: Do not modify the linked list.

Follow up: Can you solve it without using extra space?

URL: <https://leetcode.com/problems/linked-list-cycle-ii/>


```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def detectCycle(self, head):
        """
        :type head: ListNode
        :rtype: ListNode
        """
        if head == None:
            return head
        else:
            fast = head
            slow = head

            has_cycle = False
            while fast != None and fast.next != None:
                slow = slow.next
                fast = fast.next.next
                if fast == slow:
                    has_cycle = True
                    break

            if has_cycle == False:
                return None

            slow = head
            while fast != slow:
                fast = fast.next
                slow = slow.next

            return slow
```

Palindrome Linked List

Given a singly linked list, determine if it is a palindrome.

Follow up: Could you do it in $O(n)$ time and $O(1)$ space?

URL: <https://leetcode.com/problems/palindrome-linked-list/>

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def isPalindrome(self, head):
        """
        :type head: ListNode
        :rtype: bool
        """
        if head == None:
            return True
        elif head != None and head.next == None:
            return True
        else:
            fast = head
            slow = head
            stack = []
            while fast != None and fast.next != None:
                stack.append(slow.val)
                slow = slow.next
                fast = fast.next.next

            #madam
            if fast != None:
                slow = slow.next

            while slow != None:
                if slow.val != stack.pop():
                    return False
                else:
                    slow = slow.next

            return True
```

Remove Linked List Elements

Remove all elements from a linked list of integers that have value val.

Example Given: 1 --> 2 --> 6 --> 3 --> 4 --> 5 --> 6, val = 6 Return: 1 --> 2 --> 3 --> 4 --> 5

URL: <https://leetcode.com/problems/remove-linked-list-elements/>

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def removeElements(self, head, val):
        """
        :type head: ListNode
        :type val: int
        :rtype: ListNode
        """
        if head == None:
            return head
        elif head != None and head.next == None:
            if head.val == val:
                return None
            else:
                return head
        else:
            dummy = ListNode(0)
            dummy.next = head
            prev = dummy

            while head != None:
                if head.val == val:
                    prev.next = head.next
                    head = prev
                prev = head
                head = head.next

            return dummy.next
```

Remove Duplicates from Sorted Linked List

Given a sorted linked list, delete all duplicates such that each element appear only once.

For example, Given 1->1->2, return 1->2. Given 1->1->2->3->3, return 1->2->3.

URL: <https://leetcode.com/problems/remove-duplicates-from-sorted-list/>

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def deleteDuplicates(self, head):
        """
        :type head: ListNode
        :rtype: ListNode
        """
        if head == None:
            return head
        elif head != None and head.next == None:
            return head
        else:
            lookup = {}
            current = head
            prev = head
            while current != None:
                if current.val in lookup:
                    prev.next = prev.next.next
                else:
                    lookup[current.val] = True
                    prev = current
                current = current.next

            return head
```

Remove Duplicates from Sorted Linked List II

Given a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list.

For example, Given 1->2->3->3->4->4->5, return 1->2->5. Given 1->1->1->2->3, return 2->3.

URL: <https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii/>

```
# Definition for singly-linked list
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def deleteDuplicates(self, head):
        """
        :type head: ListNode
        :rtype: ListNode
        """
        if head == None:
            return head
        else:
            dup_dict = {}
            current = head
            while current != None:
                if current.val in dup_dict:
                    dup_dict[current.val] += 1
                else:
                    dup_dict[current.val] = 1
                current = current.next

            list_values = []
            current = head
            while current != None:
```



```
        if dup_dict[current.val] > 1:
            pass
        else:
            list_values.append(current.val)
            current = current.next
    if list_values == []:
        return None
    else:
        node1 = ListNode(list_values[0])
        head = node1
        for entries in list_values[1:]:
            new_node = ListNode(entries)
            node1.next = new_node
            node1 = new_node

    return head
```

Remove Nth node from End of List

Given a linked list, remove the nth node from the end of list and return its head.

For example,

Given linked list: 1->2->3->4->5, and n = 2.

After removing the second node from the end, the linked list becomes 1->2->3->5.

Note: Given n will always be valid. Try to do this in one pass.

URL: <https://leetcode.com/problems/remove-nth-node-from-end-of-list/>

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def removeNthFromEnd(self, head, n):
        """
        :type head: ListNode
        :type n: int
        :rtype: ListNode
        """
        if head == None:
            return head
        else:
            dummy = ListNode(0)
            dummy.next = head
            fast = dummy
            slow = dummy
            for i in range(n):
                fast = fast.next

            while fast.next != None:
                fast = fast.next
                slow = slow.next

            slow.next = slow.next.next

            return dummy.next
```

CREATE BY - ATUL KUMAR (LINKEDIN)
🐦 @atulkumarx

📌 CODING BUGS 📌 NOTES GALLERY