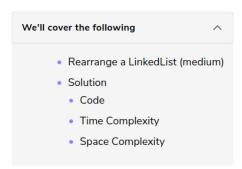


Solution Review: Problem Challenge 2



Rearrange a LinkedList (medium)

Given the head of a Singly LinkedList, write a method to modify the LinkedList such that the **nodes from the second half of the LinkedList are inserted alternately to the nodes from the first half in reverse order**. So if the LinkedList has nodes $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 1$ null, your method should return $1 \rightarrow 6 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 1$

Your algorithm should not use any extra space and the input LinkedList should be modified in-place.

Example 1:

```
Input: 2 -> 4 -> 6 -> 8 -> 10 -> 12 -> null
Output: 2 -> 12 -> 4 -> 10 -> 6 -> 8 -> null
```

Example 2:

```
Input: 2 -> 4 -> 6 -> 8 -> 10 -> null
Output: 2 -> 10 -> 4 -> 8 -> 6 -> null
```

Solution

This problem shares similarities with Palindrome LinkedList. To rearrange the given LinkedList we will follow the following steps:

- We can use the Fast & Slow pointers method similar to Middle of the LinkedList to find the middle node
 of the LinkedList.
- 2. Once we have the middle of the LinkedList, we will reverse the second half of the LinkedList.
- 3. Finally, we'll iterate through the first half and the reversed second half to produce a LinkedList in the required order.

Code

Here is what our algorithm will look like:

```
def reorder(head):
      if head is None or head.next is None:
      slow, fast = head, head
        slow = slow.next
      head second half = reverse(slow) # reverse the second half
      head first half = head
      # rearrange to produce the LinkedList in the required order
while head_first_half is not None and head_second_half is not None:
        temp = head_first_half.next
        head first half.next = head second half
        head_first_half = temp
         temp = head_second_half.next
        head_second_half.next = head_first_half
        head_second_half = temp
      # set the next of the last node to 'None'
if head_first_half is not None:
        head_first_half.next = None
46 def reverse(head):
47 prev = None
      while head is not None:
        next = head.next
        head.next = prev
        prev = head
        head = next
      return prev
56 def main():
57 head = Noo
      head = Node(2)
      head.next = Node(4)
      head.next.next = Node(6)
      head.next.next.next = Node(8)
      head.next.next.next = Node(10)
      head.next.next.next.next = Node(12)
      reorder(head)
      head.print_list()
```

Time Complexity

The above algorithm will have a time complexity of O(N) where 'N' is the number of nodes in the LinkedList.

Space Complexity

The algorithm runs in constant space O(1).

