To implement the size() method for a DoublyLinkedList class without maintaining the size as an instance variable, you can iterate through the list and count the number of nodes. Here's how you can implement it:

**Implementation:**

class DoublyLinkedList:

# Node class for doubly linked list

class Node:

def \_\_init\_\_(self, element):

self.element = element # Data of the node

self.prev = None # Pointer to the previous node

self.next = None # Pointer to the next node

def \_\_init\_\_(self):

# Initialize the doubly linked list with header and trailer sentinels

self.header = self.Node(None) # Sentinel node at the beginning

self.trailer = self.Node(None) # Sentinel node at the end

self.header.next = self.trailer

self.trailer.prev = self.header

def size(self):

# Initialize a counter to keep track of the size of the list

count = 0

# Start from the first real node (right after the header sentinel)

current = self.header.next

# Traverse the list, counting nodes until we reach the trailer sentinel

while current != self.trailer:

count += 1

current = current.next

return count

**Explanation:**

1. **Node Class:** The inner Node class defines a node in the doubly linked list, with pointers for both the previous and next nodes.
2. **Initialization:** The list is initialized with two sentinel nodes: header and trailer. The header sentinel’s next points to the first node, and the trailer sentinel’s prev points to the last node.
3. **Size Method:**
   * We initialize a count variable to track the number of nodes.
   * Starting from the first real node (the node immediately after the header sentinel), we iterate through the list, incrementing the count for each node.
   * The loop stops when we reach the trailer sentinel.
   * The method returns the count, which represents the size of the list.

**Example:**

# Example usage

dll = DoublyLinkedList()

dll.add\_last(1)

dll.add\_last(2)

dll.add\_last(3)

dll.add\_last(4)

print(dll.size()) # Output: 4

This implementation runs in O(n)O(n)O(n) time, where nnn is the number of nodes in the list, since we have to traverse the entire list to count the nodes.