Insert In Sorted Way In A Sorted DLL

solved by	Senan
Platform	GeeksForGeeks
difficulty	Medium
tags	Linked List
language	C++
solved on	@31/10/2024
link	https://www.geeksforgeeks.org/problems/insert-in-sorted-way-in-a-sorted-dll/1
Completion	✓

Intuition

The task is to insert a new node into a sorted doubly linked list such that the list remains sorted after insertion. We start by identifying where the new node should go by traversing the list until we find the right spot based on the sorted order.

Approach

1. **Create a New Node**: Create the new node with the given data `x`.
2. **Handle Empty List**: If the list is empty, return the new node as it will be the head.
3. **Traverse the List**: Iterate through the list until finding a node with a value greater than or equal to `x`. Track the previous node to help with insertion.
4. **Insert the Node**:
    - **At the Head**: If we haven't moved (i.e., `prev` is `NULL`), insert the new node at the head, set its next pointer to the current head, and update the previous pointer of the head to the new node.
    - **In the Middle or End**: Insert the new node between `prev` and `temp`. Update the next and previous pointers to connect `newNode` properly.
5. **Return the Head**: Return the head of the modified list.

Complexity

Time Complexity:

The time complexity is **O(n)**, where **n** is the number of nodes in the list. In the worst case, we may need to traverse the entire list to find the correct insertion point.

Space Complexity:

The space complexity is **O(1)** as we are only using a constant amount of extra space for the new node.

Code

```
class Solution {
  public:
   Node* sortedInsert(Node* head, int x) {
        Node* newNode = new Node();
        newNode->data = x;
        newNode->next = NULL;
        newNode->prev = NULL;

        if (head == NULL) {
            return newNode;
        }

        Node* temp = head;
        Node* prev = NULL;

        while (temp && temp->data < x) {
            prev = temp;
            temp = temp->next;
        }

        if (prev == NULL) {
            newNode->next = head;
            head->prev = newNode;
            return newNode;
        } else {
            prev->next = newNode;
            newNode->prev = prev;
            if (temp) {
                newNode->next = temp;
                temp->prev = newNode;
            }
            return head;
        }
    }
};
```