Doubly Linked List

Wednesday, 24 November 2021 8:32 PM

Doubly Linked Lists

- Given a LL & a ref. of a node. You need to insert a node before it!

TC: O(N)

- We need the ptr to previous node!

- Given a LL. Traverse it backwards..

- Reverse it first!
- No way to go to prev node.

- Given a LL. Given a ptr to a node. Can you delete that node?

O(N)

- No way to go to prev node.

- We do not have prev ptr in DLL.
- Left next & prev ptr

Node struct for DLL.

```
class Node {
    int data;
    Node next;
    Node prev;
}
```

- Delete

- Traversing

Operations

- Add a node at head

- Create a new node & update
  - new_node.next = head;
  - new_node.prev = NULL;
  - if(head != NULL) head.prev = new_node;
  - head = new_node;

O(1)

- Add a new node before given node

O(1)

- Create a new node
- new_node.next = given_node;
- if(given_node != null) given_node.prev = new_node;
- new_node.prev = given_node.prev;
- if(new_node.prev != null) new_node.prev.next = new_node;
- given_node.prev = new_node;

- Given a ref. of a node. Delete that node.

```
if (head == del)
    head = head.next;
if( del.next != NULL)
    del.next.prev = del.prev
if( del.prev != NULL)
    del.prev.next = del.next;
free(del);
```

O(1)

Given a sorted LL & an int. x.
Check if 2 values exist in DLL st their sum == x.

```
l = head;
r = getTail(head);
while ( l != r && l.next != r) {
    if( l.data + r.data == x) {
        l = l.next;
        r = r.prev;
    }
    else if( l.data + r.data > x) {
        r = r.prev;
    }
    else {
        l = l.next;
    }
}
return false;
```

10 + 40 = 50 < 58

20 + 40 = 60 > 58

20 + 35 = 55 < 58

Not 30 = 43 = = 58

TC: O(N)

SC: O(1)

3L < 58

50 < 58

57 < 58

53 > 58

53 > 58

58 == 58

Delete duplicates from sorted DLL.

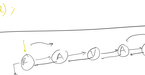
```
curr = head
while (curr != NULL && curr.next != NULL) {
    if( curr.data == curr.next.data) {
        delete(head, curr.next);
    }
    else {
        curr = curr.next;
    }
}
```

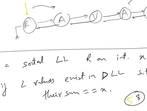
TC: O(N)

SC: O(1)

LRU Cache

LRU Cache → [Least Recently Used]





TC per op

O(1)