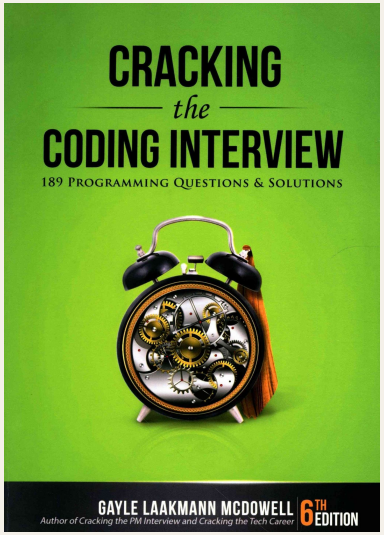
Primer on Linked Lists

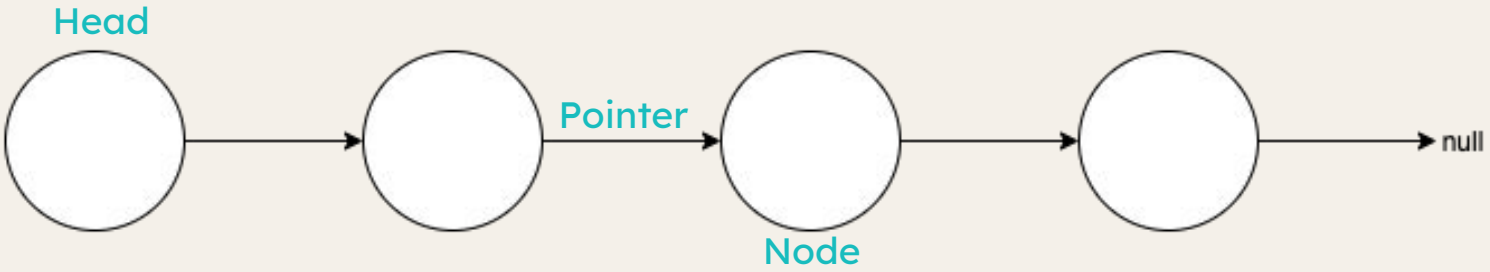
Anindita Ghosh

Materials



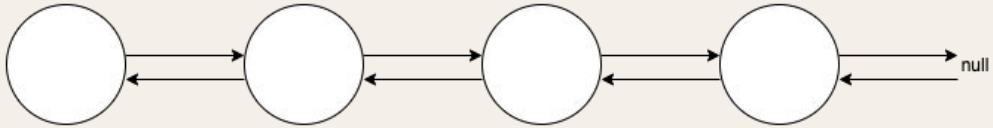
Brush up - 30 days of code [Hackerrank]
Learn - Cracking the Coding Interview
Practice - Leetcode

What are linked lists?

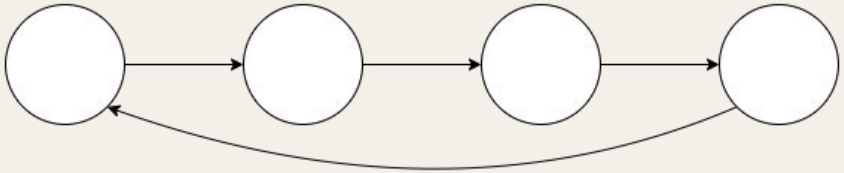


Variations on LL

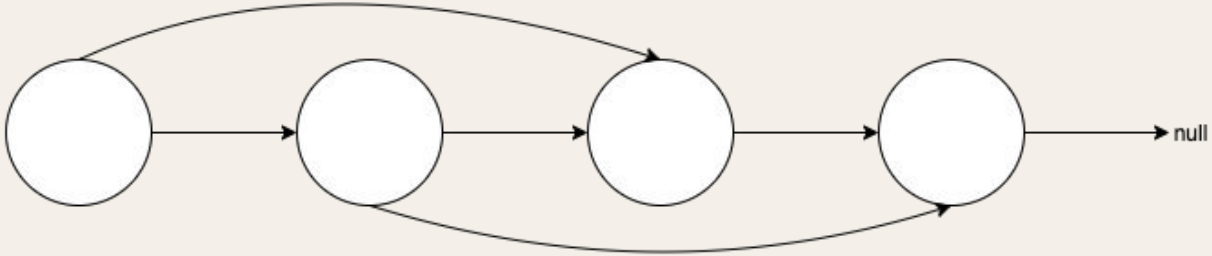
Doubly Linked



Cyclic



Multiple Pointers

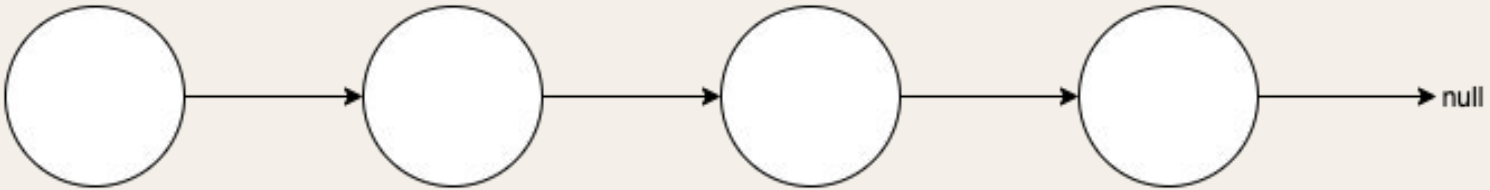


Implementation

```
public class ListNode{
    int val;
    ListNode next;

    ListNode(){}

    ListNode(int val){this.val = val;}

    ListNode(int val, ListNode next){
        this.val = val;
        this.next = next;
    }
}
```

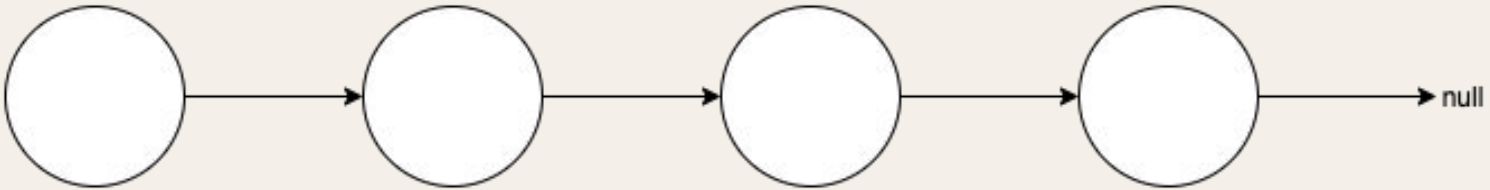
Operations: Contains

```
public boolean contains(ListNode head, int val){
    ListNode curr = head;

    while(curr != null){
        if(curr.val == val) return true;
        curr = curr.next
    }

    return false;
}
```



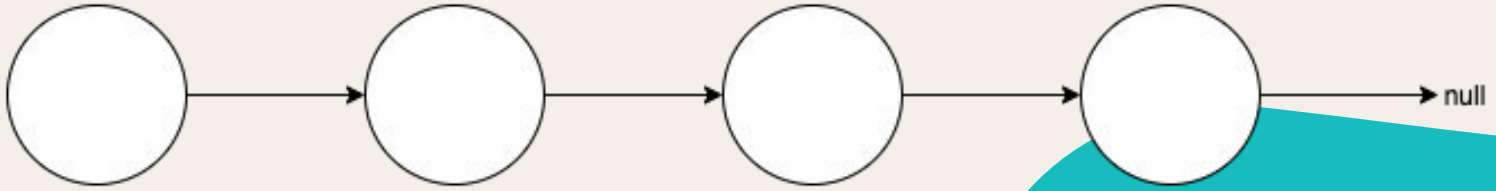
Operations: Add

```
public boolean add(ListNode head, ListNode node){
    ListNode curr = head;

    while(curr.next != null){
        curr = curr.next
    }

    curr.next = node
    return true;
}
```



Operations: Delete

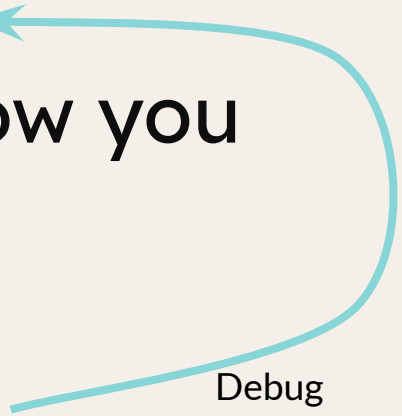
```
public ListNode delete(ListNode head, int val){
    ListNode curr = head;
    ListNode res = null;

    while(curr.next != null){
        if(curr.next.val == val){
            res = curr.next
            curr.next = curr.next.next
            break;
        }
        curr = curr.next
    }

    return res
}
```



All operations involve traversing the linked list and manipulating pointers

Answering Process

1. Define your linked list structure
2. Voice your idea of what to do
   - Use specific examples to show you understand the problem
3. Code it up
4. Go line by line on one example



State time and space complexity

Idea
Pseudocode
Actual Code

Problem:
Given the head of a singly linked list, return the middle node of the linked list. If there are two middle nodes, return the **second middle node**

Idea
Pseudocode
Actual Code

We're given the head - Need to traverse linked list
Middle - Count the number of elements & divide by 2 as int

2nd middle???

Division to int is a floor operation

1 -> 2 -> 3 -> 4 -> 5 5 elems / 2 = 2 return 3

1 -> 2 -> 3 -> 4 4 elems / 2 = 2 return 3

The node we want is n/2 where n is num elems

Problem:
Given the head of a singly linked list, return the middle node of the linked list. If there are two middle nodes, return the **second middle node**

Idea
Pseudocode
Actual Code

```
curr = head
len = 0

while(not at end){
        len++
        curr = curr.next
}

res = head

for(int i until at len/2){
        res = res.next
}

return res
```

Problem:
Given the head of a singly linked list, return the middle node of the linked list. If there are two middle nodes, return the **second middle node**

Idea
Pseudocode
Actual Code

```
public ListNode middle(ListNode head){
    ListNode curr = head;
    int len = 0;

    while(curr != null){
        len++;
        curr = curr.next;
    }

    ListNode res = head;
    for(int i=0; i < len/2; i++){
        res = res.next;
    }

    return res;
}
```

1 -> 2 -> 3 -> 4 -> 5

1 -> 2 -> 3 -> 4

Problem:
Given the head of a singly linked list, return the middle node of the linked list. If there are two middle nodes, return the **second middle node**

Problem:
Given the head of a sorted linked list, delete all duplicates such that each element appears only once.
Return the linked list sorted as well.

Idea
Pseudocode
Actual Code

Sorted so in order

1 -> 2 -> 2 -> 2-> 3 -> 3 -> 4 -> 4

If pointing to node with same value, find first node with different value and point to it

Need to keep track of first one & first diff val; 2 pointers!

Return linked list, so need to return head. Will head change? It shouldn't so can just return input variable

What about all duplicates? 1 -> 1 -> 1 -> 1

Need to return 1; 2nd pointer will reach null so need to set next to null

Problem:
Given the head of a sorted linked list, delete all duplicates such that each element appears only once.
Return the linked list sorted as well.

Idea
Pseudocode
Actual Code

```
prev = head
curr = head.next

while(neither are null){
        if(prev.val == curr.val) curr = curr.next
        else {
                prev.next = curr
                curr = curr.next
                prev = prev.next
        }
}

prev.next = curr
return head
```

Problem:
Given the head of a sorted linked list, delete all duplicates such that each element appears only once.
Return the linked list sorted as well.

```
public ListNode removeDuplicates(ListNode head){
    ListNode prev = head;
    ListNode curr = head.next;

    while(curr != null && prev != null){
        if(prev.val == curr.val){
            curr = curr.next;
        } else {
            prev.next = curr;
            curr = curr.next;
            prev = prev.next;
        }
    }

    prev.next = curr;
    return head;
}
```

Idea
Pseudocode
Actual Code

Problem:
Given the head of a sorted linked list, delete all duplicates such that each element appears only once.
Return the linked list sorted as well.

Things to keep in mind

- Early stopping cases? Head is only elem or head is null?
- Will there be any chance of a new head?
- How many variables need to be tracked? One or two?
- Which pointers do you move?
- Which pointers do you reset?
- Which variables may reach a premature null?
- What cases do you have to be careful of? Edge cases and general case?

Further Problems

Given the head of a singly linked list, reverse the list, and return the reversed list.

1 -> 2 -> 3 -> 4 return 1 <- 2 <- 3 <- 4

Given the head of a singly linked list, remove every 2nd element and return the modified list.

1 -> 2 -> 3 -> 4 -> 5 -> 6 return 1 -> 3 -> 5

Given the head of a singly linked unsorted list, return the first duplicate element. Can you do this without using any additional data structures?

1 -> 5 -> 2 -> 3 -> 2 -> 5 return 2