→ 7/8 → 87.5% vs 100% ✓

→ Golang mid-term ✓

→ Mid-term (70% → 55% ✓)

10/20 ✗

→ TBD → (F)

{ 2nd March → Final Exam }

-> Binary Search

-> Range Query (PS)

-> OOP

-> Hashmap/Set (usage)

F/D → C-

↑ Re-exam

Basic (easy)

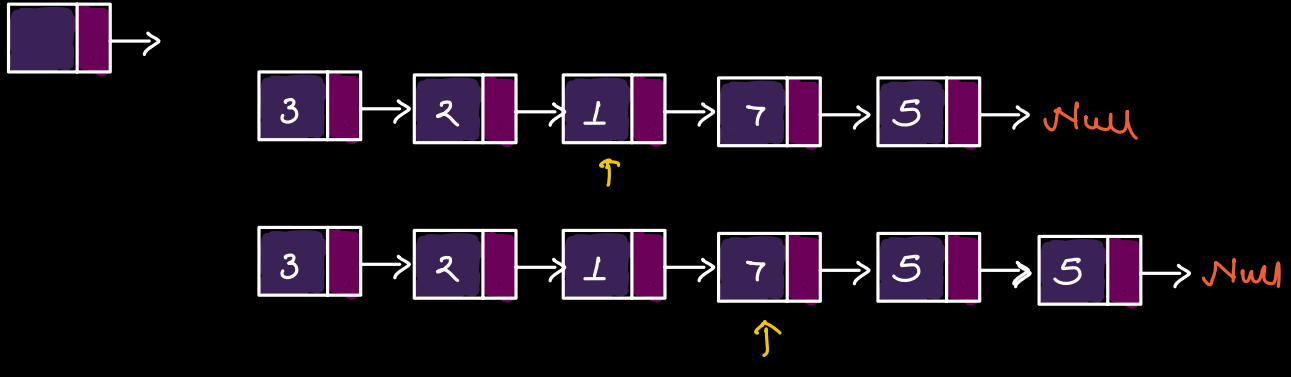
[C-, A⁹]

70.5%

70%

Rahul ← Nischal

Q Given a LL. Return the mid-node of the LL.

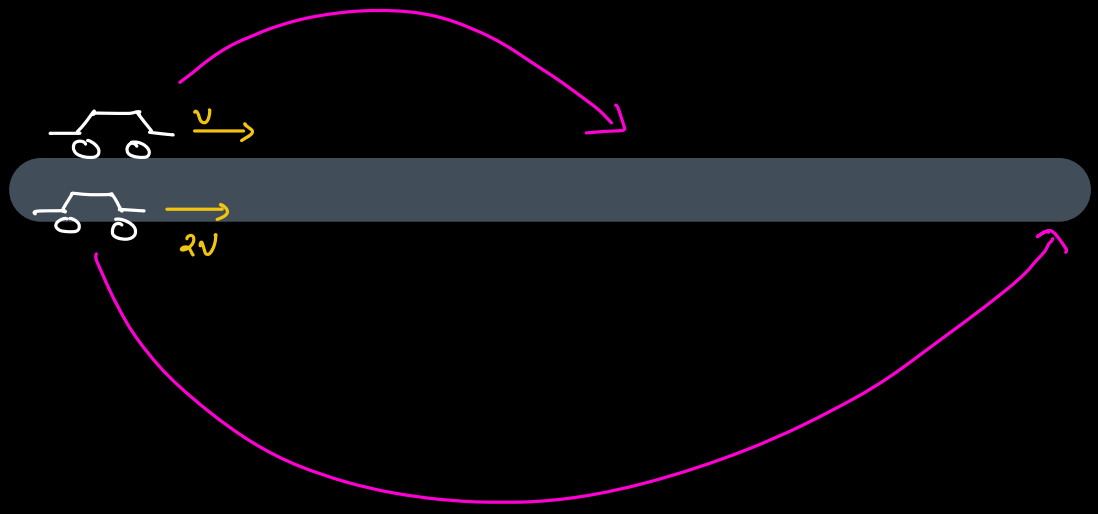


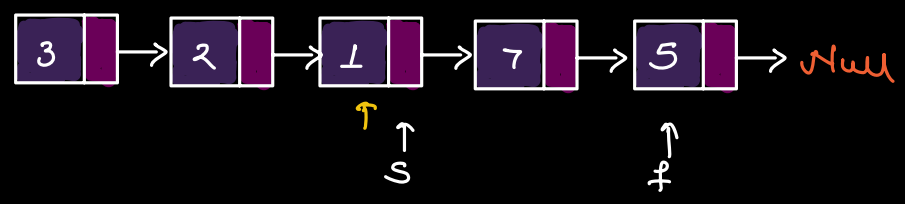
→ Find the length → l

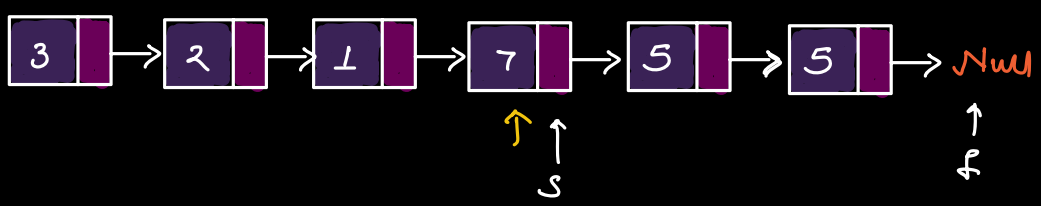
→ Iterate l/2 times

TC: O(N)

SC: O(1)

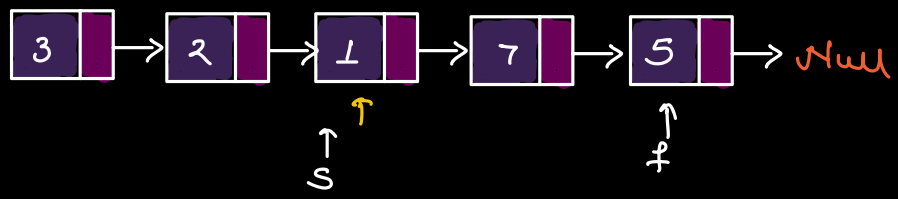


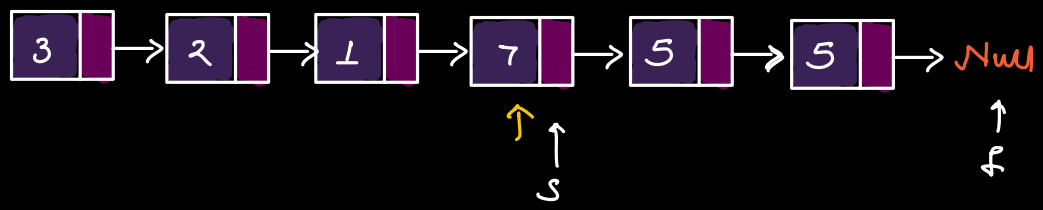





```
Node getMid(Node head){
    Node slow = head, fast = head;
    while(fast != null && fast.next != null){
        slow = slow.next;
        fast = fast.next.next;
    }
    ret slow;
}
```

TC: O(N) SC: O(1)





Edge Case

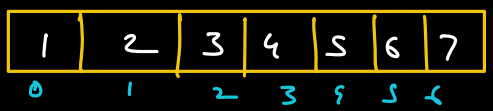
- → Null LL
- → LL of 1, 2 or 3 nodes
- → Odd & even length

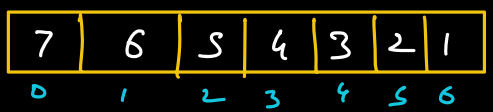
+

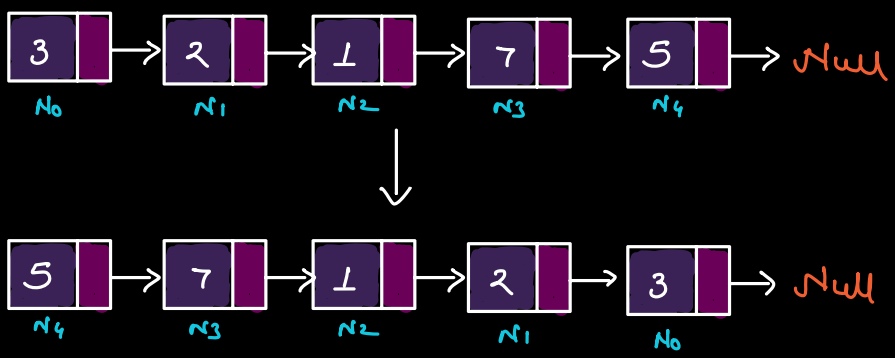
Question specific edge cases

Q Given a LL. Reverse it.

```
Class Node {
    int data;
    Node next;
}
```



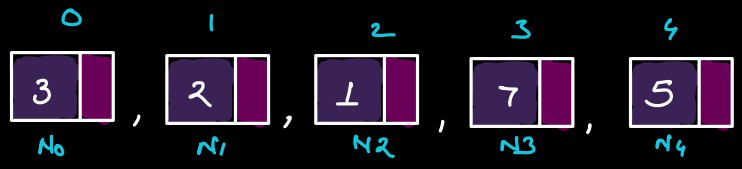




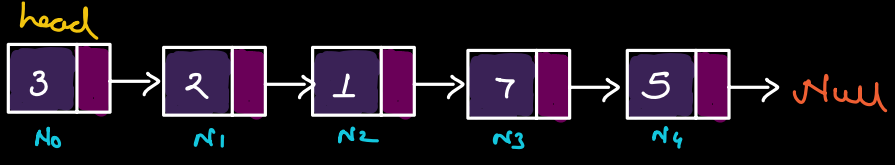
ArrayList<Node> list = new ArrayList<Node>();

TC: O(N)
SC: O(N)

→ Insert all nodes in list → O(N)

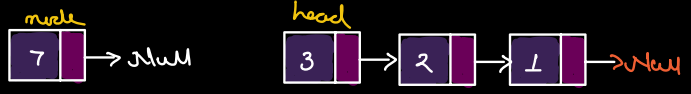


→ Iterate from index N-1 to 0 & create reversed LL
↳ O(N)

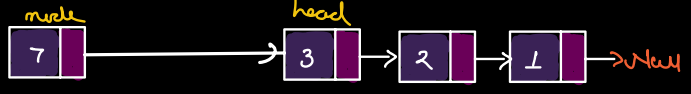


Null
head2

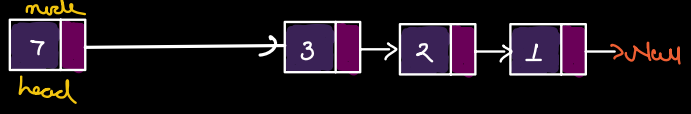
Node insertAtStart (Node head, Node node) {



node.next = head;

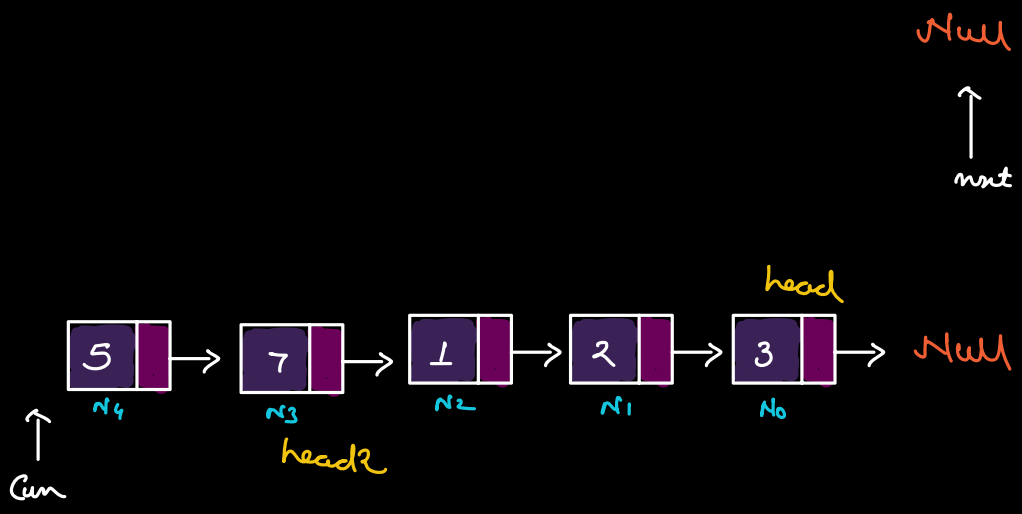


head = node;



ret head;

}

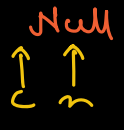


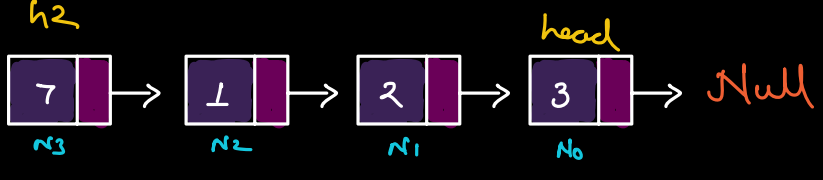

```
Node reverse (Node head){
    Node cur = head, n = head;
    Node head2 = null;
    While (cur != null){
        n = cur.next;
        head2 = insertAtStart (head2, cur);
        cur = n;
    }
    ret head2;
}
```

TC: O(N)

SC: O(1)

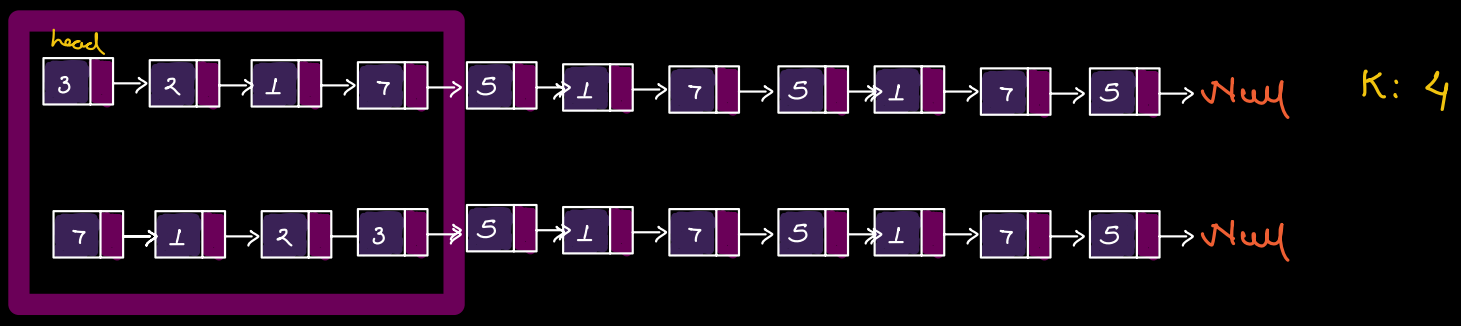
↳ in-place



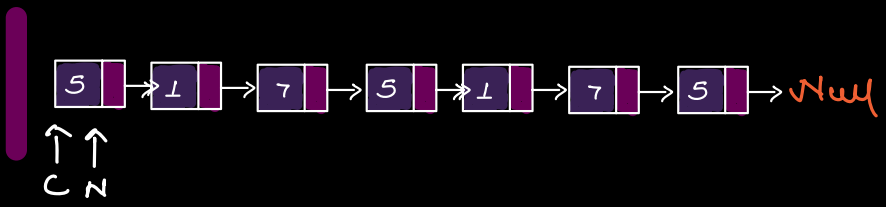


* HW : Reverse using recursion

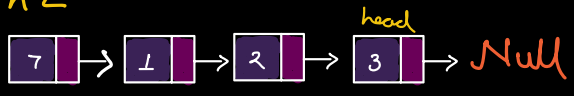
Q Given a LL. Reverse first K nodes of the LL



K = 4

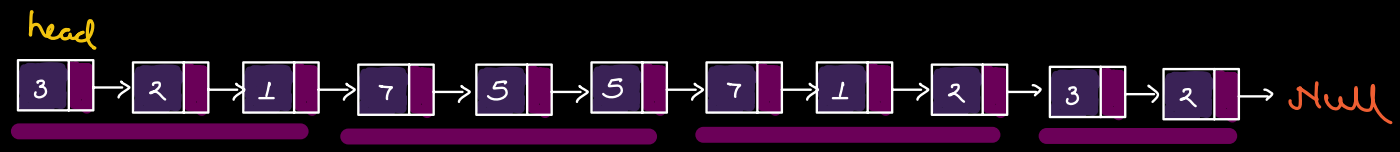


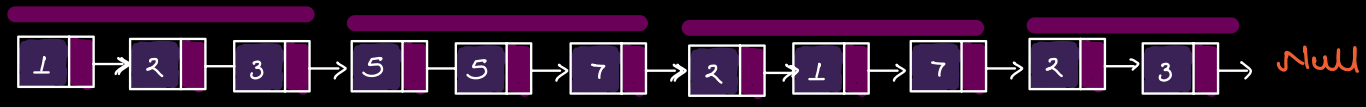
h2

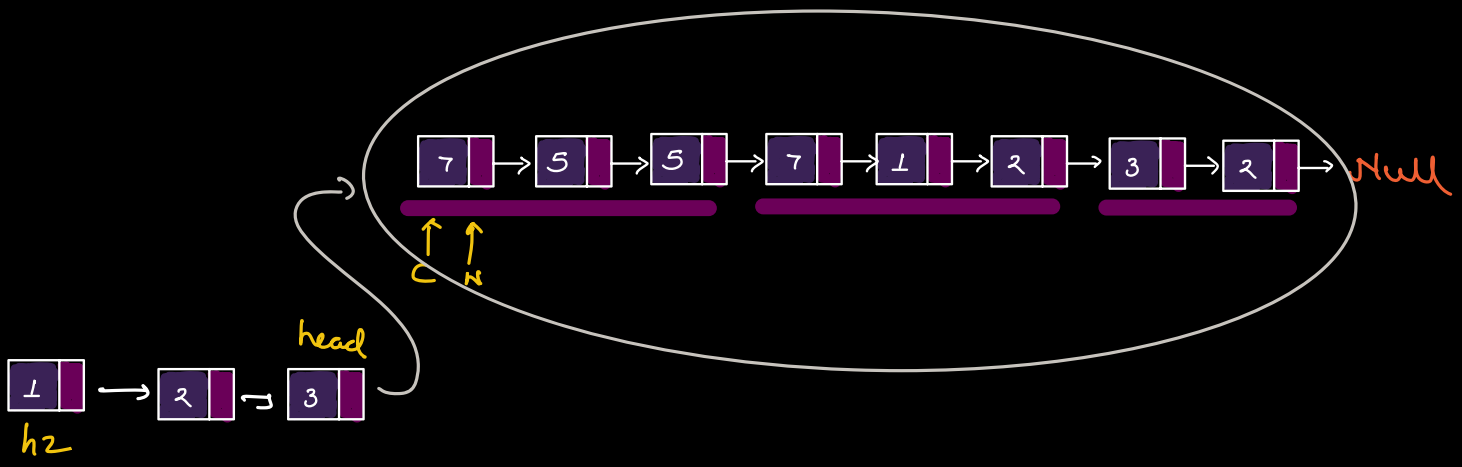



```
Node reverse (Node head, int K){
    Node cur = head, n = head;
    Node head2 = null;
    while (cur != null && K > 0){
        n = cur.next;
        head2 = insertAtStart(head2, cur);
        cur = n;
        K--;
    }
    head.next = cur;
    ret head2;
}
```

Q Given a LL. Reverse every sub-list of size K.








```
Node reverseInKGroups (Node head, int K) {

    {
        Reverse first K nodes
        While ( ) {

        }

    head.next = reverseInKGroups(curr, k)
    ret h2;

}
```