Sorting and Detecting Loop

Rajat Sharma
sharath r
Saurabh Ruikar
Khushi Raj
Subhashini
Vishal Mosa
Akansh Nirmal
Madhan Kumar M S
Pankaj
Rajendra
Purusharth A
Vasanth
Sneha L
suyash gupta
Gowtham
Bhaveshkumar
Sanket Giri
Gagan Kumar S
Sumit Adwani
Shradha Srivastava
Abhishek Sharma

Middle of the Linked List

Merge two sorted linked lists

Merge Sort

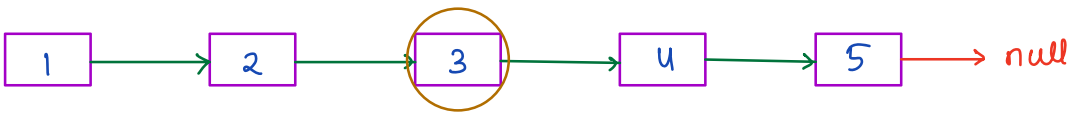
Circular Linked List

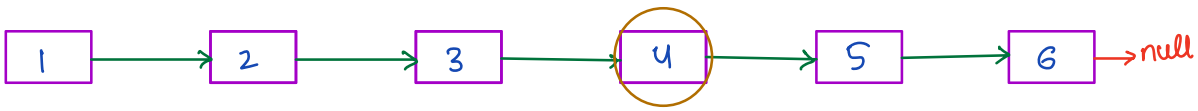
Current PSP

63 % → 70 %.

Middle of the Linked List

Find the middle element in the linked list





Idea from previous class

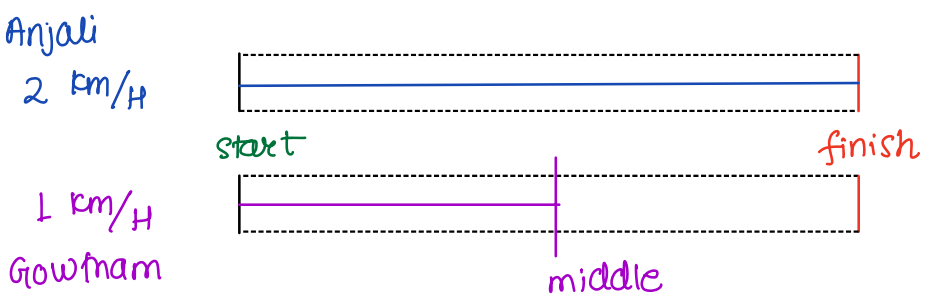
⟶ Find the size of linked list — N

⟶ Traverse to $n/2^{th}$ index

Personal Advice

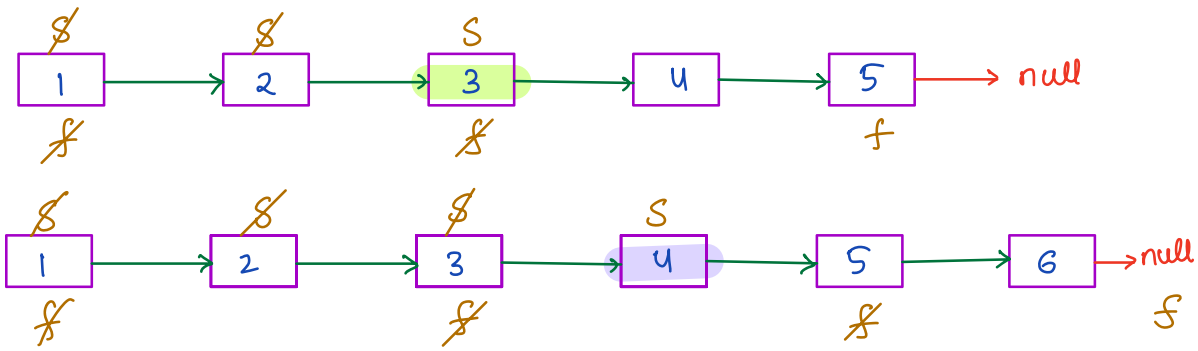
Always use above to calculate mid unless you are in an actual interview.

Idea 2



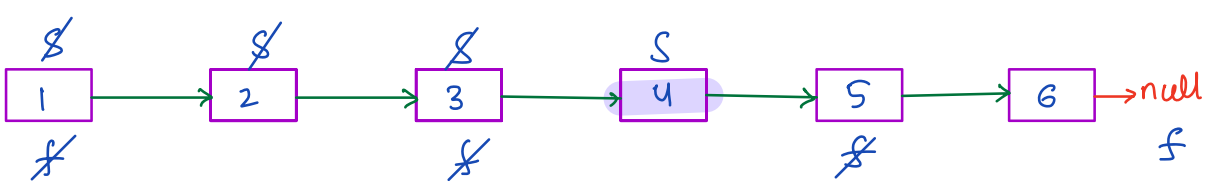
Slow // 1 jump

fast // 2 jumps



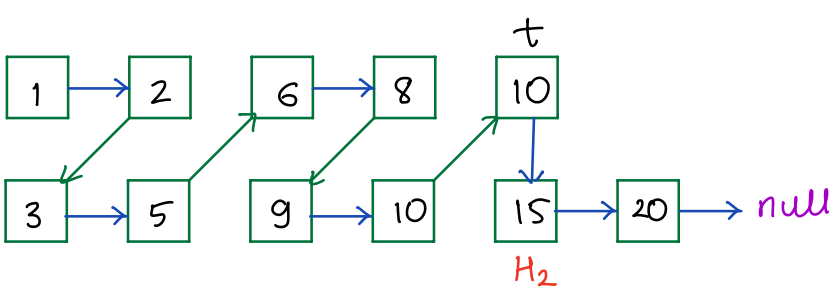
Pseudocode

```
Node   getMiddle ( Node  head ) {
        slow = head
        fast = head

        while ( fast != null && fast.next != null) {
                slow = slow.next
                fast = fast.next.next
        }

        return slow
}
```



Merge two sorted linked lists

Q → Merge two sorted lists into one sorted list.



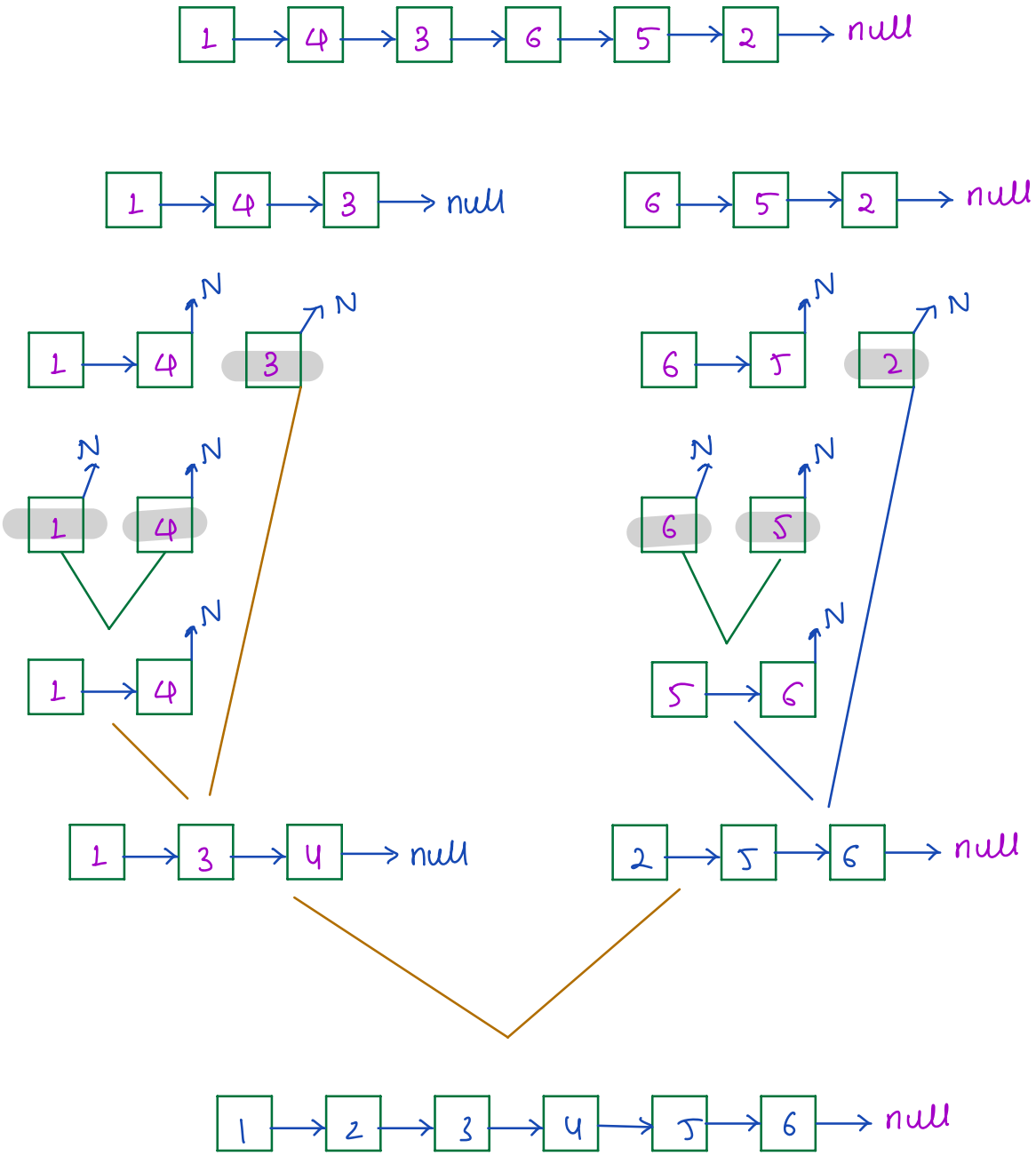

```
Node merge (head1, head2) {
    if (head1 == null) return head2
    if (head2 == null) return head1

    head = null

    if (head1.data < head2.data) {
        head = head1
        head1 = head1.next
    } else {
        head = head2
        head2 = head2.next
    }

    temp = head
    n1 = head1
    n2 = head2
```

```
while ( h1 != null && h2 != null ) {
    if (h1.data < h2.data ) {
        temp.next = h1
        h1 = h1.next
    }
    else {
        temp.next = h2
        h2 = h2.next
    }
    temp = temp.next
}

if ( h1 == null )  temp.next = h2
if ( h2 == null )  temp.next = h1

return head
}
```

TC: O(N + M)

SC: O(1)

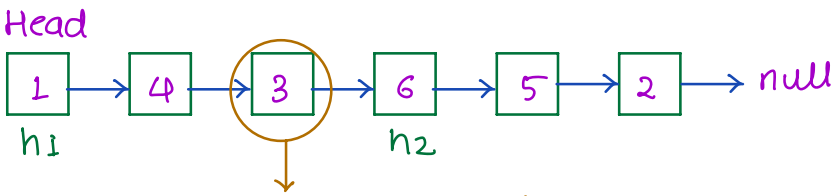
Merge Sort


1 4 3 6 5 2 null
1 4 3 null
6 5 2 null
N
1 4
3
6 5
2
1
4
6
5
1 4
5 6
1 3 4 null
2 5 6 null
1 2 3 4 5 6 null

Pseudocode

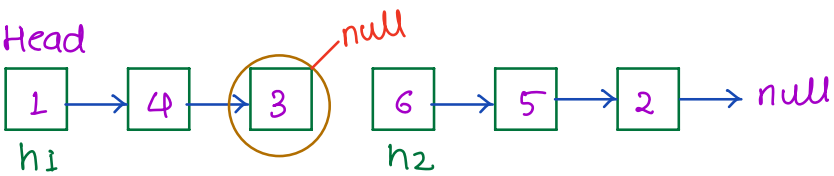
Node

```
mergeSort (head) {
    if (head == null || head.next == null) {
        return head
    }
```




```
    mid = getMiddle(head)
    h1 = head
    h2 = mid.next
    mid.next = null
```

} TC : O(N)

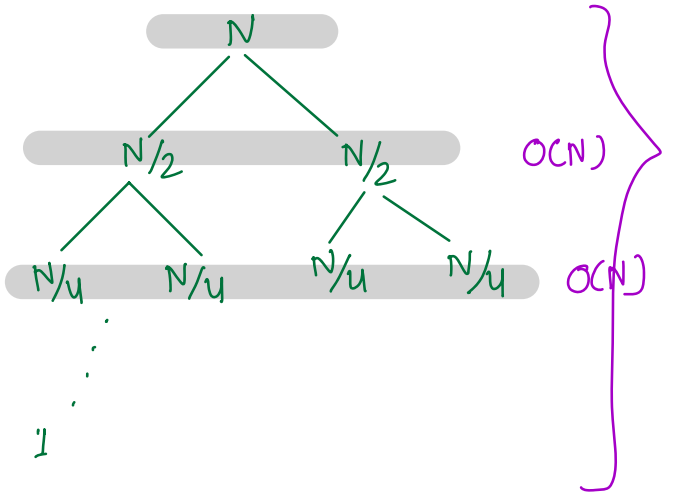



```
    // sort h1 separately
    h1 = mergeSort(h1)
    // sort h2 separately
    h2 = mergeSort(h2)

    return merge(h1, h2)
```

} O(N)

```
}
```

Break 22:25



levels = log N

TC: O(N log N)

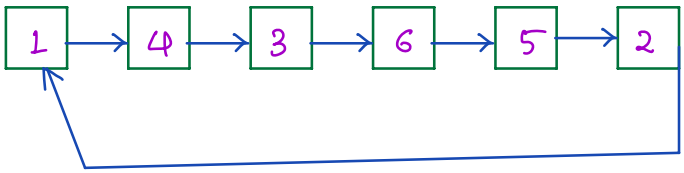
SC: O(log N)

should we do quick sort on LL ?

NO → since merge sort is space optimized for LL

ie. Merge sort is already inplace for LL

Circular Linked List



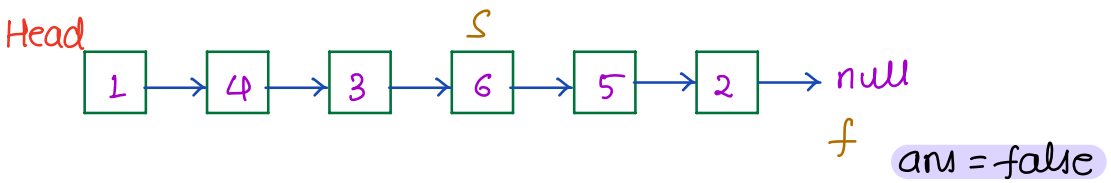
1> which node should be the head ?

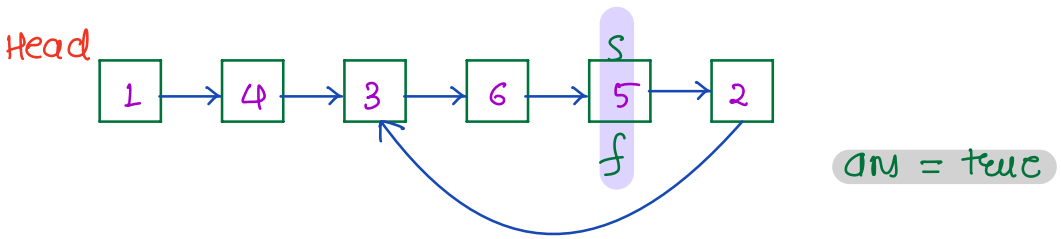
Any node can be

2> How to travel if there is no null node ?

```
temp = head

while (temp.next != head) {
    :
    :
}
```

Q> Check if the given linked list has a cycle





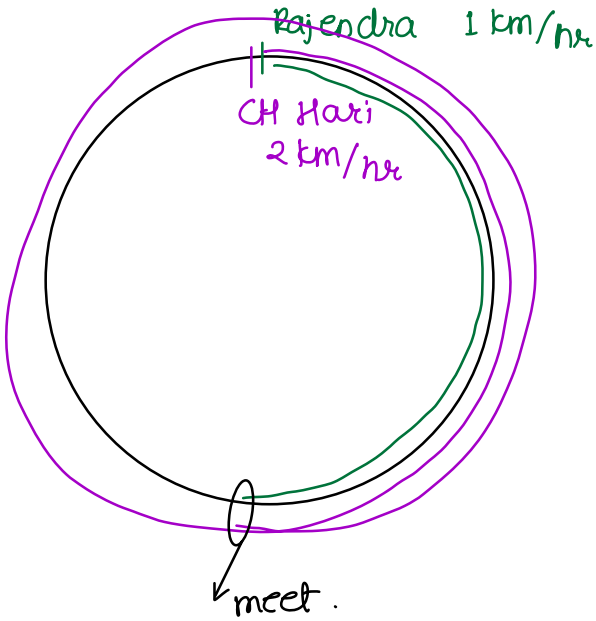
Bruteforce

Use hashset to save the node itself {! node.data}
If there is a duplicate return true
if you reach null return false.

```
Set
while (temp != null) {
        if (temp is present in set) return true
        set.add(temp)
        temp = temp.next
}
return false
```

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

Optional → override equals & hashcode.



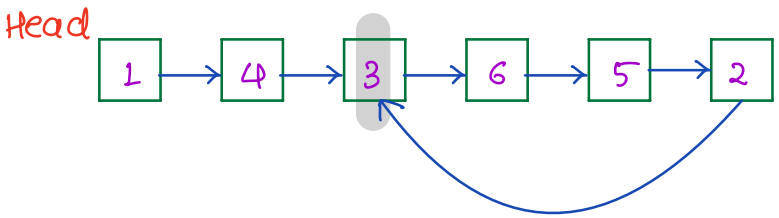

```
bool hasCycle ( Node head ) {
        slow = head
        fast = head

        while ( fast != null && fast.next != null ) {
                slow = slow.next
                fast = fast.next.next
                if ( slow == fast ) return true
        }

        return false
}
```

TC : O(N)

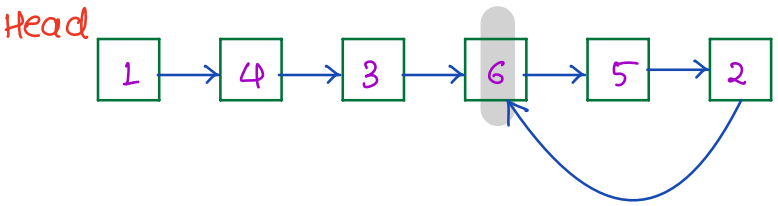
SC : O(1)

compares #adr of slow & fast

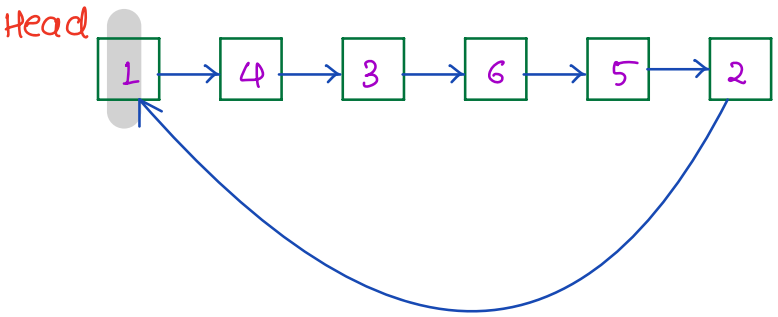
Q → Given a linked list with cycle, find the start node of the cycle ? ***



ans = 3



ans = 6

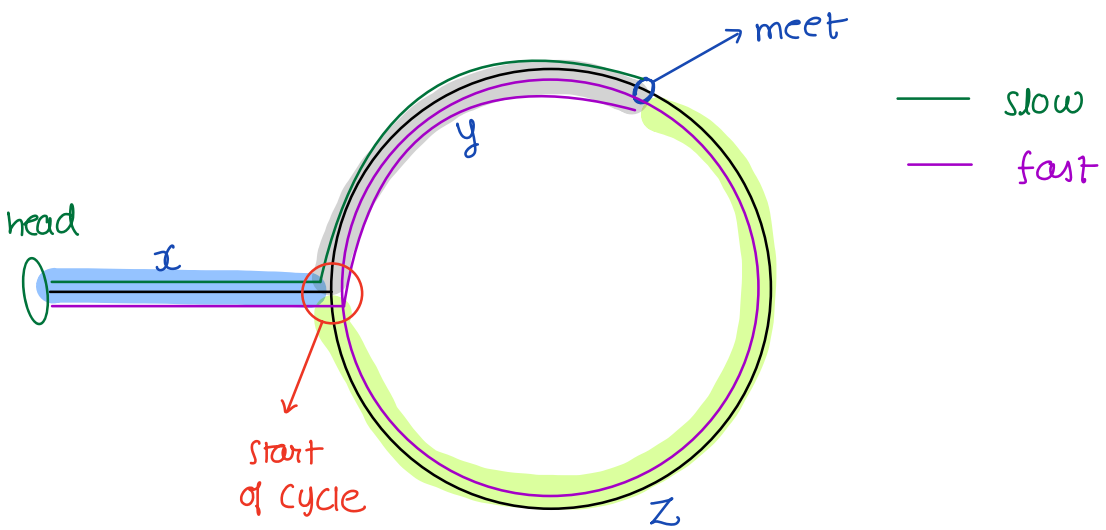


ans = 1

Brute force

use hashset to save the node itself {! node.data}

If there is a duplicate → start of cycle



distance travelled

slow = x+y

fast = x+y+z+y

2* slow = fast

$2x + 2y = x + 2y + z$

$2x = x + z$

$x = z$

---

slow = head

fast = head

```
while (fast != null && fast.next != null) {
    slow = slow.next
    fast = fast.next.next
    if (slow == fast) break
}
```

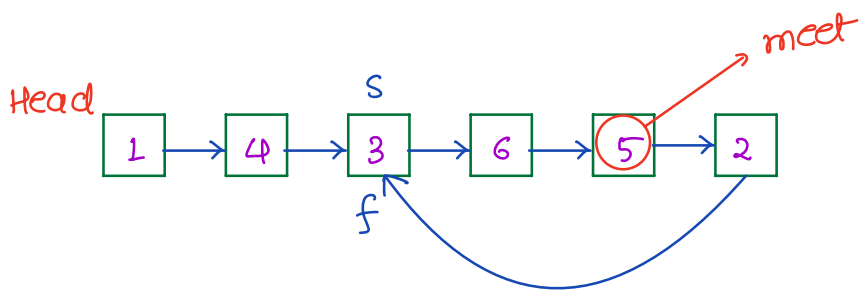
TC: O(N)

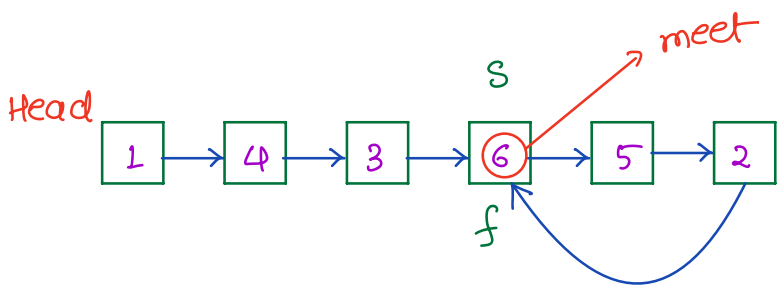
slow = head

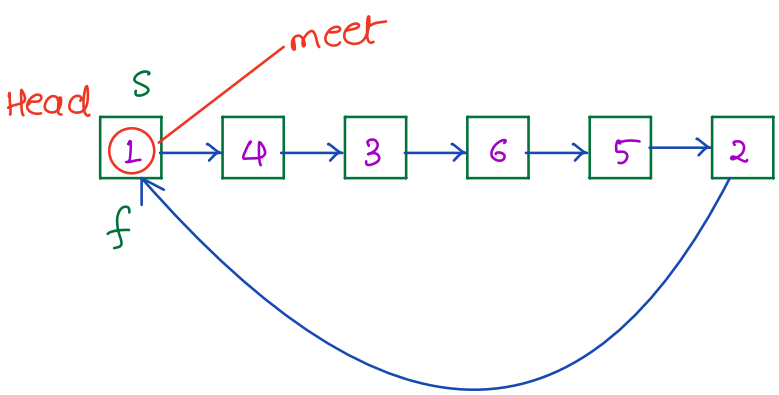
SC : O(1)

```
while (slow != fast) {
    slow = slow.next
    fast = fast.next
}
```

return slow







Doubt Session