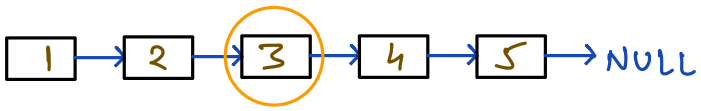
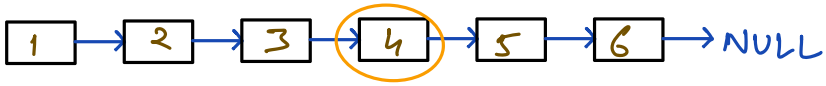
Welcome ☺

Agenda: Slow/Fast Pointers
Circular L.L
1-2 ques's.

Q Find middle element in L.L



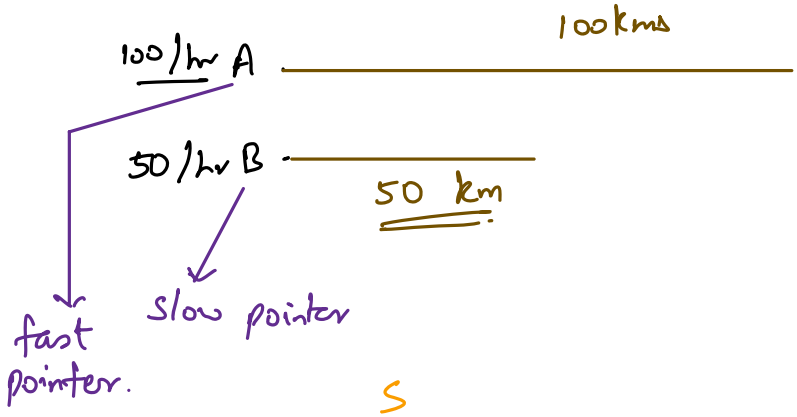


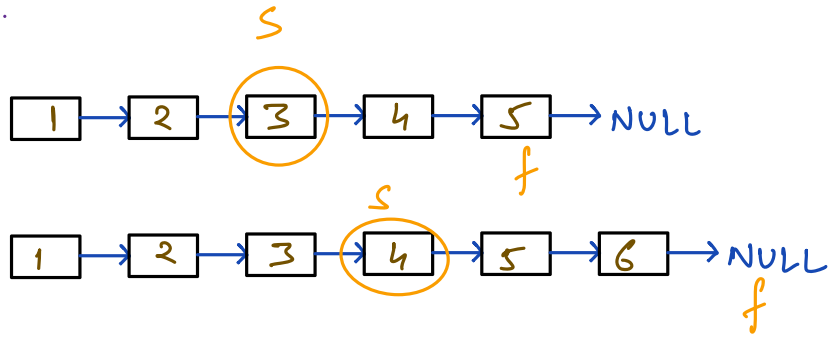
Sol 1 Middle element in an array → $A[N/2]$

1) Find length of L.L → O(N)
2) Traverse till middle = length/2 → O(N/2)

T.C ⇒ O(N) S.C ⇒ O(1)

Sol 2 **Slow & Fast Pointer.**



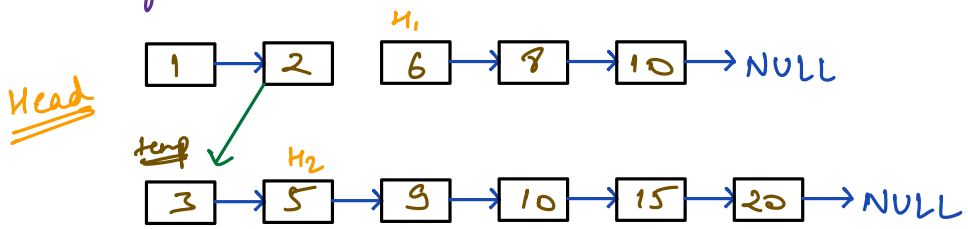


Code

```
s = head        f = head
while (f != NULL AND f.next != NULL)
{
    s = s.next
    f = f.next.next
}
return s
```

T.C ⇒ O(N)
S.C ⇒ O(1)

Q Merge two sorted L.L into one sorted L.L S.C = O(1)



Code

1. NULL check.

```
if (H1 == NULL)
    return H2
if (H2 == NULL)
    return H1
```

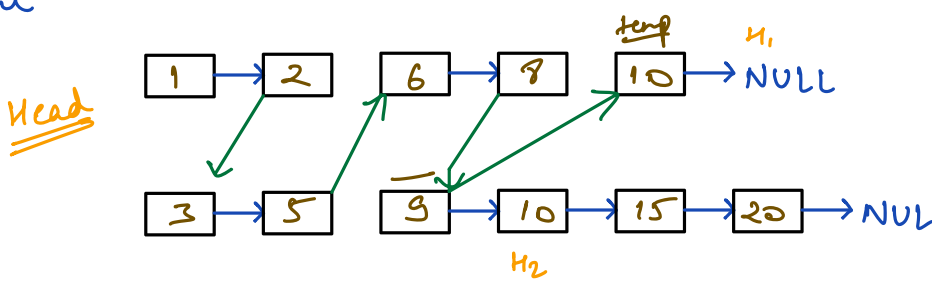
2. Assign Head.

```
if (H1.data ≤ H2.data)
{
    Head = H1
    H1 = H1.next
}
else {
    Head = H2
    H2 = H2.next
}
```

temp = Head.

3. while (H_1 != NULL && H_2 != NULL)

```
{
    if ( H1.data <= H2.data )
    {
        temp.next = H1
        H1 = H1.next
        temp = temp.next
    }
    else
    {
        temp.next = H2
        H2 = H2.next
        temp = temp.next
    }
}
```



4. Handle remaining nodes

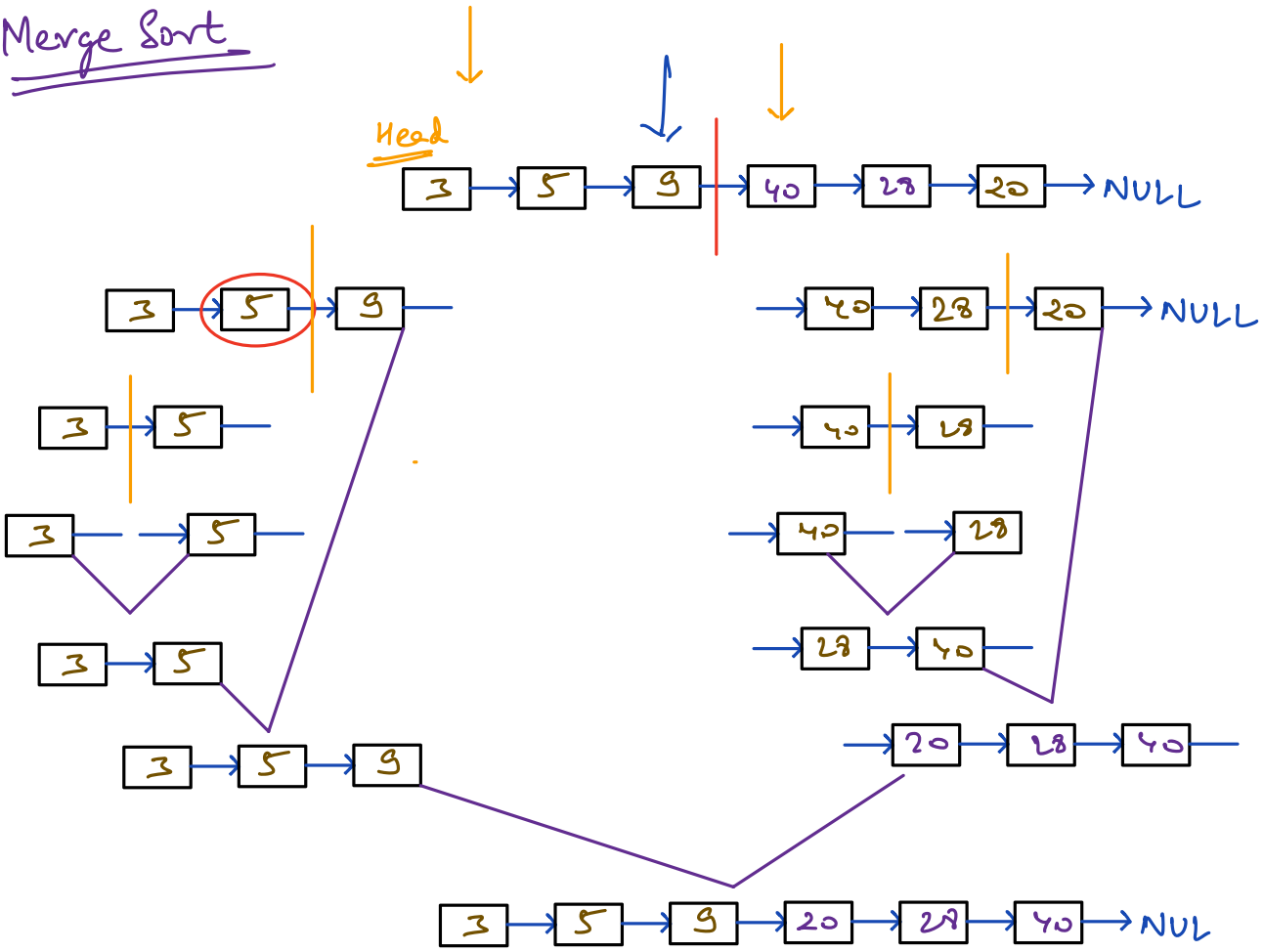
```
if (H1 == NULL)
{
    temp.next = H2
}
else
    temp.next = H1
```

T.C ⇒ O(N+M)

S.C ⇒ O(1)

return Head.

Merge Sort



Code

```
Node MergeSort (Head)
{
    if (Head == NULL || Head.next == NULL)
        return Head.
    mid = getMiddle (Head)  → O(N)
    H1 = Head.
    H2 = mid.next
    mid.next = NULL
    MergeSort (H1)
    MergeSort (H2)
    return Merge (H1, H2)  → O(N) → Conquer.
}
```

Divide

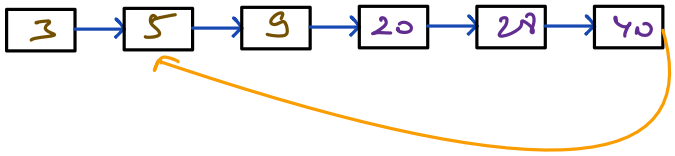
T.C → O(NlogN)

S.C → O(logN)

↓

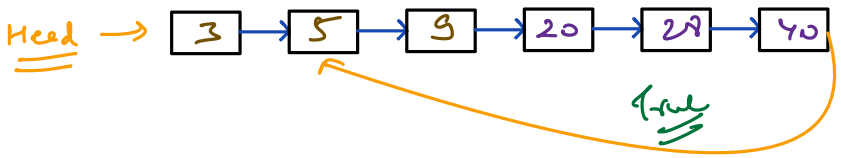
recursion stack.

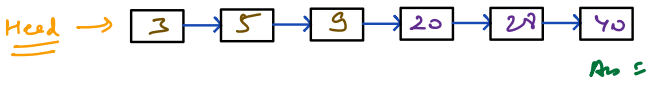
Circular Linked List



1) Any node can be Head.
2) There is no tail node.

Q) Check if the given LL has a cycle ? S.C = O(1)

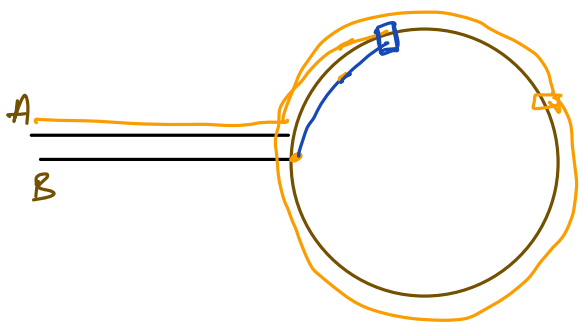




Ans = false

Soln 1) Use hashset.
2) Whenever duplicate found, that is my cycleStart.

Sol²



s = head f = head

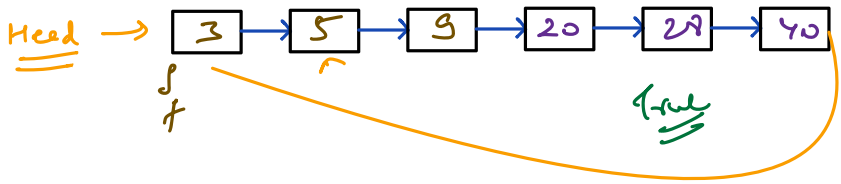
Code

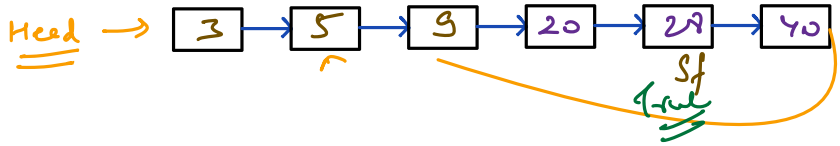
```
while (f != NULL AND f.next != NULL)
{
    s = s.next
    f = f.next.next
    if (s == f) return True.
}
return false.
```

~~S.C~~ ⇒ O(N)
S.C ⇒ O(1)

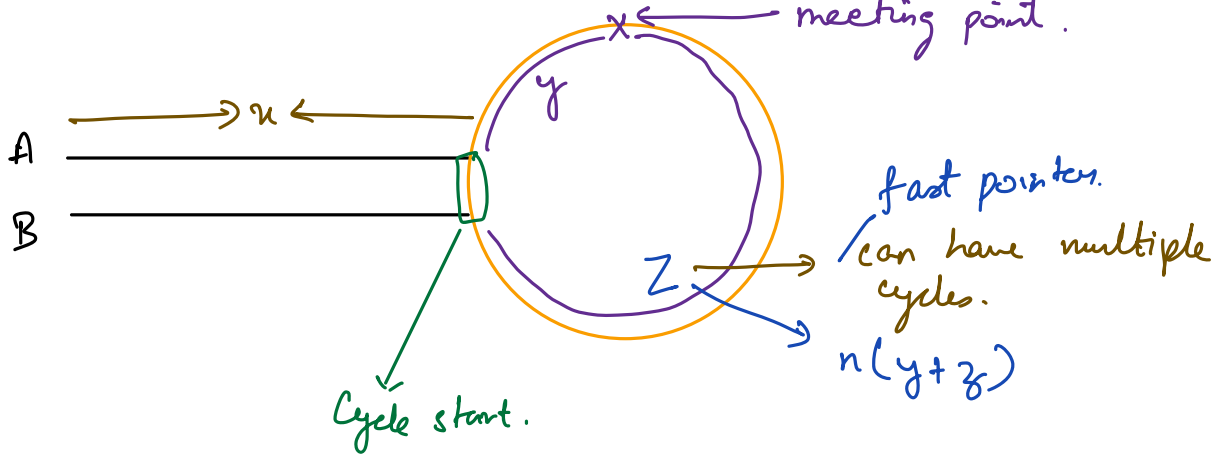
Q Given a L.L with cycle, find the start of cycle ?

S.C ⇒ O(1)





⇒



2 × (Slow pointer speed) = fast pointer speed

⇒ Distance travelled by fast pointer will be double the distance travelled by slow pointer.

$$2(x+y) = x+y+z+y$$

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

$$\Rightarrow \quad x = z$$

Code

```
s = head        f = head
while (f != NULL AND f.next != NULL)
{
    s = s.next
    f = f.next.next
    if (s == f) break.
}
x = Head.
while (x != s)
{
    x = x.next
    s = s.next
}
return x
```

T.C → O(N)

S.C → O(1)