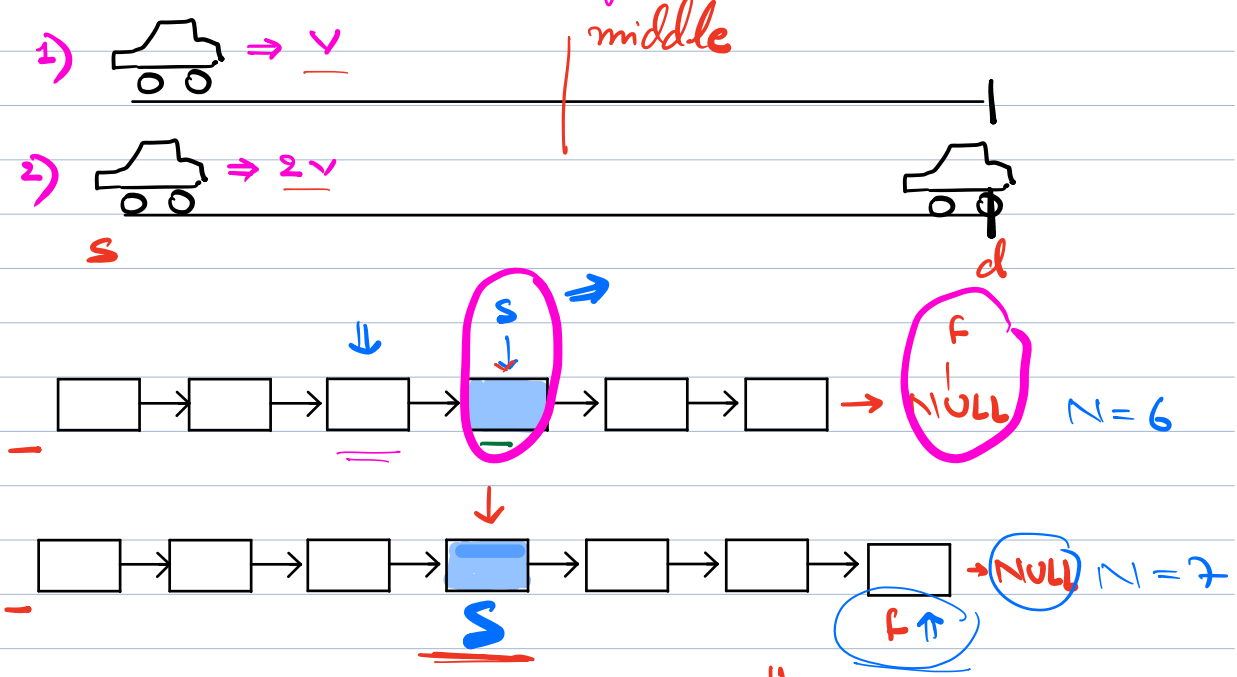
Given a LL. find the middle node.

→ Slow & fast pointer.



(fast == NULL OR fast Next == NULL) → stop

```
Node getMid (head) {
    slow = head;
    fast = head
    while (fast != NULL && fast.Next != NULL){
        slow = slow.next;
        fast = fast.next.next;
    }
```

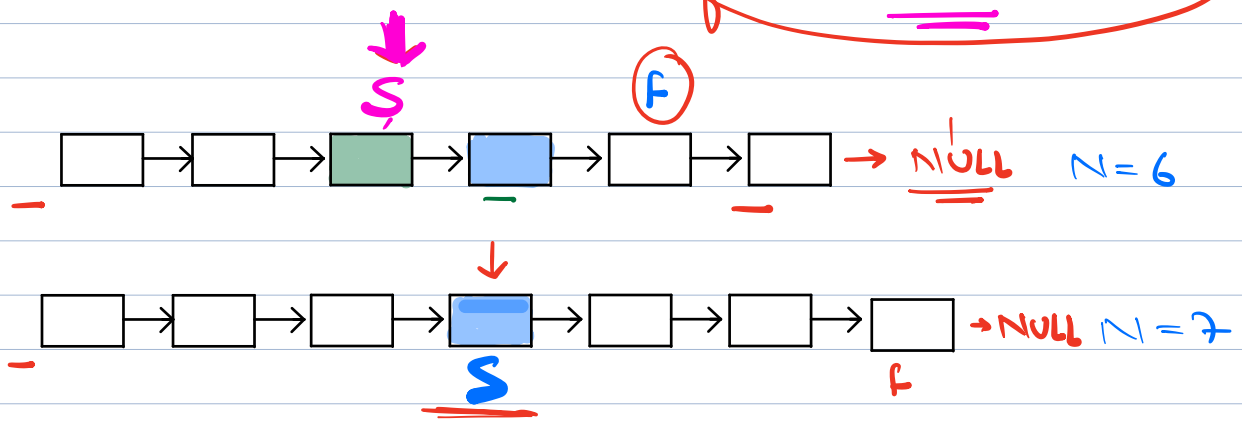
val

return slow;

}

!= NULL

fast.next.next = NULL



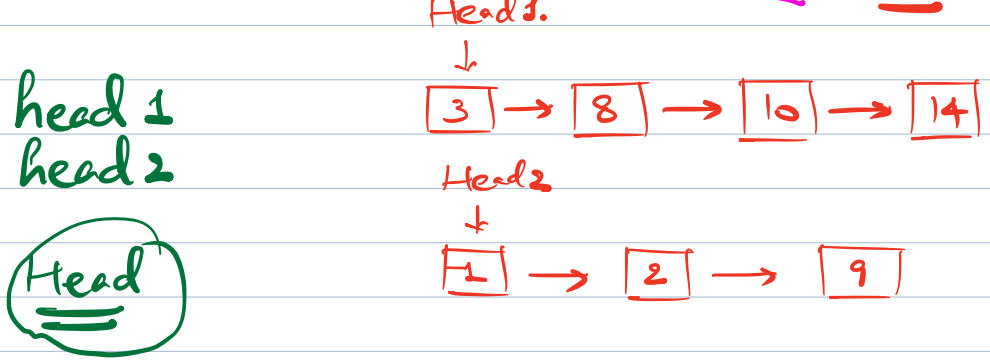
while (fast.next != NULL && fast.next.next != NULL)

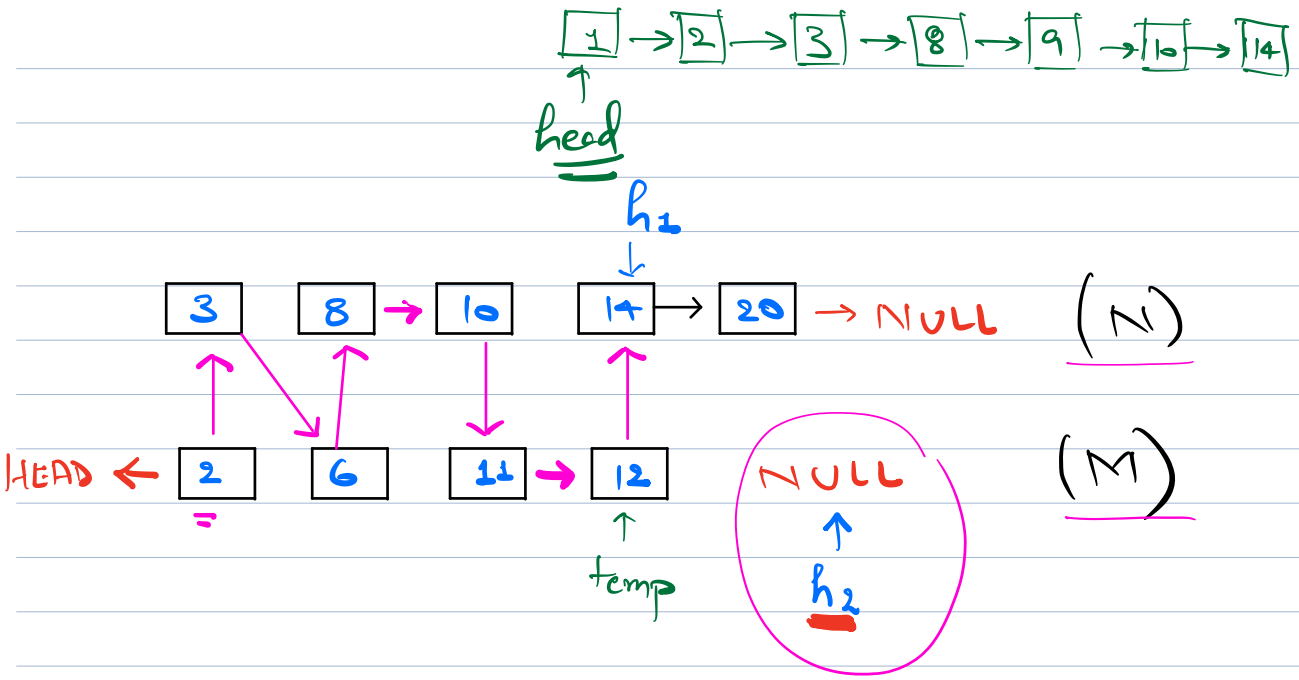
Amazon

Given 2 sorted linked Lists.

Merge both of them to create a new sorted linked list

S.C. O(1)





Code

```
Node merge (h1, h2) {
    if (h1.val < h2.val) {
        Head = h1;
        h1 = h1.next;
    }
    else {
        Head = h2
        h2 = h2.next;
    }
}

temp = Head;
while (h1 != NULL && h2 != NULL) {
    if (h1.val < h2.val) {
        temp.next = h1;
```

Initialisation

```
            h1 = h1.next;
            temp = temp.next;
        }
        else {
            temp.next = h2;
            h2 = h2.next;
            temp = temp.next;
        }
    }

    if (h1 == NULL) {
        temp.next = h2;
    }
    else {
        temp.next = h1;
    }

    return Head
}
```

T.C. = $O(N+M)$

S.C. = $O(1)$

Google, MS, Amazon, LinkedIn

Given a Linked List.

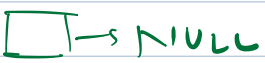
Sort the linked list using Merge Sort.

```
Merge Sort ( ) {
    mid
```

mergeSort (low, mid)
mergeSort (mid+1, high)
merge ();

}

Code

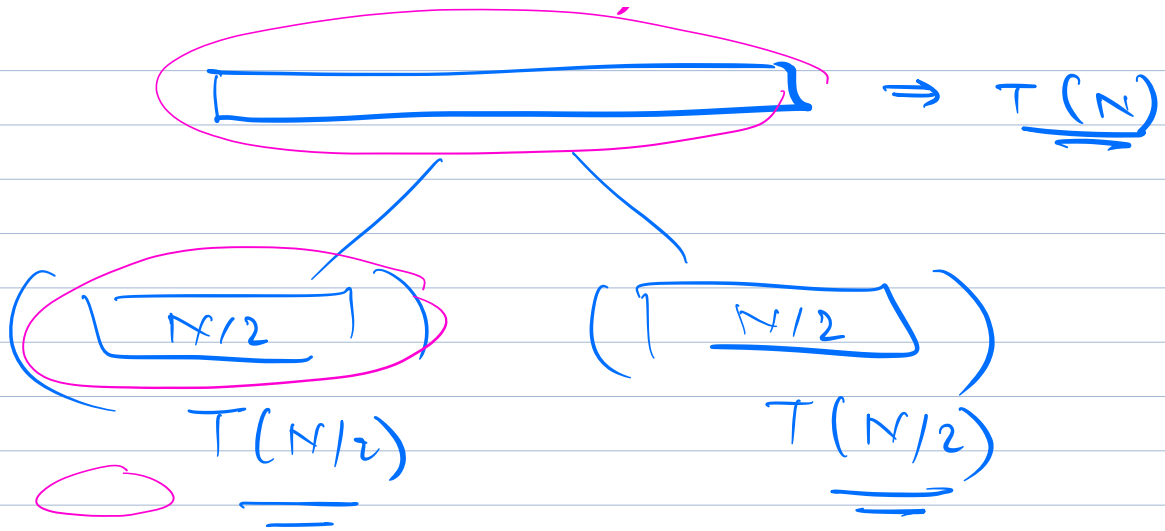



```
Node mergeSort (head) {
    if (head == NULL || head.next == NULL) {
        return head;
    }
    Node mid = get1stMid(head);                    O(N)
    Node h2 = mid.next;
    mid.next = NULL
    Node h1 = mergeSort(head);
    h2 = mergeSort(h2);
    head = merge(h1, h2);                          O(N)
    return head
}
```

Base Case

T.C. = $O(N \log N)$

$T(N) = 2T(N/2) + O(N)$

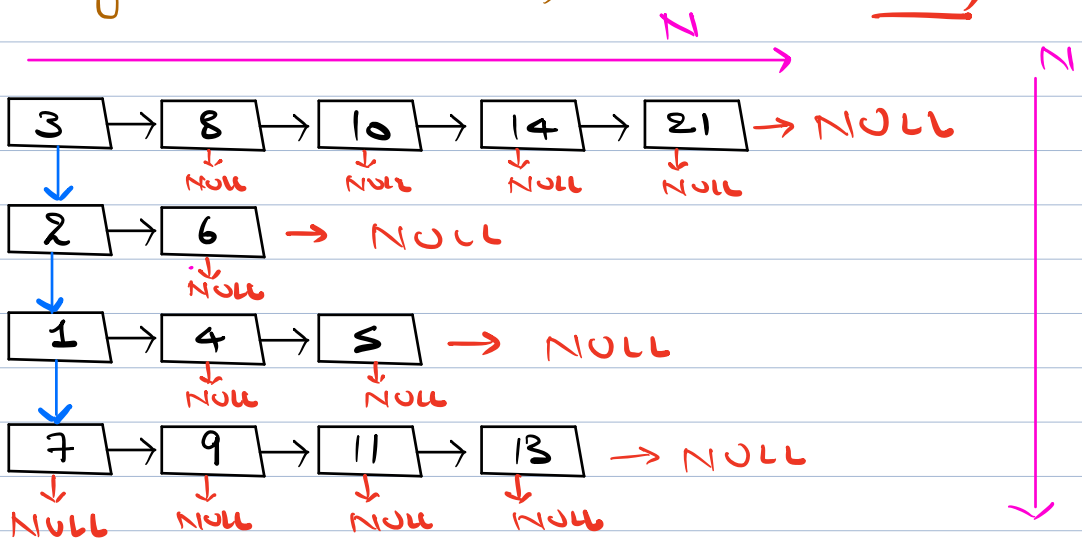


S.C. = O(log N)

Break till 10:58

Google

Given a N×N 2D List (sorted horizontally). flatten it to a single list (sorted) S.C. O(1)

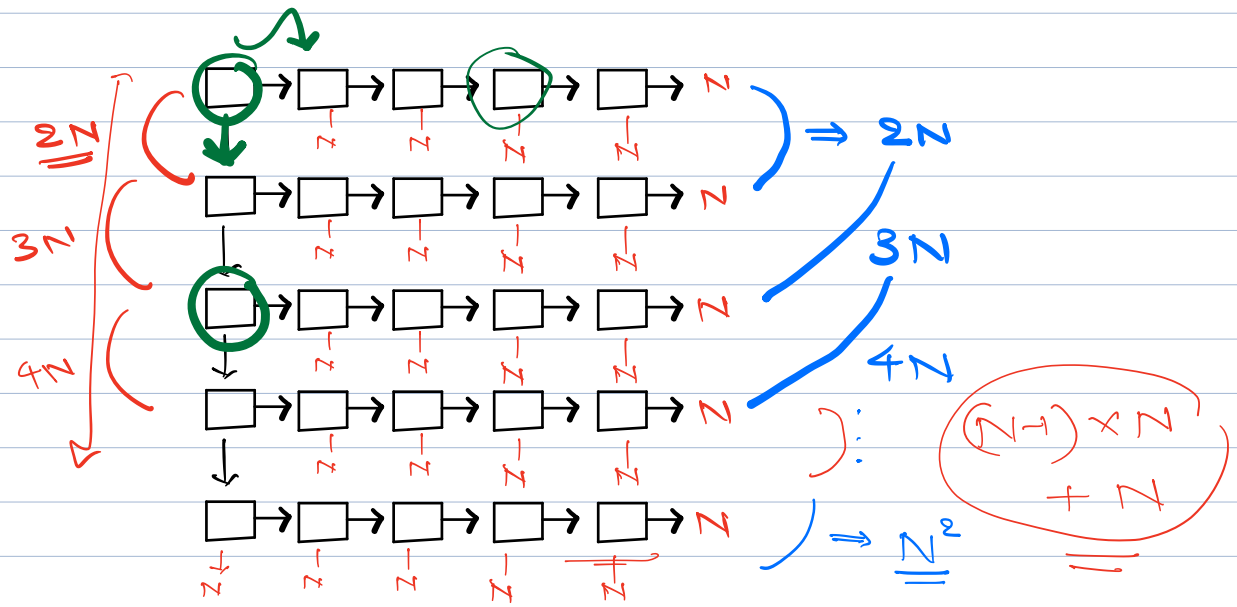


Node {

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10

11 → 13 → 14 → 21

```
int val;
Node next;
Node down;
}
```



⇒ Merge 2 lists at a time.

iterations = $2N + 3N + 4N \ldots\ldots N \times N$

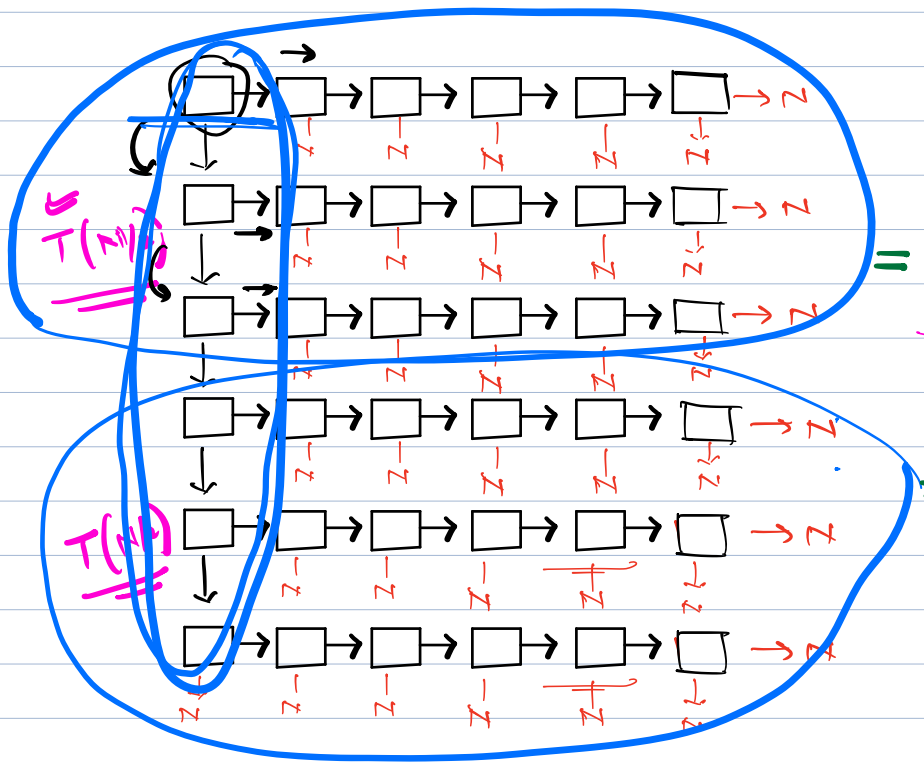
$N(2+3+4+5 \ldots\ldots N)$

$N \times N^2$

$= O(N^3)$

a^N — $a \times a^{N-1}$ (a^{N-1}: $N-2$) $\Rightarrow O(N)$

$a^{N/2} \times a^{N/2}$



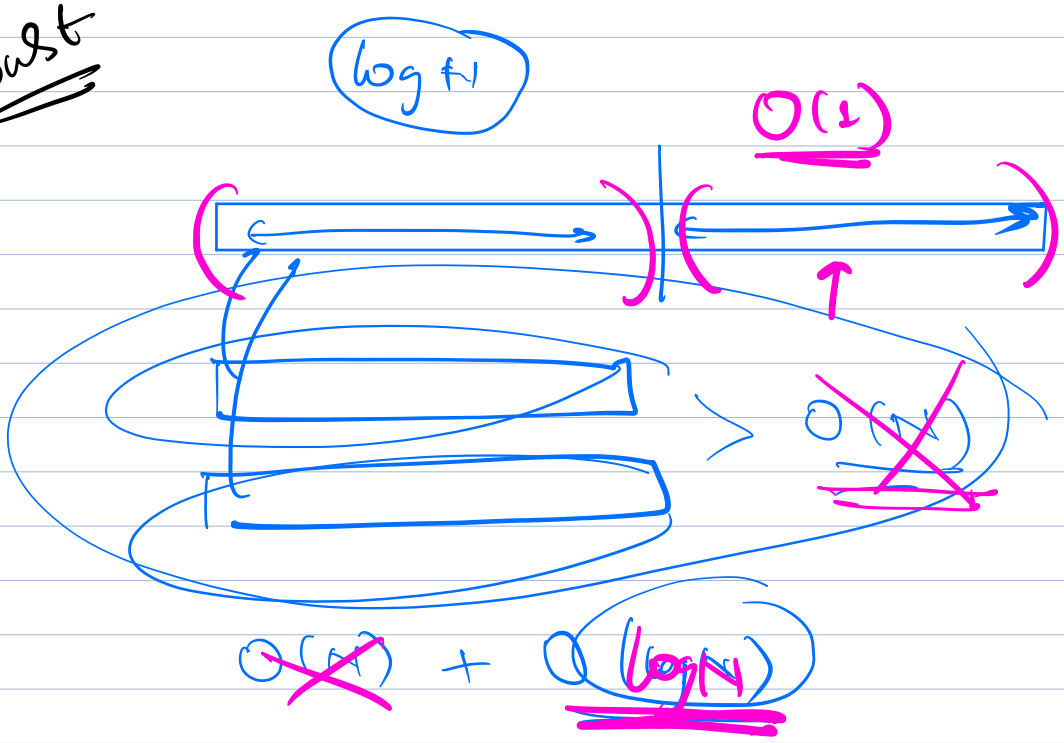
$$T(N) = 2T\left(\frac{N}{2}\right) + O(N^2) \Rightarrow \text{H.W.}$$

```
Node mergeLists(head) {
    if (head == NULL || head.down == NULL) {
        return head;
    }
}
```

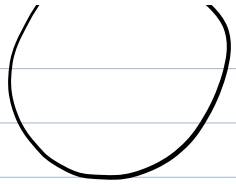
// Ass: mergeLists ⇒ merges all lists whose heads are connected to a single list.

```
Node mid = getIstMid(head)   // Replace next with down.
h2 = mid.down
mid.down = NULL;

head = mergeLL(head);
h2 = mergeLL(h2);

head = merge(head, h2);

return head;
}
```

Doubt



while (____ . next != null) ⇒ $N + \frac{N}{2}$



$= \frac{3N}{2}$

Iterations $\Rightarrow \frac{N}{2}$