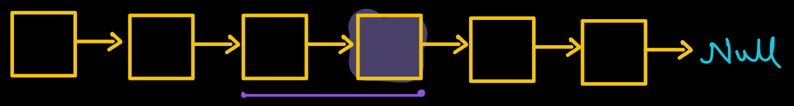
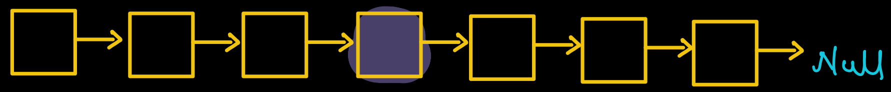
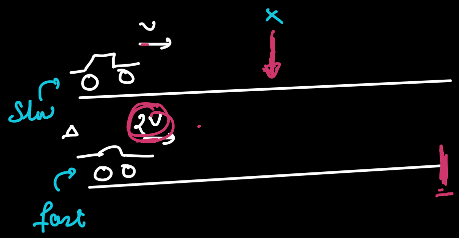
Q Given a LL. Find the middle node.

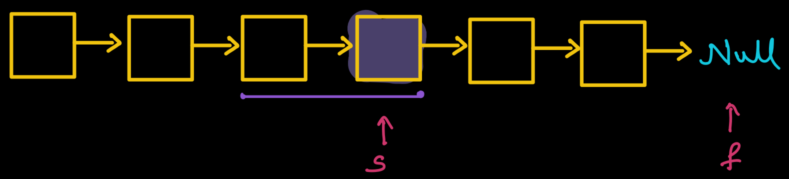


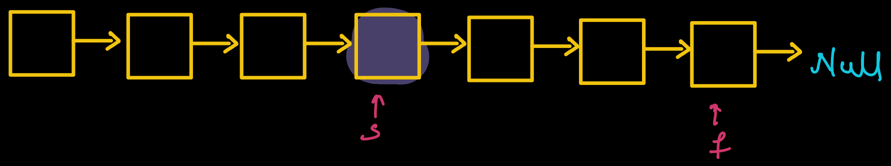


```
ListNode getMid ( head) {

    slow = head;
    fast = head;

    While ( fast != null && fast.next != null)

        slow = slow.next;
        fast = fast.next.next;

}
```

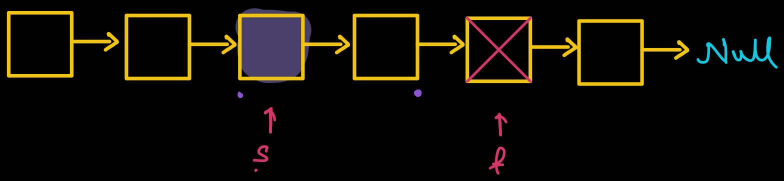


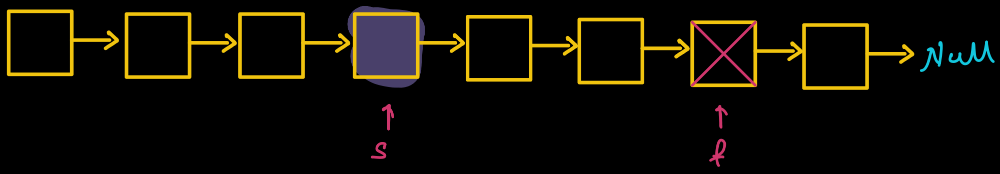
node.next
node.val
↓
slow
fast
fast.next



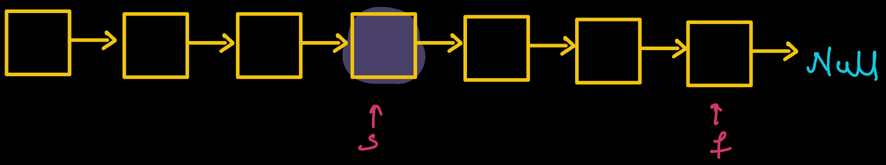


If n is even. Return the 1st mid.



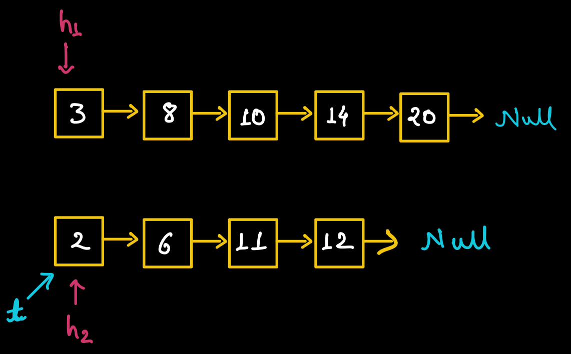


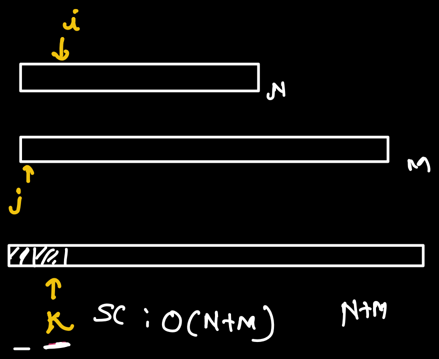
```
ListNode getMid ( head) {

        slow = head;
        fast = head;

        While ( ~~fast != null &&~~ fast.next != null && fast.next.next != Null)
            slow = slow.next;
            fast = fast. next. next;

    }
```

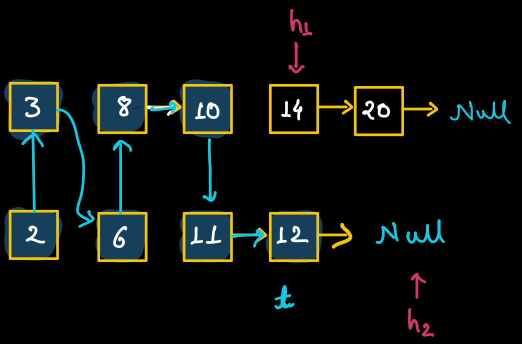


Q Given 2 sorted lists. Do in-place merging of them to create a new sorted list. → SC : O(1)

Amazon



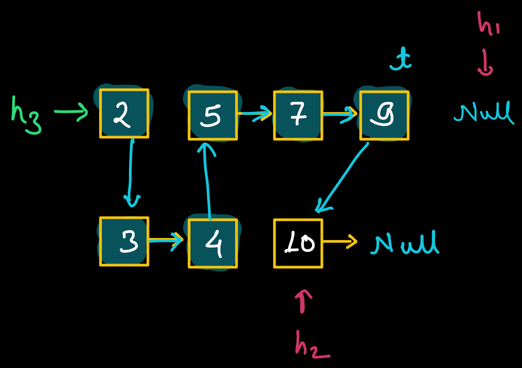




```
if (h1.val < h2.val){
        t.next = h1 ;
        h1 = h1.next;
}

else {
        t.next = h2;
        h2 = h2.next;
}
t = t.next;
```

```
List Node   merge ( h1,  h2) {
        // fin the head
        if ( h1 . val  <  h2. val) {
                h3  =  h1;
                h1  =  h1 . next;
        }
        else {
                h3  =  h2;
                h2  =  h2. next;
        }

        t  =  h3 ;

        While (  h1 != null && h2  != null)  {

            if ( h1. val  <  h2. val) {
                    t. next  =  h1;
                    h1  =  h1. next;
            }
            else {
                    t. next  =  h2;
                    h2  =  h2. next;
            }
            t = t. next;
        }
        if( h1 = = null)  {  t . next  =  h2;}
        else {
                t. next  =  h1;
        }
        return  h3
}
```

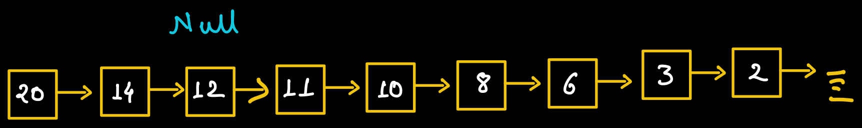
TC : O(N+M)
SC : O(1)



Follow-Up Question

Merge the two sorted (Asc) lists to form a sorted list in DES order.

Null

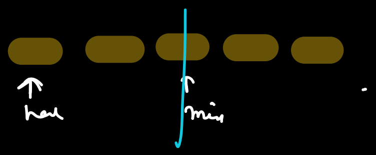


Google
MS
Amazon

Q Given a LL. Sort it using merge-sort.

```
Merge Sort ( data ) {
    sorted 1st half = mergeSort ( first half );
    sorted 2nd half = mergeSort ( sec half );
    ret  merge ( sorted 1st half , sorted 2nd half )
}
```

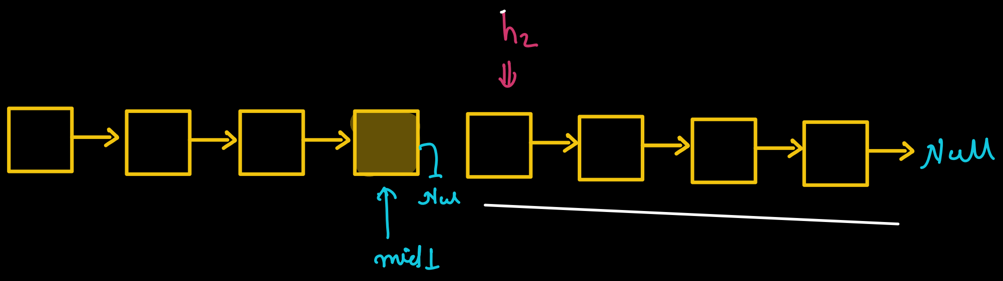
```
ListNode  mergeSort ( ListNode head ) {
    if ( head == null || head.next == null )
        ret head;
    // Assumption : mergeSort ( node )
                     → sorts the list from node to null

    ListNode mid = get 1st Mid ( head ); ⇒ O(N)
        h2       =  mid. next ;
    mid . next   =  null ;

    ListNode h1 = mergeSort ( head );
    ListNode h2 = mergeSort ( h2 );
    ret merge ( h1 , h2 );      ⇒ O(N)
                └→ SC: O(1)
}
```



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

TC: $O(N \log N)$

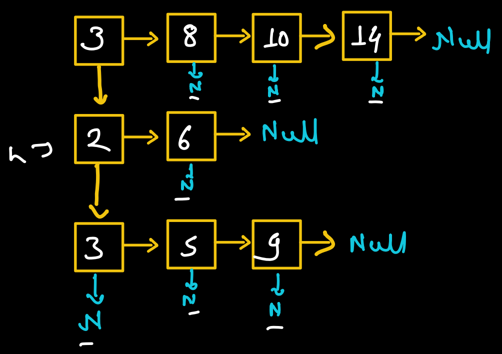
SC: $O(\log N)$ (Recursion stack)

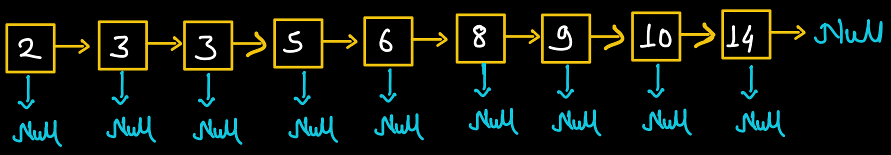


Google

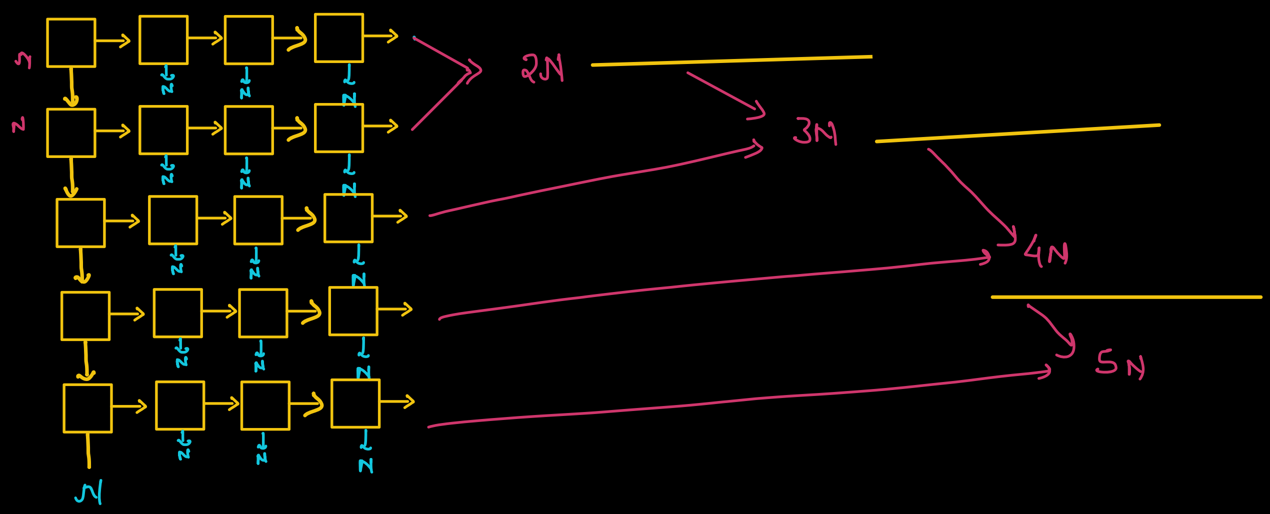
Q Given a 2D list. Flatten it to a singly list.
N×N
(sorted horizontally) (sorted)

```
class ListNode {

    int val;
    ListNode next;
    ListNode down;

    Public ListNode(int x) {
        this.val = x;
        this.next = null;
        this.down = null;
}
```





Approach 1 Merge 2 lists at a time



$$\# \text{ iteration} = 2N + 3N + 4N + 5N + \cdots\cdots N \times N$$

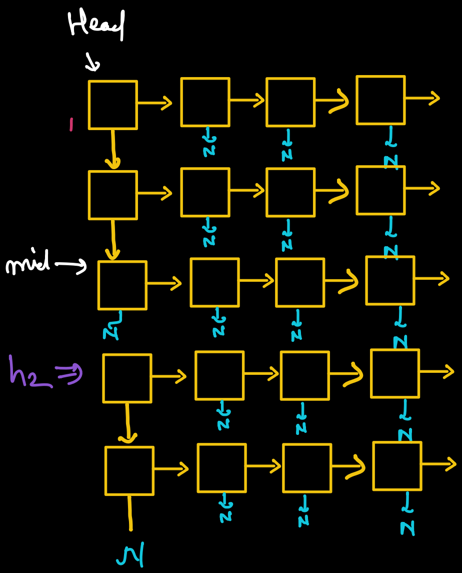
$$= N(\underbrace{2+3+4+5 \;----\; N}_{O(N^2)})$$

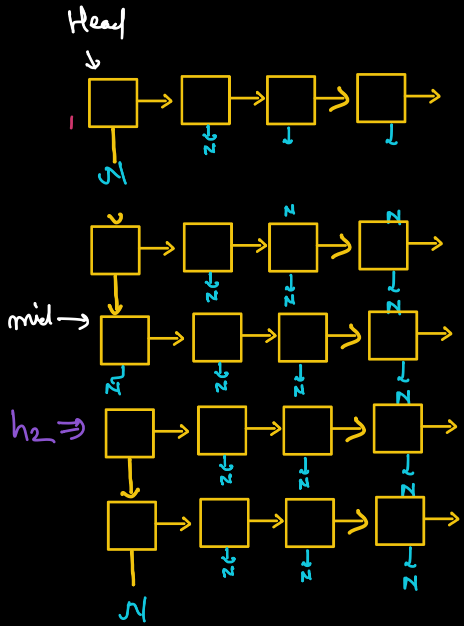
$$= O(N^3)$$

$$\text{Pow}(a, n) = a^n$$

$$\begin{tikzcd} a^n \arrow[r] \arrow[rd, bend right] & a \times a^{n-1} \\ & a^{n/2} \times a^{n/2} \; ✓ \end{tikzcd}$$

```
ListNode merge2DList ( head) {
    if ( head == null || head.down == null )
                                   ret head;

Assumption : merge2DList(node) => Merges all lists whose heads
                                  are connected to node into
                                  a singly list.

    ListNode mid = getMid(head) // using down points
                        └──→ O(N)          (not next)
             h2  = mid.down;
        mid.down = null;
            head = merge2DList( head);
              h2 = merge2DList( h2);
             ret merge (head, h2);  => O(N^2)
}
```

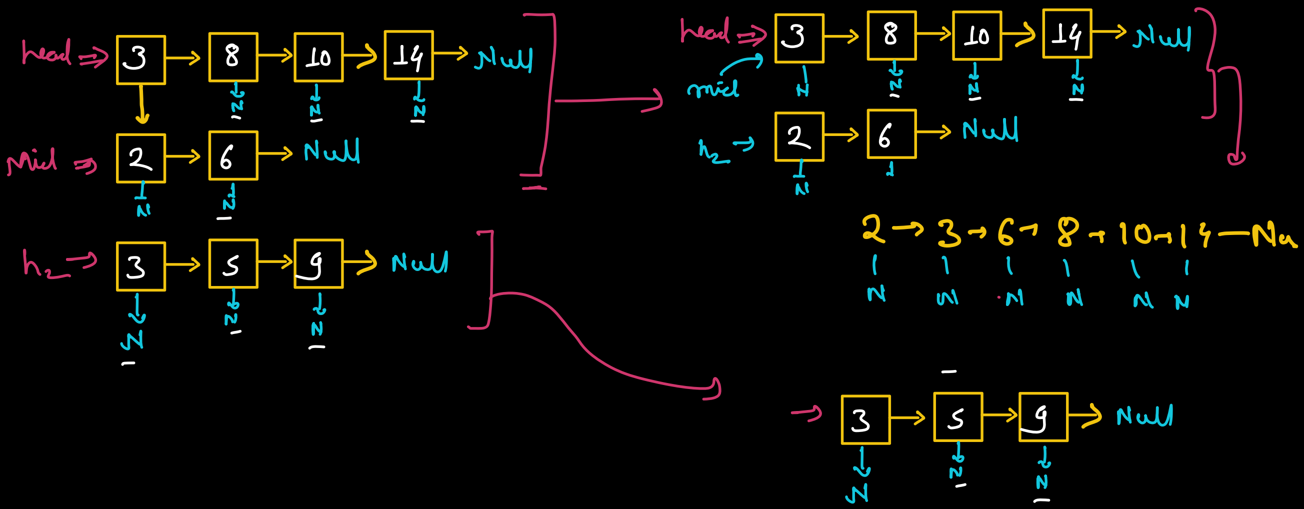




TC: $O(N^2 \log N)$

SC: $O(\log N)$

$$\boxed{T(N) = 2\,T(N/2) + O(N^2)}$$



① Problem Solving class ⇒ (Sunday)

② ⟶