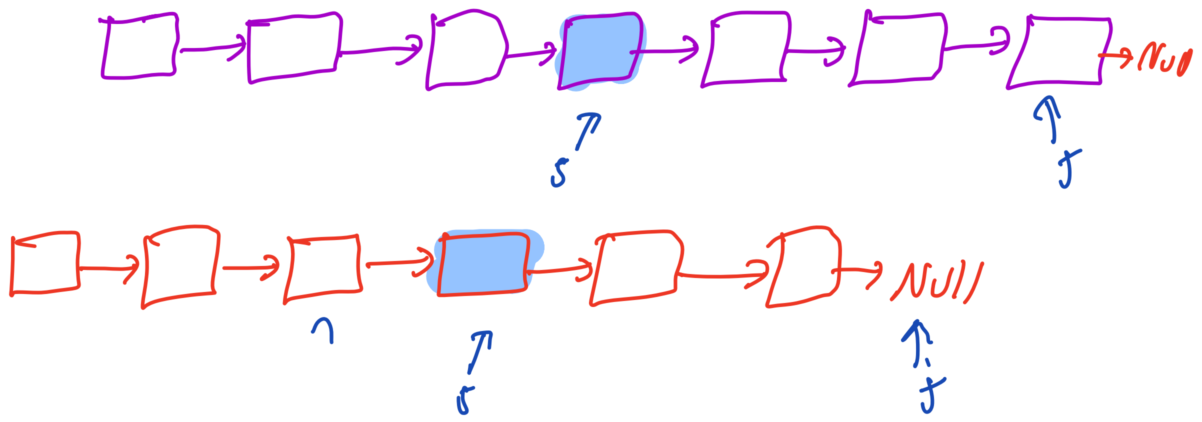
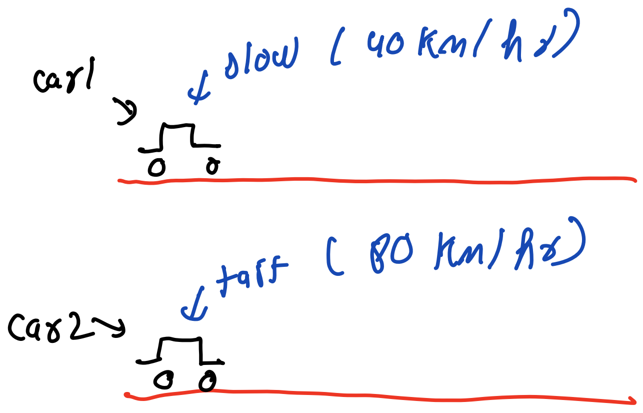
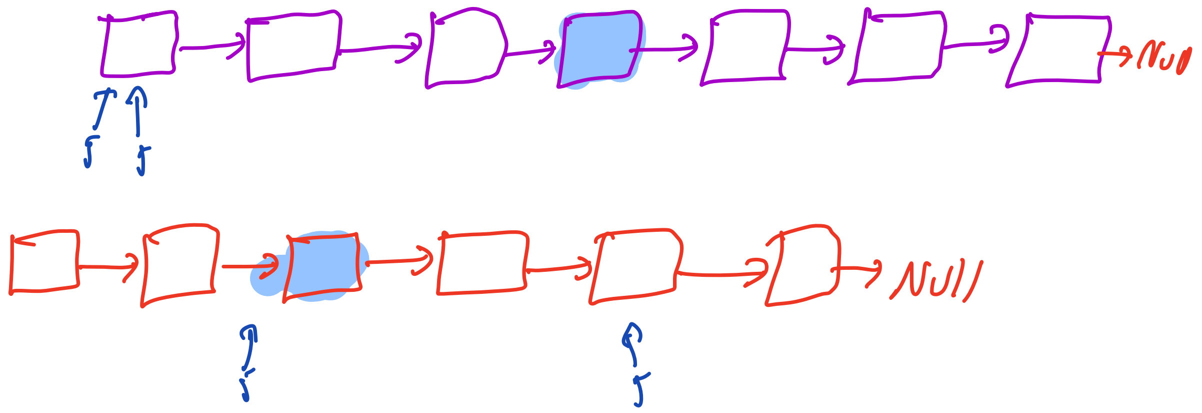
Link List 2

Q ⇛ Given a LL. Find the middle node

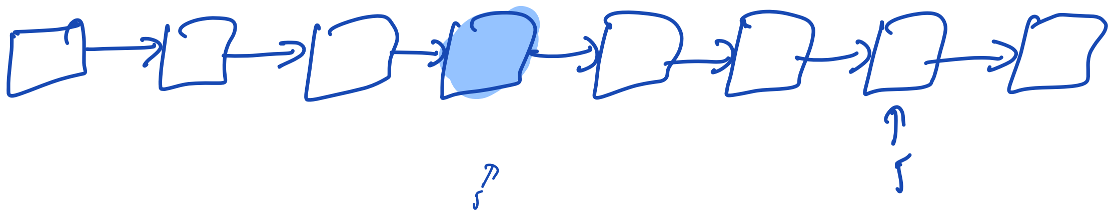




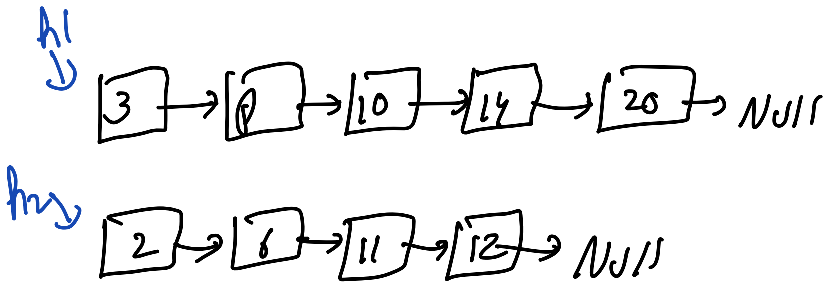
```
ListNode getMid( head)
{
    slow = head
    fast = head

    while ( fast != null && fast.next != null
    {
        slow = slow.next;
        fast = fast.next.next;
    }
    return slow;
}
```

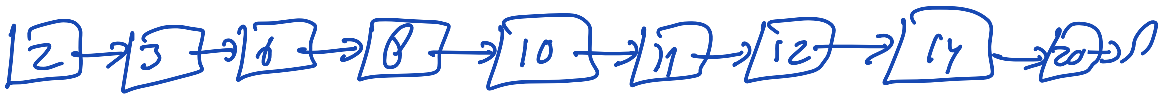


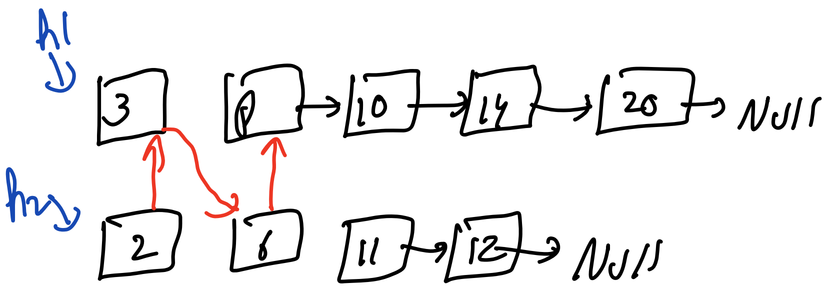
```
ListNode getMid( head)
{
    slow = head
    fast = head

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

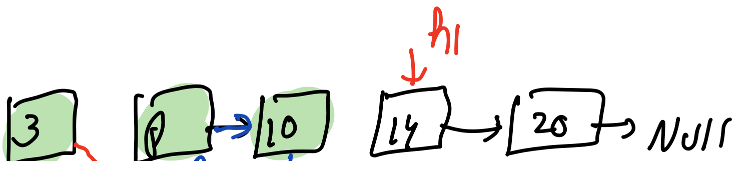


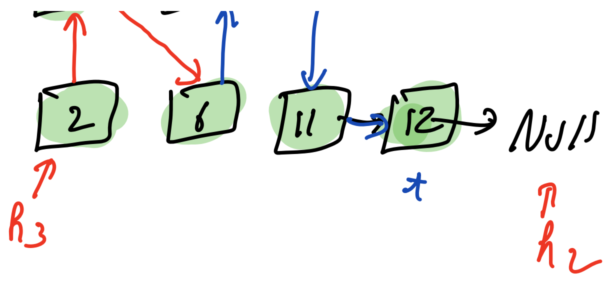
Q ⇒ Given 2 sorted lists. Do in-place merging of them to create a new sorted list







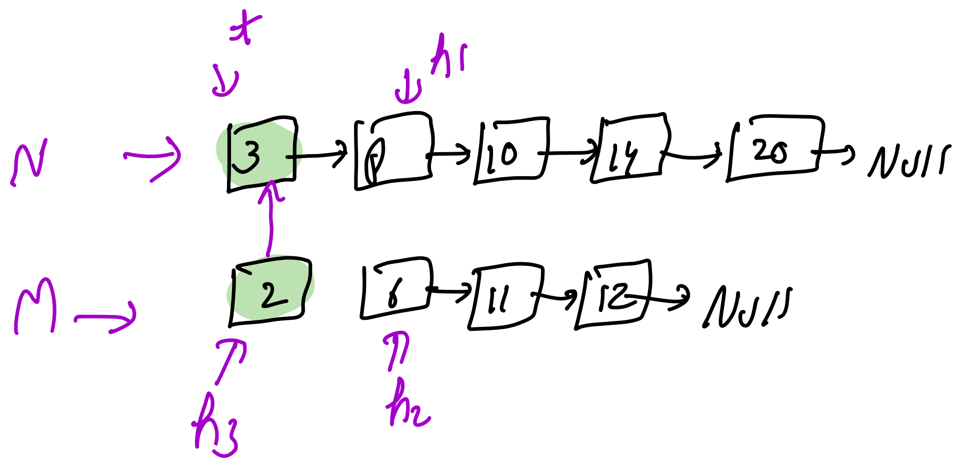




```
ListNode merge (h1, h2)
{
    if ( h1.val < h2.val)
    {
        h3 = h1
        h1 = h1.next;
    }
    else
    {
        h3 = h2
        h2 = h2.next;
    }
    t = h3;
while ( h1 != null && h2 != null)
{
    if ( h1.val < h2.val)
    {
        t.next = h1
        h1 = h1.next;
    }
    else
```

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

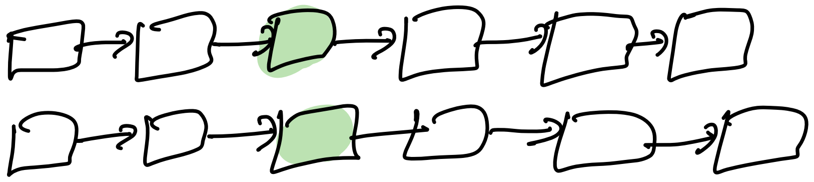
TC : O(N + M)
SC : O(1)



Google
MI ⊖ ⇒
Amazon

Given a LL. sort it usin merge sort

```
Mergesort( — )
{ sorted1st half = mergesort( first half)
  sorted 2nd half = mergesort( second half
        ret  merge (sorted 1st h, sort 2nd h
    }
```



```
ListNode  mergesort( ListNode  head)
{
        if ( head == null || head.next == null)
                ret  head;

        ListNode  mid =  get1stMid (head)  → O(N)
             h2   = mid.next
        mid.next  = null;

        ListNode  h1 =  merge sort (head);
        ListNode  h2 =  mege sort ( h2);
          ret   merge (h1 , h2);  → O(N)
                                  → O(1) : SC
}
```

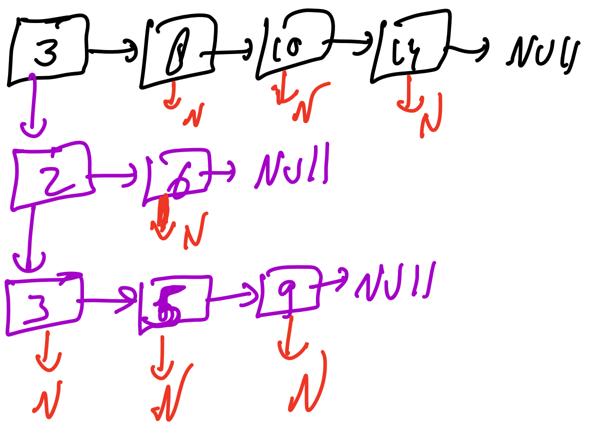
TC: O(N log N)
SC: O(log N)

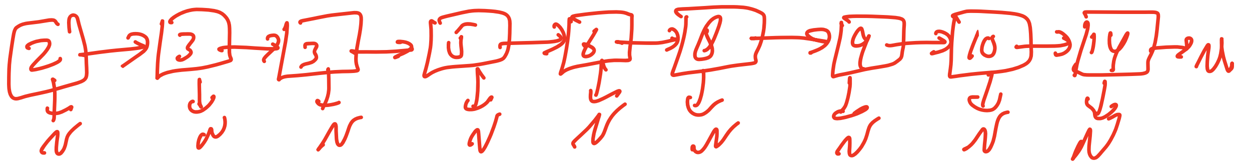
Break : 8 : 53

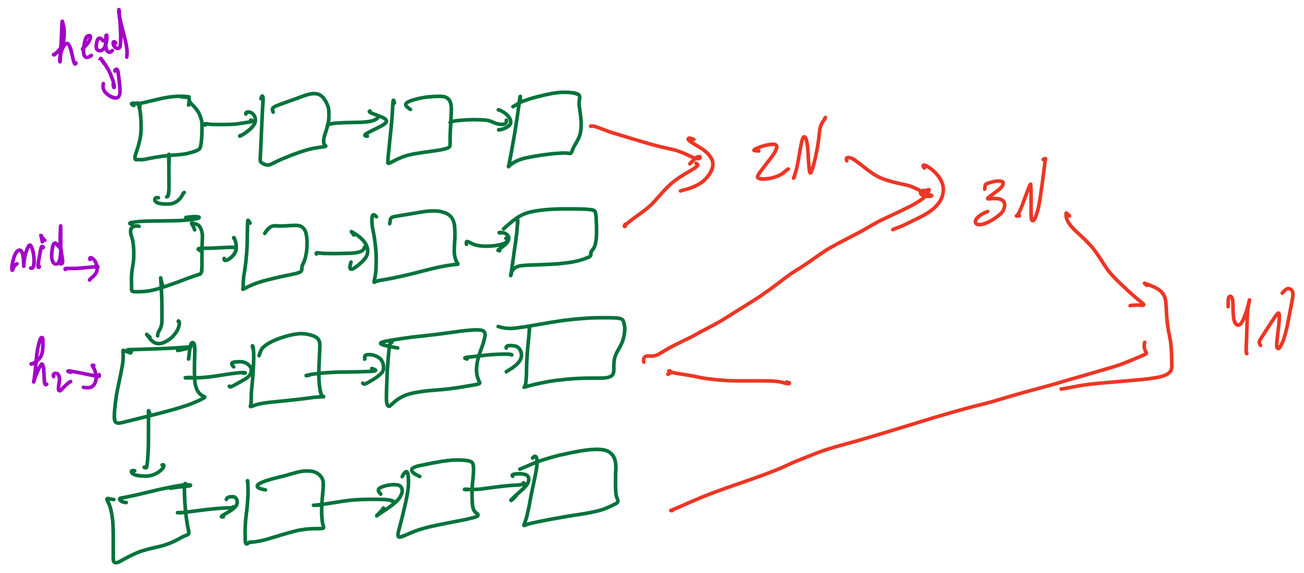
Google
Q ⇒

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

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





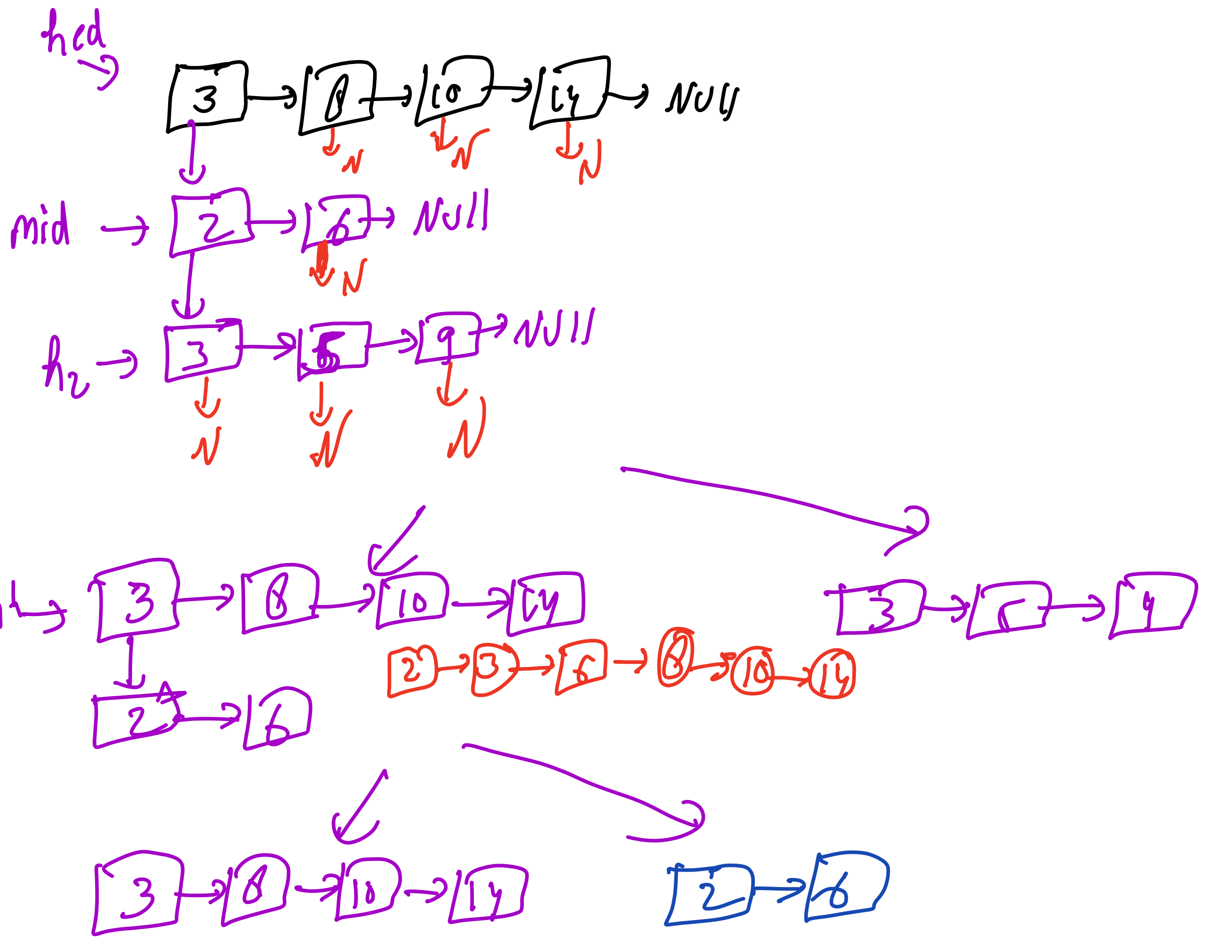


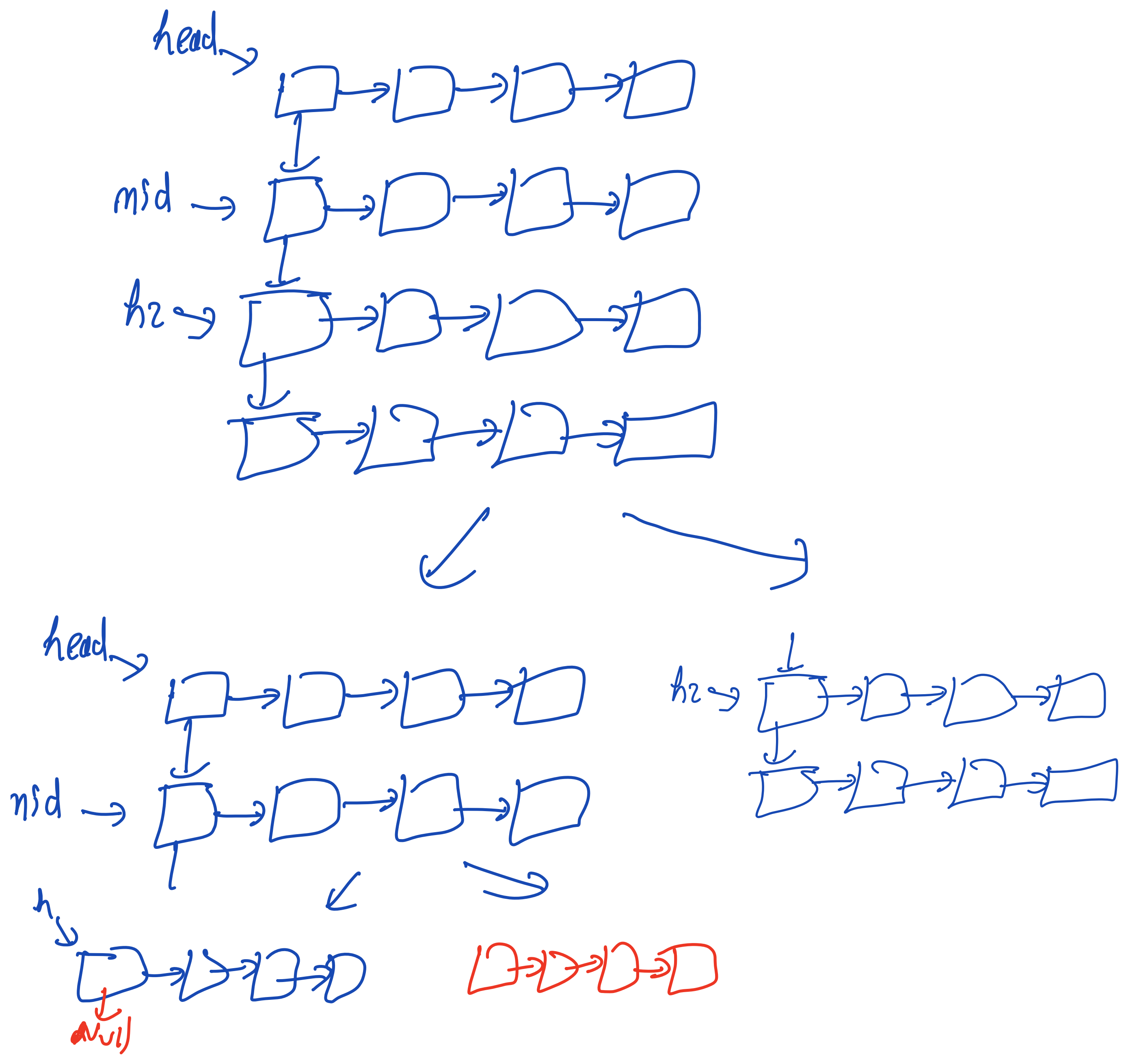
iteration = 2N + 3N + 4N + ---- + N*N

$$= \underbrace{N}_{} \underbrace{(2 + 3 + 4 + \cdots + N)}_{N^2}$$

$$\Rightarrow O(N^3)$$

$$\text{pow}(a, n) \begin{cases} \rightarrow a * a^{n-1} \\ \rightarrow a^{n/2} * a^{n/2} \end{cases}$$





```
List Node merge 2DList ( head)
{
    if (  head= =null  ||  head.down==null)
            return  head;

    List Node  mid =  getMid  (head)
```

// using down pointers

```
    h2 = mid.down
    mid.down = null

    head = merge2DList(head);
    h2 = merge2DList(h2);
    ret merge(head, h2);  → O(N^2)
}
```

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

$T(N) = 2T\left(\frac{n}{2}\right) + O(N^2)$

SC: $O(\log N)$