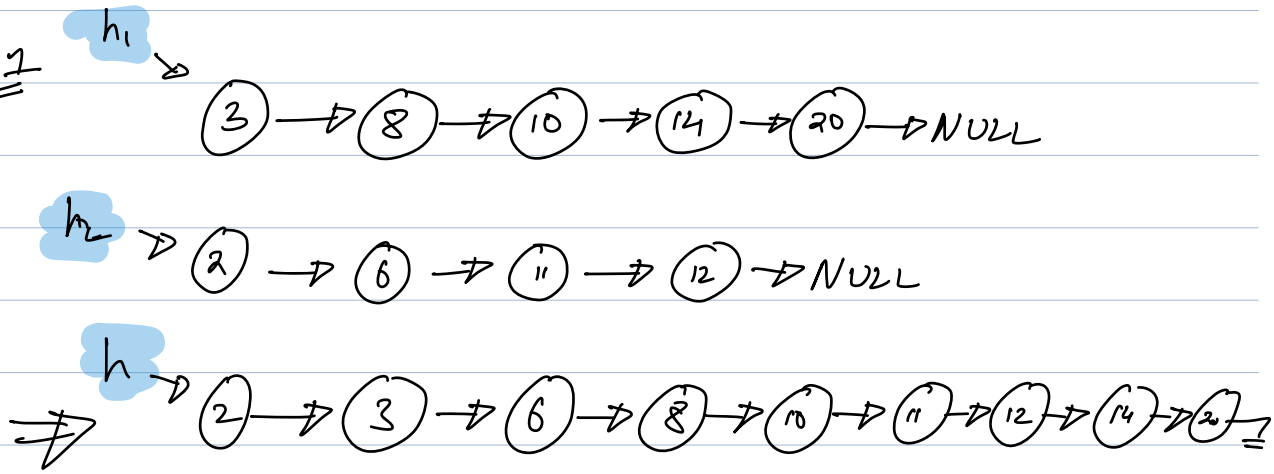
Q Given two sorted linked list. Merge them into a single sorted LL!

Ex1




```
Node merge (Node head1, Node head2) {

}
```

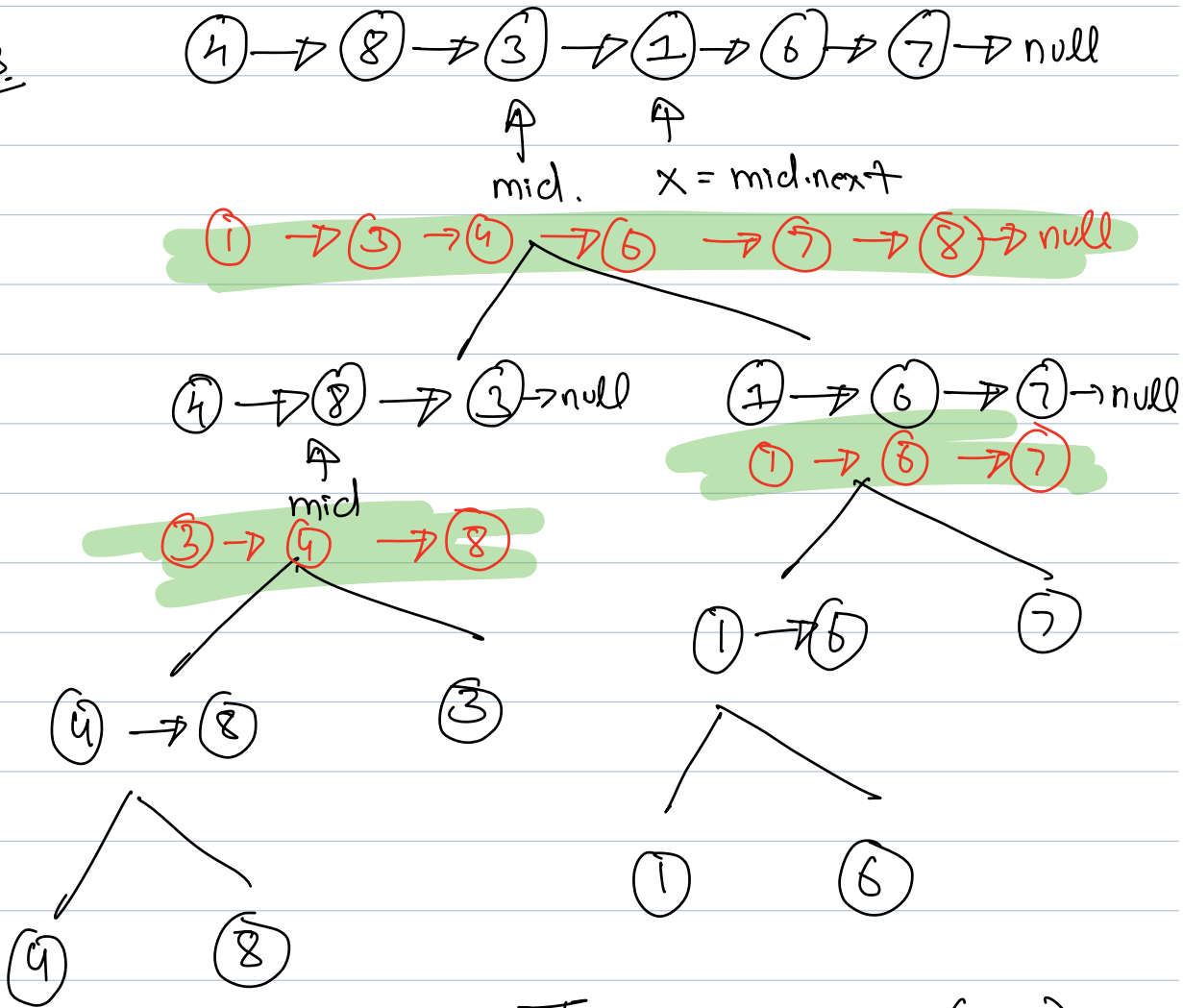
Tc : O(m+n)

Sc : O(1)

Merge Sort in linked list

Q Given a linked list. Sort it!

Ans



TC : $n + 2T(n/2)$

SC : $\log(n)$

TC : $O(n \log n)$

Pseudo Code !

```
Node mergeSort (Node head) {

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

//O(n)  Node mid ⇒ findMiddleNode (head);
      Node h2 ⇒ mid.next
      mid.next = null;

      head = mergeSort ( head);
      h2 =   mergeSort ( h2);

//O(n)  return  merge (head, h2);

}
```

$$T(n) : n + 2T(n/2)$$

$$T(n) : \underline{O(n \log n)}$$

Q1 Given a Linked list. Every node has 2 pointers.

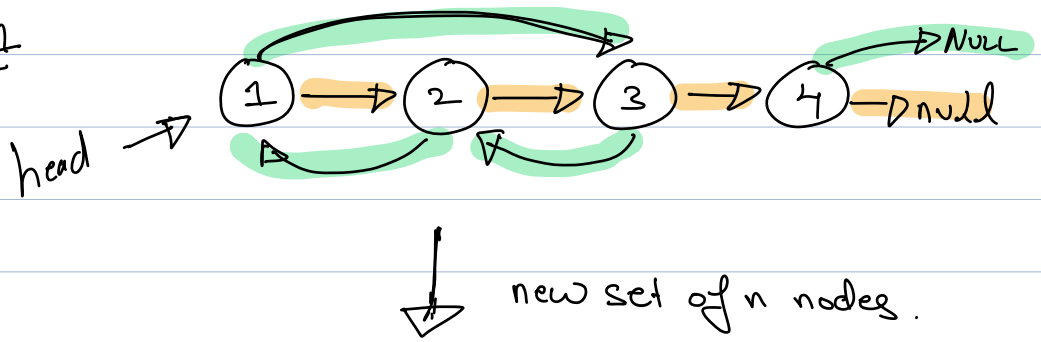
1) next pointer : As usual.

2) random pointer : Can point to any node or null

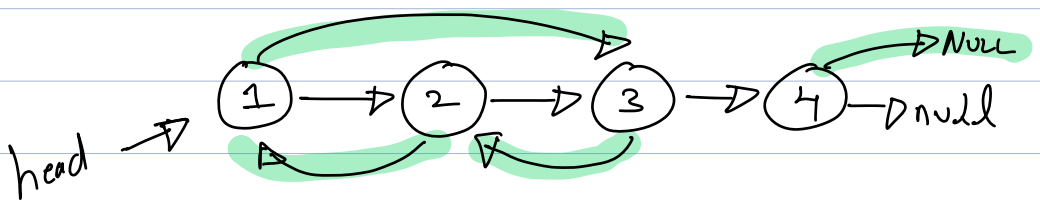
10:25

Make a deep clone of this. Brand new n nodes which the same structure.

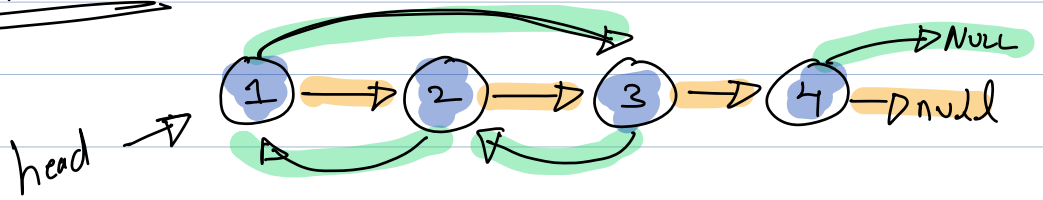
Ex 1

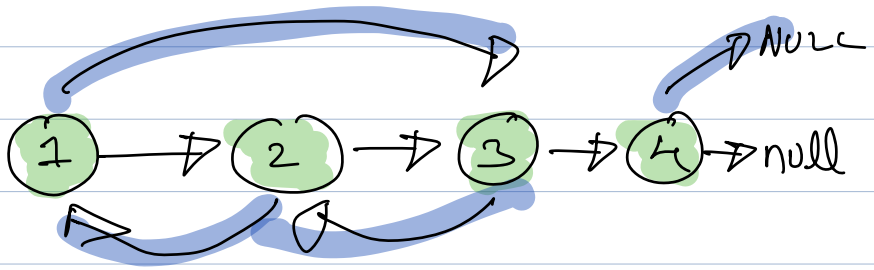


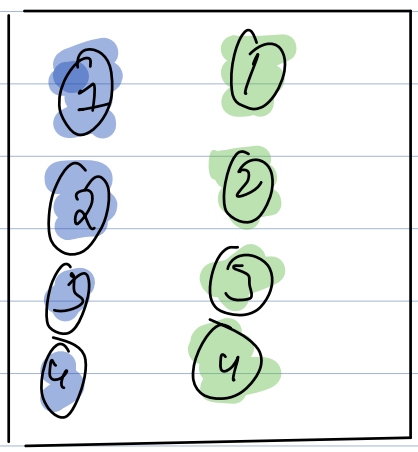
new set of n nodes.



Approach 1 :



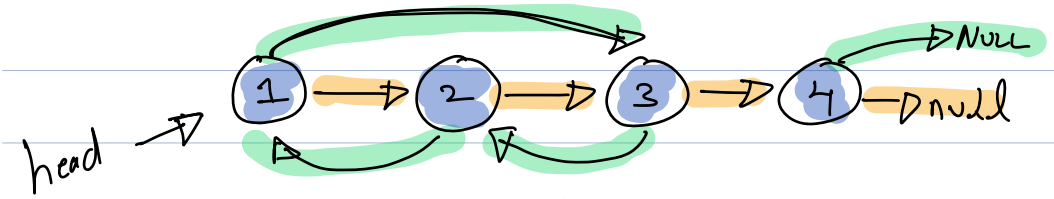


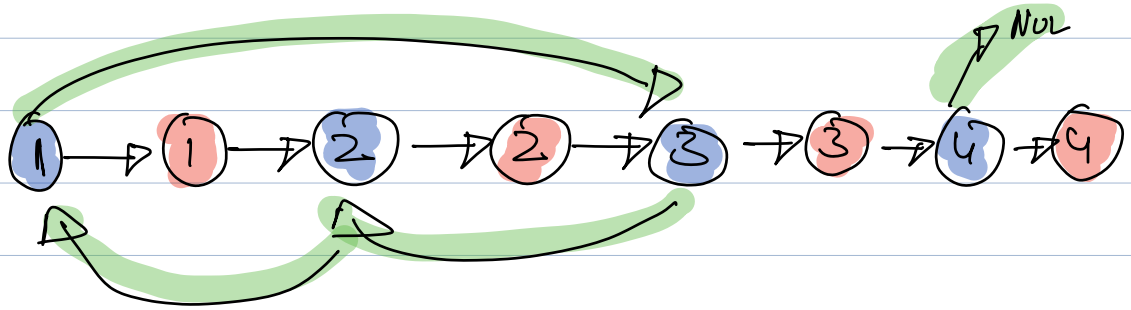


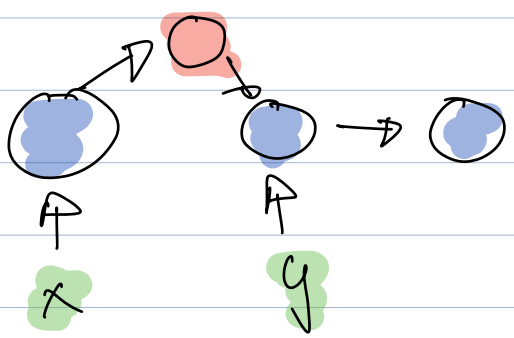
TC : O(n)

SC : O(n)

Step 1 : Create new nodes and attach it to the odd linked list.



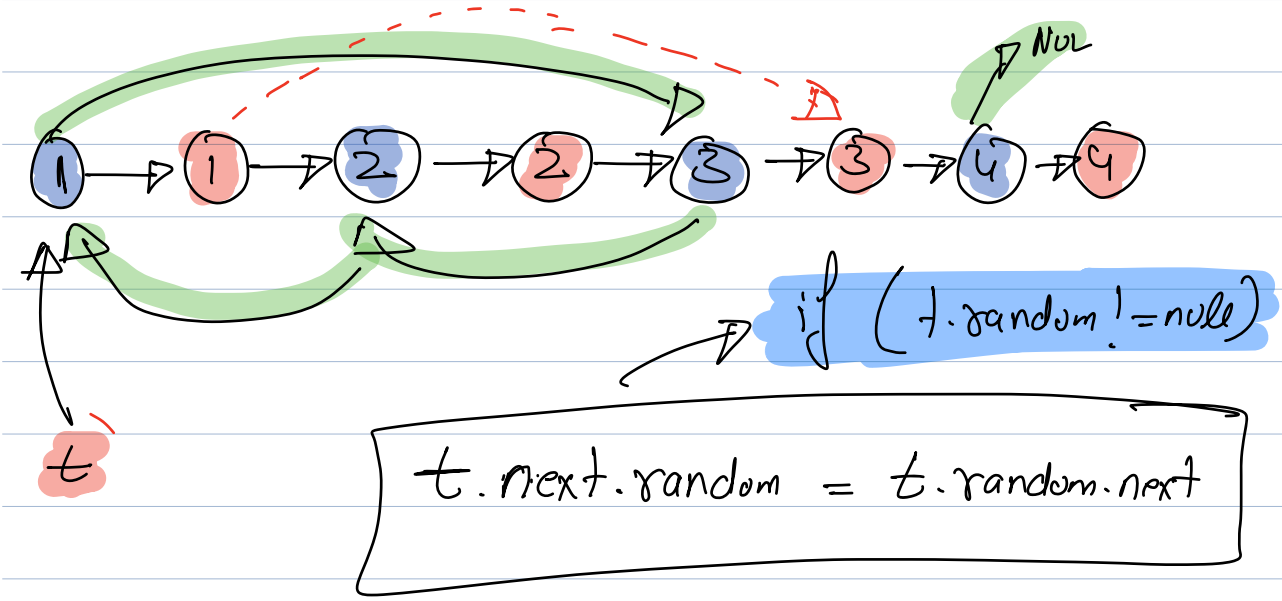




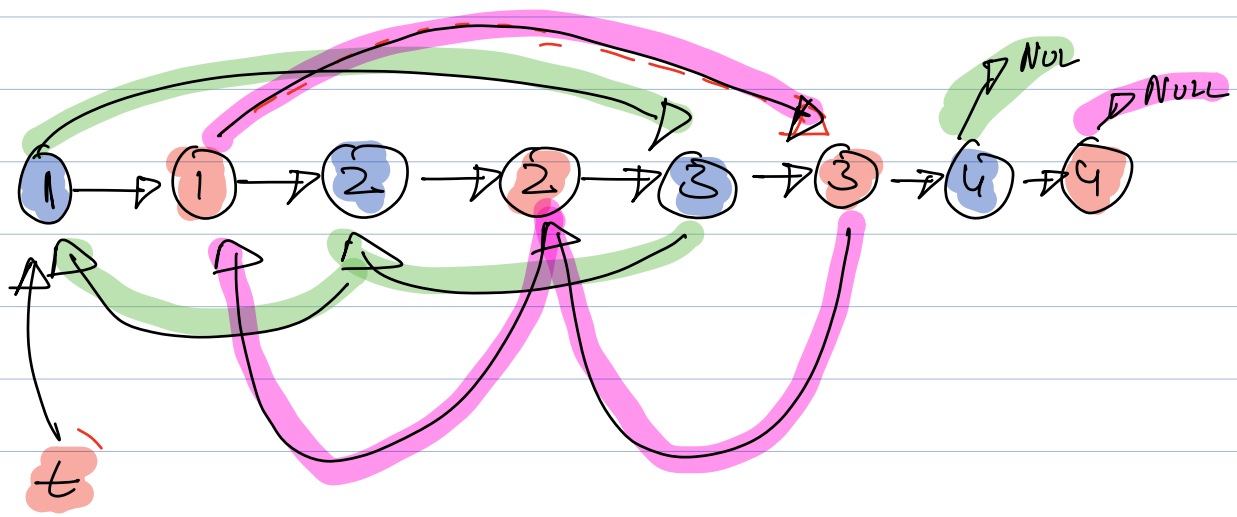
x.next = new Node(x.val)

x.next.next = y.

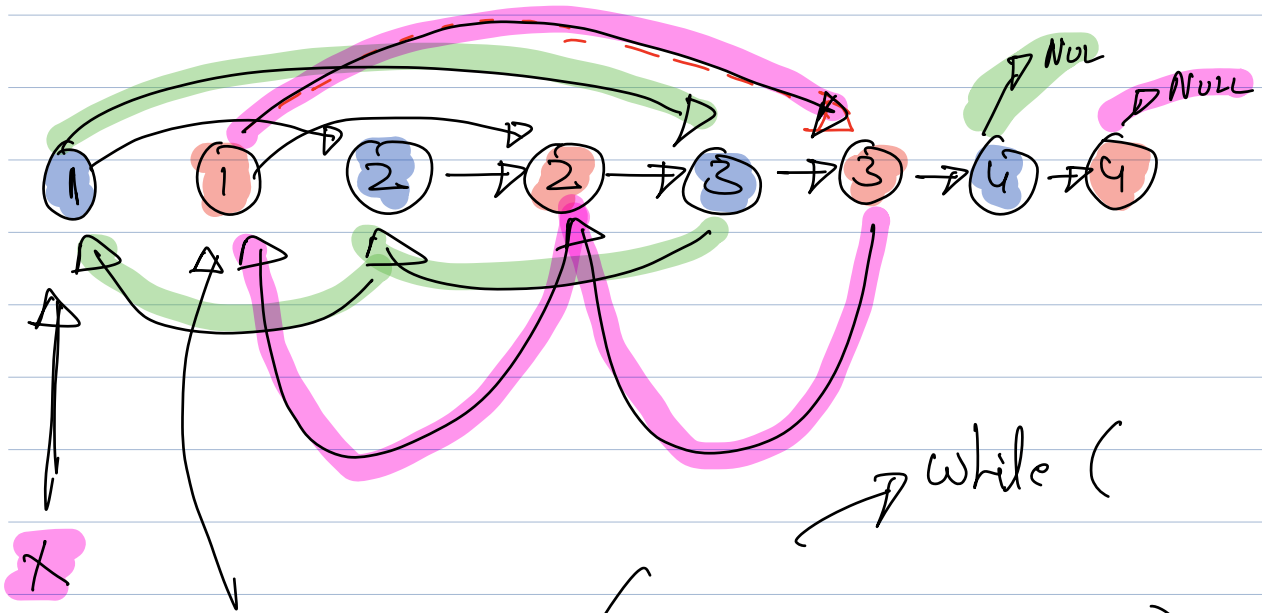
Step 2 : Point the random links to the correct Node



⇒ Run while
t = t.next.next;



Step 3:



x

y

while ( )

x.next ⇒ y.next

y.next ⇒ y.next.next

might be an edge case.

x. ⇒ x.next

y ⇒ y.next

TC: O(n) SC: O(1)

Q2 Flatten a linked list.

⇒ We have a linked of a different structure.
Each node has 2 pointers.

next pointer : Pointer to the next node.

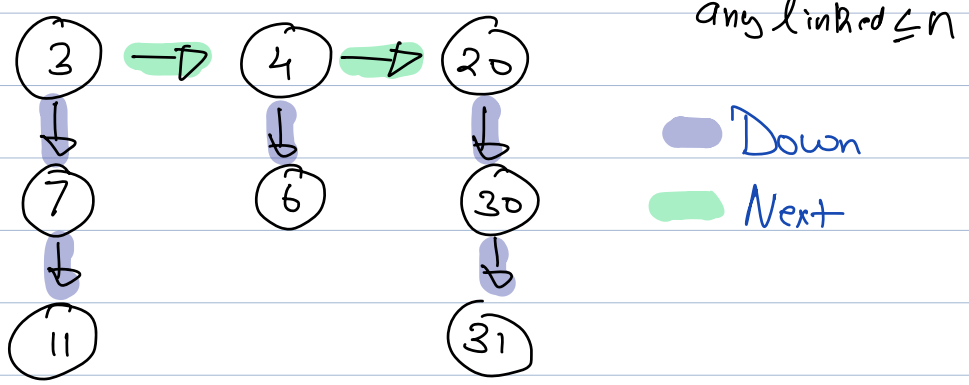
down pointer : Pointer to the linked list where this node is the head.

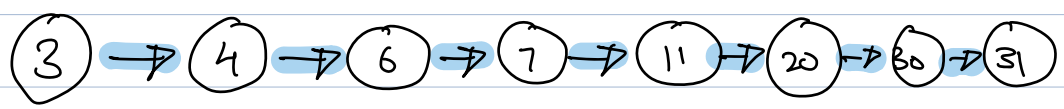
All linked list are sorted.

Flatten everything to a single sorted list. Use down pointers to link node in the flattened list.

length of any linked $\leq n$

Ex1

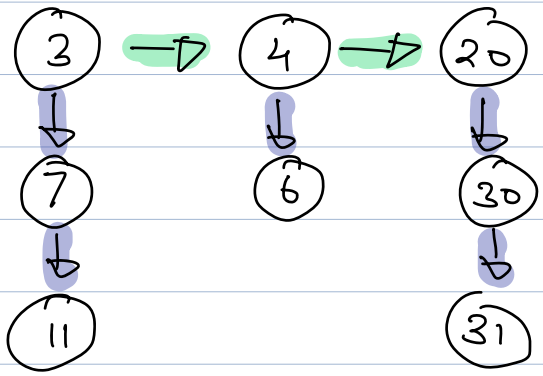


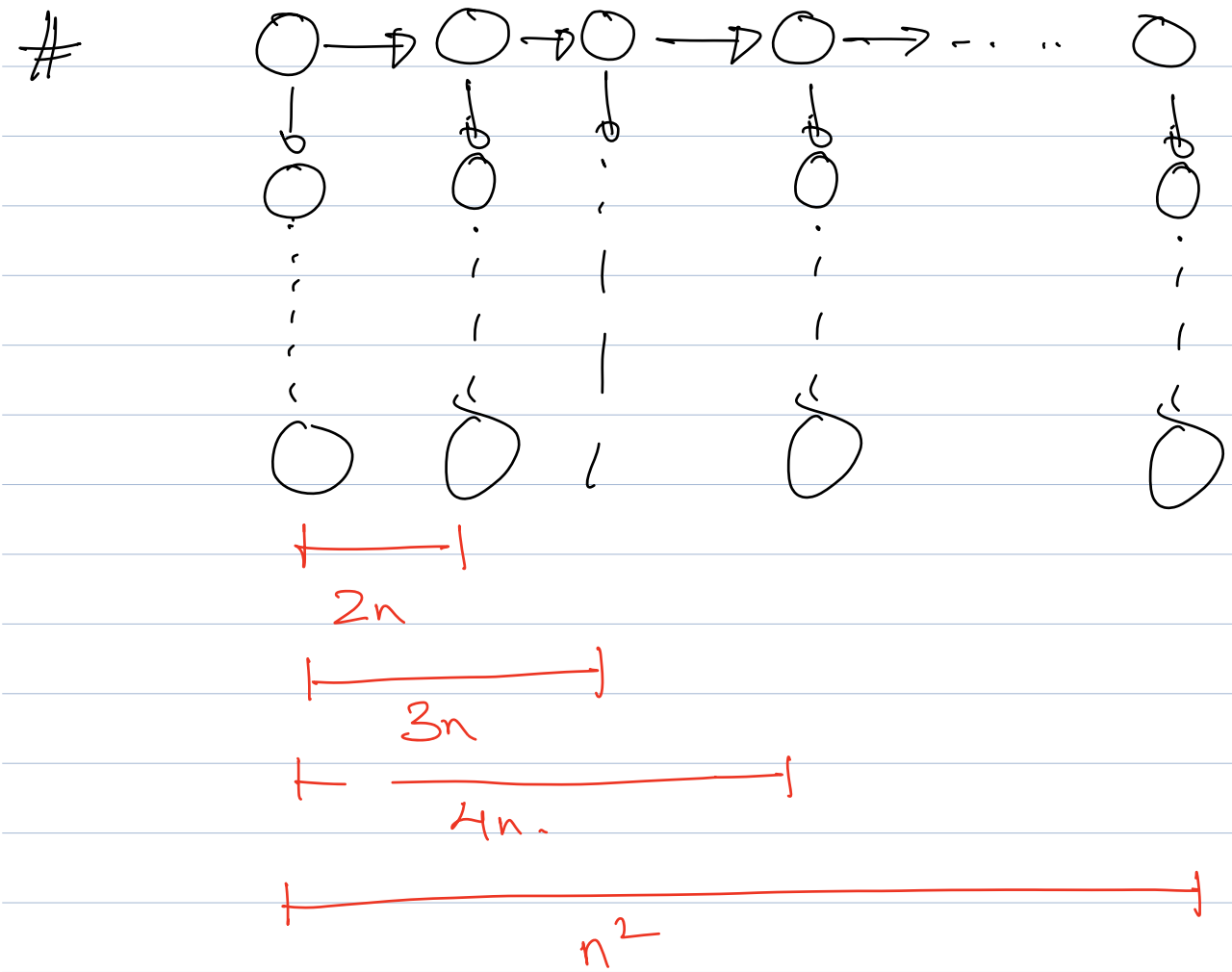
⇒ 

head

Down pointers

Approach 1 :



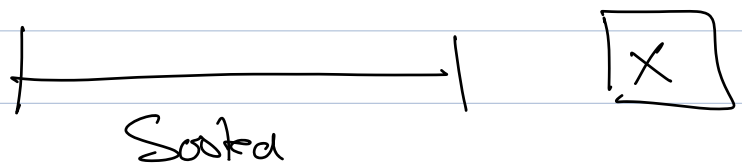


TC: $2n + 3n + 4n + n^2$

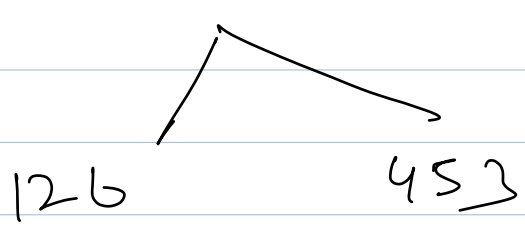
n { $2 + 3 + 4 \ldots n$ }

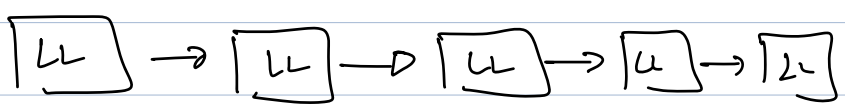
$n \times n^2 \Rightarrow n^3$

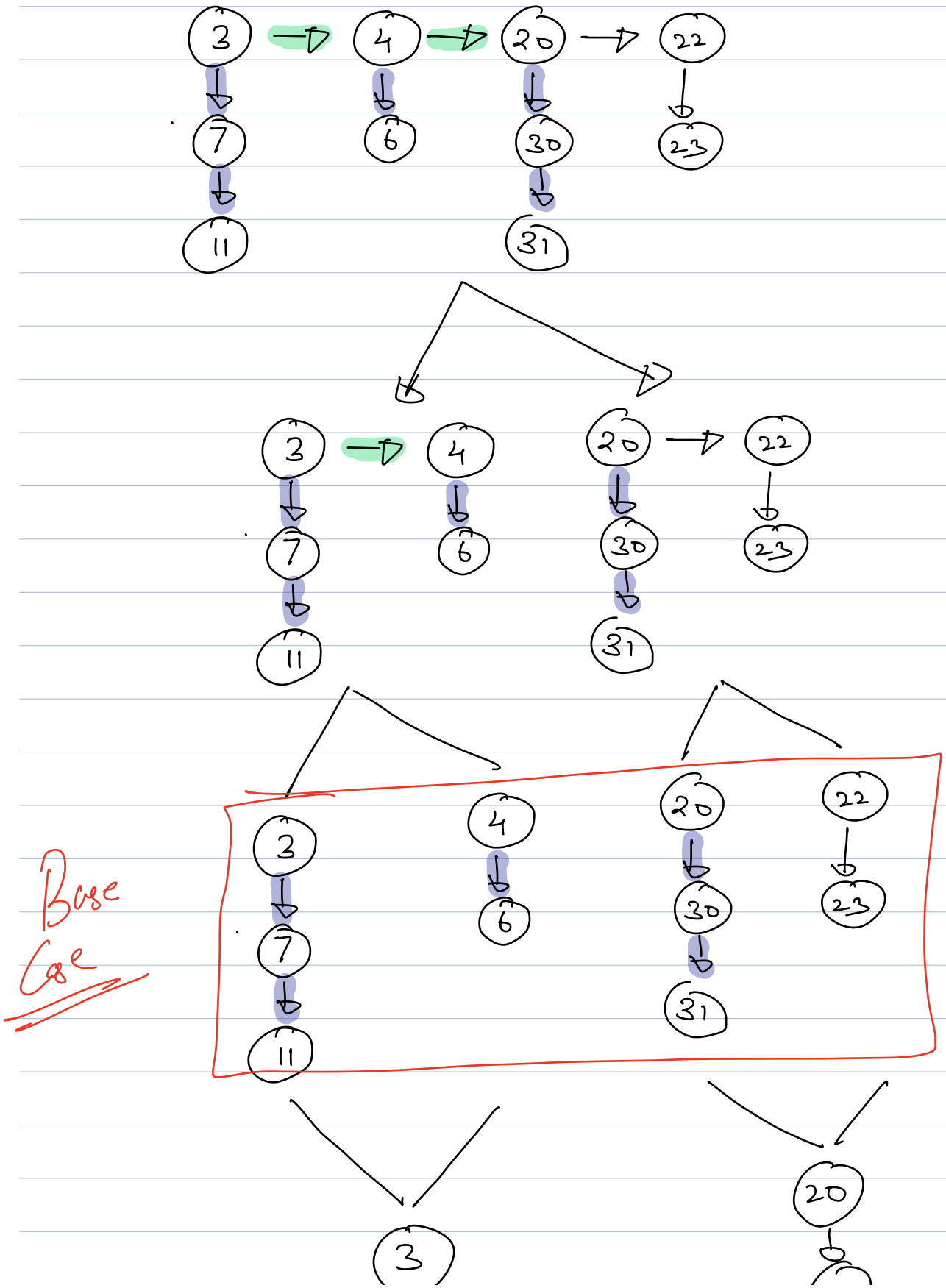
Insertion Sort

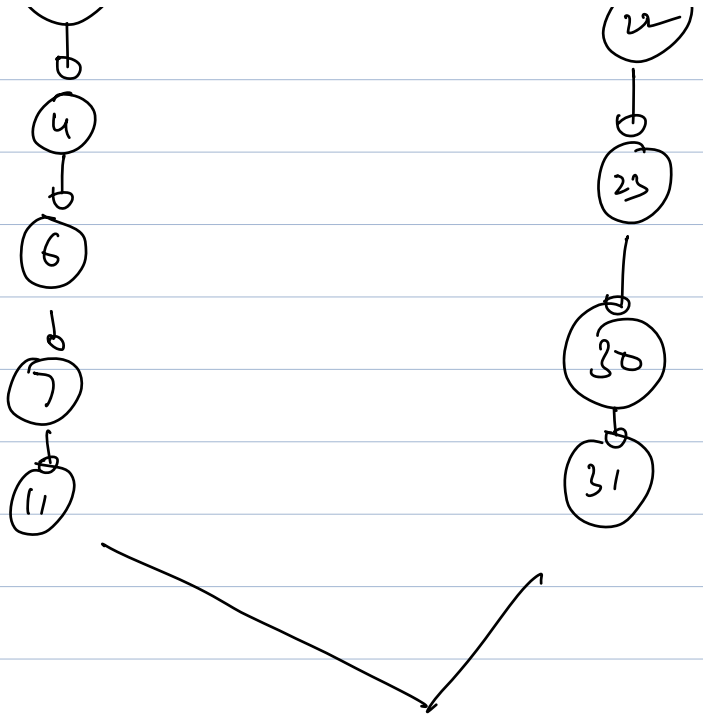


1 2 6 4 5 3



\# 


3
4
20
22
7
6
30
23
11
31
3
4
20
22
7
6
30
23
11
31
Base Case
3
4
20
22
7
6
30
23
11
31
3
20



Total Elements $\Rightarrow x$

TC $\Rightarrow x \log x$

$\Rightarrow n^2 \log n^2$

$2n^2 \log n$

$\simeq n^2 \log n$

$x = n^2$

SC: $\log n$

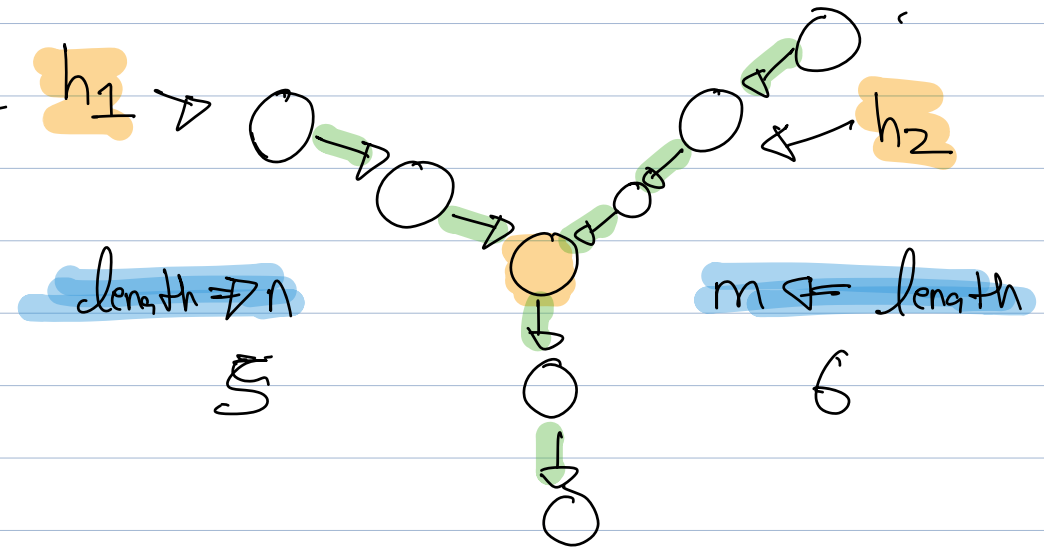
$T(x) \Rightarrow x + 2T(x/2)$

Pseudo Code

```
Node flatten ( Node head) {

    if ( head == null || head.next == null)
        return head;

    Node mid = find Middle Node (head);
    Node h2 = mid.next
    mid.next = null

    Node head = flatten (head);
    Node h2 = flatten (h2);

    return merge (head, h2);
}
```

Q: Given a Y shaped Linked list. Find the intersection.

Ex1



Approach 1: Find length of both Linked list.

$$diff = |m - n|$$

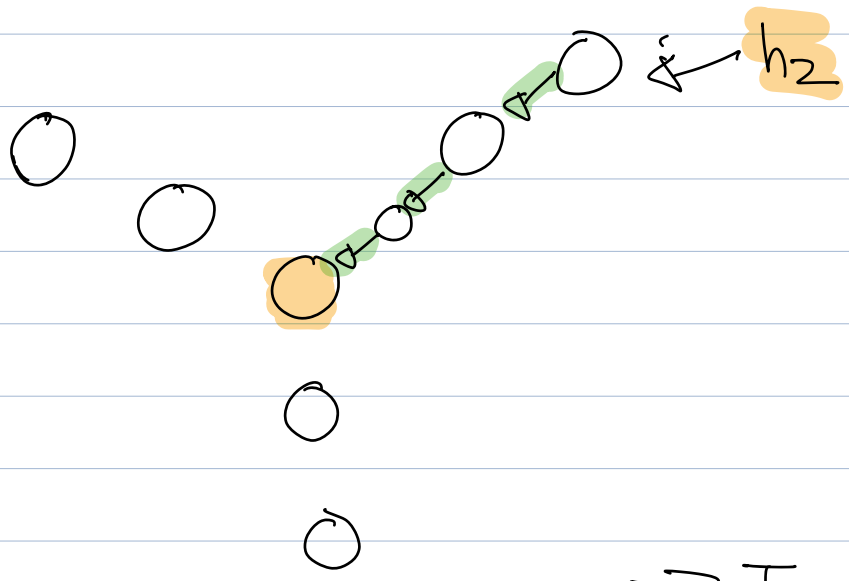
⇒ traverse $|m-n|$ nodes in the larger list.

TC: $O(m+n)$
SC: $O(1)$

Approach 2: Hashset.

TC: $O(m+n)$
SC: $O(\min(m,n))$

Approach 3



⇒ TC: O(m+n)
SC: O(1)

We are destructing the linked list.

Approach 4