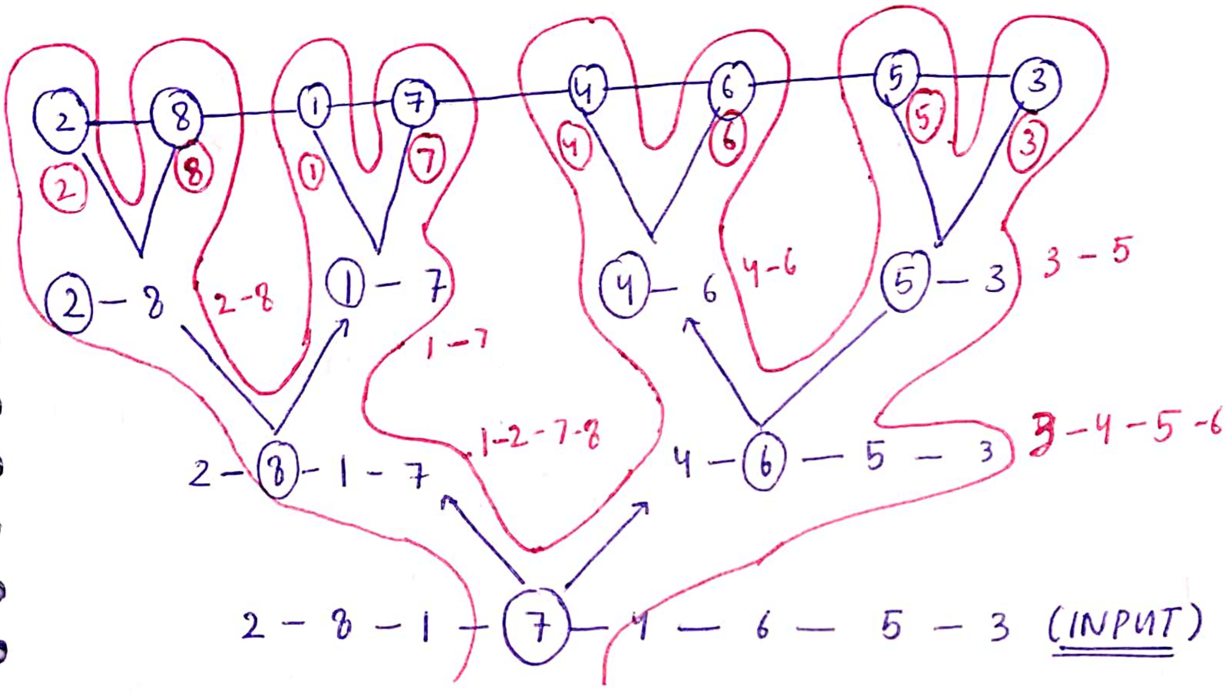
③ Merge Sort a Linked List



1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 (OUTPUT)

T(n) = O(n)

Time to sort Linked List from [head to tail]

T(n) = 2 * T

[T(n/2) = 2 *

[T(n/4) = 2* T

⋮

[T(1) = 2 * T

T(n) = $2^{\log_2 n}$ *

T(n) = 2 * n

T(n) = 2n +

T(n) = O(n

Bes

Time

```
public static Node midNode (Node head, Node tail) {
  Node f = head;
  Node s = head;
  while ( f != tail && f.next != tail) {
       f = f.next.next;
       s = s.next;
   }
   return s;
 }
public static LinkedList mergeSort (Node head, Node tail) {
     if ( head == tail) {
      LinkedList br = new LinkedList ();
      br.addLast (head.data);
      return br;
     }
  Node mid = midNode (head, tail);
  LinkedList fsh = mergeSort (head, mid);
  LinkedList ssh = mergeSort (mid.next, tail);
  LinkedList sl = mergeTwoSortedLists (fsh, ssh);
  return Sl;

  }
}
```

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

Time to sort Linked list from [head to tail]

Time to [redacted] Find mid node of LinkedList

Time to sort [head, mid] & [mid.next, tail] respectively

Time to merge 2 sorted LinkedList

$$T(n) = 2 * T(n/2) + 2 * O(n)$$

$$[T(n/2) = 2 * T(n/4) + 2 * O(n/2)] * 2$$

$$[T(n/4) = 2 * T(n/8) + 2 * O(n/4)] * 4$$

$$\vdots$$

$$[T(1) = 2 * T(0) + 2 * O(1)] * 2^{\log_2 n}$$

$$T(n) = 2^{\log_2 n} * (2T(0)) + \left\{ 2n + \frac{4n}{2} + \frac{8n}{4} + \dots 2 \cdot 2^{\log_2 n} \right\}$$

$$T(n) = 2 * n + \{ 2n + 2n + \dots (\log_2 n) \text{ times} \}$$

$$T(n) = 2n + \{ 2n * \log_2 n \}$$

$$T(n) = O(n\log_2 n + n) \simeq \boxed{O(n\log_2 n)}$$

Best & average & worst Case Time Complexity of Merge Sort Linked List.