Lesson-9 – Merge Sort

Divide & Conquer Technique

- Merge-sort of an array by partitioning into segments of the input array
- Merge-sort on an input sequence ***S*** with ***n*** integers consists of three steps:
    - Divide: partition ***S*** into two segments of about ***n***/2 elements each (lo..mid) and (mid+1..hi)
    - Conquer: recursively sort the two segments.
    - Combine: merges the two segments back into S in the merge step

Dividing Process: Partition into n/2 elements, recursively call the Mergesort on each partition until list size becomes 1. Then merge the elements in the sorted order.

If array size n is ODD,
Input = [4 7 2 1 3 5 9]. Mid = Int(n/2) = 7/2 = 3, n = Number of elements. N = arr.length

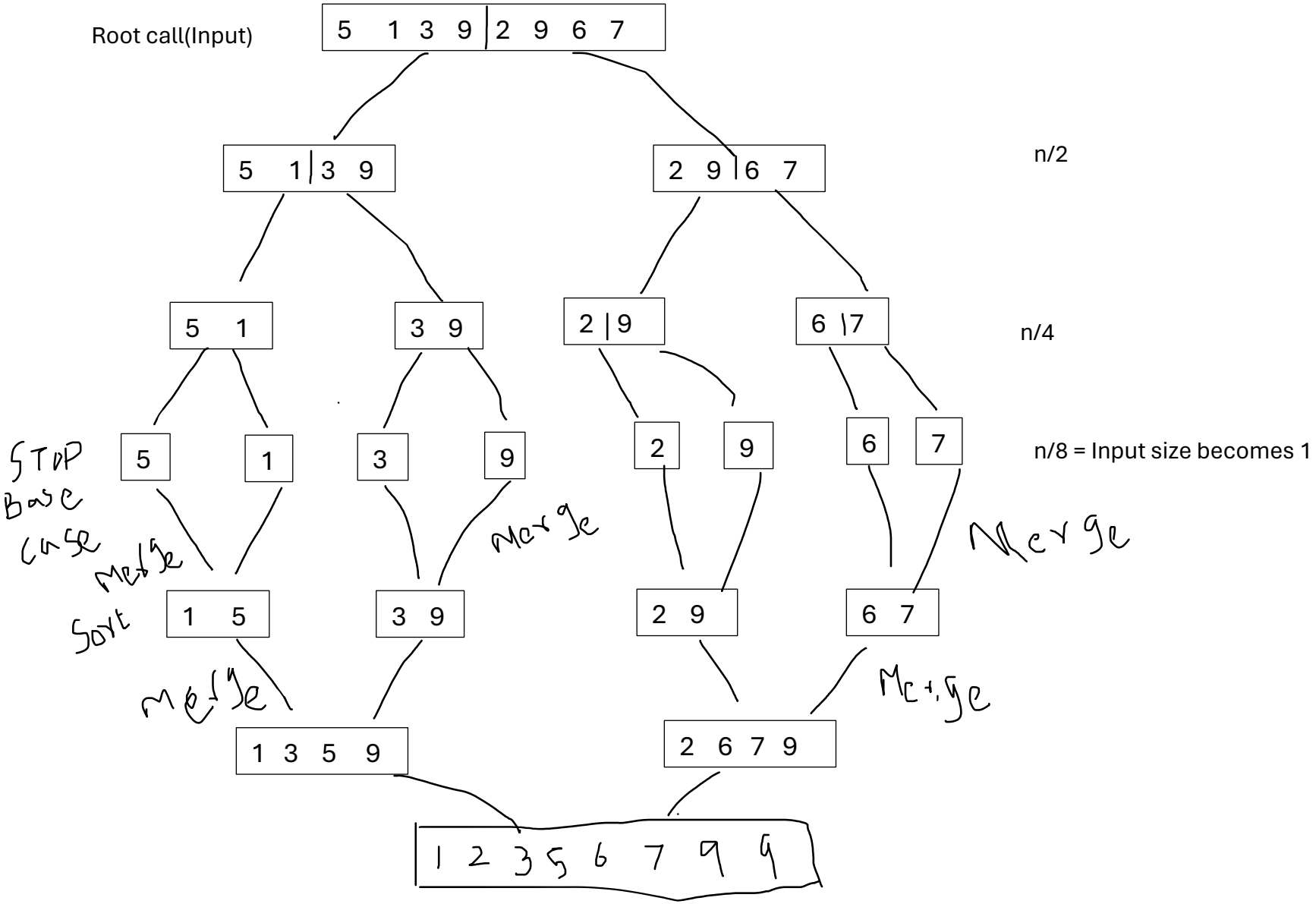
Index 0 1 2 3 4 5 6

Left = 0 to mid, right mid+1 to high

Left [4 7 2 1] Right = [3 5 9]

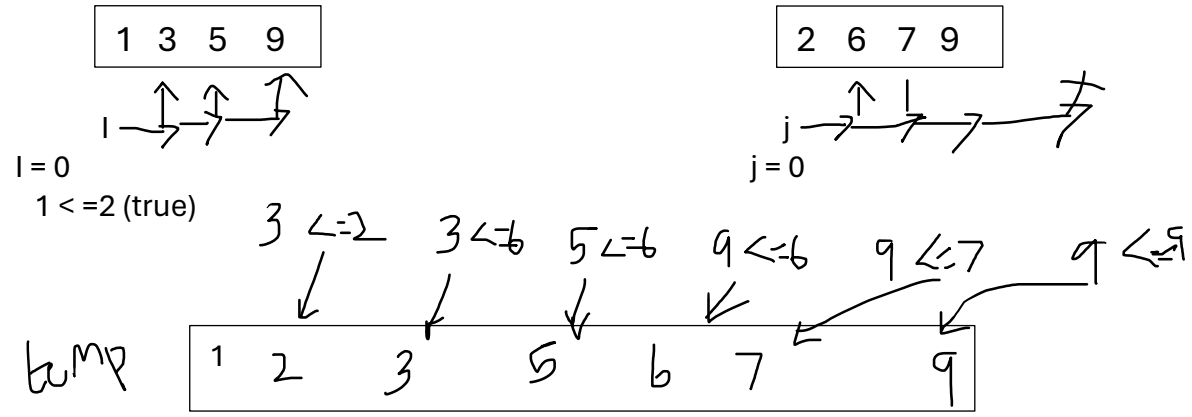
Input Array: 5 1 3 9 2 9 6 7 n=8

Dividing Part



Merging Part

1. You have to maintain two index pointers i(left) and j(right)
2. Compare the i index value with the j index value.
3. If the index value i position is has a small value or equal, copy into temp array and increment the i value. j is in the same position.
4. If the index value I position is has a greater value or equal copy into temp array and increment the j value. i is in the same position.
5. Copy the remining elements left over into temp array after Step 3 & 4.



Run time Analysis:

- Any path in the tree from root to a leaf node (representing a list with length either 0 or 1) involves list lengths n, n/2, n/4,..., 1 (or 0).
- Therefore, height of recursion tree is Θ(log n)
- Total work done at each level (partition, merge) is Θ(n)
- Thus, the total running time of merge-sort is Θ(n log n)

Lesson-9- Review Questions

1. What is Divide and Conquer strategy. (Slide-2)
2. How to perform Merge-sort using tree approach.
3. How recursion used in Merge Sort.
4. How to analyze the performance of Mergesort-Tree.(Slide-17)