Understanding Merge Sort: Recursion and Merging

1. Partitioning (Breaking into Subarrays)

Recursion Begins with Division: The merge sort algorithm starts by recursively dividing the array into smaller subarrays. This division continues until each subarray contains only one element. The recursive `merge-sort` function calls itself to handle the left and right halves of the array.

Recursive Calls Create the Subarrays: These recursive calls keep breaking down the array into smaller parts:

- For example, an array $\{38, 27, 43, 3, 9, 82, 10\}$ is first split into $\{38, 27, 43\}$ and $\{3, 9, 82, 10\}$.
- $\{38, 27, 43\}$ is then split into $\{38\}$ and $\{27, 43\}$.
- $\{27, 43\}$ is further split into $\{27\}$ and $\{43\}$.

2. Recursion "Bottoms Out"

Base Case (Single Element Subarrays): The recursion bottoms out when the array is split into single-element subarrays. At this point, the recursive calls no longer divide the array, as a single element is inherently sorted.

3. Merging Begins

Merge Work Starts with the Smallest Subarrays: Once the recursion bottoms out, the merge process begins. The merge operation starts with the smallest subarrays—those containing only one element each.

- For example, $\{27\}$ and $\{43\}$ are merged into $\{27, 43\}$.

Recursive Unwinding: As the recursion stack unwinds, the merge function is called to combine the results of the previous merges:

- $\{27, 43\}$ is merged with $\{38\}$ to form $\{27, 38, 43\}$.
- On the other side, $\{3\}$ and $\{9\}$ merge to form $\{3, 9\}$, which then merges with $\{10, 82\}$ to form $\{3, 9, 10, 82\}$.

4. Final Merge (Combining the Halves)

Combining Larger Subarrays: The merging continues up the recursion stack, combining larger and larger subarrays. The final step merges the two halves of the array:

- The subarrays $\{27, 38, 43\}$ and $\{3, 9, 10, 82\}$ are merged to form the fully sorted array $\{3, 9, 10, 27, 38, 43, 82\}$.

5. Selection Process During Merging

Selecting Elements for Merging: During the merging process, elements from the left and right subarrays (`L` and `R`) are compared, and the smallest element is selected and copied back into the original array `A`. This is like sorting face-up cards by repeatedly picking the smallest from two piles.

Final Sorted Array: The process culminates in the final merge step, where the last two large subarrays are merged, producing the sorted array.

Summary of the Process

Partitioning/Division: This occurs first, with the array being recursively split until each subarray has only one element.

Recursion Bottoms Out: The base case is reached when subarrays cannot be split further.

Merging: The actual sorting happens during the merge phase, which begins as the recursion unwinds. Small subarrays are merged first, progressively leading to the merging of larger subarrays.

Final Merge: The last step combines the two halves of the array into a fully sorted array.