Understanding ”c” and ”cn” in Time Complexity Analysis

”c” (The Constant Factor)

The constant c represents a fixed amount of work done for each operation, independent of the size of the input. It could be the time taken to compare two elements, swap them, or perform any other basic operation in the algorithm.

Why It’s Ignored in Big O Notation: In Big O notation, we focus on how the running time scales with the input size, not the exact number of operations. Since c is a constant and does not change with n, it does not impact the growth rate of the algorithm.

”cn” (The Work Done at Each Level)

The term cn represents the work done at each level of the recursion tree in merge sort. Here, n is the size of the input at that level, and c is the constant work done for each element.

Why It’s Important: Since the amount of work done at each level is proportional to n, and there are $\log n$ levels, the total work is $cn \times \log n$. This is crucial because it reflects how the algorithm’s running time scales with the input size.

Putting It All Together

Total Work Calculation

The total work done by merge sort can be expressed as:

$$T(n) = cn \log n + cn$$

- The first term $cn \log n$ represents the work done across all levels of the recursion tree.
- The second term cn represents any additional work done at the base level or to combine the results at the end.

Simplification in Big O Notation

In Big O notation, we drop the constant c and lower-order terms because they don’t affect the overall growth rate as n becomes large. Thus:

$$T(n) = O(n \log n)$$

The $+n$ term is dropped because $n \log n$ dominates n for large n, and Big O notation only captures the dominant term.

Summary

- c is a constant that represents fixed work per operation and is ignored in Big O notation.
- cn represents the work at each level of the recursion tree and is used to calculate the overall time complexity.
- The final Big O notation drops constants and lower-order terms, focusing on how the algorithm scales with input size, leading to $O(n \log n)$ for merge sort.