"Swipe & Master It!"

Time & Space Complexity in DSA



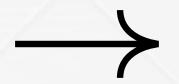
Written By **@CodeVish**



Understanding Time & Space Complexity in DSA

When solving problems in Data Structures & Algorithms (DSA), understanding Time Complexity and Space Complexity is crucial. These concepts help us analyze the efficiency of our code and optimize solutions for scalability.





Time Complexity

- O(1) Constant Time: Execution time remains the same, regardless of input size.
- O(log n) Logarithmic Time: Fast-growing input size results in a much smaller number of operations (e.g., Binary Search).
- <u>O(n)</u> Linear Time: Execution grows proportionally with input size (e.g., iterating through an array).
- $O(n^2)$ Quadratic Time: Performance decreases significantly with large inputs (e.g., nested loops in Bubble Sort).



 \rightarrow

Space Complexity

This determines how much additional memory an algorithm needs. It considers:

Auxiliary Space — Extra memory used beyond the input storage.

Recursive Call Stack — Recursive solutions can increase space usage (e.g., DFS in trees).



Written By **@CodeVish**

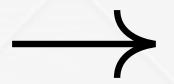




Why does this matter?

Optimizing time & space complexity makes code faster, scalable, and cost-effective - critical for real-world applications, from search engines to Al models.



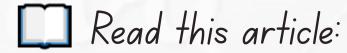


Check out this great resource :



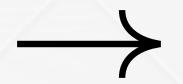
Watch this video:

Time & Space Complexity Explained - Kunal Kushwaha



Big-O Notation Guide — GeeksforGeeks





Follow for more tips!!



Vishnu

Software Developer





Like if you learn something new, follow if you got value