Time and Space Complexity in DSA - Cheat Sheet

1. What is Time Complexity?

Time complexity is the amount of time an algorithm takes to run as a function of the input size 'n'. It reflects how the number of operations grows with the input.

2. What is Space Complexity?

Space complexity is the total memory used by an algorithm including input and auxiliary space. It helps evaluate memory efficiency.

3. Common Time/Space Complexities

O(1) - Constant Time: No matter the input size, time remains constant.

O(log n) - Logarithmic Time: Input is reduced by half in each step (e.g., Binary Search).

O(n) - Linear Time: One operation per input element.

O(n log n) - Log-linear Time: Sorting algorithms like Merge Sort.

O(n^2) - Quadratic Time: Nested loops, e.g., comparing all pairs.

O(m + n) - Sum of sizes of two inputs.

O(2^n) - Exponential Time: Recursive calls that double each step.

O(n!) - Factorial Time: Permutations and combinations.

4. Identifying Time Complexity in Code

Single loop: O(n)

Nested loop: O(n^2)

Separate loops: O(m + n)

Binary Search: O(log n)

Recursive Tree (e.g., Fibonacci): O(2^n) unless optimized with memoization.

5. How to Remember Forever

Use analogies:

- O(1): Fixed strike (array access).

Time and Space Complexity in DSA - Cheat Sheet

- O(log n): Halving a cake (binary search).
- O(n): Scanning row (loop).
- O(n^2): Everyone meets everyone (nested loops).
- O(n log n): Divide & conquer (merge sort).
- O(2^n)/O(n!): Every path considered (recursion/permutations).

6. Input Constraints Guide

n <= 10^6: O(n log n) or better.

n <= 100: O(n^2) is okay.

n <= 20: O(2^n) or O(n!) is acceptable.

7. Final Problem Solving Framework

Ask yourself:

1. What is the size of input?
2. What operations are repeated?
3. Can I optimize with sorting, hashing, or prefix/suffix?

Use a cheat table to quickly identify patterns.