DSA in Python Cheat Sheet

by aman-senpai via cheatography.com/191339/cs/39766/

Complexity Analysis (Time/Space)

Complexity	Notation	Description
O(1)	Constant	Time/Space complexity remains constant regardless of input.
O(log n)	Logarithmic	Time/Space complexity grows logarithmically with the input.
O(n)	Linear	Time/Space complexity grows linearly with the input.
O(n log n)	Linearithmic	Time/Space complexity grows linearly multiplied by log n.
O(n^2)	Quadratic	Time/Space complexity grows quadratically with the input.
O(n^3)	Cubic	Time/Space complexity grows cubically with the input.
O(2^n)	Exponential	Time/Space complexity grows exponentially with the input.
O(n!)	Factorial	Time/Space complexity grows factorially with the input.

Big O Notation
Big O notation is used to analyze the efficiency of algorithms and represent their upper-bound time complexity. It helps us understand how the algorithm's performance scales with the input size.

Best, Average, and Worst Case Complexity

Case	Description
Best Case	The minimum time complexity under optimal input.
Average Case	The expected time complexity for random input.
Worst Case	The maximum time complexity for any input.

To determine the complexity of an algorithm, follow these steps:

1. Identify the major operations in the algorithm.
2. Analyze how many times each operation is executed concerning the input size.
3. Eliminate lower-order terms and constants to find the dominant term.

Choose the appropriate complexity notation from the Big O table.