

How fast is your code? How hungry is your code? ■■

■ What You Will Learn

- What Big-O notation means in plain English
- The most common complexities: $O(1)$, $O(n)$, $O(n^2)$, $O(\log n)$
- How to calculate complexity of simple code
- Space complexity — memory usage
- Best, worst, and average case

■ Intuition First

Time complexity is NOT about seconds. It's about **how much slower your code gets as input grows**. If you double the input, does runtime double? Or quadruple? That's what Big-O tells you.

■ Think Like This

Finding a word in a book: $O(n)$ = read every page. $O(\log n)$ = use the index (binary search). $O(1)$ = you already know the page number. Bigger input, bigger difference between these approaches.

■ The Complexity Ladder

FASTEST SLOWEST

[illegible]

$O(1)$ $O(\log n)$ $O(n)$ $O(n \log n)$ $O(n^2)$ $O(2^n)$

Example speeds for $n = 1,000$:

$O(1) \rightarrow 1$ operation

$O(\log n) \rightarrow 10$ operations

$O(n) \rightarrow 1,000$ operations

$O(n \log n) \rightarrow 10,000$ operations

$O(n^2) \rightarrow 1,000,000$ operations $\leftarrow$ gets slow FAST

$O(2^n) \rightarrow \text{astronomical!}$

■ ■ How to Calculate — Step by Step

■ Common Mistakes

■ Confusing time with actual speed

$O(n^2)$ with $n=10$ might be faster than $O(n)$ with $n=1,000,000$. Big-O is about growth rate, not absolute speed.

■ Not simplifying correctly

$O(n + 500)$ is NOT $O(500)$. Drop constants to get $O(n)$. But $O(n + n^2)$ drops the smaller term to get $O(n^2)$.

■ Practice Problems

#	Problem	Difficulty
1	Identify complexity of a loop printing 1 to n	Easy
2	What's the complexity of two separate loops (not nested)?	Easy
3	Calculate complexity of bubble sort (nested loops over n)	Medium
4	Explain why binary search is $O(\log n)$	Medium
5	Given pseudocode, identify time and space complexity	Medium

■ Quick Revision Box

- ✓ $O(1)$ = constant: doesn't grow with input
- ✓ $O(n)$ = linear: one loop over input
- ✓ $O(n^2)$ = quadratic: loop inside a loop
- ✓ $O(\log n)$ = logarithmic: input halved each step
- ✓ Drop constants and smaller terms
- ✓ Space complexity = memory used (not time)
- ✓ Worst case = what we usually analyze

■ Visual Thinking

■ Imagine It Like This

Plot these on a graph with x = input size, y = time. $O(1)$ is a flat line. $O(n)$ is a gentle slope. $O(n^2)$ is a steep curve that shoots up fast. $O(2^n)$ goes vertical almost instantly. Animation: Show all 4 curves growing as n increases — watch $O(n^2)$ and $O(2^n)$ leave the others in the dust.