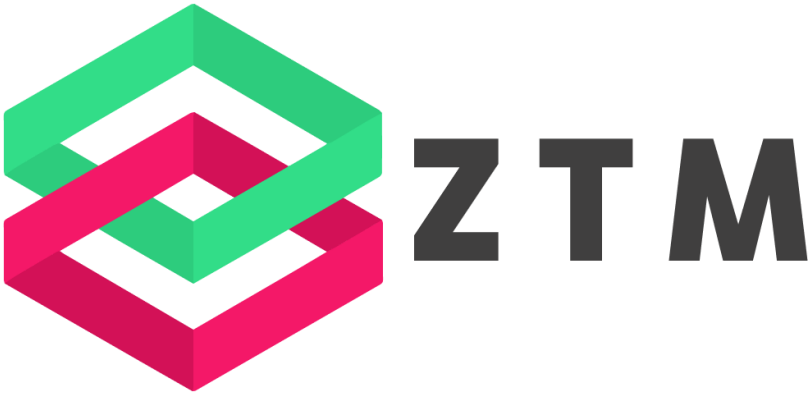

Big O Cheat Sheet

ANDREI NEAGOIE

V1.01

HEEELLLOOOOO!

I'm Andrei Neagoie, Founder and Lead Instructor of the Zero To Mastery Academy.

After working as a Senior Software Developer over the years, I now dedicate 100% of my time teaching others valuable software development skills, help them break into the tech industry, and advance their careers.

In only a few years, **over 600,000 students** around the world have taken my courses and many of them are now working at top tier companies like Apple, Google, Amazon, Tesla, IBM, Facebook, and Shopify, just to name a few.

This cheatsheet provides you with the key Big O concepts that you need to know and remember especially if you're currently interviewing at any top tech companies.

If you want not only learn data structures and algorithms (and Big O) but also the exact steps to take to get more interviews, more job offers, and a higher salary, then check out my Ultimate Coding Bootcamp.

Happy Coding!
Andrei

Founder & Lead Instructor, Zero To Mastery
Andrei Neagoie

P.S. I also recently wrote a book called Principles For Programmers. You can download the first five chapters for free here.

Big Os

O(1) Constant – no loops

O(log N) Logarithmic – usually searching algorithms have log n if they are sorted (Binary Search)

O(n) Linear – for loops, while loops through n items

O(n log(n)) Log Linear – usually sorting operations

O(n^2) Quadratic – every element in a collection needs to be compared to ever other element. Two nested loops

O(2^n) Exponential – recursive algorithms that solves a problem of size N

O(n!) Factorial – you are adding a loop for every element

Iterating through half a collection is still O(n)

Two separate collections: O(a * b)

What Can Cause Time in a Function?

Operations (**+**, **-**, *****, **/**)

Comparisons (**<**, **>**, **==**)

Looping (**for**, **while**)

Outside Function call (**function()**)

Rule Book

Rule 1: Always worst Case

Rule 2: Remove Constants

Rule 3:

- Different inputs should have different variables: **O(a + b)**
- A and B arrays nested would be: **O(a * b)**

+ for steps in order

* for nested steps

Rule 4: Drop Non-dominant terms

What Causes Space Complexity?

- Variables
- Data Structures
- Function Call
- Allocations