Big O Cheat Sheet

We created this Big O Cheat Sheet initially for students of Master the Coding Interview: Data Structures + Algorithms (my Coding Interview Bootcamp course) but we're now sharing it with any and all Developers that want to learn and remember the basics of Big O. If you'd like to download a PDF version of this Big O Sheet, enter your email below and we'll send it to you!

Download This Big O PDF Cheatsheet For Free

surya.aug4@gmail.com

EMAIL ME THE BIG O PDF CHEATSHEET

YOUR BIG O PDF CHEATSHEET IS ON ITS WAY TO YOUR INBOX!

If you've stumbled across this page and are just starting to learn Big O, nice work! Big O is one of the most important topics for any software developer or engineer. Even if (... when!) you're coding 10 years from now, this is a topic that will be around for a long time and will make you a better developer. This was true for me and will be for you as well.

Many of the biggest tech companies (Google, Amazon, Facebook... aka Meta, etc.) ask questions about Big O as part of their interview process for good reason.

If you're currently stuck in an endless cycle of YouTube tutorials and not having success getting hired as a developer at the company of your dreams, I can help. Come join the Zero To Mastery Academy and take my Ultimate Coding Interview Bootcamp. You'll 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.

I've also made the entire Big O introduction section of my Coding Interview Bootcamp completely free (no signup or credit card necessary) so that you can learn more about Big O for free.

Please enjoy this cheatsheet and if you'd like to submit any corrections or suggestions, feel free to email us at support@zerotomastery.io.

CONTENTS

What Can Cause Time in a Function:

Operations, **Comparisons**, **Looping**, **Outside Function call**

Rule Book:

Rule 1, **Rule 2**, **Rule 3**, **Rule 4**

What Causes Space Complexity:

Variables, **Data Structures**, **Function Call**, **Allocations**

BIG O'S

`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

R

+ for steps in order

* for nested steps

Rule 4: Drop Non-dominant terms

WHAT CAUSES SPACE COMPLEXITY?

- Variables
- Data Structures
- Function Call
- Allocations

Back To Top