



Evaluating Algorithms



01

ANALYSIS & MEMORY

02

USE CASES &
PRACTICE

03

MOVING FORWARD



01

Complexity Analysis

What is Complexity Analysis?

“The process of determining the efficiency of an algorithm.”

Time and Space Complexity

Time Complexity describes how quickly an algorithm runs

Space Complexity describes how much memory an algorithm takes up

- ★ The faster an algorithm runs, the better solution it is
- ★ The less space an algorithm takes up, the better solution it is
- ★ The specific use case will determine which algorithm is better when space-time complexity is equal

Big O Notation

A measure of how quickly the runtime grows relative to the input, as the input gets arbitrarily large.

1. **Constant:** $O(1)$
2. **Logarithmic:** $O(\log(n))$
3. **Linear:** $O(n)$
4. **Log-linear:** $O(n \log(n))$
5. **Quadratic:** $O(n^2)$
6. **Cubic:** $O(n^3)$
7. **Exponential:** $O(2^n)$
8. **Factorial:** $O(n!)$

These are examples of common complexities from fastest to slowest.

Big O Notation

```
def say_hi_n_times(n):  
    for time in xrange(n):  
        print "hi"
```

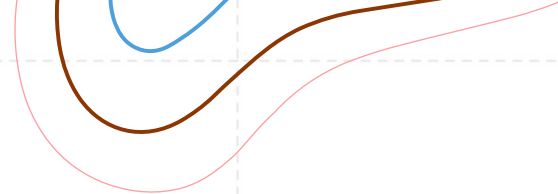
This function takes $O(1)$ space.
It takes a fixed number of variables,
so the space complexity is constant

```
def list_of_hi_n_times(n):  
    hi_list = []  
    for time in xrange(n):  
        hi_list.append("hi")  
    return hi_list
```

This function takes $O(n)$ space, also
called "linear time complexity."
The size of `hi_list` scales with the
size of the input.

Worst Case

Worst case analysis gives the **maximum number of basic operations** that have to be performed during execution of the algorithm.



02

Case Studies



When will you use this?

1. *When solving a problem, it will help to identify the best Data Structure to use based on time-space complexity*
2. In industries with regular routines (ex: **Healthcare**): how long a routine will take, or how much memory (on/offline) it will require.
3. **Web developers**: You want web apps to scale. If your app has a bottleneck that scales with $O(n^2)$, to handle just twice as many, you'll need 4 times the computational power.

Interview Question Practice

1. Provide an example of $O(1)$ algorithm
2. What is complexity of this code snippet?
for (int i = 0; i < n; i++){
if(array[i] == numToFind){ return i; }}
3. What is complexity of push and pop for a Stack implemented using a LinkedList?
4. What is the big O notation of this function?
 $f(x) = \log n + 3n$

The image features decorative contour lines in the corners. In the top-left, there are blue, brown, and pink lines. In the bottom-left, there are blue, brown, and orange lines. The background is a light gray grid.

03

Moving Forward

Today & Onwards

Practice

> Review practice problems to understand Big O in practice

Application

> think of examples in your industry where Big O might be useful

Prepare

> Review concepts before interviews which include coding

Practice

1. <https://leetcode.com/problemset/algorithms/>
2. <https://www.fullstack.cafe/interview-questions/big-o-notation>
3. <https://devinterview.io/blog/bigONotation-interview-questions/29/>
4. <https://betterprogramming.pub/use-big-o-notation-to-design-better-algorithms-9e5769c2e47b> (Examples)
5. <https://www.quora.com/Is-Big-O-notation-something-that-is-discussed-in-the-real-world-or-is-it-a-construct-of-academia-What-I-mean-is-do-programmers-in-industry-and-the-work-world-discuss-their-algorithms-in-terms-of-Big-O> (Real world applications)

Recommended books

Cracking the code interview - Gayle laakmann Mcdowell

Elements of programming interviews - Bunch of people

Introduction to algorithms - Cormen



THANKS!

mariana@psychonautgirl.space

CREDITS: This presentation template was
created by **Slidesgo**, including icons by **Flaticon**,
infographics & images by **Freepik**