FREE eBook

# LEARNING time-complexity

Free unaffiliated eBook created from **Stack Overflow contributors.**

#time-complexity

# Table of Contents

# About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: time-complexity

It is an unofficial and free time-complexity ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official time-complexity.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapter 1: Getting started with time-complexity

## Remarks

This section provides an overview of what time-complexity is, and why a developer might want to use it.

It should also mention any large subjects within time-complexity, and link out to the related topics. Since the Documentation for time-complexity is new, you may need to create initial versions of those related topics.

## Examples

### Installation or Setup

Time complexity is a property of

- **Problems** someone might want to solve computationally,
- **Algorithms** designed to solve such problems and
- **Programs** implementing such algorithms.

An abstract concept requires no installation or setup. Simply take any problem, algorithm, or code and ask "How long will this take?”

### Hello, world!

```
echo "Hello, world!"
```

Even in bash, this program works similarly in most other languages. The program has no input and will always function the same in an idealized world - run time should never change. Thus Hello World has *constant complexity*.

Almost all elementary operations are assumed to have constant complexity. This forms the basic building blocks of most programs.

Read Getting started with time-complexity online: https://riptutorial.com/time-complexity/topic/6948/getting-started-with-time-complexity

# Chapter 2: Landau Notation

## Remarks

All five classes in the Landau system describe *asymptotic* behaviour, i.e. the behaviour when the size of the problem tends to infinity. While this might look irrelevant to our – very finite – real world problems, experience has shown that behaviour of real world algorithms mirrors this infinite behaviour well enough on real data to be of practical use.

## Examples

### Big O

Big O notation provides upper bounds for the growth of functions. Intuitively for `f ∊ O(g)`, `f` grows *at most* as fast as `g`.

Formally `f ∊ O(g)` if and only if there is a positive number `C` and a positive number ``n such that for all positive numbers `m > n` we have

$$C \cdot g(m) > f(m).$$

# Intuition of this definition

## Intuition regarding C

`C` is responsible for swallowing constant factors in the functions. If `h` is two times `f`, we still have `h ∊ O(g)` since `C` can be twice as big. For this there are two rationales:

- Easier notation: `f ∊ O(n)` is preferable to `f ∊ O(7.39 n)`.
- Abstraction: Any units of time are swallowed in these considerations because there is nothing to gain from them; they differ between machines and the algorithms can be evaluated free of that. Since `C` swallows constant factors, the complexity classes stay the same even on a machine ten times as fast.

## Intuition regarding n

`n` is responsible for swallowing initial turbulences. One algorithm might have an initialization overhead that is enormous for small inputs, but pays off in the long run. The choice of `n` allows sufficiently big inputs to get the focus while the initial stretch is ignored.

## Intuition regarding m

`m` ranges over all values greater than `n` - to formalize the idea “from `n` onwards, this holds”.

Read Landau Notation online: https://riptutorial.com/time-complexity/topic/6951/landau-notation

# Credits

| S. No | Chapters | Contributors |
|---|---|---|
| 1 | Getting started with time-complexity | 4444, Community, Hermann Döppes |
| 2 | Landau Notation | 4444, Hermann Döppes |