# 1 Big-O Notation

This document contains some basic proofs related to Big-O Notation.

## 1.1 Introduction

- We use Big-O notation to express the amount of resources used by algorithms (time complexity and space complexity).
- Denote by $f(N)$ the running time of a program for inputs of size $N$.
  - Ex 1. $f(x) = 1.5N^2 + 4N + 3$
  - Ex 2. $f(x) = 2N^4 + 10N^3 + 4N^2 + 900log(N) + 3000$
  - Ex 3. $f(x) = 2N - 100$ for $(N > 10)$
- The Big-O notiation will be the fastest increasing term for large inputs.

For example: if $f(N) = 2N^4 + 10N^3 + 4N^2 + 900log(N) + 3$ then the time complexity is $O(N^4)$.

## 1.2 Why are we ignoring low order terms?

Because they become negligible as $N$ grows.

For example: if $f(N) = 4N^4 + 100N^3 + 9000N^2 + N$.

| | $4N^4$ | $100N^3$ | $9000N^2$ | $N$ |
|---|---|---|---|---|
| $N = 100$ | $4 \times 10^8$ | $10^8$ | $9 \times 10^7$ | $100$ |
| $N = 10^8$ | $4 \times 10^{32}$ | $10^{26}$ | $9 \times 10^{19}$ | $10^8$ |
| $N = 10^{12}$ | $4 \times 10^{48}$ | $10^{38}$ | $9 \times 10^{27}$ | $10^{12}$ |

## 1.3 Why are we ignoring multiplicative constants?

Because if we buy a computer that runs twice as fast or a computer with 4 cores, then the running time decreases accordingly.

But it will not change the order of magnitude.

In practice, constants do matter!

In theory, we ignore them.

## 1.4 Formal Definition

**Definition 1.** *Let $f(N)$ and $g(N)$ be two functions on positive integers.*

*We say that $f = O(g)$ if there exists a large constant $C$ (ex. $C = 1000$) such that $f(N) < C * g(N)$ for all sufficiently large $N$.*

*Equivalently, $f = O(g)$ if there is a $C > 0$ (ex. $C = 1000$) such that $f(N)/g(N) <$ for all $N$ large enough.*

$$f = O(g) \quad \text{if} \quad \lim_{\sup N \to \infty} \frac{f(N)}{g(N)} < \infty \tag{1}$$

## 1.5 Examples

Example 1. $f(N) = 5N^2 + 4N + 3$. We want to show $f = O(N^2)$.

*Proof.* Let $C = 12$. Then

$$\begin{aligned} F(N) &= 5N^2 + 4N + 3 < 5N^2 + 4N \\ &< 5N^2 + 4N^2 + 3N^2 \quad [\text{for } N > 2] \\ &= 12N^2 \\ &= CN^2 \end{aligned}$$

Therefore $f = O(N^2)$.

□

Example 2. $f(N) = 5N^2 + 4N + 3$. We want to show $f = O(N^3)$.

*Proof.* Let $C = 12$. Then

$$\begin{aligned} f(N) &= 5N^2 + 4N + 3 \\ &< 5N^3 + 4N^3 + 3N^3 \quad [\text{for } N > 2] \\ &= 12N^3 \\ &< CN^3 \end{aligned}$$

Therefore $f = O(N^3)$.

$\square$

## 1.6 Common orders of magnitude

- $N^2 = O(N^a)$ for all $a \geq 2$
- $log(N)$ is smaller than any power of $N$ for all large enough $N$
  - That is, $log(N) = O(N^{0.1})$ or $log^{10}(N) = O(N)$.