# CSE331 Introduction to Algorithms
# Lecture 2: Asymptotic Notations I

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

July 23, 2021

# Introduction

- References for this lecture and the next one:
  - Lecture notes will be posted on blackboard.
  - Chapter I-3: *Growth of Functions* of the textbook presents it differently.
- We will consider real-valued functions of integer variables.
  - Can also be seen as *sequences*.
- Usually values will be positive as we are interested in running times.
- We will study their *asymptotic* behavior, i.e. at infinity.
- In particular, we will introduce the *asymptotic notations*

$$o(\cdot),\ O(\cdot),\ \Theta(\cdot),\ \Omega(\cdot)$$

# Little-o Notation

## Definition (Little-o notation)

We write $f(n) = o(g(n))$ if and only if there exists a function $\rho(n)$ such that $f(n) = \rho(n)g(n)$ for all $n$, and $\lim_{n\to\infty} \rho(n) = 0$.

- Simpler definition if $g(n) \neq 0$ for all $n$:

$$f(n) = o(g(n)) \text{ iff } \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

- We do not even need $g(n) \neq 0$ for all $n$. It suffices, for instance, that $g(n) \neq 0$ for all $n > 100$.

## Example

$n = o(n^2)$, because $n/n^2 = 1/n \to 0$.

# Motivation

- Worst case running times:

$$\text{Insertion sort: } T_1(n) = a_1 n^2 + b_1 n + c_1$$
$$\text{Merge sort: } T_3(n) = a_3 n \log n + b_3 n$$

  where $a_1 > 0$, $a_3 > 0$, $b_1, b_3, c_1$ are unknown constants.
- For instance, we could have $T_1(n) = 4n^2 - 3n + 6$ and $T_3(n) = 8n \log n + 7n$.
- In any case, regardless of the values of these constants, $T_3(n)/T_1(n) \to 0$ and thus $T_3(n) = o(T_1(n))$.
- It shows that Merge sort is much faster than Insertion sort for large values of $n$, i.e. for sorting a large array.
- So the $o(\cdot)$ notation allows us to compare these functions without even knowing the constants.

# Examples

$$
\begin{aligned}
1/n + 1/n^2 &= o(1) \\
2n + 5 &= o(n^2) \\
10n \log n + 7n + 5 &= o(n^2) \\
n^2 + 2n + 1 &= o(n^3) \\
n^{10} + 5n^3 &= o(2^n) \\
2^n + 3^n + 4^n &= o(n!)
\end{aligned}
$$

- What does $f(n) = o(1)$ mean?
- It means $\lim_{n\to\infty} f(n) = 0$.

# Reformulation of the Definition

## Definition

We say that $f(n) = o(g(n))$ if, for every real number $\varepsilon > 0$, there exists an integer $n_0$ such that $n \geqslant n_0$ implies $|f(n)| \leqslant \varepsilon |g(n)|$.

- Using quantifiers:

$$\forall \varepsilon > 0 \;\; \exists n_0 : \;\; n \geqslant n_0 \Rightarrow |f(n)| \leqslant \varepsilon |g(n)|$$

## Example

Prove that $2n + 5 = o(n^2)$.

# Properties

## Proposition

*For all real numbers $\alpha$, $\beta$,*

1. $\log n = o(n^\alpha)$ *whenever* $\alpha > 0$.
2. $n^\alpha = o(n^\beta)$ *whenever* $\alpha < \beta$.
3. $n^\beta = o(\gamma^n)$ *whenever* $\gamma > 1$.
4. $\gamma^n = o(n!)$

The relation $f(n) = o(g(n))$ is sometimes denoted $f(n) \prec g(n)$. (*Hardy* notation.) So the proposition above can be rewritten:

$$\log n \prec n^\alpha \prec n^\beta \prec \gamma^n \prec n! \qquad \text{whenever } 0 < \alpha < \beta \text{ and } \gamma > 1$$

# Properties

## Proposition

*For every functions $f$, $g$ and $h$, and for every constant $\lambda > 0$,*

1. $f(n) = o(h(n))$ *implies* $\lambda f(n) = o(h(n))$.
2. $f(n) = o(h(n))$ *and* $g(n) = o(h(n))$ *implies* $f(n) + g(n) = o(h(n))$.
3. $f(n) = o(h(n))$ *implies* $f(n)g(n) = o(g(n)h(n))$.
4. $f(n) = o(g(n))$ *and* $g(n) = o(h(n))$ *implies* $f(n) = o(h(n))$.

## Example

Prove that $5n^2 + 2n + 4 = o(n^3)$ using these properties.

# $O$-Notation

- Suppose Insertion sort and Bubble sort have running times

$$T_1(n) = 4n^2 - 3n + 6 \text{ and } T_2(n) = 3n^2 + 6n - 2.$$

- Do we have $T_1(n) = o(T_2(n))$ or $T_2(n) = o(T_1(n))$?
  - No, both statements are wrong, because $T_1(n)/T_2(n) \to 4/3$.
- The big-O notation allows us to compare these two functions:

**Definition**

We write $f(n) = O(g(n))$ if there exist two constants $c > 0$ and $n_0 \in \mathbb{N}$ such that $n \geqslant n_0$ implies $|f(n)| \leqslant c|g(n)|$.

- Difference with $o(\cdot)$:
  "There *exists* a constant $c$" instead of "For all $\varepsilon > 0$"

# $O$-Notation

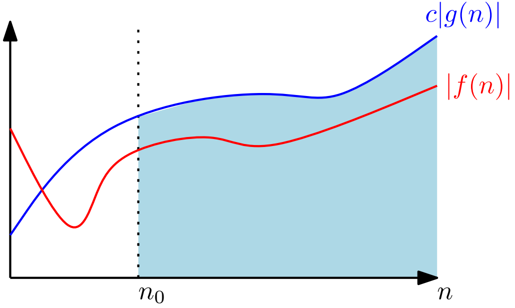


Figure: $f(n) = O(g(n))$

- The relation $f(n) = O(g(n))$ gives an *asymptotic upper bound* on $f(n)$.
- Intuition: $f(n)$ is at most a constant factor times $g(n)$ for large enough $n$.

# Examples

- Let $T_1(n) = 4n^2 - 3n + 6$ and $T_2(n) = 3n^2 + 6n - 2$.
- Prove that $T_1(n) = O(T_2(n))$.
- Remark: $T_2(n) = O(T_1(n))$ is also true.
- $f(n) = O(1)$ means that $f(n)$ is *bounded*: there exists a constant $C > 0$ such that $|f(n)| \leqslant C$ for all $n \in \mathbb{N}$. In computer science, we rather say that $f(n)$ is *constant* when $f(n) = O(1)$.

# Properties

## Proposition

*For all functions $f$, $g$, $h$, $\varphi$ and for all constant $\lambda > 0$,*

1. (i) $f(n) = O(f(n))$
2. (ii) $f(n) = o(g(n))$ *implies* $f(n) = O(g(n))$.
3. (iii) $f(n) = O(h(n))$ *implies* $\lambda f(n) = O(h(n))$.
4. (iv) $f(n) = O(h(n))$ *and* $g(n) = O(h(n))$ *implies* $f(n) + g(n) = O(h(n))$.
5. (v) $f(n) = O(h(n))$ *and* $g(n) = O(\varphi(n))$ *implies* $f(n)g(n) = O(h(n)\varphi(n))$.
6. (vi) $f(n) = o(g(n))$ *and* $g(n) = O(h(n))$ *implies* $f(n) = o(h(n))$.
7. (vii) $f(n) = O(g(n))$ *and* $g(n) = o(h(n))$ *implies* $f(n) = o(h(n))$.
8. (viii) $f(n) = O(g(n))$ *and* $g(n) = O(h(n))$ *implies* $f(n) = O(h(n))$.

# Interpretation

- Hardy's notation: We write

$$f(n) \prec g(n) \text{ iff } f(n) = o(g(n))$$
$$f(n) \preccurlyeq g(n) \text{ iff } f(n) = O(g(n))$$

- Then (i) means: $f(n) \preccurlyeq f(n)$.
- (viii) means: $f(n) \preccurlyeq g(n)$ and $g(n) \preccurlyeq h(n)$ implies $f(n) \preccurlyeq h(n)$.
- (vii) means: $f(n) \preccurlyeq g(n)$ and $g(n) \prec h(n)$ implies $f(n) \prec h(n)$.
- So $o(\cdot)$ is similar with $<$, and $O(\cdot)$ is similar with $\leqslant$.

# Polynomials

## Example

Prove that $f(n) = 5n^3 - 2n^2 + 5$ satisfies $f(n) = O(n^3)$ using the properties above.

- More generally:

## Proposition

*We say that $f(n)$ is a degree-$d$ polynomial in $n$ if there are constants $a_0, \ldots, a_d$ such that $a_d \neq 0$, and $f(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0$ for all $n$. If $f(n)$ satisfies these conditions, then $f(n) = O(n^d)$.*

- Will be useful in our lecture on solving recurrences:

## Proposition

*If $f(n) = O(n^d)$, then there exists a constant $c > 0$ such that $|f(n)| \leqslant cn^d$ for all $n \geqslant 1$.*