

CSC 301: Analysis of Algorithms

Lecture 4: Asymptotic Analysis

William Rebelsky
[rebelskyw]

Grinnell College

Monday 29th January, 2024

Reminders

- Quiz 0 is due tonight
- Homework 0 is due Wednesday
- Please read the Good Proof Writing document before Wednesday
- Tokens: Let me know if you have any events you would like me to add
- Office Hours: 1604 Wednesday and Friday 1-2:50

Table of Contents

1 Notation

- Basic Notation
- Formal Definitions
- Limit Definitions

2 Comparing Functions

3 Practice

Notation

- O -notation: Upper Bound on the asymptotic behavior of an algorithm
- Ω -notation: Lower Bound on the asymptotic behavior of a function
- Θ -notation: Tight Bound on the asymptotic behavior of a function
- Example: $56n^3 + 2n^2 + n + 1000$
 - Highest order term: n^3
 - $O(n^3)$, but it is also true that it is $O(n^4)$, $O(n^5)$, etc
 - $\Omega(n^3)$, but it is also true that it is $\Omega(n^2)$, $\Omega(n)$, etc
 - $\Theta(n^3)$

Example

- $f(n) = 12n^5 + 13n^4$

Example (basic definitions)

- $f(n) = 12n^5 + 13n^4$
 - $O(n^i), i \geq 5$
 - $\Omega(n^j), j \leq 5$
 - $\Theta(n^5)$

O -notation, formally

- Asymptotic upper bound (within a constant factor)
- $O(g(n)) = \{f : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}$
- a function $f(n)$ belongs to the set $O(g(n))$ if there exists a constant c such that $f(n) \leq cg(n)$ for all n
- O -notation requires that $f(n)$ be asymptotically nonnegative

Ω -notation, formally

- Asymptotic lower bound (within a constant factor)
- $\Omega(g(n)) = \{f : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}$
- a function $f(n)$ belongs to the set $\Omega(g(n))$ if there exists a constant c such that $f(n) \leq cg(n)$ for all n
- Ω -notation requires that $f(n)$ be asymptotically nonnegative

Θ -notation, formally

- Asymptotic tight bound (within a constant factor)
- $\Theta(g(n)) = \{f : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ for all } n \geq n_0\}$
- Θ -notation requires that $f(n)$ be asymptotically nonnegative
- O and Θ notation are sometimes conflated

Other Notation

- o -notation
- ω -notation
- We will not work with these in this class
- Page 74 in the book if you are curious

Example

- $f(n) = 12n^5 + 13n^4$

Examples (formal definitions)

- $f(n) = 12n^5 + 13n$
 - $O(n^5)$: $c = 20$, $n_0 = 10$
 - $\Omega(n^1)$, $c = 1$, $n_0 = 10$
 - $\Theta(n^5)$, $c_1 = 1$, $c_2 = 20$, $n_0 = 10$

Limit Definitions

$\lim_{n \rightarrow \infty} \frac{f}{g}$	note
$\{0, \mathbb{R}^+\}$	f is $O(g)$
$\{\mathbb{R}^+, \infty\}$	f is $\Omega(g)$
$\{\mathbb{R}^+\}$	f is $\Theta(g)$

Table: Limit Table

Proof will be left to the reader

Example

- $f(n) = 12n^5 + 13n^4$

Examples (Limit definitions)

- $f(n) = 12n^5 + 13n^4$
 - $g(n) = n^5$
 - $\lim_{n \rightarrow \infty} \frac{f}{g}$
 - $\lim_{n \rightarrow \infty} \frac{12n^5 + 13n^4}{n^5}$
 - 12
 - so $f(n)$ is $\Theta(g(n))$

Table of Contents

1 Notation

- Basic Notation
- Formal Definitions
- Limit Definitions

2 Comparing Functions

3 Practice

Comparing Functions

A function $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$ if and only if $f(n)$ is $\Theta(g(n))$

Transitivity

Statement 1	and Statement 2	Implies Statement 3
$f(n) = \Theta(g(n))$	$g(n) = \Theta(h(n))$	$f(n) = \Theta(h(n))$
$f(n) = O(g(n))$	$g(n) = O(h(n))$	$f(n) = O(h(n))$
$f(n) = \Omega(g(n))$	$g(n) = \Omega(h(n))$	$f(n) = \Omega(h(n))$

Table: Transitivity of asymptotic comparisons

Reflexivity

- $f(n) = O(f(n))$
- $f(n) = \Omega(f(n))$
- $f(n) = \Theta(f(n))$

Transpose Symmetry

- $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$

Examples (comparison)

For each pair f, g state whether f is $O(g)$, $\Theta(g)$, $\Omega(g)$, or none of the above

- $f(n) = n, g(n) = n^{1+\sin(n)}$
- $f(n) = \log(n), g(n) = n$

Table of Contents

1 Notation

- Basic Notation
- Formal Definitions
- Limit Definitions

2 Comparing Functions

3 Practice

Perfect Arrays

Given an array of size N and you have to tell whether the array is perfect or not. An array is said to be perfect if its reverse array matches the original array. If the array is perfect then return True else return False.

- Give a solution
- Prove correctness (loop invariant)
- Argue runtime