Digital Logic Design: a rigorous approach ©

Chapter 7: Asymptotics

Guy Even   Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

April 19, 2020

Book Homepage:
http://www.eng.tau.ac.il/~guy/Even-Medina

The functions we study

We study functions that describe the number of gates in a circuit, the delay of a circuit (length of longest path), the running time of an algorithm, number of bits in a data structure, etc. In all these cases it is natural to assume that

$$\forall n \in \mathbb{N}: \quad f(n) \geq 1.$$

Assumption

The functions we study are functions $f : \mathbb{N} \to \mathbb{R}^{\geq 1}$.

Order of Growth Rates

- We want to compare functions asymptotically (how fast does $f(n)$ grow as $n \to \infty$).
- Ignore constants (not because they are not important, but because we want to focus on "high order" terms).

big-O, big-Omega, big-Theta

Definition

Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq 1}$ denote two functions.

1. We say that $f(n) = O(g(n))$, if there exist constants $c \in \mathbb{R}^{+}$ and $N \in \mathbb{N}$, such that,
$$\forall n > N : f(n) \leq c \cdot g(n) \, .$$
2. We say that $f(n) = \Omega(g(n))$, if there exist constants $c \in \mathbb{R}^{+}$ and $N \in \mathbb{N}$, such that,
$$\forall n > N : f(n) \geq c \cdot g(n) \, .$$
3. We say that $f(n) = \Theta(g(n))$, if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

What does "$=$" actually mean here?!

Warning on Notation

What does the equality sign in $f = O(g)$ mean?

- $O(g)$ in fact refers to a set of functions:

$$O(g) \triangleq \{h : \mathbb{N} \to \mathbb{R}^{\geq 1} \mid \exists c \exists N \forall n > N : h(n) \leq c \cdot g(n)\}$$

- Would have been much better to write $f \in O(g)$ instead of $f = O(g)$.
- But we want to abuse notation and write expressions like:

$$\begin{aligned}(2n^2 + 3n) \cdot 5 \log(n^2) &= O(n^2 \cdot \log n^2) \\ &= O(n^2 \cdot \log n) \; .\end{aligned}$$

Justification: transitivity.

big-O, big-Omega, big-Theta

Definition

Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq 1}$ denote two functions.

1. We say that $f(n) = O(g(n))$, if there exist constants $c \in \mathbb{R}^+$ and $N \in \mathbb{N}$, such that,
$$\forall n > N : f(n) \leq c \cdot g(n) .$$
2. We say that $f(n) = \Omega(g(n))$, if there exist constants $c \in \mathbb{R}^+$ and $N \in \mathbb{N}$, such that,
$$\forall n > N : f(n) \geq c \cdot g(n) .$$
3. We say that $f(n) = \Theta(g(n))$, if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

If $f(n) = O(g(n))$, then "asymptotically, $f(n)$ does not grow faster than $g(n)$".
If $f(n) = \Omega(g(n))$, then "asymptotically, $f(n)$ grows as least as fast as $g(n)$".
Finally, if $f(n) = \Theta(g(n))$, then "asymptotically, $f(n)$ grows as fast as $g(n)$".

Examples

$$n = O(n^2)$$

Examples

$$\log(n) = O(n)$$

Examples

$$10n = O(n), 10^2 n = O(n), \ldots, 10^{100} n = O(n)$$

Examples

$$n \cdot \log \log \log n \neq O(n)$$

Remark

When proving that $f(n) = O(g(n))$, it is not necessary to find the "smallest" constant c.

Example

Suppose you want to prove that $n + \sqrt{n} = O(n^{1.1})$. Then, it suffices to prove that for $n > 2^{100}$:

$$n + \sqrt{n} \leq 10^6 \cdot n^{1.1} \ .$$

Any other constants you can prove the statement for are just as good!

Constant Function

Claim

$f(n) = O(1)$ *iff there exists a constant* c *such that* $f(n) \leq c$, *for every* n.

Sum of Functions

Claim

If $f_i(n) = O(g(n))$ *for* $i \in \{1, \ldots, k\}$ *(*k *is a constant), then* $f_1(n) + \cdots + f_k(n) = O(g(n))$*.*

Asymptotics of Arithmetic Series

Claim

If $\{a_n\}_n$ is an arithmetic sequence with $a_0 \geq 0$ and $d > 0$, then $\sum_{i \leq n} a_i = \Theta(n \cdot a_n)$.

Consequence:

$$\sum_{i=1}^{n} i = \Theta(n^2) \ .$$

Asymptotics of Geometric Series

Claim

If $\{b_n\}_n$ is a geometric sequence with $b_0 \geq 1$ and $q > 1$, then $\sum_{i \leq n} b_i = \Theta(b_n)$.

Consequence: If $q > 1$, then $\sum_{i=1}^{n} q^i = \Theta(q^n)$.

Asymptotic Algebra (big-O)

Abbreviate: $f_i = O(h)$ means $f_i(n) = O(h(n))$.

Claim

Suppose that $f_i = O(g_i)$ for $i \in \{1, \ldots, k\}$, then:

$$\max\{f_i\}_i = O(\max\{g_i\}_i)$$
$$\sum_i f_i = O(\sum g_i)$$
$$\prod_i f_i = O(\prod_i g_i)\ .$$

Consequences:

$2n = O(n)$	mult. by constant
$50n^2 + 2n + 1 = O(n^2)$	polynomial with positive leading coefficient

Asymptotic Algebra (big-Omega)

Claim

Suppose that $f_i = \Omega(g_i)$ for $i \in \{1, \ldots, k\}$, then:

$$\min\{f_i\}_i = \Omega(\min\{g_i\}_i)$$
$$\sum_i f_i = \Omega(\sum g_i)$$
$$\prod_i f_i = \Omega(\prod_i g_i)\ .$$

Consequences:

$$2n = \Omega(n) \quad \text{mult. by constant}$$
$$10^{-6} \cdot n^2 + 2n + 1 = \Omega(n^2) \quad \text{polynomial with positive leading coefficient}$$

Asymptotics as an Equivalence Relation

Claim

$$f = O(f) \qquad \textit{reflexivity}$$

$$f = O(g) \;\not\Longrightarrow\; g = O(f) \qquad \textit{no symmetry}$$

$$(f = O(g)) \;\wedge\; (g = O(h)) \;\Longrightarrow\; f = O(h) \qquad \textit{transitivity}$$

What about Ω?

Big-Omega vs. Big-O

Claim

Assume $f(n), g(n) \geq 1$, *for every* n. *Then,*

$$f(n) = O(g(n)) \quad \Leftrightarrow \quad g(n) = \Omega(f(n)).$$

Recurrence Equations

In this section we deal with the problem of solving or bounding the rate of growth of functions $f : \mathbb{N}^+ \to \mathbb{R}$ that are defined recursively. We consider the typical cases that we will encounter later.

Recurrence 1

Consider the recurrence

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ n + f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases}$$

- Why is $f(n)$ interesting?
- What is the rate of growth of $f(n)$?

Recurrence 1 - motivation

$$\hat{f}(n) \triangleq \begin{cases} 0 & \text{if } n = 1 \\ \frac{n}{2} + \hat{f}(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases} \quad [f(n) = 1 + 2\hat{f}(n) = \Theta(\hat{f}(n))]$$

Algorithm 1 $MAX(x_1, \ldots, x_n)$ - assume n is a power of 2

1. Base Case: If $n = 1$ then return x_1.
2. Reduction Rule:
   1. For $i = 1$ to $\frac{n}{2}$ Do: $y_i \leftarrow \max\{x_{2i-1}, x_{2i}\}$
   2. Return $MAX(y_1, \ldots, y_{n/2})$

Claim

Number of comparisons in $MAX(x_1, \ldots, x_n)$ equals $\hat{f}(n)$.

Recurrence 1 - analysis

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ n + f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases}$$

Lemma

The rate of growth of the function $f(n)$ is $\Theta(n)$.

- start by proving for powers of 2.
- what if n is not a power of 2?
- what about $f(n) = n + f(\lceil \frac{n}{2} \rceil)$?

Is it enough to solve for powers of 2?

Lemma

Assume that:

1. *The functions $f(n)$ and $g(n)$ are both monotonically nondecreasing.*
2. *The constant ρ satisfies, for every $k \in \mathbb{N}$,*

$$\frac{g(2^{k+1})}{g(2^k)} \leq \rho \, .$$

Then,

1. *If $f(2^k) = O(g(2^k))$, then $f(n) = O(g(n))$.*
2. *If $f(2^k) = \Omega(g(2^k))$, then $f(n) = \Omega(g(n))$.*

Recurrence 2.

Consider the recurrence

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ n + 2 \cdot f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases}$$

- Why is $f(n)$ interesting?
- What is the rate of growth of $f(n)$?

Recurrence 2 - motivation

$$\hat{f}(n) \triangleq \begin{cases} 0 & \text{if } n = 1 \\ (n-1) + 2\hat{f}(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases} \quad [f(n) = \hat{f}(n) + 2n - 1]$$

Algorithm 2 $SORT(x_1, \ldots, x_n)$ - assume n is a power of 2

1. Base Case: If $n = 1$ then return x_1.
2. Reduction Rule:
    1. $(y_1, \ldots, y_{n/2}) \leftarrow SORT(x_1, \ldots, x_{n/2})$
    2. $(y_{n/2+1}, \ldots, y_n) \leftarrow SORT(x_{n/2+1}, \ldots, x_n)$
    3. Return $MERGE((y_1, \ldots, y_{n/2}), (y_{n/2+1}, \ldots, y_n))$

Claim

Number of comparisons in $SORT(x_1, \ldots, x_n)$ equals $\hat{f}(n)$.

Recurrence 2 - analysis

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ n + 2 \cdot f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases}$$

Lemma

The rate of growth of the function $f(n)$ is $\Theta(n \log n)$.

- prove for powers of 2
- extend to arbitrary n

Recurrence 3.

Consider the recurrence

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ n + 3 \cdot f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases}$$

Lemma

The rate of growth of the function $f(n)$ is $\Theta(n^{\log_2 3})$.

hint: $f(2^k) = 3^{k+1} - 2^{k+1}$.

Example - 1

Consider the recurrence

$$f(n) \triangleq \begin{cases} c & \text{if } n = 1 \\ a \cdot n + b + f\left(\left\lfloor \frac{n}{2} \right\rfloor\right) & \text{if } n > 1, \end{cases}$$

where a, b, c are non-negative constants and $a \neq 0$.

Lemma

The rate of growth of the function $f(n)$ is $\Theta(n)$.

proof: $f(2^k) = 2a \cdot 2^k + b \cdot k + c - 2a$...

Example -2

Consider the recurrence

$$f(n) \triangleq \begin{cases} c & \text{if } n = 1 \\ a \cdot n + b + 2 \cdot f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1, \end{cases}$$

where constants $a, b, c \geq 0$ and $a \neq 0$.

Lemma

The rate of growth of the function $f(n)$ is $\Theta(n \log n)$.

proof: We claim that $f(2^k) = a \cdot k2^k + (b + c) \cdot 2^k - b$....

Example - 3

Consider the recurrence

$$F(k) \triangleq \begin{cases} 1 & \text{if } k = 0 \\ 2^k + 2 \cdot F(k-1) & \text{if } k > 0, \end{cases}$$

Lemma

$F(k) = (k+1) \cdot 2^k$.

Proof: Define $f(n) \triangleq F(\lceil \log_2 n \rceil)$. Observe that $f(2^x) \triangleq F(x)$....

Examples with floor and ceiling

1.

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ 1 + f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1, \end{cases}$$

2.

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ n + f(\lfloor \frac{n}{2} \rfloor) + f(\lceil \frac{n}{2} \rceil) & \text{if } n > 1, \end{cases}$$

If $n = 2^k$, then floor and ceiling functions can be ignored!