

Algorithms and Data Structures DV2

Arne Andersson



Today's lectures

Textbook chapters 1-3

- Introduction
- Overview of Algorithmic Mathematics
- The Logarithm Function
- Growth of Functions
- Exercises

Informationsteknologi

Institutionen för informationsteknologi | www.it.uu.se



Today's lectures

Textbook chapters 1-3

- Introduction
- Overview of Algorithmic Mathematics
- The Logarithm Function
- Growth of Functions
- Exercises

Informationsteknologi

Institutionen för informationsteknologi | www.it.uu.se



What is Computer Science?

- Programming
- Algorithms & Data Structures
- The Rest

Informationsteknologi

Institutionen för informationsteknologi | www.it.uu.se



What is Algorithms and Data Structures?

■ Algorithm:
Problem → Algorithm → Solution

■ Data Structure:
Query → Algorithm → Reply
↓ ↑
Data Structure

Informationsteknologi

Institutionen för informationsteknologi | www.it.uu.se



Where do we use Algorithms & Data Structures?

- Everywhere!
- Hash tables, Tree structures, sorting algorithms, priority queues, graph algorithms, etc are common components in almost all standard software.
- Examples of areas where algorithms & data structures play a central role: Data Mining, Bioinformatics, Search Engines, Data Bases...

Informationsteknologi

Institutionen för informationsteknologi | www.it.uu.se

But, don't we have standard libraries, and no need for learning about how they work?

- Yes, there are some standard libraries, but....

- Bad knowledge about the underlying principles will cause poor usage of algorithms & data structures

- Often one has to do good choices between standard solutions

- Sometimes, one will have to develop extensions and variations and, most fun, entirely own solutions

...and, as extra bonuses

- Training in algorithmic thinking makes you a better and smarter professional, with implications far outside Computer Science
- and it's fun!

An Example of Algorithmic Thinking

- Can we maintain an index (data structure) that lets us search for any text string on the entire Internet?
- How much would it cost?
- How long would the query time be?

Today's lectures

Textbook chapters 1-3

- Introduction
- Overview of Algorithmic Mathematics
- The Logarithm Function
- Growth of Functions
- Exercises

Algorithmic Mathematics

- Used to analyze running time of algorithms
- With running time, we can mean
 - Running time on some specific or all input
 - Worst Case time
 - Best Case time
 - Average time or Expected time
 - (Expected Worst Case time)
 - More to come...

Algorithmic Mathematics

- Some major components
 - The logarithm function
 - Asymptotics
 - Recurrence equations (induction)

Algorithmic Mathematics

- Some major components
 - The logarithm function

Binary search among n keys requires $\lceil \log(n+1) \rceil$ comparisons

- Asymptotics

Binary search takes $\Theta(\log n)$ time

- Recurrence equations (induction)

$$\begin{cases} T(n) = T(n/2) + C_1 \\ T(1) = C_2 \end{cases} \\ \Rightarrow T(n) = \Theta(\log n)$$

Algorithmic Mathematics

- Som major components
 - The logarithm function
 - Asymptotics
 - Recurrence equations (induction)

Binary search among n keys requires $\lceil \log(n+1) \rceil$ comparisons

Binary search takes $\Theta(\log n)$ time

$$\begin{cases} T(n) = T(n/2) + C_1 \\ T(1) = C_2 \end{cases} \\ \Rightarrow T(n) = \Theta(\log n)$$

Solution :

$$\begin{aligned} T(n) &= T(n/2) + C_1 \\ &= T(n/4) + 2C_1 \\ &= T(n/8) + 3C_1 \\ &\dots \text{etc...} \\ &= T(1) + \log n \cdot C_1 \\ &= \Theta(\log n) \end{aligned}$$

Today's lectures

Textbook chapters 1-3

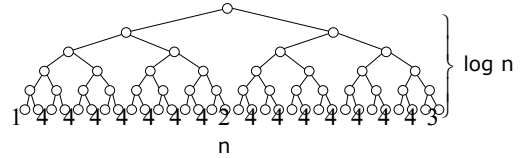
- Introduction
- Overview of Algorithmic Mathematics
- The Logarithm Function
- Growth of Functions
- Exercises

The Logarithm function

- In Computer Science, $\log(n)$ or $\lg(n)$ means the logarithm in base 2
- (In Mathematics, we often use \ln , the natural logarithm, and \lg as the logarithm in base 10)
- So, here,

$$\log n = \lg n = \log_2 n$$

What Is the logarithm function, really?



What Is the logarithm function, really?

- $\log n$ is the number of bits we need to represent the numbers $0..n$
- $\log n$ is height of a perfectly balanced binary tree with n nodes
- $\log n$ is the number of times we can repeat halving, starting with n , until we only have one left

Some other functions, based on log

- $\log n$ is the number of times we can repeat **halving**, starting with n , until we only have one left
- $\log \log n$ is the number of times we can repeat **taking the square root**, starting with n , until we only have two left
- $\log^* n$ is the number of times we can repeat **taking the logarithm**, starting with n , until we only have one left

n	$\log n$	$\lceil \log \log n \rceil$
2	1	1
4	2	1
8	3	2
16	4	2
32	5	3
64	6	3
128	7	3
256	8	3
512	9	4
1024	10	4
10^3	≈ 10	4
10^6	≈ 20	5
10^9	≈ 30	5

Today's lectures

Textbook chapters 1-3

- Introduction
- Overview of Algorithmic Mathematics
- The Logarithm Function
- **Growth of Functions**
- Exercises

O (Big Oh or Ordo)

$f(n) = O(g(n))$ if $0 \leq f(n) \leq c \cdot g(n), n \geq n_0$
 $g(n)$ is an asymptotic upper bound on $f(n)$
 Informally: $f(n)$ grows as most as $g(n)$

Examples

$$2 \log n + 1/n = O(\log n)$$

$$n^2 + \log n = O(n^2)$$

$$n^2 + \log n = O(n^3)$$

Ω (Big Omega)

$f(n) = \Omega(g(n))$ if $0 \leq c \cdot g(n) \leq f(n), n \geq n_0$
 $g(n)$ is an asymptotic lower bound on $f(n)$
 Informally: $f(n)$ grows as least as $g(n)$

Examples

$$2 \log n + 1/n = \Omega(\log n)$$

$$n^2 + \log n = \Omega(n^2)$$

$$n^2 + \log n \neq \Omega(n^3) \text{ but } n^2 + \log n = \Omega(n)$$

Θ (Theta)

$f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$
 $g(n)$ is an asymptotically tight bound on $f(n)$
 Informally: $f(n)$ grows in the same way as $g(n)$

Examples

$$2 \log n + 1/n = \Theta(\log n)$$

$$n^2 + \log n = \Theta(n^2)$$

$$n^2 + \log n \neq \Theta(n^3) \text{ and } n^3 + \log n \neq \Theta(n) \text{ but } n^3 + \log n = \Theta(n^3)$$

o (Little Oh) and ω (Little Omega)

$f(n) = o(g(n))$ if $f(n) = O(g(n))$ and $f(n) \neq \Theta(g(n))$
 Informally: $f(n)$ grows slower than $g(n)$

$f(n) = \omega(g(n))$ if $f(n) = \Omega(g(n))$ and $f(n) \neq \Theta(g(n))$
 Informally: $f(n)$ grows faster than $g(n)$

Examples

$$2 \log n + 1/n = o(\log^2 n)$$

$$n^2 + \log n = \omega(n)$$

Summary, $O, \Omega, \Theta, o, \omega$

The following statements are true

$2n \log n + 3n = O(n \log n)$	$2n \log n + 3n$ grows as most as	$n \log n$
$2n \log n + 3n = \Omega(n \log n)$	$2n \log n + 3n$ grows as least as	$n \log n$
$2n \log n + 3n = \Theta(n \log n)$	$2n \log n + 3n$ grows in the same way as	$n \log n$
$2n \log n + 3n = o(n^2 \log n)$	$2n \log n + 3n$ grows slower than	$n^2 \log n$
$2n \log n + 3n = O(n^2 \log n)$	O follows directly from	o
$2n \log n + 3n = \omega(\sqrt{n \log n})$	$2n \log n + 3n$ grows faster than	$\sqrt{n \log n}$
$2n \log n + 3n = \Omega(\sqrt{n \log n})$	Ω follows directly from	ω

Another way to think intuitively about $O, \Omega, \Theta, o, \omega$

$f(n) = O(g(n))$	corresponds to	$f(n) \leq c \cdot g(n)$
$f(n) = \Omega(g(n))$	corresponds to	$f(n) \geq c \cdot g(n)$
$f(n) = \Theta(g(n))$	corresponds to	$f(n) = c \cdot g(n)$
$f(n) = o(g(n))$	corresponds to	$f(n) < c \cdot g(n)$
$f(n) = \omega(g(n))$	corresponds to	$f(n) > c \cdot g(n)$

A simple Rule of Thumb:

We only need to consider the leading term

$$f(n) = n^2 + n \log n + \underbrace{2^n}_{\text{leading term}} + \sqrt{\log \log n}$$

$$f(n) = \Theta(2^n)$$

Som more mathematics

- Monotonicity of functions
- Floors and ceilings
- Modular arithmetic
- Polynomials
- Exponentials, Logarithms
- Factorials
- Functional iteration, f^*
- Fibonacci numbers



$$\lfloor 3.14 \rfloor = 3 \quad \lceil 3.14 \rceil = 4$$

$$x \bmod 2 = 1 \text{ iff } x \text{ is odd}$$

$f(n)$ is polynomially bounded if $f(n) = O(n^i)$ for some constant i

So, $n^{10} \sqrt{\log n}$ is polynomially bounded

$$\log(2^n) = n \quad 2^{\log n} = n$$

$$2^n \cdot 2^{\sqrt{n}} = 2^{n+\sqrt{n}} \quad \log(xy) = \log x + \log y$$

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n \quad n! = o(n^n)$$

$$n! = \omega(2^n) \quad \log(n!) = \Theta(n \log n)$$

$$\text{Log}^* n$$

$$F_0 = 0 \quad F_1 = 1 \quad F_i = F_{i-1} + F_{i-2}$$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Today's lectures

Textbook chapters 1-3

- Introduction
- Overview of Algorithmic Mathematics
- The Logarithm Function
- Growth of Functions
- Exercises**

Exercise 3.1-1

Prove

$$\max(f(n), g(n)) = \Theta(f(n) + g(n))$$

Example

$$\max(n^2, \log n) = \Theta(n^2 + \log n) = \Theta(n^2)$$

Proof sketch : Prove upper and lower bounds separately

Exercise 3.1-2

Prove

$$(n+a)^b = \Theta(n^b)$$

where a is a constant

Example

$$(n+10)^b = \Theta(n^b)$$

Exercise 3.1-3

Explain why the sentence

"The running time of Algorithm A is at least $O(n^2)$ " is meaningless.

Exercise 3.1-4

Is $2^{n+1} = O(2^n)$?

Is $2^{2^n} = O(2^n)$?

Exercise 3.1-5

Prove Theorem 3.1:

For any two functions $f(n)$ and $g(n)$, $f(n) = \Theta(g(n))$ if and only if

$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Proof by definition

Exercise 3.1-6

Prove that the running time of an algorithm is $\Theta(g(n))$ iff its worst - case running time is $O(g(n))$ and its best - case running time is $\Omega(g(n))$

Exercise 3.2-1

Show that if $f(n)$ and $g(n)$ are monotonically increasing, then $f(n) + g(n)$ is monotonically increasing and $f(g(n))$ is monotonically increasing

Further, if $f(n)$ and $g(n)$ are nonnegative, then $f(n) \cdot g(n)$ is monotonically increasing.

Exercise 3.2-2

Prove Equation (3.15)

$$a^{\log_b c} = c^{\log_b a}$$

We use Equation (3.14)

$$\log_b a = \frac{\log_c a}{\log_c b} = \frac{\log a}{\log b}$$