

# Basics -- 2

From: Data Structures and Their Algorithms,  
by Harry R. Lewis and Larry Denenberg  
(Harvard University: Harper Collins  
Publishers)



## Review: Logarithms, Powers and Exponentials

Let  $b$  be any real greater than 1 and let  $x$  be any real greater than 0. The logarithm to the base  $b$  of  $x$ , denoted  $\log_b x$  is defined to be the number  $y$  such that

$$b^y = x$$

$$\log_b 1 = 0$$

$$\log_b x > 0 \text{ if } x > 1$$

$$\log_b b = 1$$

$$\log_b x < 0 \text{ if } 0 < x < 1$$

Any logarithmic function is a monotone increasing function of its argument, that is



$$\log_b x_1 > \log_b x_2 \text{ provided that } x_1 > x_2$$

**Doubling the argument increases the base 2 logarithm by 1.  
That is,**

$$\log_2 2x = (\log_2 x) + 1$$

**Why?**

$$2^{(\log_2 x) + 1} = 2^{\log_2 x} \cdot 2 = 2 \cdot x$$

$$2^{\log_2 2x} = 2x$$



## Review: Logarithms

$$\log_b (x_1 \cdot x_2) = \log_b x_1 + \log_b (x_2)$$

$$\log_b (x_1 / x_2) = \log_b x_1 - \log_b x_2$$

$$\log_b x^c = c \cdot \log_b x$$



Suppose  $a$  and  $b$  are both greater than 1, what is the relation of  $\log_a x$  to  $\log_b x$ ?

Since  $x = a^{\log_a x}$

$$\begin{aligned}\log_b x &= \log_b(a^{\log_a x}) \\ &= \log_a x \cdot \log_b a\end{aligned}$$

This is a constant. So, any two logarithmic functions differ only by a constant factor.

For example, suppose we know that an algorithm executes  $\lg x$  instructions, where  $x$  is the size of the input.

$$\lg x = \lg(10^{\log_{10} x})$$

$$\begin{aligned}&= \log_{10} x \cdot \lg 10 \\ &\sim 3.32 * \log_{10} x\end{aligned}$$



**The number of bits in the usual binary notation for the positive integer  $N$  is  $\lfloor \text{Lg}N \rfloor + 1$ .**

**For example, how many bits are required to represent**

**41      0 1 0 1 0 0 1      6 bits  $\lfloor \text{Lg}41 \rfloor + 1 = 5+1 = 6$**   
          64 32 16 8 4 2 1

**3                      0 1 1      2 bits  $\lfloor \text{Lg}3 \rfloor + 1 = 1+1 = 2$**   
                          4 2 1



**The number of digits in the usual base 10 notation for the positive integer  $N$  is  $\lfloor \text{Log}_{10}N \rfloor + 1$ .**

**For example, how many digits are required to represent 31?**

$$\begin{array}{r} 31 \\ 10 \end{array} \quad \begin{array}{r} 3 \ 1 \\ 10 \ 1 \end{array} \quad 2 \text{ digits } \lfloor \text{Log}_{10}31 \rfloor + 1 = 1+1 = 2$$



**Any function from reals to reals of the form  $g(x) = x^\alpha$  for some constant  $\alpha > 0$  is called a simple power. Any simple power is an increasing function of its argument.**

**Examples:**

**$x^2$ ,  $x^3$  and  $x^{1/3}$  are simple powers**





**An exponential function is one of the form  $h(x) = c^x$  for some constant  $c > 1$ .**

**Examples:**

**$2^x$  and  $100^x$  are exponential functions of  $x$**



## Dominates

- Let  $f$  and  $g$  be functions from reals to reals.  $f$  dominates  $g$  if the ratio  $f(n) / g(n)$  increases without bound as  $n$  increases without bound. In other words, if for any  $c > 0$  there is an  $n_0 > 0$  such that  $f(n) > c \cdot g(n)$  for all  $n > n_0$ .
- Examples:  $f(n) = n^2$  dominates  $g(n) = 2n$  since for any  $c$   $n^2 > c \cdot 2n$  whenever  $n > 2c$ .
- $f(n) = 10n$  does not dominate  $g(n) = 2n$  since the ratio of  $f(n) / g(n)$  is never larger than 5.



## **Theorem:**

**Any exponential function dominates any simple power,  
any simple power dominates any logarithmic function.**



**Let  $N$  be the set of nonnegative integers  $\{0,1,\dots\}$ . Let  $R$  be the set of real numbers and let  $R^*$  be the set of nonnegative real numbers.**

**Let  $g$  be a function from  $N$  to  $R^*$ . Then  $O(g)$  is the set of all functions  $f$  from  $N$  to  $R^*$  such that, for some constants  $c > 0$  and  $N_0 \geq 0$ .**

$$f(N) \leq c \cdot g(N) \text{ for all } N \geq N_0.$$

**In other words,  $f \in O(g)$  if the value of  $f$  is bounded from above by a fixed multiple of the value of  $g$  for all sufficiently large values of the argument.**



## Examples:

For any  $f$  it is the case that  $f \in O(f)$ .  
Any constant multiple of  $f$  is in  $O(f)$ .  
 $F(n) = 13n + 7$  is in  $O(n)$ .

Why?

$$13n + 7 \leq 14n \text{ for } n \geq 7$$

So the definition is satisfied with  $c = 14$ ,  $n_0 = 7$ .

$$1000n \in O(.0001n^2)$$

Why?

Let  $c = 10^7$  and  $n_0 = 0$  in the definition of  $O()$ .

$$f(n) = n$$

$$f(n) \in O(n^2)$$



## Little o

For any function  $g$ ,  $o(g)$  is the set of all functions that are dominated by  $g$ . That is, the set of all functions  $f$  such that for each constant  $c > 0$  there is an  $n_0 > 0$  such that

$$f(n) < c \cdot g(n) \text{ for all } n > n_0.$$

**Examples:**

Let  $f(n) = n$  and  $g(n) = n^2$  then  
 $f(n) \in o(g(n))$

Let  $f(n) = n^2$  and  $g(n) = 2^n$  then  
 $f(n) \in o(g(n))$



## Theorem: Growth Rates

1. The power  $n^\alpha$  is in  $O(n^\beta)$  if and only if  $\alpha \leq \beta$  ( $\alpha, \beta > 0$ ); and  $n^\alpha$  is in  $o(n^\beta)$  if and only if  $\alpha < \beta$ .

Examples:

$$n \in O(n^3)$$

$$n \in o(n^3)$$

Intuitively

$$n \leq n^3$$

$$n < n^3$$



**2.  $\log_b n \in o(n^\alpha)$  for any  $b$  and  $\alpha$ .**

**Examples:**

$$\log_{10} n \in o(n)$$

$$\log_2 n \in o(n^{1/2})$$





**3.  $n^\alpha \in o(c^n)$  for any  $\alpha > 0$  and  $c > 1$ .**

Examples:

$$n^2 \in o(4^n)$$

$$n^{100} \in o(2^n)$$



4.  $\log_a n \in O(\log_b n)$  for any  $a$  and  $b$ .

$$\log_2 n \in O(\log_{10} n)$$

$$\log_{10} n \in O(\log_2 n)$$



5.  $c^n \in O(d^n)$  if and only if  $c \leq d$ , and  $c^n \in o(d^n)$  if and only if  $c < d$ .

Examples:

$$3^n \in O(4^n)$$

$$3^n \in o(4^n)$$



**6. Any constant function  $f(n) = c$  is in  $O(1)$ .**

For example:

A 32-bit add instruction  $O(1)$ .



**Big-O only provides an upper bound.**

**For example:**

**$17n^2 \in O(n^2)$  but**

**$17n^2 \in O(n^{37})$**

**$17n^2 \in O(2^n)$**



## Big Omega (Big- $\Omega$ ):

**Big- $\Omega$  notation is exactly the converse of Big-O notation;  
 $f \in \Omega(g)$  if and only if  $g \in O(f)$ .**

**$f \in O(g)$  implies that  $f$  grows at most as quickly as  $g$ .  
 $f \in \Omega(g)$  implies that  $f$  grows at least as quickly as  $g$ .**

### Examples:

let  $f(n) = n$   
 $f(n) \in O(n^2)$   
 $n^2 \in \Omega(n)$



## Big theta (Big $\theta$ ):

$$\theta(f) = O(f) \cap \Omega(f)$$

**Example:**

**Let  $f(n) = 4n$  then**

$$f(n) \in O(n)$$

$$f(n) \in \Omega(n)$$

**so**

$$f(n) \in \theta(n)$$

**The set of functions  $\theta(f)$  is the order of  $f$ .**



# A Quiz

Suppose we use a phone book to look up a number in the standard way. Let  $T(n)$  be the number of operations (comparisons) required.

In the worst case  $T(n) \in O(\text{Log } N)$ .

*True or False:*

*Also, in the worst case,*

$T(n) \in O(n)$  \_\_\_\_\_

$T(n) \in \theta(n)$  \_\_\_\_\_

$T(n) \in \Omega(n)$  \_\_\_\_\_



# A Quiz

Suppose we use a phone book to look up a number in the standard way. Let  $T(n)$  be the number of operations (comparisons) required.

In the worst case  $T(n) \in O(\text{Log } N)$ .

*True or False:*

*Also, in the worst case,*

$T(n) \in O(n)$     *True*

$T(n) \in \theta(n)$     *False*

$T(n) \in \Omega(n)$     *False*

# A Quiz

Suppose we use a phone book to look up a number. Let  $T(n)$  be the number of operations (comparisons) required.

In the best case  $T(n) \in O(1)$ .

*True or False:*

*Also, in the best case,*

$T(n) \in O(n)$  \_\_\_\_\_

$T(n) \in \theta(n)$  \_\_\_\_\_

$T(n) \in \Omega(n)$  \_\_\_\_\_

# A Quiz

Suppose we use a phone book to look up a number. Let  $T(n)$  be the number of operations (comparisons) required.

In the best case  $T(n) \in O(1)$ .

*True or False:*

*Also, in the best case,*

*$T(n) \in O(n)$  True*

*$T(n) \in \theta(n)$  False*

*$T(n) \in \Omega(n)$  False*

# A Quiz

Suppose we want to delete the last item on a singly linked list. Let  $T(n)$  be the number of operations (comparisons) required.

There are no cases to consider:  $T(n) \in O(n)$ .

*True or False:*

$$T(n) \in O(\lg n)$$

$$T(n) \in \theta(n)$$

$$T(n) \in \Omega(\lg n)$$

# A Quiz

Suppose we want to delete the last item on a singly linked list. Let  $T(n)$  be the number of operations (comparisons) required.

There are no cases to consider:  $T(n) \in O(n)$ .

*True or False:*

*$T(n) \in O(\lg n)$  False*

*$T(n) \in \theta(n)$  True*

*$T(n) \in \Omega(\lg n)$  True*

# Remember

*When working with Big O, Big  $\theta$  and Big  $\Omega$ , be sure to always consider only large  $n$ .*

*In addition, pin the case down first and then consider Big O, Big  $\theta$ , and Big  $\Omega$ .*

*Lastly, remember that some times “case” does not apply.*

# Algorithms and Problems

In this class, we will mostly be analyzing algorithms (counting operations) in terms of Big O, Big  $\theta$  and Big  $\Omega$ .

**Problems** may also be analyzed. The lower bound for a particular problem is the worst case running time of the fastest algorithm that solves that problem.

Later, we will look at an argument that comparison based sorting is  $\Omega(n \text{ Log } n)$ . What does that mean?