CS/MATH 111, Discrete Structures - Fall 2018. Discussion 2

Andres, Sara, Elena
University of California, Riverside
October 4, 2018

Outline

- Proof by Induction
- Logarithm
- Big-O notation
- Big-omega notation
- Big-Theta notation
- Execution time

Proof by Induction

Let's prove the second statement via induction.

Property *Q(n)*:

$$1 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Check *Q(1)*:

$$1 = 1 \times 2 \times 3 \; / \; 6 = 1$$

which is true. Now assume *Q(n)* is true, let's prove *Q(n+1)*:

$$\begin{aligned}
&1 + 2^2 + 3^2 + \ldots + n^2 + (n+1)^2 = \\
&= \frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \\
&= \frac{n(n+1)(2n+1) + 6(n+1)^2}{6} = \frac{(n+1)\left(n(2n+1) + 6(n+1)\right)}{6} = \\
&= \frac{(n+1)(2n^2 + n + 6n + 6)}{6} = \frac{(n+1)(2n^2 + 7n + 6)}{6} = \\
&= \frac{(n+1)(n+2)(2n+3)}{6} = \frac{(n+1)(n+2)(2(n+1)+1)}{6}
\end{aligned}$$

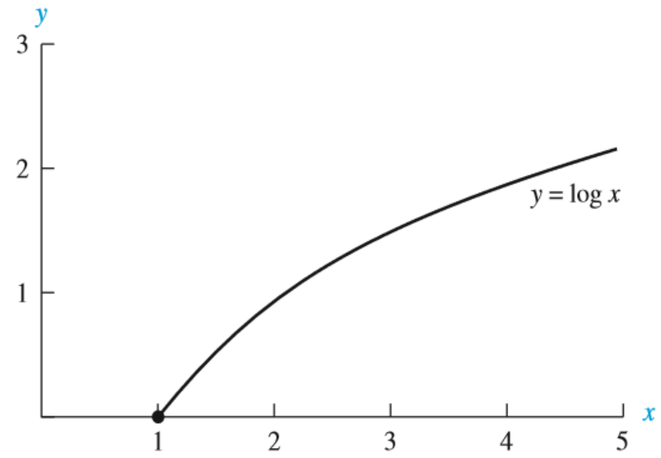
The last expression is exactly property Q(n+1), which finishes the induction proof.

Logarithm

Logarithmic Functions

Suppose that b is a real number with $b > 1$. Then the exponential function b^x is strictly increasing (a fact shown in calculus). It is a one-to-one correspondence from the set of real numbers to the set of nonnegative real numbers. Hence, this function has an inverse $\log_b x$, called the **logarithmic function to the base *b***. In other words, if b is a real number greater than 1 and x is a positive real number, then

$$b^{\log_b x} = x.$$



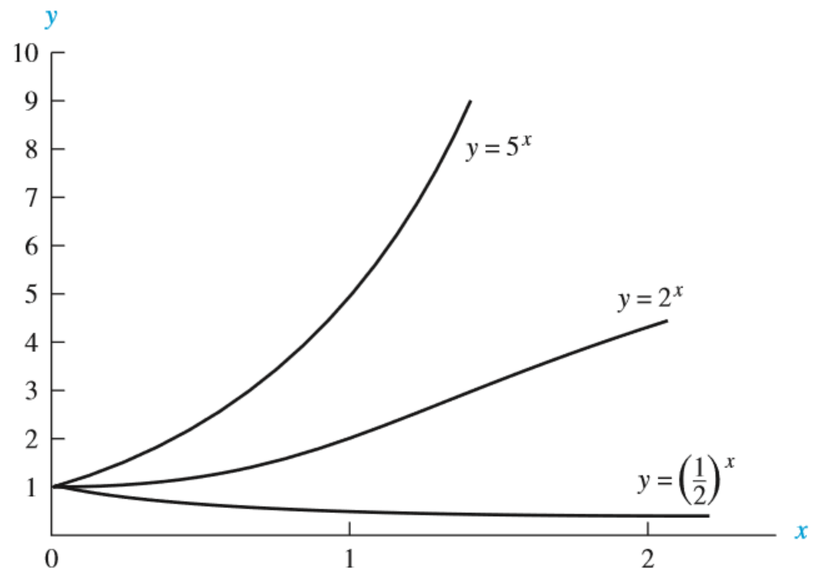


FIGURE **Graphs of the Exponential Functions to the Bases $\frac{1}{2}$, 2, and 5.**

THEOREM

Let b be a real number greater than 1. Then

1. $\log_b(xy) = \log_b x + \log_b y$ whenever x and y are positive real numbers, and
2. $\log_b(x^y) = y \log_b x$ whenever x is a positive real number and y is a real number.

THEOREM Let a and b be real numbers greater than 1, and let x be a positive real number. Then

$$\log_a x = \log_b x / \log_b a.$$

Proof: To prove this result, it suffices to show that

$$b^{\log_a x \cdot \log_b a} = x.$$

Big-o notation

Big-O Notation

The growth of functions is often described using a special notation. Definition 1 describes this notation.

DEFINITION 1 Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $O(g(x))$ if there are constants C and k such that

$$|f(x)| \leq C|g(x)|$$

whenever $x > k$. [This is read as "$f(x)$ is big-oh of $g(x)$."]

Remark: Intuitively, the definition that $f(x)$ is $O(g(x))$ says that $f(x)$ grows slower that some fixed multiple of $g(x)$ as x grows without bound.

Big-Omega and Big-Theta Notations

DEFINITION 3

Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$. When $f(x)$ is $\Theta(g(x))$ we say that f is big-Theta of $g(x)$, that $f(x)$ is of *order* $g(x)$, and that $f(x)$ and $g(x)$ are of the *same order*.

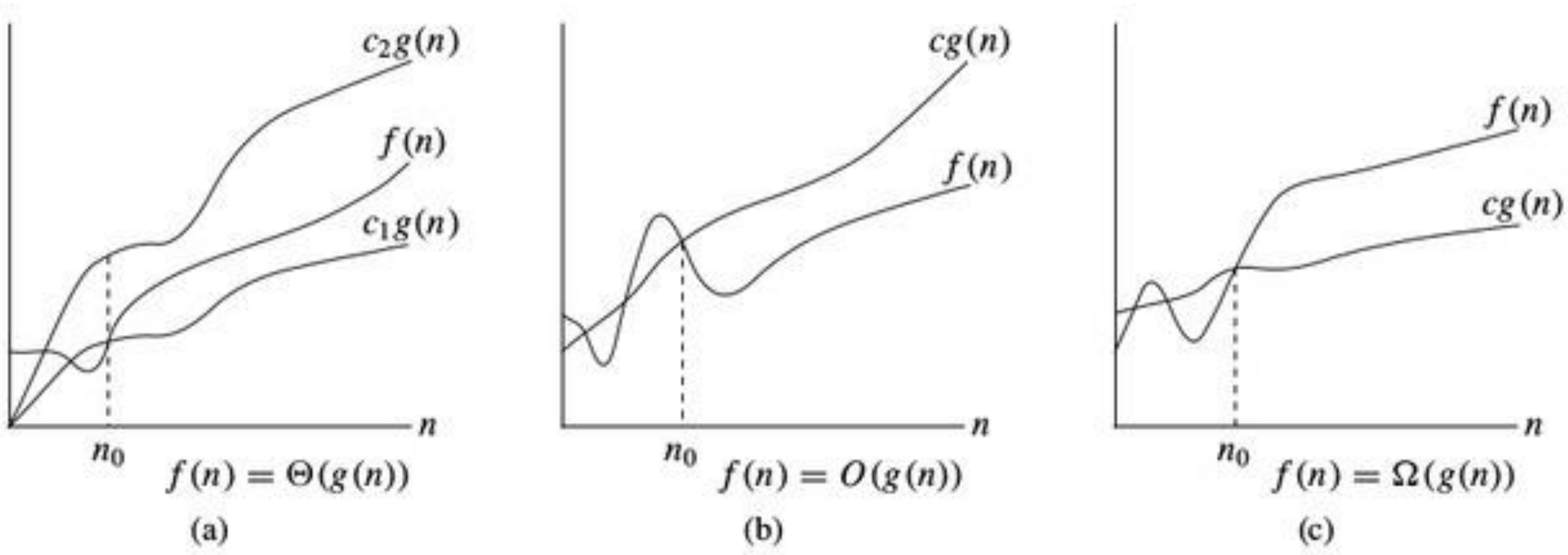
When $f(x)$ is $\Theta(g(x))$, it is also the case that $g(x)$ is $\Theta(f(x))$. Also note that $f(x)$ is $\Theta(g(x))$ if and only if $f(x)$ is $O(g(x))$ and $g(x)$ is $O(f(x))$

$f(x)$ is $\Theta(g(x))$ if and only if there are real numbers C_1 and C_2 and a positive real number k such that

$$C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$$

whenever $x > k$. The existence of the constants C_1, C_2, and k tells us that $f(x)$ is $\Omega(g(x))$ and that $f(x)$ is $O(g(x))$, respectively.

Usually, when big-Theta notation is used, the function $g(x)$ in $\Theta(g(x))$ is a relatively simple reference function, such as x^n, c^x, $\log x$, and so on, while $f(x)$ can be relatively complicated.


$c_2 g(n)$
$f(n)$
$c_1 g(n)$
n
n_0
$f(n) = \Theta(g(n))$
(a)
$cg(n)$
$f(n)$
n
n_0
$f(n) = O(g(n))$
(b)
$f(n)$
$cg(n)$
n
n_0
$f(n) = \Omega(g(n))$
(c)

Execution Time

```
x := 0;
i := n ;
while (i > 1) do begin
        x := x + 1;
        i := i div 2;
end;
```

```
x := 0;
i := n;
while (i > 1) do  begin
        x := x + 1;
         i := i div 2;
end;
```

i	x
32	1
16	2
8	3
4	4
2	5
1	-

($5 = \log_2 32$)   O(log)

```
for i:=1 to n do
    for j:=i to n do
        write('OK');
```

```
for i:=1 to n do
    for j:=i to n do
        write('OK');
```

$$\frac{n(n+1)}{2} \qquad O(n^2)$$