Computer Science 2
Data Structures and Algorithms
V22.0102
Otávio Braga
New York University
Outline

- Review of logarithms
- Big-Oh notation
- Examples
\( \log_b a \)

- The number of times we have to multiply \( b \) by itself to get \( a \)
  \[ b^{\log_b a} = a \]
- The exponent to which we have to raise \( b \) to get \( a \)
- The number of times we have to divide \( a \) by \( b \) to get 1
  \[ \frac{a}{b^{\log_b a}} = 1 \]
Useful Properties

\[ \log_c(ab) = \log_c a + \log_c b \]

\[ a = c^{\log_c a} \]

\[ b = c^{\log_c b} \]

\[ ab = (c^{\log_c a})(c^{\log_c b}) = c^{\log_c a + \log_c b} \]

\[ \log_c(ab) = \log_c a + \log_c b \]
Useful Properties

\[ \log_b a^n = n \log_b a \]

\[ a = b^{\log_b a} \]

\[ a^n = (b^{\log_b a})^n = b^{n \log_b a} \]

\[ \log_b a^n = n \log_b a \]
Useful Properties

$$\log_b \left( \frac{1}{a} \right) = -\log_b a$$

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

$$\frac{1}{a} = \frac{1}{b^{\log_b a}} = b^{-\log_b a}$$

$$\log_b \left( \frac{1}{a} \right) = -\log_b a$$
Useful Properties

\[
\log_c (a/b) = \log_c a - \log_c b
\]

\[
\log_c (a/b) = \log_c (a \cdot \frac{1}{b})
\]

\[
= \log_c a + \log_c \frac{1}{b}
\]

\[
= \log_c a - \log_c b
\]
Change of Base

\[ \log_b a = \frac{\log_c a}{\log_c b} \]

\[ a = b^{\log_b a} \]

\[ a = c^{\log_c a} \]

\[ a = c^{\log_c b^{\log_b a}} = c^{\log_b a \log_c b} \]

\[ \log_c a = \log_b a \log_c b \]
Change of Base
(special case)

\[ \log_b a = \frac{1}{\log_a b} \]
Big-Oh Notation

\[ f(n) = O(g(n)) \]

- \( g \) is a loose upper bound for \( f \).
- We can find constants \( c \) and \( n_0 \) such that

\[ f(n) \leq cg(n), \text{ for } n \geq n_0 \]
Logarithms and Big-Oh

\[ \log_b a = \frac{\log_c a}{\log_c b} \]

- \( \log a \) in bases b and c are related by the multiplication of a constant, so

\[ \log_b n = O(\log_c n) \]

- i.e., the base doesn't actually matter
- We typically use base 2 in CS for convenience
Example 1
Exponentiation

- We want to compute $x^n$
- Trivial solution: multiply $x$ by itself $n-1$ times
  - $O(n)$
  - But
    $$ x^0 = 1 $$
    $$ x^n = \begin{cases} 
      x^{\lfloor n/2 \rfloor} x^{\lfloor n/2 \rfloor}, & \text{if } n \text{ is even} \\
      x^{\lfloor n/2 \rfloor} x^{\lfloor n/2 \rfloor} x, & \text{if } n \text{ is odd}
    \end{cases} $$
- So we compute $y = x^{(n/2)}$ and then
  - $x^n = y*y*x$
Example 1
Exponentiation

- Analysis

<table>
<thead>
<tr>
<th>Problem</th>
<th>Problem Size</th>
<th>Work</th>
<th>Recursion Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^n$</td>
<td>n</td>
<td>$O(1)$</td>
<td>1</td>
</tr>
<tr>
<td>$x^{(n/2)}$</td>
<td>n/2</td>
<td>$O(1)$</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x^0$</td>
<td>1</td>
<td>$O(1)$</td>
<td>H</td>
</tr>
</tbody>
</table>

- Total work $O(H)$ is how many times we have to divide $n$ by 2 in order to reach 1, i.e., $O(lg n)$
Example 2

Binary Search

- Problem: Look for x in an ordered list of numbers:
  \[ L = a_1, a_2, \ldots, a_n, \text{ where } a_1 \leq a_2 \leq \ldots \leq a_n \]

- Trivial Solution: Linear Search
  - Compare x to every number in L

- Better Solution
  - Binary search