Algorithm Design and Analysis

LECTURE 3
- Asymptotic Notation
- Basic Data Structures

Adam Smith

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Asymptotic notation

$O$-notation (upper bounds):

We write $f(n) = O(g(n))$ if there exist constants $c > 0$, $n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.

**Example:** $2n^2 = O(n^3)$  \hspace{2cm} (c = 1, n_0 = 2)

*funny, “one-way” equality*

functions, not values
Set Definition

\[ O(g(n)) = \{ f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \} \]

**Example:** \( 2n^2 \in O(n^3) \)

(*Logicians: \( \lambda n.2n^2 \in O(\lambda n.n^3) \), but it’s convenient to be sloppy, as long as we understand what’s really going on.*)
Examples

• \(10^6 n^3 + 2n^2 - n + 10 = O(n^3)\)

• \(n^{1/2} + \log(n) = O(n^{1/2})\)

• \(n (\log(n) + n^{1/2}) = O(n^{3/2})\)

• \(n = O(n^2)\)
**Ω-notation (lower bounds)**

*O*-notation is an *upper-bound* notation. It makes no sense to say $f(n)$ is at least $O(n^2)$.

$$\Omega(g(n)) = \{ f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \}$$

**Example:**

$$\sqrt{n} = \Omega(\lg n) \quad \text{ (} c = 1, n_0 = 16 \text{)}$$
**Ω-notation (lower bounds)**

- **Be careful:** “Any comparison-based sorting algorithm requires at least $O(n \log n)$ comparisons.”
  - Meaningless!
  - Use $\Omega$ for lower bounds.
**Θ-notation (tight bounds)**

\[ \Theta(g(n)) = O(g(n)) \cap \Omega(g(n)) \]

**Example:** \[ \frac{1}{2} n^2 - 2n = \Theta(n^2) \]

Polynomials are simple:
\[ a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0 = \Theta(n^d) \]
o-notation and ω-notation

$O$-notation and $Ω$-notation are like $\leq$ and $\geq$.

$o$-notation and $ω$-notation are like $<$ and $>$. 

\[
o(g(n)) = \{ f(n) : \text{for any constant } c > 0, \text{ there is a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0 \}\]

**Example:**

\[2n^2 = o(n^3) \quad (n_0 = 2/c)\]
o-notation and ω-notation

$O$-notation and $Ω$-notation are like $\leq$ and $\geq$.

$o$-notation and $ω$-notation are like $<$ and $>$. 

\[
ω(g(n)) = \{ f(n) : \text{for any constant } c > 0, \\
\text{there is a constant } n_0 > 0 \\
\text{such that } 0 \leq cg(n) < f(n) \\
\text{for all } n \geq n_0 \} 
\]

**Example:** \[ \sqrt{n} = ω(\lg n) \quad (n_0 = 1 + \frac{1}{c}) \]
Common Functions: Asymptotic Bounds

- **Polynomials.** $a_0 + a_1n + \ldots + a dn^d$ is $\Theta(n^d)$ if $a_d > 0$.
- **Polynomial time.** Running time is $O(n^d)$ for some constant $d$ independent of the input size $n$.
- **Logarithms.** $\log_a n = \Theta(\log_b n)$ for all constants $a, b > 0$.
  
  For every $x > 0$, $\log n = O(n^x)$.

- **Exponentials.** For all $r > 1$ and all $d > 0$, $n^d = O(r^n)$.
- **Factorial.** $n! = (\sqrt{2\pi n}) \left( \frac{n}{e} \right)^n (1 + o(1)) = 2^{\Theta(n \log n)}$ grows faster than every exponential.

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Sort by asymptotic order of growth

1. \( n^{1/\log(n)} \)
2. \( \log(n) \)
3. \( \sqrt{n} \)
4. \( n \)

5. \( n \log(n) = \Theta(\log(n!)) \)

6. \( n^2 \)
7. \( n^{1,000,000} \)
8. \( n^{1,000,000} \)
9. \( 2^n \)
10. \( n! \)

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Conventions for formulas

**Convention:** A set in a formula represents an anonymous function in the set.

**Example:**

\( f(n) = n^3 + O(n^2) \)

(right-hand side)

means

\( f(n) = n^3 + h(n) \)

for some \( h(n) \in O(n^2) \).
Convention: A set in a formula represents an anonymous function in the set.

Example: \( n^2 + O(n) = O(n^2) \) means for any \( f(n) \in O(n) \):

\[ n^2 + f(n) = h(n) \]

for some \( h(n) \in O(n^2) \).
• Transitivity.
  – If $f = O(g)$ and $g = O(h)$ then $f = O(h)$.
  – If $f = \Omega(g)$ and $g = \Omega(h)$ then $f = \Omega(h)$.
  – If $f = \Theta(g)$ and $g = \Theta(h)$ then $f = \Theta(h)$.

• Additivity.
  – If $f = O(h)$ and $g = O(h)$ then $f + g = O(h)$.
  – If $f = \Omega(h)$ and $g = \Omega(h)$ then $f + g = \Omega(h)$.
  – If $f = \Theta(h)$ and $g = O(h)$ then $f + g = \Theta(h)$.
Exercise: Show that $\log(n!) = \Theta(n \log n)$

- Stirling’s formula: $n! = \left(\sqrt{2\pi n}\right) \left(\frac{n}{e}\right)^n (1 + o(1))$

\[
\log(n!) = \log(\sqrt{2\pi n}) + n \log(n) - n \log(e) + \log(1 + o(1)) = \log(1) + o(1)
\]

Since $\log$ is continuous,

\[
\log(\sqrt{2\pi n}) + n \log(n) - n \log(e) + \log(1 + o(1)) = \log(1) + o(1)
\]

\[
= n \left(\log(n) - \log(e) + \frac{\log(2\pi n)}{n}\right) + o(1)
\]

\[
= n \left(\log(n) - O(1)\right) + o(1)
\]

\[
= n \log n (1 - O(\frac{1}{\log n})) + o(1)
\]

\[
= n \log n (1 \pm o(1)) = \Theta(n \log n)
\]
## Overview

<table>
<thead>
<tr>
<th>Notation</th>
<th>… means …</th>
<th>Think…</th>
<th>E.g.</th>
<th>Lim $f(n)/g(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(n) = O(n)$</td>
<td>$\exists \ c &gt; 0, \ n_0 &gt; 0, \ \forall \ n &gt; n_0 : 0 \leq f(n) &lt; cg(n)$</td>
<td>Upper bound</td>
<td>$100n^2$</td>
<td>If it exists, it is $&lt; \infty$</td>
</tr>
<tr>
<td>$f(n) = \Omega(g(n))$</td>
<td>$\exists \ c &gt; 0, \ n_0 &gt; 0, \ \forall \ n &gt; n_0 : 0 \leq cg(n) &lt; f(n)$</td>
<td>Lower bound</td>
<td>$n^{100}$</td>
<td>If it exists, it is $&gt; 0$</td>
</tr>
<tr>
<td>$f(n) = \Theta(g(n))$</td>
<td>both of the above: $f=\Omega(g)$ and $f = O(g)$</td>
<td>Tight bound</td>
<td>$\log(n!) = \Theta(n \log n)$</td>
<td>If it exists, it is $&gt; 0$ and $&lt; \infty$</td>
</tr>
<tr>
<td>$f(n) = o(g(n))$</td>
<td>$\forall \ c &gt; 0, \ n_0 &gt; 0, \ \forall \ n &gt; n_0 : 0 \leq f(n) &lt; cg(n)$</td>
<td>Strict upper bound</td>
<td>$n^2 = o(2^n)$</td>
<td>Limit exists, $= 0$</td>
</tr>
<tr>
<td>$f(n) = \omega(g(n))$</td>
<td>$\forall \ c &gt; 0, \ n_0 &gt; 0, \ \forall \ n &gt; n_0 : 0 \leq cg(n) &lt; f(n)$</td>
<td>Strict lower bound</td>
<td>$n^2 = \omega(\log n)$</td>
<td>Limit exists, $= \infty$</td>
</tr>
</tbody>
</table>