CIS 313:
Intermediate Data Structure

week of Jan 7
themes

• computational complexity, start to measure it
• simple data structures (mostly review)
• tree based structures
  • binary trees
  • binary heaps, binomial heaps
  • self adjusting trees: AVL, Red-Black
  • (2,4) trees, B-trees
• sorting, order statistics, voting
First algorithm

find the maximum number in an array

Input: a sequence of numbers $a_1, a_2, ..., a_n$
Output: the maximum number in the input sequence
Algorithm:
  $\max = a_1$
  for $i = 2$ to $n$:
    if $a_i > \max$:
      $\max = a_i$
  return $\max$

How long does this take?
Maybe: $n$ variable assignments, $n-1$ comparisons, $n-2$ increments, one return?
how do we talk about algorithm speed?

• use functions of the size of the input $n$ (typically the number of input numbers/items in this class), i.e., $T(n)$

• apply asymptotic notation for these functions

• it ignores constants and only focuses on the highest-order term
  • why? machine independence, constants not important asymptotically
  • asymptotically = “in the long run or in the limit”

• see description and definitions in text (section 3.1, pp 43-52)

• $O$, $\Omega$, $\Theta$, $o$, $\omega$
Time spent at 1,000,000 operations per second:

<table>
<thead>
<tr>
<th>input size</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>...</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>10^{-5} seconds</td>
<td>2 \cdot 10^{-5} seconds</td>
<td>3 \cdot 10^{-5} seconds</td>
<td>4 \cdot 10^{-5} seconds</td>
<td>5 \cdot 10^{-5} seconds</td>
<td>6 \cdot 10^{-5} seconds</td>
<td>...</td>
<td>10^{-4} seconds</td>
</tr>
<tr>
<td>(n^2)</td>
<td>10^{-4} seconds</td>
<td>4 \cdot 10^{-4} seconds</td>
<td>9 \cdot 10^{-4} seconds</td>
<td>1.6 \cdot 10^{-3} seconds</td>
<td>2.5 \cdot 10^{-3} seconds</td>
<td>3.6 \cdot 10^{-3} seconds</td>
<td>...</td>
<td>.01 second</td>
</tr>
<tr>
<td>(n^3)</td>
<td>10^{-3} seconds</td>
<td>8 \cdot 10^{-3} seconds</td>
<td>2.7 \cdot 10^{-3} seconds</td>
<td>6.4 \cdot 10^{-2} seconds</td>
<td>.125 second</td>
<td>.216 second</td>
<td>...</td>
<td>1 second</td>
</tr>
<tr>
<td>(n^{10})</td>
<td>2.7 hours</td>
<td>118 days</td>
<td>18 years</td>
<td>333 years</td>
<td>3,103 years</td>
<td>19,213 years</td>
<td>...</td>
<td>31,775 centuries</td>
</tr>
<tr>
<td>(2^n)</td>
<td>10^{-3} seconds</td>
<td>1 second</td>
<td>17 minutes</td>
<td>12 days</td>
<td>35.7 years</td>
<td>36,634 years</td>
<td>...</td>
<td>4 \cdot 10^{14} centuries</td>
</tr>
<tr>
<td>(3^n)</td>
<td>.06 second</td>
<td>58 minutes</td>
<td>6.5 years</td>
<td>3863 centuries</td>
<td>2 \cdot 10^8 centuries</td>
<td>1.3 \cdot 10^{13} centuries</td>
<td>...</td>
<td>1.6 \cdot 10^{32} centuries</td>
</tr>
<tr>
<td>(n!)</td>
<td>3.6 seconds</td>
<td>773 centuries</td>
<td>8 \cdot 10^{16} centuries</td>
<td>2.6 \cdot 10^{32} centuries</td>
<td>9.7 \cdot 10^{48} centuries</td>
<td>2.6 \cdot 10^{66} centuries</td>
<td>...</td>
<td>3 \cdot 10^{142} centuries</td>
</tr>
<tr>
<td>(2^{2^n})</td>
<td>&gt;10^{292} centuries</td>
<td>&gt;10^{315637} centuries</td>
<td>ouch!</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
big-Oh formally

$f(n) = O(g(n))$ if and only if (iff)

$\exists c > 0 \exists N \forall n \geq N \quad f(n) \leq c \cdot g(n)$

- $c$ is the dropped constant
- $N$ is the crossover point so that ...
- ... if $n$ is big enough $f$ is bounded above by $c \cdot g$
- the growth rate of $g$ bounds the growth rate of $f$ from above

example: let $f(n) = 3n^3 + 5n^2 + n + 17$

some true statements:

- $f(n) = O(n^3)$
- $f(n) = O(n^4)$
- $f(n) = O(17n^3)$
- $f(n) = 3n^3 + O(n^2)$
Big Omega and Theta

\[ f(n) = \Omega(g(n)) \iff \exists c > 0 \exists N \forall n \geq N \ f(n) \geq c \cdot g(n) \]

thus, the growth rate of \( g \) is less than or equal to the growth rate of \( f \) (ignoring the constant)

\[ f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n)) \]

- here \( f \) and \( g \) have the same growth rate
- sort of like saying \( A \leq B \) and \( A \geq B \) implies that \( A = B \)

now we can say \( f(n) = 3n^3 + 5n^2 + n + 17 \)
- \( f(n) = \Omega(n^3) \)
- \( f(n) = \Omega(n^2) \)
- \( f(n) = \Theta(n^3) \)
- \( f(n) = 3 \cdot n^3 + \Theta(n^2) \)
$f(n) = \Theta(g(n))$

$f(n) = O(g(n))$

$f(n) = \Omega(g(n))$
little-oh and little-omega

\( f(n) = o(g(n)) \text{ iff } \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \)

or

\( \forall c > 0 \exists N \forall n \geq N \ f(n) \leq c \cdot g(n) \)

in other words, the growth rate of \( f \) is strictly less than that of \( g \)

\( f(n) = \omega(g(n)) \text{ iff } \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \)

or

\( \forall c > 0 \exists N \forall n \geq N \ f(n) \geq c \cdot g(n) \)

the growth rate of \( f \) is strictly greater than that of \( g \)

examples:
- \( f(n) = o(n^2) \)
- \( f(n) = \omega(n^4) \)
- \( f(n) = 3 \cdot n^3 + o(n^3) \)
- \( \frac{1}{n} = o(1) \)
some properties

- Transitivity:
  \[ f(n) = \alpha(g(n)) \text{ and } g(n) = \alpha(h(n)) \implies f(n) = \alpha(h(n)) \quad (\alpha \in \{O, \Omega, \Theta, o, \omega\}) \]

- Reflexivity:
  \[ f(n) = \alpha(f(n)) \quad (\alpha \in \{O, \Omega, \Theta\}) \]

- Symmetry:
  \[ f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n)) \]

- Transpose Symmetry:
  \[ f(n) = O(g(n)) \iff g(n) = \Omega(f(n)) \]
  \[ f(n) = o(g(n)) \iff g(n) = \omega(f(n)) \]
common functions

• $n^k$, where $k$ is a constant (polynomial)
• $2^n$, $3^n$, $c^n$ (exponential)
• $\log_2 n$, $\log_c n$, $\ln n$ (logarithmic – usually $\log$ $n$ implies base 2)
  • fact: $\log_2 n = O(\log_c n)$ (why?)
• $O(n \log n)$ (also poly, but very common)
• $n!$ (factorial)
• $2^{(\log n)^2}$ (super-poly, sub-exponential) (ok, not so common)
other functions

• factorials: \( n! = n \cdot (n - 1) \cdot (n - 2) \cdots 3 \cdot 2 \cdot 1 \)

• Stirling’s Approximation: \( n! = \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot (1 + \Theta \left(\frac{1}{n}\right)) \)

• importantly \( \log n! = \Theta(n \cdot \log n) \)

• binomial coefficients

• Fibonacci sequence: \( F_0 = 0, F_1 = 1, F_{k+2} = F_{k+1} + F_k \)

• (Fibonacci used for AVL trees)
more examples

10 \log n + \log \log n \text{ is } O(\log n)? O(n)? O(n^{0.0000001})? \Omega(\log n)? O((\log n)^{0.5})? \Omega((\log n)^{0.5})

2^{3^{2000}} \text{ is } O(1)? \Omega(1)? 2^{3^{2000}} n \text{ is } O(n)?

2/n \text{ is } O(1/n)? O(1/\sqrt{n})? O(1/n^{1.7})? O(1)?

f(n) = \begin{cases} 
0.1 n & \text{if } n \text{ is odd} \\
3 n^2 & \text{if } n \text{ is even}
\end{cases}
\text{ is } O(n)? O(n^{1.5})? O(n^2)? \Omega(n)? \Omega(n^{1.5}) \Omega(n^2)