Asymptotic Notation

• Objectives: you should be able to
  – Recall the definitions big-O, big-Theta, and big-Omega asymptotic bounds on time costs,
  – Determine whether a given polynomial function is contained within a given asymptotic bound,
  – Derive asymptotic bounds for common iterative and recursive algorithms such as sorting and searching,
  – Use little-o notation to sort functions in order of increasing growth rate.
Asymptotic Notation

• Reading:
  – CLRS pp. 43-53
  – (Optional) Shaffer §§ 3.3 - 3.8
Asymptotic Notation

- We are interested in the resources consumed by computer algorithms: we often want to achieve more, with fewer resources.
  - The primary cost we track is time: we want to know how long does an algorithm take.
  - We are also sometimes interested in other resources such as memory space, network bandwidth, disk bandwidth, or other things.
- Step counting answers the time-cost question, but it's too difficult and excessively detailed.
Asymptotic Notation

• Example: insertion sort.
  - We found its time cost was a 2nd-degree (quadratic) polynomial.
  - We usually group all quadratic time-cost functions into one big group, denoted $\Theta(n^2)$.
    • Why? Because we assume the constant coefficients are roughly equal to one another, and not very important.
    • It's a coarse but practical way to guess at the speed of an algorithm.
Asymptotic Notation

- So all such 2nd-degree polynomials are grouped together $\Theta(n^2)$.
- That means we are talking about a set of functions.
- What exactly is the definition of this set?
Asymptotic Notation

• So all such 2nd-degree polynomials are grouped together $\Theta(n^2)$.

• That means we are talking about a set of functions.

• What exactly is the definition of this set?

$$\Theta(n^2) = \{ f(n) \mid \exists c_1 > 0, c_2 > 0, n_0 > 0, \forall n \geq n_0 (0 \leq c_1 n^2 \leq f(n) \leq c_2 n^2) \}$$
\[\Theta(g(n)) = \left\{ f(n) \mid \exists c_1 > 0, \exists c_2 > 0, \exists n_0 > 0,\right. \\
\left. \forall n \geq n_0, \right. \\
\left. (0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)) \right\}\]
$\Theta(g(n))$ is defined as...

\[\Theta(g(n)) = \left\{ f(n) \mid \exists c_1 > 0, \exists c_2 > 0, \exists n_0 > 0, \forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \right\}\]
Θ(\(g(n)\)) is defined as a set of functions such that:

\[
Θ(\(g(n)\)) = \left\{ f(n) \mid \exists c_1 > 0, \exists c_2 > 0, \exists n_0 > 0, \forall n \geq n_0, \right.

\left(0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\right) \right\}
\]
Θ(\(g(n)\)) is defined as a set of functions such that there exist positive constants \(c_1, c_2, n_0\) such that...

\[ \Theta(g(n)) = \left\{ f(n) \mid \exists c_1 > 0, \exists c_2 > 0, \exists n_0 > 0, \forall n \geq n_0, \right. \]
\[ \left. (0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)) \right\} \]
$\Theta(g(n))$ is defined as a set of functions such that there exist positive constants $c_1, c_2, n_0$ such that for all large $n$, that is, greater than or equal to $n_0$, the following holds:

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$
\( \Theta(g(n)) \) is defined as a set of functions such that there exist positive constants \( c_1, c_2, n_0 \) such that for all large \( n \) (that is, g.e. than \( n_0 \)), the members are bounded by \( g \).
Lower bound on $f(n)$: $f(n)$ cannot be too small.
Upper bound on $f(n)$: $f(n)$ cannot get too large.
Why so formal a definition?

So that we can prove theorems about it!
About this notation

• What does $n$ mean when we write $\Theta(n)$?
  – Is it a number?
  – Is it a free variable?

• What does $n$ mean when we write $\Theta(n^2)$?

• What does the one mean when we write $\Theta(1)$?
About this notation

• What does \( n \) mean when we write \( \Theta(n) \)?
  – Is it a number?
  – Is it a free variable?

• What does \( n \) mean when we write \( \Theta(n^2) \)?

• What does the one mean when we write \( \Theta(1) \)?

• It’s how we represent function \( g(n) \) in the definition.
  – Examples: \( g(n) = n \). \( g(n) = n^2 \). \( g(n) = 1 \).
About this notation

• It's conventional to write the “=” symbol even though we really mean set-membership, “∈”

• For example,

\[ 37 \, n^2 = \Theta(n^2). \]
About this notation

- It's conventional to write the “=” symbol even though we really mean set-membership, “∈”

- For example,

\[ 37 n^2 = \Theta(n^2). \]

  - Step-counting result for insertion sort:

\[
\frac{c_4 + c_5 + c_6}{2} n^2 + \left( c_1 + c_2 + c_3 + \frac{c_4 - c_5 - c_6}{2} + c_7 \right) n - c_2 - c_3 - c_4 - c_7 = \Theta \left( n^2 \right). 
\]
More commonly used than $\Theta(g(n))$

- but -

$O(g(n))$ only provides an **upper** bound
\(O(g(n)) = \{ \text{functions that grow no faster than } g(n) \} \)

\(O(g(n)) = \{ f(n) : \text{???} \} \)
\[ \mathcal{O}(g(n)) = \{ \text{functions that grow no faster than } g(n) \} \]

\[ O(g(n)) = \left\{ f(n) : \exists c > 0, n_0 > 0 \right\} \]

\[ \forall n \geq n_0 \]

\[ 0 \leq f(n) \leq c g(n) \]
Why use \( O(g(n)) \) when analyzing programs?
- Do we even care about lower bound?
- Special cases may be fast -- not our concern.

Often, \( O(g(n)) \) is the right choice for simple analysis
- \( \Theta(g(n)) \) is useful for some proofs

Be clear!
In Practice...

<table>
<thead>
<tr>
<th>Formal Definition</th>
<th>Informal Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n^2 = O(n^2)$</td>
<td>$(this\ class)$</td>
</tr>
<tr>
<td>$n = O(n^2)$</td>
<td>$(the\ rest\ of\ the\ world)$</td>
</tr>
<tr>
<td>$1 = O(n^2)$</td>
<td>$(the\ rest\ of\ the\ world)$</td>
</tr>
</tbody>
</table>

$n^2 \neq O(n^2)$

$n \neq O(n^2)$
Asymptotically Tight?

- $\Theta(g(n))$ “provides bounds which are asymptotically tight”
  - Provides a precise statement of the growth pattern: excludes anything growing faster or slower.

- $O(g(n))$ is a bigger set – it also includes all slower-growing functions.
A Metaphor

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

\[ ? \]

\[ ? \]

\[ ? \]

\[ ? \]
Topic 03: Asymptotic Notation

- An Example Method
- $\Theta(g(n))$
- $O(g(n))$
- $\Omega(g(n))$, $o(g(n))$, $\omega(g(n))$
- $\lg n$
- Ordering functions by growth
\(\Omega(g(n))\)

\[
\Omega(g(n)) = \{ \text{funcs that grow at least as fast as } g(n) \}
\]

\[
\Omega(g(n)) = \{ ??? \}
\]
\[ \Omega(g(n)) = \{ \text{funcs that grow at least as fast as } g(n) \} \]

\[ \Omega(g(n)) = \left\{ f(n): \exists c > 0, n_0 > 0 \quad \forall n \geq n_0 \quad c \cdot g(n) \leq f(n) \right\} \]
Θ, O, Ω

Θ: Upper and lower bounds
O: Upper bounds
Ω: Lower bounds

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

\[ f(n) = \Theta(g(n)) \rightarrow f(n) = \Omega(g(n)) \]
$\Theta$, $O$, $\Omega$

$\Theta$: Upper and lower bounds
$O$: Upper bounds
$\Omega$: Lower bounds

$\Theta(g(n)) \subseteq O(g(n))$

$\Theta(g(n)) \subseteq \Omega(g(n))$
\( o(g(n)) \)

- Provides an upper bound which is guaranteed not to be “asymptotically tight”

\[
1 = o(n^2) \\
n = o(n^2) \\
n^2 \neq o(n^2)
\]
\begin{align*}
o (g(n)) &= \{ \text{funcs that grow more slowly than } g(n) \} \\
\text{Concept:} \\
\text{Given some function } f(n) = o(g(n)), \text{ no matter how small a scaling factor we put on } g(n), \text{ and no matter how large a scaling factor we put on } f(n), \text{ } g(n) \text{ will eventually catch up, and pass } f(n). \end{align*}
\( o(g(n)) \)

\[ o(g(n)) = \{ \text{funcs that grow more slowly than } g(n) \} \]

\[ o(g(n)) = \{ f(n) : \text{ ??? } \} \]

**Hint:**

We only need a single scaling constant – which we'll apply to \( g(n) \), just like the previous definitions.
\( o(g(n)) \)

\[ o(g(n)) = \{ \text{funcs that grow more slowly than } g(n) \} \]

\[ o(g(n)) = \left\{ \begin{array}{l}
  f(n): \quad \forall c > 0 \\
  \exists n_0 > 0 \\
  \forall n \geq n_0 \\
  0 \leq f(n) < cg(n)
\end{array} \right\} \]
\( o(g(n)) \)

\[ o(g(n)) = \{ \text{funcs that grow more slowly than } g(n) \} \]

\[ o(g(n)) = \left\{ f(n) : \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \right\} \]
So Why Do We Care?

- Often, we want to explicitly state if two functions are different.
  - Suppose that we wanted to know the relationship between $l(n)$ and $p(n)$ ...

\[
\begin{align*}
  l(n) &= \Theta(lg^b n) \\
  p(n) &= \Theta(n^a)
\end{align*}
\]

Assume: $a > 0$
So Why Do We Care?

- Often, we want to explicitly state if two functions are **different**.
  - Suppose that we wanted to know the relationship between $l(n)$ and $p(n)$ ...

\[
\begin{align*}
l(n) &= \Theta(lg^b n) \\
p(n) &= \Theta(n^a) \\
l(n) &= o(p(n))
\end{align*}
\]

Assume: $a > 0$

We'll discuss the relationship of polynomials and logarithms in more depth later.
ω(g(n)) = \{ \text{funcs that grow more quickly than } g(n) \}

ω(g(n)) = \{ f(n) : \text{??} \}
\( \omega(g(n)) = \{ \text{funcs that grow more quickly than } g(n) \} \)

\[
\omega(g(n)) = \left\{ \begin{array}{l}
  f(n): \quad \forall c > 0 \\
  \exists n_0 > 0 \\
  \forall n \geq n_0 \\
  0 \leq c g(n) < f(n)
\end{array} \right\}
\]
\( f(n) = \omega(g(n)) \)

\( g(n) \quad ? \quad f(n) \)
\[ f(n) = \omega(g(n)) \]

\[ g(n) = o(f(n)) \]
Topic 03: Asymptotic Notation

- An Example Method
- $\Theta(g(n))$
- $O(g(n))$
- $\Omega(g(n))$, $o(g(n))$, $\omega(g(n))$
- $\log n$
- Ordering functions by growth
Some Definitions

\[ \log n = \log_{10} n \]
\[ \ln n = \log_e n \]
\[ \lg n = \log_2 n \]

\[ \lg^k n = (\lg n)^k \]
\[ \lg \lg n = \lg (\lg n) \]

\[ \lg^* n = \text{"iterated logarithm"} \]
Log Identities

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

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

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

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

\[ \log_b a = \frac{1}{\log_a b} \]

\[ a^{\log_b c} = c^{\log_b a} \]
$\lg n$

- Algorithms with logs in them are very common
- Logarithms grow very slowly

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n^2$</th>
<th>$\lg n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>4</td>
</tr>
<tr>
<td>32</td>
<td>1024</td>
<td>5</td>
</tr>
</tbody>
</table>
$\Theta$ - equivalence

$\lg n = \Theta (\log n) = \Theta (\ln n)$

Why?
We've already seen how polynomials and logs relate:

\[ \log_b n = o(n^a) \]

\[ \log^{100} n = o(n) \]

Assume: \( a > 0 \)
• So how do we compare these pairs of functions?

\[ n^a \quad ? \quad n^a \log^b n \]

\[ n^a \log^b n \quad ? \quad n^{a+\epsilon} \]

Assume: \( a, b, \epsilon > 0 \)
• So how do we compare these pairs of functions?

\[ n^a = o\left(n^a \log^b n\right) \]

\[ n^a \log^b n = o\left(n^{a+\epsilon}\right) \]

Assume: \( a, b, \epsilon > 0 \)
$2^n$

- $2^n$ is very huge – far huger than any polynomial.

$$n^a = o(2^n)$$
$2^n$

- $2^n$ is very huge – far huger than any polynomial.

NOTE:

I'm not claiming that I've proved this step. But it's a reasonable transform.

\[
n^a = o\left(2^n\right)
\]

\[
lg n^a = o\left(lg 2^n\right)
\]

\[
a lg n = o\left(n\right)
\]
Topic 03: Asymptotic Notation

• An Example Method
• \( \Theta(g(n)) \)
• \( O(g(n)) \)
• \( \Omega(g(n)), o(g(n)), \omega(g(n)) \)
• \( \lg n \)

• Ordering functions by growth
Ordering Functions by Growth

- We can organize functions into groups
  - Some you've seen already
  - Many you haven't yet
Some Famous Families

What it's Famous For

\[ \Theta(1) \]
\[ \Theta(\log n) \]
\[ \Theta(n) \]
\[ \Theta(n \log n) \]
\[ \Theta(n^2) \]
\[ \Theta(2^n) \]
\[ \Theta(n^n) \]
# Some Famous Families

<table>
<thead>
<tr>
<th>What it's Famous For</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta(1)$ simple ops</td>
</tr>
<tr>
<td>$\Theta(\log n)$ binary search</td>
</tr>
<tr>
<td>$\Theta(n)$ inspect all elements</td>
</tr>
<tr>
<td>$\Theta(n \log n)$ the best sorts</td>
</tr>
<tr>
<td>$\Theta(n^2)$ check all pairs</td>
</tr>
<tr>
<td>$\Theta(2^n)$ all boolean combinations</td>
</tr>
<tr>
<td>$\Theta(n^n)$ give up, you've lost</td>
</tr>
</tbody>
</table>
### Some Famous Families

<table>
<thead>
<tr>
<th></th>
<th>$n=1$</th>
<th>$n=10$</th>
<th>$n=1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta(1)$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\Theta(\log n)$</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>$\Theta(n)$</td>
<td>1</td>
<td>10</td>
<td>1000</td>
</tr>
<tr>
<td>$\Theta(n \log n)$</td>
<td>0</td>
<td>10</td>
<td>3000</td>
</tr>
<tr>
<td>$\Theta(n^2)$</td>
<td>1</td>
<td>100</td>
<td>1000000</td>
</tr>
<tr>
<td>$\Theta(2^n)$</td>
<td>2</td>
<td>1024</td>
<td>googol, cubed</td>
</tr>
<tr>
<td>$\Theta(n^n)$</td>
<td>1</td>
<td>1000000000000</td>
<td>???</td>
</tr>
</tbody>
</table>
Noteworthy Names

Θ(1)  "constant time"
Θ(\log n)  "log"
o(\(n\))  "sublinear"
Θ(\(n\))  "linear"
Θ(\(n \log n\))  "log-linear" or "linearithmic"
Θ(\(n^k\))  "polynomial"
Θ(\(2^{n^k}\))  "exponential"
Topic 03: Asymptotic Notation

- An Example Method
- \( \Theta(g(n)) \)
- \( O(g(n)) \)
- \( \Omega(g(n)), o(g(n)), \omega(g(n)) \)
- \( \log n \)
- Ordering functions by growth

**Summary**