Lecture 2
Asymptotic Notation
- $O$, $\Omega$, and $\Theta$-notation
Recurrences
- Substitution method
- Iterating the recurrence
- Recursion tree
- Master method

Prof. Erik Demaine
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 \( n \geq n_0 \).
Asymptotic notation

\textbf{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 $n \geq n_0$.

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

*functions, not values*
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{1cm} (c = 1, n_0 = 2)

functions, not values

funny, “one-way” equality
Set definition of O-notation

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

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

**Example:** $2n^2 \in O(n^3)$
Set definition of $O$-notation

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

**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.)
Macro substitution

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

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

**Example:** \( f(n) = n^3 + O(n^2) \) means
\[
f(n) = n^3 + h(n)
\]
for some \( h(n) \in O(n^2) \).
Macro substitution

**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 \( h(n) \in O(n^2) \).
Ω-notation (lower bounds)

O-notation is an upper-bound notation. It makes no sense to say $f(n)$ is at least $O(n^2)$. 
\( \Omega \text{-notation (lower bounds)} \)

\( O \text{-notation is an upper-bound notation. It makes no sense to say } f(n) \text{ is at least } O(n^2). \)

\[
\Omega(g(n)) = \{ f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \}
\]
\( \Omega \)-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:** \( \sqrt{n} = \Omega(lg\,n) \) \((c = 1, n_0 = 16)\)
Θ-notation (tight bounds)

\[ \Theta(g(n)) = O(g(n)) \cap \Omega(g(n)) \]
Θ-notation (tight bounds)

$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

**Example:**  $$\frac{1}{2}n^2 - 2n = \Theta(n^2)$$
\( o \)-notation and \( \omega \)-notation

\( O \)-notation and \( \Omega \)-notation are like \( \leq \) and \( \geq \).
\( o \)-notation and \( \omega \)-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:** \( 2n^2 = o(n^3) \) \( (n_0 = 2/c) \)
\( \omega \)-notation and \( \Omega \)-notation

\( O \)-notation and \( \Omega \)-notation are like \( \leq \) and \( \geq \).
\( o \)-notation and \( \omega \)-notation are like \( < \) and \( > \).

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

**Example:** \( \sqrt{n} = \omega(\lg n) \quad (n_0 = 1+1/c) \)
Solving recurrences

- The analysis of merge sort from *Lecture 1* required us to solve a recurrence.
- Recurrences are like solving integrals, differential equations, etc.
  - Learn a few tricks.
- *Lecture 3*: Applications of recurrences to divide-and-conquer algorithms.
Substitution method

The most general method:

1. **Guess** the form of the solution.
2. **Verify** by induction.
3. **Solve** for constants.
Substitution method

The most general method:
1. **Guess** the form of the solution.
2. **Verify** by induction.
3. **Solve** for constants.

**Example:** \( T(n) = 4T(n/2) + n \)
- [Assume that \( T(1) = \Theta(1) \).]
- Guess \( O(n^3) \). (Prove \( O \) and \( \Omega \) separately.)
- Assume that \( T(k) \leq ck^3 \) for \( k < n \).
- Prove \( T(n) \leq cn^3 \) by induction.
Example of substitution

\[ T(n) = 4T\left(\frac{n}{2}\right) + n \]
\[ \leq 4c\left(\frac{n}{2}\right)^3 + n \]
\[ = \left(\frac{c}{2}\right)n^3 + n \]
\[ = cn^3 - (\left(\frac{c}{2}\right)n^3 - n) \leftarrow \text{desired} - \text{residual} \]
\[ \leq cn^3 \leftarrow \text{desired} \]

whenever \( (c/2)n^3 - n \geq 0 \), for example, if \( c \geq 2 \) and \( n \geq 1 \).
Example (continued)

- We must also handle the initial conditions, that is, ground the induction with base cases.

  - **Base:** $T(n) = \Theta(1)$ for all $n < n_0$, where $n_0$ is a suitable constant.

  - For $1 \leq n < n_0$, we have “$\Theta(1)$” $\leq cn^3$, if we pick $c$ big enough.
Example (continued)

- We must also handle the initial conditions, that is, ground the induction with base cases.

- **Base:** \( T(n) = \Theta(1) \) for all \( n < n_0 \), where \( n_0 \) is a suitable constant.

- For \( 1 \leq n < n_0 \), we have “\( \Theta(1) \)” \( \leq cn^3 \), if we pick \( c \) big enough.

---

This bound is not tight!
A tighter upper bound?

We shall prove that $T(n) = O(n^2)$. 
A tighter upper bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \leq ck^2$ for $k < n$:

$$
T(n) = 4T(n/2) + n \\
\leq 4c(n/2)^2 + n \\
= cn^2 + n \\
= O(n^2)
$$
A tighter upper bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \leq ck^2$ for $k < n$:

\[
T(n) = 4T(n/2) + n \\
\leq 4c(n/2)^2 + n \\
= cn^2 + n \\
= O(n^2) \quad \text{Wrong!} \quad \text{We must prove the I.H.}
\]
A tighter upper bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \leq ck^2$ for $k < n$:

\[
T(n) = 4T(n/2) + n \\
\leq 4c(n/2)^2 + n \\
= cn^2 + n \\
= O(n^2) \quad \text{Wrong! We must prove the I.H.}
\]

\[
= cn^2 - (-n) \quad [\text{desired} - \text{residual}] \\
\leq cn^2 \quad \text{for no choice of } c > 0. \quad \text{Lose!}
\]
A tighter upper bound!

**Idea:** Strengthen the inductive hypothesis.
- *Subtract* a low-order term.

*Inductive hypothesis:* \( T(k) \leq c_1 k^2 - c_2 k \) for \( k < n \).
A tighter upper bound!

**Idea:** Strengthen the inductive hypothesis.
- *Subtract* a low-order term.

**Inductive hypothesis:** \( T(k) \leq c_1 k^2 - c_2 k \) for \( k < n \).

\[
T(n) = 4T(n/2) + n
\]
\[
= 4(c_1 (n/2)^2 - c_2 (n/2)) + n
\]
\[
= c_1 n^2 - 2c_2 n + n
\]
\[
= c_1 n^2 - c_2 n - (c_2 n - n)
\]
\[
\leq c_1 n^2 - c_2 n \quad \text{if } c_2 \geq 1.
\]
A tighter upper bound!

**Idea:** Strengthen the inductive hypothesis.

- *Subtract* a low-order term.

**Inductive hypothesis:** \( T(k) \leq c_1 k^2 - c_2 k \) for \( k < n \).

\[
T(n) = 4T(n/2) + n \\
= 4(c_1 (n/2)^2 - c_2 (n/2)) + n \\
= c_1 n^2 - 2c_2 n + n \\
= c_1 n^2 - c_2 n - (c_2 n - n) \\
\leq c_1 n^2 - c_2 n \quad \text{if} \ c_2 \geq 1.
\]

Pick \( c_1 \) big enough to handle the initial conditions.
Recursion-tree method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- The recursion-tree method can be unreliable, just like any method that uses ellipses (...).
- The recursion-tree method promotes intuition, however.
- The recursion tree method is good for generating guesses for the substitution method.
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$: 
Example of recursion tree

Solve \( T(n) = T(n/4) + T(n/2) + n^2 \):

\[ T(n) \]
Example of recursion tree

Solve \( T(n) = T(n/4) + T(n/2) + n^2: \)

\[
\begin{array}{c}
T(n/4) \\
\quad \quad n^2 \\
T(n/2)
\end{array}
\]
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

```
  n^2
 /   \
(n/4)^2  (n/2)^2
 /     \   /     \
T(n/16) T(n/8) T(n/8) T(n/4)
```
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

```
   n^2
  /   \
(n/4)^2  (n/2)^2
  /   \
(n/16)^2  (n/8)^2
   /   \
    :    :    :    :
Θ(1)
```
Example of recursion tree

Solve \( T(n) = T(n/4) + T(n/2) + n^2 \):

\[
\begin{array}{c}
\Theta(1) \\
/ \\
\vdots \\
(n/4)^2 \\
/ \\
(n/16)^2 \\
/ \\
(n/8)^2 \\
/ \\
(n/8)^2 \\
/ \\
(n/4)^2 \\
/ \\
(n/2)^2 \\
/ \\
(n/4)^2 \\
/ \\
(n/2)^2 \\
/ \\
(\ldots) \\
/ \\
(n^2) \\
/ \\
(\ldots) \\
/ \\
(n^2) \\
/ \\
(\ldots) \\
/ \\
(n^2)
\end{array}
\]
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

\[
\begin{align*}
T(n) &= n^2 - \frac{5}{16}n^2 - \frac{25}{256}n^2 - \ldots \\
\text{Total} &= n^2 \left( 1 + \frac{5}{16} + \left( \frac{5}{16} \right)^2 + \left( \frac{5}{16} \right)^3 + \ldots \right) \\
&= \Theta(n^2) \quad \text{geometric series}
\end{align*}
\]
The master method

The master method applies to recurrences of the form

\[ T(n) = a \ T(n/b) + f(n) , \]

where \( a \geq 1 \), \( b > 1 \), and \( f \) is asymptotically positive.
Three common cases

Compare $f(n)$ with $n^{\log ba}$:

1. $f(n) = O(n^{\log ba - \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially slower than $n^{\log ba}$ (by an $n^\varepsilon$ factor).

   Solution: $T(n) = \Theta(n^{\log ba})$. 
Three common cases

Compare $f(n)$ with $n^{\log_b a}$:

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.
   
   • $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an $n^\varepsilon$ factor).

   **Solution:** $T(n) = \Theta(n^{\log_b a})$.

2. $f(n) = \Theta(n^{\log_b a} \log^k n)$ for some constant $k \geq 0$.
   
   • $f(n)$ and $n^{\log_b a}$ grow at similar rates.

   **Solution:** $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.
Three common cases (cont.)

Compare \( f(n) \) with \( n^{\log_b a} \):

3. \( f(n) = \Omega(n^{\log_b a + \varepsilon}) \) for some constant \( \varepsilon > 0 \).
   
   - \( f(n) \) grows polynomially faster than \( n^{\log_b a} \) (by an \( n^{\varepsilon} \) factor),

   and \( f(n) \) satisfies the \textit{regularity condition} that 
   \( af(n/b) \leq cf(n) \) for some constant \( c < 1 \).

\textbf{Solution:} \( T(n) = \Theta(f(n)) \).
Examples

Ex.  \[ T(n) = 4T(n/2) + n \]
\[ a = 4, \ b = 2 \implies n^{\log_b a} = n^2; \ f(n) = n. \]

Case 1: \[ f(n) = O(n^{2 - \varepsilon}) \text{ for } \varepsilon = 1. \]
\[ \therefore T(n) = \Theta(n^2). \]
Examples

**Ex.** \( T(n) = 4T(n/2) + n \)
\( a = 4, \ b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n. \)

**Case 1:** \( f(n) = O(n^2 - \varepsilon) \) for \( \varepsilon = 1. \)
\( \therefore T(n) = \Theta(n^2). \)

**Ex.** \( T(n) = 4T(n/2) + n^2 \)
\( a = 4, \ b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2. \)

**Case 2:** \( f(n) = \Theta(n^2 \log^0 n), \) that is, \( k = 0. \)
\( \therefore T(n) = \Theta(n^2 \log n). \)
Examples

Ex. $T(n) = 4T(n/2) + n^3$

$a = 4$, $b = 2$ $\Rightarrow$ $n^{\log_b a} = n^2$; $f(n) = n^3$.

Case 3: $f(n) = \Omega(n^2 + \varepsilon)$ for $\varepsilon = 1$

and $4(n/2)^3 \leq cn^3$ (reg. cond.) for $c = 1/2$.

$\therefore$ $T(n) = \Theta(n^3)$. 
Examples

\textbf{Ex.} \quad T(n) = 4T(n/2) + n^3 \\
\quad a = 4, \ b = 2 \Rightarrow n^{\log_b a} = n^2; \ f(n) = n^3. \\
\textbf{Case 3:} \quad f(n) = \Omega(n^2 + \varepsilon) \text{ for } \varepsilon = 1 \\
\text{and } 4(n/2)^3 \leq cn^3 \text{ (reg. cond.) for } c = 1/2. \\
\therefore T(n) = \Theta(n^3). \\

\textbf{Ex.} \quad T(n) = 4T(n/2) + n^2/\log n \\
\quad a = 4, \ b = 2 \Rightarrow n^{\log_b a} = n^2; \ f(n) = n^2/\log n. \\
Master method does not apply. In particular, for every constant \( \varepsilon > 0 \), we have \( n^{\varepsilon} = \omega(\log n) \).
Idea of master theorem

Recursion tree:

```
     f(n)
    /   \   a
f(n/b)  f(n/b)    ···  f(n/b)
    /         \   a
f(n/b^2) f(n/b^2)    ···  f(n/b^2)
      \     /
       ···
      /    /
    T(1)
```
Idea of master theorem

**Recursion tree:**

```
  f(n) --------------- f(n)
     |                 |
     a                 a

  f(n/b)     f(n/b)   ... f(n/b) ------ af(n/b)
          |               |
          a             a

  f(n/b^2) f(n/b^2) ... f(n/b^2) ------ a^2 f(n/b^2)

  ...  

  T(1)
```
Idea of master theorem

Recursion tree:

\[ h = \log_b{n} \]

\[ f(n) \quad \frac{a}{f(n)} \quad f(n) \]

\[ f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad \frac{af(n/b)}{a} \]

\[ f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad \frac{a^2f(n/b^2)}{a^2} \]

\[ \vdots \]

\[ T(1) \]
Idea of master theorem

**Recursion tree:**

- $f(n)$
- $a$
- $f(n/b)$, $f(n/b)$, ...
- $af(n/b)$
- $a^2f(n/b^2)$
- #leaves = $a^h$
- $= a^{\log_b n}$
- $= n^{\log_b a} T(1)$

$h = \log_b n$

$T(1)$
Idea of master theorem

**Recursion tree:**

\[ \begin{align*}
f(n) & \quad f(n) \\
\downarrow & \quad \downarrow \\
f(n/b) & \quad f(n/b) \\
\downarrow & \quad \downarrow \\
f(n/b^2) & \quad f(n/b^2) \\
\vdots & \quad \vdots \\
T(1) & \quad \Theta(n^{\log_b a} T(1))
\end{align*} \]

**CASE 1:** The weight increases geometrically from the root to the leaves. The leaves hold a constant fraction of the total weight.
Idea of master theorem

**Recursion tree:**

\[ f(n) \quad \cdots \quad f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b) \quad a \]

\[ f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad a \]

\[ f(n) \quad \cdots \quad f(n) \quad a \]

\[ h = \log_b n \]

**CASE 2: \((k = 0)\)** The weight is approximately the same on each of the \(\log_b n\) levels.

\[ n^{\log_b a} T(1) \]

\[ \Theta(n^{\log_b a \lg n}) \]
Idea of master theorem

Recursion tree:

\[ f(n) \quad a \]  
\[ f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad a^2 f(n/b^2) \]
\[ h = \log_b n \]

CASE 3: The weight decreases geometrically from the root to the leaves. The root holds a constant fraction of the total weight.

\[ n^{\log_b a} T(1) \]
\[ \Theta(f(n)) \]
Appendix: geometric series

$$1 + x + x^2 + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x} \quad \text{for } x \neq 1$$

$$1 + x + x^2 + \cdots = \frac{1}{1 - x} \quad \text{for } |x| < 1$$