LECTURE 2

Asymptotic Notation
• $O$-, $\Omega$-, and $\Theta$-notation

Recurrences
• Substitution method
• Iterating the recurrence
• Recursion tree
• Master method

Prof. Charles E. Leiserson
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

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

functions, not values  
funny, “one-way” equality

© 2001–4 by Charles E. Leiserson
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

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

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

*O*-notation is an *upper-bound* notation. It makes no sense to say \( f(n) \) is at least \( O(n^2) \).
Ω–notation (lower bounds)

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

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

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

**Example:** \( \sqrt{n} = \Omega(\lg n) \)
\( \Theta(g(n)) = \Omega(g(n)) \cap \Omega(g(n)) \)
Θ-notation (tight bounds)

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

**Example:** \( \frac{1}{2} n^2 - 2n = \Theta(n^2) \)
Θ-notation (tight bounds)

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

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

*Theorem.* The leading constant and low-order terms don’t matter. \(\square\)
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

\[ T(n) = 4T(n/2) + n \]
\[ \leq 4c(n/2)^3 + n \]
\[ = (c/2)n^3 + n \]
\[ = cn^3 - ((c/2)n^3 - n) \quad \text{desired} - \text{residual} \]
\[ \leq cn^3 \quad \text{desired} \]

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 $k < n$:

$T(n) = 4T(n/2) + n$
$\leq 4c(n/2)^2 + n$
$= cn^2 + n$
$= O(n^2)$  \textbf{Wrong!} 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 \( k < n \):

\[
T(n) = 4T(n/2) + n \\
\leq 4c(n/2)^2 + n \\
= cn^2 + n \\
= O(n^2) \quad \text{Wrong! We must prove the I.H.} \\
= cn^2 - (-n) \quad \text{[desired – residual]} \\
\leq cn^2 \quad \text{for no choice of } c > 0. \quad \text{Lose!}
\]
A tighter upper bound!

**IDEA:** Strengthen the inductive hypothesis.

- *Subtract* a low-order term.

*Inductive hypothesis:* \( T(k) \leq c_1k^2 - c_2k \) 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 \( k < n \).

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

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

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

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

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

$\begin{align*}
    & n^2 \\
    & \quad | \\
    & (n/4)^2 \\
    & \quad | \\
    & T(n/16) \quad T(n/8) \\
    & \quad | \\
    & (n/2)^2 \\
    & \quad | \\
    & T(n/8) \quad T(n/8) \quad T(n/4)
\end{align*}$
Example of recursion tree

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

\[
\begin{align*}
\Theta(1) & \\
(n/4)^2 & \quad (n/2)^2 \\
(n/16)^2 & \quad (n/8)^2 & \quad (n/8)^2 & \quad (n/4)^2
\end{align*}
\]
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 \( T(n) = T(n/4) + T(n/2) + n^2: \)

\[
T(n) = \begin{cases} 
\frac{5}{16}n^2 & \text{if } n = \frac{1}{16}
\end{cases}
\]

\( \Theta(1) \)

© 2001–4 by Charles E. Leiserson
Example of recursion tree

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

\[
\begin{align*}
T(n) &= T(n/4) + T(n/2) + n^2 \\
&= (n/4)^2 + (n/2)^2 + n^2 + n^2 \\
&= (n/4)^2 + (n/2)^2 + (n/8)^2 + (n/2)^2 + (n/4)^2 + (n/16)^2 + \cdots \\
&= \sum_{i=0}^{\infty} \frac{n^2}{16^i} \\
&= \frac{1}{1 - 1/16} \cdot n^2 (1 + \frac{5}{16} + (\frac{5}{16})^2 + (\frac{5}{16})^3 + \cdots) \\
&= \Theta(n^2)
\end{align*}
\]

geometric series
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:** $T(n) = \Theta(n^{\log ba})$. 

© 2001–4 by Charles E. Leiserson
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:** \( T(n) = \Theta(n^{\log_b a \log^{k+1} n}) \).

© 2001–4 by Charles E. Leiserson
Three common cases (cont.)

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

3. $f(n) = \Omega(n^{\log ba} + \varepsilon)$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially faster than $n^{\log ba}$ (by an $n^\varepsilon$ factor),
   - and $f(n)$ satisfies the regularity condition that $af(n/b) \leq cf(n)$ for some constant $c < 1$.

Solution: $T(n) = \Theta(f(n))$. 

© 2001–4 by Charles E. Leiserson
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 \( \varepsilon = 1. \)

\( \therefore \ T(n) = \Theta(n^2). \)
Examples

**Ex.**  \( T(n) = 4T(n/2) + n \)

\[ a = 4, \ b = 2 \implies n^{\log b a} = n^2; \ f(n) = n. \]

**CASE 1:** \( f(n) = O(n^{2 - \varepsilon}) \) for \( \varepsilon = 1. \)

\[ \therefore T(n) = \Theta(n^2). \]

**Ex.**  \( T(n) = 4T(n/2) + n^2 \)

\[ a = 4, \ b = 2 \implies n^{\log b a} = n^2; \ f(n) = n^2. \]

**CASE 2:** \( f(n) = \Theta(n^2 \lg^0 n) \), that is, \( k = 0. \)

\[ \therefore T(n) = \Theta(n^2 \lg n). \]
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

**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 \( c = 1/2. \)

\[ \therefore \ T(n) = \Theta(n^3). \]

**Ex.**  \( T(n) = 4T(n/2) + n^2/\lg n \)

\[ a = 4, \ b = 2 \Rightarrow n^{\log_b a} = n^2; \ f(n) = n^2/\lg n. \]

Master method does not apply. In particular, for every constant \( \varepsilon > 0, \) we have \( n^\varepsilon = \omega(\lg n). \)
Idea of master theorem

Recursion tree:

\[ f(n) \]
\[ f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \]
\[ f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \]
\[ \vdots \]
\[ T(1) \]
Idea of master theorem

Recursion tree:

\[
\begin{align*}
&f(n) \\
&a \\
&f(n/b) \\
&f(n/b) \quad \cdots \quad f(n/b) \\
&af(n/b) \\
&f(n/b^2) \\
&f(n/b^2) \quad \cdots \quad f(n/b^2) \\
&a^2 f(n/b^2) \\
&T(1)
\end{align*}
\]
Idea of master theorem

Recursion tree:

\[ T(n) = \begin{cases} 
T(1) & \text{if } n = 1 \\
a T(n/b) + f(n) & \text{if } n/b \leq a \cdot f(n) \leq n/b \end{cases} \]

where

- \( a \) is the number of subproblems in the recursion.
- \( f(n) \) is the work done outside the subproblems.
- \( b \) is the factor by which the size of the subproblems decreases.
- \( h = \log_b n \) is the height of the recursion tree.

The recursion tree is shown with levels of subproblems and the recurrence relation is applied recursively down the tree.

© 2001–4 by Charles E. Leiserson
Idea of master theorem

Recursion tree:

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

\[ h = \log_b n \]

#leaves = \( a^h \)
\[ = a^{\log_b n} \]
\[ = n^{\log_b a} \]

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

\[ \leq \]

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

Recursion tree:

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

\[ h = \log_b n \]

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

\[ \Theta(n^{\log_b a} T(1)) \]
Idea of master theorem

Recursion tree:

\[ f(n) \]

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

\[ a \]

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

\[ a \]

\[ T(1) \]

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

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

**Recursion tree:**

\[ h = \log_b n \]

\[ f(n) \]

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

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

\[ \vdots \]

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

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

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

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