Advanced Algorithmics (4AP)  
Maths – Recurrences, NP, etc  

Jaak Vilo  
2009 spring
Recurrences

- $T(n) = T(n/2) + n \Rightarrow T(n) = O(n \log n)$

- $T(n) = a \cdot T(b \cdot n) + f(n)$

- How to solve

- Master theorem
• http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-046JFall-2005/VideoLectures/detail/embed02.htm

• http://en.wikipedia.org/wiki/Master_theorem
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*
• Expand into series

• Guess

• Induction

• ...
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

\[ T(n) = 4T(n/2) + n \]
\[ \leq 4c(n/2)^3 + n \]
\[ = (c/2)n^3 + n \]
\[ = cn^3 - ((c/2)n^3 - n) \]
\[ \leq cn^3 \]

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.

This bound is not tight!
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 $k < n$:

$T(n) = 4T(n/2) + n$

$\leq 4c(n/2)^2 + n$

$= cn^2 + n$

$= O(n^2)$  \(\times\)  \text{Wrong!}  \text{ We must prove the I.H.}

$= cn^2 - (-n)  \quad [\text{desired} - \text{residual}]$

$\leq cn^2  \quad \text{for no choice of } c > 0.  \text{ Lose!}$
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 \( k < n \).

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

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

$T(n)$
Example of recursion tree

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

\[
\begin{align*}
\Theta(1) & \quad \vdots \\
(n/16)^2 & \quad (n/8)^2 & \quad (n/8)^2 & \quad (n/4)^2 & \quad \frac{5}{16}n^2 & \quad \frac{25}{256}n^2 \\
(n/4)^2 & \quad (n/2)^2 & \quad n^2
\end{align*}
\]

Total $= n^2 \left( 1 + \frac{5}{16} + \left( \frac{5}{16} \right)^2 + \left( \frac{5}{16} \right)^3 + \cdots \right)$

$= \Theta(n^2)$  geometric series
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 $n^{\log_b a}$:

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. $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$.
   - $f(n)$ and $n^{\log_b a}$ grow at similar rates.
   
   **Solution:** $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.
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 regularity condition that $af(n/b) \leq cf(n)$ for some constant $c < 1$.

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

• Many divide-and-conquer recurrence equations have the form:

\[ T(n) = \begin{cases} 
  c & \text{if } n < d \\
  aT(n/b) + f(n) & \text{if } n \geq d 
\end{cases} \]

• The Master Theorem:
  1. if \( f(n) = O(n^{\log_b a - \varepsilon}) \), then \( T(n) = \Theta(n^{\log_b a}) \)
  2. if \( f(n) = \Theta(n^{\log_b a \log^k n}) \), then \( T(n) = \Theta(n^{\log_b a \log^{k+1} n}) \)
  3. if \( f(n) = \Omega(n^{\log_b a + \varepsilon}) \), then \( T(n) = \Theta(f(n)) \),
     provided \( af(n/b) \leq \delta f(n) \) for some \( \delta < 1 \), for all \( n \geq d \)
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:** \( f(n) = \Theta(n^2 \lg^k n) \), 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 \( c = 1/2. \)
\( \therefore T(n) = \Theta(n^3). \)

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:

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

Recursion tree:

\[ h = \log_{b} n \]

\[ T(1) = n^{\log_{b} a} \]

\[ \#\text{leaves} = a^{h} = a^{\log_{b} n} = n^{\log_{b} a} \]
Idea of master theorem

**Recursion tree:**

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

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

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

\[ \vdots \]

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

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

**Recursion tree:**

\[ f(n) \rightarrow f(n/b) \rightarrow f(n/b^2) \rightarrow \cdots \rightarrow f(1) \]

\[ i = \log_b n \]

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

\[ \Theta(n^{\log_b a} \log n) \]

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

Recursion tree:

\[ f(n) \quad f(n) \quad f(n) \]
\[ f(n/b) \quad f(n/b) \quad \ldots \quad f(n/b) \quad a \cdot f(n/b) \]
\[ f(n/b^2) \quad f(n/b^2) \quad \ldots \quad f(n/b^2) \quad a^2 \cdot f(n/b^2) \]
\[ \vdots \]
\[ T(1) \quad n^{\log_b a} T(1) \quad \Theta(f(n)) \]

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

September 12, 2005

Copyright © 2001-5 Erik D. Demaine and Charles E. Leiserson

L2.56
Master Method

- Many divide-and-conquer recurrence equations have the form:

\[ T(n) = \begin{cases} 
  c & \text{if } n < d \\
  aT(n/b) + f(n) & \text{if } n \geq d 
\end{cases} \]

- The Master Theorem:
1. if \( f(n) \) is \( O(n^{\log_b a - \varepsilon}) \), then \( T(n) \) is \( \Theta(n^{\log_b a}) \)
2. if \( f(n) \) is \( \Theta(n^{\log_b a \log^k n}) \), then \( T(n) \) is \( \Theta(n^{\log_b a \log^{k+1} n}) \)
3. if \( f(n) \) is \( \Omega(n^{\log_b a + \varepsilon}) \), then \( T(n) \) is \( \Theta(f(n)) \), provided \( af(n/b) \leq \delta f(n) \) for some \( \delta < 1 \). for all \( n \geq d \)
The Monkey Puzzle Problem (cont.)

arrangements always exist

arrangements never exist
### Reasonable vs. Unreasonable Time

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>300</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>10</td>
<td>50</td>
<td>100</td>
<td>300</td>
<td>1000</td>
</tr>
<tr>
<td>$5N$</td>
<td>50</td>
<td>250</td>
<td>500</td>
<td>1500</td>
<td>5000</td>
</tr>
<tr>
<td>$N \cdot \log_2 N$</td>
<td>33</td>
<td>282</td>
<td>665</td>
<td>2469</td>
<td>9966</td>
</tr>
<tr>
<td>$N^2$</td>
<td>100</td>
<td>2500</td>
<td>10000</td>
<td>90000</td>
<td>7 digits</td>
</tr>
<tr>
<td>$N^3$</td>
<td>1000</td>
<td>125000</td>
<td>7 digits</td>
<td>8 digits</td>
<td>10 digits</td>
</tr>
<tr>
<td>$2^N$</td>
<td>1024</td>
<td>16 digits</td>
<td>31 digits</td>
<td>91 digits</td>
<td>302 digits</td>
</tr>
<tr>
<td>$N!$</td>
<td>7 digits</td>
<td>65 digits</td>
<td>161 digits</td>
<td>623 digits</td>
<td>unimaginably</td>
</tr>
<tr>
<td>$N^N$</td>
<td>11 digits</td>
<td>85 digits</td>
<td>201 digits</td>
<td>744 digits</td>
<td>unimaginably</td>
</tr>
</tbody>
</table>

The number of protons in the universe has 79 digits.

The number of microseconds since the Big Bang has 24 digits.

3.5 Complexity Theory 2007
### Reasonable vs. Unreasonable Time (cont.)

<table>
<thead>
<tr>
<th>$N$</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N^2$</td>
<td>$\frac{1}{10000}$ sec</td>
<td>$\frac{1}{2500}$ sec</td>
<td>$\frac{1}{400}$ sec</td>
<td>$\frac{1}{100}$ sec</td>
<td>$\frac{9}{100}$ sec</td>
</tr>
<tr>
<td>$N^5$</td>
<td>$\frac{1}{10}$ sec</td>
<td>3.2 sec</td>
<td>5.2 min</td>
<td>2.8 hour</td>
<td>28.1 day</td>
</tr>
<tr>
<td>$2^N$</td>
<td>$\frac{1}{1000}$ sec</td>
<td>1 sec</td>
<td>35.7 year</td>
<td>400 trillion centuries</td>
<td>75 digit many centuries</td>
</tr>
<tr>
<td>$N^N$</td>
<td>2.8 hour</td>
<td>3.3 trillion years</td>
<td>70 digit many centuries</td>
<td>185 digit many centuries</td>
<td>728 digit centuries</td>
</tr>
</tbody>
</table>

the Big Bang was about 15 billion year ago

3.6 Complexity Theory 2007
Reasonable vs. Unreasonable Time (cont.)

- the sphere of algorithmic problems (version I)

- unreasonable, exponential algorithms

- reasonable, polynomial algorithms
- reducing Hamiltonian path to travelling salesman
Reducing Oranges to Apples

- three-coloring reduces to satisfiability
  
  countries $C_1, C_2, \ldots, C_N$

- uniqueness of color
  
  \[
  (C_i \text{ is red} \land \neg C_i \text{ is blue} \land \neg C_i \text{ is yellow}) \\
  \lor (\neg C_i \text{ is red} \land C_i \text{ is blue} \land \neg C_i \text{ is yellow}) \\
  \lor (\neg C_i \text{ is red} \land \neg C_i \text{ is blue} \land C_i \text{ is yellow})
  \]

- no conflicting colors
  
  \[
  \neg(C_i \text{ is red} \land C_j \text{ is red}) \\
  \land \neg(C_i \text{ is blue} \land C_j \text{ is blue}) \\
  \land \neg(C_i \text{ is yellow} \land C_j \text{ is yellow})
  \]

  for adjacent countries $C_i$ and $C_j$
Problems That Are Even Harder

- Presburger arithmetic

\[ \forall X \forall Y \exists Z : X + Z = Y \land \exists W : W + W = Y \]

double exponential complexity \( O(2^{2^N}) \)

- WS1S

\[ \forall B : 0 \in B \land (\forall X : X \in B \implies X + 2 \in B) \implies \forall Y : (\exists W : (Y = W + W \implies Y \in B)) \]

non elementary complexity: no \( K \) fold exponential algorithm

3.19 Complexity Theory 2007
Research on Complexity Classes and Intractability 188–190

- some complexity classes with sample problems

fin