Introduction to Algorithms
6.046J/18.401J/SMA5503

Lecture 2
Prof. Erik Demaine
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.
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

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

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.

---

*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 $k < n$:

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

for no choice of $c > 0$. 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
\leq 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 > 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 is good for generating guesses for the substitution method.
- The recursion-tree method can be unreliable, just like any method that uses ellipses (...).
- The recursion-tree method promotes intuition, however.
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 $T(n) = T(n/4) + T(n/2) + n^2$: 

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

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

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

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

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

\[
\begin{array}{c}
\text{(n/4)}^2 \\
\text{(n/16)}^2 \\
\vdots \\
\Theta(1)
\end{array} \quad \begin{array}{c}
\text{(n/2)}^2 \\
\text{(n/8)}^2 \\
\vdots \\
\text{(n/4)}^2
\end{array}
\]

\[\Theta(1)\]
Example of recursion tree

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

\[
\begin{align*}
T(n) &= n^2 - \sum_{i=0}^{\log_4 n} (n/4)^i \\
&= n^2 - \frac{(n/4)^{\log_4 n + 1} - n^2}{n/4 - 1} \\
&= n^2 - \frac{5}{16} n^2 \\
&= \Theta(1)
\end{align*}
\]
Example of recursion tree

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

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

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

\[
\begin{array}{c}
\text{Total} = n^2 \left( 1 + \frac{5}{16} + \left( \frac{5}{16} \right)^2 + \left( \frac{5}{16} \right)^3 + \ldots \right) \\
= \Theta(n^2) \quad \text{geometric series}
\end{array}
\]
The master method

The master method applies to recurrences of the form

\[ T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n), \]

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:** $T(n) = \Theta(n^{\log_b a \lg^{k+1} n})$. 

Day 3

Introduction to Algorithms

L2.19
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:** $T(n) = \Theta(f(n))$. 
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:** $f(n) = \Theta(n^2 \log^k n)$, 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 \( 4(cn/2)^3 \leq cn^3 \) (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 \( n^\varepsilon = \omega(\lg n). \)
General method (Akra-Bazzi)

\[ T(n) = \sum_{i=1}^{k} a_i T(n/b_i) + f(n) \]

Let \( p \) be the unique solution to

\[ \sum_{i=1}^{k} \left( \frac{a_i}{b_i^p} \right) = 1. \]

Then, the answers are the same as for the master method, but with \( n^p \) instead of \( n^{\log_b a} \).

(Akra and Bazzi also prove an even more general result.)
Idea of master theorem

Recursion tree:

\[ f(n) \]
\[ f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad a f(n/b) \]
\[ f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad a^2 f(n/b^2) \]
\[ \vdots \]
\[ T(1) \]

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

\( h = \log_b n \)
Idea of master theorem

**Recursion tree:**

$$f(n) \quad a \quad f(n)$$

$$f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad af(n/b)$$

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

$$\vdots$$

$$T(1) \quad n^{\log_b a} T(1) \quad \Theta(n^{\log_b a})$$

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:

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

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

\[ f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad a^3 f(n/b^3) \]

\[ \vdots \]

\[ T(1) \]

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

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

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

Recursion tree:

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

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

- Next time: applying the master method.
- For proof of master theorem, see CLRS.
Appendix: geometric series

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

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