CS473 - Algorithms I

Lecture 3

Solving Recurrences
Solving Recurrences

• The analysis of merge sort Lecture 1 required us to solve a recurrence.

• Recurrences are like solving integrals, differential equations, etc.
  ➢ Learn a few tricks.

• Lecture 4 : Applications of recurrences.
Recurrences

• Function expressed recursively

\[ T(n) = \begin{cases} 
1 & \text{if } n = 1 \\
T\left(\lceil n / 2 \rceil\right) + 1 & \text{if } n > 1 
\end{cases} \]

• Solve for \( n = 2^k \)
Recurrences

• Claimed answer: $T(n) = \lg n + 1 = \Theta (\lg n)$
  - Substitute claimed answer for $T$ in the recurrence
  - Note: resulting equations are true when $n = 2^k$

i.e.  \[
\lg n + 1 = \begin{cases} 
1 & \text{if } n=2^0=1 \\
(\lg( \lceil n / 2 \rceil) + 1) & \text{if } n=2^k > 1
\end{cases}
\]

Tedious technicality: haven’t shown $T(n) = \Theta (\lg n)$

– But, since $T(n)$ is monotonically non-decreasing function of $n$

\[n \leq 2^{\lg n} \Rightarrow T(n) \leq T(2^{\lceil \lg n \rceil})\]

\[= \lg(2^{\lceil \lg n \rceil}) + 1 = \lceil \lg n \rceil + 1 = \Theta (\lg n)\]

– Thus, ceiling didn’t matter much
Recurrences

• Technically, should be careful about floors and ceilings (as in the book)

• But, usually it is okay
  – To ignore floor/ceiling
  – Just solve for exact powers of 2 (or b)
Boundary Conditions

• Usually assume $T(n) = \Theta(1)$ for small $n$
  – Does not usually affect soln. (if polynomially bounded)

• Example: Initial condition affects soln.
  – Exponential $T(n) = (T(n/2))^2$
    If $T(1) = c$ for a constant $c > 0$, then
    $T(2) = (T(1))^2 = c^2$, $T(4) = (T(2))^2 = c^4$,
    $T(n) = \Theta(c^n)$

E.g.,
\[
\begin{align*}
T(1) = 2 &\Rightarrow T(n) = \Theta(2^n) \\
T(1) = 3 &\Rightarrow T(n) = \Theta(3^n)
\end{align*}
\]

However, $2^n \neq \Theta(3^n)$

Difference in soln. is more dramatic with
\[
T(1) = 1 \Rightarrow T(n) = \Theta(1^n) = \Theta(1)
\]
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

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

whenever \( \left( \frac{c}{2} \right) n^3 - n \geq 0 \), 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 \( k < n \):

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

for no choice of \( c > 0 \). Lose!
A Tighter Upper Bound!

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

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

$$
T(n) = 4T(n/2) + n \\
\leq 4 \left( c_1 \left( \frac{n}{2} \right)^2 - c_2 \left( \frac{n}{2} \right) \right) + n \\
= c_1 n^2 - 2c_2 n + n \\
= c_1 n^2 - c_2 n - (c_2 n - n) \\
\leq c_1 n^2 - c_2 n \text{ if } c_2 > 1
$$

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.

• 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 $T(n) = T(n/4) + T(n/2) + n^2$:

$T(n)$
Example of Recursion Tree

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

$$
\begin{array}{c}
n^2 \\
T(n/4) \quad T(n/2)
\end{array}
$$
Example of Recursion Tree

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

\[
\begin{array}{ccc}
& n^2 & \\
(n/4)^2 & (n/2)^2 & \\
T(n/16) & T(n/8) & T(n/8) \\
& T(n/4) & \\
\end{array}
\]
Example of Recursion Tree

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

$\Theta(1)$
Example of Recursion Tree

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

$\Theta(1)$
Example of Recursion Tree

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

$\Theta(1)$

$\frac{5}{16} n^2$

$\frac{n^2}{2}$

$\frac{(n/4)^2}{2}$

$\frac{(n/8)^2}{2}$

$\frac{(n/8)^2}{2}$

$\frac{(n/4)^2}{2}$

$\frac{(n/2)^2}{2}$
Example of Recursion Tree

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

\[ \Theta(1) \]

\[ (n/16)^2 \quad (n/8)^2 \quad (n/8)^2 \quad (n/4)^2 \quad 25/256 n^2 \]

\[ (n/4)^2 \quad (n/2)^2 \quad 5/16 n^2 \]

\[ n^2 \]

\[ n^2 \]
Example of Recursion Tree

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

\[
T(n) = n^2 + \frac{n^2}{16} + \frac{n^2}{16} + \frac{n^2}{16} + \frac{n^2}{16} + \ldots
\]

Total = \( n^2 (1 + \frac{5}{16} + \frac{25}{256} + \ldots) \)

= \( \Theta(n^2) \) geometric series
The Master Method

- The master method applies to recurrences of the form

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

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

• Compare $f(n)$ with $n^{\log_b a}$:

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

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

3. \( f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \) 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

\[ a f(n/b) \leq c f(n) \]

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

  \( \Theta(n^2) \)

- 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, \( k=0 \)

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

\( T(n) = \Theta (n^3) \)

• Ex: \( T(n) = 4T(n/2) + n^2 / \log n \)
  \[ a=4, b=2 \Rightarrow n^{\log_b a} = n^2 ; f(n) = n^2 / \log n \]

Master method does not apply. In particular, for every constant \( \varepsilon > 0 \), we have \( n^{\varepsilon} = \omega (\log n) \)
General Method (Akra-Bazzi)

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

Let \( p \) be the unique solution to

\[ \sum_{i=1}^{k} \left( \frac{a_i}{b^p_i} \right) = 1 \]

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:

\[ f(n) \]

\[ f(n/b) \]

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

\[ \text{#leaves} = a^h \]

\[ T(1) = a \log_b n \]

\[ n \log_b a T(1) \]
Idea of Master Theorem

Recursion tree:

\[ \begin{array}{cccccccc}
& f(n) & \cdots & f(n) \\
\text{CASE 1:} & f(n/b) & f(n/b) & \cdots & f(n/b) & a f(n/b) \\
& f(n/b^2) & f(n/b^2) & \cdots & f(n/b^2) & a^2 f(n/b^2) \\
& \vdots & & & & & & \\
\end{array} \]

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

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

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:

\[ f(n) \quad \underbrace{f(n/b) \quad f(n/b) \quad \ldots \quad f(n/b)}_{a} \quad a f(n/b) \]

\[ f(n/b^2) \quad f(n/b^2) \quad \ldots \quad f(n/b^2) \quad \underbrace{a^2 f(n/b^2)}_{a^2 f(n/b^2)} \]

\[ h = \log_b n \]

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

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

Recursion tree:

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

\[ T(n) = \begin{cases} 
T(1) & \text{if } n = 1 \\
af(n/b) + \sum_{i=0}^{\log_b n} a^i f(n/b^i) & \text{if } n > 1
\end{cases} \]
Proof of Master Theorem: Case 1 and Case 2

• Recall from the recursion tree (note \( h = \log_b n \) = tree height)

\[
T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{h-1} a^i f\left(\frac{n}{b^i}\right)
\]

- Leaf cost
- Non-leaf cost = \( g(n) \)
Proof of Case 1

\[ n^{\frac{\log_b a}{f(n)}} = \Omega(n^{\varepsilon}) \quad \text{for some } \varepsilon > 0 \]

\[ n^{\frac{\log_b a}{f(n)}} = \Omega(n^{\varepsilon}) \Rightarrow \frac{f(n)}{n^{\log_b a}} = O(n^{-\varepsilon}) \Rightarrow f(n) = O(n^{\log_b a - \varepsilon}) \]

\[ g(n) = \sum_{i=0}^{h-1} a^i O\left((n/b^i)^{\log_b a - \varepsilon}\right) = O\left(\sum_{i=0}^{h-1} a^i (n/b^i)^{\log_b a - \varepsilon}\right) \]

\[ = O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{h-1} a^i b^{i\varepsilon} / b^{i\log_b a}\right) \]
Case 1 (cont’)

\[
\sum_{i=0}^{h-1} \frac{a^i b^{i\varepsilon}}{b^{i\log_b a}} = \sum_{i=0}^{h-1} a^i \frac{(b^\varepsilon)^i}{(b^{\log_b a})^i} = \sum_{i=0}^{h-1} a^i \frac{b^{\varepsilon i}}{a^i} = \sum_{i=0}^{h-1} (b^\varepsilon)^i
\]

= An increasing geometric series since $b > 1$

\[
\frac{b^{\varepsilon h} - 1}{b^\varepsilon - 1} = \frac{(b^h)^\varepsilon - 1}{b^\varepsilon - 1} = \frac{(b^{\log_b n})^\varepsilon - 1}{b^\varepsilon - 1} = \frac{n^\varepsilon - 1}{b^\varepsilon - 1} = O(n^\varepsilon)
\]
Case 1 (cont’)

\[- g(n) = O\left(n^{\log_b a - \varepsilon} O\left(n^\varepsilon\right)\right) = O\left(\frac{n^{\log_b a}}{n^\varepsilon} O\left(n^\varepsilon\right)\right) = O\left(n^{\log_b a}\right)\]

\[- T(n) = \Theta\left(n^{\log_b a}\right) + g(n) = \Theta\left(n^{\log_b a}\right) + O\left(n^{\log_b a}\right) = \Theta\left(n^{\log_b a}\right)\]

Q.E.D.
Proof of Case 2 (limited to $k=0$)

$$\frac{f(n)}{n^\log_b a} = \Theta(\lg^0 n) = \Theta(1) \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow f(n / b^i) = \Theta \left( \frac{n}{b^i} \right)^{\log_b a}$$

$$\therefore g(n) = \sum_{i=0}^{h-1} a^i \Theta \left( (n / b^i)^{\log_b a} \right)$$

$$= \Theta \left( \sum_{i=0}^{h-1} a^i \frac{n^{\log_b a}}{b^{i \log_b a}} \right) = \Theta \left( n^{\log_b a} \frac{1}{(b^{\log_b a})^i} \right) = \Theta \left( n^{\log_b a} \frac{1}{a^i} \right)$$

$$= \Theta \left( n^{\log_b a} \log_b n \right) = \Theta(n^{\log_b a} \log n)$$

$$T(n) = n^{\log_b a} + \Theta(n^{\log_b a} \log n)$$

$$= \Theta \left( n^{\log_b a} \log n \right) \quad \text{Q.E.D.}$$
Conclusion

• Next time: applying the master method.