Design and Analysis of Algorithms

CSE 5311
Lecture 3 Divide-and-Conquer

Junzhou Huang, Ph.D.
Department of Computer Science and Engineering
Reviewing: $\Theta$-notation

**Definition:**

$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \}$

**Basic Manipulations:**

- Drop low-order terms; ignore leading constants.
- Example: $3n^3 + 90n^2 - 5n + 6046 = \Theta(n^3)$
Reviewing: Insertion Sort Analysis

**Worst case:** Input reverse sorted.

\[
T(n) = \sum_{j=2}^{n} O(j) = \Theta(n^2) \quad \text{[arithmetic series]}
\]

**Average case:** All permutations equally likely.

\[
T(n) = \sum_{j=2}^{n} O(j/2) = \Theta(n^2)
\]

*Is insertion sort a fast sorting algorithm?*

- Moderately so, for small \( n \).
- Not at all, for large \( n \).
Reviewing: Recurrence for Merge Sort

\[ T(n) = \begin{cases} 
\Theta(1) & \text{if } n = 1; \\
2T(n/2) + \Theta(n) & \text{if } n > 1.
\end{cases} \]

- We shall usually omit stating the base case when \( T(n) = \Theta(1) \) for sufficiently small \( n \), but only when it has no effect on the asymptotic solution to the recurrence.
- Next Lecture will provide several ways to find a good upper bound on \( T(n) \).
Reviewing: Recursion Tree

Solve \( T(n) = 2T(n/2) + cn \), where \( c > 0 \) is constant.

\[ h = \lg n \]

\[ \Theta(1) \quad \text{#leaves} = n \quad \Theta(n) \]

Total = \( \Theta(n \lg n) \)
Solving Recurrences

- **Recurrence**
  - The analysis of integer multiplication from last lecture required us to solve a recurrence
  - Recurrences are a major tool for analysis of algorithms
  - Divide and Conquer algorithms which are analyzable by recurrences.

- **Three steps at each level of the recursion:**
  - **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
  - **Conquer** the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
  - **Combine** the solutions to the subproblems into the solution for the original problem.
Recall: Integer Multiplication

- Let $X = \begin{bmatrix} A \\ B \end{bmatrix}$ and $Y = \begin{bmatrix} C \\ D \end{bmatrix}$ where $A, B, C$ and $D$ are $n/2$ bit integers
- **Simple Method:** $XY = (2^{n/2}A + B)(2^{n/2}C + D)$
- **Running Time Recurrence**
  $$T(n) < 4T(n/2) + \Theta(n)$$

How do we solve it?
Substitution Method

*The most general method:*

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

**Example:** \( T(n) = 4T(n/2) + \Theta(n) \)

- [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

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

We can imagine \( \Theta(n) = 100n \). Then, whenever \((c/2)n^3 - 100n \geq 0\), for example, if \( c \geq 200 \) and \( n \geq 1 \).
Example

- 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.

\[ \text{This bound is not tight!} \]
A Tighter Upper Bound?

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

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

$$T(n) = 4T(n/2) + 100n$$
$$\leq cn^2 + 100n$$
$$\leq cn^2$$

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 \( k < n \).

\[
T(n) = 4T(n/2) + 100n \\
\leq 4(c_1 (n/2)^2 - c_2 (n/2)) + 100n \\
= c_1 n^2 - 2c_2 n + 100n \\
= c_1 n^2 - c_2 n - (c_2 n - 100n) \\
\leq c_1 n^2 - c_2 n \quad \text{if} \quad c_2 > 100.
\]

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 (…).
- However, the recursion-tree method promotes intuition
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 $T(n) = T(n/4) + T(n/2) + n^2$:
Example of Recursion Tree

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

$$n^2$$

$$\frac{(n/4)^2}{T(n/16)} \quad \frac{(n/2)^2}{T(n/4)}$$

$$\frac{1}{T(n/8)} \quad \frac{1}{T(n/8)}$$
Example of Recursion Tree

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

\[
\begin{align*}
& n^2 \\
& (n/4)^2 & (n/2)^2 \\
& (n/16)^2 & (n/8)^2 & (n/8)^2 & (n/4)^2 \\
& \vdots & \vdots & \vdots & \vdots \\
& \Theta(1)
\end{align*}
\]
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 (n/8)^2 (n/4)^2
 / / /
Θ(1)
```

$\Theta(1)$
Example of Recursion Tree

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

```
        n^2
       /   \
     /     /
   n^2   5/16 n^2
   /     /
(n/4)^2  (n/2)^2
   /     /
(n/16)^2  (n/8)^2
   /     /   /
Θ(1)  (n/8)^2  (n/4)^2
```
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  (n/16)^2
  /     \\    \     \
θ(1)    (n/8)^2  (n/4)^2
  /     \\    \     \
5/16  25/256 n^2
```

\[ \Theta(1) \]
Example of Recursion Tree

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

$$
\begin{align*}
    & n^2 \\
\quad \quad & \frac{5}{16}n^2 \\
\quad \quad \quad & \frac{25}{256}n^2 \\
\quad \quad \quad \quad & \vdots \\
\quad \quad \quad \quad & \Theta(1)
\end{align*}
$$

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
Appendix: Geometric Series

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

\[ 1 + x + x^2 + \cdots = \frac{1}{1 - x} \quad \text{for } |x| < 1 \]
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.
Idea of Master Theorem

**Recursion tree:**

\[ h = \log_b n \]

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

#leaves = \( a^h \)

= \( a^{\log_b n} \)

= \( n^{\log_b a} \)
Case (I)

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:** \( T(n) = \Theta(n^{\log_b a}) \).
Idea of Master Theorem

Recursion tree:

\[ f(n) \xrightarrow{a} f(n) \]
\[ f(n/b) \xrightarrow{a} f(n/b) \]
\[ f(n/b^2) \xrightarrow{a} f(n/b^2) \]

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

\[ f(n) = n^{\log_b a - \varepsilon} \quad \text{and} \quad a f(n/b) = a (n/b)^{\log_b a - \varepsilon} = b^\varepsilon n^{\log_b a - \varepsilon} \]
Case (II)

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

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

Solution: \( T(n) = \Theta(n^{\log_b a} \log n) \).
Idea of Master Theorem

**Recursion tree:**

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

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

\[ \vdots \]

\[ T(1) \]

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

\[ f(n) = n^{\log_b a} \quad \text{and} \quad af(n/b) = a \left(\frac{n}{b}\right)^{\log_b a} = n^{\log_b a} \]

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

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 $a f(n/b) \leq c f(n)$ for some constant $c < 1$.

   **Solution:** $T(n) = \Theta(f(n))$. 
Idea of master theorem

Recursion tree:

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

\[ h = \log_b n \]

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

\[ T(1) \]

\[ \Theta(f(n)) \]

\[ f(n) = n^{\log_b a + \varepsilon} \quad \text{and} \quad a \quad f(n/b) = a \quad (n/b)^{\log_b a + \varepsilon} = b^{-\varepsilon} \quad n^{\log_b a + \varepsilon} \]
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. \( T(n) = 4T(n/2) + n^3 \)
\[ a = 4, \quad b = 2 \implies n^{\log_b a} = n^2; \quad f(n) = n^3. \]

CASE 3: \( f(n) = \Omega(n^2 + \varepsilon) \) for \( \varepsilon = 1 \)
and \( 4(n/2)^3 \leq cn^3 \) (reg. cond.) for \( c = \frac{1}{2} < 1 \)
\[ \therefore T(n) = \Theta(n^3). \]

Ex. \( T(n) = 4T(n/2) + n^2/\lg n \)
\[ a = 4, \quad b = 2 \implies n^{\log_b a} = n^2; \quad f(n) = n^2/\lg n. \]
Master method does not apply. In particular, for every constant \( \varepsilon > 0 \), we have \( n^\varepsilon = \omega(\lg n) \).