CS 2022 : DATA STRUCTURES & ALGORITHMS

Lecture 4: Analyzing Recursion

Malaka Walpola

Outline

- Analyzing Merge Sort
- Solving Recurrences
  - ***Slides taken from [3]***

Learning Outcomes

- After successfully studying contents covered in this lecture, students should be able to,
  - Analyze the time complexity of merge sort
  - Analyze the complexity of algorithms with recursion
  - Use substitution method, recursion-tree method and the master method to solve recurrences

Analysis of Merge Sort

Code	Cost	Times
MERGE-SORT(A, p, r)	$T(n)$	1
1. IF p < r	c_1	1
2. q ← ⌊(p + r)/2⌋	c_2	1
3. MERGE-SORT(A, p, q)	$T(n/2)$	1
4. MERGE-SORT(A, q + 1, r)	$T(n/2)$	1
5. MERGE(A, p, q, r)	$F(n)$	1

$F(n) \in \Theta(n)$

$$T(n) = c_1 + c_2 + 2T\left(\frac{n}{2}\right) + F(n) = 2T\left(\frac{n}{2}\right) + F'(n)$$

Analysis of Merge Sort

- $$T(n) = \begin{cases} 1 & if \quad n = 1 \\ 2T\left(\dfrac{n}{2}\right) + F'(n) & if \quad n > 1 \end{cases}$$

- Solve Using Recursion Tree

Solving Recurrences

- Like Solving Integrals, Differential Equations, etc.
  - Learn a few tricks
- Methods for Solving Recurrences
  - Substitution method
  - Iteration method (Recursion-tree method)
  - Master method

Substitution Method

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

Substitution Method

- Example
  - $T(n) = 4T(n/2) + n$
  - Boudry condition $T(1) = \Theta(1)$
- Solution
  - Guess $\theta(n^3)$ (Prove O and Ω separately)
  - Assume that $T(k) \le ck^3$ for $k < n$
  - Prove $T(n) \le cn^3$ by induction

Substitution Method

- Solution cont.

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\leqq 4c(n/2)^3 + n \\
&= (c/2)n^3 + n \\
&= cn^3 - ((c/2)n^3 - n) \leftarrow \text{desired} - \text{residual} \\
&\leqq cn^3 \leftarrow \text{desired}
\end{aligned}
$$

whenever $(c/2)n^3 - n \geq 0$, for example,

if $c \geq 2$ and $n \geq 1$.

Substitution Method

- Solution cont.
  - Initial conditions (ground the induction with base cases)
  - ***Base:*** $T(n) = \Theta(1)$ for all $n < n_0$, where n_0 is a suitable constant
  - For $1 \le n < n_0$, we have "$\Theta(1)$" $\le cn^3$, if we pick c big enough.
- ***This bound is not tight!***

Substitution Method

- A Tighter Upper Bound: $T(n) = O(n^2)$
  - Assume that $T(k) \le ck^2$ for $k < n$

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4c(n/2)^2 + n \\
&= O(n^2) \textbf{\textit{ - Wrong}} \\
&= cn^2 - (- n) \text{ [desired –residual]} \\
&\le cn^2 \text{ for no choice of } c > 0 \textbf{\textit{ - Wrong}}
\end{aligned}
$$

Substitution Method

- A Tighter Upper Bound Cont.
  - Strengthen the inductive hypothesis
    - Subtract a low-order term
  - Strengthened inductive hypothesis
    - $T(k) \leq c_1k^2 - c_2k$ for $k < n$

Substitution Method

- A Tighter Upper Bound Cont.
  - Inductive hypothesis: $T(k) \le c_1k^2 - c_2k$ for $k < n$

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4(c_1(n/2)^2 - c_2(n/2) + n \\
&= c_1n^2 - 2\,c_2n - (c_2n - n) \\
&\le c_1n^2 - c_2n \quad \text{if} \quad c_2 < 1
\end{aligned}
$$

  - Pick c_1 big enough to handle the initial conditions

Recursion-Tree Method

- Models the Costs (Time) of a Recursive Execution
- Good For
  - Generating guesses for the substitution method
- Can Be Unreliable
- Examples
  - $T(n) = T(n/4) + T(n/2) + n^2$

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 $n^{\log_b a}$
  - Once the case is identified, the solution is readily available

Master Method – Case I

- $f(n) = O(n^{\log_b a-\varepsilon})$ for some constant ε > 0
- Idea
  - $f(n)$ grows polynomially slower than $n^{\log_b a}$ by an n^{ε} factor
- Solution
  - $T(n) = \Theta(n^{\log_b a})$

Master Method – Case II

- $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$
- Idea
  - $f(n)$ and $n^{\log_b a}$ grow at similar rates
- Solution
  - $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$

Master Method – Case III

- $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant ε > 0
- Idea
  - $f(n)$ grows polynomially faster than $n^{\log_b a}$ by an n^{ε} factor **and** $f(n)$ satisfies the ***regularity condition*** that $a\,f(n/b) \le c\,f(n)$ for some constant $c < 1$
- Solution
  - $T(n) = \Theta(f(n))$

Master Method Examples

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

  $a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2$

  $f(n) = n$

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

  $\therefore T(n) = \Theta(n^2)$

Master Method Examples

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

  $a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2$

  $f(n) = n^2$

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

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

Master Method Examples

- $T(n) = 4T(n/2) + n^3$

  $a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2$

  $f(n) = n^3$

  CASE 3: $f(n) = \Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$ **and**

  $4(n/2)^3 \le cn^3$ (reg. cond.) for $c = 1/2$

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

Master Method Examples

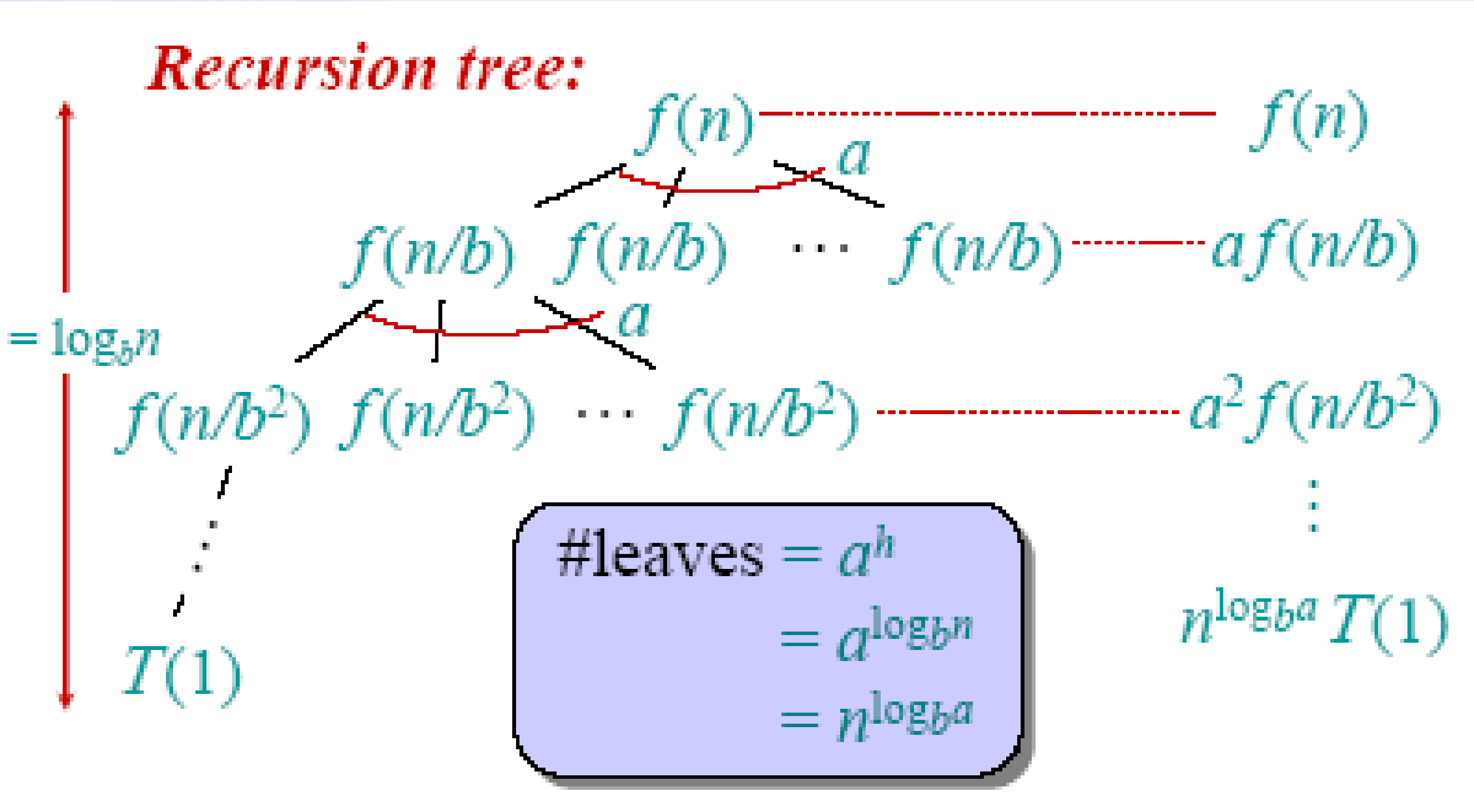
- $T(n) = 4T(n/2) + n^2/\lg n$

  $a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2$

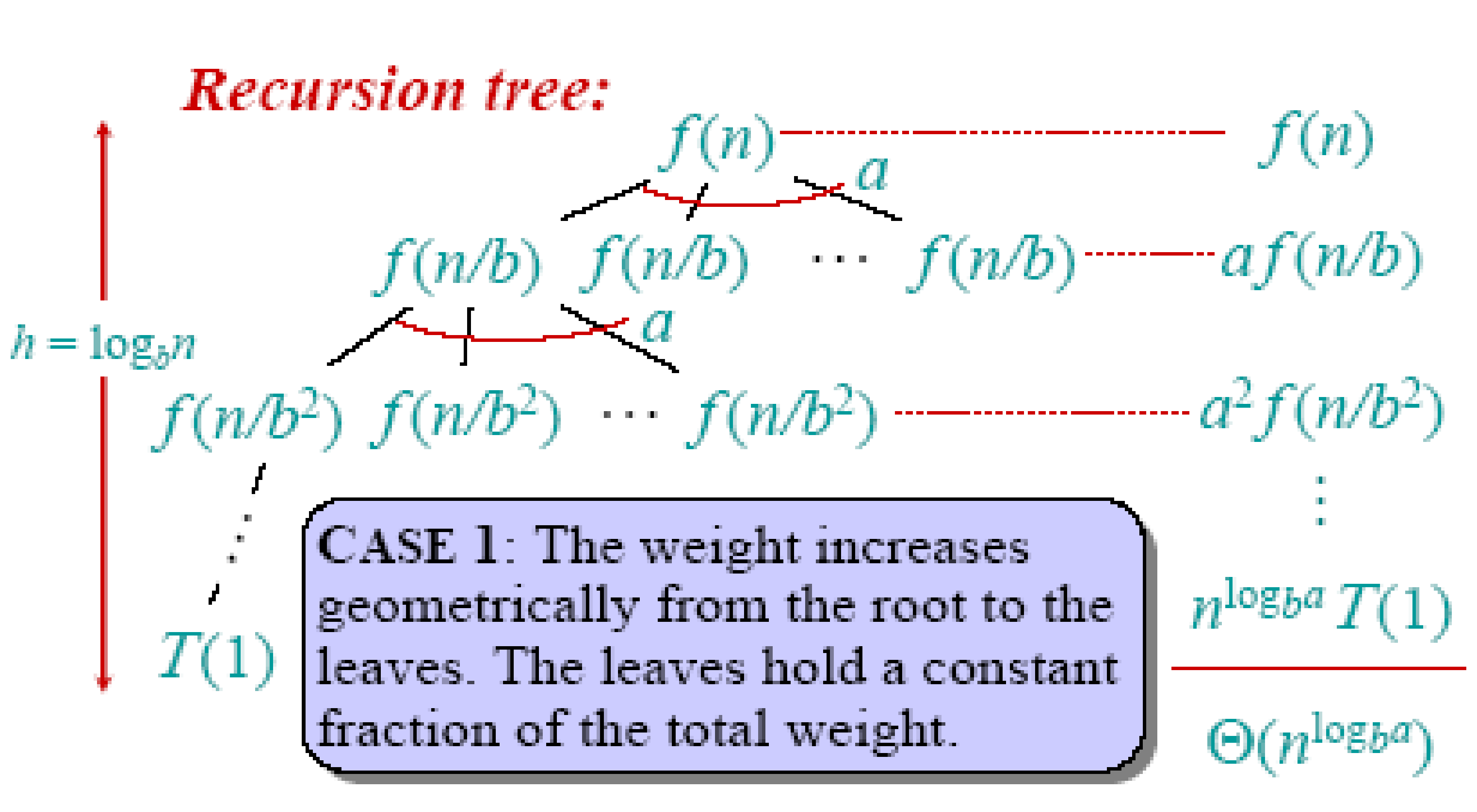
  $f(n) = n^2/\lg n$

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

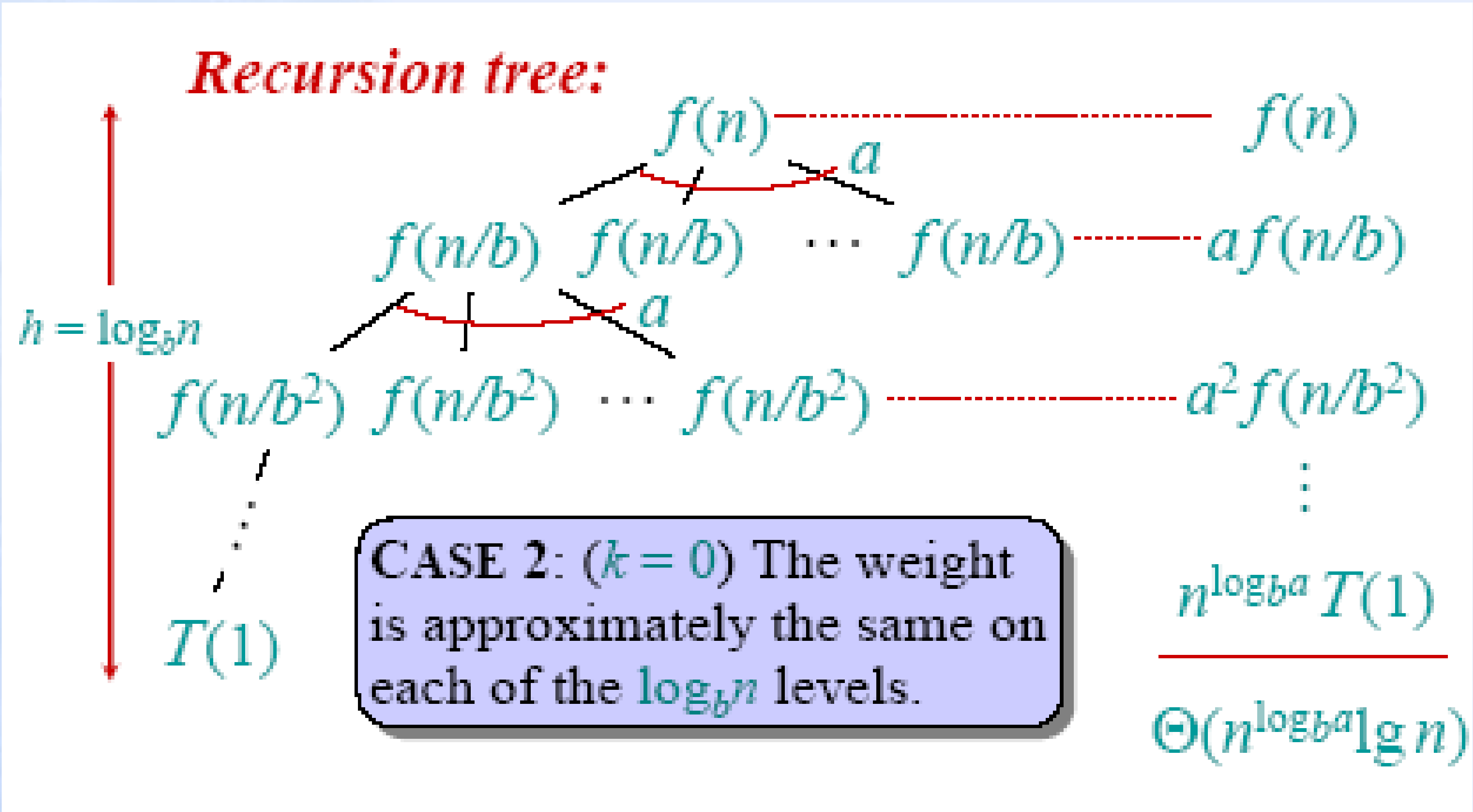
Idea of Master Theorem



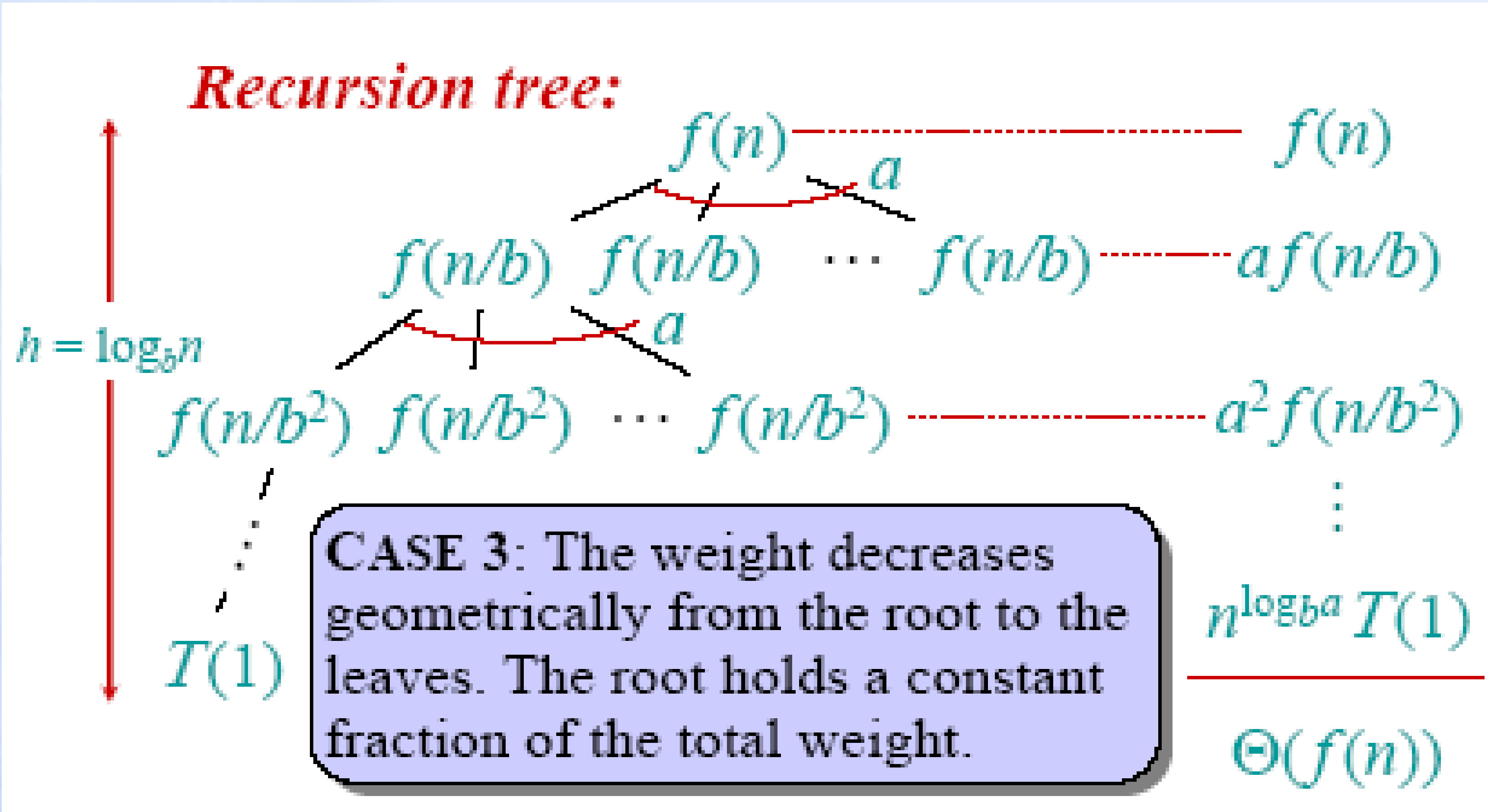
Idea of Master Theorem



Idea of Master Theorem



Idea of Master Theorem



Divide & Conquer Algorithms and Recurrences

- Time Complexity - $T(n)$
- Strategy for Divide & Conquer Approach
  - **Divide** – Time complexity $f_1(n)$
  - **Conquer:** Solve **a subset** of sub-problem recursively – Time complexity $a\ T(n/b)$
    - a – Number of sub problems solved
    - b – Factor sub problems are smaller
  - **Combine** – Time complexity $f_2(n)$
- $T(n) = a\ T(n/b) + f_1(n) + f_2(n)$
- $T(n) = a\ T(n/b) + f(n)$

SELF STUDYING

Self Studying

- Reading Assignment – IA
  - Analysis of Merge Sort: Section 2.3.2
  - Divide & Conquer & Recurrences: Chapter 4 pages 65 – 74
  - Solving Recurrences: Sections 4.3, 4.4, 4.5
- Solve the Recursion of Merge Sort Using Different Methods

Appendix: Geometric Series

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

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

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, 3rd Ed. Cambridge, MA, MIT Press, 2009.

[2] S. Baase and Allen Van Gelder, *Computer Algorithms: Introduction to Design and Analysis*, 3rd Ed. Delhi, India, Pearson Education, 2000.

[3] Lecture slides from Prof. Erik Demaine of MIT, available at http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-2-asymptotic-notation-recurrences-substitution-master-method/lec2.pdf