

# **CSE 548: Analysis of Algorithms**

## **Lecture 5**

### **( Divide-and-Conquer Algorithms: The Master Theorem )**

**Rezaul A. Chowdhury**

**Department of Computer Science**

**SUNY Stony Brook**

**Fall 2016**

# A Useful Recurrence

Consider the following recurrence:

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise;} \end{cases}$$

where,  $a \geq 1$  and  $b > 1$ .

Arises frequently in the analyses of *divide-and-conquer* algorithms.

Consider the following recurrences from previous lectures.

**Karatsuba's Algorithm:**  $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n)$

**Strassen's Algorithm:**  $T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$

**Fast Fourier Transform:**  $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

# How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$

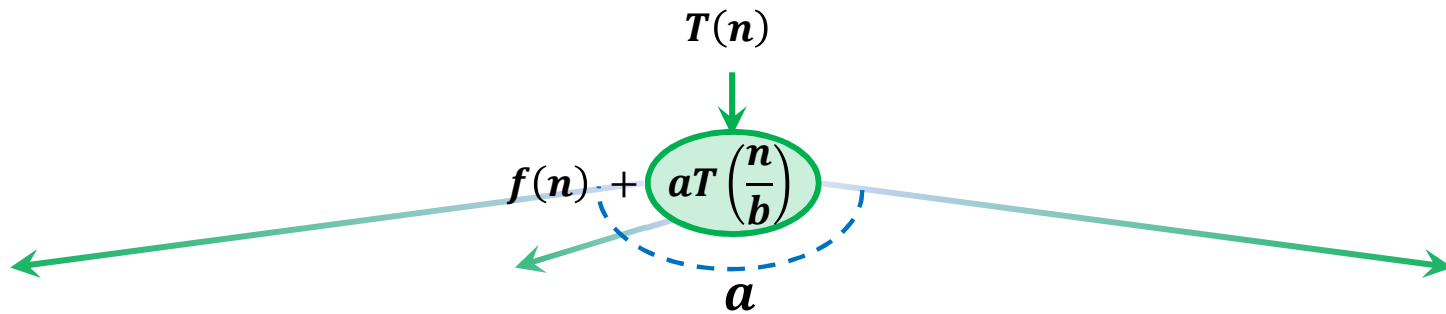
# How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$

$$\begin{array}{c} T(n) \\ \downarrow \\ f(n) + aT\left(\frac{n}{b}\right) \end{array}$$

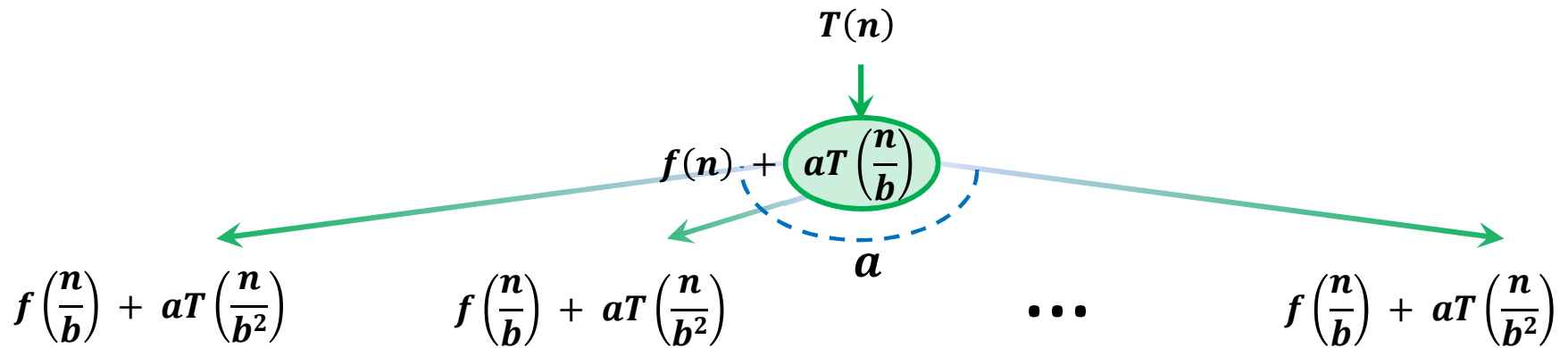
# How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



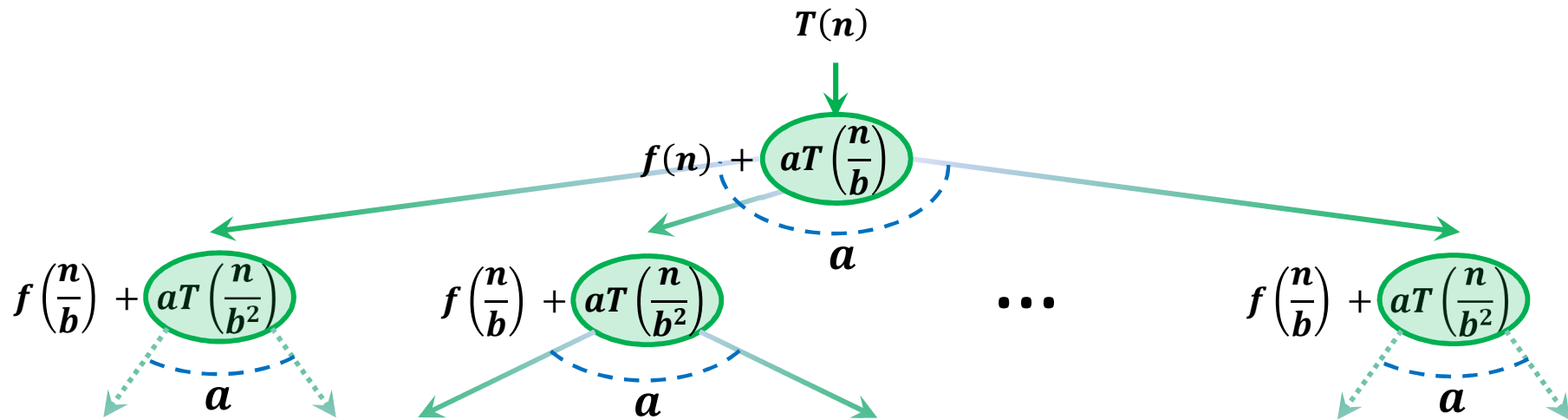
# How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



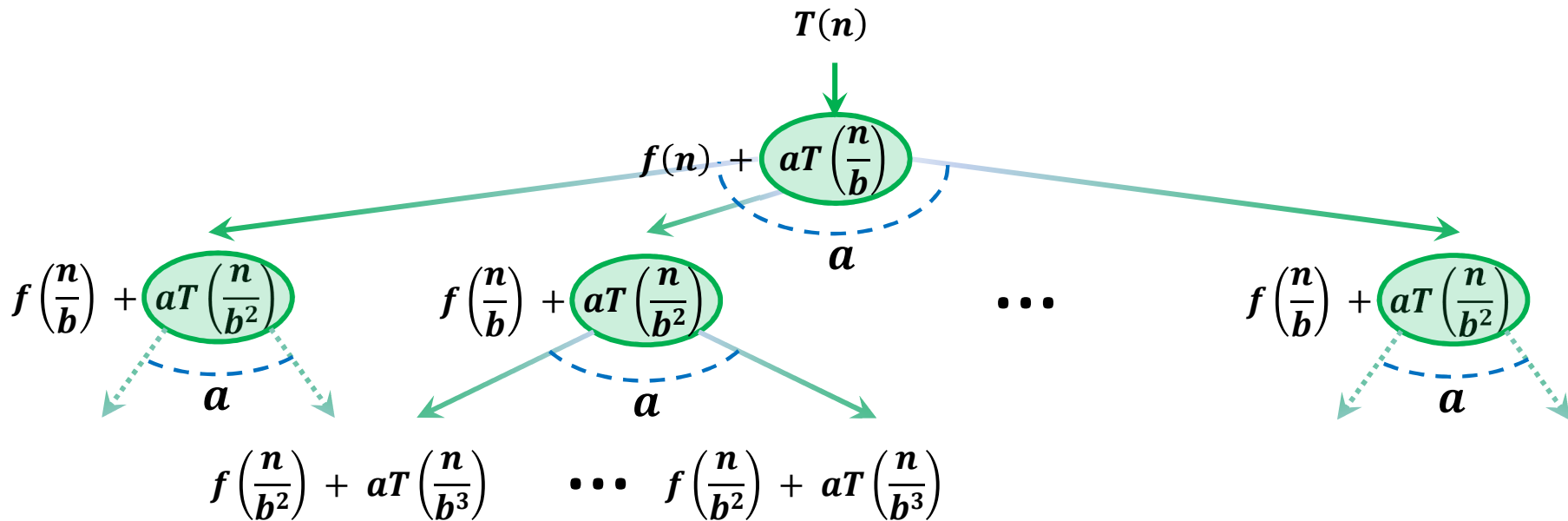
# How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



# How the Recurrence Unfolds

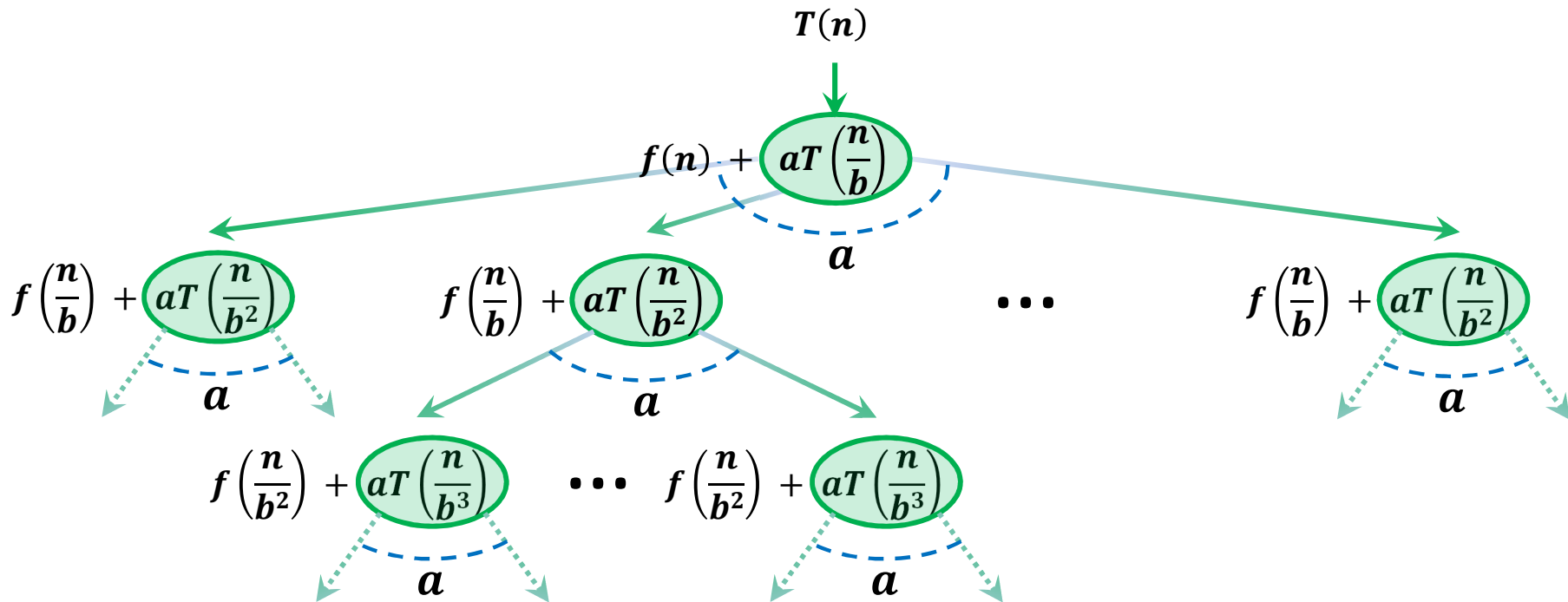
$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$





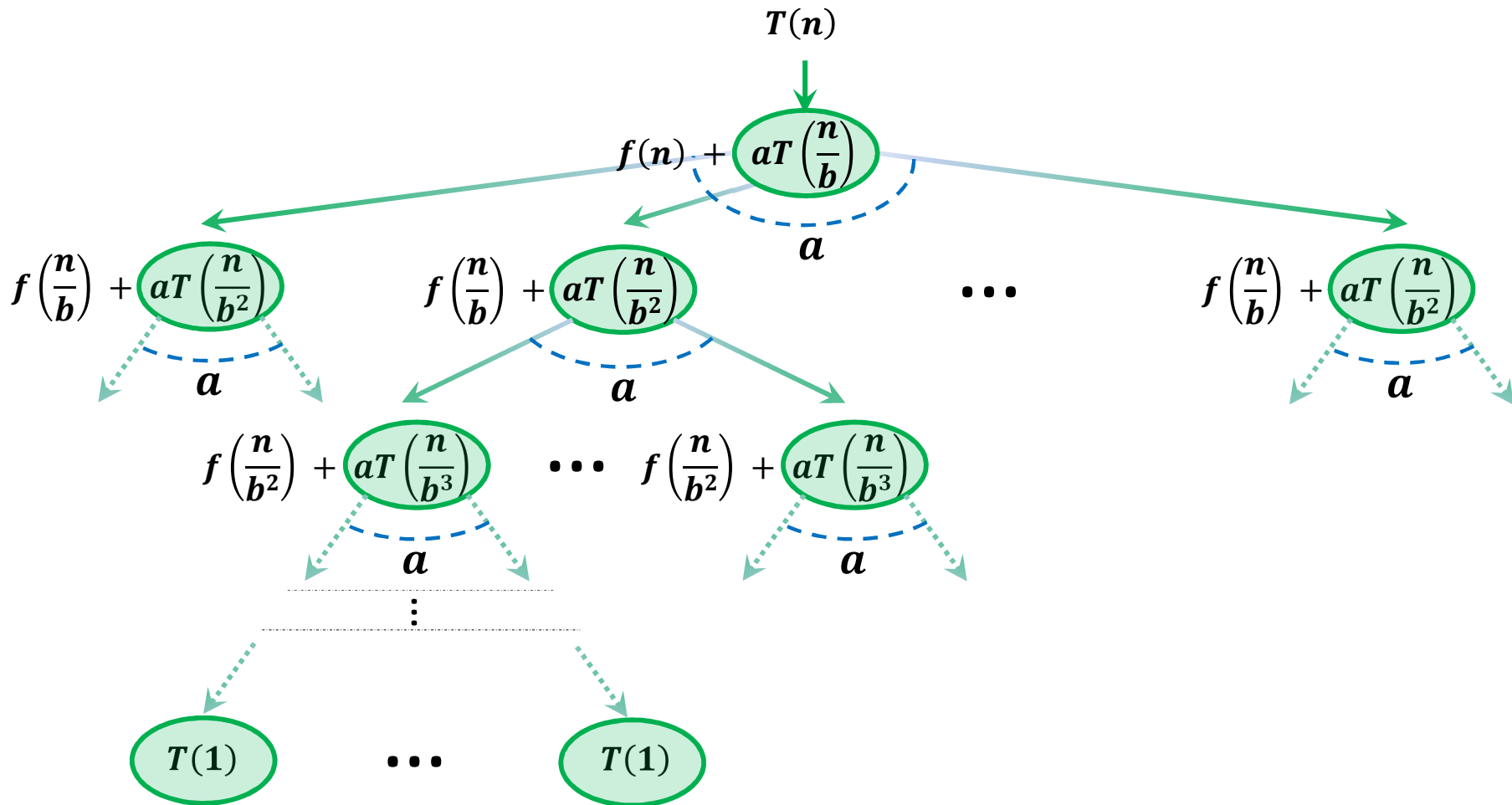
# How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



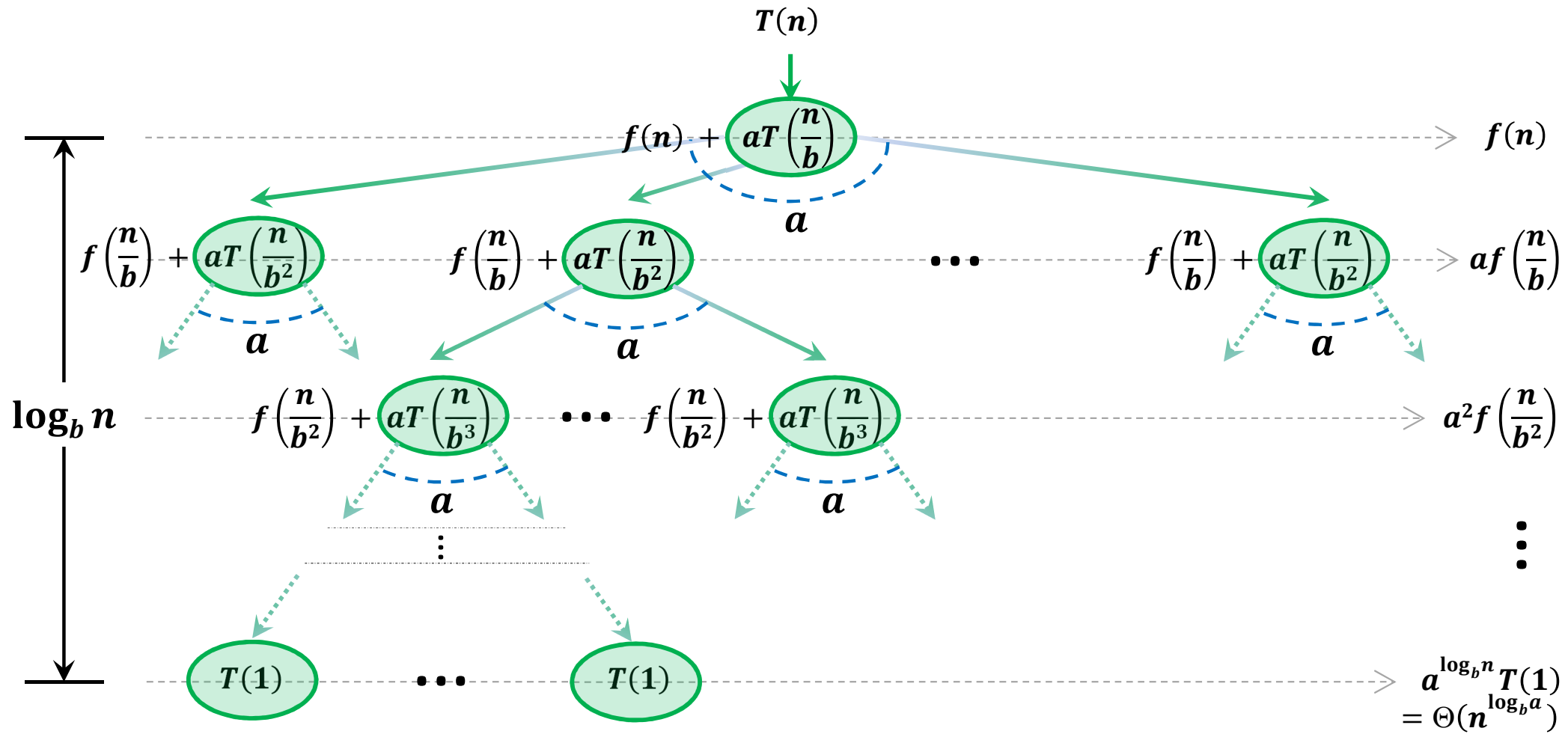
# How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



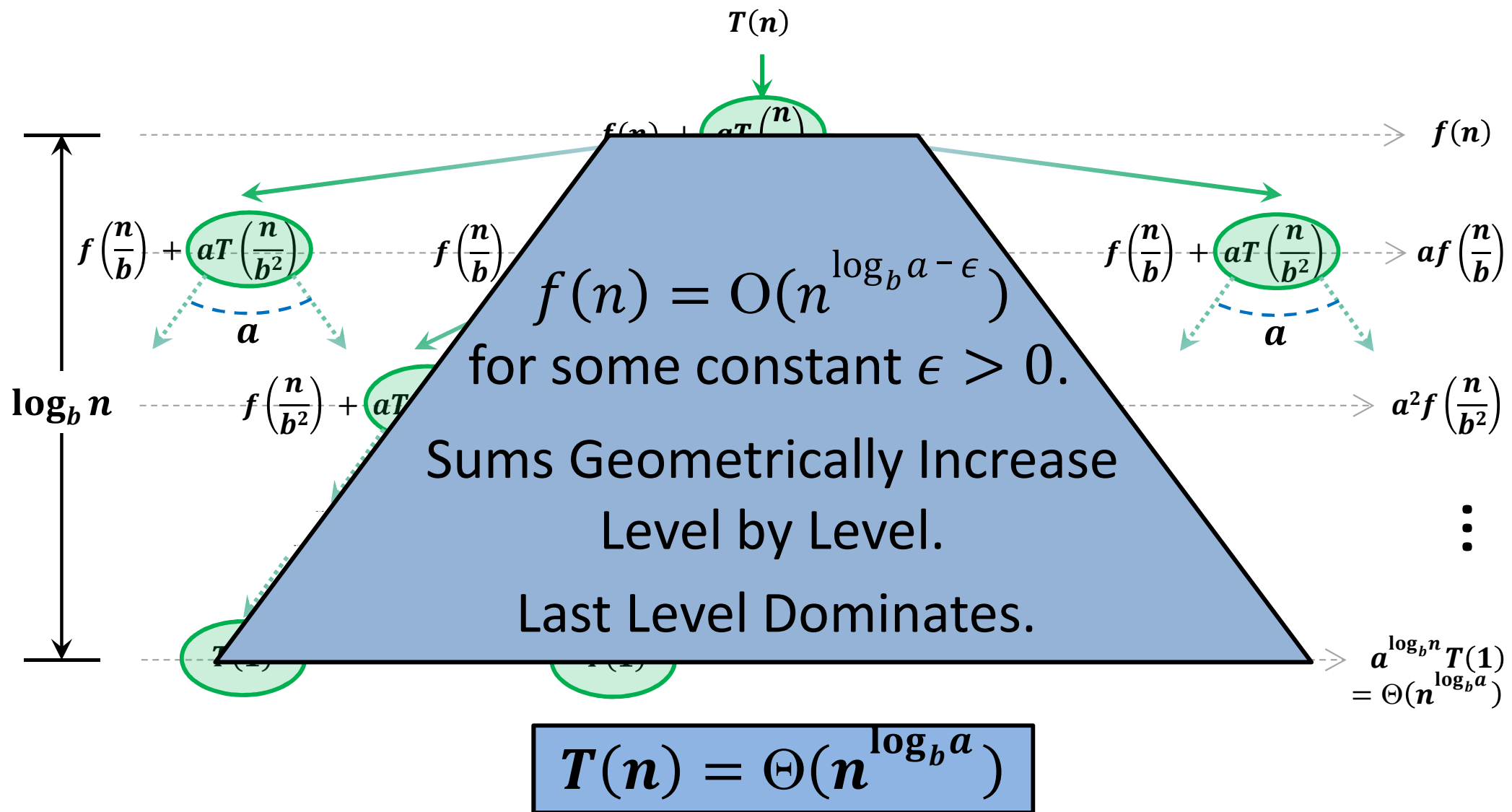
# How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



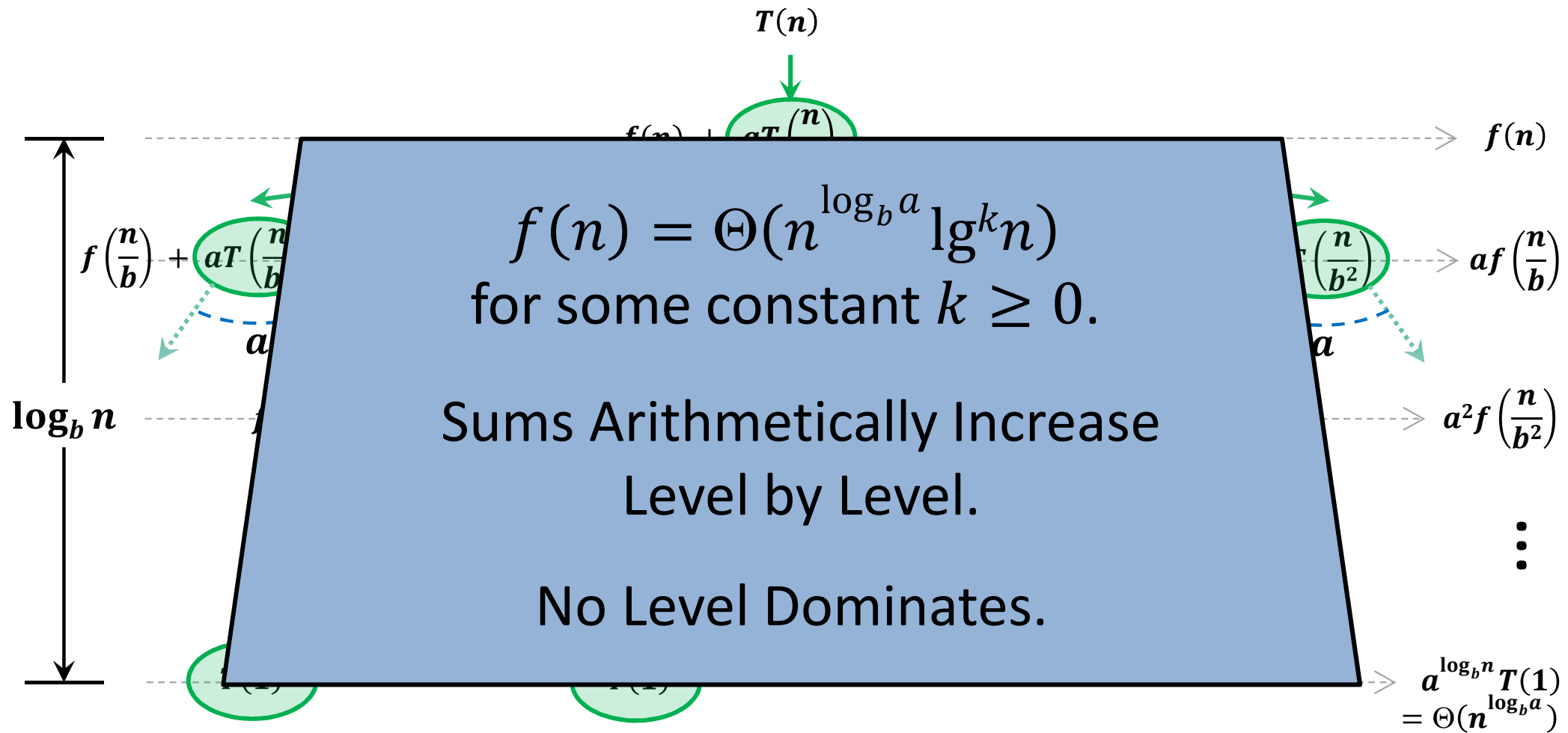
# How the Recurrence Unfolds: Case 1

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



# How the Recurrence Unfolds: Case 2

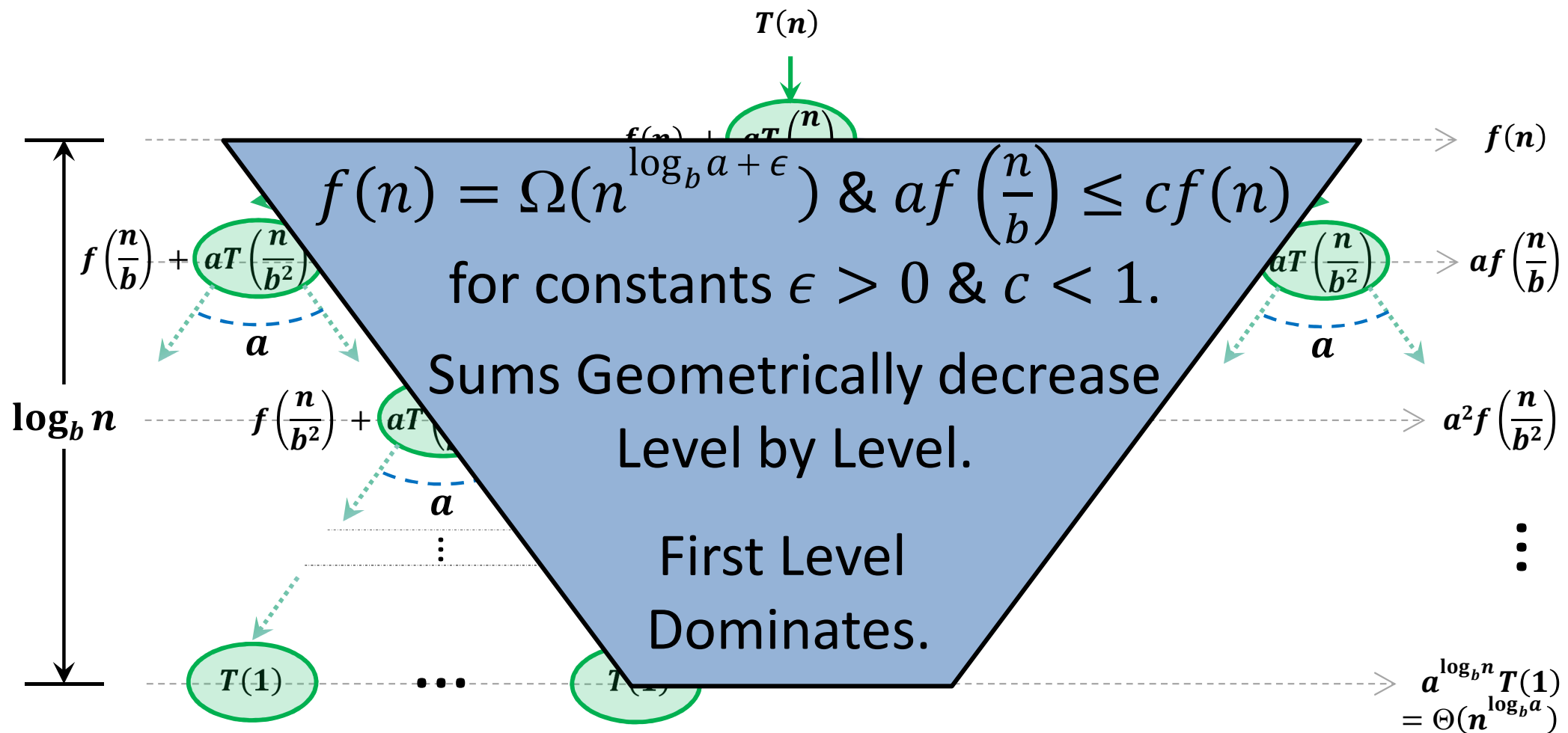
$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



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

# How the Recurrence Unfolds: Case 3

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



$$T(n) = \Theta(f(n))$$

# The Master Theorem

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise } (a \geq 1, b > 1). \end{cases}$$

**Case 1:**  $f(n) = O(n^{\log_b a - \epsilon})$  for some constant  $\epsilon > 0$

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

**Case 2:**  $f(n) = \Theta(n^{\log_b a} \lg^k n)$  for some constant  $k \geq 0$ .

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

**Case 3:**  $f(n) = \Omega(n^{\log_b a + \epsilon})$  and  $af\left(\frac{n}{b}\right) \leq cf(n)$   
for constants  $\epsilon > 0$  and  $c < 1$ .

$$T(n) = \Theta(f(n))$$

# Example Applications of Master Theorem

**Example 1:**  $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n)$

Master Theorem Case 1:  $T(n) = \Theta(n^{\log_2 3})$

**Example 2:**  $T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$

Master Theorem Case 1:  $T(n) = \Theta(n^{\log_2 7})$

**Example 3:**  $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

Master Theorem Case 2:  $T(n) = \Theta(n \log n)$

Assuming that we have an infinite number of processors, and each recursive call in example 2 above can be executed in parallel:

**Example 4:**  $T(n) = T\left(\frac{n}{2}\right) + \Theta(n^2)$

Master Theorem Case 3:  $T(n) = \Theta(n^2)$



# Recurrences not Solvable using the Master Theorem

**Example 1:**  $T(n) = \sqrt{n} T\left(\frac{n}{2}\right) + n$

$a = \sqrt{n}$  is not a constant

**Example 2:**  $T(n) = 2T\left(\frac{n}{\log n}\right) + n^2$

$b = \log n$  is not a constant

**Example 3:**  $T(n) = \frac{1}{2}T\left(\frac{n}{2}\right) + n^2$

$a = \frac{1}{2}$  is not  $\geq 1$

**Example 4:**  $T(n) = 2T\left(\frac{4n}{3}\right) + n$

$b = \frac{3}{4}$  is not  $> 1$ .

# Recurrences not Solvable using the Master Theorem

**Example 5:**  $T(n) = 3T\left(\frac{n}{2}\right) - n$

$f(n) = -n$  is not positive

**Example 6:**  $T(n) = 2T\left(\frac{n}{2}\right) + n^2 \sin n$

violates regularity condition of case 3

**Example 7:**  $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

$f(n) = O(n^{\log_b a})$ , but  $\neq O(n^{\log_b a - \epsilon})$  for any constant  $\epsilon > 0$

**Example 8:**  $T(n) = T\left(\frac{n}{2}\right) + 2T\left(\frac{n}{4}\right) + n$

$a$  and  $b$  are not fixed