# Solving Recurrences: Master Theorem

Dr. Md. Abul Kashem Mia, Professor, CSE Dept, BUET

# Solving Recurrences

- Iteration Method
- Master Method
- Recursion Tree Method

# Solving Recurrences: Iteration Method

- Given: a *divide and conquer* algorithm
  - An algorithm that divides the problem of size $n$ into $a$ subproblems, each of size $n/b$, where $a \geq 1$, $b > 1$
  - The $a$ subproblems are solved recursively, each in time $T(n/b)$
  - Let the cost of each stage (i.e., the work to divide the problem + combine solved subproblems) be described by $cn$
- Then, the recurrence is

$$\boxed{T(n) = \begin{cases} c & n = 1 \\ aT\left(\dfrac{n}{b}\right) + cn & n > 1 \end{cases}}$$

# Solving Recurrences: Iteration Method

- The "iteration method"
  - Expand the recurrence
  - Work some algebra to express as a summation
  - Evaluate the summation
- We show by using

$$\boxed{T(n) = \begin{cases} c & n = 1 \\ aT\left(\dfrac{n}{b}\right) + cn & n > 1 \end{cases}}$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\dfrac{n}{b}\right) + cn & n > 1 \end{cases}$$

- T(n) =

  aT(n/b) + cn

  a(aT(n/b/b) + cn/b) + cn

  $a^2T(n/b^2)$ + cna/b + cn

  $a^2T(n/b^2) + cn(a/b + 1)$

  $a^2(aT(n/b^2/b) + cn/b^2) + cn(a/b + 1)$

  $a^3T(n/b^3) + cn(a^2/b^2) + cn(a/b + 1)$

  $a^3T(n/b^3) + cn(a^2/b^2 + a/b + 1)$

  …

  $a^kT(n/b^k) + cn(a^{k-1}/b^{k-1} + a^{k-2}/b^{k-2} + \ldots + a^2/b^2 + a/b + 1)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So we have
  - $T(n) = a^k T(n/b^k) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
- For $n/b^k = 1$
  - $n = b^k \quad \rightarrow \quad k = \log_b n$
  - $T(n) = a^k T(1) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= a^k c + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= ca^k b^k / b^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= cn\, a^k / b^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a = b$?
  - $T(n) = cn(k + 1)$

    $= cn(\log_b n + 1)$

    $= \Theta(n \log n)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$
  - So:

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$
  - So:

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

  - $T(n) = cn \cdot \Theta(1) = \Theta(n)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$T(n)=\begin{cases} c & n=1 \\ aT\left(\frac{n}{b}\right)+cn & n>1 \end{cases}$$

- So with k = $\log_b n$
  - T(n) = cn($a^k/b^k$ + ... + $a^2/b^2$ + a/b + 1)
- What if a > b?

$$\frac{a^k}{b^k}+\frac{a^{k-1}}{b^{k-1}}+\cdots+\frac{a}{b}+1 \quad = \quad \frac{(a/b)^{k+1}-1}{(a/b)-1} \quad = \quad \Theta\left((a/b)^k\right)$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 \quad = \quad \frac{(a/b)^{k+1} - 1}{(a/b) - 1} \quad = \quad \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 \quad = \quad \frac{(a/b)^{k+1} - 1}{(a/b) - 1} \quad = \quad \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 \quad = \quad \frac{(a/b)^{k+1} - 1}{(a/b) - 1} \quad = \quad \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$$

*recall logarithm fact:* $a^{\log_b n} = n^{\log_b a}$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 \quad = \quad \frac{(a/b)^{k+1} - 1}{(a/b) - 1} \quad = \quad \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$

*recall logarithm fact:* $a^{\log_b n} = n^{\log_b a}$

$= cn \cdot \Theta(n^{\log_b a} / n) = \Theta(cn \cdot n^{\log_b a} / n)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 \quad = \quad \frac{(a/b)^{k+1} - 1}{(a/b) - 1} \quad = \quad \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$

*recall logarithm fact:* $a^{\log_b n} = n^{\log_b a}$

$= cn \cdot \Theta(n^{\log_b a} / n) = \Theta(cn \cdot n^{\log_b a} / n)$

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

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So…

$$T(n) = \begin{cases} \Theta(n) & a < b \\ \Theta(n \log_b n) & a = b \\ \Theta\left(n^{\log_b a}\right) & a > b \end{cases}$$

# The Master Theorem

- Given: a *divide and conquer* algorithm
  - An algorithm that divides the problem of size $n$ into $a$ subproblems, each of size $n/b$, where $a \geq 1$, $b > 1$
  - The $a$ subproblems are solved recursively, each in time $T(n/b)$
  - Let the cost of each stage (i.e., the work to divide the problem + combine solved subproblems) be described by the function $f(n)$, where $f$ is asymptotically positive
  - $T(n)$ is monotonically increasing function
- Then, the Master Theorem gives us a *cookbook* for the algorithm's running time.

# The Master Theorem: Pitfalls

- You cannot use the Master Theorem if
  - $T(n)$ is not monotone, e.g. $T(n) = \sin(x)$
  - $f(n)$ is not a polynomial, e.g., $T(n) = 2T(n/2) + 2^n$
  - $b$ cannot be expressed as a constant, e.g.

$$T(n) = T(\sqrt{n})$$

- Note that the Master Theorem does not solve all recurrence equations

# The Master Theorem

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = a\,T(n/b) + f(n).$$

Then $T(n)$ has the following asymptotic bounds:

- If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then

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

- If $f(n) = \Theta(n^{\log_b a})$ then

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

- If $f(n) = O(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ then

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

# Excuse me, what did it say ???

- Essentially, the Master theorem compares the function $f(n)$ with the function $g(n) = n^{\log_b(a)}$.

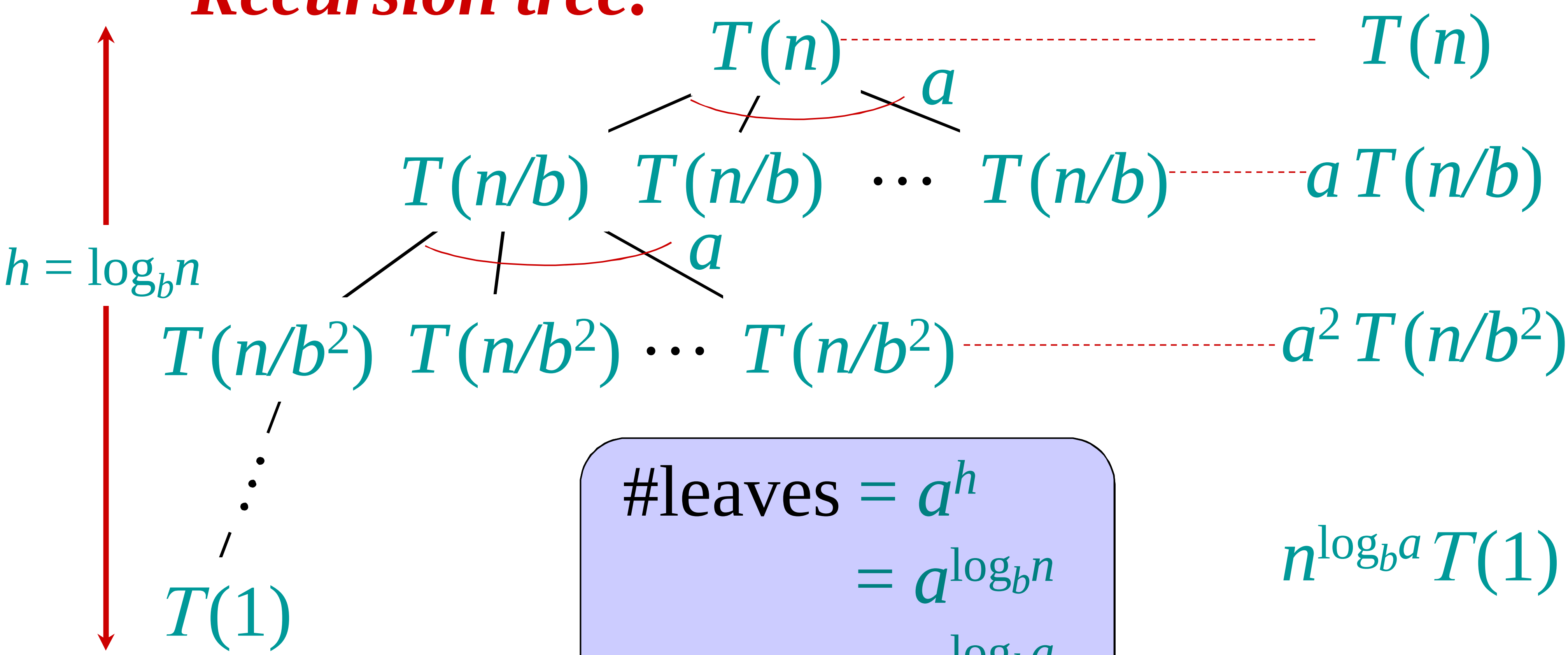
  Roughly, the theorem says:

  - If $f(n) << g(n)$ then $T(n) = \Theta(g(n))$
  - If $f(n) \approx g(n)$ then $T(n) = \Theta(g(n) \log_b n)$
  - If $f(n) >> g(n)$ then $T(n) = \Theta(f(n))$

- Now go back and memorize the theorem !!!

# Idea of Master Theorem

***Recursion tree:***




$$\#\text{leaves} = a^h = a^{\log_b n} = n^{\log_b a}$$

# Idea of Master Theorem

Let us iteratively substitute the recurrence:

$$
\begin{aligned}
T(n) &= aT(n/b) + f(n) \\
&= a(aT(n/b^2)) + f(n/b)) + f(n) \\
&= a^2T(n/b^2) + af(n/b) + f(n) \\
&= a^3T(n/b^3) + a^2 f(n/b^2) + af(n/b) + f(n) \\
&= \ldots \\
&= a^{\log_b n}T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f(n/b^i) \\
&= n^{\log_b a}T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f(n/b^i)
\end{aligned}
$$

# Idea of Master Theorem

- Thus, we obtained

$$T(n) = n^{\log_b(a)}\, T(1) + \Sigma\, a^i\, f(n/b^i)$$

The proof proceeds by distinguishing three cases:

- The first term is dominant:

$$f(n) = O(n^{\log_b(a)-\varepsilon})$$

- Each term of the summation is equally dominant:

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

- The second term is dominant and can be bounded by a geometric series:

$$f(n) = \Omega(n^{\log_b(a)+\varepsilon})$$

# Master Theorem: 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 $n^{\log_b a}$

   ***Solution:*** $T(n) = \Theta(n^{\log_b a})$.

2. $f(n) = \Theta(n^{\log_b a})$
   - $f(n)$ and $n^{\log_b a}$ grow at similar rates.

   ***Solution:*** $T(n) = \Theta(n^{\log_b a} \log n)$.

3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially faster than $n^{\log_b a}$

   ***Solution:*** $T(n) = \Theta(f(n))$.

# Master Theorem: Examples

***Ex.*** $T(n) = 4T(n/2) + n$

$a = 4, b = 2 \quad \Rightarrow n^{\log_b a} = n^2; \quad f(n) = n.$

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

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

***Ex.*** $T(n) = 4T(n/2) + n^2$

$a = 4, b = 2 \quad \Rightarrow n^{\log_b a} = n^2; \quad f(n) = n^2.$

**CASE 2**: $f(n) = \Theta(n^2)$.

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

***Ex.*** $T(n) = 4T(n/2) + n^3$

$a = 4, b = 2 \quad \Rightarrow n^{\log_b a} = n^2; \quad f(n) = n^3.$

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

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