ADVANCED ANALYSIS OF ALGORITHMS
CPS 5440

OMAR DIB

UNIT 2: RECURSIVE FUNCTIONS. DIVIDE-AND-CONQUER. THE MASTER THEOREM.

ASYMPTOTIC NOTATION

- O-notation (upper bounds):
- We write $f(n) = O(g(n))$ if there exist constants $c > 0, n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.

- EXAMPLE:
  - $f(n) = 2n + 3 = O(n)\ (c = 5, n_0 = 1)$
    - $2n + 3 \leq 2n + 3n\ for\ all\ n \geq 1$
    - $2n + 3 \leq 5n$
    - $f(n) = O(n)$
  - $1 < logn < \sqrt{n} < \boxed{n < nlogn < n^2 < n^3 < \ \ldots < 2^n < 3^n < n^n}$
  - $f(n) = 2n^2 = O(n^2)\ (c = 2, n_0 = 1)$

    $= O(n^3)\ (c = 1, n_0 = 2)$

    ....

ASYMPTOTIC NOTATION

- Ω-notation (lower bounds)
- We write $f(n) = \Omega(g(n))$ if there exist constants $c > 0, n_0 > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.

- EXAMPLE:
- $f(n) = 2n + 3 = \Omega(n)$ $(c = 1, n_0 = 1)$
  - $2n + 3 \geq n$ $for\ all\ n \geq 1$
  - $f(n) = \Omega(n)$
- $\boxed{1 < logn < \sqrt{n} < n} < nlogn < n^2 < n^3 < \ ... < 2^n < 3^n < n^n$

- $f(n) = \sqrt{n} = \Omega(\lg n)(c = 1, n_0 = 16)$

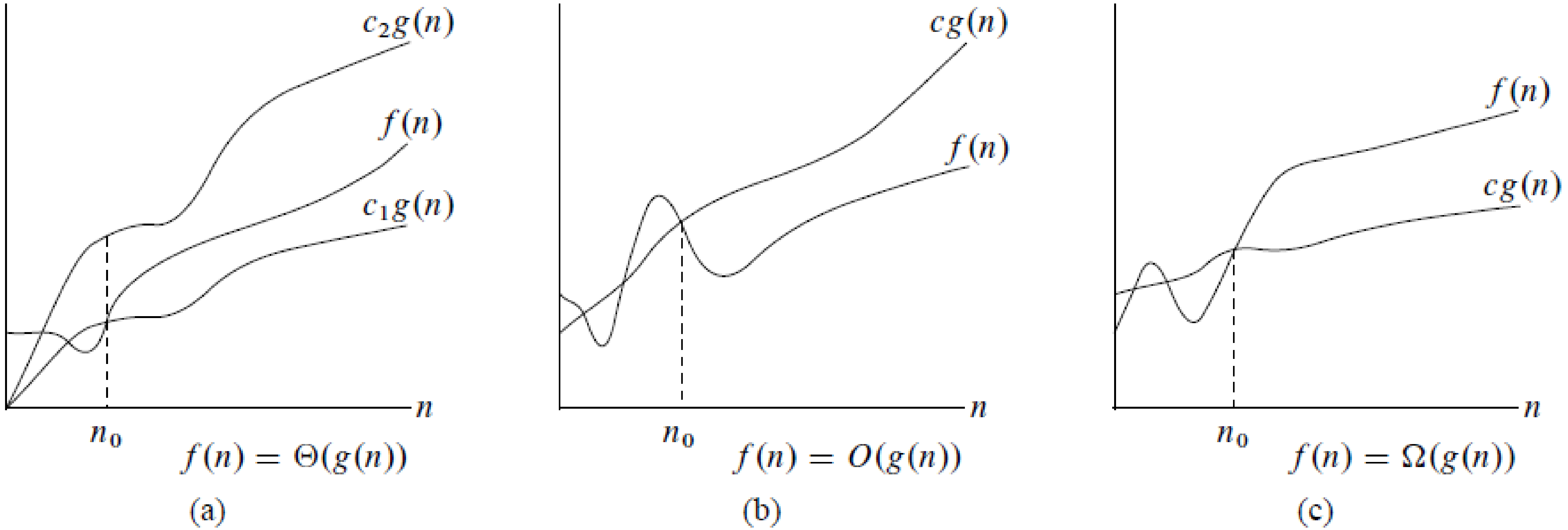
ASYMPTOTIC NOTATION

- Θ-notation (tight bounds)
- We write $f(n) = \Theta(g(n))$ if there exist constants $c_1, c_2, n_0 > 0$ such that $c_1\, g(n) \leq f(n) \leq c_2\, g(n)$ for all $n \geq n_0$.
- $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$

- EXAMPLE:
- $f(n) = 2n + 3 = \Theta(n)$ $(c_1 = 1, c_2 = 5, n_0 = 1)$
  - $n \leq 2n + 3 \leq 5n \; for \; all \; n \geq 1$
  - $f(n) = \Theta(n)$
- $1 < logn < \sqrt{n} < n < nlogn < n^2 < n^3 < \ldots < 2^n < 3^n < n^n$

- $\frac{1}{2}n^2 - 2n = \Theta\,(n^2)$

ASYMPTOTIC NOTATION



ASYMPTOTIC NOTATION VS (BEST, WORST, AND AVERAGE CASE)

- Asymptotic notation and (Best, Worst, and Average case) are different!

SOLVING RECURRENCES

SOLVING RECURRENCES

- The analysis of merge sort from Lecture 1 required us to solve a recurrence.
- Recurrences are like solving integrals, differential equations, etc.
  - Learn a few tricks.

SUBSTITUTION METHOD: CONCEPT

- Substitution method for solving recurrences consists of two steps:
- Guess the form of the solution, e.g., $T(n) = O\big(g(n)\big)$,then
- Use mathematical induction to find constants $(c \ and \ n_0)$ in the form and show that the solution works
  - **Step 1 (Base step)** – Prove that the guess is true for the initial value
  - **Step 2 (Inductive step)** – Prove that if the guess is true for $T(k) \le \mathrm{c}\ g(k), \forall\ k < n$, then this implies that $T(n) \le \mathrm{c}\ g(n)$, for some $c > 0$ and $n \ge n_0$
- The inductive hypothesis is applied to smaller values, similar like recursive calls bring us closer to the base case
- The substitution method is a powerful way to establish lower or upper bounds on a recurrence
- It applies in cases when it is easy to guess the form of the solution

SUBSTITUTION METHOD: MAKING A GOOD GUESS

- There is no general way to guess the correct solution to recurrences.
- Guessing a solution takes experience and, occasionally, creativity.
- Some heuristics can help us make a good guess ($e.g.$, Iteration Method, Recursion Tree)
- If a recurrence is similar to a one, we have seen before, then guessing a similar solution is reasonable
- For example, $T(n) = 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 17) + n$, we make the guess that $T(n) = O(n \lg n)$ like Merge Sort
- Another way to make a good guess is to prove the loose upper and lower bounds on the recurrence and then reduce the range of uncertainty. For example:
  - Start with and prove the initial lower bound of $T(n) = \Omega(n)$ for the recurrence
  - Start with and prove the initial upper bound of $T(n) = O(n^2)$ for the recurrence
  - Then gradually lower the upper bound and raise the lower bound until convergence to correct, asymptotically tight solution of $T(n) = \Theta(n \lg n)$
- Sometimes the correct guess at an asymptotic bound on the solution of a recurrence does not work. This can be solved by revising the guess and subtracting a lower-order term in the guess.

SUBSTITUTION METHOD: MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$

- Guess: $T(n) = O(n \lg n)$, or $T(n) \leq \mathrm{c}\,.\,n \lg n$, for some constant c and $n_0 \leq n$

- Hypothesis: $T(k) \leq \mathrm{c}\,.\,k \lg k\,,\ \forall\ k < n,$ we will use $k = \frac{n}{2}$

- Inductive Step:

$$T(n) = 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

$$\leq 2\,.\,c\,.\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

$$\leq c\,.\,n \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

$$= c\,.\,n \lg n \ - c\,.\,n \lg 2 + n$$

$$= c\,.\,n \lg n \ - c\,.\,n + n$$

$$\leq \mathrm{c}\,.\,n \lg n \qquad if: -c.n + n \leq 0 \ \Rightarrow c \ \geq 1$$

SUBSTITUTION METHOD: MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$

- Guess: $T(n) = O(n \lg n)$, or $T(n) \leq c \,.\, n \lg n$, for some constant c and $n_0 \leq n$
- Hypothesis: $T(k) \leq c \,.\, k \lg k$, $\forall\ k < n$
- From inductive step: $T(n) \leq c \,.\, n \lg n$ when $c \geq 1$
- Base step: $T(1) \leq c \,.\, 1 \lg 1$?
  - Impossible as $T(1) = 1 \nleq c \,.\, 1 \lg 1 = 0$. (Problem!)
  - But we only want to show that $T(n) \leq c \,.\, n \lg n$ for sufficiently large values of n; $i.e.$, $\forall\ n \geq n_0$.
  - Solution: Try $n_0 > 1$
- Base steps (check boundaries)
  - We must check both $T(2)$ and $T(3)$ simultaneously because of the nature of the recursive equation
  - Check $T(2)$ and $T(3)$

SUBSTITUTION METHOD : MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$
- Guess: $T(n) = O(n \lg n)$, or $T(n) \leq \mathrm{c} \,.\, n \lg n$, for some constant c and $n_0 \leq n$
- Hypothesis: $T(k) \leq \mathrm{c} \,.\, k \lg k$, $\forall\ k < n$
- From inductive step: $T(n) \leq \mathrm{c} \,.\, n \lg n$ when $c \geq 1$
- Base step: $T(1) \leq \mathrm{c} \,.\, n \lg n \ \forall\ n \geq 1\ (n_0 > 1)$
- Base step boundaries:
  - $T(1) = 1 \Longrightarrow \begin{cases} T(2) = 4 \\ T(3) = 5 \end{cases}$
  - We want to satisfy simultaneously
  - $\begin{cases} 4 = T(2) \leq \mathrm{c} \,.2 \lg 2 \\ 5 = T(3) \leq \mathrm{c} \,.3 \lg 3 \end{cases} \Longrightarrow \begin{cases} c \geq 2 \\ c \geq 1.052 \end{cases} \Longrightarrow c \geq 2$
- We prove that $T(n) \leq \mathrm{c} \,.\, n \lg n$, *with* $c = 2$, *and* $n_0 = 2$, *So* $T(n) = O(n \lg n)$

❑ Recursion-tree method

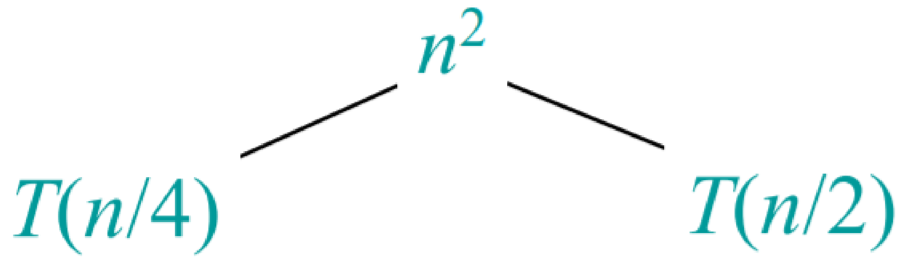
- It models the costs (time) of a recursive execution of an algorithm.
- It can be unreliable, just like any method that uses ellipses (…).
- It promotes intuition, however.
- Good for generating guesses for the substitution method.

RECURSION TREE

Solve $T(n) = \mathrm{T}(\mathrm{n}/4) + \mathrm{T}\ (\mathrm{n}/2) + \mathrm{n}^2$

$$T(n)$$

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



RECURSION TREE

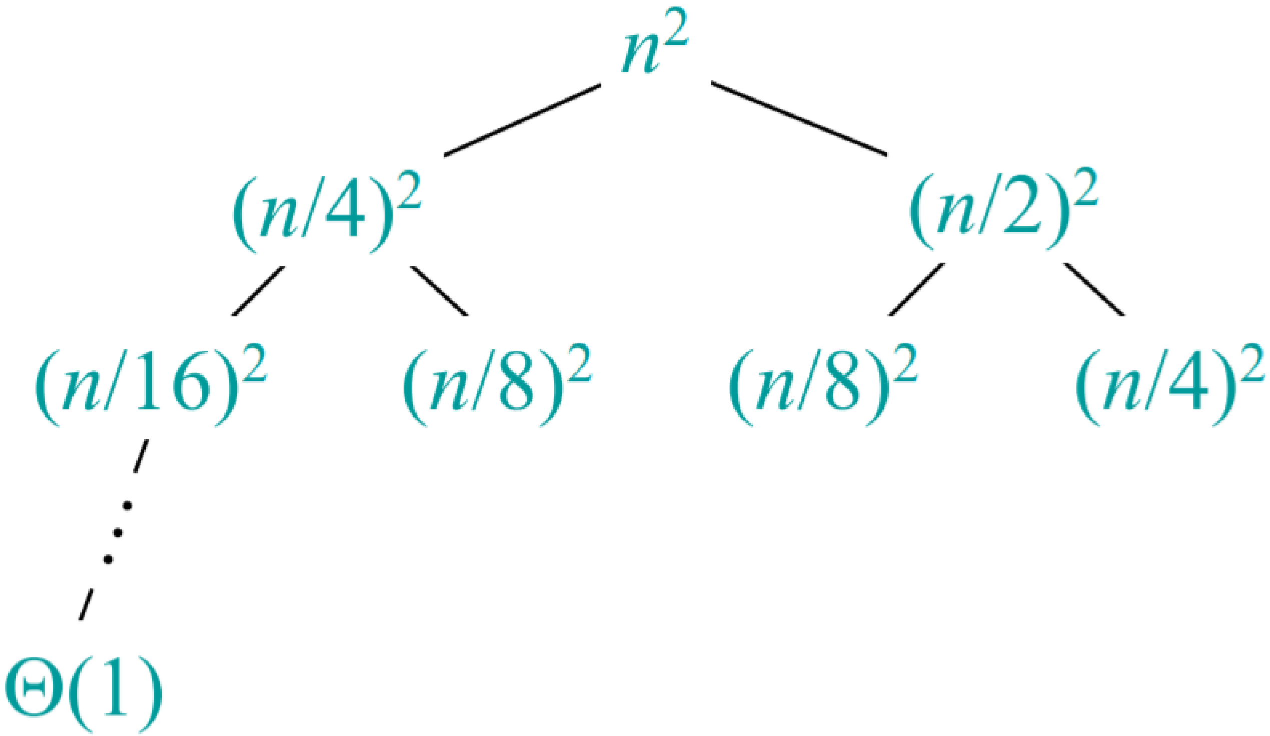
Solve $T(n) = T(n/4) + T(n/2) + n^2$

$$n^2$$

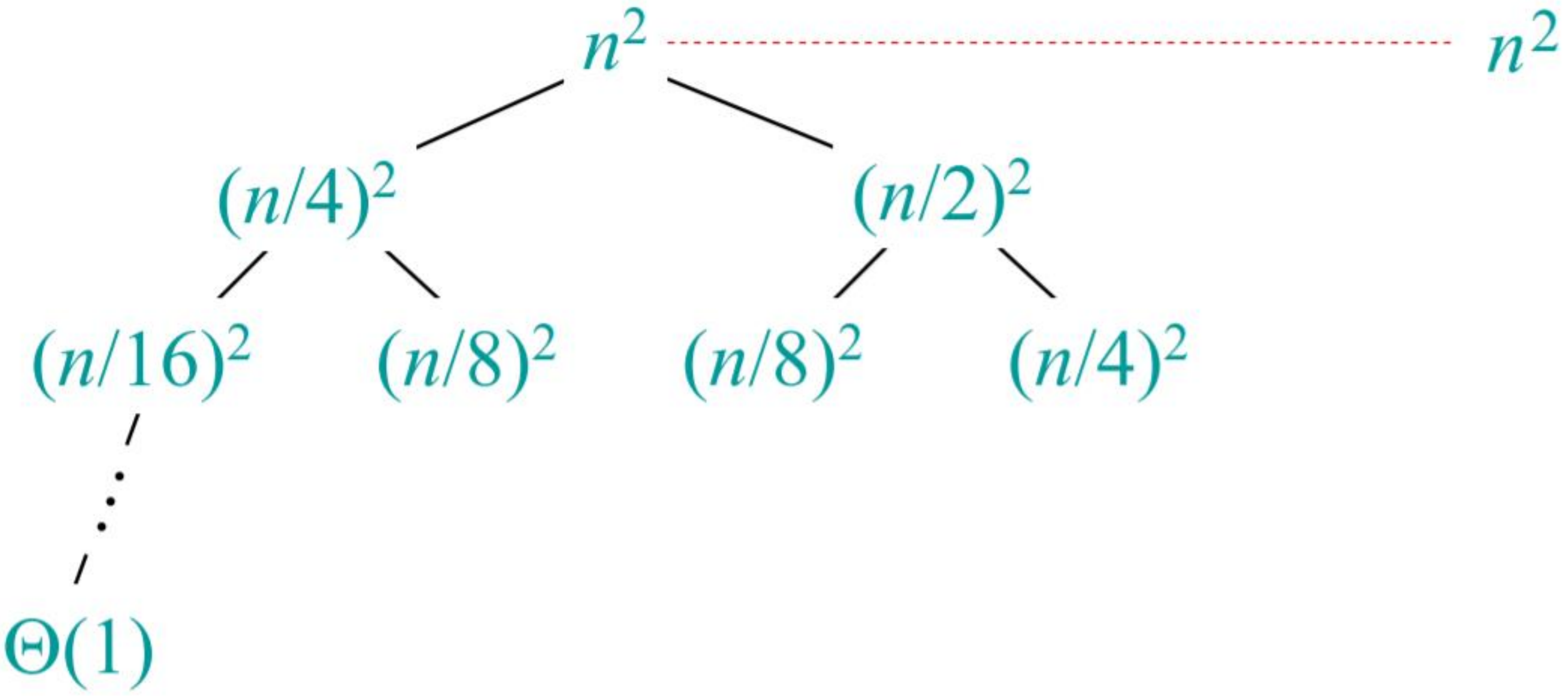
$$(n/4)^2 \qquad\qquad (n/2)^2$$

$$T(n/16) \quad T(n/8) \quad T(n/8) \quad T(n/4)$$

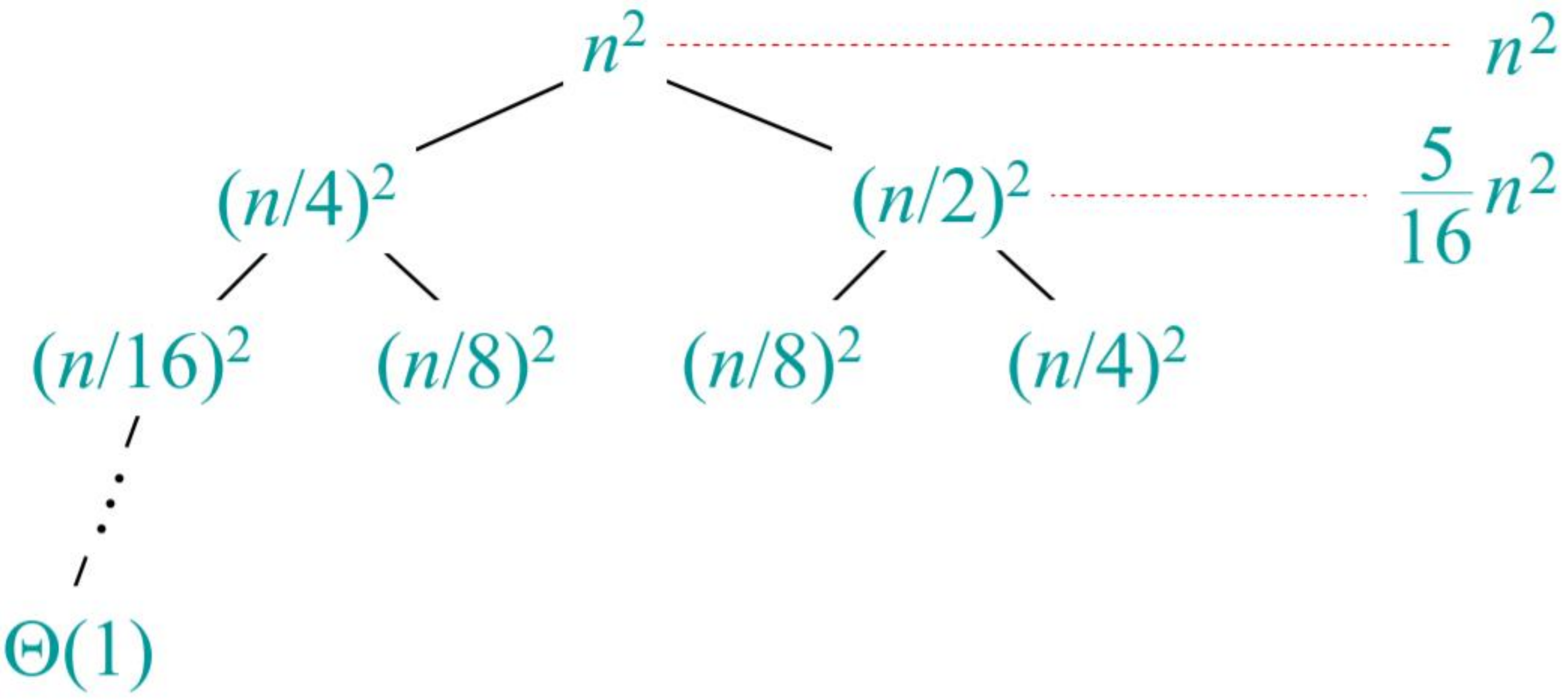
Solve $T(n) = T(n/4) + T(n/2) + n^2$



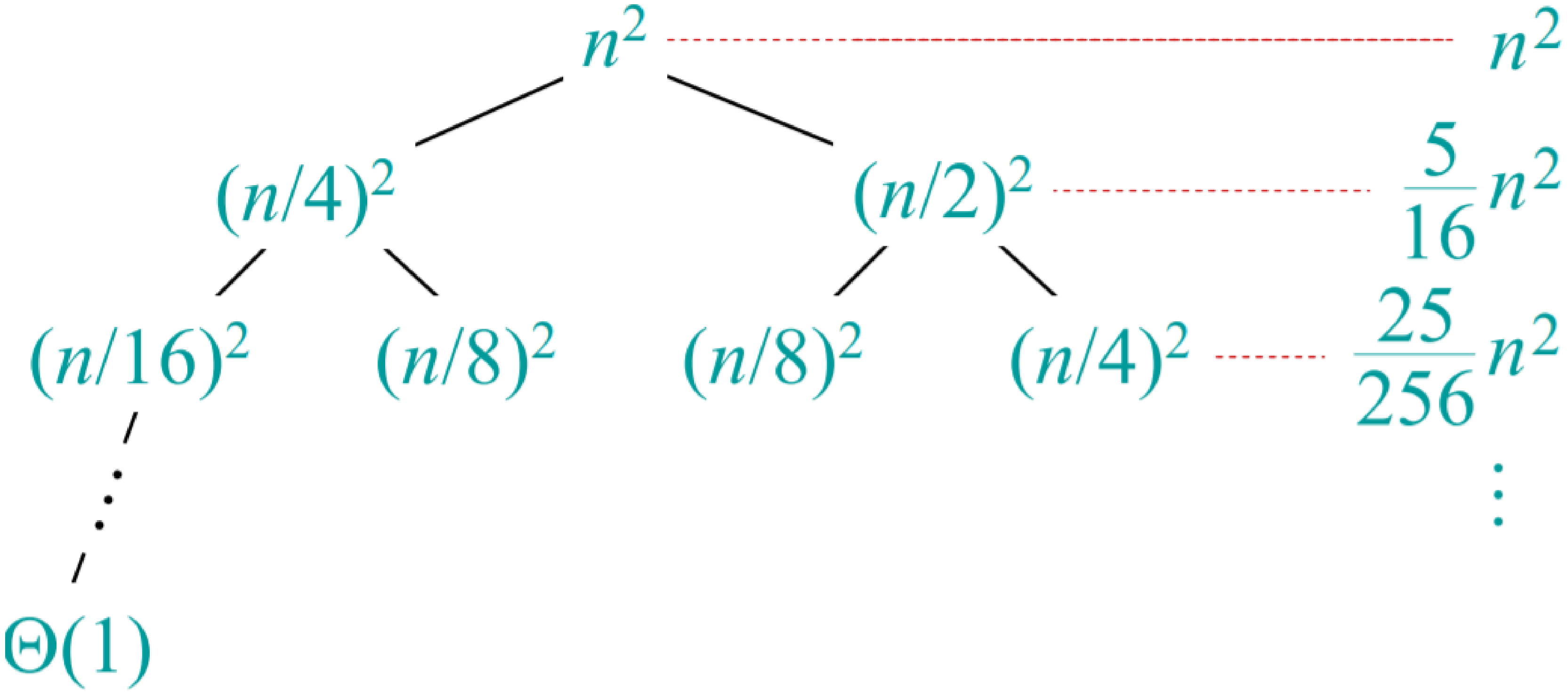
Solve $T(n) = T(n/4) + T(n/2) + n^2$



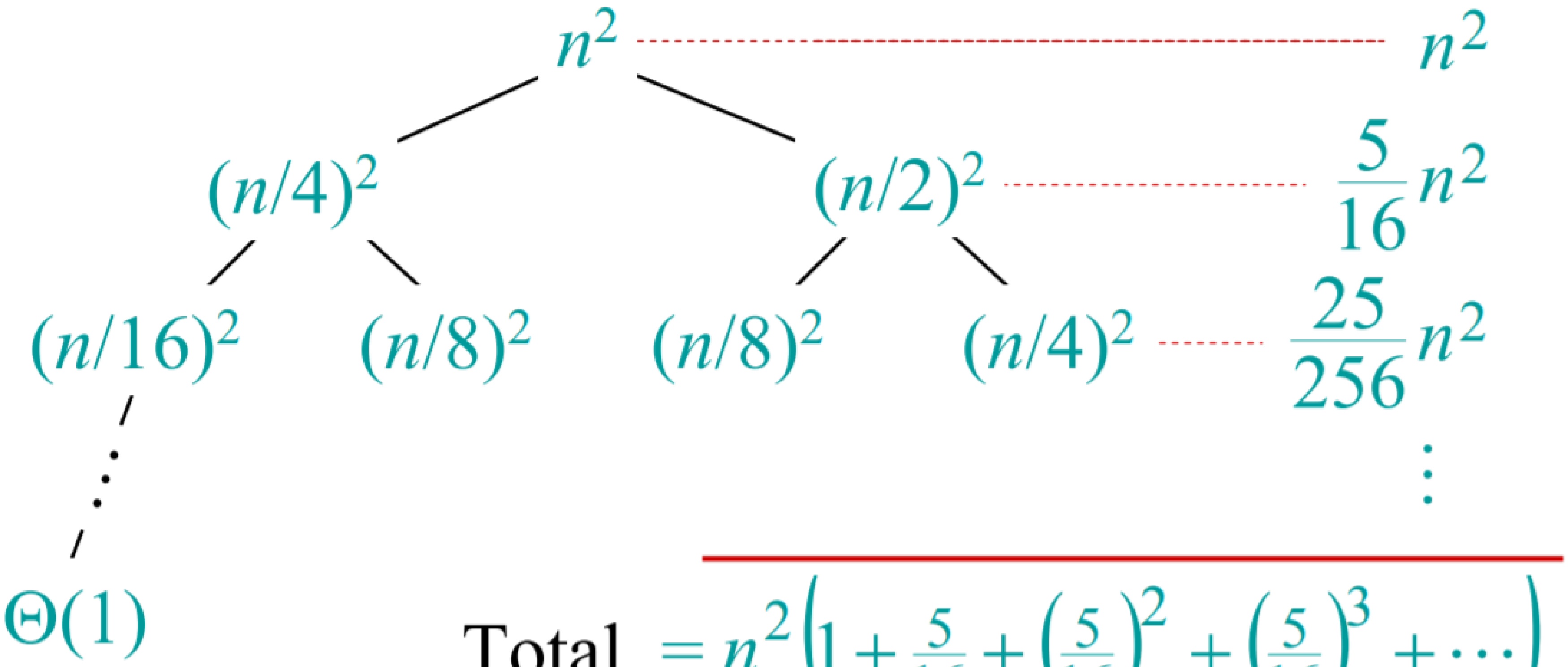
Solve $T(n) = T(n/4) + T(n/2) + n^2$



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



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



$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

$$= \Theta(n^2)$$ *geometric series*

❑ Iteration method

- Do some iterations to find a pattern
- Use the base step along with the pattern to guess the time complexity
- Finding the pattern might be challenging
  - Tip: Do not rush to simplify the calculation
- Good for generating guesses for the substitution method.

ITERATION METHOD

- $T(n) = \begin{cases} 0 & n = 0 \\ T(n-1) + 2n & n > 0 \end{cases}$

$\boldsymbol{k}$	$\boldsymbol{T(n)}$
1	$\boldsymbol{T(n) = T(n-1) + 2n}$
2	$T(n) = T(n-2) + 2n - 2 + 2n \quad = \boldsymbol{T(n-2) + 2(2n) - 2}$
3	$T(n) = T(n-3) + 2n - 4 + 2(2n) - 2 = \boldsymbol{T(n-3) + 3(2n) - 4 - 2}$
4	$T(n) = T(n-4) + 2n - 6 + 3(2n) - 4 - 2 = \boldsymbol{T(n-4) + 4(2n) - 6 - 4 - 2}$
$\vdots$	
k	$T(n) = T(n-k) + k(2n) - 2[(k-1) + (k-2) + \cdots + 2 + 1]$ $= \boldsymbol{T(n-k) + k(2n) - 2\sum_{i=1}^{k-1} i}$

$T(0) = 0 \Rightarrow T(n-k) = 0 \Rightarrow n = k \Rightarrow$

$$T(n) = T(0) + n(2n) - 2\sum_{i=1}^{k-1} i = 0 + 2n^2 - 2\left[\frac{(n-1)(n)}{2}\right]$$

$$= 2n^2 - n^2 + n = n^2 + n$$

$$= \mathbf{O}(\boldsymbol{n^2})$$

$$\boldsymbol{\sum_{i=1}^{n} i = \frac{n(n+1)}{2}}$$

THE MASTER THEOREM.

MOTIVATION: ASYMPTOTIC BEHAVIOR OF RECURSIVE ALGORITHMS

- When analyzing algorithms, recall that we only care about the asymptotic behavior
- Recursive algorithms are no different
- Rather than solving exactly the recurrence relation associated with the cost of an algorithm, it is sufficient to give an asymptotic characterization
- The main tool for doing this is the master theorem

- 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.

THREE COMMON CASES

- $T(n) = a\,T(n/b) + f(n)$, where $a \geq 1, b > 1,$
- Compare $f(n)$ with $n^{\log_b a}$:
- Case 1:

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

  - $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^{ε} factor).

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

THREE COMMON CASES

- $T(n) = 9\,T\left(\frac{n}{3}\right) + n$
- $a = 9$
- $b = 3$
- $f(n) = n$

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

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

Master theorem can be applied

THREE COMMON CASES

- $T(n) = 9\,T\left(\frac{n}{3}\right) + n$ $//a = 9; b = 3; f(n) = n$
- Compare $f(n)$ with $n^{log_b a}$
  - $log_b a = log_3 9 = 2 \Rightarrow n^{log_b a} = n^2$
  - $f(n) = n$
- $O\left(n^{log_b a - \varepsilon}\right) = O(n^{2-\varepsilon}) = O(n) = f(n)\ for\ \varepsilon = 1$
- $Then{:}\ T(n) = \Theta\left(n^{log_b a}\right) = \Theta\left(n^{log_3 9}\right) = \Theta(n^2)$

THREE COMMON CASES

- $T(n) = a\,T(n/b) + f(n)$, where $a \geq 1, b > 1,$
- Compare $f(n)$ with $n^{log_b a}$:
- Case 2:

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

$f(n)$ and $n^{log_b a}$ grow at similar rates.

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

THREE COMMON CASES

- $T(n) = T\left(\frac{2n}{3}\right) + 1$
- $a = 1$
- $b = \frac{3}{2}$
- $f(n) = 1$

$$T(n) = aT(n/b) + f(n),$$

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

Master theorem can be applied

- $T(n) = T\left(\frac{2n}{3}\right) + 1$ // $a = 1;\ b = \frac{3}{2};\ f(n) = 1$
- $log_b a = log_{3/2} 1 = 0$
- $\Theta(n^{log_b a}) = \Theta(n^0) = \Theta(1) = f(n) = 1$

Then: $T(n) = \Theta(n^{log_b a} \lg n) = \Theta(1 \lg n) = \Theta(\lg n)$

THREE COMMON CASES

- $T(n) = a\,T(n/b) + f(n)$, where $a \geq 1, b > 1,$
- Compare $f(n)$ with $n^{log_b a}$:
- Case 3:

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

$f(n)$ grows polynomially faster than $n^{log_b a}$ (by an n^{ε} factor).

and $f(n)$ **satisfies the regularity condition**:

$a\,f\left(\frac{n}{b}\right) \leq c\,f(n)$ for some constant $c < 1\ \forall\ n.$

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

THREE COMMON CASES

- $T(n) = 3\,T\left(\frac{n}{4}\right) + nlgn$
- $a \;= 3$
- $b \;= 4$
- $f(n) = nlgn$

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

where $a \;\geq\; 1, b \;>\; 1,$

Master theorem can be applied

THREE COMMON CASES

- $T(n) = 3\,T\left(\frac{n}{4}\right) + nlgn$ // $a = 3;\ b = 4;\ f(n) = nlgn$
- $log_b a = log_4 3 = 0.793$
- $\Omega\left(n^{log_b a+\varepsilon}\right) = \Omega\left(n^{0.793+\varepsilon}\right) = \Omega(n) = nlgn = f(n)\ for\ \varepsilon = 0.2$
- $a\ f\left(\frac{n}{b}\right) = 3\frac{n}{4}\lg\frac{n}{4} \le c.f(n)$for some constant $c < 1\ \forall\ n$
  - Let assume $c = $ ¾ $\rightarrow \frac{3}{4}n\lg\frac{n}{4} \le \frac{3}{4}nlgn$ = $True$ for all n

$Then{:}\, T(n) = \Theta(f(n)) = \Theta(nlgn)$

MASTER THEOREM: EXAMPLES

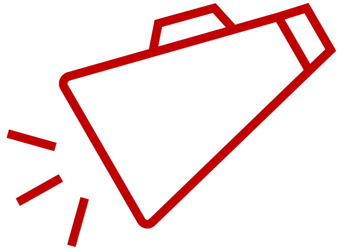
- $T(n) = 4\,T\left(\frac{n}{2}\right) + n$
- // $a = 4;\ b = 2; f(n) = n$
- $log_b a = log_2 4 = 2 \Longrightarrow n^{log_b a} = n^2$
- $O\left(n^{log_b a - \varepsilon}\right) = O(n^{2-\varepsilon}) = O(n) = f(n)\ for\ \varepsilon = 1$//Case 1
- $Then{:}\ T(n) = \Theta\left(n^{log_b a}\right) = \Theta(n^2)$

- $T(n) = 4\,T\left(\frac{n}{2}\right) + n^2$
- // $a = 4;\ b = 2; f(n) = n^2$
- $log_b a = log_2 4 = 2$
- $\Theta\left(n^{log_b a}\right) = \Theta(n^2) = f(n)$ //Case 2
- *Then:* $T(n) = \Theta\left(n^{log_b a} \lg n\right) = \Theta(n^2 \lg n)$

MASTER THEOREM: EXAMPLES

- $T(n) = 4\,T\left(\frac{n}{2}\right) + n^3$
- // $a = 4;\; b = 2; f(n) = n^3$
- $log_b a = log_2 4 = 2 \Longrightarrow n^{log_b a} = n^2$
- $\Omega\left(n^{log_b a + \varepsilon}\right) = \Omega\left(n^{2+\varepsilon}\right) = f(n)\; for\, \varepsilon = 1$
- ***And*** (**regularity condition**) $a\, f\left(\frac{n}{b}\right) = 4\left(\frac{n}{2}\right)^3 = \frac{n^3}{2} \le c.f(n) \le c.n^3$ $for\ some\ constant\ c = \frac{1}{2} < 1\; \forall\, n$
- ***Then:*** $T(n) = \Theta\big(f(n)\big) = \Theta(n^3)$

MASTER THEOREM: EXAMPLES

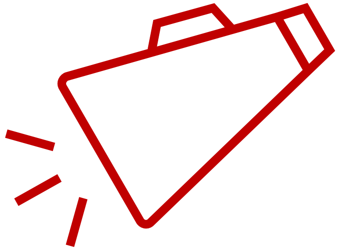


- $T(n) = 4\,T\left(\frac{n}{2}\right) + \frac{n^2}{\lg n}$
- // $a\ = 4;\ b\ = 2; f(n) = \frac{n^2}{\lg n}$
- $log_b a = log_2 4 = 2 \Longrightarrow n^{log_b a}\ =\ n^2$
- You might mistakenly think that case 1 should apply, since:

$f(n) = \frac{n^2}{\lg n}$ is asymptotically smaller than $n^{log_b a}\ =\ n^2$.

- However, Master method does not apply here.
- In particular, for every constant $\varepsilon\ >\ 0$, we have $n^{\varepsilon}\ =\ \omega\,(\lg n)$.
- $f(n)$ does not grow polynomially slower than $n^{log_b a}$ by an n^{ε} factor.
- Consequently, the recurrence falls into the gap between case 1 and case 2.

MASTER THEOREM: EXAMPLES



- $T(n) = 2\,T(^n/_2) + n \lg n$
- // $a = 2;\ b = 2; f(n) = n \lg n$
- $log_b a = log_2 2 = 1 \Longrightarrow n^{log_b a} = n$
- You might mistakenly think that case 3 should apply, since:

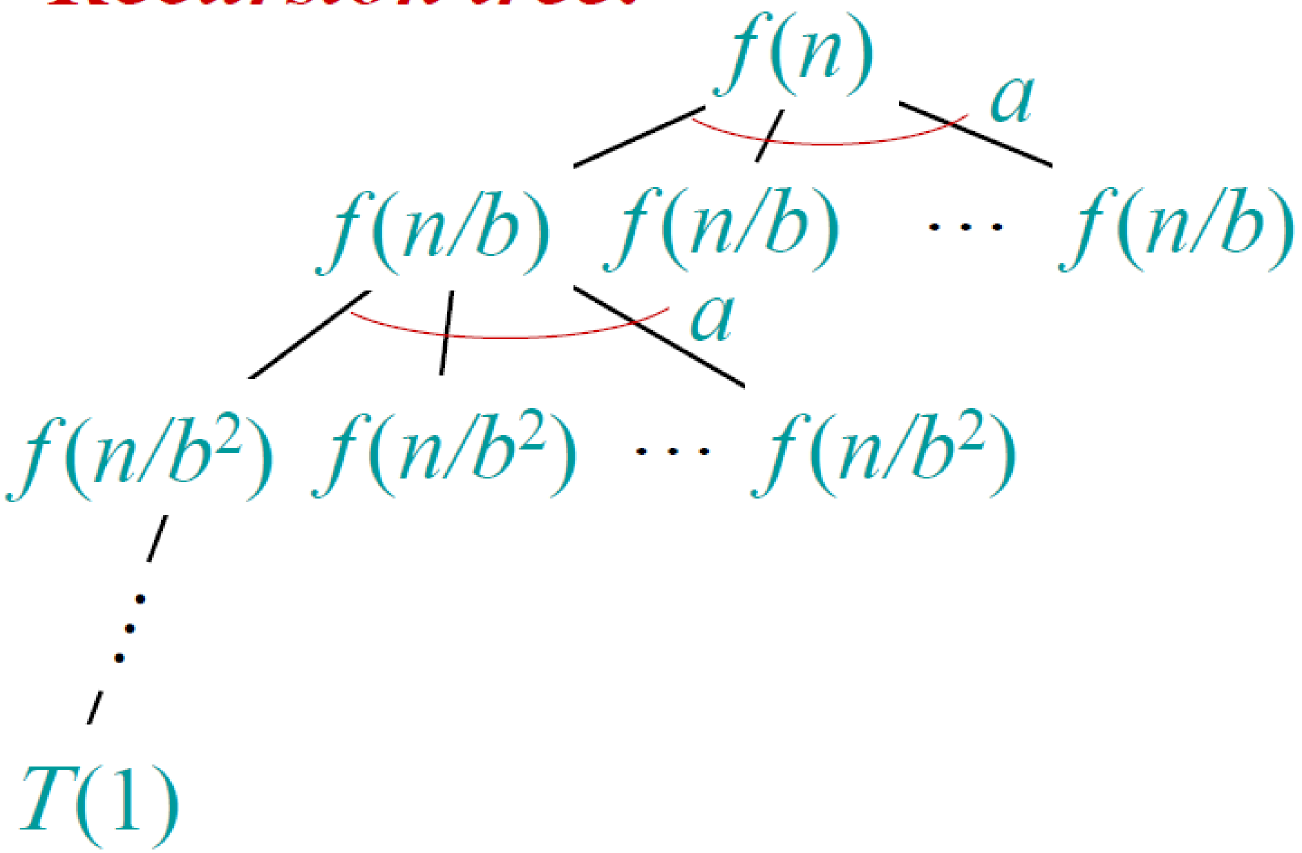
$f(n) = n \lg n$ is asymptotically larger than $n^{log_b a} = n$.

- The problem is that it is not <u>polynomially</u> larger.
- Master method does not apply here.
- The ratio $f(n)/\,n^{log_b a} = (n \lg n)/n = \lg n$ is asymptotically less than n^{ε} for any positive constant ε.
- Consequently, the recurrence falls into the gap between case 2 and case 3.

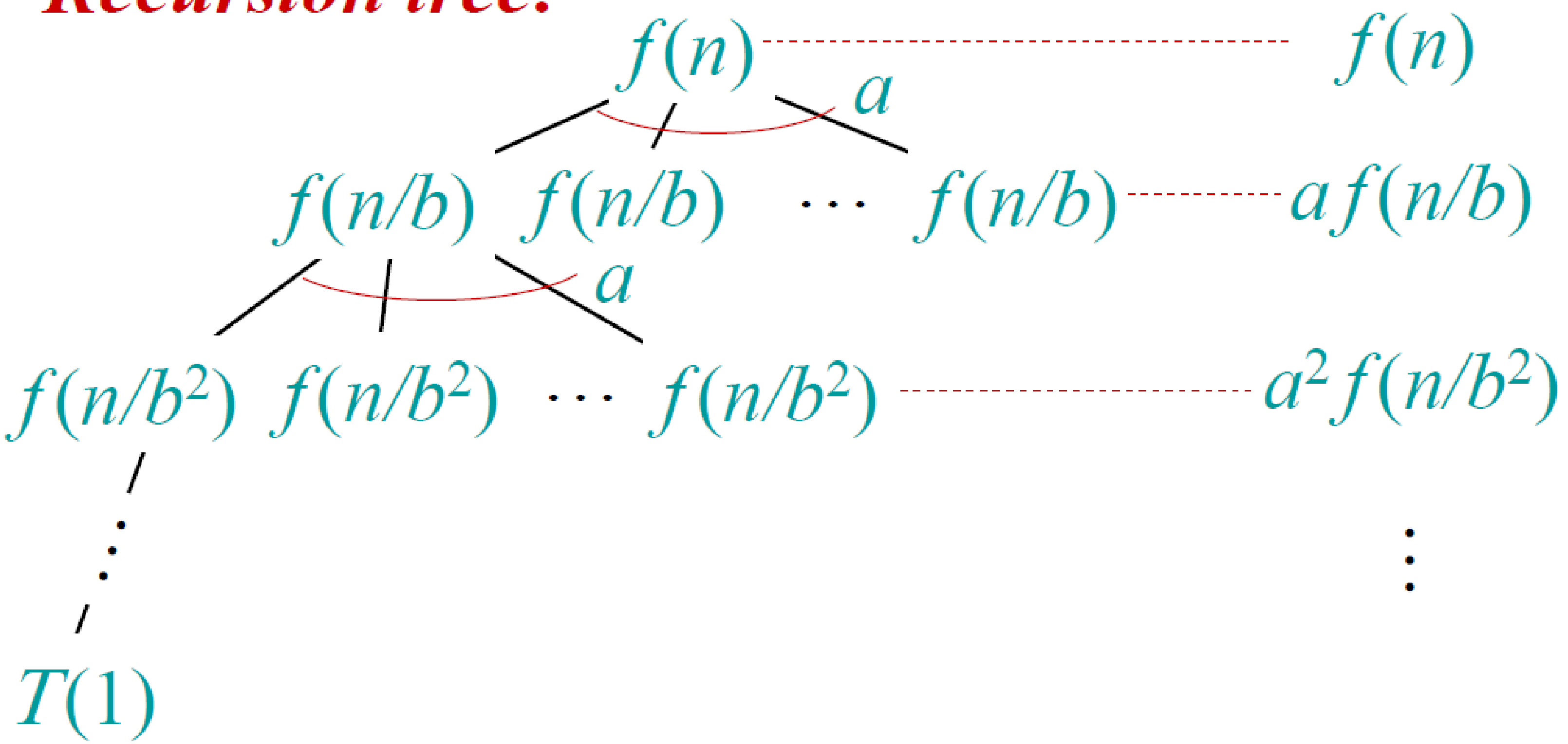
IDEA OF MASTER THEOREM

$T(n) = a\,T(n/b) + f(n)$,
where $a \geq 1, b > 1$,

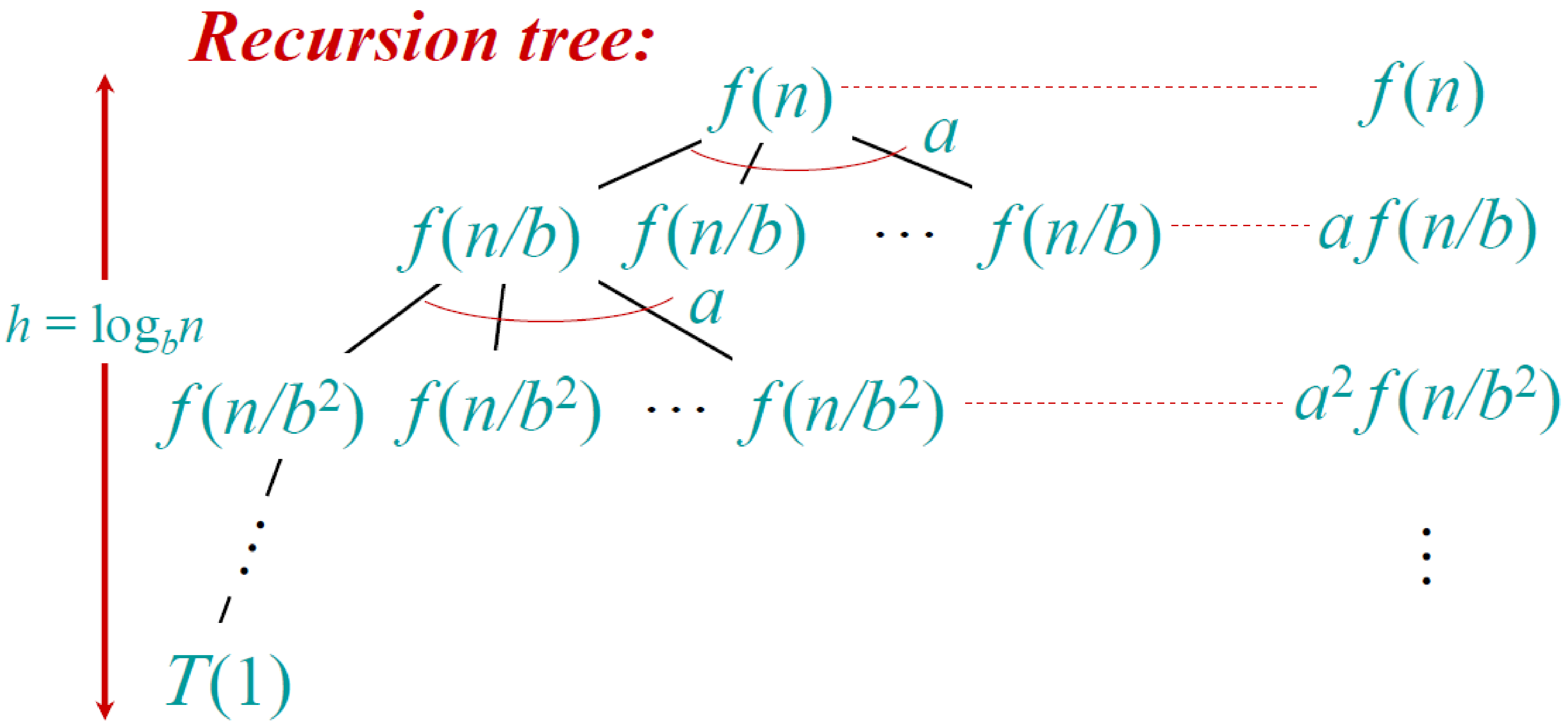
Recursion tree:



Recursion tree:

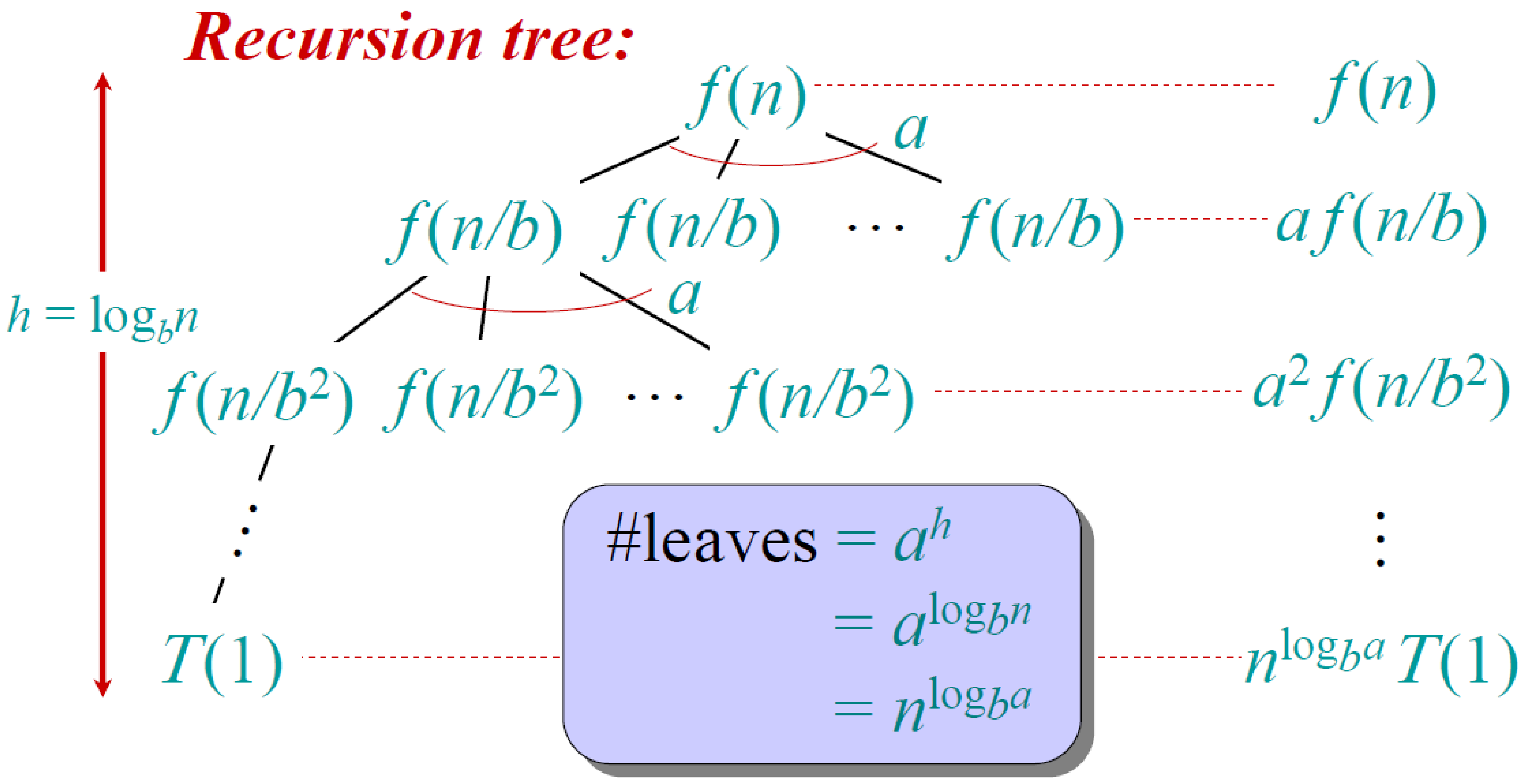


IDEA OF MASTER THEOREM



$$T\left(\frac{n}{b^k}\right) = T(1) \Rightarrow \frac{n}{b^k} = 1 \Rightarrow n = b^k \Rightarrow k = \log_b n$$

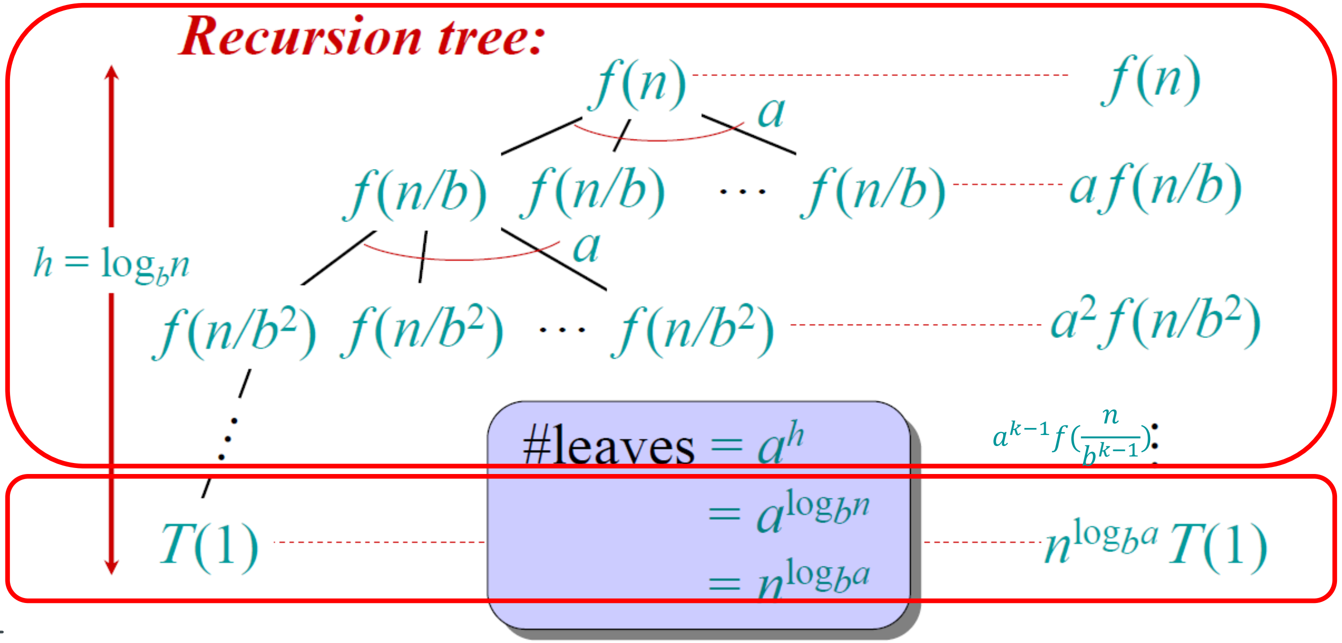
IDEA OF MASTER THEOREM



IDEA OF MASTER THEOREM

$I_c = Total\ cost\ of\ Internal\ Nodes$

$L_c = Total\ cost\ of\ Leaf\ Nodes$

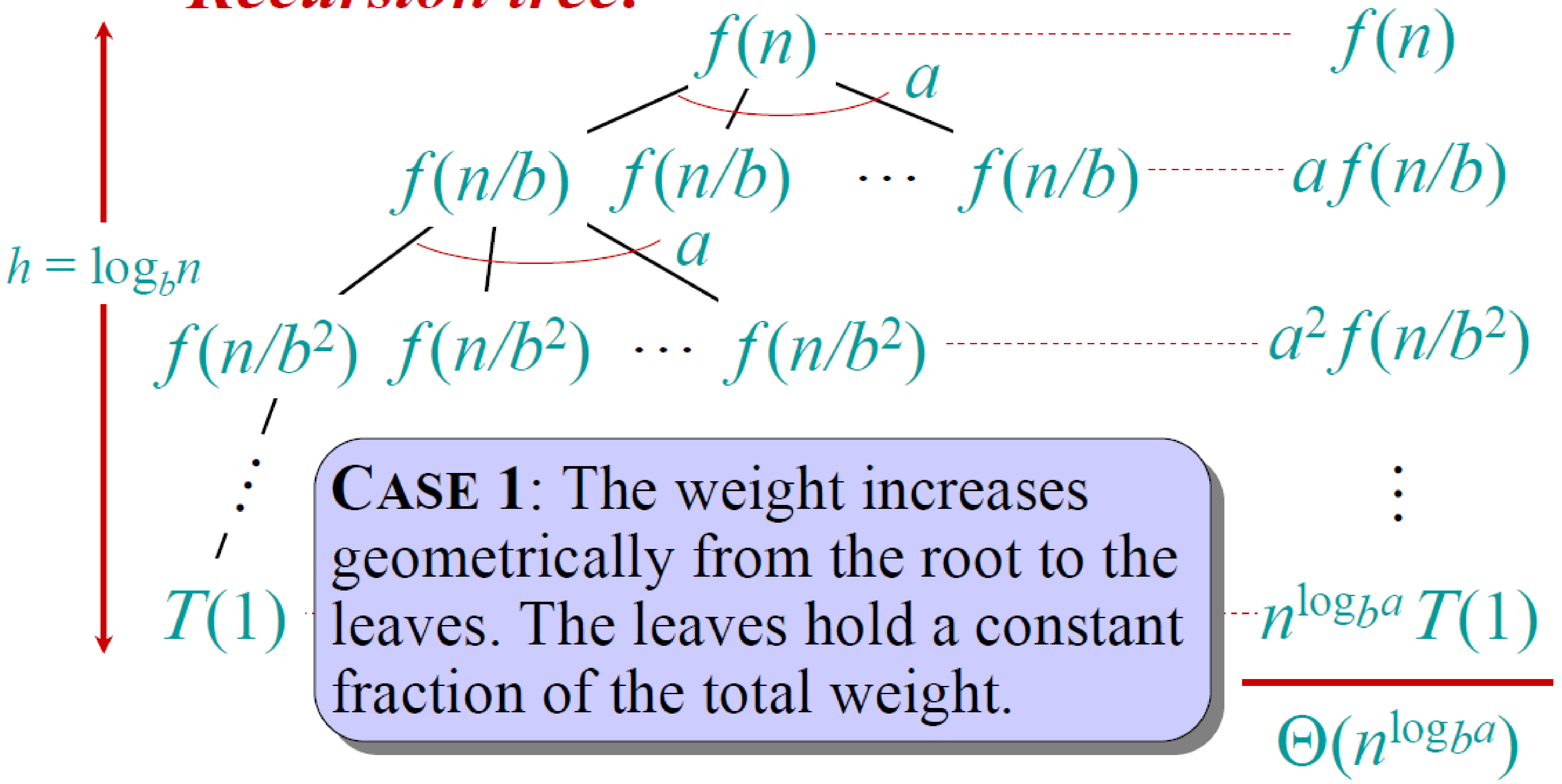


$$Total_c = I_c + L_c$$

$$= \left[f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + \cdots + a^{k-1} f\left(\frac{n}{b^{k-1}}\right) \right] + \left[n^{\log_b a}\right]$$

$$= \left[\sum_{k=0}^{\log_b n - 1} a^k f\left(\frac{n}{b^k}\right) \right] + \left[n^{\log_b a}\right]$$

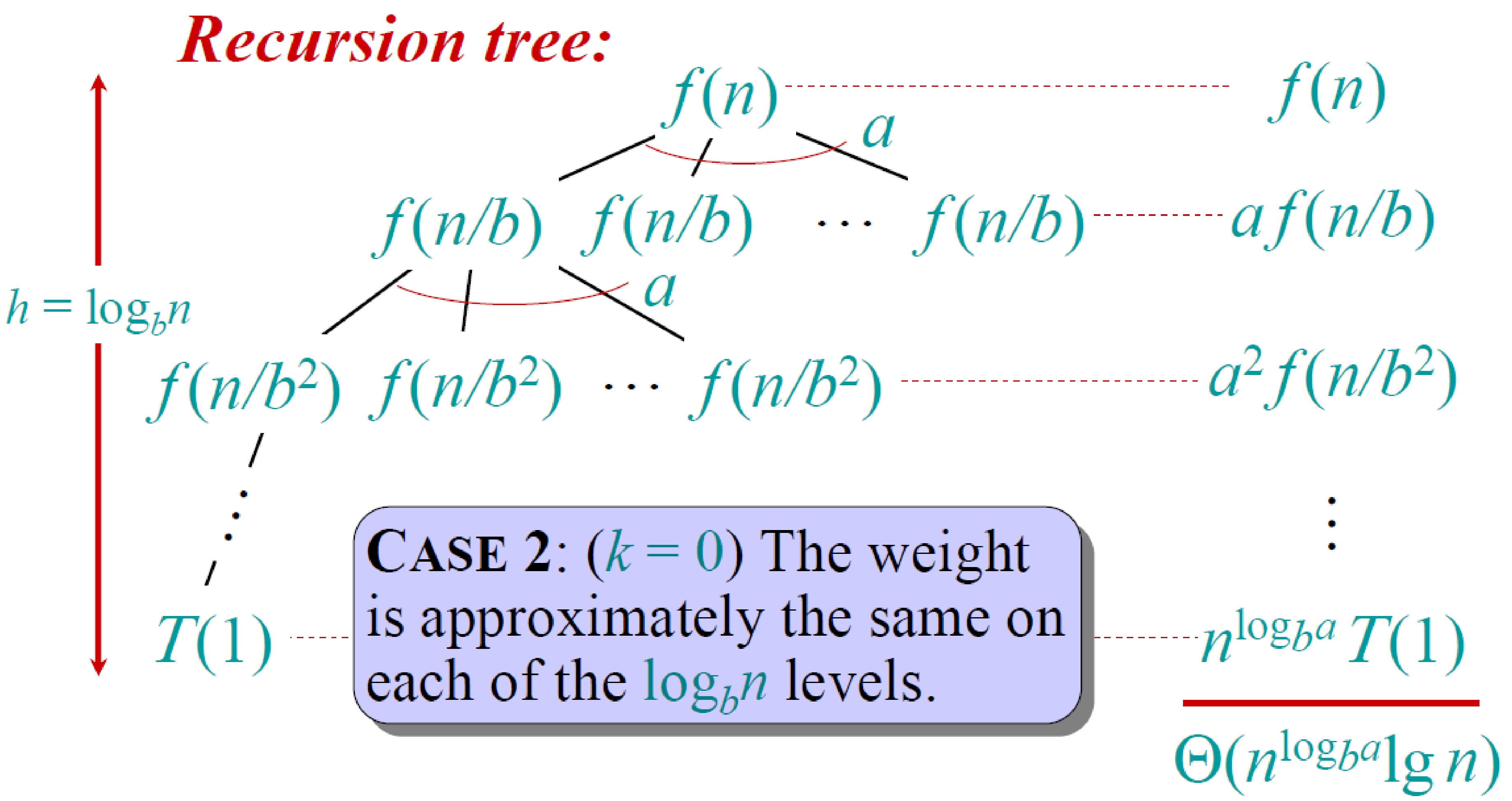
Recursion tree:



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

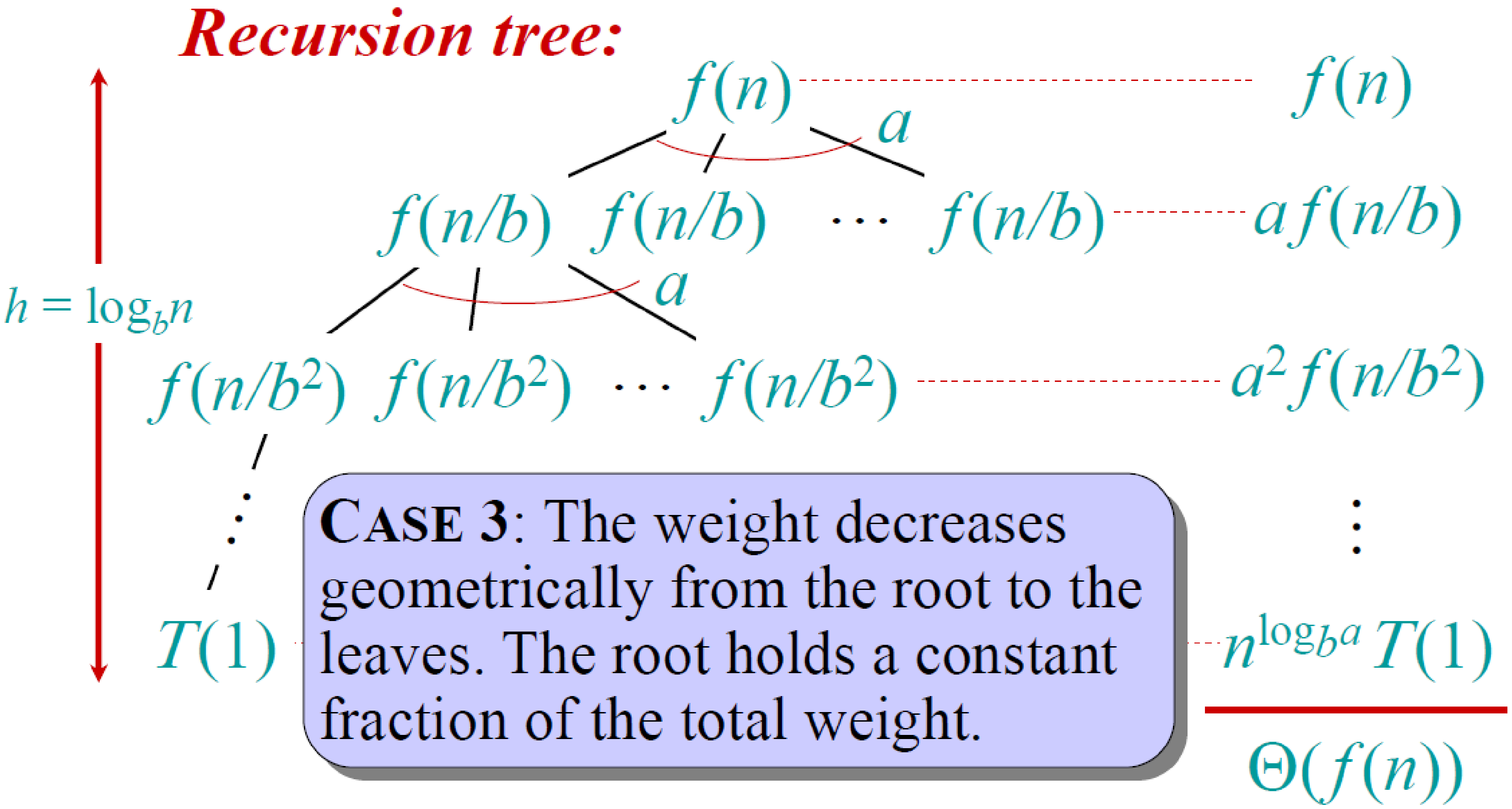
$$Total_c = \left[f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + \cdots + a^{k-1} f\left(\frac{n}{b^{k-1}}\right) \right] + \left[n^{\log_b a}\right]$$

IDEA OF MASTER THEOREM



$$Total_c = \left[f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + \cdots + a^{k-1} f\left(\frac{n}{b^{k-1}}\right) \right] + \left[n^{\log_b a} \right]$$

IDEA OF MASTER THEOREM



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

and $f(n)$ **satisfies the regularity condition**:

$a\, f\left(\frac{n}{b}\right) \leq c\, f(n)$ for some constant $c < 1\ \forall\, n$

$$Total_c = \left[f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + \cdots + a^{k-1} f\left(\frac{n}{b^{k-1}}\right) \right] + \left[n^{\log_b a} \right]$$

- 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., $T(n) = 2T(n/n) + 2^n$
  - b cannot be expressed as a constant, e.g., $T(n) = T(\sqrt{n})$
- Note that the Master Theorem does not solve the recurrence equation

- $T(n) = \begin{cases} C & if\ n = 0; \\ T(n-1) + C & if\ n > 0 \end{cases}$

$\Theta\ (n)$

- $T(n) = \begin{cases} C & if\ n = 0; \\ T(n-1) + n & if\ n > 0 \end{cases}$

$O(n^2)$

MORE RECURRENCE SOLVING

- $T(n) = \begin{cases} C & if\ n = 0; \\ T(n-1) + \log n & if\ n > 0 \end{cases}$

$O(nlogn)$

- $T(n) = \begin{cases} C & if\ n = 0; \\ 2T(n-1) + C & if\ n > 0 \end{cases}$

$O(2^n)$

MORE RECURRENCE SOLVING

- $T(n) = \begin{cases} C & if\ n = 1; \\ T(\frac{n}{2}) + C & if\ n > 1 \end{cases}$

$O(logn)$

- $T(n) = \begin{matrix} C & if\ n = 1; \\ T(\frac{n}{2}) + n & if\ n > 1 \end{matrix}$

$O(n)$

MORE RECURRENCE SOLVING

- $T(n) = \begin{cases} C & if\ n = 1; \\ 2T(\frac{n}{2}) + n & if\ n > 1 \end{cases}$

$O(nlogn)$


Q
A