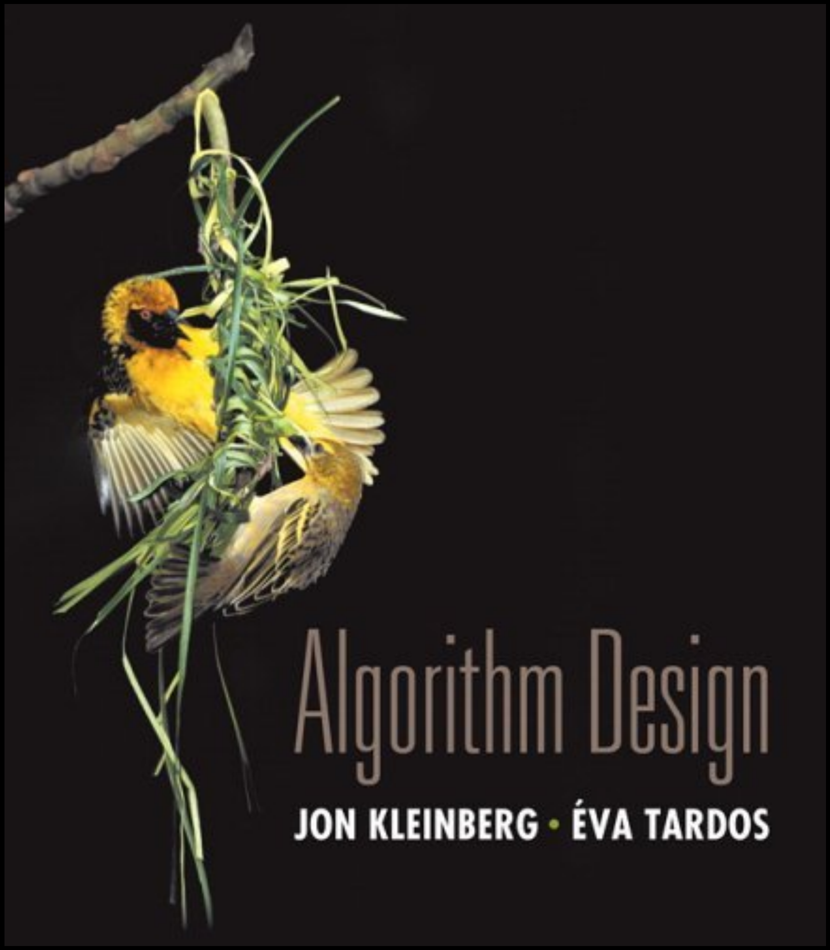
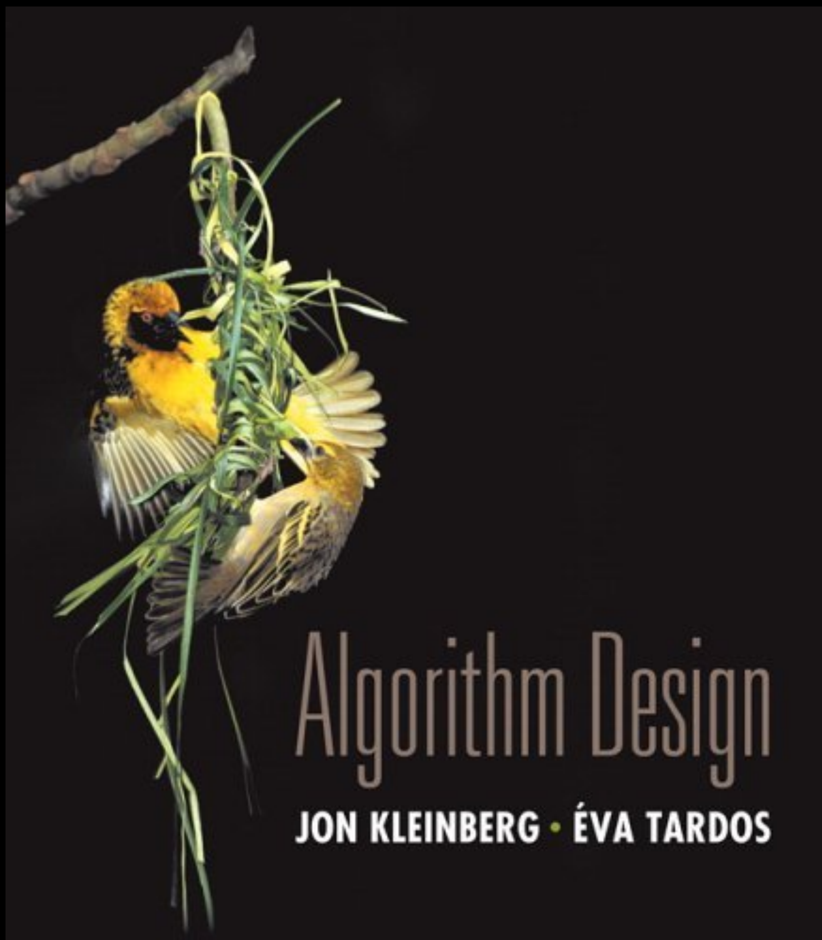


Chapter 5

Divide and Conquer



Chapter 5

Solving Recurrences

Solving Recurrences

- Recurrences are a major tool for analysis of algorithms.
  - Divide and Conquer algorithms are analyzable by recurrences.
- Also note, recurrence relations are used to design algorithms for combinatorial functions and can embody dynamic programming idea.

Methods for Solving Recurrences

- Using Substitution and Mathematical Induction
- Using Recursion-tree
- Using Master Theorem

Example: Integer Multiplication

- Let $X = \boxed{A \mid B}$ and $Y = \boxed{C \mid D}$ 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

$$T(n) < 4T(n/2) + 100n$$

How do we solve it?

Divide-and-Conquer Multiplication: Warmup

To multiply two n-bit integers a and b:

- Multiply four $\frac{1}{2}n$-bit integers, recursively.
- Add three pairs of $\frac{1}{2}n$-bit integers and shift to obtain result.

$$
\begin{aligned}
x &= 2^{n/2} \cdot x_1 + x_0 \\
y &= 2^{n/2} \cdot y_1 + y_0 \\
xy &= \left(2^{n/2} \cdot x_1 + x_0\right)\left(2^{n/2} \cdot y_1 + y_0\right) = 2^n \cdot x_1 y_1 + 2^{n/2} \cdot \left(x_1 y_0 + x_0 y_1\right) + x_0 y_0
\end{aligned}
$$

Ex. $\quad a = \underbrace{1000}_{x_1}\underbrace{1101}_{x_0} \qquad b = \underbrace{1110}_{y_1}\underbrace{0001}_{y_0}$

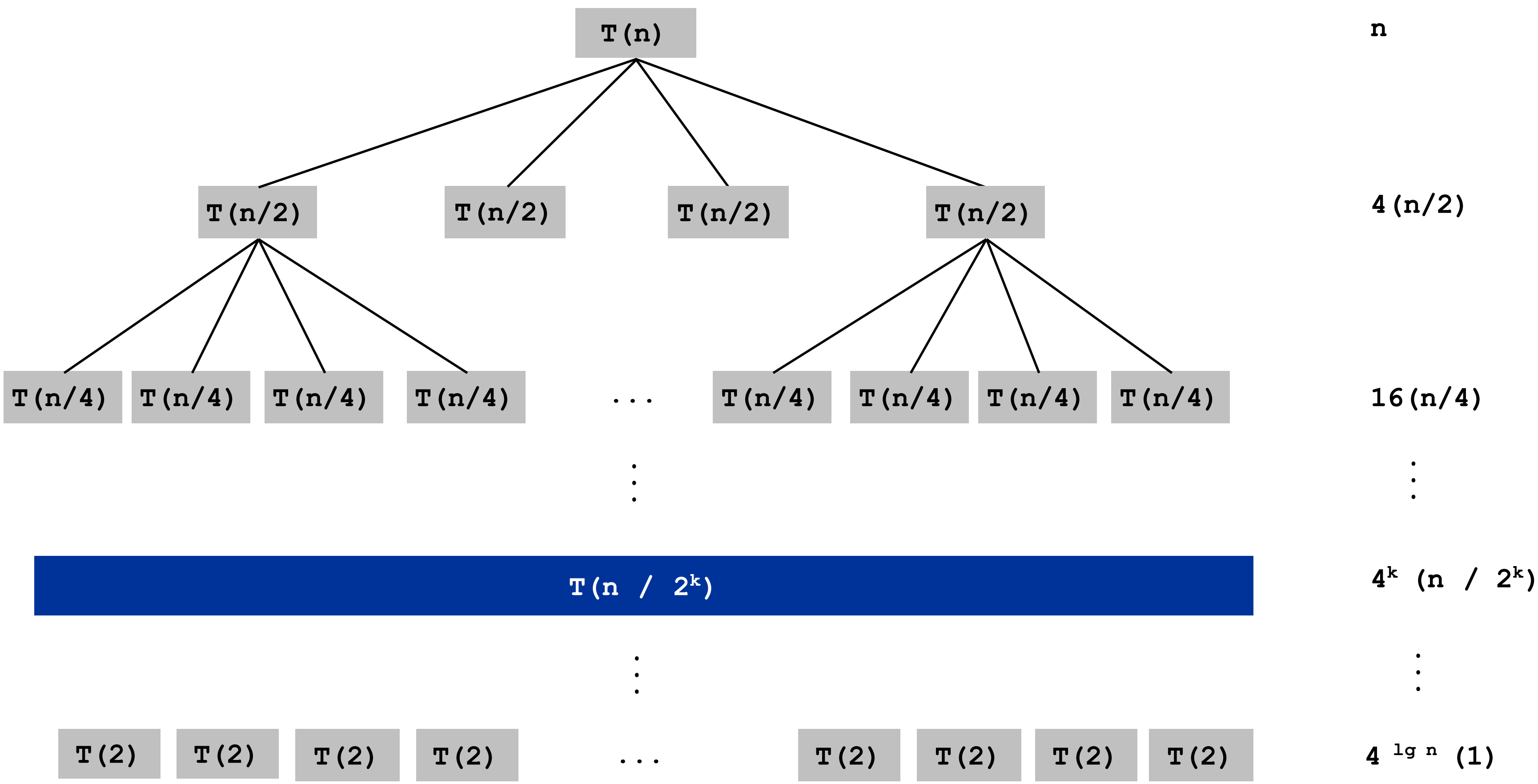
$$
\mathrm{T}(n) = \underbrace{4T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n)}_{\text{add, shift}} \Rightarrow \mathrm{T}(n) = \Theta(n^2)
$$

assumes n is a power of 2

Recursion Tree

$$T(n) = \begin{cases} 0 & \text{if } n = 0 \\ 4T(n/2) + n & \text{otherwise} \end{cases}$$

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



Substitution method

The most general method:

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

Example: $T(n) = 4T(n/2) + 100n$

- [Assume that $T(1) = \Theta(1)$.]
- Guess $O(n^3)$. (Prove O and Ω separately.)
- Assume that $T(k) \leq ck^3$ for $k < n$.
- Prove $T(n) \leq cn^3$ by induction.

Example of substitution

$$
\begin{aligned}
T(n) &= 4T(n/2) + 100n \\
&\le 4c(n/2)^3 + 100n \\
&= (c/2)n^3 + 100n \\
&= cn^3 - ((c/2)n^3 - 100n) \quad \leftarrow \textit{desired} - \textit{residual} \\
&\le cn^3 \quad \leftarrow \textit{desired}
\end{aligned}
$$

whenever $(c/2)n^3 - 100n \ge 0$, for example, if $c \ge 200$ and $n \ge 1$.

residual

Example (continued)

- 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 $1 \le n < n_0$, we have "$\Theta(1)$" $\le cn^3$, if we pick c big enough.

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 $k < n$:

$$\begin{aligned} T(n) &= 4T(n/2) + 100n \\ &\leq cn^2 + 100n \\ &\leq cn^2 \end{aligned}$$

for ***no*** choice of $c > 0$. Lose!

A tighter upper bound!

IDEA: Strengthen the inductive hypothesis.

- ***Subtract*** a low-order term.

Inductive hypothesis: $T(k) \le c_1 k^2 - c_2 k$ for $k < n$.

$$
\begin{aligned}
T(n) &= 4T(n/2) + 100n \\
&\le 4(c_1(n/2)^2 - c_2(n/2)) + 100n \\
&= c_1 n^2 - 2c_2 n + 100n \\
&= c_1 n^2 - c_2 n - (c_2 n - 100n) \\
&\le c_1 n^2 - c_2 n \quad \text{if } c_2 > 100.
\end{aligned}
$$

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 (…).
- The recursion-tree method promotes intuition, however.

itle]

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

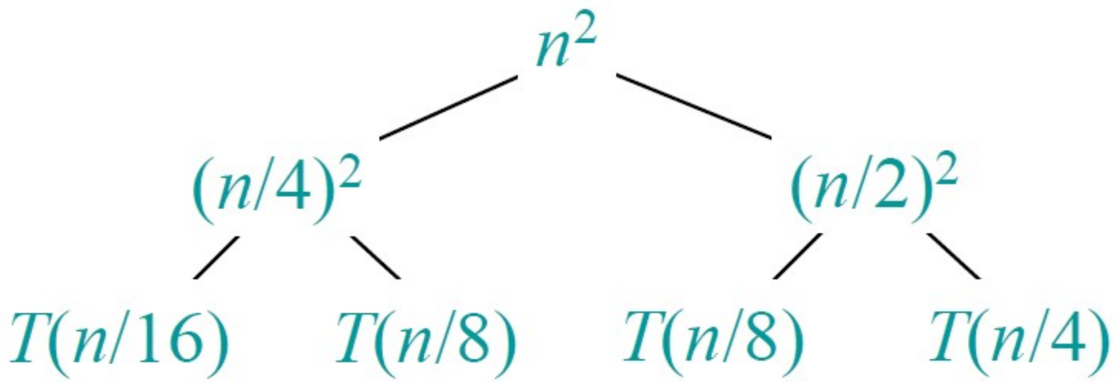
Solve $T(n) = T(n/4) + T(n/2) + n^2$:

$$n^2$$

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

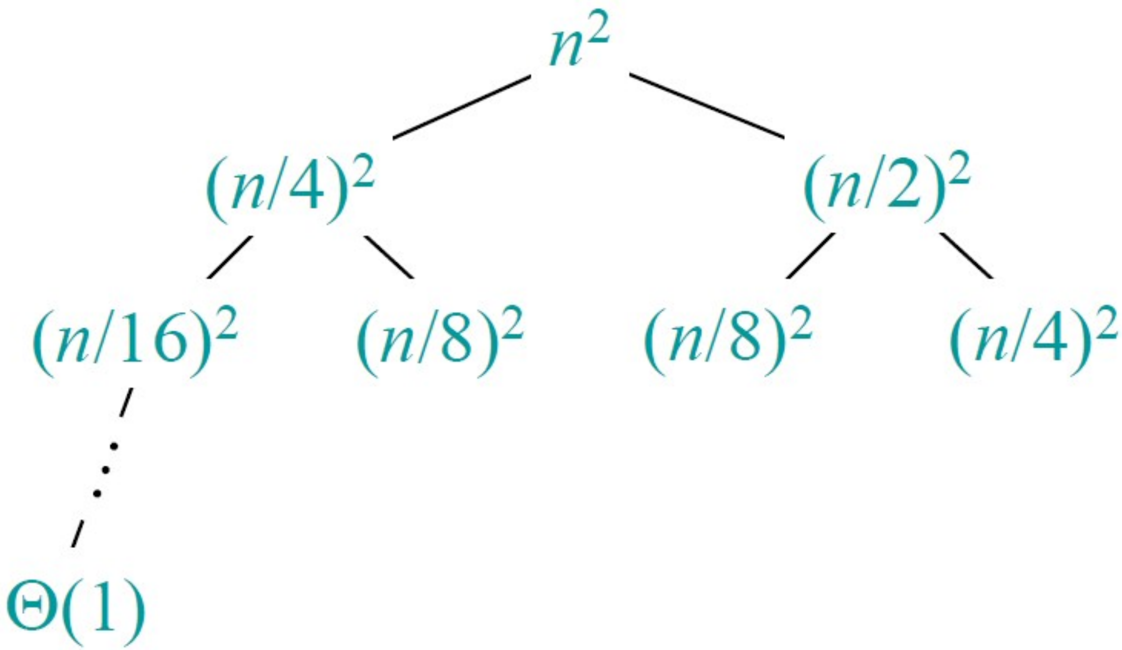
Example of recursion tree

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



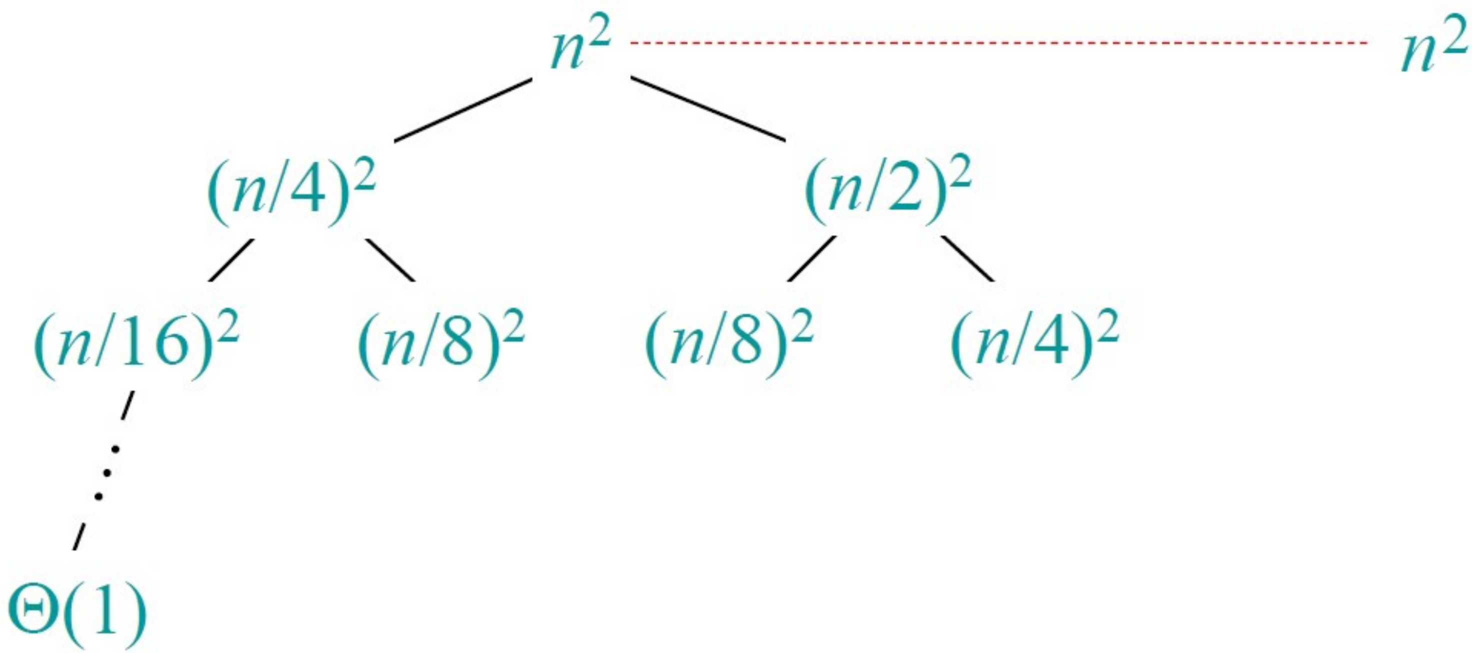
Example of recursion tree

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



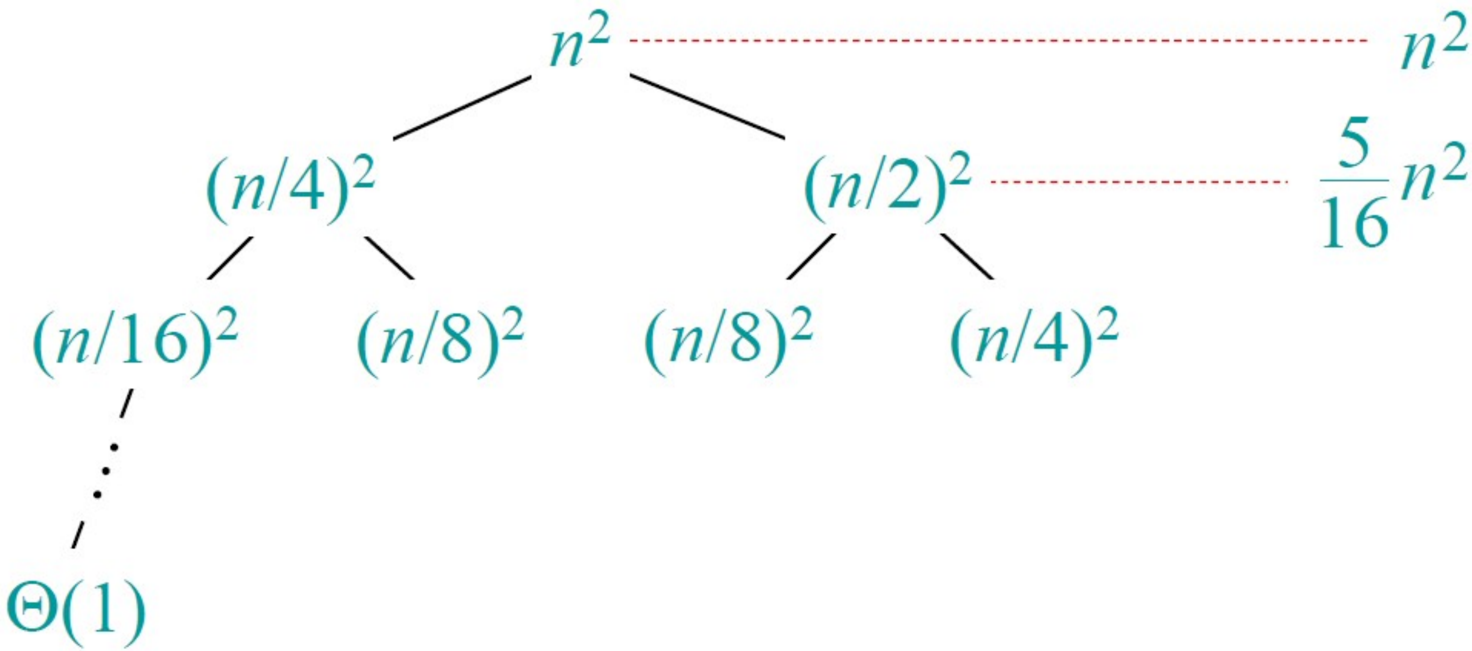
Example of recursion tree

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



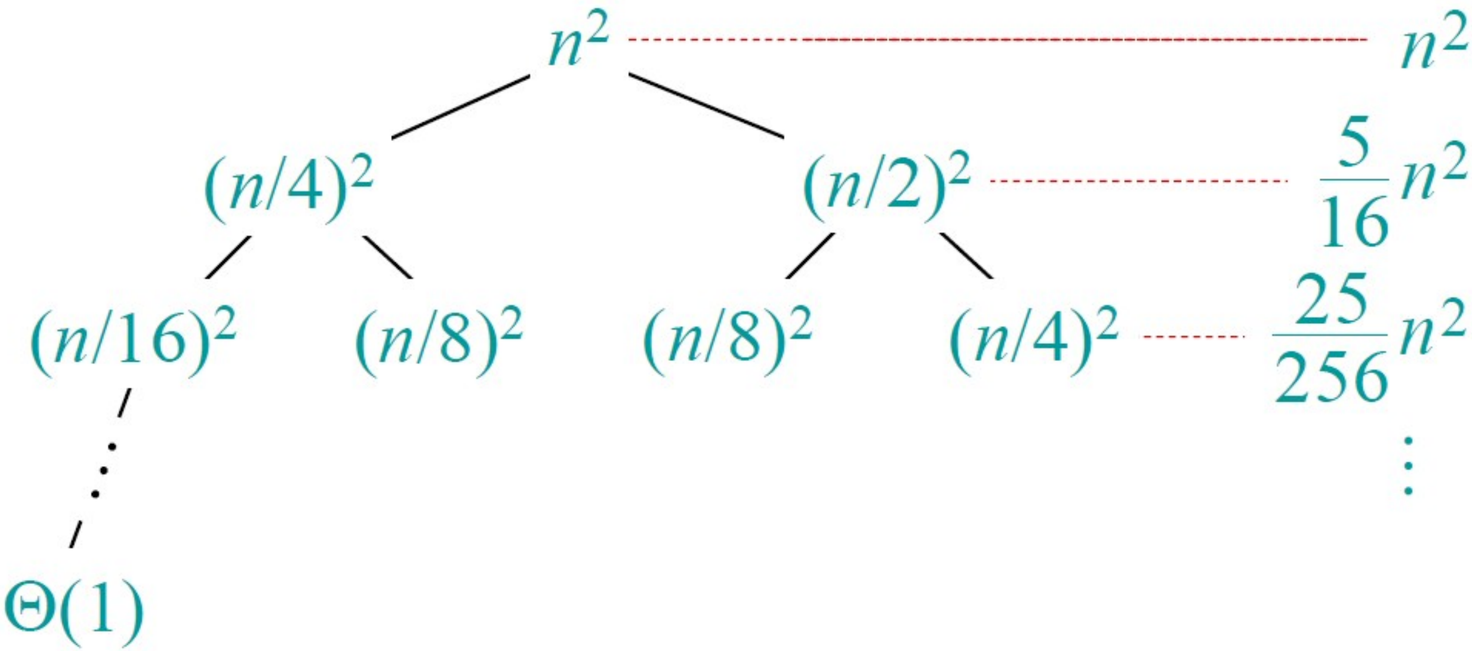
Example of recursion tree

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



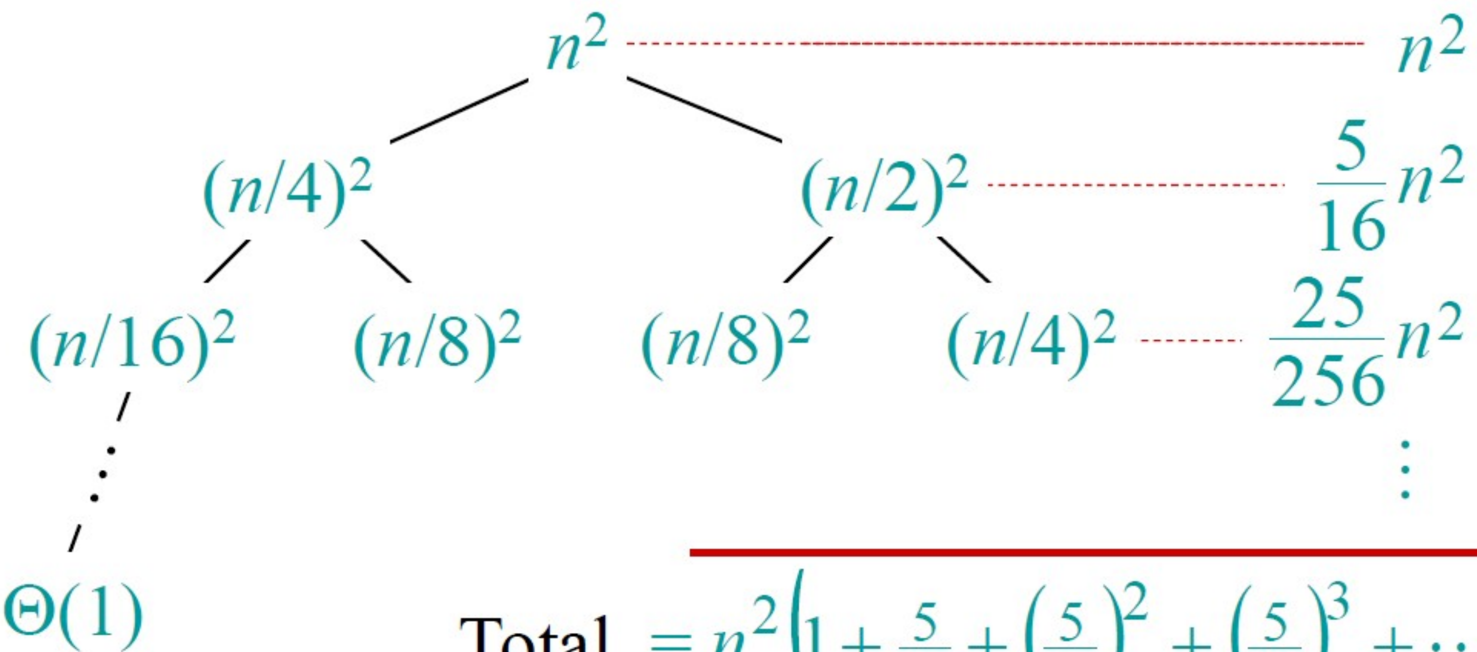
Example of recursion tree

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



Example of recursion tree

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



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

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

Appendix: geometric series

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

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

Recursion Tree of $T(n)=T(n/3)+T(2n/3)+O(n)$

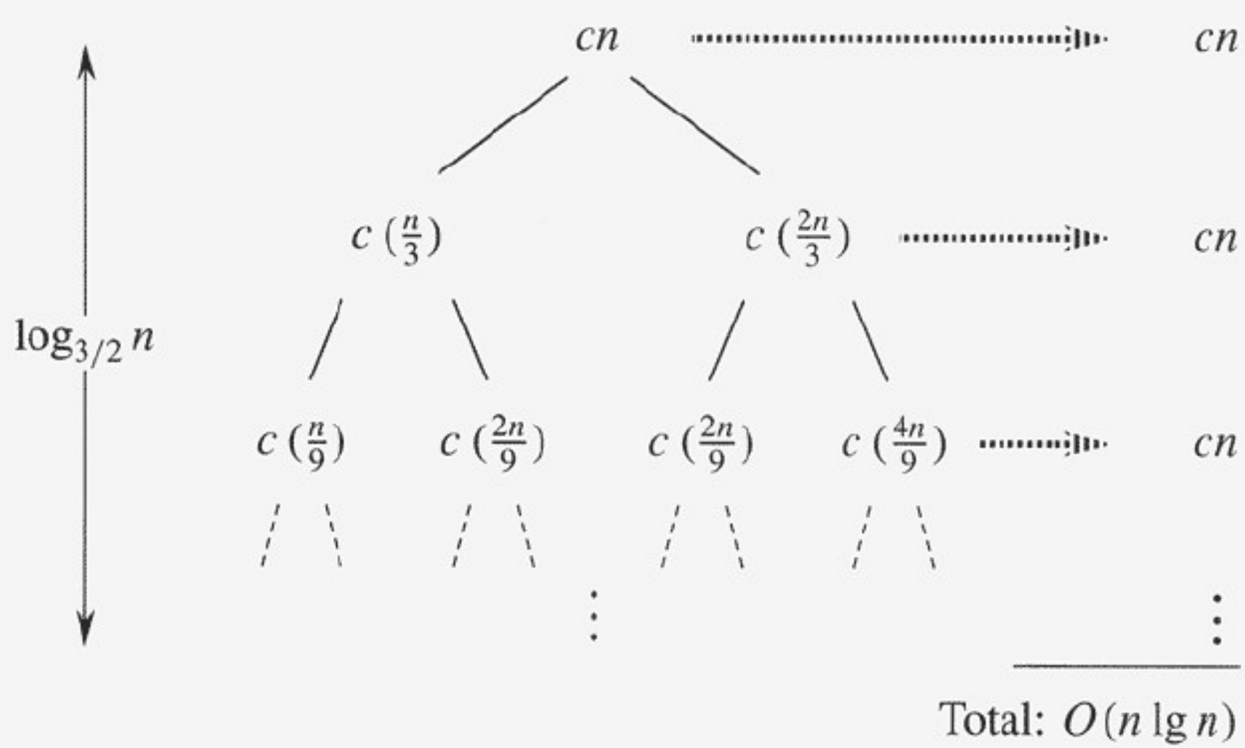


Figure 4.2 A recursion tree for the recurrence $T(n) = T(n/3) + T(2n/3) + cn$.

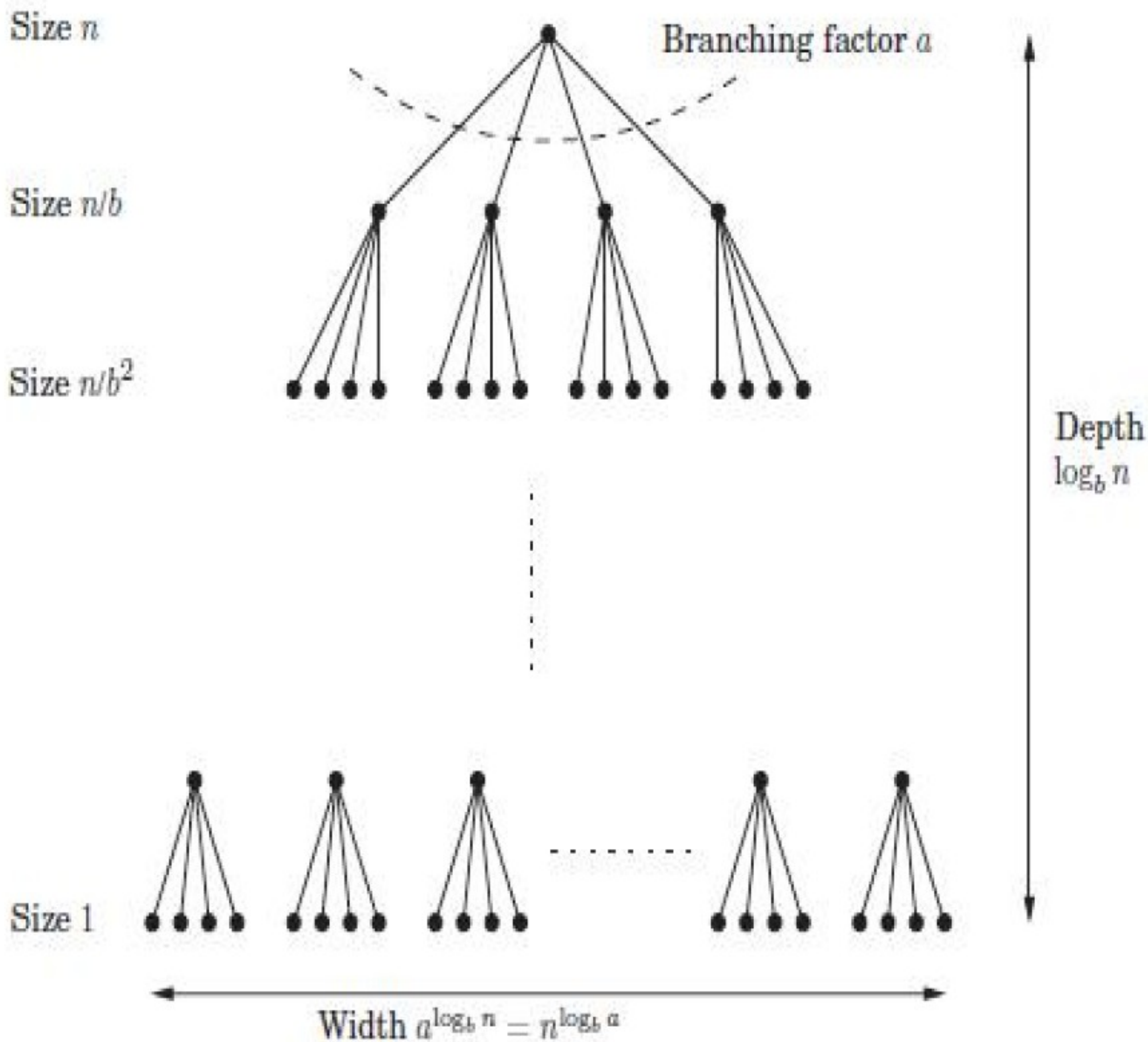
Master Theorem

- Let T(n) be <u>a monotonically increasing</u> function that satisfies

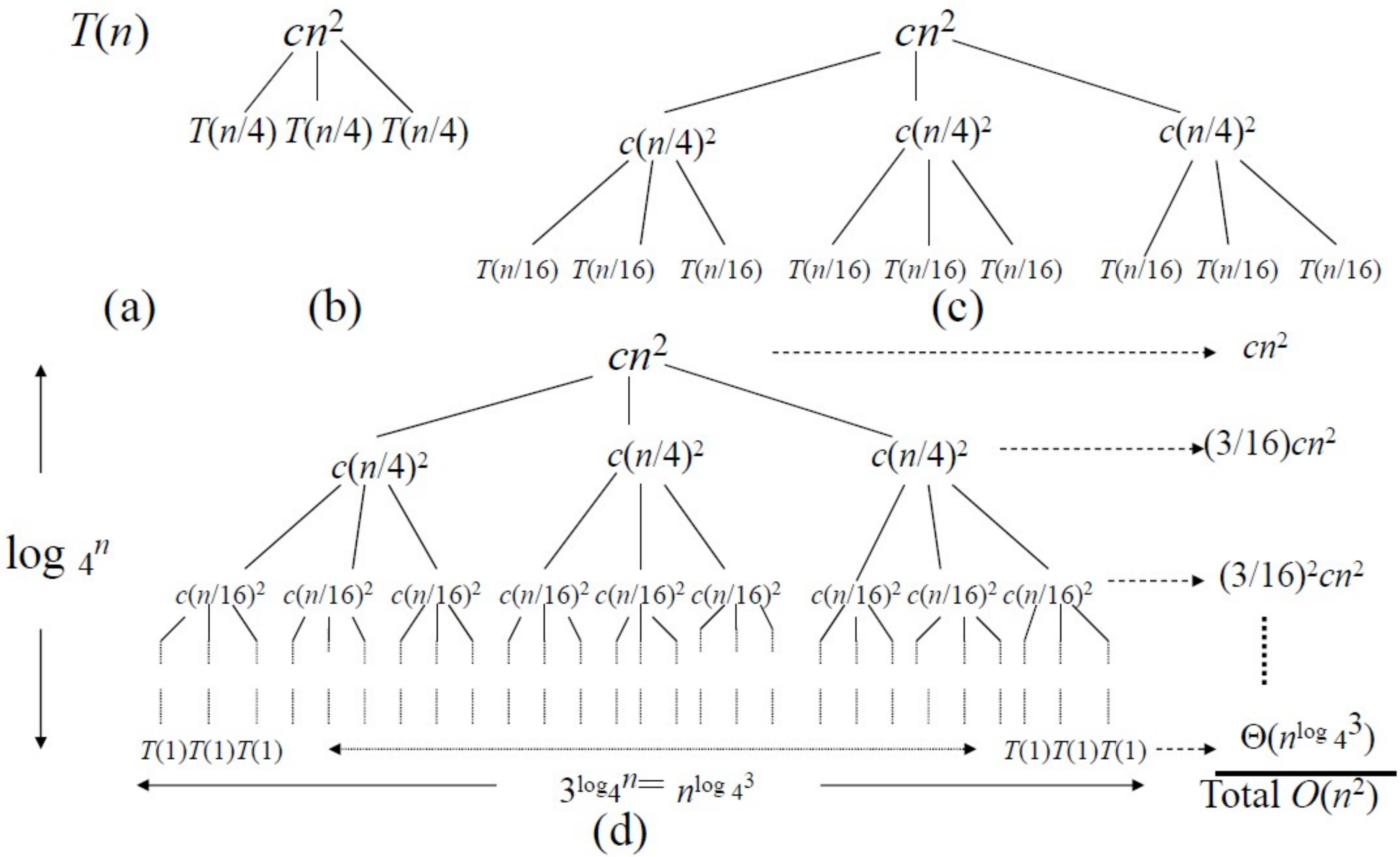
$$T(n) = a\,T(n/b) + f(n)$$
$$T(1) = c$$

where $a \geq 1$, $b \geq 2$, $c>0$. If $f(n)$ is $\Theta(n^d)$ where $d \geq 0$ then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \text{ or } a < b^d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \text{ or } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \text{ or } a > b^d \end{cases}$$


Size n
Branching factor a
Size n/b
Size n/b^2
Depth $\log_b n$
Size 1
Width $a^{\log_b n} = n^{\log_b a}$

Recursion Tree for $T(n)=3T(\lfloor n/4 \rfloor)+\Theta(n^2)$



Karatsuba Multiplication

To multiply two n-bit integers a and b:

- Add two pairs of $\frac{1}{2}n$ bit integers.
- Multiply three pairs of $\frac{1}{2}n$-bit integers, recursively.
- Add two pairs, subtract two pairs, and shift two n-bit integers to obtain result.

$$
\begin{aligned}
a &= 2^{n/2} \cdot a_1 + a_0 \\
b &= 2^{n/2} \cdot b_1 + b_0 \\
ab &= 2^{n} \cdot a_1 b_1 + 2^{n/2} \cdot \left(a_1 b_0 + a_0 b_1\right) + a_0 b_0 \\
&= 2^{n} \cdot \underbrace{a_1 b_1}_{1} + 2^{n/2} \cdot \left(\underbrace{(a_1 + a_0)(b_1 + b_0)}_{2} - \underbrace{a_1 b_1}_{1} - \underbrace{a_0 b_0}_{3}\right) + \underbrace{a_0 b_0}_{3}
\end{aligned}
$$

Karatsuba Multiplication

To multiply two n-bit integers a and b:

- Add **two pairs** of $\frac{1}{2}n$ bit integers.
- Multiply **three pairs** of $\frac{1}{2}n$-bit integers, recursively.
- Add two pairs, subtract two pairs, and shift two n-bit integers to obtain result.

$$
\begin{aligned}
a &= 2^{n/2} \cdot a_1 + a_0 \\
b &= 2^{n/2} \cdot b_1 + b_0 \\
ab &= 2^n \cdot a_1 b_1 + 2^{n/2} \cdot \left(a_1 b_0 + a_0 b_1\right) + a_0 b_0 \\
&= 2^n \cdot \underset{1}{a_1 b_1} + 2^{n/2} \cdot \left(\underset{2}{(a_1 + a_0)(b_1 + b_0)} - \underset{1}{a_1 b_1} - \underset{3}{a_0 b_0}\right) + \underset{3}{a_0 b_0}
\end{aligned}
$$

- Theorem. [Karatsuba-Ofman 1962] Can multiply two n-bit integers in $O(n^{1.585})$ bit operations.

$$
T(n) \le \underbrace{T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(1 + \lceil n/2 \rceil)}_{\text{recursive calls}} + \underbrace{\Theta(n)}_{\text{add, subtract, shift}} \Rightarrow T(n) = O(n^{\lg 3}) = O(n^{1.585})
$$

Solution to $T(n)=3T(\lfloor n/4 \rfloor)+\Theta(n^2)$

- The height is $\log_4 n$,
- #leaf nodes $= 3^{\log_4 n} = n^{\log_4 3}$. Leaf node cost: $T(1)$.
- Total cost $T(n)=cn^2+(3/16)\,cn^2+(3/16)^2\,cn^2+$

$$\cdots +(3/16)^{\log_4 n-1}\,cn^2+\Theta(n^{\log_4 3})$$

$$= (1+3/16+(3/16)^2+\cdots+(3/16)^{\log_4 n-1})\,cn^2+\Theta(n^{\log_4 3})$$

$$< (1+3/16+(3/16)^2+\cdots+(3/16)^m+\cdots)\,cn^2+\Theta(n^{\log_4 3})$$

$$= (1/(1-3/16))\,cn^2+\Theta(n^{\log_4 3})$$

$$= 16/13cn^2+\Theta(n^{\log_4 3})$$

$$= O(n^2).$$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \text{ or } a < b^d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \text{ or } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \text{ or } a > b^d \end{cases}$$

Solution to $T(n)=3T(\lfloor n/4 \rfloor)+\Theta(n)$

- The height is $\log_4 n$,
- #leaf nodes = $3^{\log_4 n} = n^{\log_4 3}$. Leaf node cost: $T(1)$.
- Total cost $T(n)=cn+(3/16)\,cn+(3/16)^2\,cn+$
  $\cdots +(3/16)^{\log_4 n-1}\,cn+\Theta(n^{\log_4 3})$
  $= (1+3/16+(3/16)^2+\cdots+(3/16)^{\log_4 n-1})\,cn+\Theta(n^{\log_4 3})$
  $< (1+3/16+(3/16)^2+\cdots+(3/16)^m+\cdots)\,cn+\Theta(n^{\log_4 3})$
  $= (1/(1-3/16))\,cn+\Theta(n^{\log_4 3})$
  $= 16/13cn+\Theta(n^{\log_4 3})$
  $= \Theta(n^{\log_4 3})$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \text{ or } a < b^d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \text{ or } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \text{ or } a > b^d \end{cases}$$

Solution to $T(n)=2T(\lfloor n/4 \rfloor)+\Theta(n)$

- The height is $\log_4 n$,
- #leaf nodes $= 2^{\log_4 n} = n^{\log_4 2}$. Leaf node cost: $T(1)$.
- Total cost $T(n)=cn+(2/16)\,cn+(2/16)^2\,cn+ \cdots +(2/16)^{\log_4 n-1}\,cn+\Theta(n^{\log_4 3})$

  $= (1+2/16+(2/16)^2+ \cdots +(2/16)^{\log_4 n-1})\,cn + \Theta(n^{\log_4 2})$

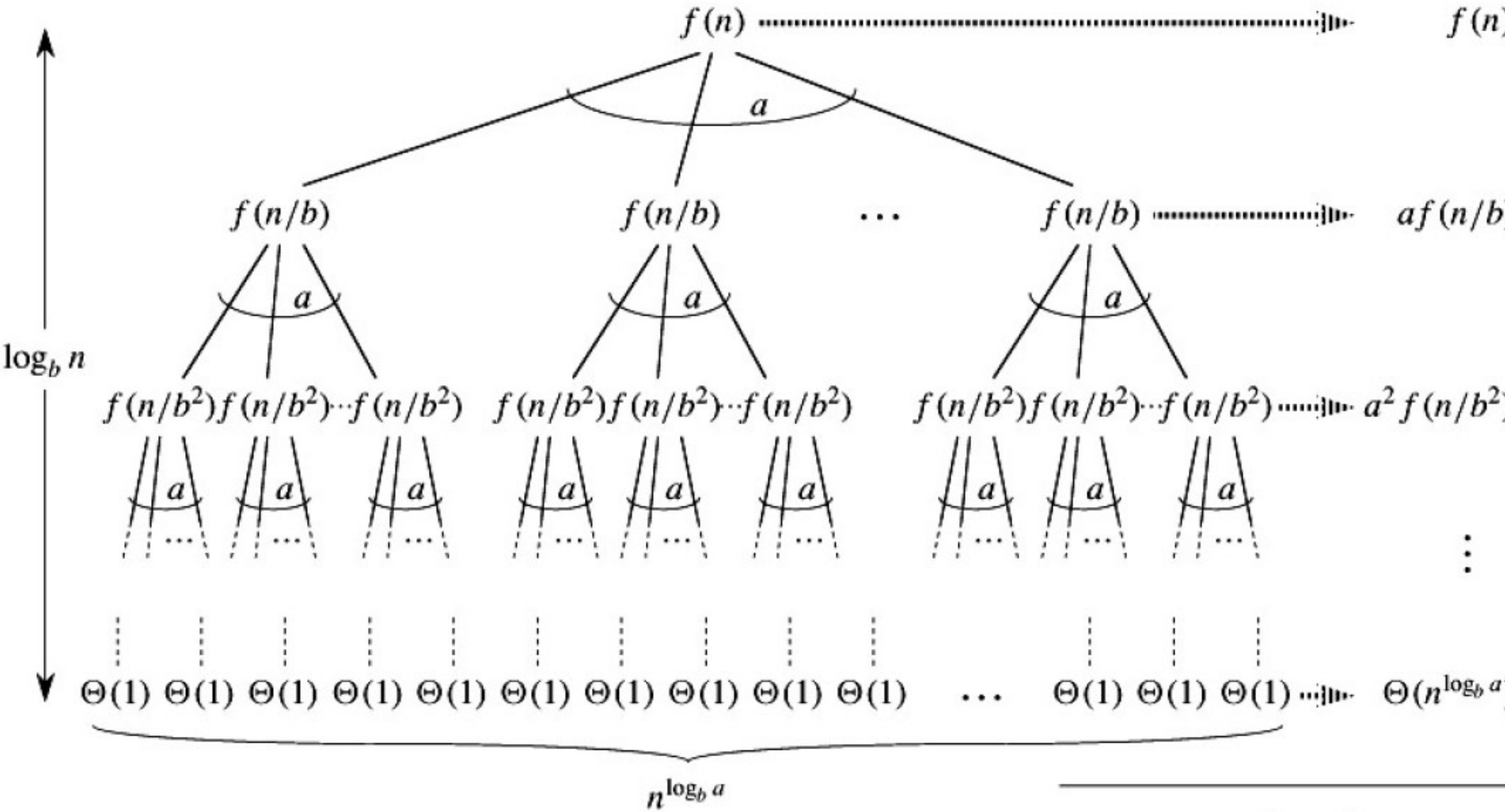
  $< (1+2/16+(2/16)^2+ \cdots +(2/16)^{m}+ \cdots)\,cn + \Theta(n^{\log_4 2})$

  $= (1/(1-2/16))\,cn + \Theta(n^{\log_4 2})$

  $= 16/14cn + \Theta(n^{\log_4 2})$

  $= O(n)$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \text{ or } a < b^d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \text{ or } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \text{ or } a > b^d \end{cases}$$



Total: $\Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$

Basis problems cost

Divide and combine cost

Master Theorem: Pitfalls

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

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

Master Theorem: Example 1

- Let $T(n) = T(n/2) + ½ n^2 + n$. What are the parameters?

  a = 1

  b = 2

  d = 2

  Therefore, which condition applies?

  $0 < 2$ or $1 < 2^2$, case 1 applies (if $\log_b a < d$ or $a < b^d$)

- We conclude that

$$T(n) \in \Theta(n^d) = \Theta(n^2)$$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \text{ or } a < b^d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \text{ or } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \text{ or } a > b^d \end{cases}$$

Master Theorem: Example 2

- Let T(n)= 2 T(n/4) + √n + 42. What are the parameters?

  a = 2

  b = 4

  d = 1/2

  Therefore, which condition applies?

  ½ = ½ or $2 = 4^{1/2}$, case 2 applies (if $\log_b a = d$ or $a = b^d$)

- We conclude that

$$T(n) \in \Theta(n^d \log n) = \Theta(\log n \sqrt{n})$$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \text{ or } a < b^d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \text{ or } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \text{ or } a > b^d \end{cases}$$

Master Theorem: Example 3

- Let T(n)= 3 T(n/2) + 3/4n + 1. What are the parameters?

  a = 3

  b = 2

  d = 1

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \text{ or } a < b^d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \text{ or } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \text{ or } a > b^d \end{cases}$$

  Therefore, which condition applies?

  $\log_2 3 > 1$ or $3 > 2^1$, case 3 applies (if $\log_b a > d$ or $a > b^d$)

- We conclude that

$$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3})$$

- Note that $\log_2 3 \approx 1.584$..., can we say that $T(n) \in \Theta(n^{1.584})$

  No, because $\log_2 3 \approx 1.5849$... and $n^{1.584} \notin \Theta(n^{1.5849})$

‘Fourth’ Condition

- Recall that we cannot use the Master Theorem if f(n), the non-recursive cost, is not a polynomial.
- There is a limited 4th condition of the Master Theorem that allows us to consider polylogarithmic functions.
- **Corollary**: If $f(n) \in \Theta(n^{\log_b a} \log^k n)$ for some k≥0 then

$$T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$$

- This final condition is fairly limited and we present it for sake of completeness.

‘Fourth’ Condition: Example

- Say we have the following recurrence relation

$$T(n)= 2\,T(n/2) + n \log n$$

- Clearly, a=2, b=2, but f(n) is not a polynomial. However, we have $f(n) \in \Theta(n \log n)$, k=1
- Therefore, by the 4th condition of the Master Theorem, we can say that

$$T(n) \in \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(n^{\log_2 2} \log^2 n) = \Theta(n \log^2 n)$$

The master method

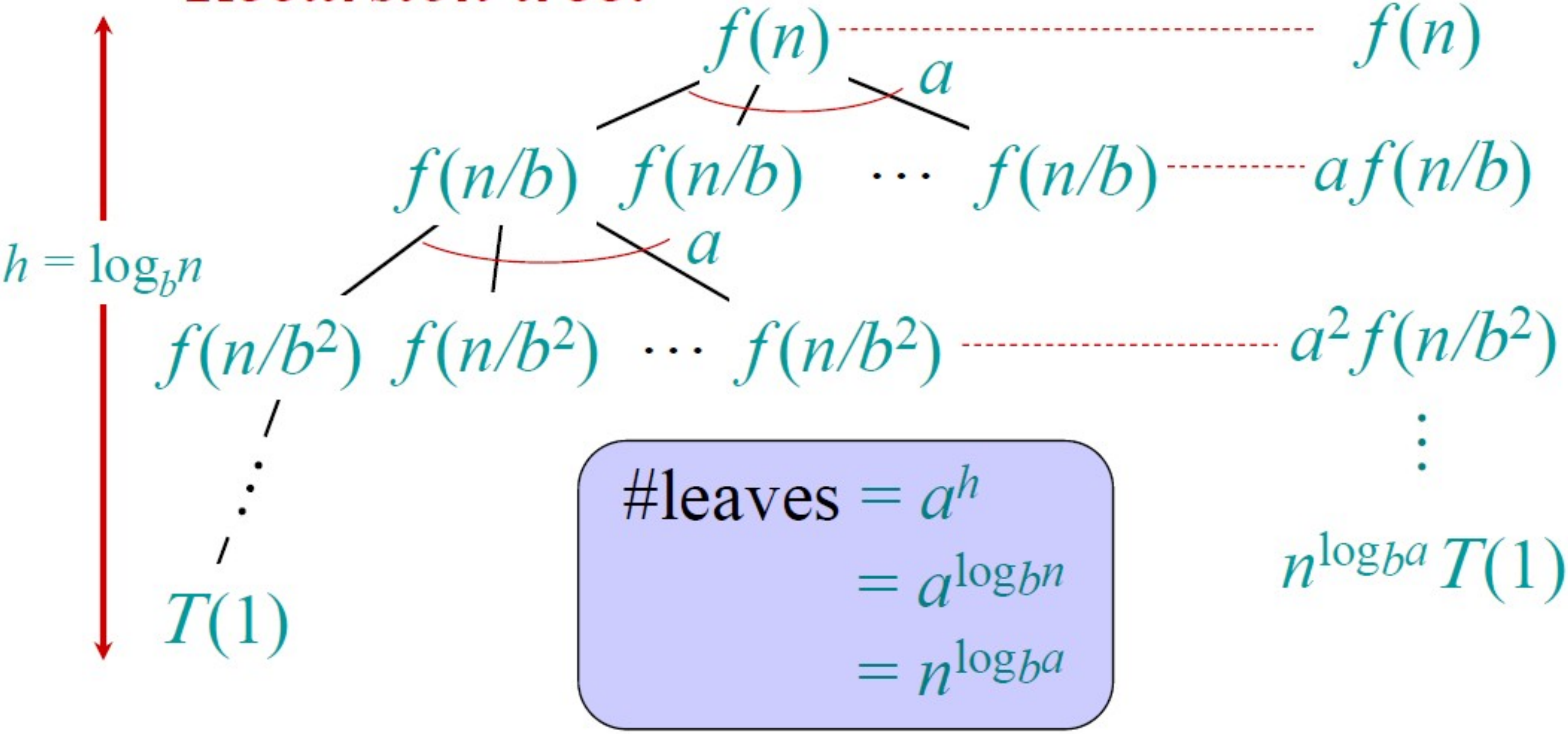
The master method applies to recurrences of the form

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

where $a \geq 1$, $b > 1$, and f is asymptotically positive.

Idea of master theorem

Recursion tree:



Three common cases

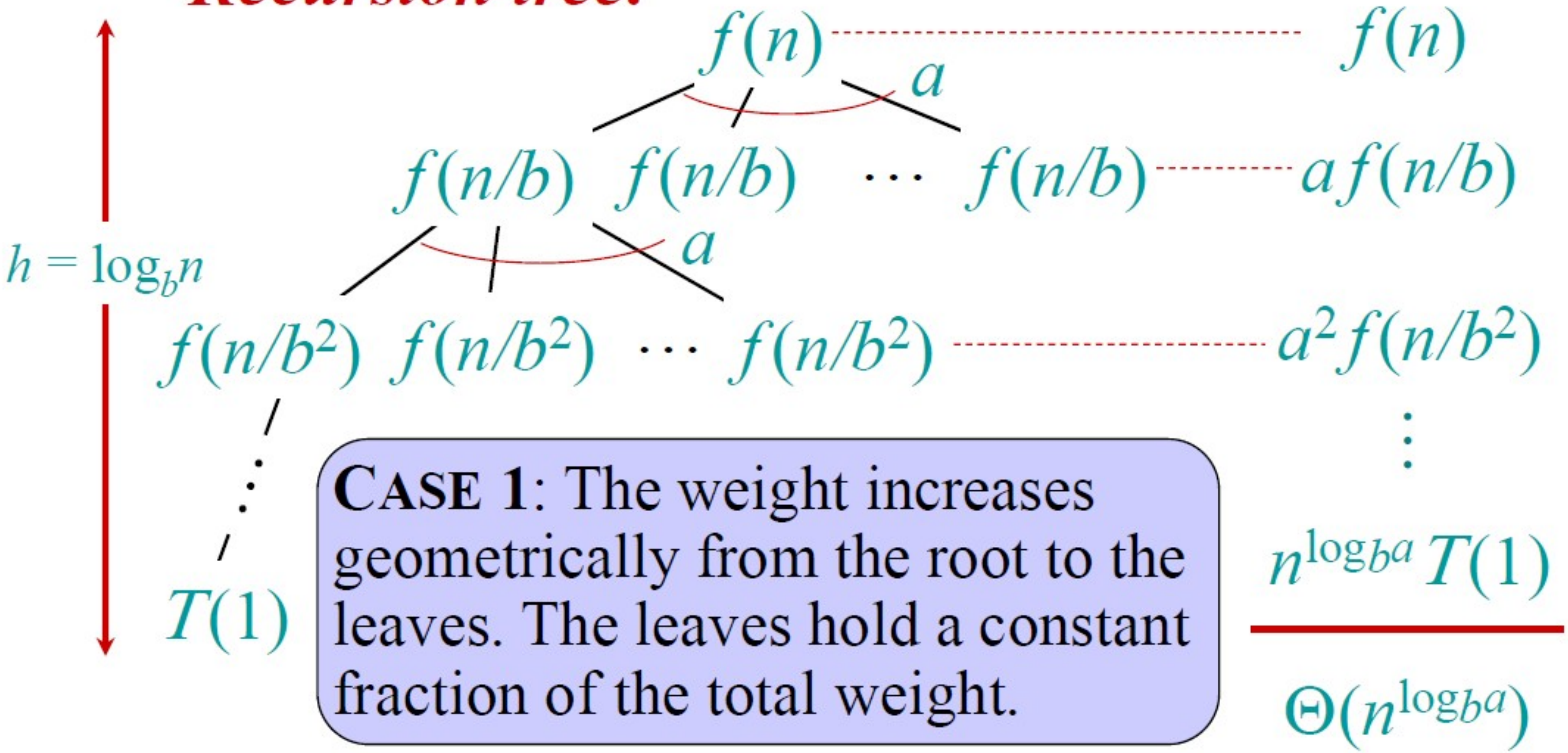
Compare $f(n)$ with $n^{\log_b a}$:

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

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

Idea of master theorem

Recursion tree:



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

Three common cases

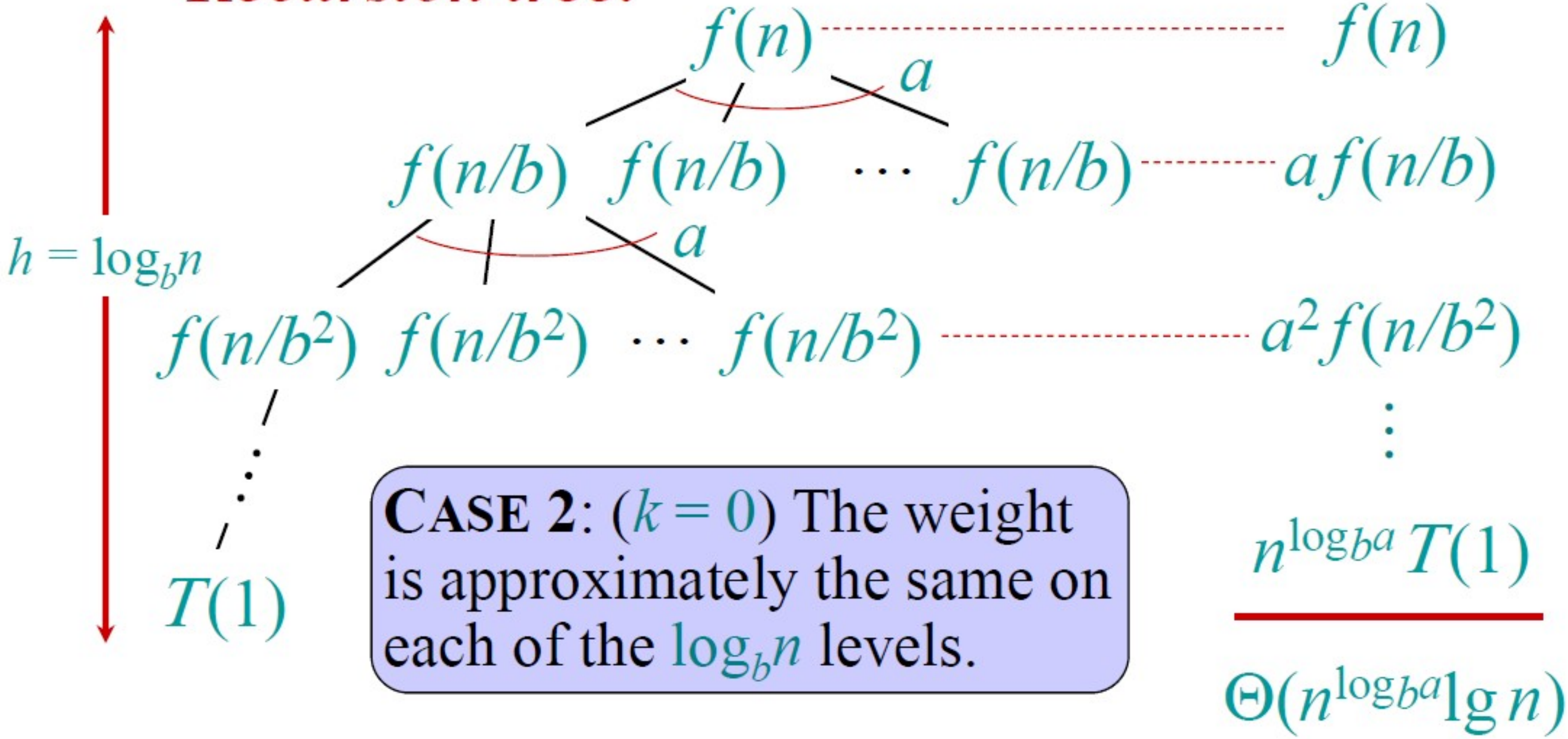
Compare $f(n)$ with $n^{\log_b a}$:

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

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

Idea of master theorem

Recursion tree:



Three common cases (cont.)

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

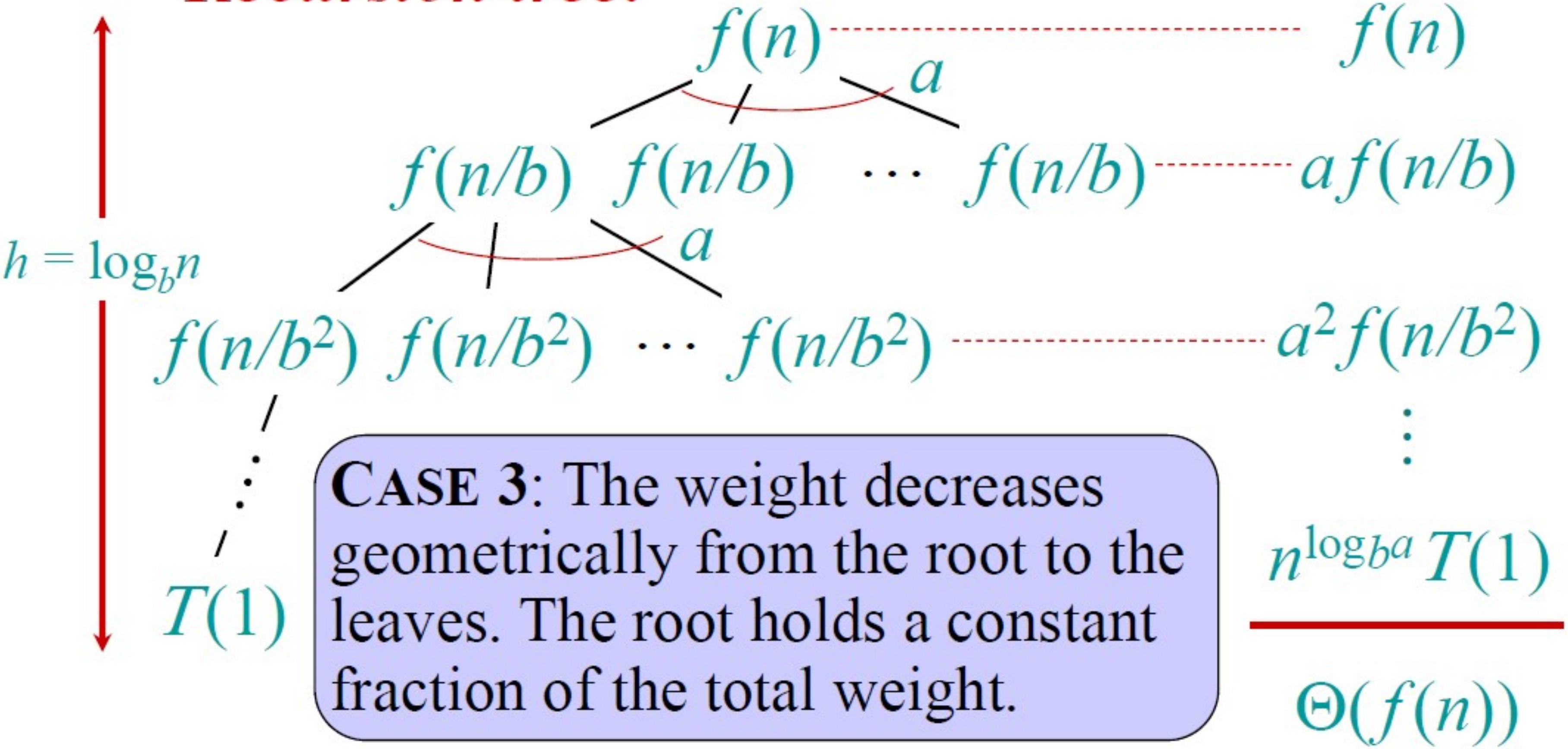
3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially faster than $n^{\log_b a}$ (by an n^{ε} factor),

   and $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \le c f(n)$ for some constant $c < 1$.

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

Idea of master theorem

Recursion tree:



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

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

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

Examples

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

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

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

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

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

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

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

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

Changing Variables

- Suppose $T(n)=2T(\sqrt{n})+\lg n$.
- Rename $m=\lg n$.   So $T(2^m)=2T(2^{m/2})+m$.
- Domain transformation:

  $S(m)=T(2^m)$

  $S(m)=2S(m/2)+m$.
- So the solution is $S(m)=O(m \lg m)$.
- Changing back to $T(n)$ from $S(m)$, the solution is

  $$\begin{aligned} T(n) &= T(2^m)=S(m) \\ &= O(m \lg m) \\ &= O(\lg n \lg \lg n). \end{aligned}$$