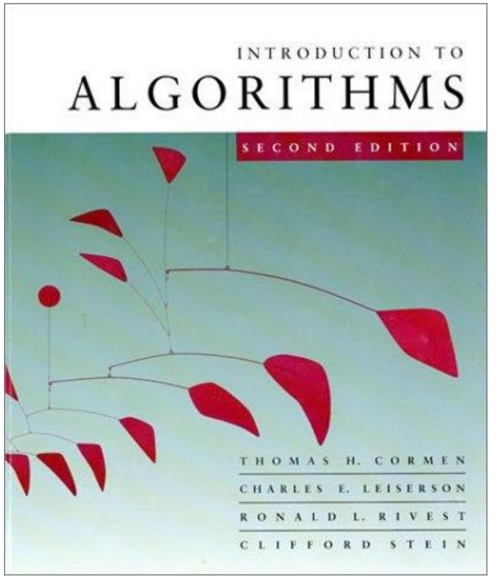
Introduction to Algorithms



Recurrence Relations

Recurrence Relations

Recurrences

- The expression:

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

  is a *recurrence*.
  - Recurrence: an equation that describes a function in terms of its value on smaller functions

Recurrence Examples

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

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

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

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

Solving Recurrences

- Substitution method
- Iteration method
- Master method

Solving Recurrences

- The substitution method (CLR 4.1)
  - "Making a good guess" method
  - Guess the form of the answer, then use induction to find the constants and show that solution works
  - Examples:
    - $T(n) = 2T(n/2) + \Theta(n)$ □ $T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor) + n$ □ ???

Solving Recurrences

- The substitution method (CLR 4.1)
  - "Making a good guess" method
  - Guess the form of the answer, then use induction to find the constants and show that solution works
  - Examples:
    - $T(n) = 2T(n/2) + \Theta(n) \rightarrow T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor) + n \rightarrow T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor)+ 17) + n \rightarrow$???

Substitution method

- *Guess the form of the solution .*
- *Use mathematical induction to find the constants and show that the solution works .*

The substitution method can be used to establish either upper or lower bounds on a recurrence.

An example (Substitution method)

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

We guess that the solution is $T(n)=0(n \lg n)$.

i.e. to show that $T(n) \leq cn \lg n$, for some constant $c > 0$ and $n \geq m$.

Assume that this bound holds for [n/2]. So , we get

$$
\begin{aligned}
T(n) &\leq 2(c\, floor(n/2) \lg(floor(n/2))) + n \\
&\leq cn \lg(n/2) + n \\
&= cn \lg n - cn \lg 2 + n \\
&= cn \lg n - cn + n \\
&\leq cn \lg n
\end{aligned}
$$

where , the last step holds as long as $c \geq 1$.

- *<u>Boundary conditions :</u>*

Suppose , $T(1)=1$ is the sole boundary condition of the recurrence .

then , for $n=1$, the bound $T(n)\leq c\, n \lg n$ yields

$T(1)\leq c \lg 1=0$, which is at odds with $T(1)=1$.

Thus ,the base case of our inductive proof fails to hold.

To overcome this difficulty , we can take advantage of the asymptotic notation which only requires us to prove

$T(n)\leq c\, n \lg n$ for $n\geq m$.

The idea is to remove the difficult boundary condition $T(1)= 1$ from consideration.

Thus , <u>we can replace $T(1)$ by $T(2)$ as the base cases</u> in the inductive proof , letting $m=2$.

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

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

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.

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

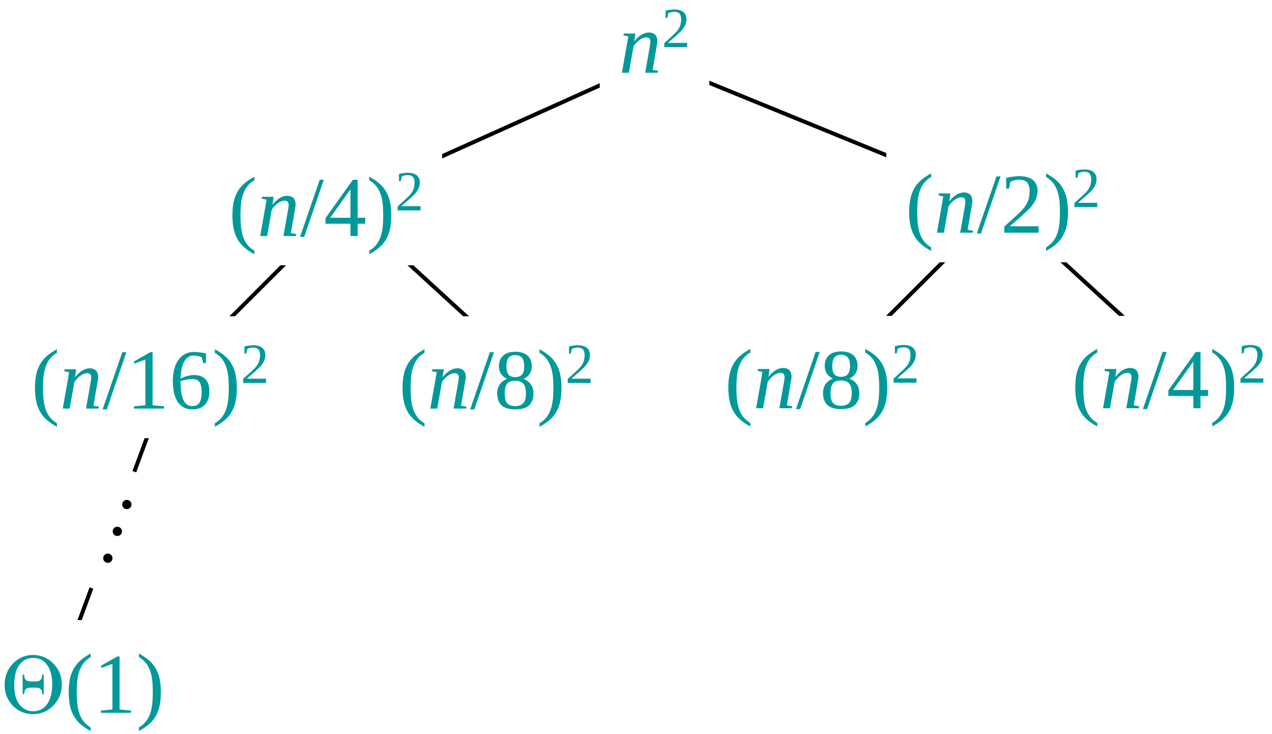
$$n^2$$

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

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

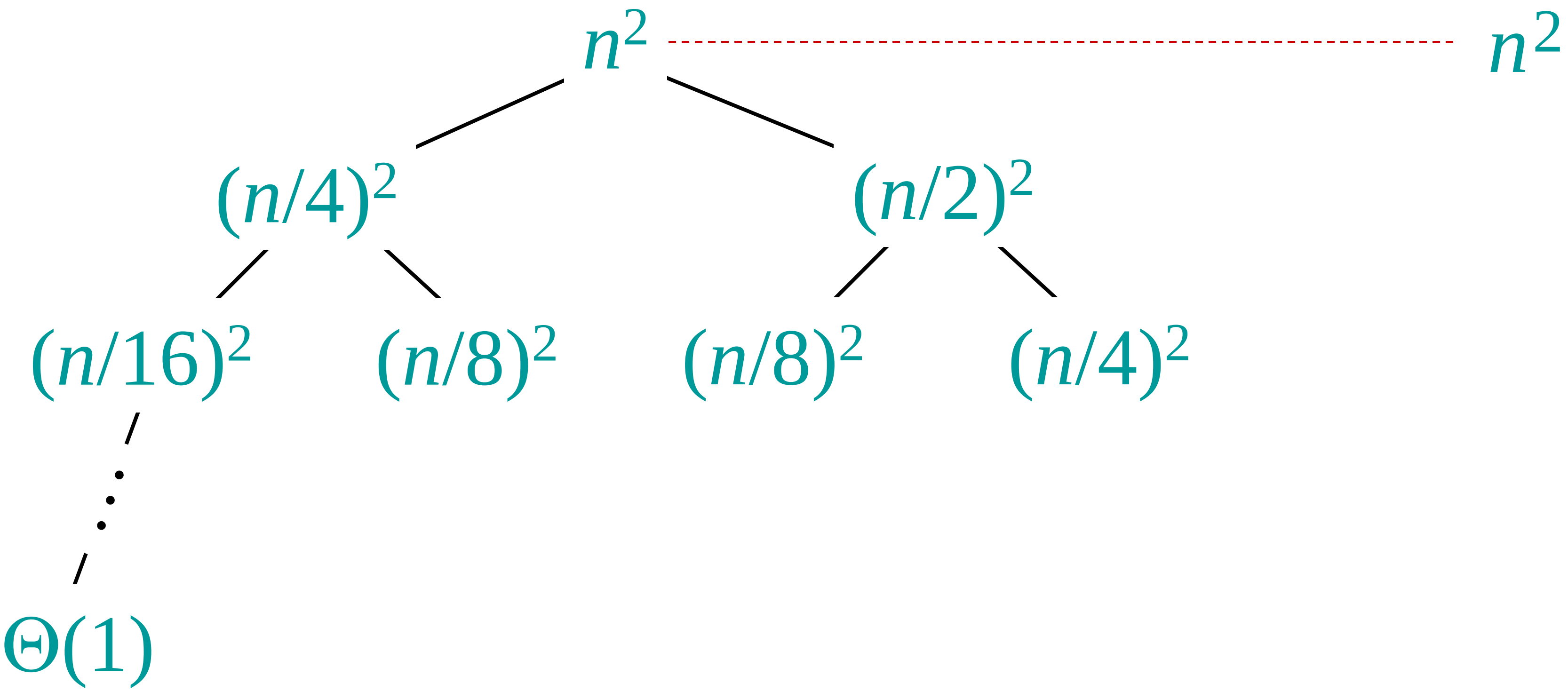
Example of recursion tree

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



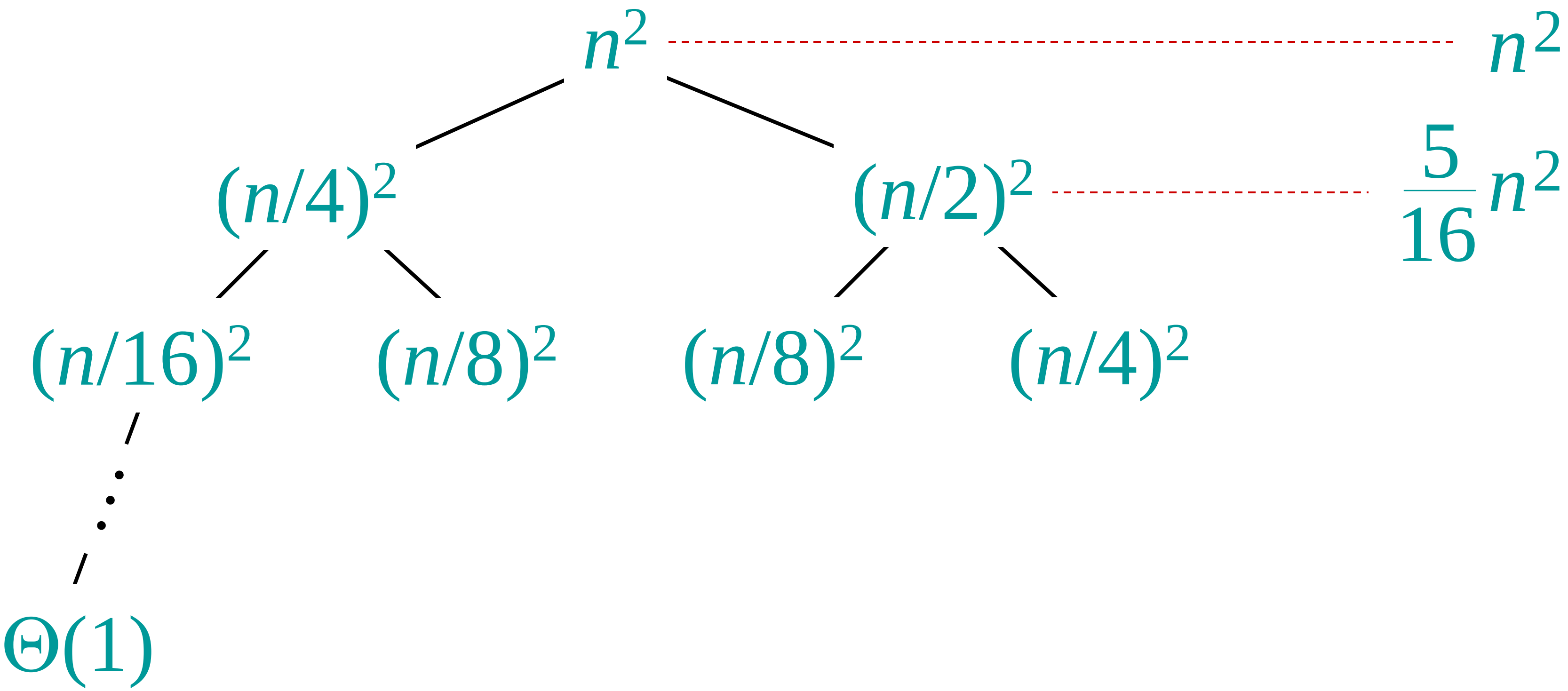
Example of recursion tree

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



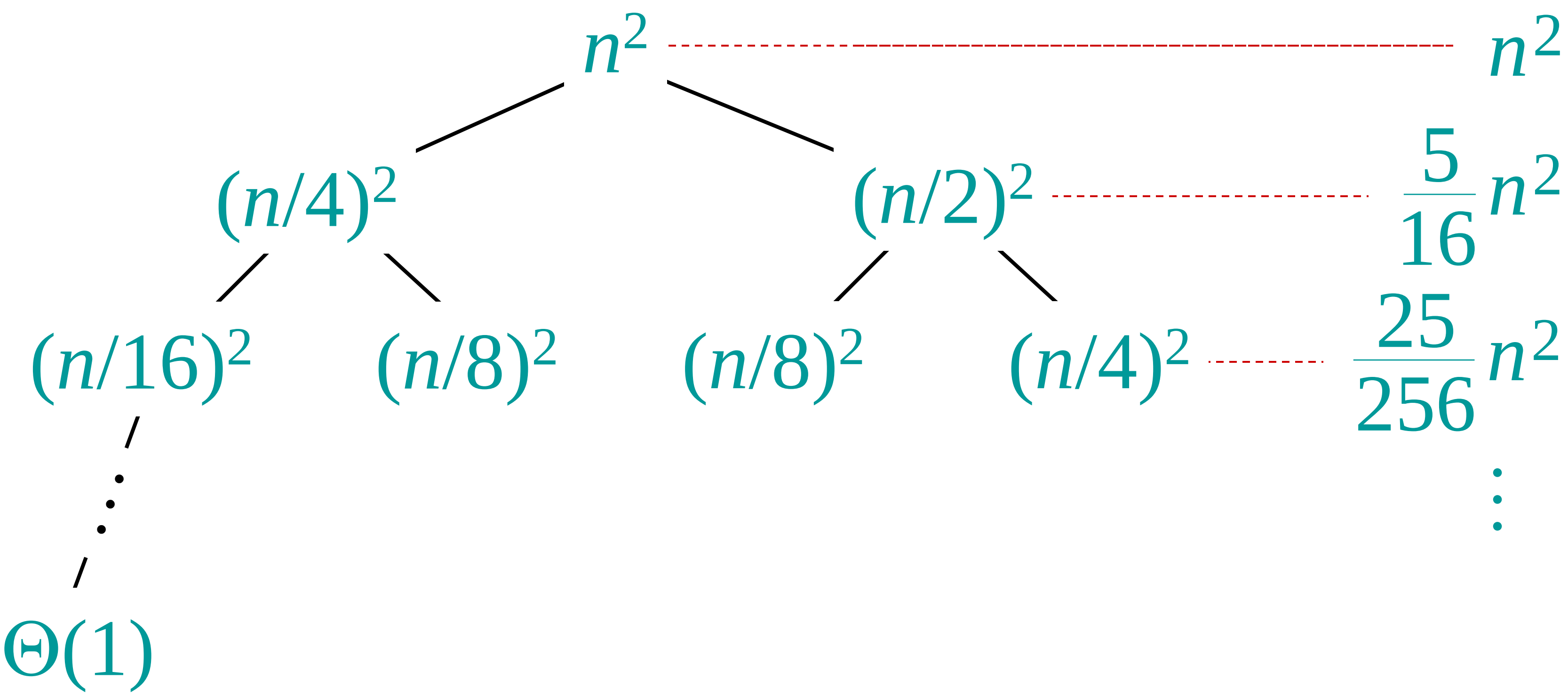
Example of recursion tree

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



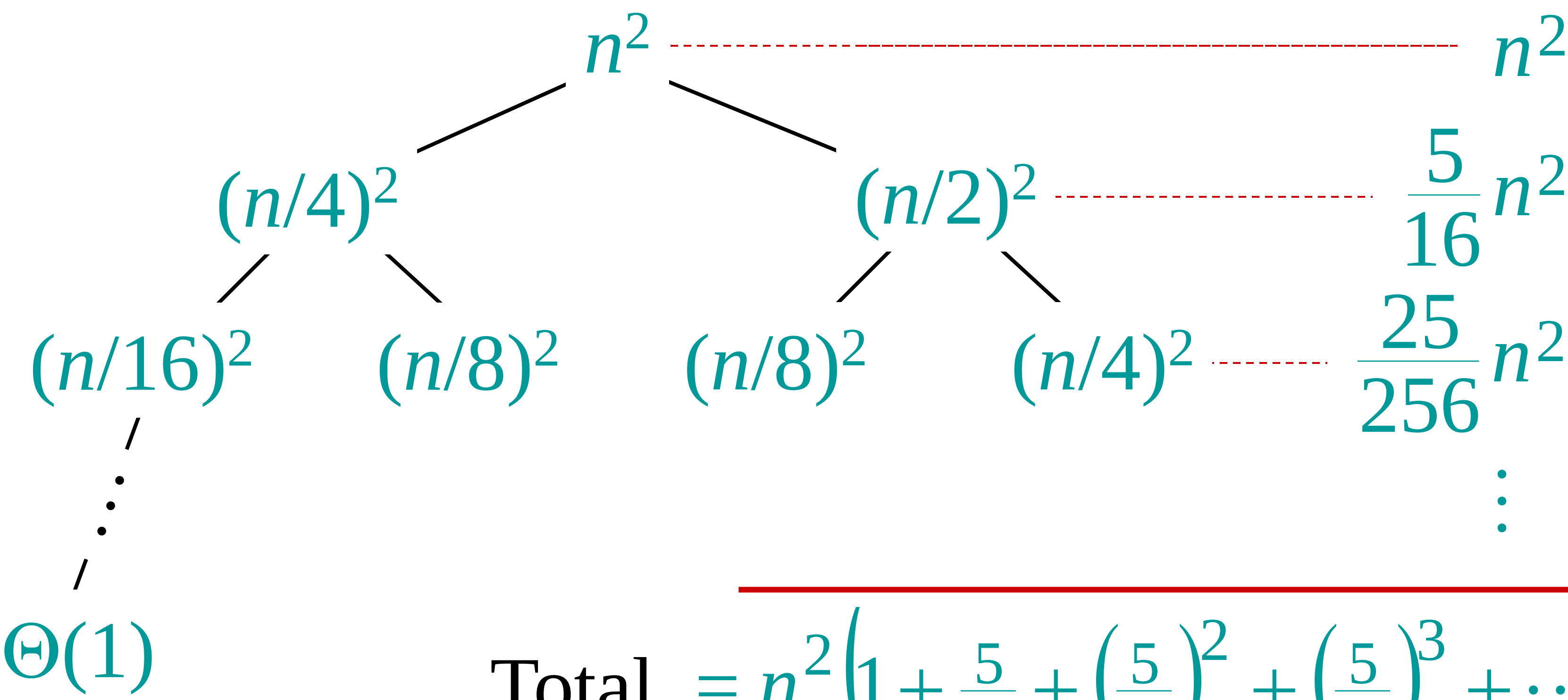
Example of recursion tree

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



Example of recursion tree

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



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

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

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

Return to last
slide viewed.

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:

$h = \log_b n$

$$f(n) \quad\cdots\cdots\quad f(n)$$

a

$$f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad\cdots\quad a\,f(n/b)$$

a

$$f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad\cdots\quad a^2 f(n/b^2)$$

$$\vdots \qquad\qquad \vdots$$

$$T(1) \qquad\qquad n^{\log_b a}\,T(1)$$

$$\begin{aligned} \#\text{leaves} &= a^h \\ &= a^{\log_b n} \\ &= n^{\log_b a} \end{aligned}$$

Three common cases

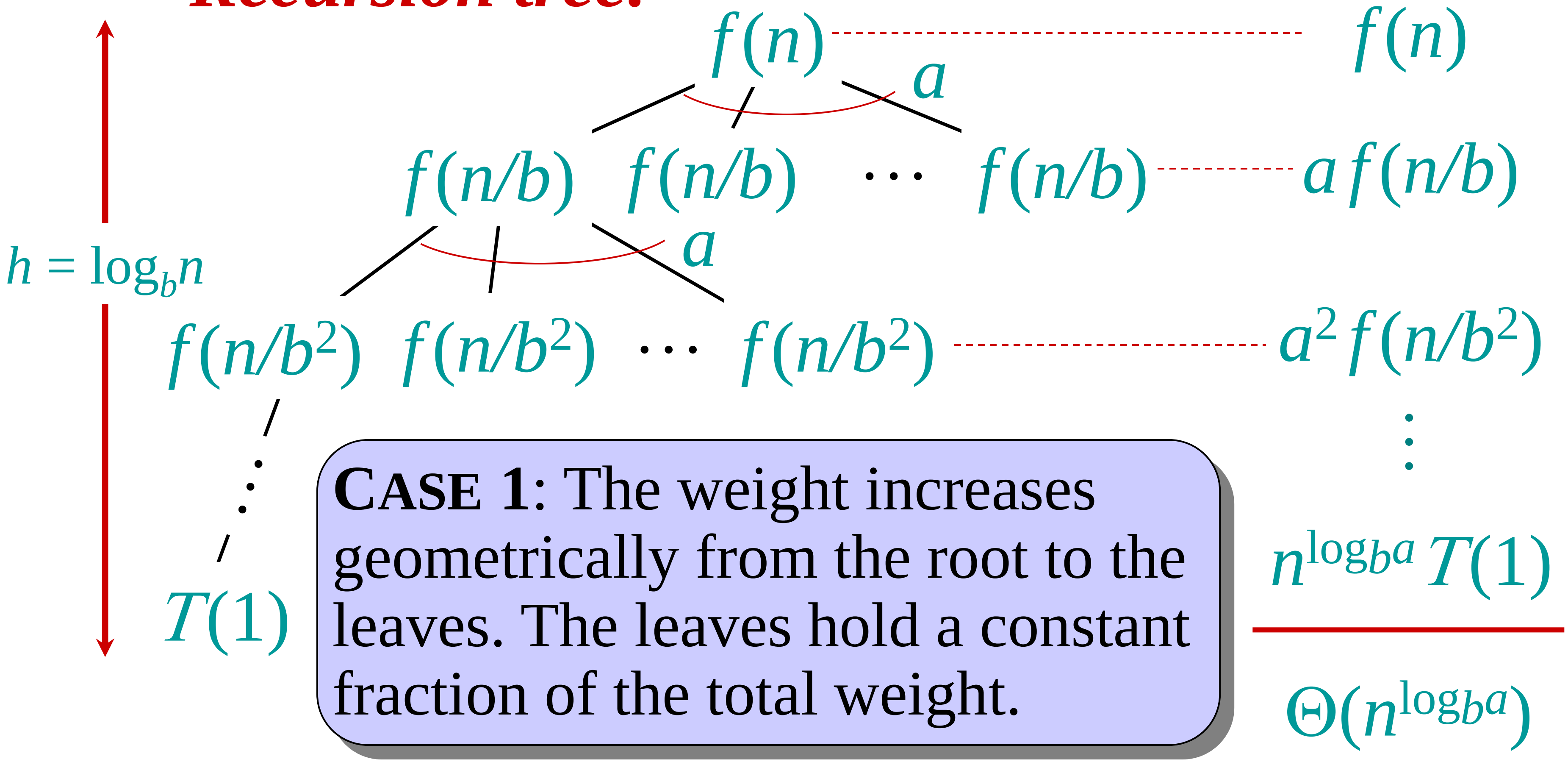
Compare $f(n)$ with $n^{\log_b a}$:

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

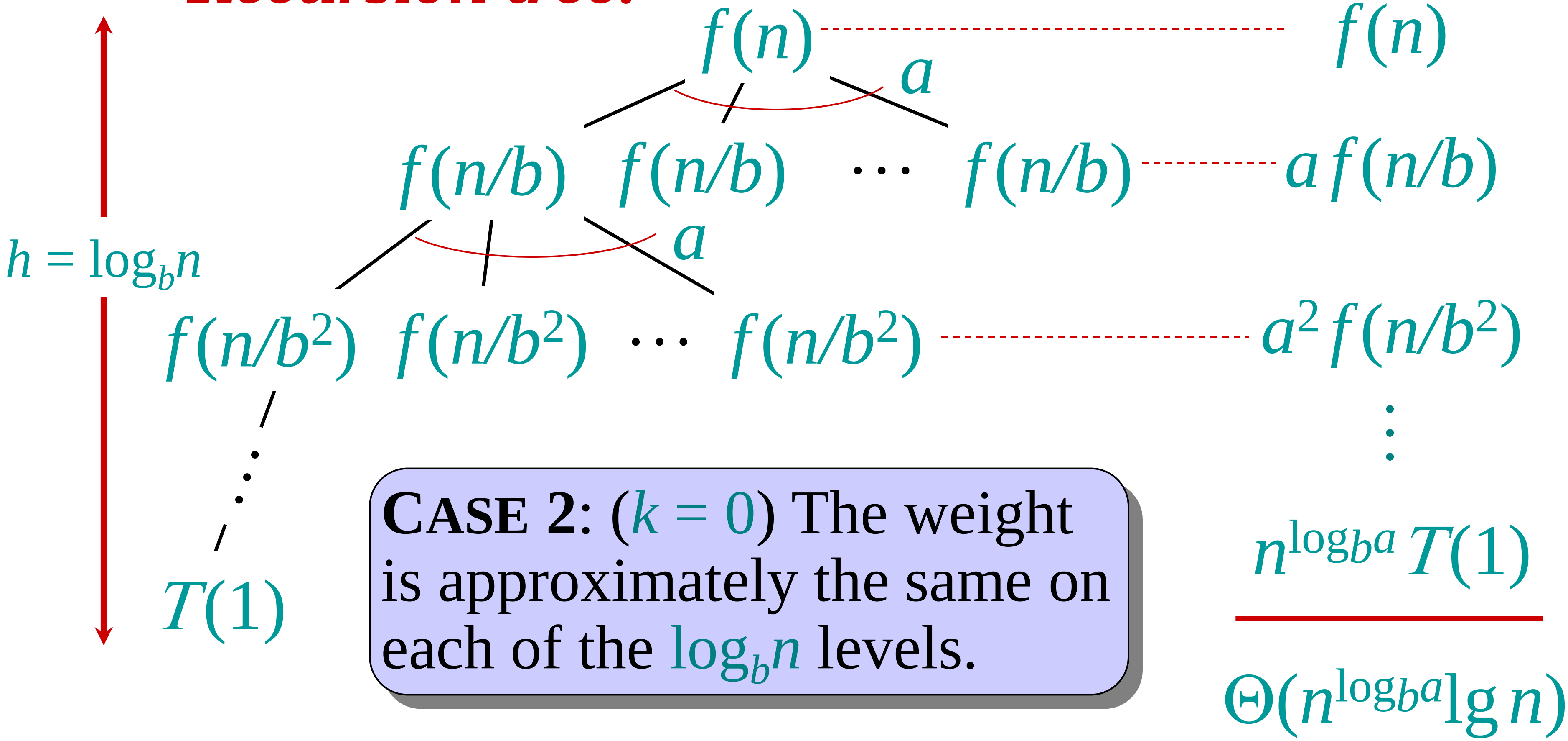
Compare $f(n)$ with $n^{\log_b a}$:

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:



CASE 2: ($k = 0$) The weight is approximately the same on each of the $\log_b n$ levels.

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 $a f(n/b) \leq c f(n)$ for some constant $c < 1$.

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

Idea of master theorem

Recursion tree:

$$f(n) \quad \cdots\cdots \quad f(n)$$

a

$$f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad \cdots \quad a\,f(n/b)$$

a

$h = \log_b n$

$$f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad \cdots \quad a^2 f(n/b^2)$$

$$\vdots$$

$$T(1) \qquad\qquad n^{\log_b a}\, T(1)$$

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

$$\Theta(f(n))$$

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 $4(cn/2)^3 \le cn^3$ (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 $n^\varepsilon = \omega(\lg n)$.

Conclusion

- Next time: applying the master method.
- For proof of master theorem, goto section