Week 2: Part 2

Recursion, Recurrences & Running time

Methods for Solving Recurrences

- Iteration method
- Substitution method
- Recursion tree method
- Master method
- Muster method

The Iteration Method

- Convert the recurrence into a summation and try to bound it using a known series
  - Iterate the recurrence until the initial condition is reached.
  - Use back-substitution to express the recurrence in terms of n and the initial (boundary) condition.

Iteration Method – Binary Search

$$\mathbf{T(n) = c + T(n/2)}$$

$T(n) = c + T(n/2)$

$\quad = c + c + T(n/4)$

$\quad = c + c + c + T(n/8)$

$T(n/2) = c + T(n/4)$

$T(n/4) = c + T(n/8)$

Stop when $n/2^i = 1 \quad \Rightarrow i = \lg n$

$$T(n) = \underbrace{c + c + \ldots + c}_{\text{n times}} + T(1)$$

$\quad = c\lg n + T(1)$

$\quad = \Theta(\lg n)$

Iteration - Mergesort

$$\mathbf{T(n) = n + 2T(n/2)}$$

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

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

$= n + n + 4T(n/4)$

$= n + n + 4(n/4 + 2T(n/8))$

$= n + n + n + 8T(n/8)$

… $= in + 2^i T(n/2^i)$ stop at $i = \lg n$

$= n\lg n + 2^{\lg n} T(1)$

$= n\lg n + nT(1)$

$= \Theta(n\lg n)$

Substitution Method

- Guess a solution

  $T(n) = O(g(n))$

  Induction goal: apply the definition of the asymptotic notation

  $T(n) \le c\, g(n)$, for some $c > 0$ and $n \ge n_0$

- Induction hypothesis: $T(k) \le c\, g(k)$ for all $k < n$

- Prove the induction goal
  - Use the **induction hypothesis** to find some values of the constants c and n_0 for which the **induction goal** holds

Substitution: $T(n) = T(n-1)+T(n-2)$

Guess: $T(n) = O(\phi^n)$

Induction goal: $T(n) \leq c\phi^n$, for some c and $n \geq n_0$

- Induction hypothesis: $T(k) \leq c\phi^k$ for $k < n$
- Proof of induction goal:

$$
\begin{aligned}
T(n) &= T(n-1) + T(n-2) \\
&\leq c\phi^{n-1} + c\phi^{n-2} \\
&\leq c\phi^{n-2} (\phi + 1) \\
&\leq c\phi^{n-2} (\phi^2) \\
T(n) &\leq c\,\phi^n
\end{aligned}
$$

$T(n) = O(\phi^n)$

Properties

$$\boldsymbol{\Phi = \frac{1+\sqrt{5}}{2}}$$

$$\boldsymbol{\Phi^2 = \frac{3+\sqrt{5}}{2}}$$

$$\boldsymbol{\Phi + 1 = \Phi^2}$$

Recursion-tree method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- Convert the recurrence into a tree:
  - Each node represents the cost incurred at various levels of recursion
  - Sum up the costs of all levels
- The recursion-tree method can be unreliable, just like any method that uses ellipses (…).
- Usually involves geometric series

Example of recursion tree

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

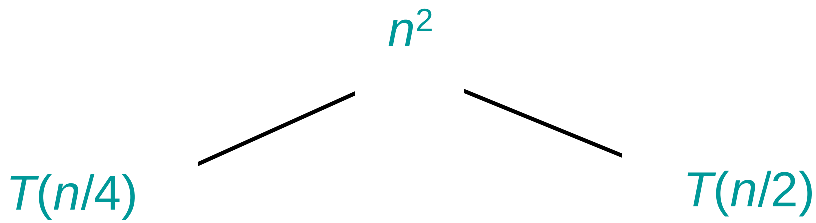
Example of recursion tree

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

$T(n)$

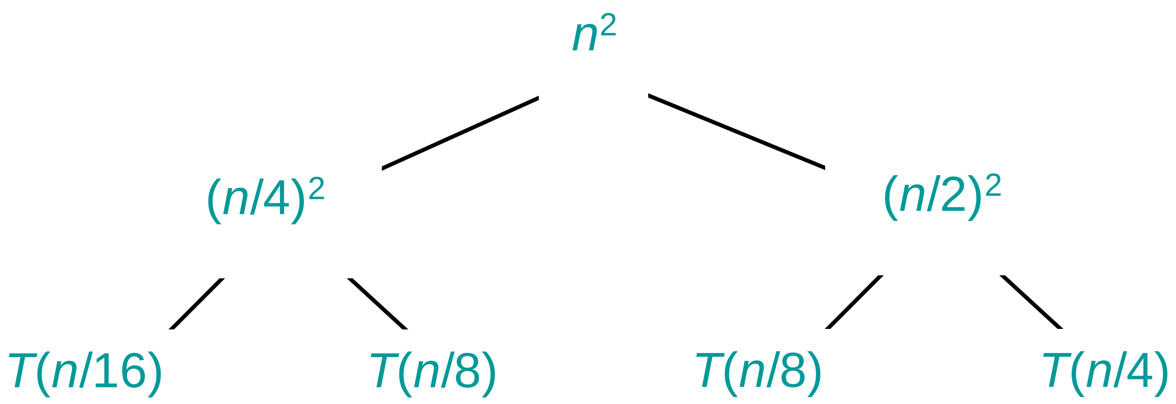
Example of recursion tree

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



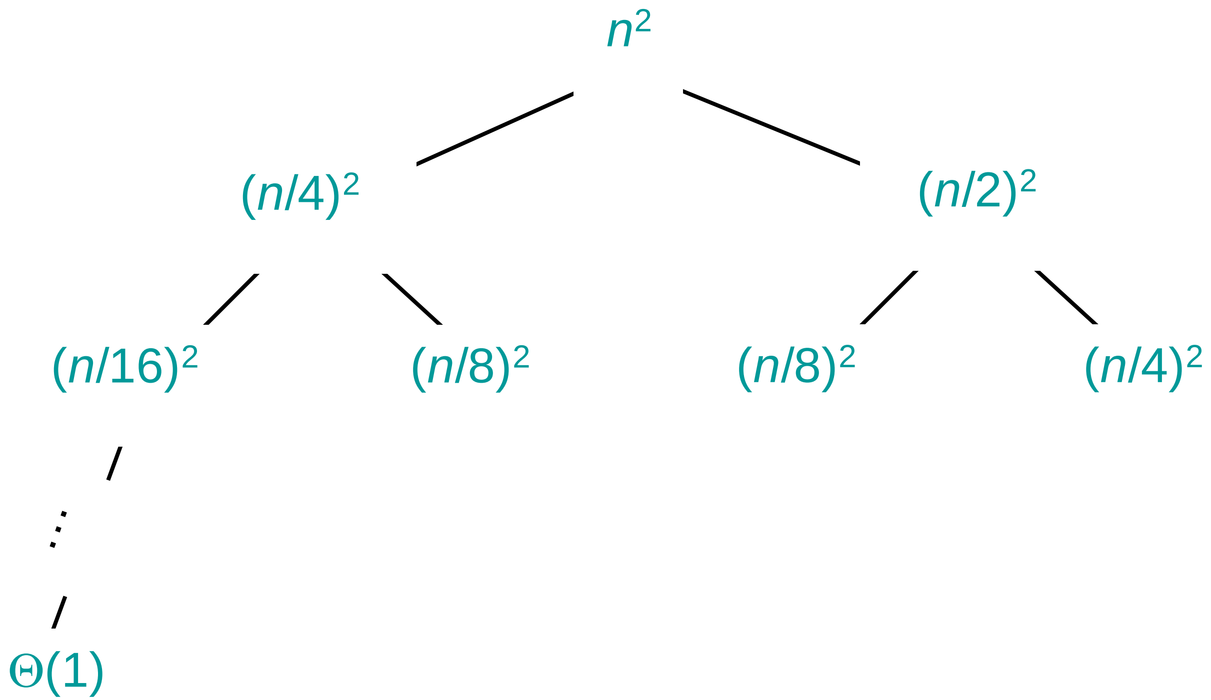
Example of recursion tree

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



Example of recursion tree

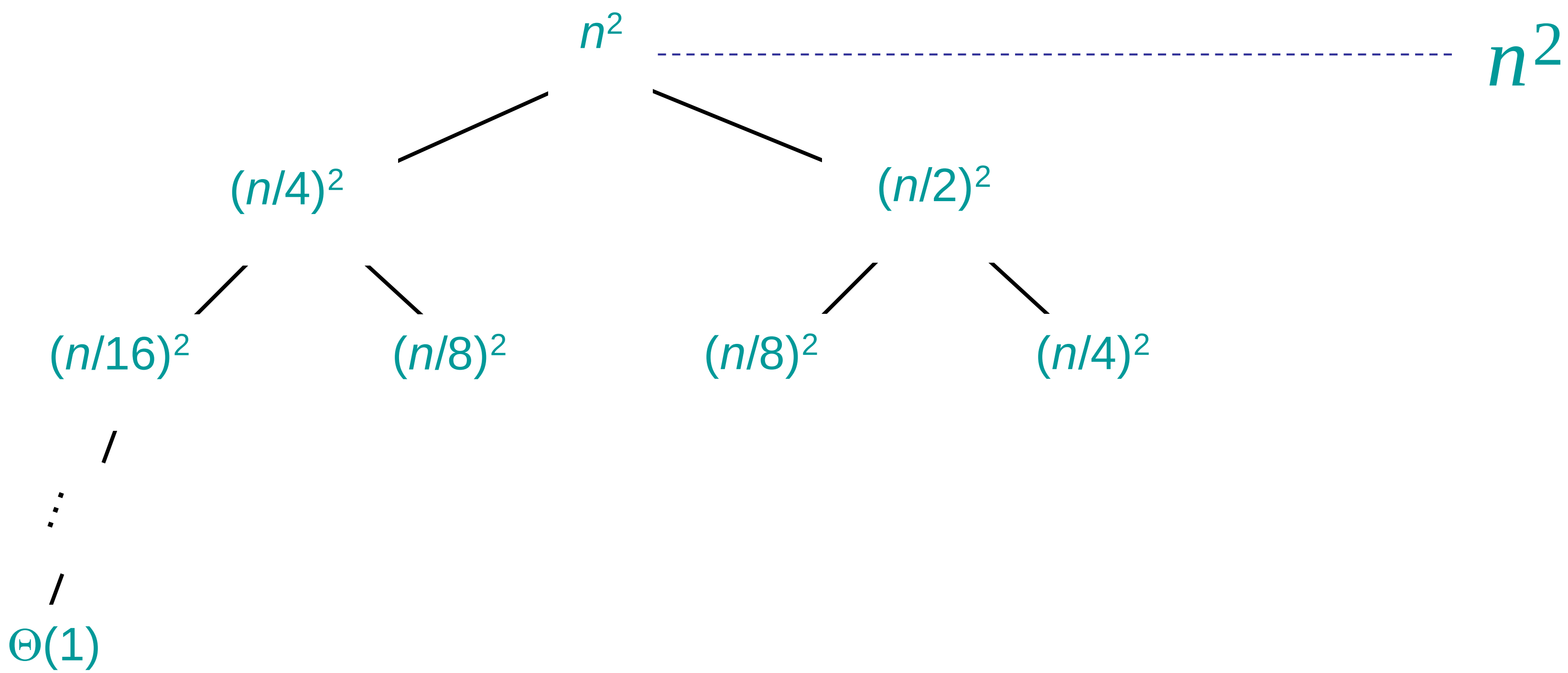
Solve $T(n) = T(n/4) + T(n/2) + n^2$:



Example of recursion tree

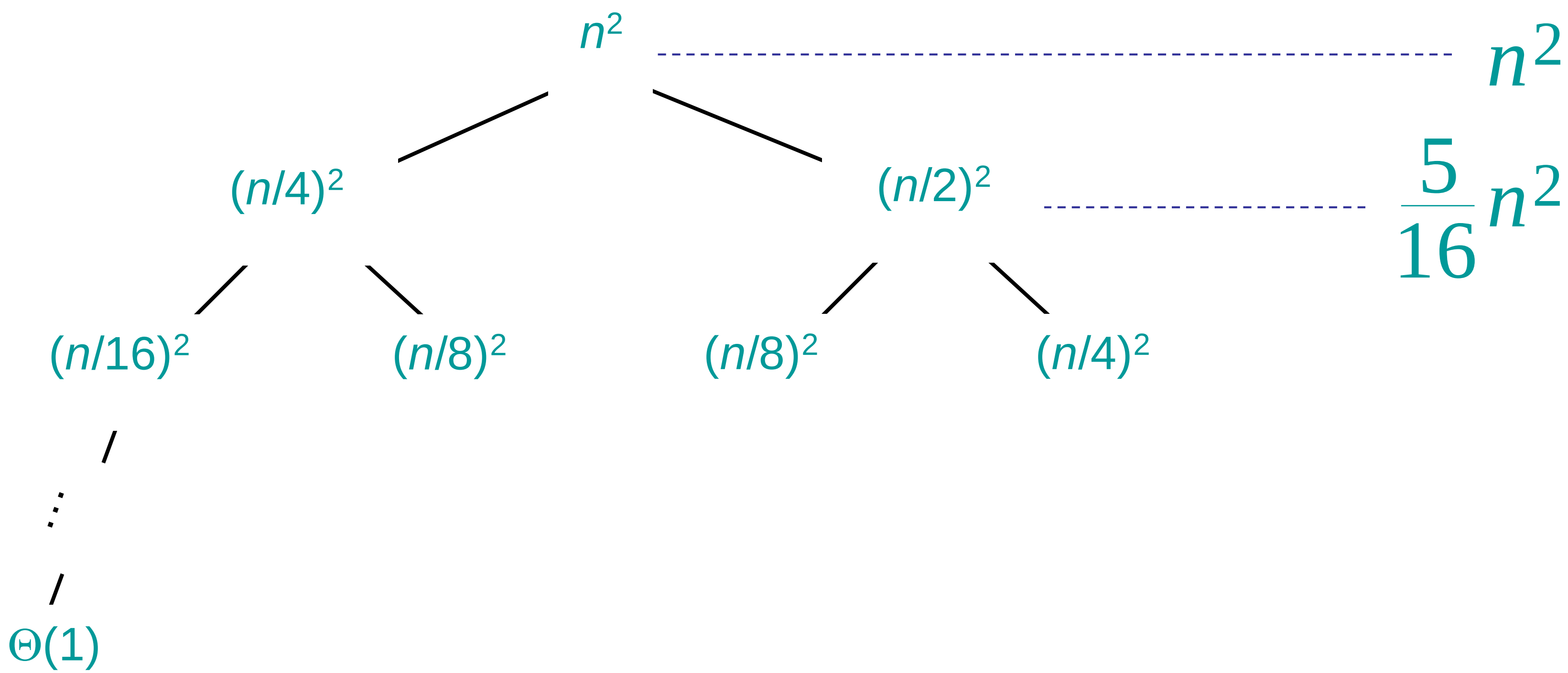
Solve $T(n) = T(n/4) + T(n/2) + n^2$:

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



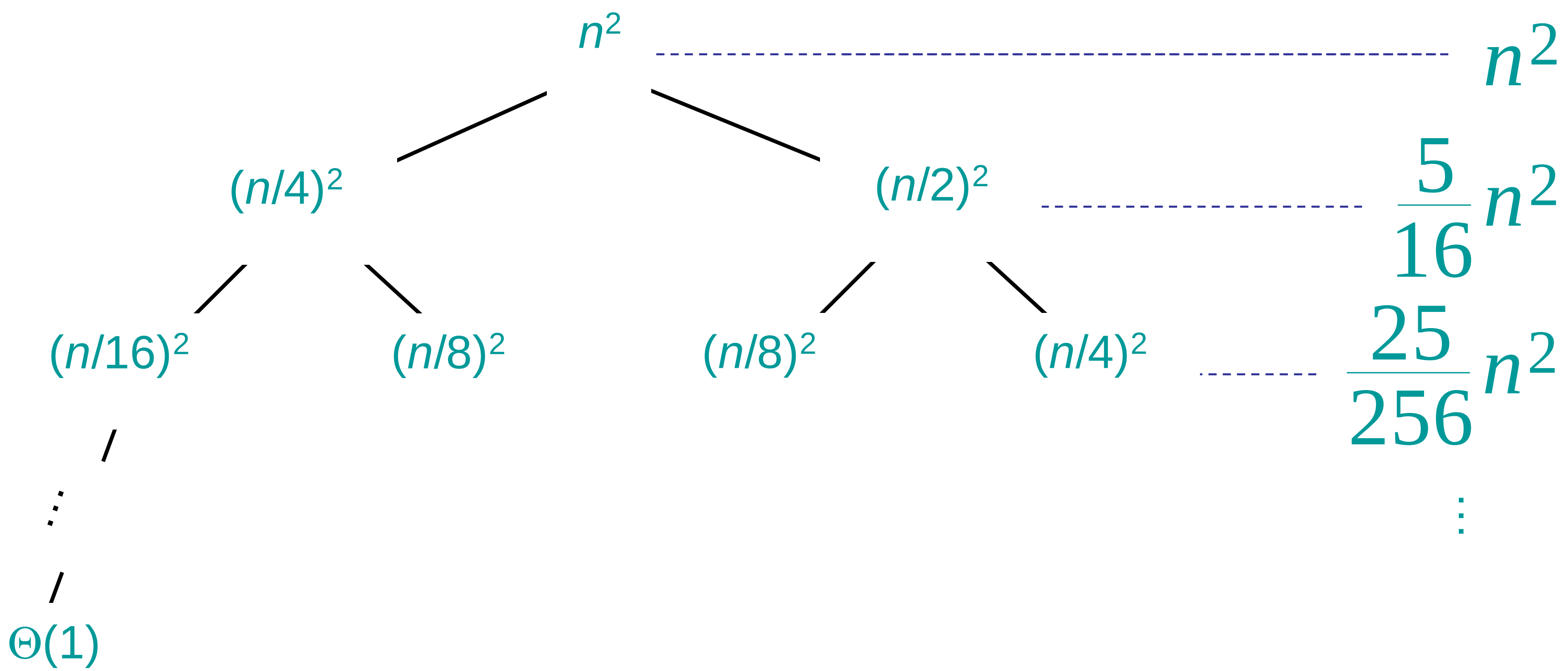
Example of recursion tree

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



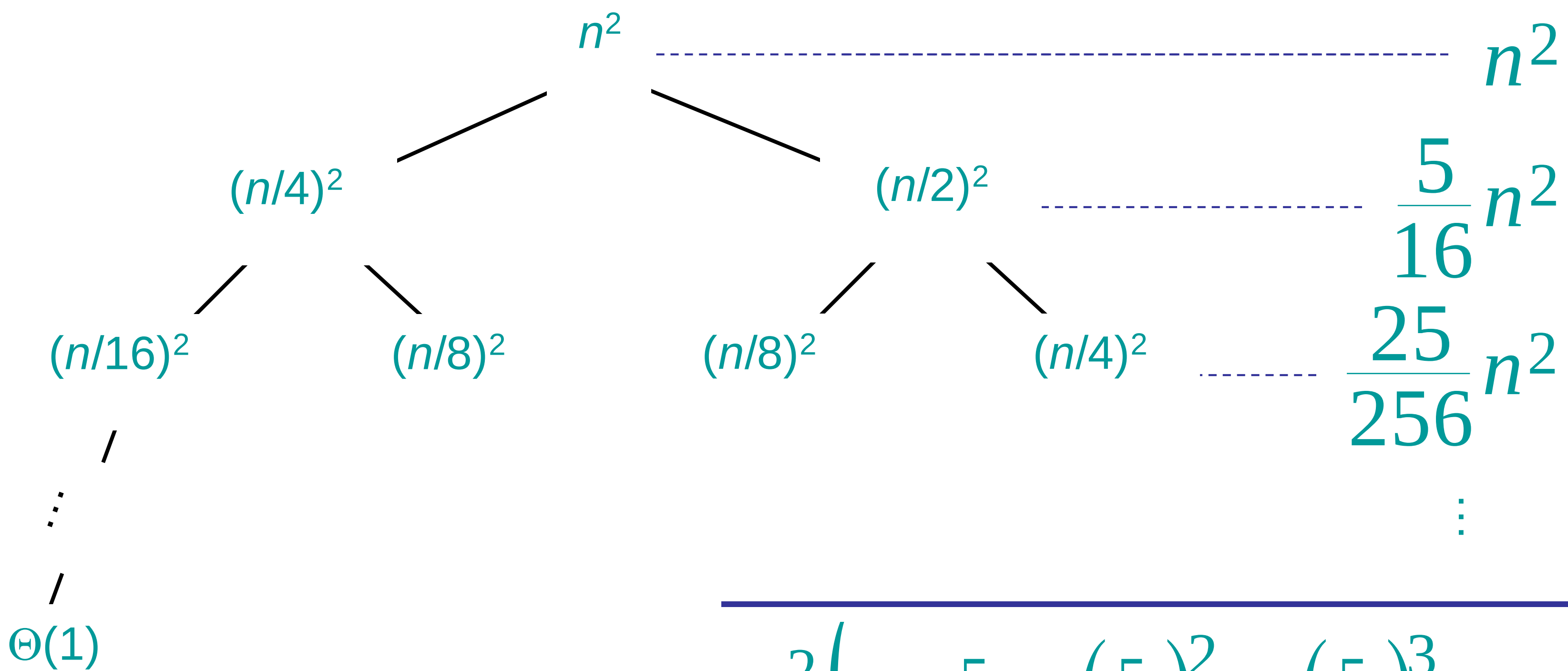
Example of recursion tree

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



Example of recursion tree

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



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

geometric series

$$= O(n^2)$$

Geometric series

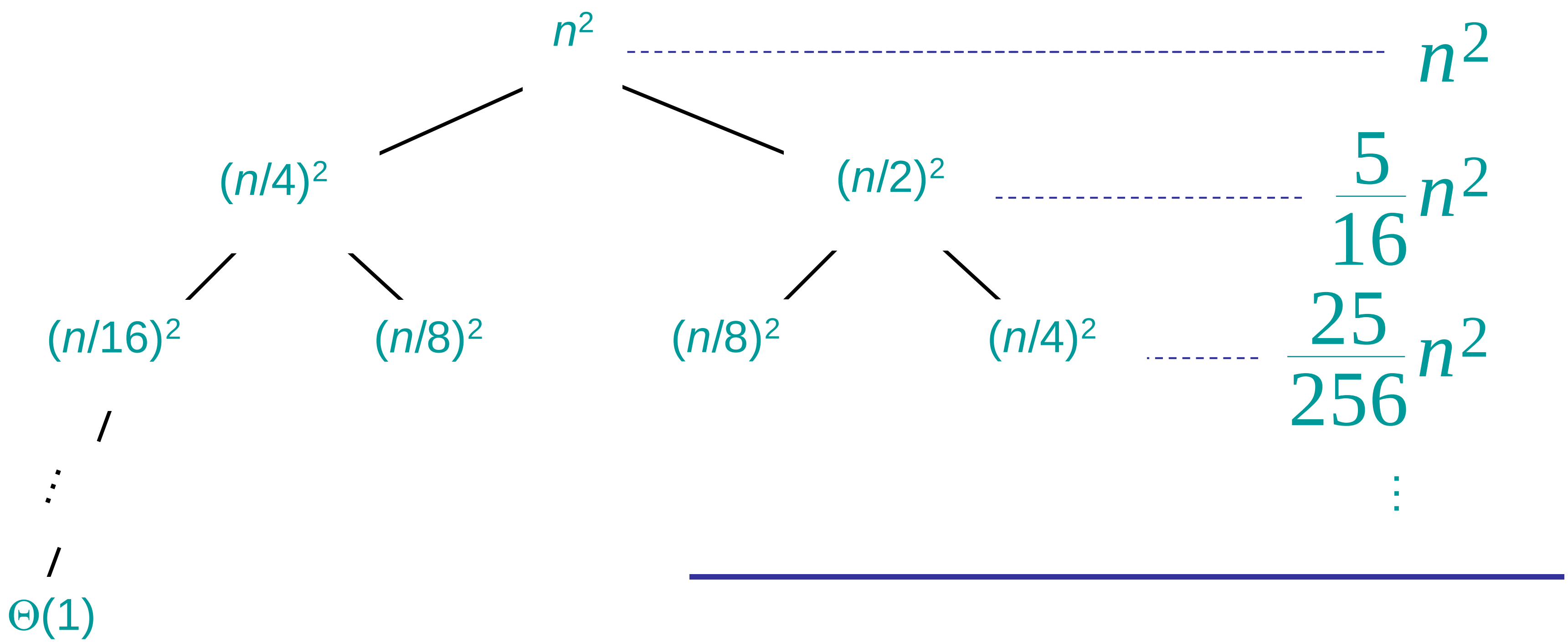
$$1 + x + x^2 + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x}$$ for $x \neq 1$

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

$$n^2\left(1 + \tfrac{5}{16} + \left(\tfrac{5}{16}\right)^2 + \left(\tfrac{5}{16}\right)^3 + \cdots\right) = n^2\left(\frac{1}{1 - \frac{5}{16}}\right) = \tfrac{16}{11} n^2$$

Example of recursion tree

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

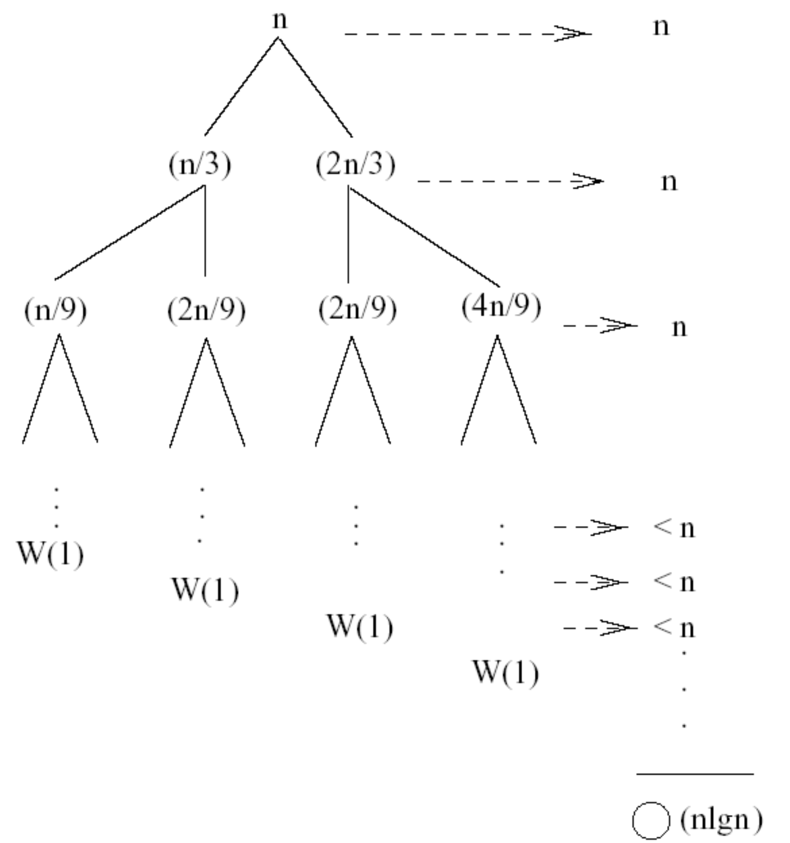


Total $\geq n^2 \quad = \Omega(n^2)$

Therefore $T(n) = \Theta(n^2)$

Recursion Tree – Example 2

T(n) = T(n/3) + T(2n/3) + n



- The longest path from the root to a leaf is:

  $n \to (2/3)n \to (2/3)^2\, n \to \dots \to 1$

- Subproblem size hits 1 when

  $1 = (2/3)^i n \Leftrightarrow i = \log_{3/2} n$

- cost of the problem at level i = n
- Total cost:

$$T(n) < n + n + \ldots = n(\log_{3/2} n) = n\frac{\lg n}{\lg\frac{3}{2}} = O(n\lg n)$$

$\Rightarrow$ T(n) = O(nlgn)

The Master Method

The master method applies to recurrences of the form

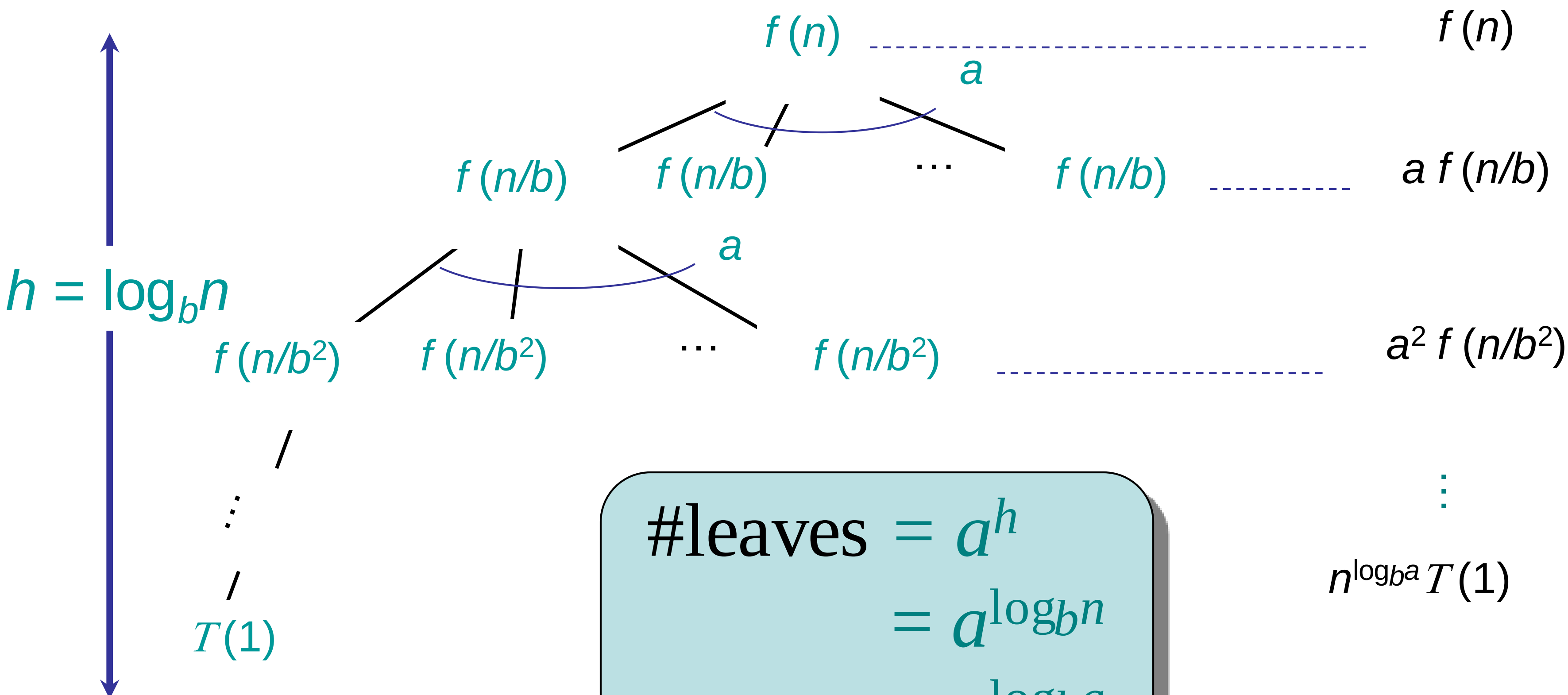
$$T(n) = a\,T(n/b) + f(n) \ ,$$

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

Idea of Master Method

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

Recursion tree:

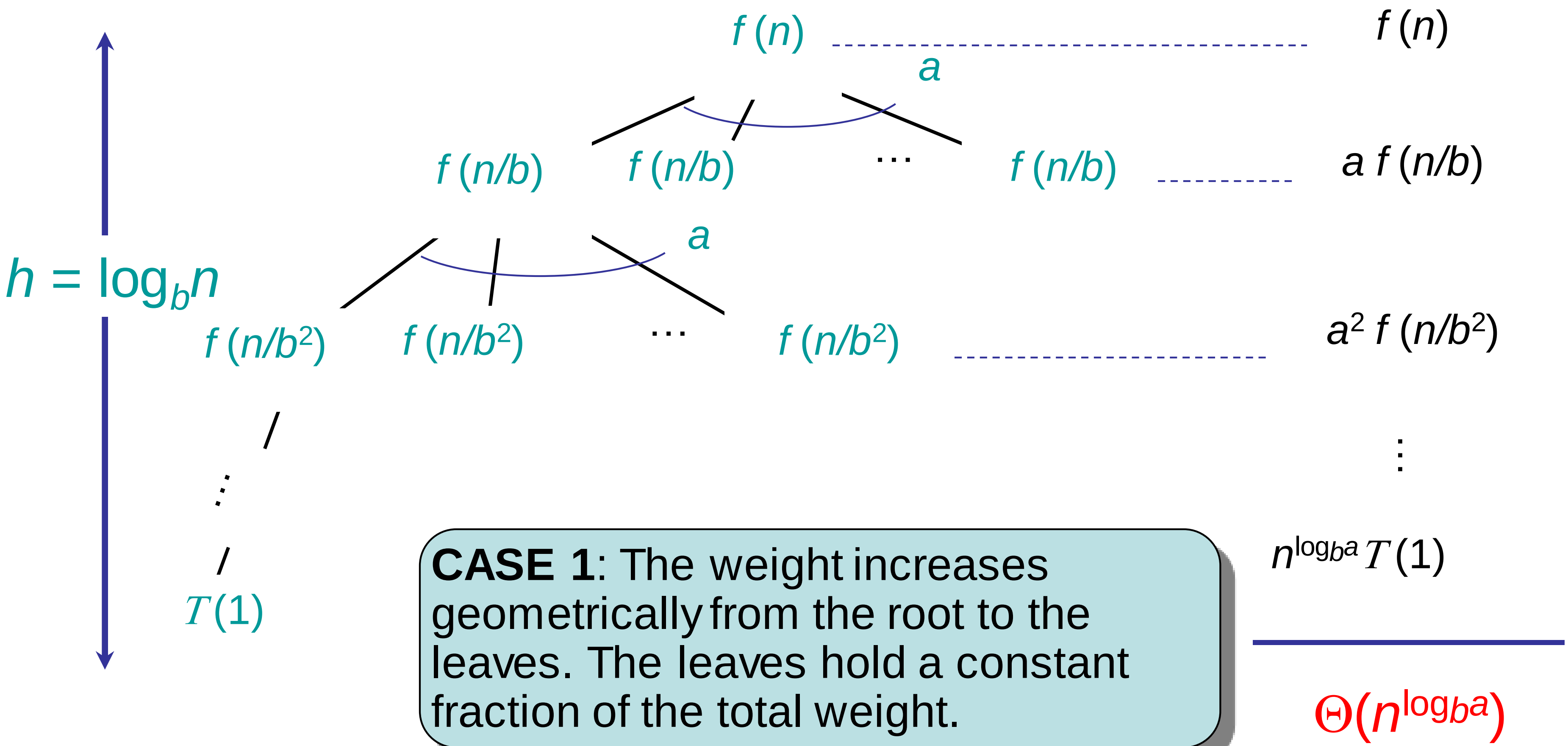


$$\begin{aligned}\#\text{leaves} &= a^h \\ &= a^{\log_b n} \\ &= n^{\log_b a}\end{aligned}$$

Idea of Master Method

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

Recursion tree:



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

Case 1

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

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

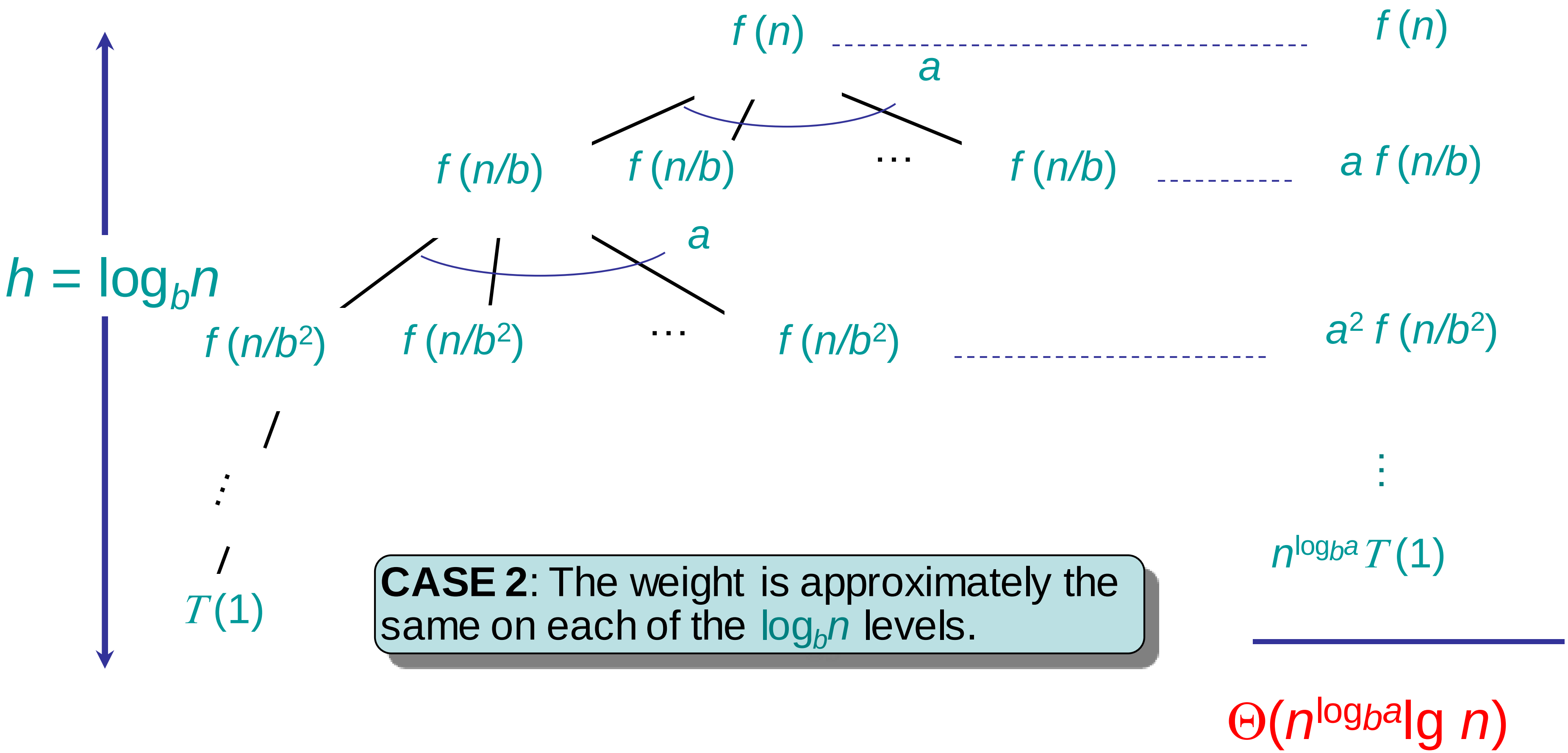
Case 1: $f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 1$.

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

Idea of Master Method

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

Recursion tree:



CASE 2: The weight is approximately the same on each of the $\log_b n$ levels.

Case 2

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

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

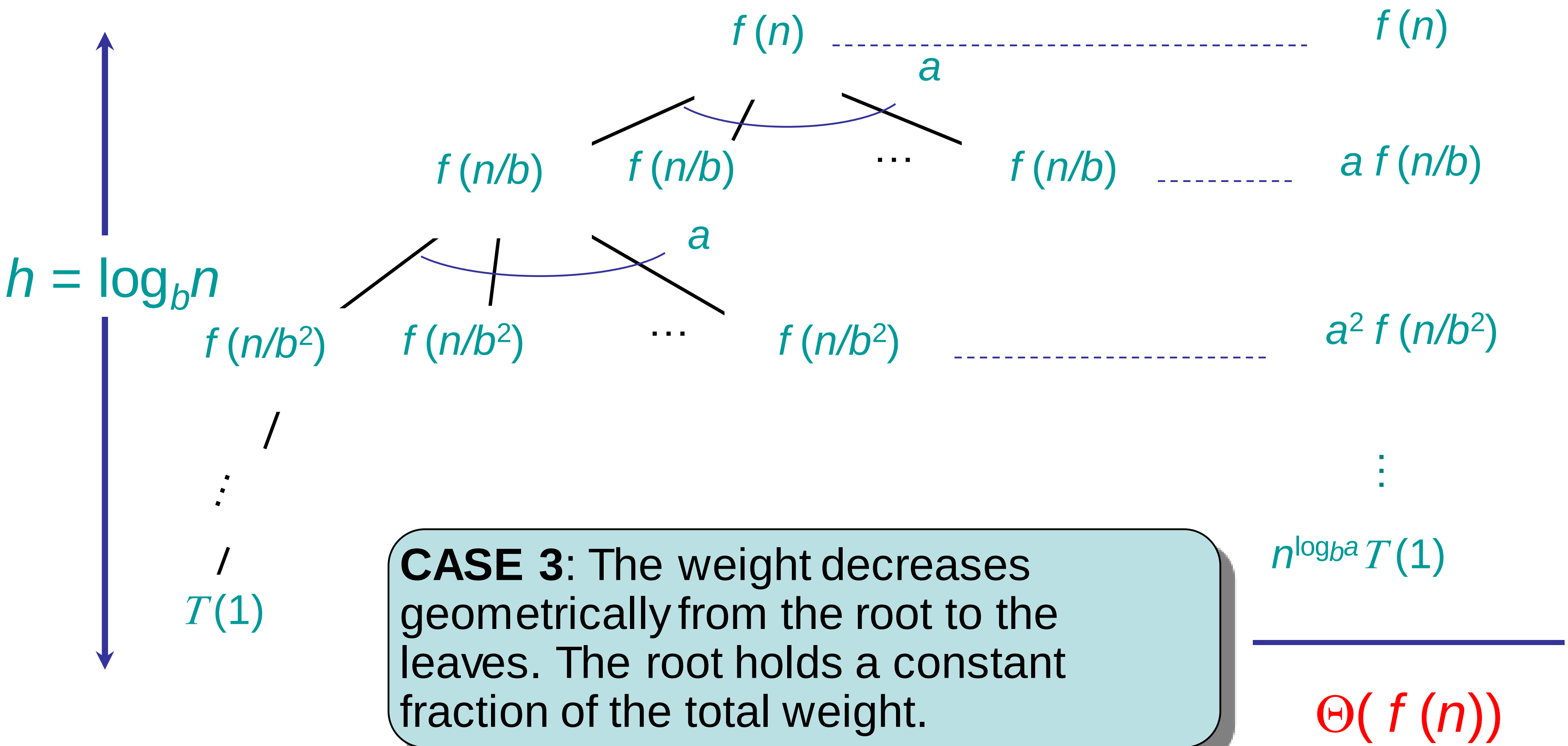
CASE 2: $f(n) = \Theta(n^2)$

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

Idea of master theorem

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

Recursion tree:



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

Case 3

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 $c = 1/2$.

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

No Cases

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.

Master Method

"Formula" for solving recurrences of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where, $a \geq 1$, $b > 1$, and $f(n) > 0$

case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

$af(n/b) \leq cf(n)$ for some $c < 1$ and all sufficiently large n, then:

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

regularity

Master Method – Binary Search

$T(n) = T(n/2) + c$ $\quad\quad a = 1, b = 2, \log_2 1 = 0$

compare $n^{\log_2 1} = n^0 = 1$ with $f(n) = c$

Case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

$f(n) = \Theta(1) \Rightarrow$ case 2

$\Rightarrow T(n) = \Theta(\lg n)$

Master Method – Example 1

$T(n) = 2T(n/2) + n^2$ $\quad$ $a = 2$, $b = 2$, $\log_2 2 = 1$

compare n with $f(n) = n^2$

case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$

$\Rightarrow f(n) = \Omega(n^{1+\varepsilon})$ case 3 $\Rightarrow$ verify regularity cond.

$a\, f(n/b) \le c\, f(n)$

$\Leftrightarrow 2\, n^2/4 \le c\, n^2 \Rightarrow c = ½$ is a solution $(c<1)$

$\Rightarrow T(n) = \Theta(n^2)$

Master Method – Example 2

$T(n) = 2T(n/2) + \sqrt{n}$ $\quad$ $a = 2$, $b = 2$, $\log_2 2 = 1$

compare n with $f(n) = n^{1/2}$

$\Rightarrow f(n) = O(n^{1-\varepsilon})$ $\quad$ case 1

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

Master Method - Example 3

$T(n) = 3T(n/4) + n\lg n \qquad a = 3,\ b = 4,\ \log_4 3 = 0.793$

compare $n^{0.793}$ with $f(n) = n\lg n$

$f(n) = \Omega(n^{\log_4 3+\varepsilon})$ case 3

check regularity condition:

$$3 * (n/4)\lg(n/4) \le (3/4)n\lg n = c * f(n),\ c=3/4$$

$\Rightarrow T(n) = \Theta(n\lg n)$

Master Method: Merge-Sort

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

where, a =2, b =2, and f(n) =n

$n^{\log_b a} = n^{\log_2 2} = n$

case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

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

Decrease and Conquer

Muster Theorem for *"decrease and conquer"* recurrences of the form

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

for some integer constants a, b > 0, d ≥ 0 .

If $f(n)$ is $O(n^d)$ then

$$T(n) = \begin{cases} O\left(n^d\right), & if\ a < 1, \\ O\left(n^{d+1}\right), & if\ a = 1 \\ O\left(n^d a^{n/b}\right), & if\ a > 1. \end{cases}$$

Decrease and Conquer: Towers

$$T(n) = 2\ T(n\text{-}1) + 1$$

$$T(n) = a\ T(n\text{-}b) + f(n)$$

$a = 2$, b= 1, f(n) = 1 so d = 0.

$$T(n) = \begin{cases} O(n^d), & if\ a < 1, \\ O(n^{d+1}), & if\ a = 1 \\ O(n^d a^{n/b}), & if\ a > 1. \end{cases}$$

T(n) is O(2^n) even better

f(n) is $\Theta(n^d)$ so we could conclude that T(n) is $\Theta(2^n)$.