Introduction to Algorithms

Topic 2 : Asymptotic Mark and Recursive Equation

XiangYang Li and Haisheng Tan

School of Computer Science and Technology
University of Science and Technology of China (USTC)

Fall Semester 2025

Outline of Topics

Table of Contents

Asymptotic Notation: O−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 $n \geq n_0$.

Asymptotic Notation: O−notation

O-notation: upper bounds

We write $f(n) = O(g(n))$ if there exist constants $c > 0, n_0 > 0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$.

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

Asymptotic Notation: O−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: $2n^2 = O(n^3) \qquad (c = 1, n_0 = 2)$

functions,
not values

Asymptotic Notation: O−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: $2n^2 = O(n^3)$ $\quad (c = 1, n_0 = 2)$

functions, not values

funny, "one-way" equality

Set Definition of O-notation

$$O(g(n)) = \{f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}.$$

Set Definition of O-notation

$$O(g(n)) = \{f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}.$$

Example: $2n^2 \in O\left(n^3\right)$

Macro Substitution

Convention: A set in a formula represents an anonymous function in the set.

Example: $f(n) = n^3 + O\left(n^2\right)$
means
$f(n) = n^3 + h(n)$
for some $h(n) \in O\left(n^2\right)$.

Asymptotic Notation: Ω-notation

O-notation is an upper-bound notation.
The Ω-notation provides a lower bound.

Set definition of Ω-notation

$$\Omega(g(n)) = \{f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \leq c \cdot g(n) \leq f(n) \text{ for all } n \geq n_0\}$$

Asymptotic Notation: Ω-notation

O-notation is an upper-bound notation.
The Ω-notation provides a lower bound.

Set definition of Ω-notation

$$\Omega(g(n)) = \{f(n) : \text{there exist constants } c > 0, n_0 > 0 \text{ such that } 0 \le c \cdot g(n) \le f(n) \text{ for all } n \ge n_0\}$$

Example: $\sqrt{n} = \Omega(\lg n)$

Asymptotic Notation: Θ-notation

Θ-notation: tight bounds

We write $f(n) = \Theta(g(n))$ if there exist constants $c_1 > 0, c_2 > 0, n_0 > 0$ such that $c_2 g(n) \geq f(n) \geq c_1 g(n) \geq 0$ for all $n \geq n_0$.

$$\Theta(g(n)) = O(g(n)) \bigcap \Omega(g(n))$$

Asymptotic Notation: Θ-notation

Θ-notation: tight bounds

We write $f(n) = \Theta(g(n))$ if there exist constants $c_1 > 0, c_2 > 0, n_0 > 0$ such that $c_2 g(n) \geq f(n) \geq c_1 g(n) \geq 0$ for all $n \geq n_0$.

$$\Theta(g(n)) = O(g(n)) \bigcap \Omega(g(n))$$

Example: $\frac{1}{2}n^2 - 2n = \Theta\left(n^2\right)$

Asymptotic Notation: Θ-notation

Θ-notation: tight bounds

We write $f(n) = \Theta(g(n))$ if there exist constants $c_1 > 0, c_2 > 0, n_0 > 0$ such that $c_2 g(n) \geq f(n) \geq c_1 g(n) \geq 0$ for all $n \geq n_0$.

$$\Theta(g(n)) = O(g(n)) \bigcap \Omega(g(n))$$

Example:

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

$$\Theta(n^0) \text{ or } \Theta(1)$$

Asymptotic Notation: Θ-notation

Θ-notation: tight bounds

We write $f(n) = \Theta(g(n))$ if there exist constants $c_1 > 0, c_2 > 0, n_0 > 0$ such that $c_2 g(n) \geq f(n) \geq c_1 g(n) \geq 0$ for all $n \geq n_0$.

$$\Theta(g(n)) = O(g(n)) \bigcap \Omega(g(n))$$

Example:

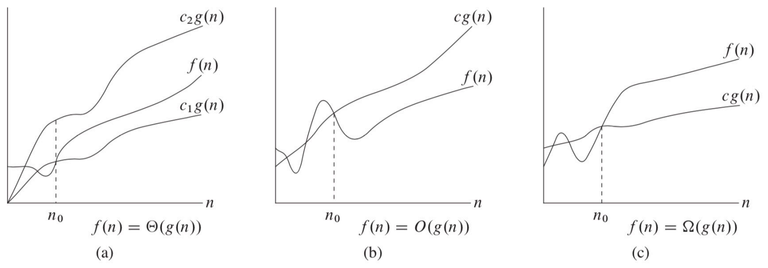
$$\frac{1}{2}n^2 - 2n = \Theta\left(n^2\right)$$

$$\Theta(n^0) \text{ or } \Theta(1)$$

Theorem:

The leading constant and low order terms do not matter.

Graphic Examples of the Θ, O, Ω



Other Asymptotic Notations

o-notation

$o(g(n)) = \{f(n)$: for all $c > 0$, there exist constants $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0\}$.

Other equivalent definition $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

ω-notation

$\omega(g(n)) = \{f(n)$: for all $c > 0$, there exist constants $n_0 > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0\}$.

Other equivalent definition $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

A Helpful Analogy

$f(n) = O(g(n))$ is similar to $f(n) \leq g(n)$.

$f(n) = o(g(n))$ is similar to $f(n) < g(n)$.

$f(n) = \Theta(g(n))$ is similar to $f(n) = g(n)$.

$f(n) = \Omega(g(n))$ is similar to $f(n) \geq g(n)$.

$f(n) = \omega(g(n))$ is similar to $f(n) > g(n)$.

Transitivity

$$f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) \text{ imply } f(n) = \Theta(h(n)).$$

$$f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) \text{ imply } f(n) = O(h(n)).$$

$$f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) \text{ imply } f(n) = \Omega(h(n)).$$

$$f(n) = o(g(n)) \text{ and } g(n) = o(h(n)) \text{ imply } f(n) = o(h(n)).$$

$$f(n) = \omega(g(n)) \text{ and } g(n) = \omega(h(n)) \text{ imply } f(n) = \omega(h(n)).$$

Reflexivity

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

$$f(n) = O(f(n))$$

$$f(n) = \Omega(f(n))$$

Symmetry & Transpose Symmetry

Symmetry

$$f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n)).$$

Transpose Symmetry

$$f(n) = O(g(n)) \text{ if and only if } g(n) = \Omega(f(n)).$$
$$f(n) = o(g(n)) \text{ if and only if } g(n) = \omega(f(n)).$$

Non-completeness

Non-completeness of O, Ω, and Θ notations

For real numbers a and b, we know that either $a < b$, or $a = b$, or $a > b$ is true.

However, for two functions $f(n)$ and $g(n)$, it is possible that neither of the following is true: $f(n) = O(g(n))$, or $f(n) = \Theta(g(n))$, or $f(n) = \Omega(g(n))$. For example, $f(n) = n$, and $g(n) = n^{1-\sin(n\pi/2)}$.

Table of Contents

Floors and Ceilings

Floor

For any real number x, we denote the greatest integer less than or equal to x by $\lfloor x \rfloor$ (read "the floor of x")

Ceiling

For any real number x, we denote the least integer greater than or equal to x by $\lceil x \rceil$ (read "the ceiling of x")

$$x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil \leq x + 1.$$

For any integer n, $\lceil n/2 \rceil + \lfloor n/2 \rfloor = n$.

For any real number $x \geq 0$ and integers $a, b > 0$,

$$\left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil = \left\lceil \frac{x}{ab} \right\rceil, \left\lfloor \frac{\lfloor x/a \rfloor}{b} \right\rfloor = \left\lfloor \frac{x}{ab} \right\rfloor, \left\lceil \frac{a}{b} \right\rceil \leq \frac{a+(b-1)}{b}, \left\lfloor \frac{a}{b} \right\rfloor \geq \frac{a-(b-1)}{b},$$

Modular Arithmetic

Mod

For any integer a and any positive integer n, the value a mod n is the remainder (or residue) of the quotient a/n:

$$a \mod n = a - n\lfloor a/n \rfloor.$$

Equivalent

If $(a \mod n) = (b \mod n)$, we write $(a \equiv b) \mod n$ and say that a is equivalent to b, modulo n.

Exponentials

$\forall\, a > 0, \quad a^0 = 1; \quad (a^m)^n = (a^n)^m = a^{mn}; \quad a^m a^n = a^{m+n}$

When $a > 1$, $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$. That is, $n^b = o(a^n)$.

For all real x, $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$
When $|x| \le 1$, $1 + x \le e^x \le 1 + x + x^2$
When $x \to 0$, $e^x = 1 + x + \Theta(x^2)$
For all x, $\lim_{n\to\infty}(1 + \frac{x}{n})^n = e^x$

Logarithms

$\lg n = \log_2 n; \quad \ln n = \log_e n; \quad \lg^k n = (\lg n)^k; \quad \lg\lg n = \lg(\lg n)$

For all real $a, b, c > 0$, and n,
$a = b^{\log_b a}; \quad \log_c(ab) = \log_c a + \log_c b;$
$\log_b a^n = n \log_b a; \quad \log_b a = \frac{\lg a}{\lg b}; \quad a^{\log_b c} = c^{\log_b a}$

When $a > 0$, $\lim_{n\to\infty} \frac{\lg^b n}{(2^a)^{\lg n}} = \lim_{n\to\infty} \frac{\lg^b n}{n^a} = 0$. That is,
$\lg^b n = o(n^a)$.

When $|x| \leq 1$, $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \ldots$
For $x > -1$, $\frac{x}{1+x} \leq \ln(1+x) \leq x$

Factorials

$$n! = \begin{cases} 1 & \text{if} \quad n = 0 \\ n \cdot (n-1)! & \text{if} \quad n > 0 \end{cases}$$

$n! \leq n^n$. A better bound:

Stirling's approximation

$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \Theta\left(\frac{1}{n}\right)\right)$

Functional iteration

functional iteration

We use the notation $f^{(i)}(n)$ to denote the function $f(n)$ iteratively applied i times to an initial value of n. Formally, let $f(n)$ be a function over the reals. For non-negative integers i, we recursively define

$$f^{(i)}(n) = \begin{cases} n & \text{if } i = 0, \\ f(f^{(i-1)}(n)) & \text{if } i > 0, \end{cases}$$

Example: if $f(n) = 2n$, then $f^{(i)}(n) = 2^i n$.

The iterated logarithm function

We use the notation $\lg^* n$ to denote the iterated logarithm.

$$\lg^* n = \min\{i \geq 0 : \lg^{(i)} n \leq 1\}.$$

Example:

$$\lg^* 2 = 1,$$
$$\lg^* 4 = 2,$$
$$\lg^* 16 = 3,$$
$$\lg^*(2^{65536}) = 5.$$

Fibonacci Numbers

Fibonacci numbers

We define the Fibonacci numbers by the following recurrence:

$$F_0 = 0,$$
$$F_1 = 1,$$
$$F_i = F_{i-1} + F_{i-2}, \quad \text{for } i \geq 2.$$

Each Fibonacci number is the sum of the two previous ones, yielding the sequence

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Table of Contents

Solving Recurrences

Recurrences go hand in hand with the divide-and-conquer paradigm. A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.

Three methods for solving recurrences

- substitution method: guess a bound and use mathematical induction to prove the guess correct.
- recursion-tree method: converts the recurrence into a tree and use techniques for bounding summations.
- master method: provides bounds of the form $T(n) = a \cdot T(\frac{n}{b}) + f(n)$.

Substitution Method

The most general method

1. Guess the form of the solution.
2. Solve for constants.

- This method only works if we can guess the form of the answer.
- The method can be used to establish either upper or lower bounds on a recurrence.

Example of Substitution

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

- Assume that $T(1) = \Theta(1)$.
- Guess $T(n) = O(n^3)$. (Note that if we guess Θ, we need prove O and Ω separately.)
- Assume that $T(k) \le ck^3$ for $k < n$ and some constant $c > 0$.
- Prove $T(n) \le cn^3$ by induction.

Example of Substitution

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

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

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

This bound is not tight!

A Tighter Upper Bound?

We shall prove that $T(n) = O(n^2)$.

A Tighter Upper Bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \leq ck^2$ for $k < n$:

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

A Tighter Upper Bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \leq ck^2$ for $k < n$:

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

Wrong! We must prove the I.H.

A Tighter Upper Bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \leq ck^2$ for $k < n$:

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4c(n/2)^2 + n \\ &= cn^2 + n \\ &= \cancel{O(n^2)} \quad \text{Wrong! We must prove the I.H.} \\ &= cn^2 - (-n) \qquad \text{[desired } - \text{ residual]} \\ &\leq cn^2 \qquad \text{for no choice of } c > 0. \text{ Lose!} \end{aligned}$$

A Tighter Upper Bound!

IDEA: Strengthen the inductive hypothesis.

- Subtract a low-order term.
  Inductive hypothesis: $T(k) \leq c_1k^2 - c_2k$ for $k < n$

A Tighter Upper Bound!

IDEA: Strengthen the inductive hypothesis.

- Subtract a low-order term.
  Inductive hypothesis: $T(k) \le c_1k^2 - c_2k$ for $k < n$

$$\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4(c_1(n/2)^2 - c_2(n/2)) + n \\
&= c_1n^2 - 2c_2n + n \\
&= c_1n^2 - c_2n - (c_2n - n) \\
&\le c_1n^2 - c_2n \text{ if } c_2 \ge 1
\end{aligned}$$

Pick c_1 big enough to handle the initial conditions.

A Tighter Lower Bound

We shall prove that $T(n) = \Omega(n^2)$.

A Tighter Lower Bound

We shall prove that $T(n) = \Omega(n^2)$.

Assume that $T(k) \geq ck^2$ for $k < n$, and for some chosen constant c.

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

Recursion-tree Method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- The recursion-tree method can be unreliable.
- The recursion tree method is good for generating guesses for the substitution method.

Example of Recursion Tree

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

$$T(n)$$

Example of Recursion Tree

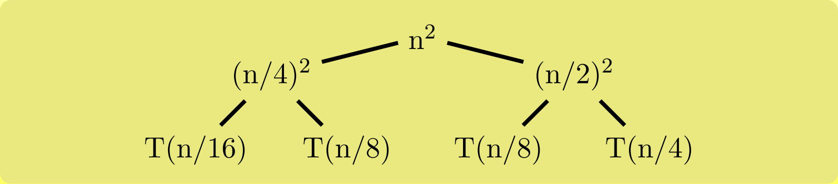
Solve $T(n) = T(n/4) + T(n/2) + n^2$:

$$n^2$$

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

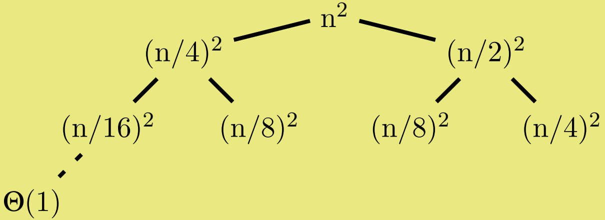
Example of Recursion Tree

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



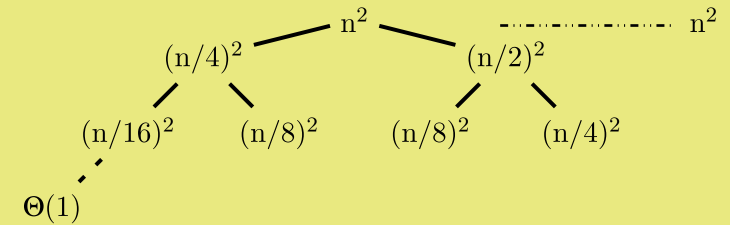
Example of Recursion Tree

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



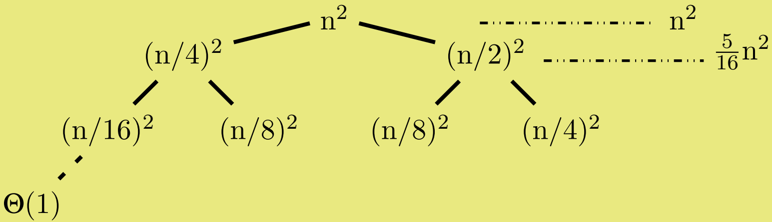
Example of Recursion Tree

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



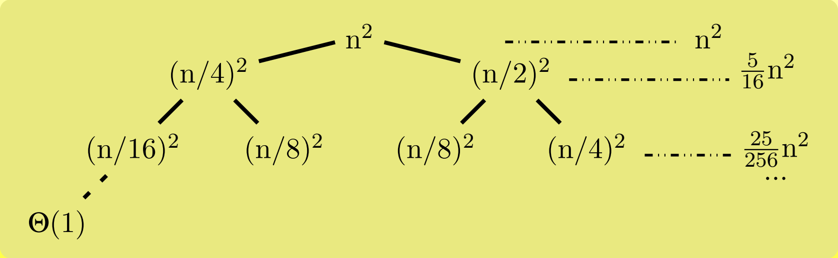
Example of Recursion Tree

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



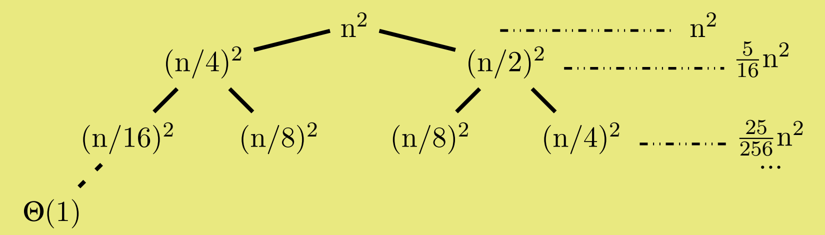
Example of Recursion Tree

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)

The Master Method

Master method

The master method applies to recurrences of the form

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

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

Three Common Cases

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

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

Three Common Cases

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

1. $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ for some constant $\varepsilon > 0$
   - $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^{ε} factor).
     Solution: $T(n) = \Theta\left(n^{\log_b a}\right)$.
2. $f(n) = \Theta\left(n^{\log_b a} \lg^k n\right)$ for some constant $k \geq 0$
   - $f(n)$ and $n^{\log_b a} \lg^k n$ grow at similar rates.
     Solution: $T(n) = \Theta\left(n^{\log_b a} \lg^{k+1} n\right)$.

Three Common Cases

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

3. $f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right)$ 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 $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n.
     Solution: $T(n) = \Theta(f(n))$.

Examples

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

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

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

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

Examples

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

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

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

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

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

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

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

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

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

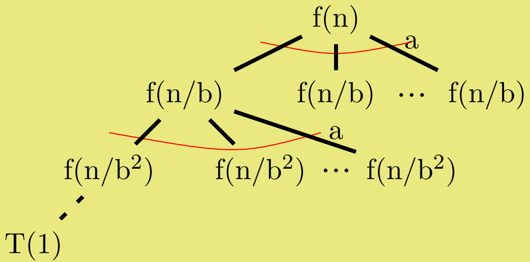
Examples

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: $f(n) = \Omega\left(n^{2+\varepsilon}\right)$ for $\varepsilon = 1$
and $4(n/2)^3 \leq cn^3$(reg. cond.) for $c = 1/2$.
$\therefore T(n) = \Theta\left(n^3\right)$.

Ex. $T(n) = 4T(n/2) + n^2/\lg n$
$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2;\quad f(n) = n^2/\lg n$.
Master method does not apply. In particular, for every constant $\varepsilon > 0$, we have $n^{\varepsilon} = \omega(\lg n)$.

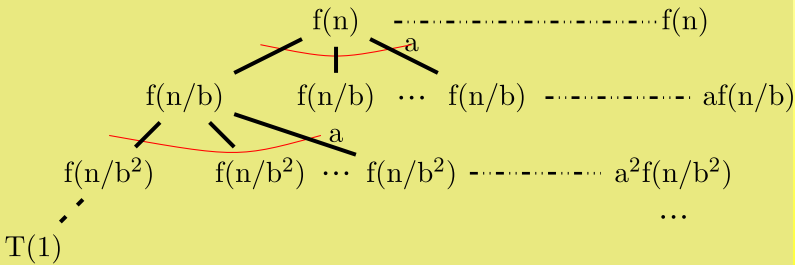
Idea of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$. Recursion tree:



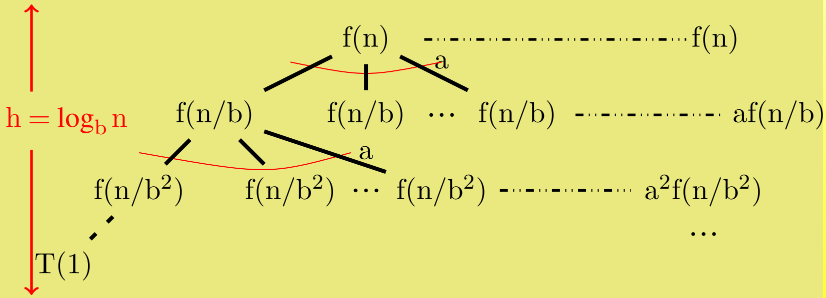
Idea of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$. Recursion tree:



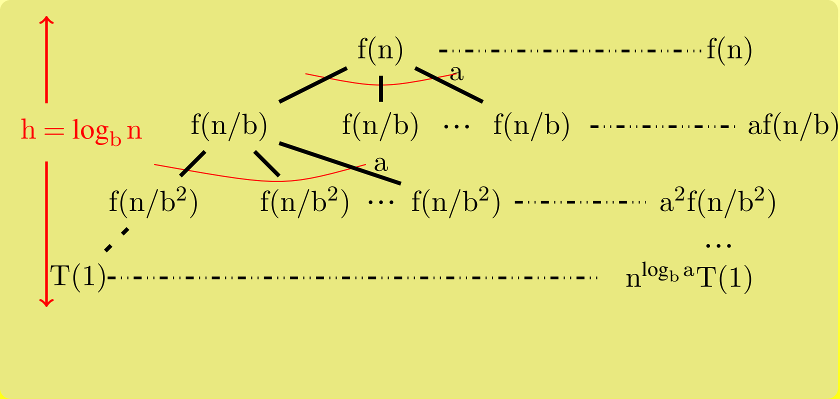
Idea of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$. Recursion tree:



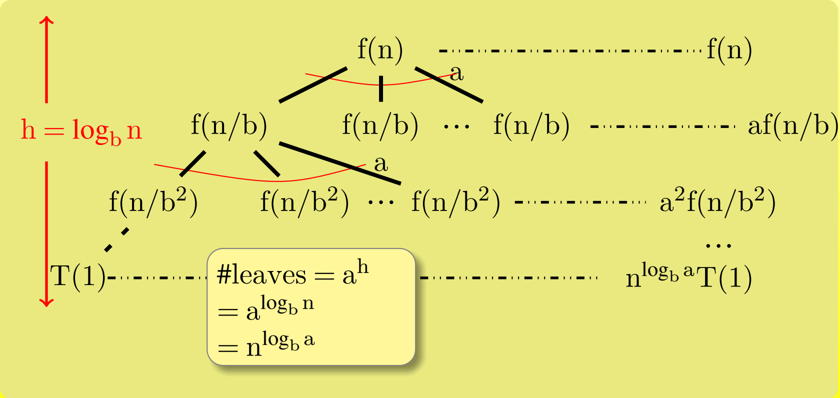
Idea of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$. Recursion tree:



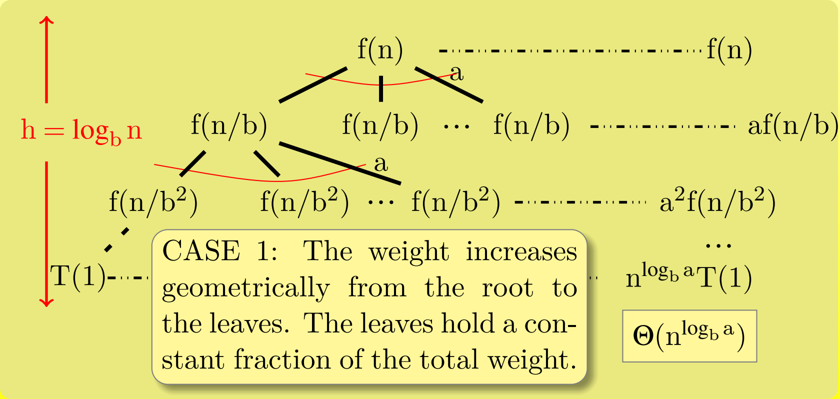
Idea of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$. Recursion tree:



Idea of Master Theorem

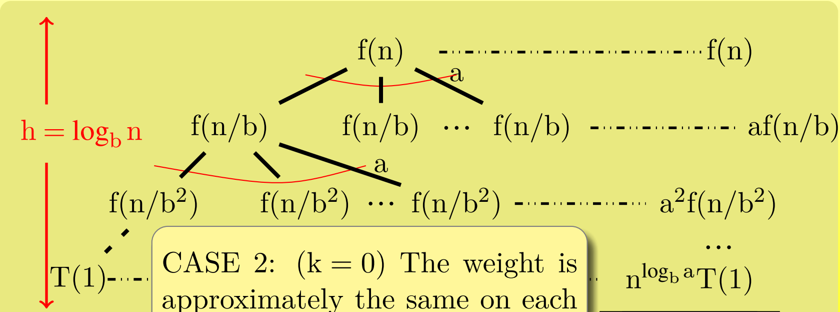
$T(n) = aT(\frac{n}{b}) + f(n)$. Recursion tree:



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

Idea of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$. Recursion tree:

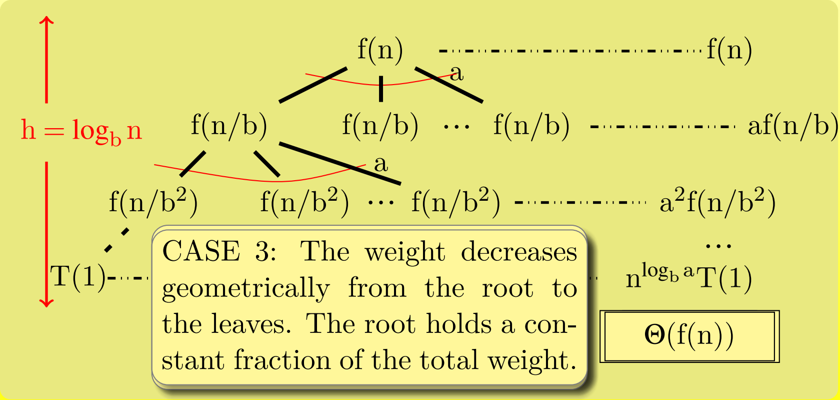


CASE 2: $(k = 0)$ The weight is approximately the same on each of the $\log_b n$ levels.

$\boxed{\Theta(n^{\log_b a} \lg n)}$

Idea of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$. Recursion tree:



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

Appendix: Geometric Series

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

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