

# 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 2022

# Outline of Topics

- 1 Asymptotic Notation:  $O$ -,  $\Omega$ - and  $\Theta$ -otation
  - $O$ -otation
  - $\Omega$ -otation
  - $\Theta$ -otation
  - Other Asymptotic Notations
  - Comparing Functions
- 2 Standard Notations and Common Functions
- 3 Recurrences
  - Substitution Method
  - Recursion Tree
  - Master Method

# Table of Contents

## 1 Asymptotic Notation: $O$ -, $\Omega$ - and $\Theta$ -otation

- $O$ -otation
- $\Omega$ -otation
- $\Theta$ -otation
- Other Asymptotic Notations
- Comparing Functions

## 2 Standard Notations and Common Functions

## 3 Recurrences

- Substitution Method
- Recursion Tree
- Master Method

# 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  $0 \leq f(n) \leq cg(n)$  for all  $n \geq n_0$ .

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

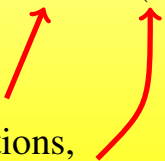
## Asymptotic Notation: $O$ -notation

### $O$ -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:**  $2n^2 = O(n^3)$  ( $c = 1, n_0 = 2$ )

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

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:**  $2n^2 \in O(n^3)$

# Macro Substitution

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

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

# Asymptotic Notation: $\Omega$ -notation

$O$ -notation is an upper-bound notation.

The  $\Omega$ -notation provides a lower bound.

Set definition of  $\Omega$ -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: $\Omega$ -notation

$O$ -notation is an upper-bound notation.  
The  $\Omega$ -notation provides a lower bound.

Set definition of  $\Omega$ -notation

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

**Example:**

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

## Asymptotic Notation: $\Theta$ -notation

### $\Theta$ -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  $n \geq n_0$ .

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

# Asymptotic Notation: $\Theta$ -notation

## $\Theta$ -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  $n \geq n_0$ .

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

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

## Asymptotic Notation: $\Theta$ -notation

### $\Theta$ -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  $n \geq n_0$ .

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

**Example:**  $\frac{1}{2}n^2 - 2n = \Theta(n^2)$   
 $\Theta(n^0)$  or  $\Theta(1)$

## Asymptotic Notation: $\Theta$ -notation

### $\Theta$ -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  $n \geq n_0$ .

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

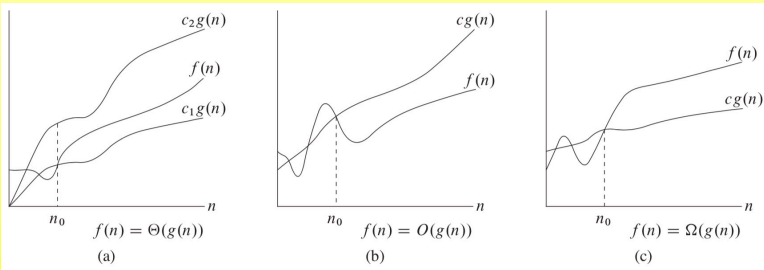
**Example:**  $\frac{1}{2}n^2 - 2n = \Theta(n^2)$   
 $\Theta(n^0)$  or  $\Theta(1)$

### Theorem:

The leading constant and low order terms do not matter.



# Graphic Examples of the $\Theta$ , $O$ , $\Omega$



## Other Asymptotic Notations

### $o$ -notation

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

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

### $\omega$ -notation

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

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

## 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

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

$f(n) = O(g(n))$  and  $g(n) = O(h(n))$  imply  $f(n) = O(h(n))$ .

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

$f(n) = o(g(n))$  and  $g(n) = o(h(n))$  imply  $f(n) = o(h(n))$ .

$f(n) = \omega(g(n))$  and  $g(n) = \omega(h(n))$  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

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

## Transpose Symmetry

$f(n) = O(g(n))$  if and only if  $g(n) = \Omega(f(n))$ .

$f(n) = o(g(n))$  if and only if  $g(n) = \omega(f(n))$ .

# Non-completeness

## Non-completeness of $O$ , $\Omega$ , and $\Theta$ 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

## 1 Asymptotic Notation: $O$ -, $\Omega$ - and $\Theta$ -otation

- $O$ -otation
- $\Omega$ -otation
- $\Theta$ -otation
- Other Asymptotic Notations
- Comparing Functions

## 2 Standard Notations and Common Functions

## 3 Recurrences

- Substitution Method
- Recursion Tree
- Master Method



# 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  $x$ ”)

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

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

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

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

# Modular Arithmetic

## Mod

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

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

## Equivalent

If  $(a \bmod n) = (b \bmod n)$ , we write  $(a \equiv b) \bmod n$  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  $a > 1$ ,  $\lim_{n \rightarrow \infty} \frac{n^b}{a^n} = 0$ . That is,  $n^b = o(a^n)$ .

For all real  $x$ ,  $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$

When  $|x| \leq 1$ ,  $1 + x \leq e^x \leq 1 + x + x^2$

When  $x \rightarrow 0$ ,  $e^x = 1 + x + \Theta(x^2)$

For all  $x$ ,  $\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x$

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

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

For  $x > -1$ ,  $\frac{x}{1+x} \leq \ln(1+x) \leq x$

# Factorials

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

$n! \leq n^n$ . 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:

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

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

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



# Table of Contents

## 1 Asymptotic Notation: $O$ -, $\Omega$ - and $\Theta$ -otation

- $O$ -otation
- $\Omega$ -otation
- $\Theta$ -otation
- Other Asymptotic Notations
- Comparing Functions

## 2 Standard Notations and Common Functions

## 3 Recurrences

- Substitution Method
- Recursion Tree
- Master Method

# 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
$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n).$$

# Substitution Method

The most general method

1. **Guess** the form of the solution.
  2. **Verify** by induction.
  3. **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  $\Theta$ , we need prove  $O$  and  $\Omega$  separately.)
- Assume that  $T(k) \leq ck^3$  for  $k < n$  and some constant  $c > 0$ .
- Prove  $T(n) \leq cn^3$  by induction.

## Example of Substitution

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

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

$\nwarrow$  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  $1 \leq n < n_0$ , we have “ $\Theta(1)$ ”  $\leq cn^3$ , 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  $k < n$ :

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

$$= O(n^2)$$

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  $k < n$ :

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

$$\leq 4c(n/2)^2 + n$$

$$= cn^2 + n$$

$$= \cancel{O(n^2)}$$

$$= cn^2 - (-n) \quad \text{[desired - residual]}$$

$$\leq cn^2 \quad \text{for no choice of } c > 0. \text{ Lose!}$$

Wrong! We must prove the I.H.

## A Tighter Upper Bound!

**IDEA:** Strengthen the inductive hypothesis.

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

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

## A Tighter Upper Bound!

**IDEA:** Strengthen the inductive hypothesis.

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

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

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

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.



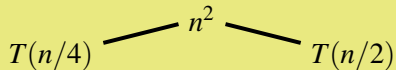
## Example of Recursion Tree

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

$$T(n)$$

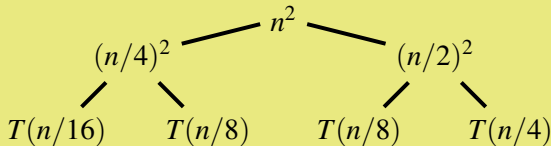
## Example of Recursion Tree

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



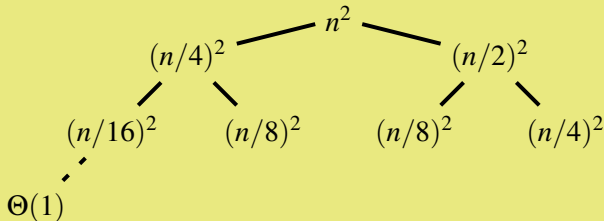
## Example of Recursion Tree

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



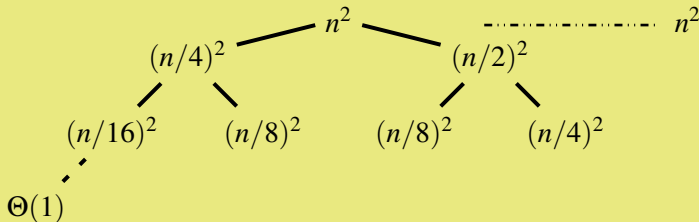
## Example of Recursion Tree

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



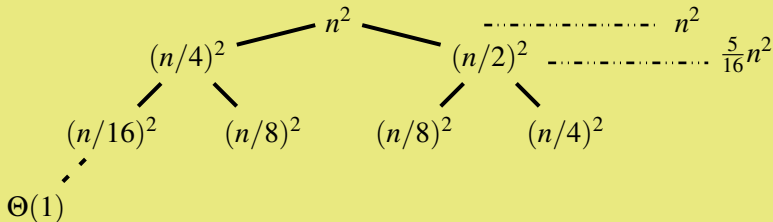
## Example of Recursion Tree

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

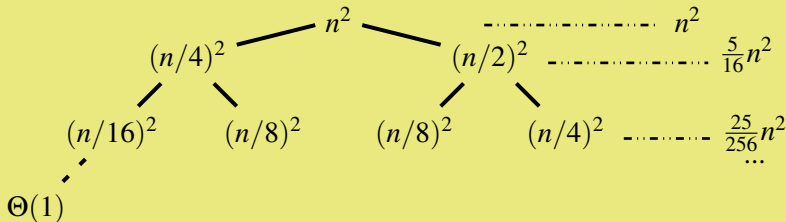


## Example of Recursion Tree

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

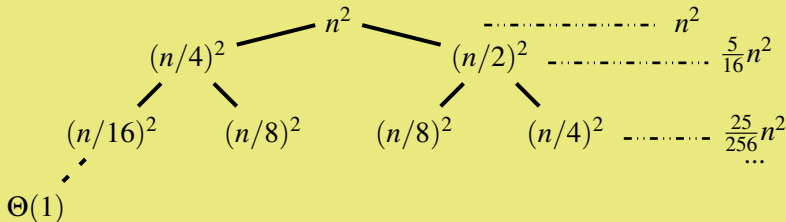


Solve  $T(n) = T(n/4) + T(n/2) + n^2$ :



## Example of Recursion Tree

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



$$\text{Total} = n^2(1 + \frac{5}{16} + (\frac{5}{16})^2 + (\frac{5}{16})^3 + \dots) = \Theta(n^2)$$

(geometric series)



# The Master Method

## Master method

The master method applies to recurrences of the form

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

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

## Three Common Cases

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

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

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

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

## Three Common Cases

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

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

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

2.  $f(n) = \Theta(n^{\log_b a} \lg^k n)$  for some constant  $k \geq 0$

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

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

## Three Common Cases

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  $n^\varepsilon$  factor),  
and  $f(n)$  satisfies the **regularity condition** that  
 $af(n/b) \leq cf(n)$  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:**  $f(n) = O(n^{2-\varepsilon})$  for  $\varepsilon = 1$

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

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

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

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

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

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

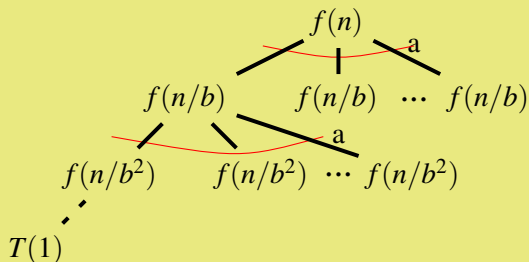
$$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2/\lg n.$$

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



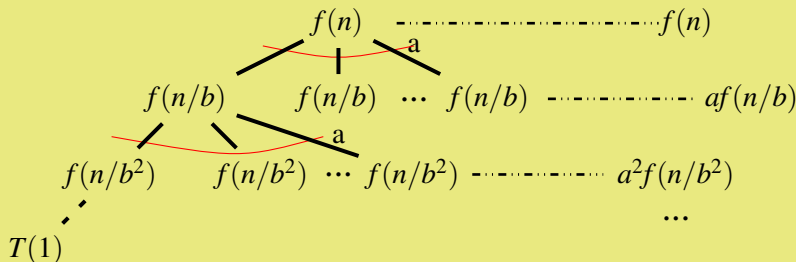
# Idea of Master Theorem

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



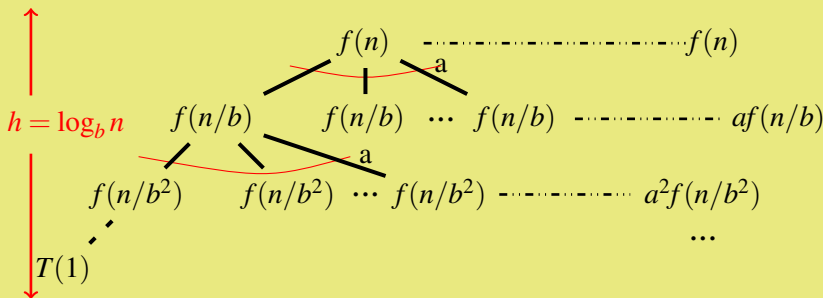
## Idea of Master Theorem

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



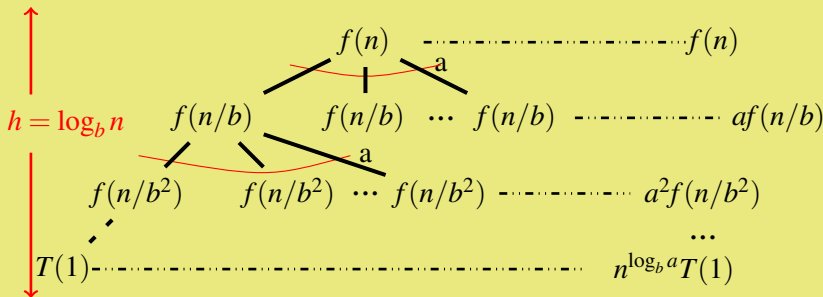
# Idea of Master Theorem

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



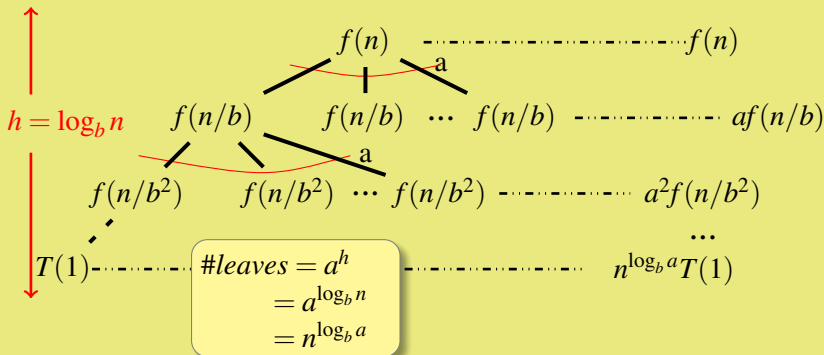
# Idea of Master Theorem

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



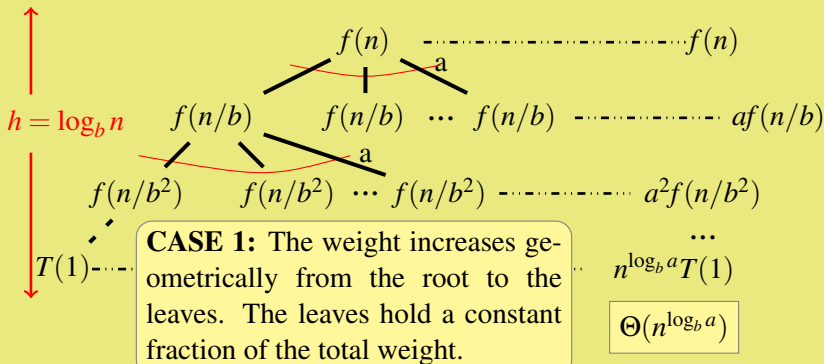
## Idea of Master Theorem

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



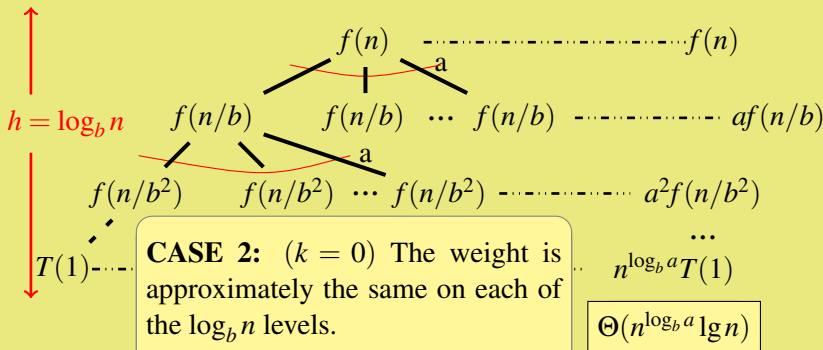
# Idea of Master Theorem

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



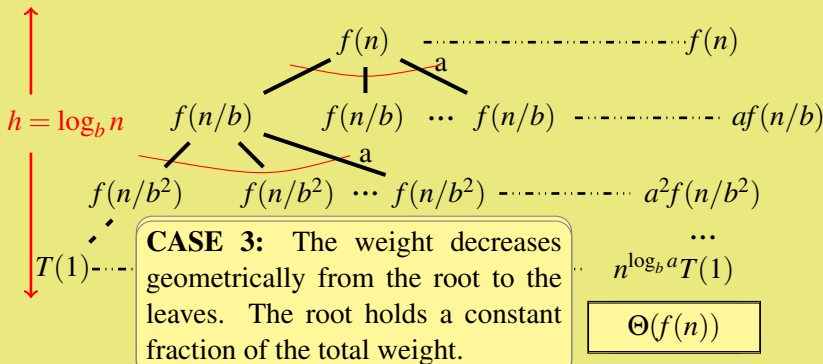
# Idea of Master Theorem

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



# Idea of Master Theorem

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





## Appendix: Geometric Series

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

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