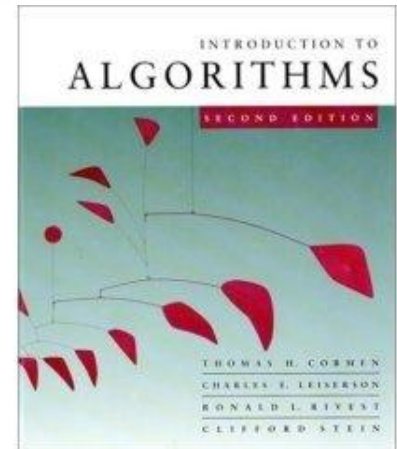




# Analisi della complessità

Algoritmi, Strutture Dati e Calcolo Parallelo

- ❑ Questo materiale è tratto dalle trasparenze del corso Introduction to Algorithms (2005-fall-6046) tenuto dal Prof. Leiserson all' MIT (<http://people.csail.mit.edu/cel/>)
- ❑ T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein Introduction to Algorithms, Second Edition, The MIT Press, Cambridge, Massachusetts London, England McGraw-Hill Book Company
- ❑ Queste trasparenze sono disponibili sui siti <http://www.pierlucalanzi.net>  
<http://www.slideshare.net/pierluca.lanzi>



Nella lezione precedente...

# Algoritmi, struttura dati e analisi (asintotica) della performance

Notazione  $O$ ,  $\Omega$ , e  $\Theta$

Come calcolare  $T(n)$ ?  
Analiticamente o sviluppando l'equazione

In questa lezione vediamo altri tre metodi:  
Sostituzione, Alberi, & Master Method

Metodo per sostituzione

# Analisi di Algoritmi Ricorsivi: Metodo per Sostituzione

Forse il metodo più generale,  
si può applicare sempre

1. "Indovinare" la forma della soluzione
2. Verificare la soluzione per induzione
3. Calcolare eventuali costanti

- Esempio:  $T(n) = 4T(n/2) + n$ 
  - ▶ Assumiamo che  $T(1) = \Theta(1)$
  - ▶ Ipotizziamo ad esempio che  $T(n)$  sia  $O(n^3)$
  - ▶ Assumendo che  $T(k) \leq ck^3$  for  $k < n$ ,  
dimostramo per induzione che  $T(n) \leq cn^3$

## □ Soluzione

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

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

- ❑ Infine dobbiamo considerare le condizioni iniziali che sono alla base della dimostrazione per induzione
- ❑ In questo caso,  $T(1) = \Theta(1)$  per  $n < n_0$ , per un opportuno  $n_0$
- ❑ Per  $1 \leq n < n_0$ , abbiamo " $\Theta(1)$ "  $\leq cn^3$ , se  $c$  è scelto opportunamente grande

Questo vincolo non è stretto

- Proviamo a dimostrare che  $T(n)$  è  $O(n^2)$
- Assumiamo  $T(n) \leq ck^2$  per  $k < n$ , dimostriamolo per  $n$

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

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

$$= cn^2 + n$$

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

$$= cn^2 - (-n) \quad \text{bound desiderato (- residuo)}$$

$$\leq cn^2 \quad \text{Falso! Per nessuna scelta di } c \text{ può essere } \leq cn^2$$

- È però possibile migliorare il limite dimostrando che  $T(n) \leq c_1n^2 - c_2n$

Alberi di ricorsione

## Alberi di Ricorsione

Calcoliamo la complessità andando a sviluppare l'esecuzione su una struttura ad albero

Considerato inaffidabile, è però molto intuitivo

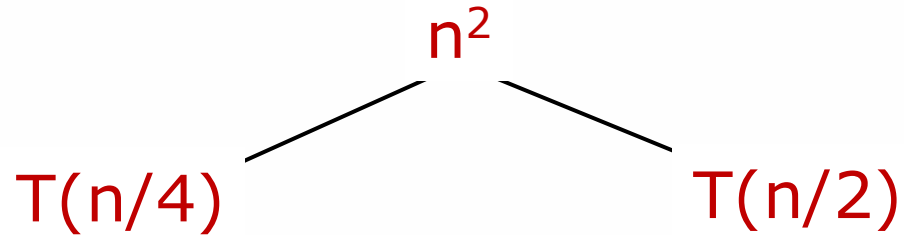
Calcoliamo  $T(n) = T(n/4) + T(n/2) + n^2$ :

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

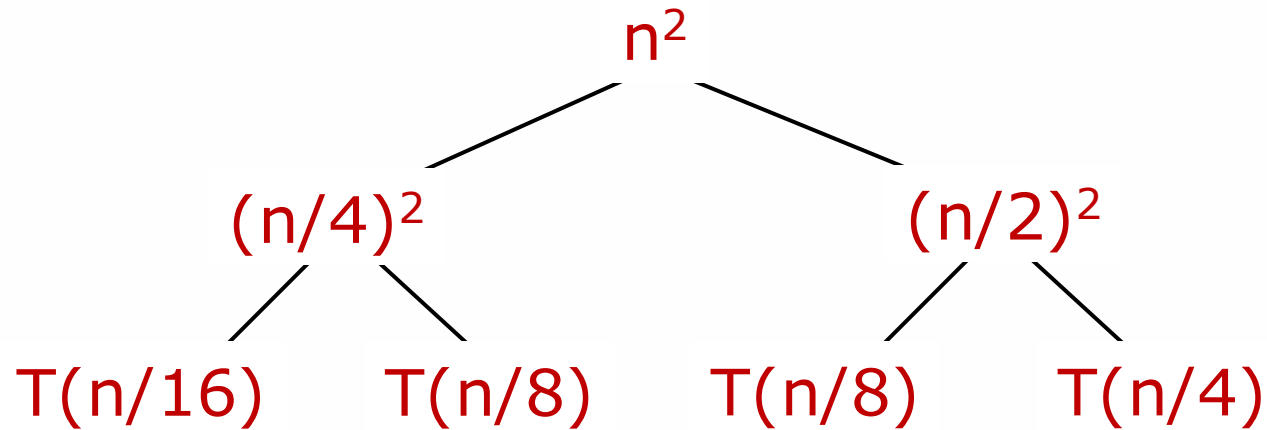
$$T(n)$$

## Esempio

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

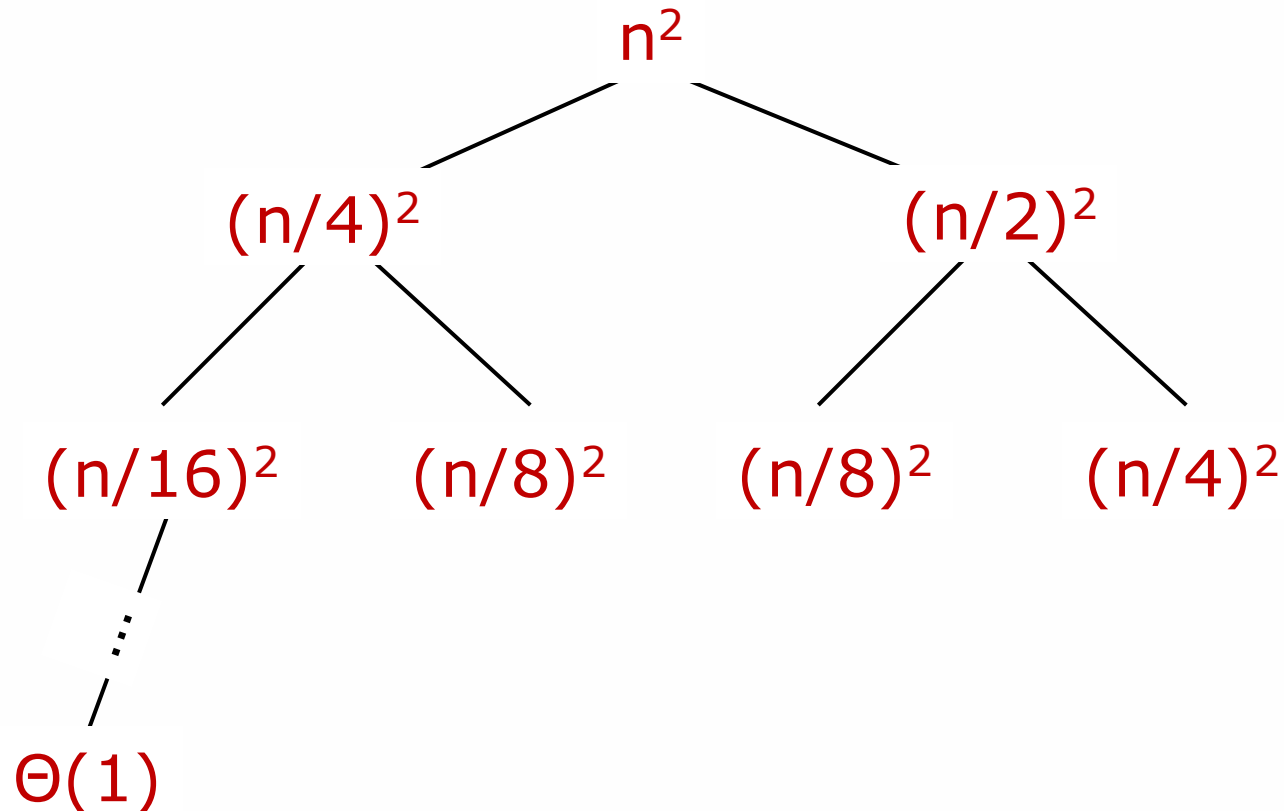


Calcoliamo  $T(n) = T(n/4) + T(n/2) + n^2$ :



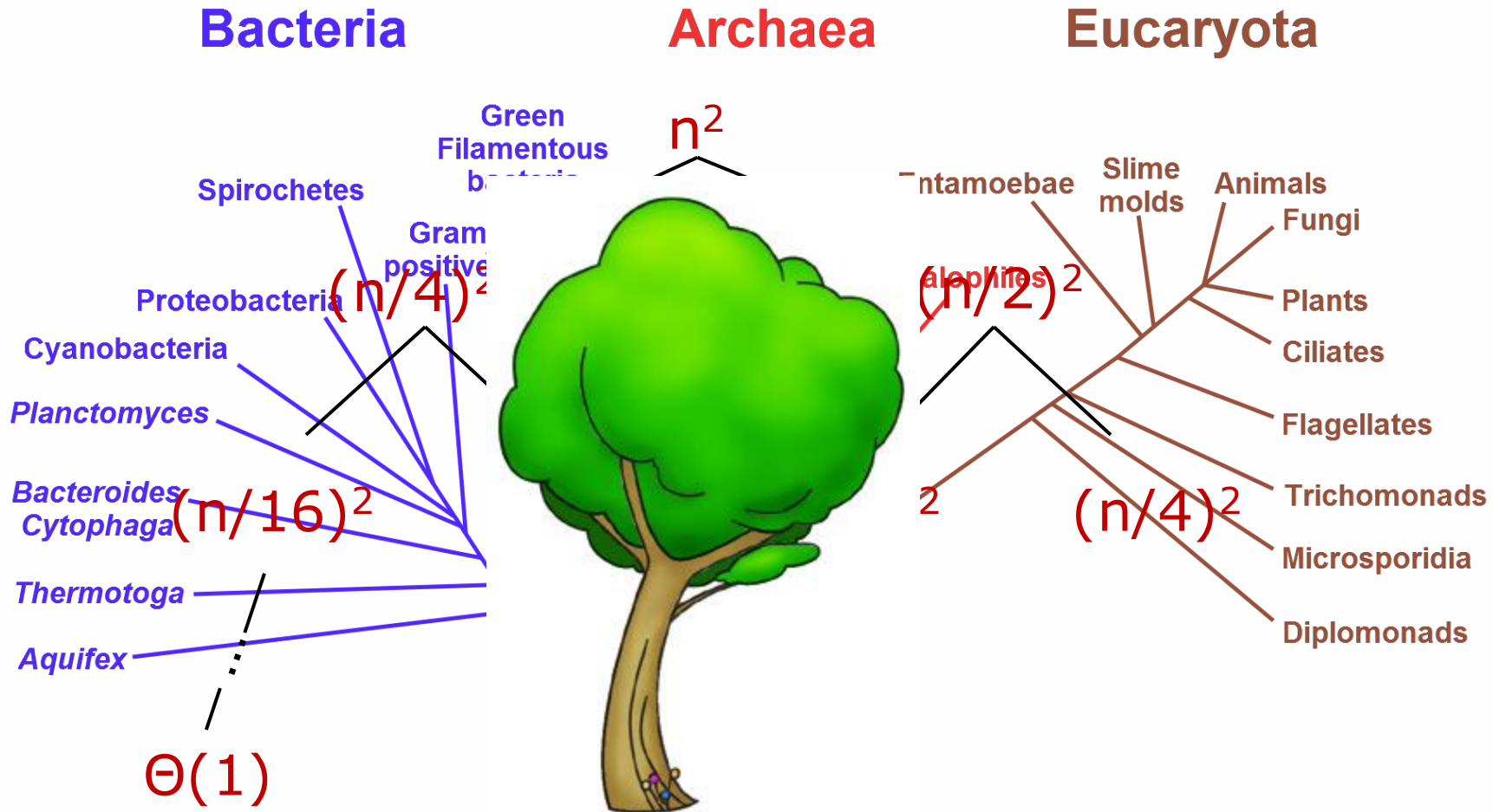
# Esempio

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



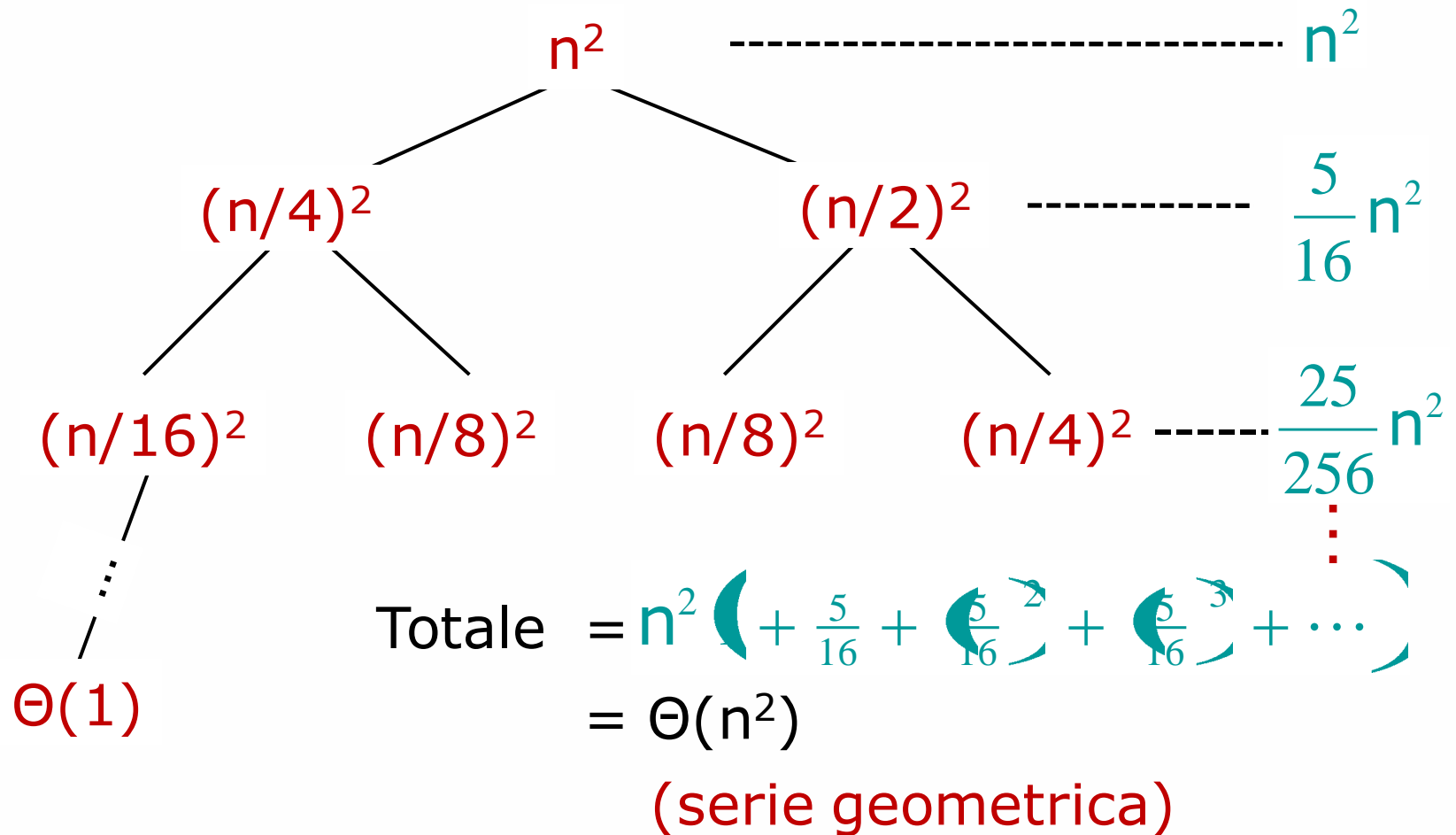
Perchè si chiama "albero"?

# Phylogenetic Tree of Life



# Esempio

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



Master method

- Si applica a forme ricorsive del tipo,

$$T(n) = aT(n/b) + f(n)$$

dove  $a \geq 1$ ,  $b > 1$ , ed  $f$  è asintoticamente positiva

- Si confronta  $f(n)$  con  $n^{\log_b a}$
- Tre casi possibili
  - ▶  $f(n)$  cresce più lentamente di  $n^{\log_b a}$
  - ▶  $f(n)$  cresce in maniera simile a  $n^{\log_b a}$
  - ▶  $f(n)$  cresce più velocemente di  $n^{\log_b a}$

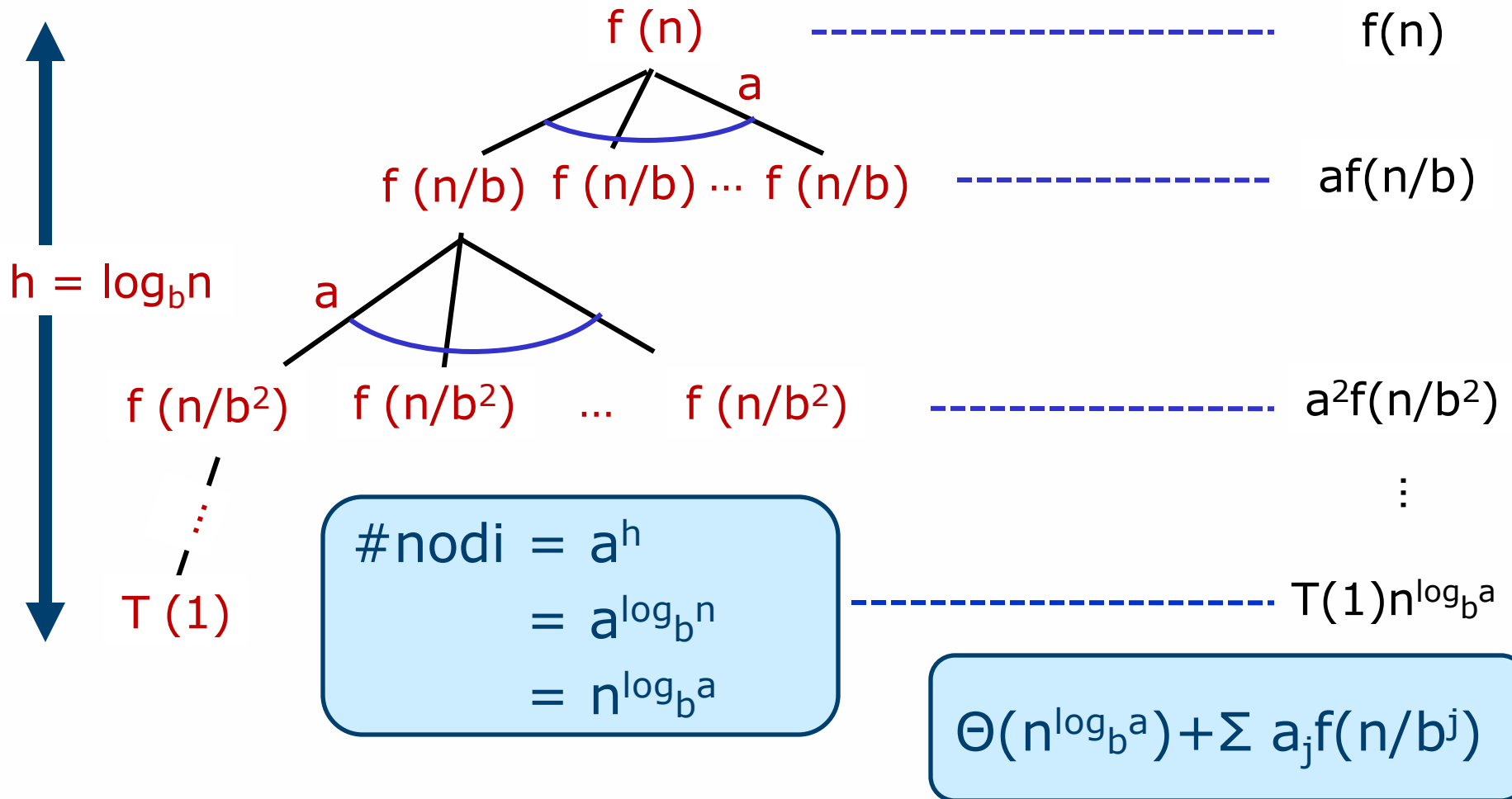
- Caso 1:  $f(n)$  cresce più lentamente di  $n^{\log_b a}$ 
  - ▶  $f(n) = O(n^{\log_b a - \varepsilon})$  per una costante  $\varepsilon > 0$
  - ▶ **Soluzione:**  $T(n) = \Theta(n^{\log_b a})$
  
- Caso 2:  $f(n)$  cresce in maniera simile a  $n^{\log_b a}$ 
  - ▶  $f(n) = \Theta(n^{\log_b a} \lg^k n)$  per una costante  $k \geq 0$
  - ▶ **Soluzione:**  $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$
  
- Caso 3:  $f(n)$  cresce più velocemente di  $n^{\log_b a}$ 
  - ▶  $f(n) = \Omega(n^{\log_b a + \varepsilon})$  per una costante  $\varepsilon > 0$
  - ▶  $f(n)$  cresce più velocemente di  $n^{\log_b a}$  (di un fattore  $n^\varepsilon$ ), ed  $f(n)$  soddisfa la condizione  $af(n/b) \leq cf(n)$  per una costante  $c < 1$
  - ▶ **Soluzione:**  $T(n) = \Theta(f(n))$

- Esempio 1:  $T(n) = 4T(n/2) + n$ 
  - ▶  $a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n$
  - ▶  $T(n) = \Theta(n^2)$
  
- Esempio 2:  $T(n) = 4T(n/2) + n^2$ 
  - ▶  $a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2$
  - ▶  $f(n) = \Theta(n^2 \lg^0 n)$ , ovvero,  $k = 0$
  - ▶  $T(n) = \Theta(n^2 \lg n)$
  
- Esempio 3:  $T(n) = 4T(n/2) + n^3$ 
  - ▶  $a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^3$
  - ▶  $f(n) = \Omega(n^{2+\varepsilon})$  con  $\varepsilon = 1$
  - ▶  $T(n) = \Theta(n^3)$

- Esempio 4:  $T(n) = 4T(n/2) + n/\lg n$ 
  - ▶  $a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n/\lg n$
  - ▶ **Il master method non si applica**

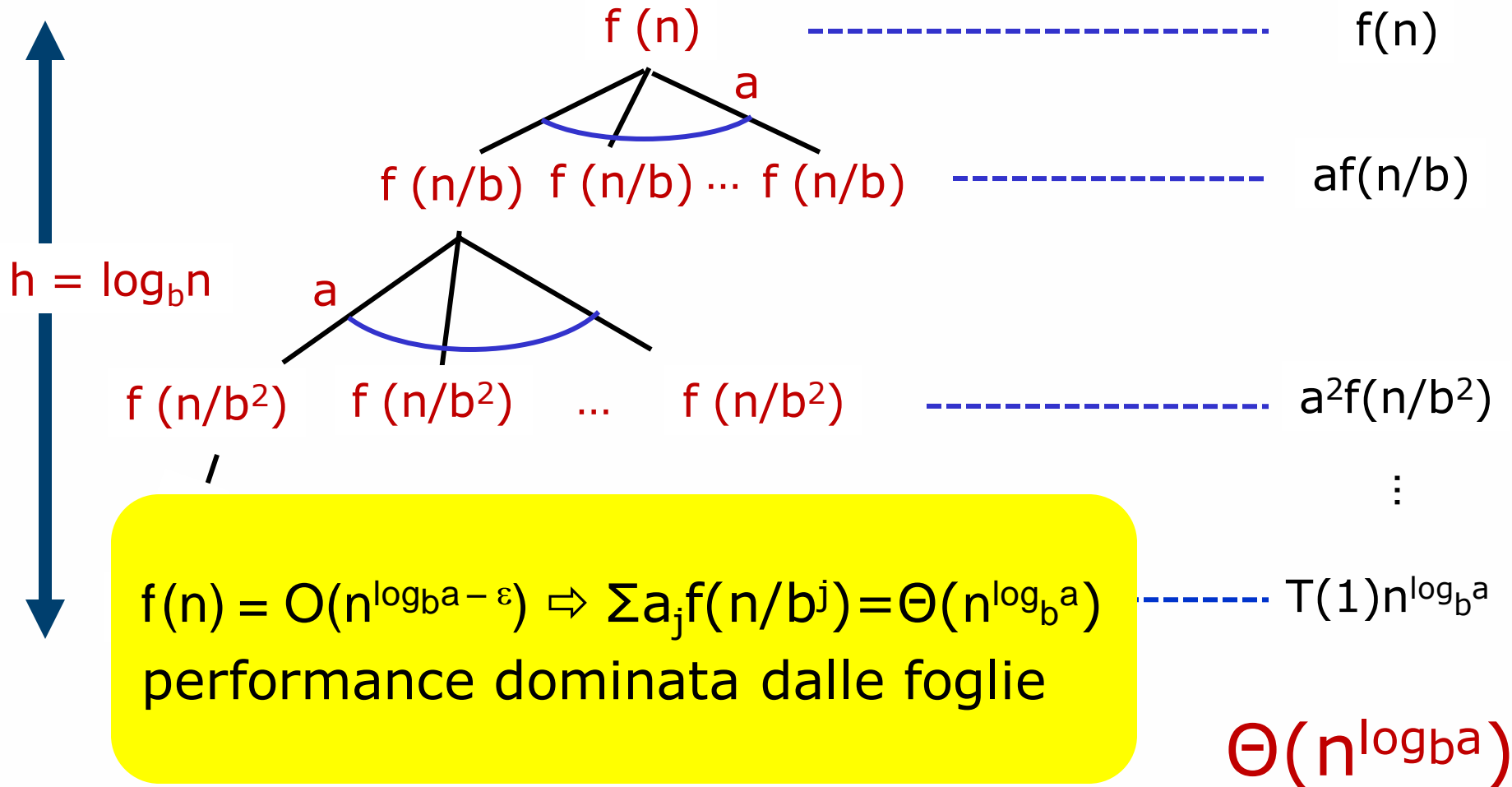
# Qual è l'Idea del Master Method?

## Albero di ricorsione



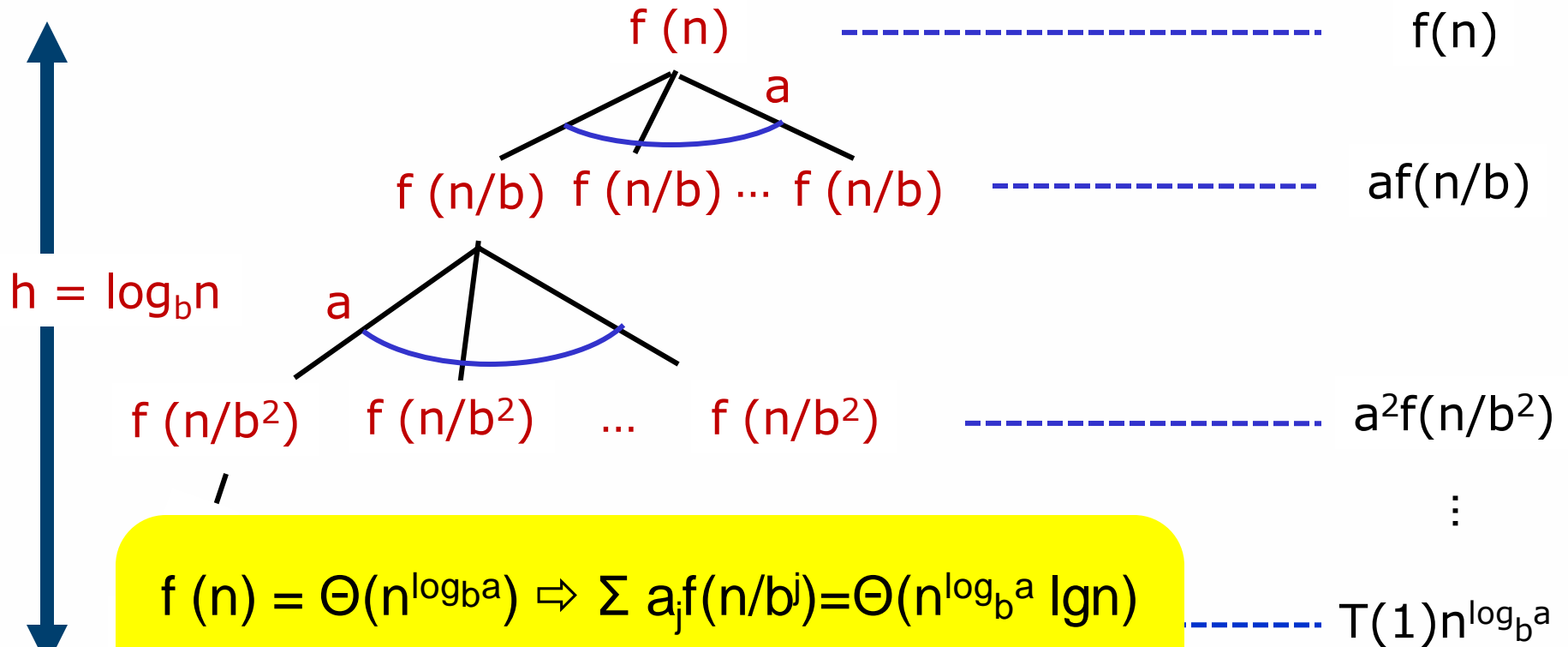
# Master Method: Caso 1

## Albero di ricorsione



# Master Method: Caso 2

## Albero di ricorsione

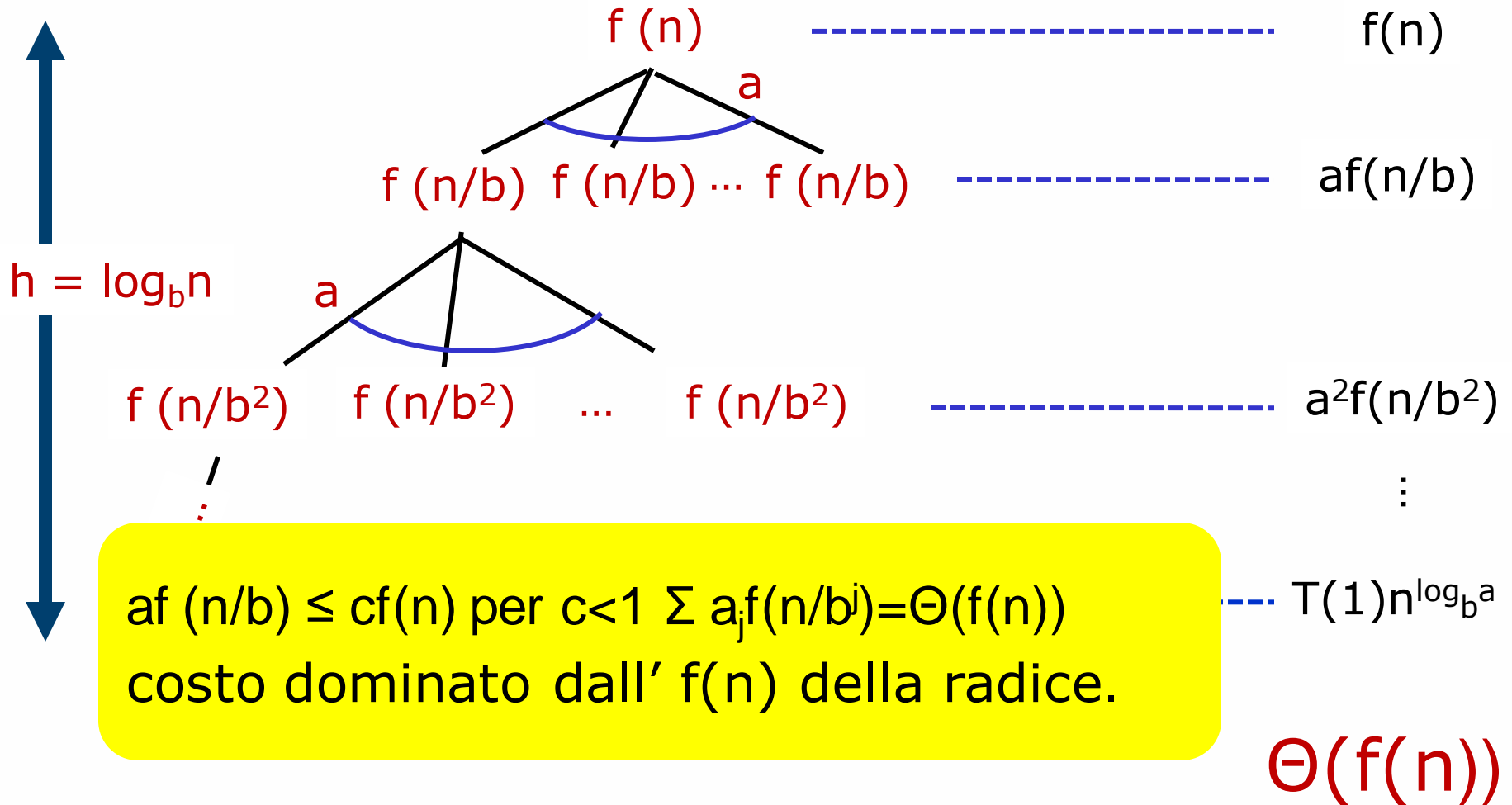


$f(n) = \Theta(n^{\log_b a}) \Leftrightarrow \sum a^j f(n/b^j) = \Theta(n^{\log_b a} \lg n)$   
costo distribuito uniformemente su  
tutti i  $\log_b n$  livelli

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

# Master Method: Caso 3

## Albero di ricorsione



Sommario

- ❑ Tre metodi per risolvere
  - ▶ Sostituzione
  - ▶ Alberi
  - ▶ Master method
  
- ❑ Sostituzione
  - ▶ Intuizione
  - ▶ Dimostrazione per induzione
  - ▶ Generale, sempre applicabile
  
- ❑ Master method
  - ▶ Applicabile a forme ricorsive del tipo  $T(n) = aT(n/b) + f(n)$  con  $a \geq 1$ ,  $b > 1$ , ed  $f$  asintoticamente positiva
  - ▶ Diretto, ma limitato a questo tipo di equazioni

## Appendice: Serie Geometriche

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

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