



---

# **CSE408**

# **Recurrence equations**

---

**Lecture #12**

# Substitution method



*The most general method:*

- 1. Guess** the form of the solution.
- 2. Verify** by induction.
- 3. Solve** for constants.

*The most general method:*

- 1. Guess** the form of the solution.
- 2. Verify** by induction.
- 3. Solve** for constants.

**EXAMPLE:**  $T(n) = 4T(n/2) + n$

- [Assume that  $T(1) = \Theta(1)$ .]
- Guess  $O(n^3)$ . (Prove  $O$  and  $\Omega$  separately.)
- Assume that  $T(k) \leq ck^3$  for  $k < n$ .
- Prove  $T(n) \leq cn^3$  by induction.

# Example of substitution



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

whenever  $(c/2)n^3 - n \geq 0$ , for  
example, if  $c \geq 2$  and  $n \geq 1$ .  
*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  $1 \leq n < n_0$ , 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 recursion tree models the costs (time) of a recursive execution of an algorithm.
- The recursion-tree method can be unreliable, just like any method that uses ellipses (...).
- The recursion-tree method promotes intuition, however.
- The recursion tree method is good for generating guesses for the substitution method.

# Example of recursion tree



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



# Example of recursion tree

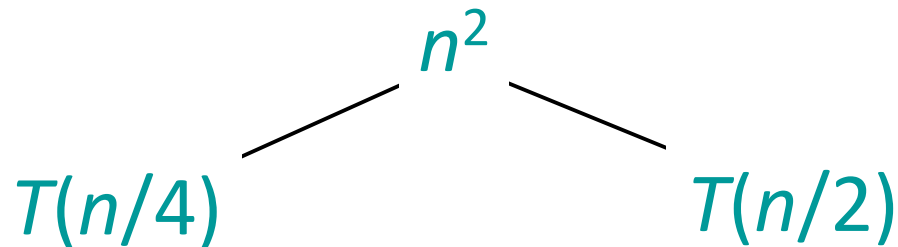


Solve  $T(n) = T(n/4) + T(n/2) + n^2$ :  
 $T(n)$

# Example of recursion tree



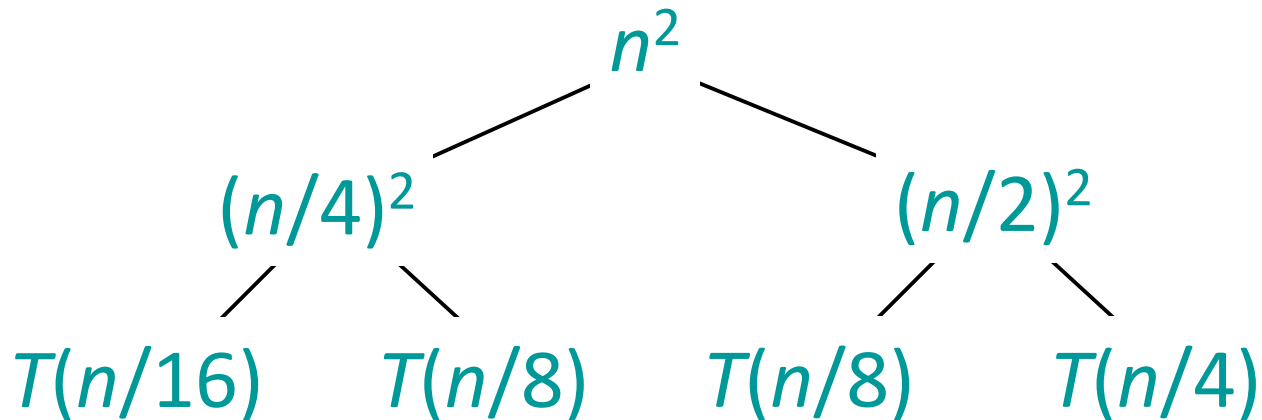
Solve  $T(n) = T(n/4) + T(n/2) + n^2$ :



# Example of recursion tree



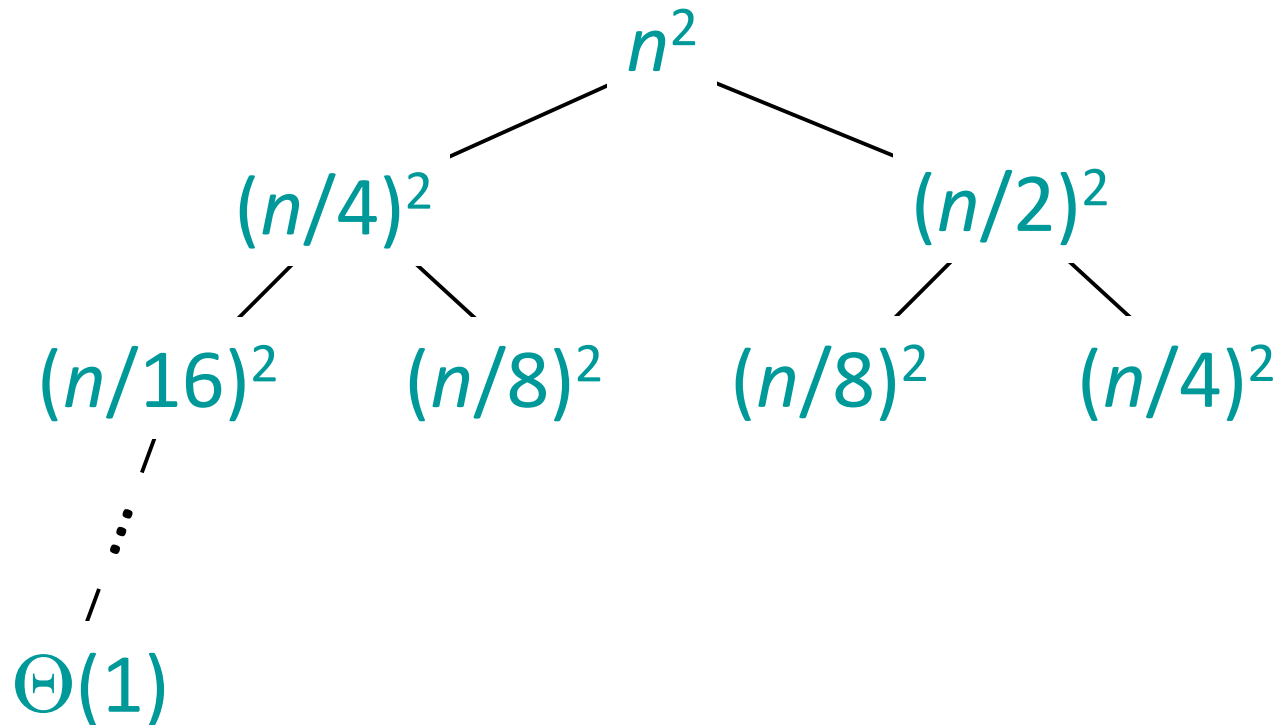
Solve  $T(n) = T(n/4) + T(n/2) + n^2$ :



# Example of recursion tree



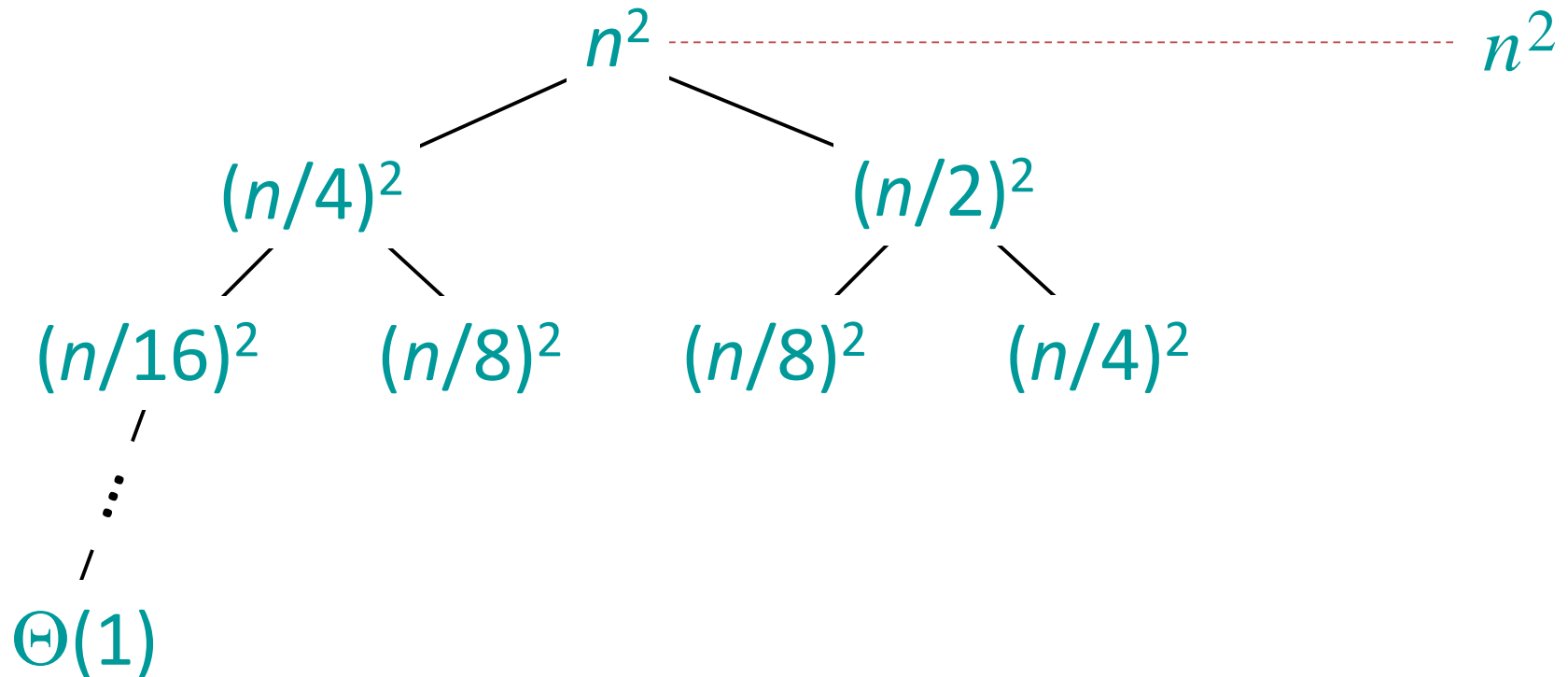
Solve  $T(n) = T(n/4) + T(n/2) + n^2$ :



# Example of recursion tree



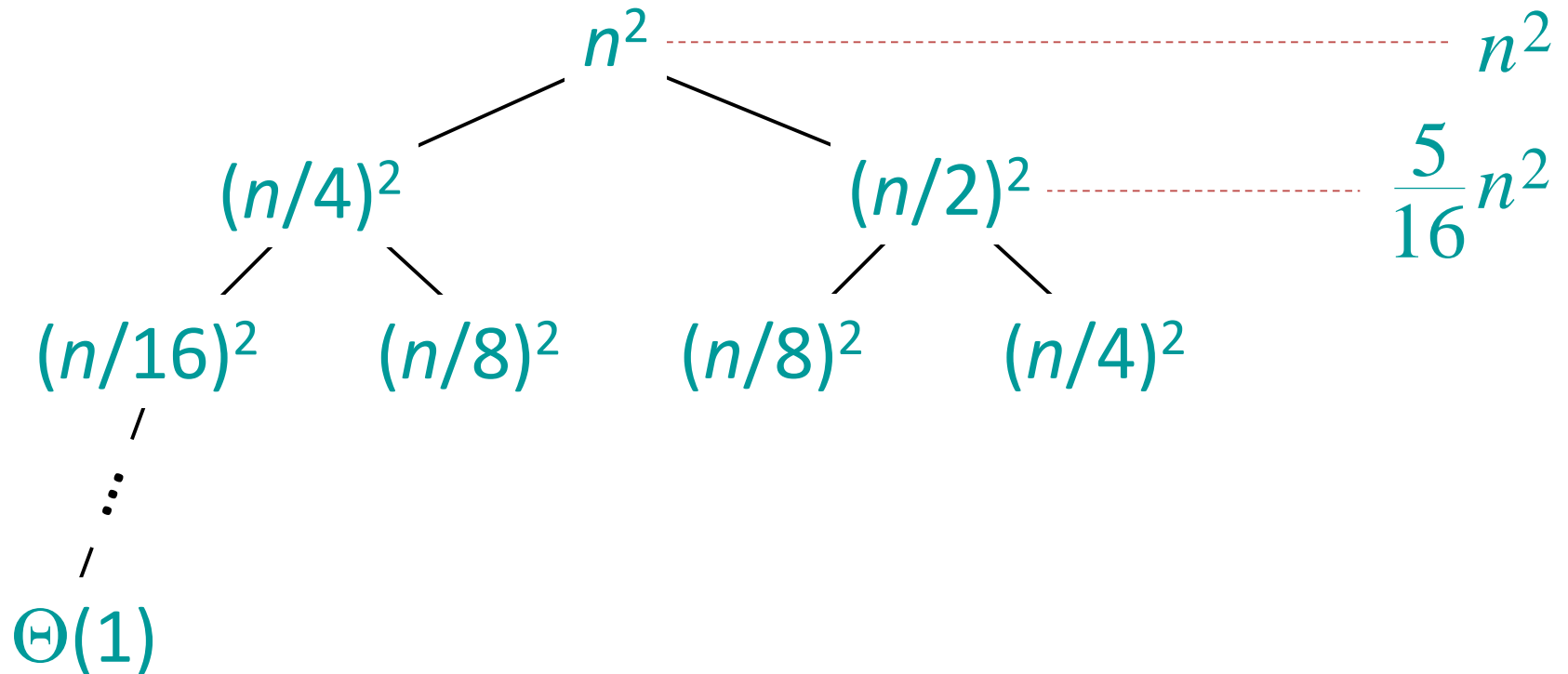
Solve  $T(n) = T(n/4) + T(n/2) + n^2$ :



# Example of recursion tree



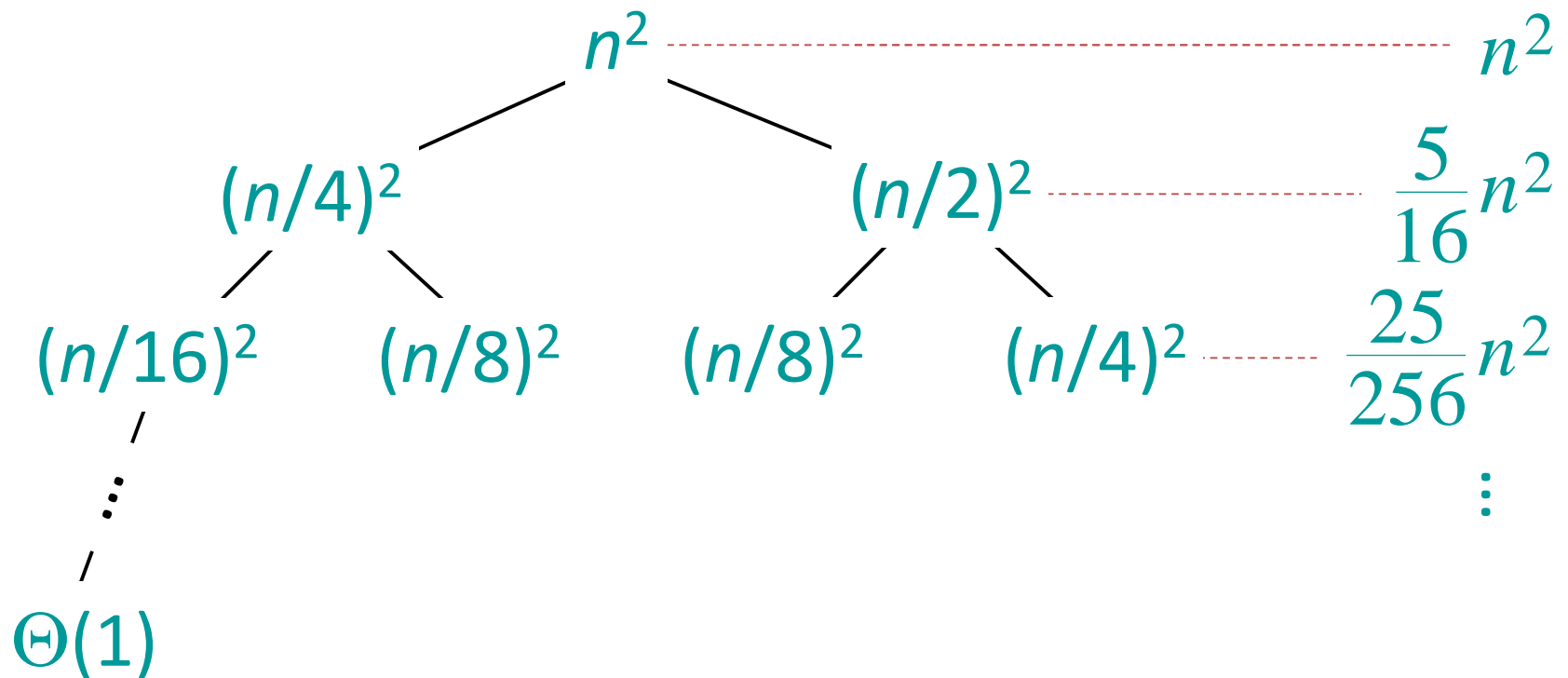
Solve  $T(n) = T(n/4) + T(n/2) + n^2$ :



# Example of recursion tree



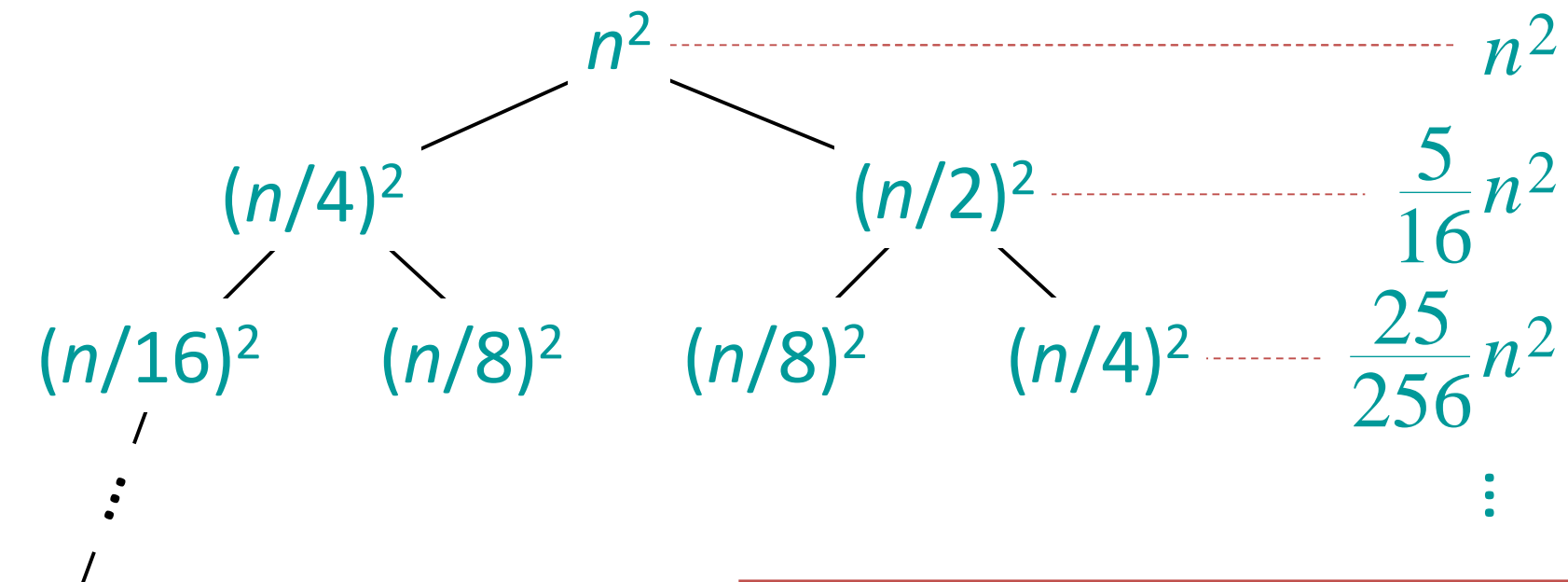
Solve  $T(n) = T(n/4) + T(n/2) + n^2$ :



# Example of recursion tree



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



$\Theta(1)$

---

$$\begin{aligned}\text{Total} &= n^2 \left( 1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \dots \right) \\ &= \Theta(n^2) \quad \text{geometric series} \quad \text{[i]}$$



# Recursion Tree

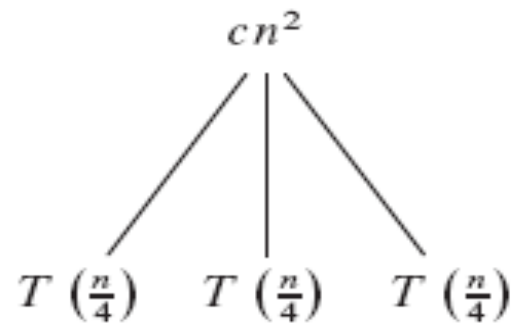


$$T(n) = 3T(n/4) + cn^2.$$

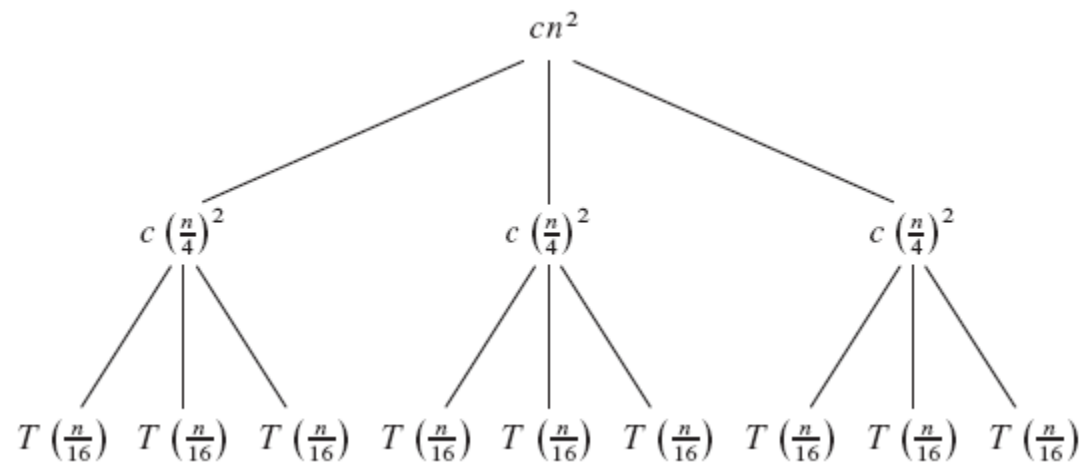
# Recursion Tree



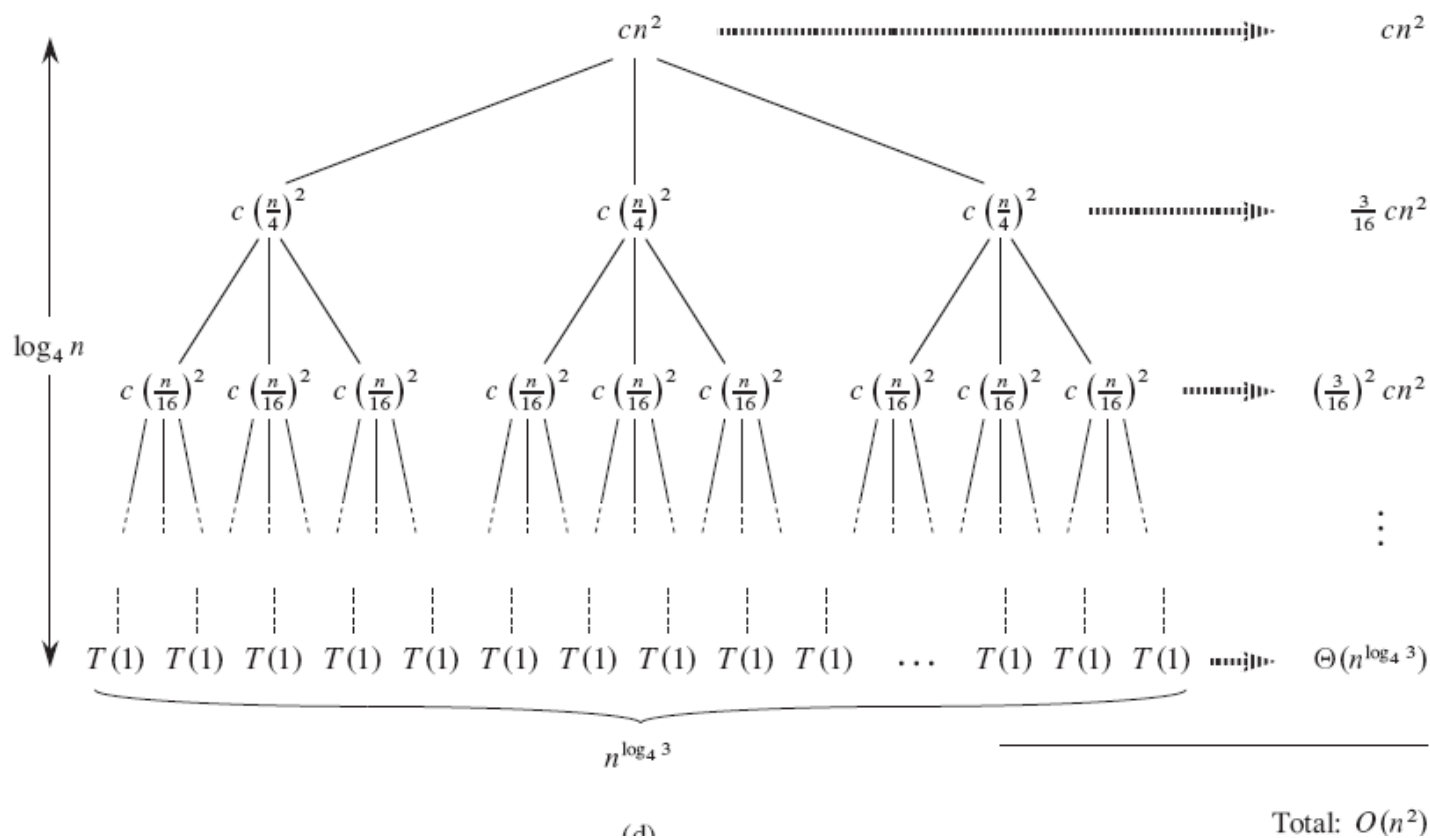
$T(n)$



# Recursion Tree



# Recursion Tree



# Recursion Tree



Because subproblem sizes decrease by a factor of 4 each time we go down one level, we eventually must reach a boundary condition. How far from the root do we reach one? The subproblem size for a node at depth  $i$  is  $n/4^i$ . Thus, the subproblem size hits  $n = 1$  when  $n/4^i = 1$  or, equivalently, when  $i = \log_4 n$ . Thus, the tree has  $\log_4 n + 1$  levels (at depths  $0, 1, 2, \dots, \log_4 n$ ).

# Recursion Tree



Next we determine the cost at each level of the tree. Each level has three times more nodes than the level above, and so the number of nodes at depth  $i$  is  $3^i$ .

Because subproblem sizes reduce by a factor of 4 for each level we go down from the root, each node at depth  $i$ , for  $i = 0, 1, 2, \dots, \log_4 n - 1$ , has a cost of  $c(n/4^i)^2$ . Multiplying, we see that the total cost over all nodes at depth  $i$ , for  $i = 0, 1, 2, \dots, \log_4 n - 1$ , is  $3^i c(n/4^i)^2 = (3/16)^i cn^2$ . The bottom level, at depth  $\log_4 n$ , has  $3^{\log_4 n} = n^{\log_4 3}$  nodes, each contributing cost  $T(1)$ , for a total cost of  $n^{\log_4 3} T(1)$ , which is  $\Theta(n^{\log_4 3})$ , since we assume that  $T(1)$  is a constant.

# Recursion Tree



$$\begin{aligned}T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\&= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\&= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3}) \quad (\text{by equation (A.5)}) .\end{aligned}$$



# Recursion Tree



$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\ &< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\ &= \frac{1}{1 - (3/16)} cn^2 + \Theta(n^{\log_4 3}) \\ &= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\ &= O(n^2) . \end{aligned}$$

# 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.

# 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  $n^\varepsilon$  factor).

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

# Three common cases



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

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

- $f(n)$  grows polynomially slower than  $n^{\log_b a}$  (by an  $n^\varepsilon$  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  $n^{\log_b a}$  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  $a f(n/b) \leq c f(n)$  for some constant  $c < 1$ .

**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,  $k = 0.$   
 $\therefore T(n) = \Theta(n^2 \lg n).$

**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  $n^\varepsilon = \omega(\lg n)$ .



Thank You !!!