

# ***Divide-and-Conquer***

***Heejin Park***

*Hanyang University*

# Asymptotic notation review

- $\Theta(n) = 3n - 1$
- $O(n) = 3n - 1$
- $O(n^2) = 3n - 1$
- $o(n^2) = 3n - 1$
- $o(n) \neq 3n - 1$
- $\Omega(n) = 3n - 1$
- $\Omega(n) = 3n^2 - 1$
- $\omega(n) \neq 3n - 1$
- $\omega(n) = 3n^2 - 1$

# Recurrences

- When an algorithm contains a recursive call to itself, its running time can often be described by a recurrence.
- A *recurrence* is an equation or inequality that describes a function in terms of its value on smaller inputs.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n=1, \\ 2T(n/2) + \Theta(n) & \text{if } n>1, \end{cases}$$

# Recurrences

## • Solving recurrences

- Obtaining asymptotic “ $\Theta$ ”, “ $O$ ” bounds on the solution.

## • Three methods for solving recurrences

- Substitution method
- Recursion-tree method
- Master method

# The substitution method

• *The substitution method* consists of two steps

1. Guess the solution.

2. Use mathematical induction to prove the guess is right.

# The substitution method

- Determining an upper bound on the recurrence

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

- Guess :

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

- Prove :

$$T(n) \leq cn \lg n$$

(for an appropriate choice of the constant  $c > 0$ )



# The substitution method

- Mathematical induction
  - Basis or boundary conditions
  - Inductive step

# The substitution method

## • Inductive step

- Assume that this bound holds for  $\lfloor n/2 \rfloor$ , that is,  
 $T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$ .

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \leq 2(c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\ &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n \\ &\text{(as long as } c \geq 1) \end{aligned}$$



# The substitution method

## ● Boundary conditions

- $T(n) \leq cn \lg n$  for  $n = 1$  (?)
- It is impossible because  $T(1) = 1$  but  $c1\lg 1 = 0$ .

# The substitution method

- Note that we don't have to prove  $T(n) = cn \lg n$  for all  $n$ .
  - We only have to prove  $T(n) = cn \lg n$  for  $n \geq n_0$  for  $n_0$ .
  - Thus, let  $n_0 = 2$ .
  - $T(2) = 2T(1) + 2 = 4$
  - $T(2) = 4 \leq c2 \lg 2$
  - $c \geq 2$  satisfies the inequality.

# The substitution method

- Observe  $T(3)$  depends directly on  $T(1)$ .
  - $T(3) = 2T(1) + 3$
  - $T(3) = 5$ .
  - To show  $T(3) = 5 \leq c3 \lg 3$ .
  - Any choice of  $c \geq 2$  satisfies the inequality.

# The recursion-tree method

- How to guess a good solution?
- We can guess the solution using the *recursion-tree method*.
  - Later, the solution is proved by the substitution method.

# The recursion-tree method

- Consider solving the following recurrence.

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2).$$

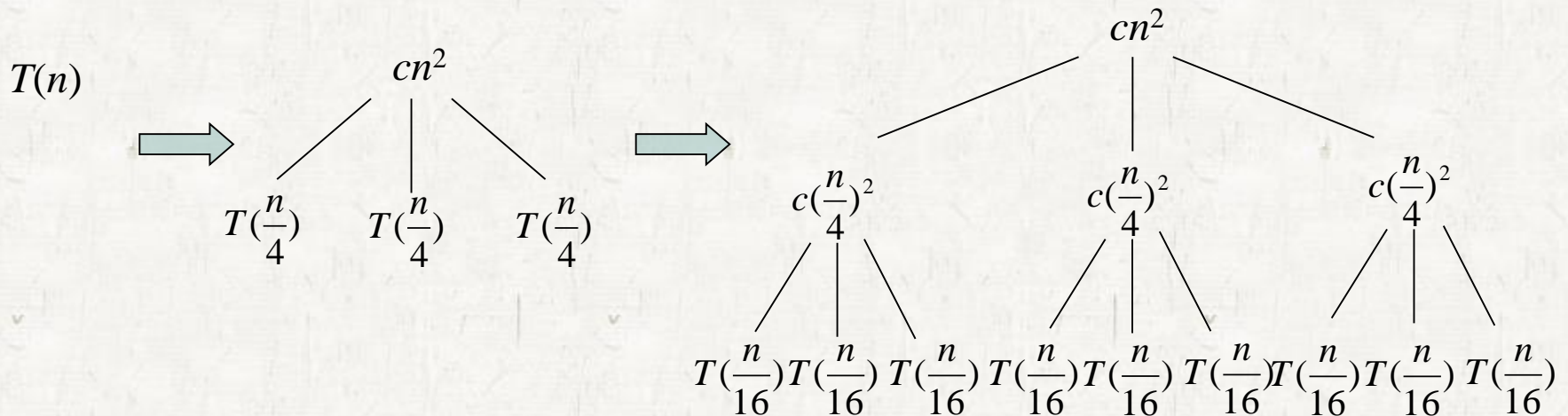
- Show  $T(n) = \Theta(n^2)$ .
  - Show  $T(n) = \Omega(n^2)$ .
    - Obvious
  - Show  $T(n) = O(n^2)$ .
    - Guess by the recursion-tree method
    - Prove by the substitution method

# The recursion-tree method

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2).$$

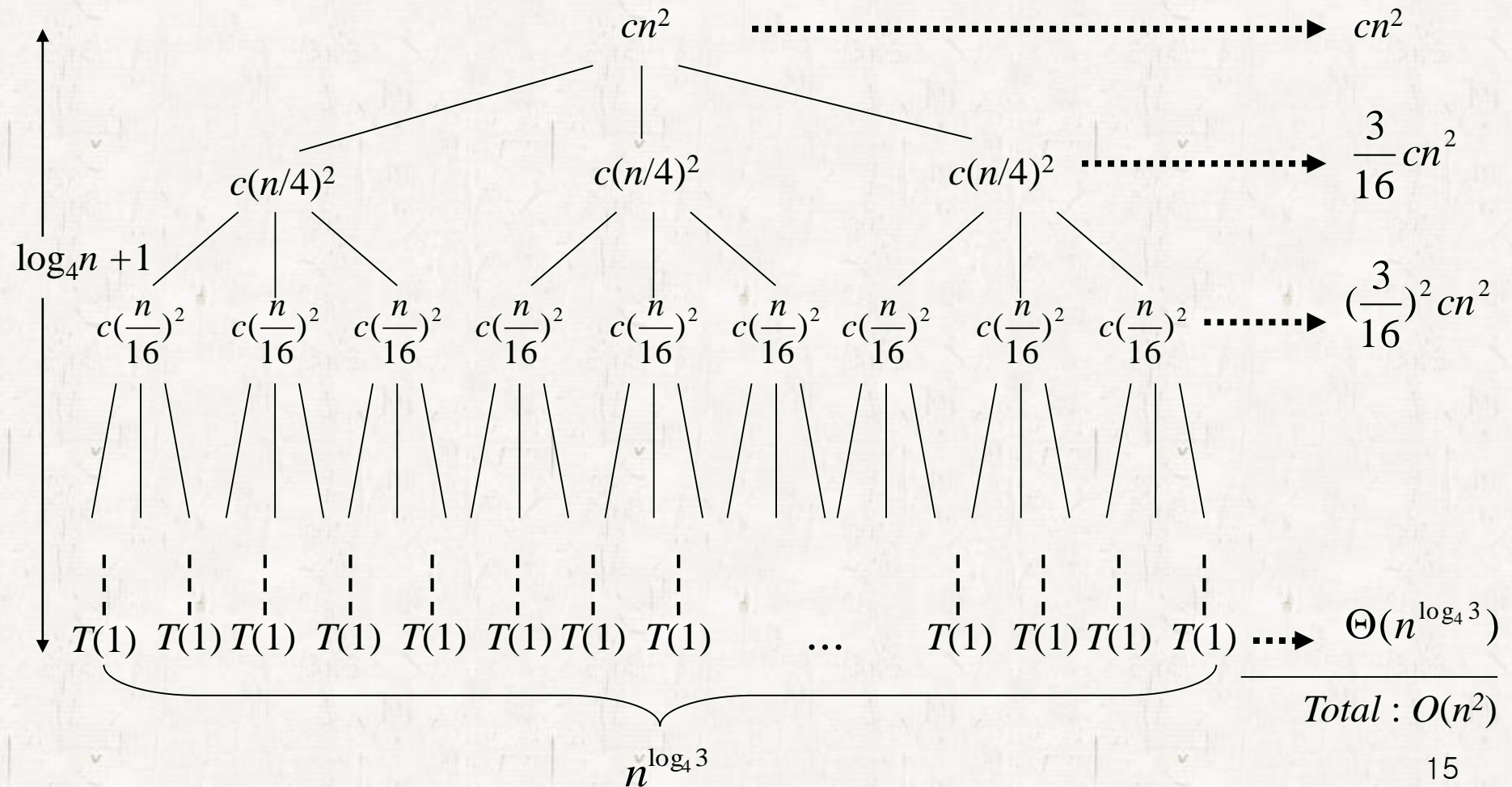
$$\Downarrow \quad n = 4^k$$

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





# The recursion-tree method



# The recursion-tree method

## • Cost computation

- Subproblem size for a node at depth  $i$ :  $n/4^i$
- The number of nodes at depth  $i$ :  $3^i$
- The number of levels:  $\log_4 n + 1$ .
  - Because the subproblem size hits  $n = 1$  when  $n/4^i = 1$  or, equivalently, when  $i = \log_4 n$ .

# The recursion-tree method

- Cost of each depth
  - The total cost of all nodes at depth  $i$ 
    - Except the last level:  $3^i c(n/4^i)^2 = (3/16)^i cn^2$
    - The last level :  $\Theta(3^{\log_4 n}) = \Theta(n^{\log_4 3})$

# The recursion-tree method

## • Cost of all depths

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

# The recursion-tree method

- We have derived a guess of  $T(n) = O(n^2)$  for the recurrence  $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$ .
- We prove  $T(n) = O(n^2)$  by the substitution method.

# The recursion-tree method

- Show that  $T(n) \leq dn^2$  (for *some*  $d > 0$  and for the *same*  $c > 0$ )

$$T(n) = 3T(\lfloor n/4 \rfloor) + cn^2$$

$$\leq 3d\lfloor n/4 \rfloor^2 + cn^2$$

$$\leq 3d(n/4)^2 + cn^2$$

$$= 3/16 dn^2 + cn^2$$

$$\leq dn^2$$

where the last step holds as long as  $d \geq (16/13)c$ .

- Since  $T(n) = \Omega(n^2)$  and  $T(n) = O(n^2)$ ,  $T(n) = \Theta(n^2)$ .

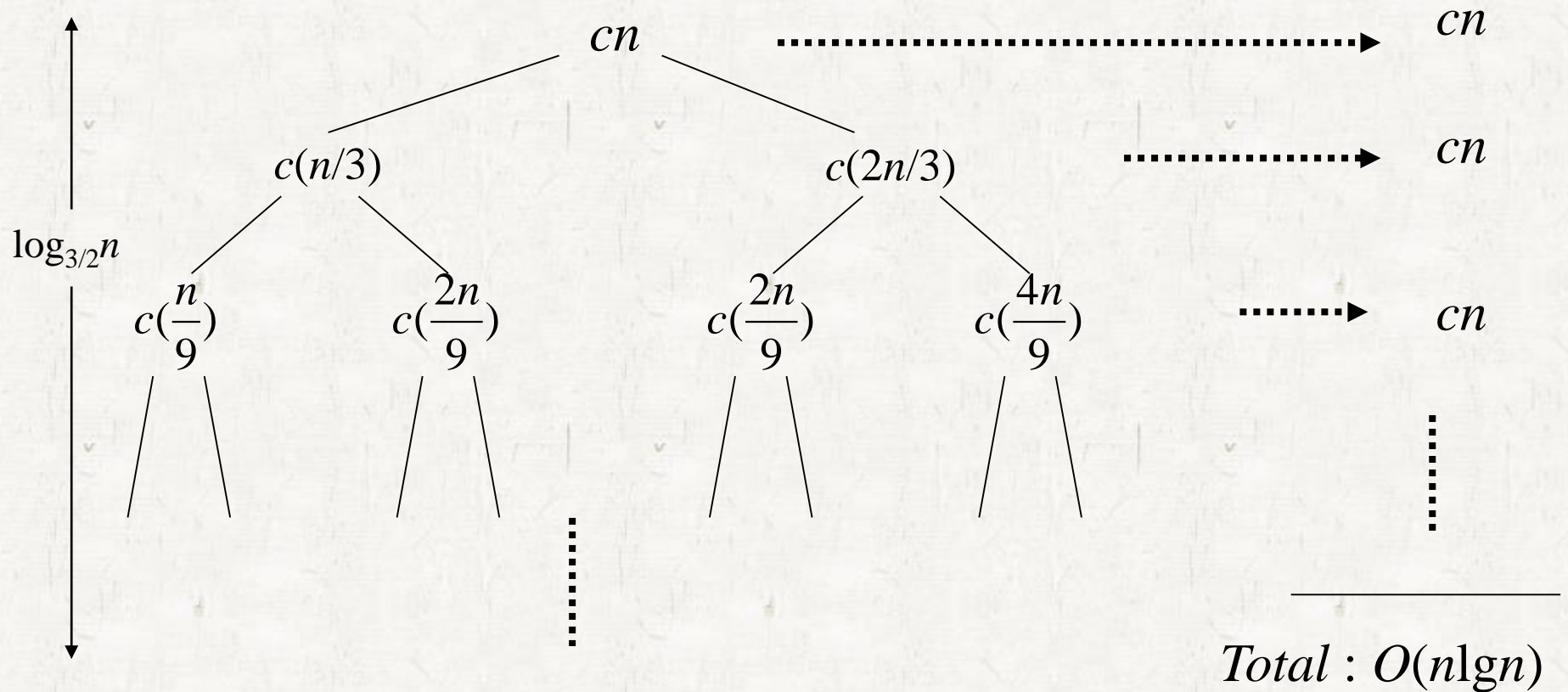


# The recursion-tree method

- Another example
  - Given  $T(n) = T(n/3) + T(2n/3) + O(n)$ ,  
to show  $T(n) = O(n \lg n)$ .

# The recursion-tree method

•  $T(n) = T(n/3) + T(2n/3) + O(n).$



# The recursion-tree method

- the cost of each level :  $cn$
- height
  - $n \rightarrow (2/3)n \rightarrow (2/3)^2 n \rightarrow \cdots \rightarrow 1$   
 $\Rightarrow (2/3)^k n = 1$  when  $k = \log_{3/2} n$ ,  
 $\Rightarrow \log_{3/2} n$ .
- Total : each level cost  $\times$  height  
 $\Rightarrow O(cn \log_{3/2} n) = O(n \lg n)$

# The recursion-tree method

- Prove the upper bound  $O(n \lg n)$
- Show that  $T(n) \leq d n \lg n$  for some constant  $d$ .

$$\begin{aligned} T(n) &\leq T(n/3) + T(2n/3) + cn \\ &\leq d(n/3) \lg(n/3) + d(2n/3) \lg(2n/3) + cn \\ &= (d(n/3) \lg n - d(n/3) \lg 3) + \\ &\quad (d(2n/3) \lg n + d(2n/3) \lg(2/3)) + cn \\ &= d n \lg n + d(-(n/3) \lg 3 + (2n/3) \lg(2/3)) + cn \end{aligned}$$

# The recursion-tree method

$$= d n \lg n + d(-(n/3) \lg 3 + (2n/3) \lg(2/3)) + cn$$

$$= d n \lg n + d(-(n/3) \lg 3 + (2n/3) \lg 2 - (2n/3) \lg 3) + cn$$

$$= d n \lg n + dn(-\lg 3 + 2/3) + cn$$

$$\leq d n \lg n, \quad \text{as long as } d \geq c/(\lg 3 - (2/3))$$

# Self-study

- **Use only recursion tree method.**
  - **Exercise 4.4-1 (4.2-1 in the 2<sup>nd</sup> ed.)**
  - **Exercise 4.4-6 (4.2-2 in the 2<sup>nd</sup> ed.)**