Divide-and-Conquer

Heejin Park

Division of Computer Science and Engineering

Hanyang University

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 "Θ", "*O*" bounds on the solution.
- **Three methods for solving recurrences**
  - Substitution method
  - Recursion-tree method
  - Master method

Technicalities

- **Neglect of technical details**
  - The assumption of integer arguments
  - Boundary conditions

Technicalities

- **The assumption of integer arguments**
  - The running time *T*(*n*) is only defined when *n* is an integer.
  - For example, the recurrence for Merge-Sort is really,

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

  - But normally, it is represented as

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

  - Because the solution is not changed.

Technicalities

- **Omit boundary conditions**
  - Boundary conditions are omitted because *T*(*n*) is normally constant for small *n*.
  - for example

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

$$\Big\downarrow$$

$$T(n) = 2T(n/2) + \Theta(n)$$

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 :

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

(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, $T(\lfloor n/2 \rfloor) \le c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$.

$$
\begin{aligned}
T(n) = 2T(\lfloor n/2 \rfloor) + n &\le 2(c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\
&\le cn \lg(n/2) + n \\
&= cn \lg n - cn \lg 2 + n \\
&= cn \lg n - cn + n \\
&\le cn \lg n
\end{aligned}
$$

(as long as $c \ge 1$)

The substitution method

- Boundary conditions
  - $T(n) \le cn \lg n$ 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

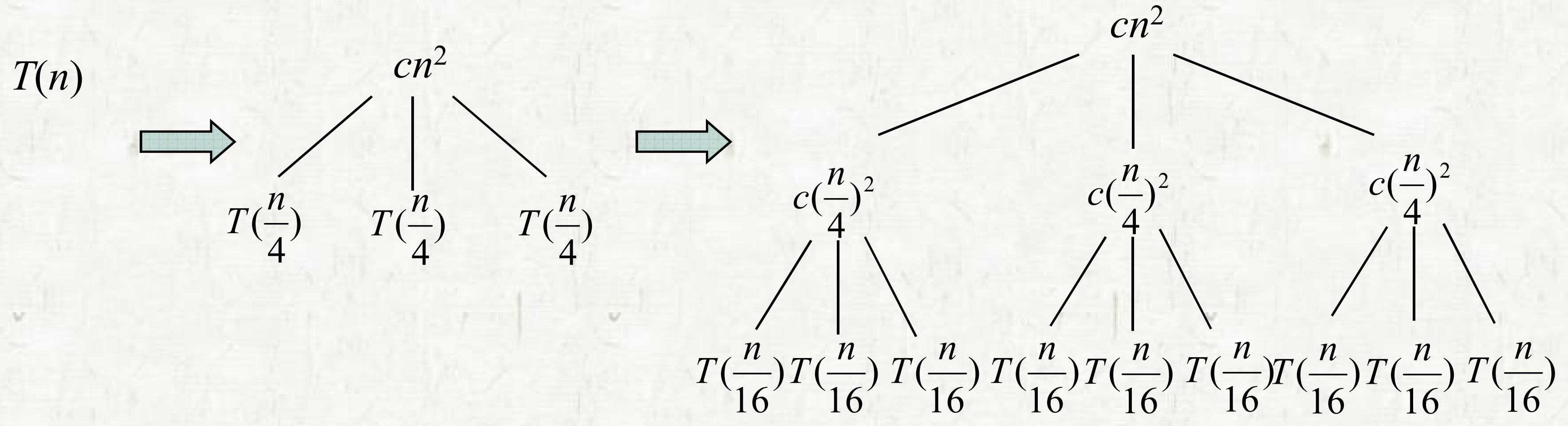
- **Consider solving the following recurrence.**
  $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$.
- Show $T(n) = \Theta(n^2)$.
  - Show $T(n) = \Omega(n^2)$.
    - Obvious
  - Show $T(n) = O(n^2)$.
    - Guess by the recursion-tree method
    - Prove by the substitution method

The recursion-tree method

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

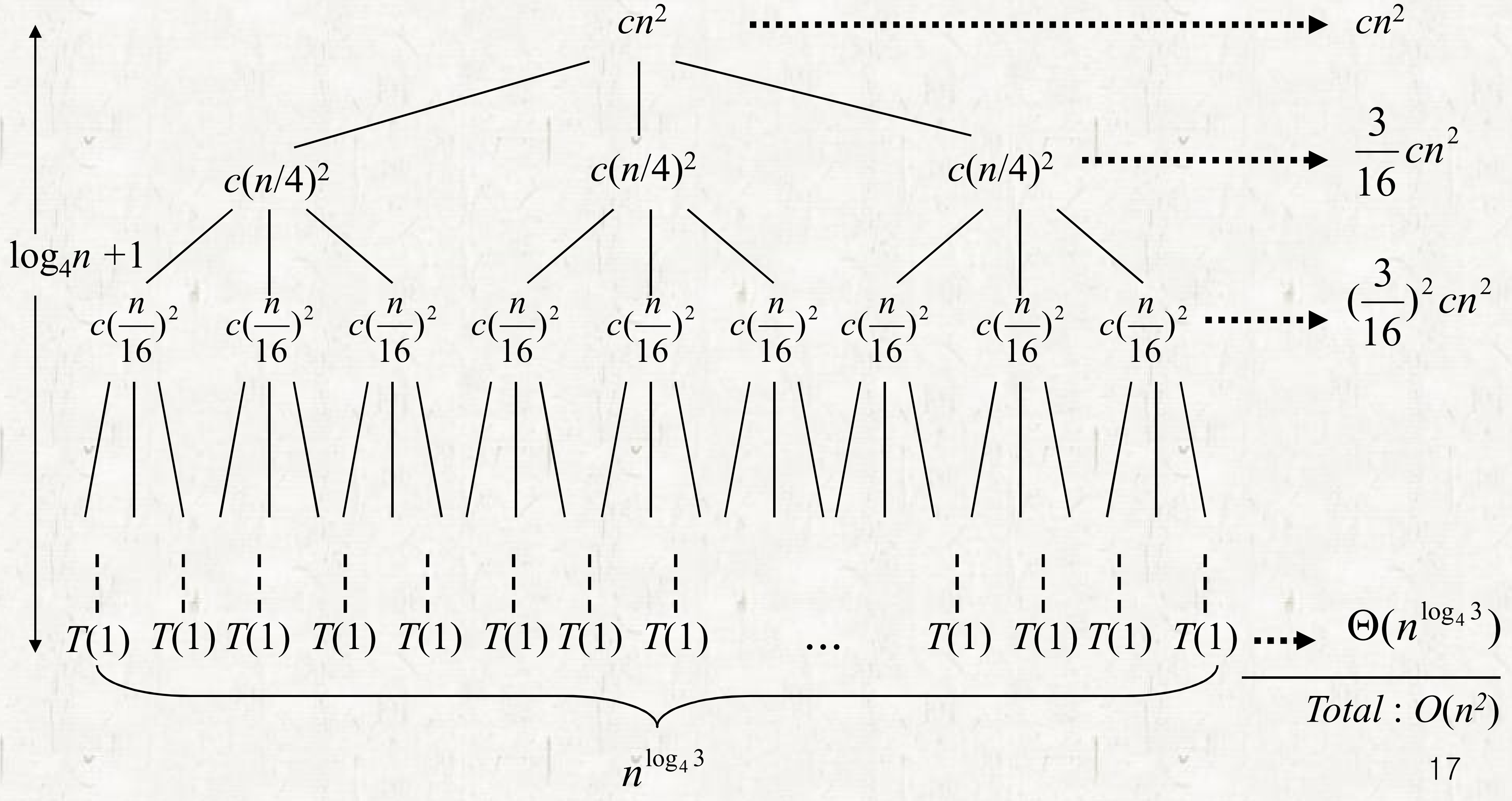
$n = 4^k$

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



$T(n)$
cn^2
$T(\frac{n}{4})$ $T(\frac{n}{4})$ $T(\frac{n}{4})$
cn^2
$c(\frac{n}{4})^2$ $c(\frac{n}{4})^2$ $c(\frac{n}{4})^2$
$T(\frac{n}{16})$ $T(\frac{n}{16})$ $T(\frac{n}{16})$ $T(\frac{n}{16})$ $T(\frac{n}{16})$ $T(\frac{n}{16})$ $T(\frac{n}{16})$ $T(\frac{n}{16})$ $T(\frac{n}{16})$

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 $T(n) \le dn^2$ (for *some* $d > 0$ and for the *same* $c > 0$)

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

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

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

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

  $\le dn^2$

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

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

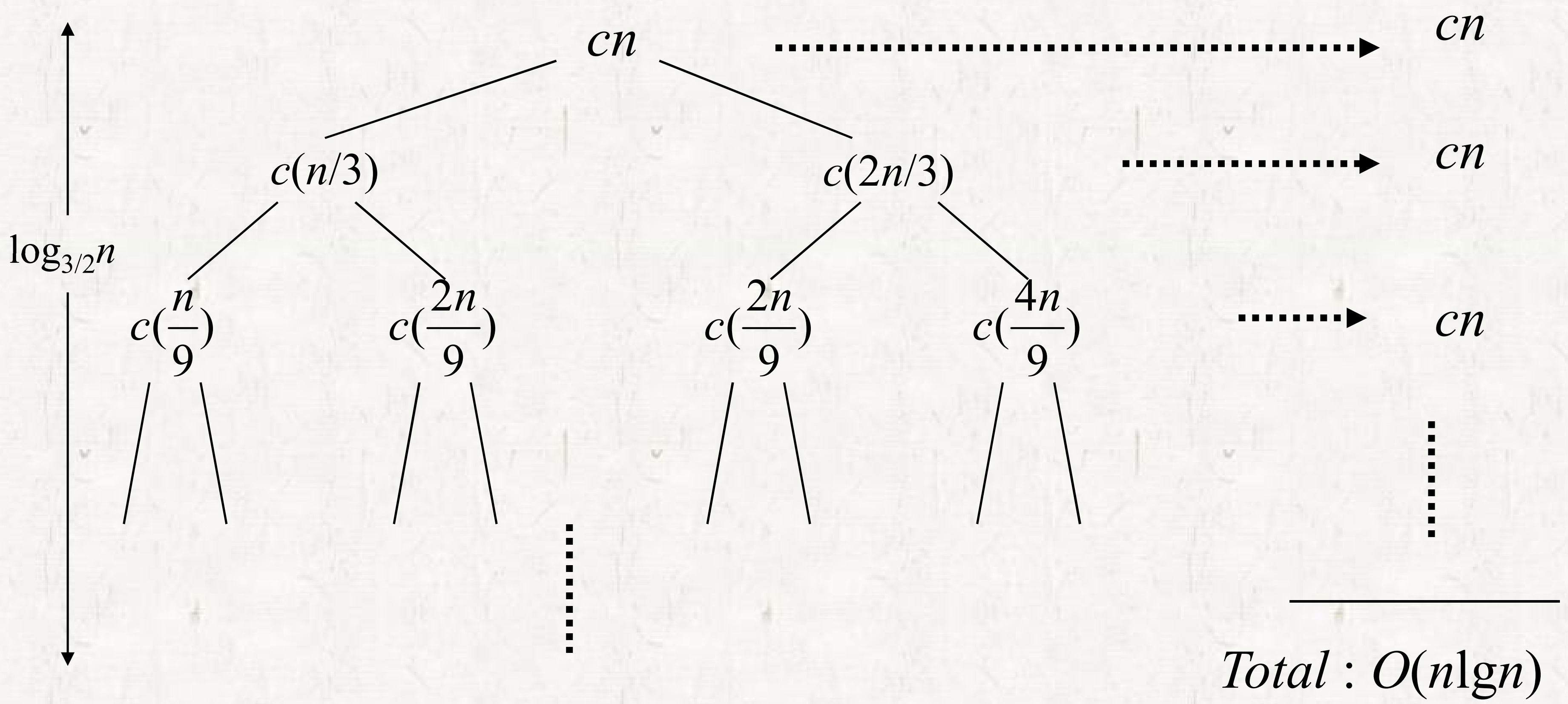
The recursion-tree method

- 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
  - $n \rightarrow (2/3)n \rightarrow (2/3)^2 n \rightarrow \cdots \rightarrow 1$

    => $(2/3)^k n = 1$ when $k = \log_{3/2} n$,

    => $\log_{3/2} n$.
- Total : each level cost X height

  => $O(cn\log_{3/2} n) = O(n \lg n)$

The recursion-tree method

- Prove the upper bound $O(n\lg n)$

- Show that $T(n) \le dn\lg n$ for some constant d. (self-study)

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

The recursion-tree method

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

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

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

$\leq dn\lg n$, 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 2nd ed.)**
  - **Exercise 4.4-6 (4.2-2 in the 2nd ed.)**