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.

Running time of merge sort of recurrence form

T(n) = Θ(nlogn) ??

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

value is smaller as execution

Recurrences

- **Solving recurrences**
  - Obtaining asymptotic "Θ", "*O*" bounds on the solution. Ω ...

- **Three methods for solving recurrences**
  - focus
  - 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.

↳ part of substitution method

The substitution method

- Determining an upper bound on the recurrence

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

(c=1)

1. recurrence form 형성

- Guess :

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

2. 우리의 추측

not recurrence form의 시간 복잡도

= recurrence form의 시간 복잡도 ??

- Prove :

$n \geq n_0$

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

3. Prove

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

The substitution method

- Mathematical induction (prove)
  - Basis or boundary conditions
  - Inductive step

The substitution method

WHEN $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$ is given

- Inductive step $\boxed{T(m) \le cm \log m}$ $\underline{m \le n-1}$ 성립 → $\underline{m = n}$ 성립 증명!!
  - Assume that this bound holds for $\lfloor n/2 \rfloor$, that is,
    $T(\lfloor n/2 \rfloor) \le c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$. → 성립한다고 가정

$\times 2 + n$ $\times 2 + n$

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

$\lfloor n/2 \rfloor \le \frac{n}{2}$

$$\le cn \lg(n/2) + n$$

$cn(\log n - \log 2)$ $\qquad$ $=1$

$$= cn \lg n - cn \lg 2 + n$$

$\Rightarrow \log \lfloor \frac{n}{2} \rfloor \le \log \frac{n}{2}$

$$= cn \lg n - cn + n$$

$= cn \log n - \underbrace{n(c-1)}_{>0}$

$$\le cn \lg n$$

(as long as $\underline{c \ge 1}$)

The substitution method

- Boundary conditions

  - $T(n) \le cn \lg n$ for $n = 1$ (?)

    $1 \le 0$???

  - It is impossible because $T(1) = 1$ but $c1\lg 1 = 0$.

    ↳ so we set $n_0 \neq 1$ when $n \geq n_0$

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 $n_0 = 2$. Basis step
  - $T(2) = 2T(1) + 2 = 4$
  - $T(2) = 4 \leq c2 \underbrace{\lg 2}_{1}$
  - $c \geq 2$ satisfies the inequality. $\Rightarrow$ $c \geq 2,\ n_0 \geq 2$

The substitution method

we have to ~~case~~ $T(3)$

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

why?

if $c=2$

$T(3) = 5 \le c3\log 3$

$5 \le 6.xx$

=> satisfy when $c=2$

↳ we can accept $c \ge 2$!!

$T(2)$ $T(3)$가 확인되어야

$T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$이 모든 n에 대해

inductive step 이 증명됨!!

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 by $T(n) = 3T(\lfloor \frac{n}{4} \rfloor) + \Theta(n^2)$ $(\because T(\lfloor \frac{n}{4} \rfloor) \geq 0)$
  - Show $T(n) = O(n^2)$.
    - Guess by the recursion-tree method
    - Prove by the substitution method

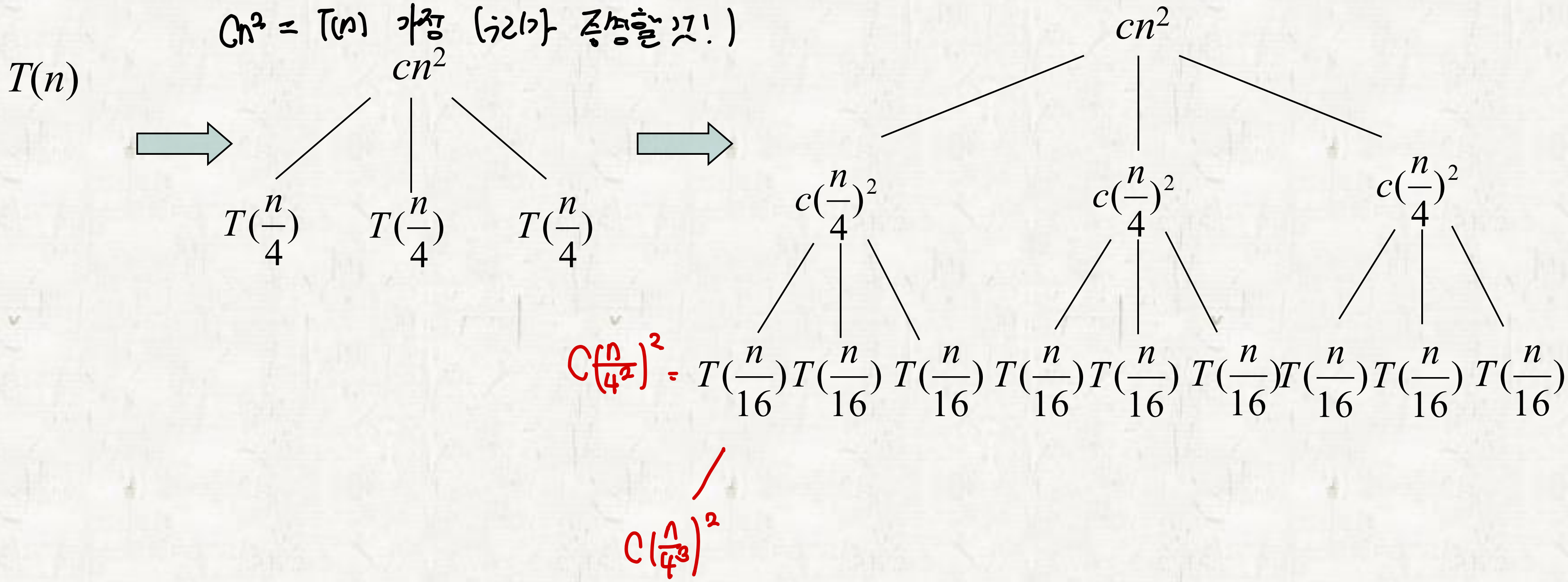
The recursion-tree method

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

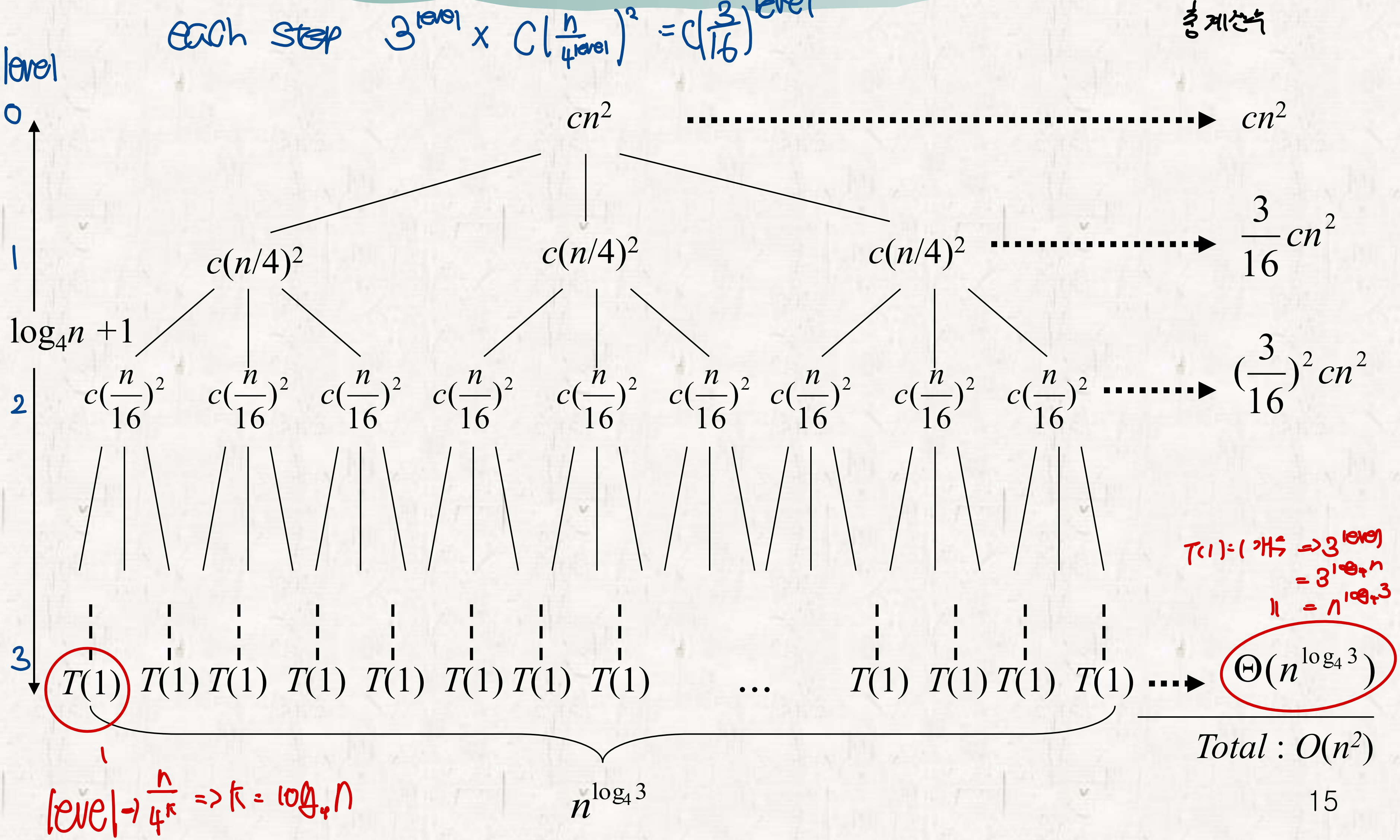
$n = 4^k$ (가정)

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

$cn^2 = T(n)$ 가정 (귀기가 증명할 것!)



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

each level's numbers of term

$\Rightarrow 3^{level}$ T(1)'s numbers

$\Rightarrow 3^{\log_4 n} \Rightarrow n^{\log_4 3}$

$$T(n) = \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \underline{\Theta(n^{\log_4 3})}$$

T(1) = 1 개수 (상수이다)

$$< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) = O(n^2) + \Theta(n^{\log_4 3})$$

$$= O(n^2)$$

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.

$T(n) \leq cn^2 \quad c > 0$

$n \geq n_0 > 0$

The recursion-tree method

Basis 'n=1'

inductive step

$T(n) = 3T(\lfloor \frac{n}{4} \rfloor) + cn^2$ (given)

- Show that $T(n) \le \boxed{d}n^2$ (for *some* $d > 0$ and for the *same* $c > 0$)

  new constant

  ↳ exact value

$T(n) = 3T(\lfloor n/4 \rfloor) + cn^2$ $\le 3d(\lfloor \frac{n}{4} \rfloor)^2$

$T(k) = 3T(\lfloor \frac{n}{4} \rfloor) + ck^2$

When $k < n$, $T(n) \le dn^2$ is qualified?

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

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

$= 3/16\, dn^2 + cn^2$ $\le dn^2$ $\Rightarrow$ $cn^2 \le \frac{13}{16} dn^2$ $\Rightarrow$ $c \le \frac{13}{16} d$ $\Rightarrow$ $d \ge \frac{16}{13} c$

Satisfied conditions

$d \ge 2c$

(for integer)

$\le dn^2$

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

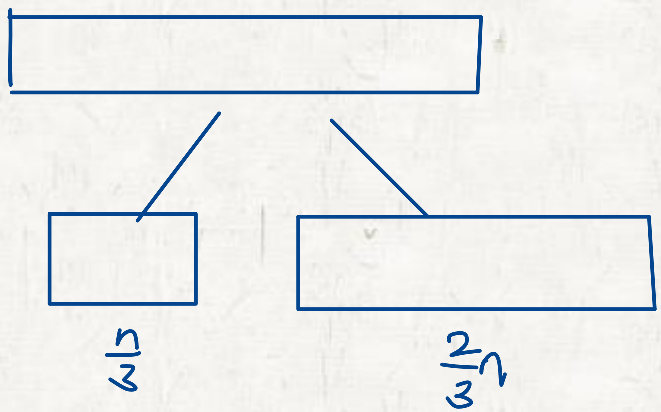
- Since $T(n) = \Omega(n^2)$ and $T(n) = O(n^2)$, $T(n) = \Theta(n^2)$.

The recursion-tree method

mergeSort recursion

$T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + O(n) \rightarrow \Theta(n\log n)$

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



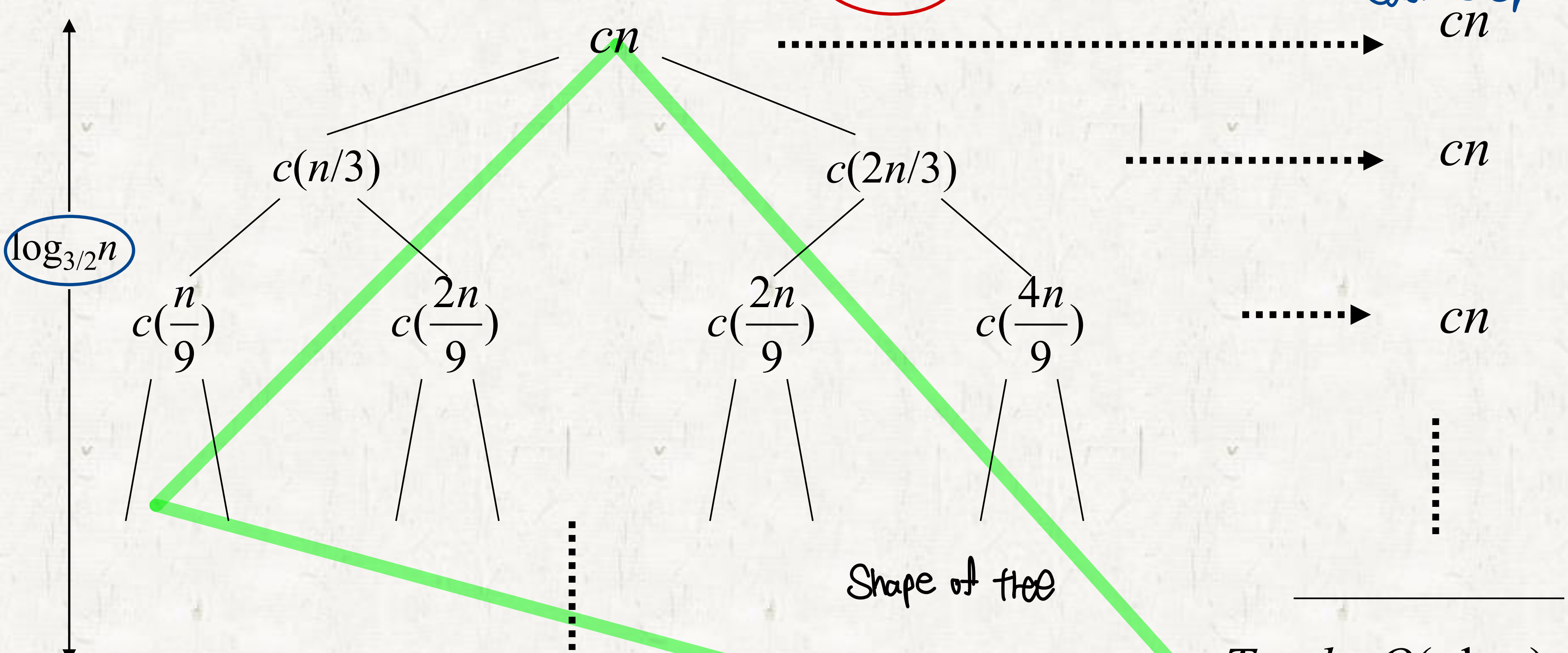
The recursion-tree method

$n = 3^k$

$T(n) \le cn$ (가정)

- $T(n) = T(n/3) + T(2n/3) + O(n)$. → cn

each step



Total : $O(n\lg n)$

$\Rightarrow cn \times (\log_{\frac{3}{2}} n)$

$\Rightarrow O(n \log n)$

height of short tree(left)

$n/3^l = 1 \Rightarrow l = \log_3 n \quad \therefore \log_3 n + 1$

height of longest tree(right)

$(\frac{3}{2})^l = n \Rightarrow \log_{\frac{3}{2}} n \quad \boxed{\therefore \log_{\frac{3}{2}} n + 1}$

We choose it!
for big-O notation

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 x height

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

The recursion-tree method

$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$ (given)

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

$T(n) \le dn\log n$ 증명

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

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

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

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

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

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

The recursion-tree method

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

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

상수무시

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

$\le \cancel{dn\lg n}$, as long as $d \ge c/(\lg 3 - (2/3))$

$\therefore dn(-\log 3 + \frac{2}{3}) + cn \le 0$

$d \ge \frac{c}{(\log 3 - \frac{2}{3})}$

Self-study

- Use only recursion tree method.
  - Exercise 4.4-1 (4.2-1 in the 2^{nd} ed.)
  - Exercise 4.4-6 (4.2-2 in the 2^{nd} ed.)