Chapter - 3

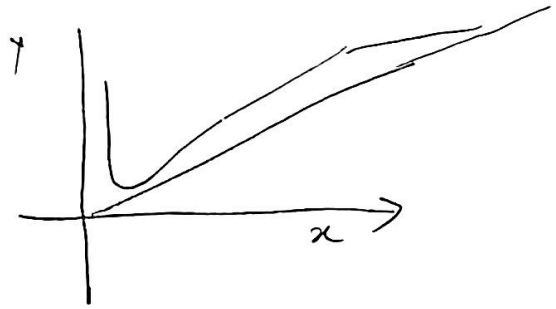
Growth of functions

Cormen et-al.

From website -

An asymptote of a curve is a line such that the distance between the curve and line approaches zero as they tend to infinity.

asymptote $\cong$ "not falling together" (in Greek)



"How the running time of an algorithm increases with the size of the input in the limit" is is our interest.

There are three notations

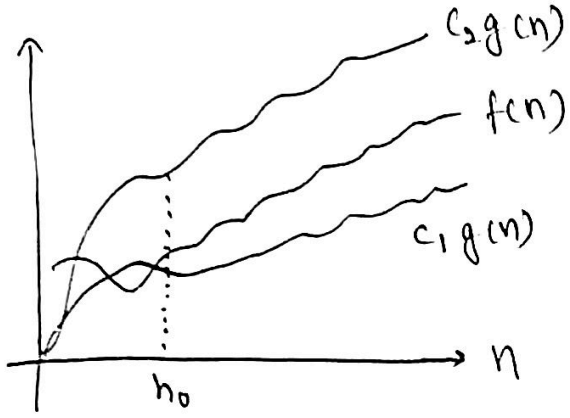
(i) Θ - notation
(ii) O - notation
(iii) Ω - notation.

Θ - notation

For a given function $g(n)$, $\Theta(g(n))$ is the set of functions.

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \}$$

Domain of function $g(n)$ and $f(n)$ is natural numbers $N = \{0, 1, 2, \dots\}$



every functions in $\theta(g(n))$ are asymptotically non-negative.

θ is asymptotically tight bound. because $f(n)$ always lies bet^n $c_1 g(n)$ & $c_2 g(n)$

Abuse → $f(n) = \theta(g(n))$

Correct → $f(n) \in \theta(g(n))$

e.g.

Example-1 $\theta(n^2) = \frac{1}{2} n^2 - 3n$

$g(n) = n^2$ | $f(n) = \frac{1}{2} n^2 - 3n$

To prove it find values of c_1, c_2 & n_0

$$\frac{c_1 n^2}{n^2} \leq \frac{\frac{1}{2} n^2 - 3n}{n^2} \leq \frac{c_2 n^2}{n^2} \quad \text{—— (1)}$$

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$$

In eq^n (1)

R.H.S. ⇒ Any value $n \geq 1/2$ & $c_2 \geq 1/2$

L.H.S ⇒ Any value $n \geq 7$ & $c_1 \leq 1/14$

∴ we must have $n \geq 7$ to satisfy both L.H.S. & R.H.S.

∴ $c_1 = 1/14$ $c_2 = 1/2$ $n_0 = 7$

Example-2 – $6n^3 \neq \theta(n^2)$

for contradiction purpose let us assume

c_2 & n_0 exist

$$\therefore \quad c_1 n^2 \leq 6n^3 \leq c_2 n^2$$

$$c_1 \leq \frac{6n^3}{n^2} \leq c_2$$

$$c_1 \leq 6n \leq c_2$$

which is not possible, since n is not constt.

Thumb rule

Set the value c_1 slightly smaller than coefficient of Highest order

the value c_2 slightly larger than coefficient of Highest order

O-notation - For a given function, $g(n)$, The $O(g(n))$ is a set of functions as defined below.

$$O(g(n)) = \{ f(n) : \text{there exist the constants } n_0 \& c \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \}$$

If $f(n) = \theta(g(n)) \Rightarrow f(n) = O(g(n))$

$$\theta(g(n)) \subseteq O(g(n))$$

e.g. $an+b = f(n)$

$g(n) = n^2$

$C + an^2 + bn = O(n^2)$

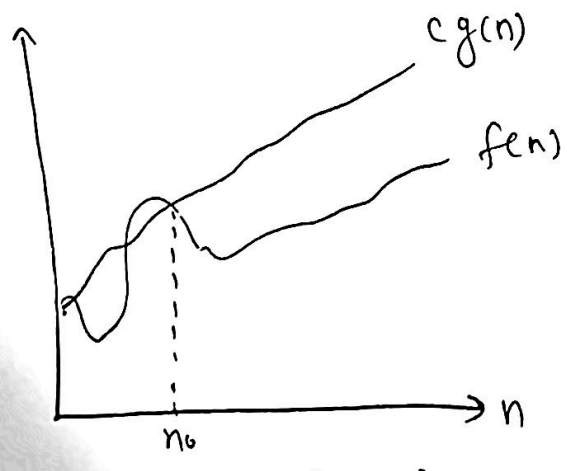
$C + an^2 + bn = \Theta(n^2)$.

O-notation describes upper bound

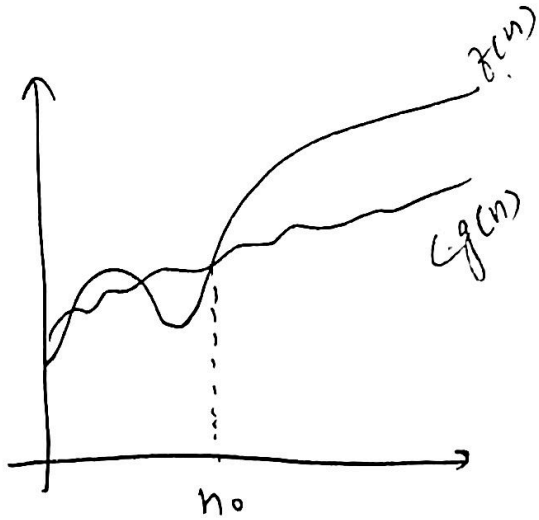
Ω - notation

It is asymptotic lower bound. For a given function, $\Omega(g(n))$ is defined as follows.

$\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0 \}$.



$f(n) = O(g(n))$



$f(n) = \Omega(g(n))$

O-notation

O-notation may be asymptotically tight or may not be tight. But o-notation is always asymptotically tight.

e.g.
$2n = O(n^2)$ is not tight

But $2n^2 = O(n^2)$ is tight

For a given function, $g(n)$, $O(g(n))$ is defined as follows.

$O(g(n)) = \{ f(n)$: for any positive constt, $c > 0$, there exists a constant, $n_0 > 0$ s.t. $0 \leq f(n) < cg(n)$ for all $n \geq n_0 \}$.

e.g. $2n = O(n^2)$

$2n^2 \neq O(n^2)$

ω-notation

For a given function $g(n)$, $\omega(g(n))$ is defined as follows.

$\omega(g(n)) = \{ f(n)$: for any positive constt, $c > 0$ there exists a constt, $n_0 > 0$ s.t. $0 \leq cg(n) < f(n)$ for all $n > n_0 \}$.

e.g.

$$\frac{n^2}{2} = \omega(n)$$

but $$\frac{n^2}{2} \neq \omega(n^2)$$

Meanings

When we say "the running time is $O(n^2)$", we mean that there is a function $f(n)$ that is $O(n^2)$ such that for any value of n, no matter what particular input of size n is chosen, the running time on that input is bounded from above by the value $f(n)$.

↑ V Imp

i.e. Equivalently, we mean that the worst-case running time is $O(n^2)$.

When we say that the running time of an algorithm is $\Omega(g(n))$, we mean that no matter what particular input of size n is chosen for each value of n, the running time on that input is at least a const times $g(n)$, for sufficiently large n.

i.e. Equivalently, we mean that the best-case running time is $\Omega(g(n))$.

But we can say that the worst-case running time of insertion sort is $\Omega(n^2)$

Because we have used modifier

Transitivity

$f(n) = \Theta(g(n)) \;\&\; g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

$f(n) = O(g(n)) \;\&\; g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$

⋮

Similarly Θ, O, Ω, o & ω are all transitive

Reflexivity

$f(n) = \Theta(f(n))$
$f(n) = O(f(n))$
$f(n) = \Omega(f(n))$

o & ω are not reflexive.

Symmetry

$f(n) = \Theta(g(n))$ iff $g(n) = \Theta(f(n))$

Transpose symmetry

$f(n) = O(g(n))$ iff $g(n) = \Omega(f(n))$
$f(n) = o(g(n))$ iff $g(n) = \omega(f(n))$.

Equivalence

$f(n) = O(g(n)) \approx a \leq b$
$f(n) = \Omega(g(n)) \approx a \geq b$
$f(n) = \Theta(g(n)) \approx a = b$
$f(n) = o(g(n)) \approx a < b$
$f(n) = \omega(g(n)) \approx a > b$

Not Trichotomy

asymptotic notations does not follow this property.

Only one rel^n bet^n fns.
like real no's
$a < b$ $a = b$ or $a > b$

Standard notations and common functions

Monotonicity

$f(n)$ is monotonically increasing if $m \le n \Rightarrow f(m) \le f(n)$

$f(n)$ is —— " —— decreasing if $m \le n \Rightarrow f(m) \ge f(n)$

$\lceil n/2 \rceil + \lfloor n/2 \rfloor = n$

$\lceil \lceil n/a \rceil / b \rceil = \lceil n/ab \rceil$

$\lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/ab \rfloor$

$a \bmod n = a - \lfloor a/n \rfloor n$ —

$a = 7 \quad n = 2$

$7 \bmod 2 = 7 - \lfloor 7/2 \rfloor 2 = 7 - 6 = 1$

$a = 2 \quad n = 7$

$2 \bmod 7 = 2 - \lfloor 2/7 \rfloor 7 = 2 - 0 = 2$

Polynomial $p(n) = \sum_{i=0}^{d} a_i n^i$ d-degree polynomial

$a^0 = 1$ | $a^{-1} = 1/a$ | $a^m a^n = a^{m+n}$

$a^1 = a$ | $(a^m)^n = a^{mn}$ |

Logarithms

$\lg n = \log_2 n$ | $\lg \lg n = \lg (\lg n)$

$\ln n = \log_e n$ |

$\lg^k n = (\lg n)^k$ |

For all $a > 0, b > 0, c > 0$

VImp → $a = b^{\log_b a}$ e.g. $16 = 2^{\log_2 16} = 2^4$

$\log_c (ab) = \log_c a + \log_c b$

$$\log_b a^n = n \log_b a$$

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$\log_b (1/a) = -\log_b a$$

$$\log_b \left(\frac{a}{d}\right) = \log_b (a) - \log_b (d)$$

$$\log_b a = \frac{1}{\log_a b}$$ ✓

VImp

$$a^{\log_b c} = c^{\log_b a}$$

eg. $2^{\log_2 16} = 16^{\log_2 2} = 16$

Factorials

$n! = 1 \cdot 2 \cdot 3 \cdots n$ Special case $0! = 1$

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta(1/n)\right)$$

$n!$

$n! = O(n^n)$

$n! = \omega(2^n)$

$\lg(n!) = \Theta(n \lg n)$

Logarithm functions

$$\lg^* n = \min \{ i \geq 0 \mid \lg^{(i)} n \leq 1 \}$$

Iterated functions

$$f^{(i)}(n) = \begin{cases} n & \text{if } n = 0 \\ f(f^{(i-1)}(n)) & \text{if } n > 0 \end{cases}$$

e.g. if $f(n) = 2n$

$$f^{(i)}(n) = 2^i n$$

e.g. $\lg^* n = \min \{ i \geq 0 : \lg^{(i)} n \leq 1 \}$

$\lg^*(16) = 3$?

$\lg^0 n = n = 16$

$\lg^{(1)}(n) = \lg n = \lg(16) = 4$

$\lg^{(2)}(n) = \lg \lg n = \lg(4) = 2$

$\lg^{(3)}(n) = \lg \lg \lg(n) = \lg(2) = 1$

Fibonacci nos

$F_0 = 0$

$F_1 = 1$

$F_i = F_{i-1} + F_{i-2}$

∴ e.g. 0, 1, 2, 3, 5, 8, 13 .....

P.T.

$$(n+a)^b = \Theta(n^b)$$

$$0 \leq c_1 n^b \leq (n+a)^b \leq c_2 n^b \quad \forall n \geq n_0$$

note that

$$n+a \leq n+|a| \leq 2n$$

$$n+a \geq n-|a| \geq \tfrac{1}{2}n$$

Thus $n \geq 2|a|$

$0 \le \frac{1}{2}n \le n+9 \le 2n$ Taking power b

$0 \le (1/2n)^b \le (n+9)^b \le (2n)^b$

$c_1 = (1/2)^b \quad c_2 = 2^b \quad n_0 = 29$

Functions on the same line are in the same equivalence class, and those higher on the page are Ω of those below them.

Exp
- $2^{2^{n+1}}$
- 2^{2^n}
- $(n+1)!$
- $n!$
- e^n
- $n \cdot 2^n$
- 2^n
- $(3/2)^n$
- $(\lg n)^{\lg n} = n^{\lg \lg n}$

- $(\lg n)!$
- n^3
- $n^2 = 4^{\lg n}$
- $n \lg n$ and $\lg(n!)$
- $n = 2^{\lg n}$
- $(\sqrt{2})^{\lg n} (= \sqrt{n})$
- $2^{\sqrt{2 \lg n}}$
- $\lg^2 n$
- $\ln n$
- $\sqrt{\lg n}$
- $\ln \ln n$
- $2^{\lg^* n}$
- $\lg^* n$ and $\lg^*(\lg n)$
- $\lg(\lg^* n)$
- $n^{1/\lg n} (=2)$ and 1

Imp

① Exponential functions grows faster than polynomial functions, which grow faster than polylogarithmic functions

② The base of a logarithm does not matter asymptotically, but the base of an exponential and the degree of a polynomial do matter.

Chapter - 4

Recurrences

(Cormen et. al.

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.

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

Solving recurrences

Three methods

(i) Substitution method
(ii) Recursion-tree method.
(iii) Master theorem method. for $T(n) = aT(n/b) + f(n)$

(1) Substitution method

(i) Guess the soln
(ii) Use induction to find constt.
(iii) Show the soln works.

Example -

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

Ex 3)

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

i.e. $T(n) \leq cn \log_2 n \quad \forall c > 0$

Induction (ii) I.H. Assume that it holds for $n_0 = n/2$

$$\therefore T(n/2) \leq c(n/2) \lg(n/2)$$

$$\therefore T(n) \leq 2\left[c(n/2) \cdot \lg(n/2)\right] + n$$

$$\leq cn \cdot \log_2(n/2) + n$$

$$= cn \lg n - cn \log_2 2 + n$$

$$= cn \lg n - cn(1) + n$$

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

$$\leq cn \lg n \quad \text{it holds for } c \geq 1$$

find boundary values (consts)

But $T(1) = c \cdot 1 \cdot \log 1 = 0$

which is not the case.

2nd Guess i) Let $T(n) = n \lg n + n$

ii) Induction

Basis $\Rightarrow n = 1$

$\therefore T(1) = 1 \cdot \lg(1) + 1 = 1$

Inductive hypothesis is $T(K) = K \lg K + K \quad \forall K < n$

Inductive step

we will use inductive hypothesis

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

$$= 2\left(\frac{n}{2} \cdot \lg \frac{n}{2} + \frac{n}{2}\right) + n$$

$$= n \lg \frac{n}{2} + n + n$$

$$= n \lg \frac{n}{2} + 2n$$

$$= n(\lg n - \lg 2) + 2n$$

$$= n \lg n - n(1) + 2n$$

$$= n \lg n + n$$

Defⁿ

A recurrence relⁿ is a function and is defined in terms of

(i) one or more base cases.

(ii) itself, with smaller arguments.

eg (1) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1) + 1 & \text{if } n > 1 \end{cases}$ Soln $T(n) = n$

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

(3) $T(n) = \begin{cases} 0 & \text{if } n = 2 \\ T(\sqrt{n}) + 1 & \text{if } n > 2 \end{cases}$ Soln $T(n) = \lg \lg n$

④ $T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n/3) + T(2n/3) + n & \text{if } n>1 \end{cases}$ Soln $T(n) = \Theta(n \lg n)$

V Imp

① In algorithm analysis, we usually express both the recurrence and its soln using asymptotic notation

e.g. we use $T(n) = 2T(n/2) + \Theta(n)$ rather the exact recurrence $T(n) = 2T(n/2) + n$

② The boundary conditions are expressed as $T(n) = O(1)$ for sufficiently small n.

e.g.

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

③ In practice, we just use asymptotics most of the time, and we ignore boundary conditions.

④ So in substitution method, we generally don't worry about boundary cases and we show soln in asymptotic notation.

⑤ Show the upper (O) and lower (Ω) bounds seperately, if you want to show boundary cases.

Example

$$T(n) = 2T(\sqrt{n}) + \lg n \quad \text{---} (1)$$

Put $m = \lg n$

$\therefore n = 2^m$

$\therefore$ Eqn (1) becomes.

$$T(2^m) = 2T(\sqrt{2^m}) + m$$

$$T(2^m) = 2T(2^{m/2}) + m \quad \text{---} (2)$$

$= 2^{m/2}$

$= 2^{m \cdot 1/2}$

$= \frac{2^m}{2}$

$= m/2$

Now we can rename Eqn (2) as

$$T(m) = 2T(m/2) + m \quad \text{---} (3)$$

We know the soln of Eqn (3).

$$T(m) = m \lg m + m = O(m \lg m)$$

$\therefore$ Now re-substituting $m = \lg n$

$$\boxed{T(n) = (\lg n \cdot \lg \lg n) + \lg n}$$

Example 4-1.1 S.T. the soln of $T(n) = T(\lceil n/2 \rceil) + 1$ is $O(\lg n)$

using substitution method.

Guess - $T(n) \leq c \lg(n-b)$ for some consts c & b.

We have to P.T. $T(n) \leq c \lg(n-b)$

Assume this holds for $\lceil n/2 \rceil$.

$$T(n) = T(\lceil n/2 \rceil) + 1$$

$$\therefore T(n) \leq c \lg(n/2 - b) + 1$$

$$< c \lg(n/2 - b + 1) + 1$$

$$= c \lg\left(\frac{(n/2 - b + 1) \times 2}{2}\right) + 1$$

$$= c \lg\left(\frac{n - 2b + 2}{2}\right) + 1$$

$$= c \lg(n - 2b + 2) - c \lg 2 + 1$$

$$\leq c \lg(n - b)$$

$\therefore$ **T is $O(\lg n)$**

Example 4.1-2 - $T(n) = 2T(n/2) + \Theta(n)$

S.T. Upper bound of $T(n) = 2T(n/2) + \Theta(n)$

Let us use substitution method.

$T(n) \leq 2T(n/2) + cn$ for some +ve const c.

① Upper bound

Guess - $T(n) \leq d \cdot n \cdot \lg n$ for some +ve const d

Now we are given c and need to find d.

$\therefore T(n) \leq 2T(n/2) + cn$

$$T(n) \leq 2T(n/2) + cn$$

$$= 2\left(d \cdot \frac{n}{2} \cdot \lg \frac{n}{2}\right) + cn$$

$$= dn \cdot \lg \frac{n}{2} + cn$$

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

$$\leq dn \lg n \quad \text{if} \quad -dn + cn \leq 0$$

$$\therefore \boxed{d \geq c}$$

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

(ii) Lower bound

$$T(n) \geq 2T(n/2) + cn$$

$$T(n) \geq 2T(n/2) + cn$$

$$= 2\left(d\frac{n}{2} \lg \frac{n}{2}\right) + cn$$

$$= dn \lg \frac{n}{2} + cn$$

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

$$\geq dn \lg n \quad \text{if} \quad -dn + cn \geq 0$$

$$\boxed{d \leq c}$$

$$\therefore T(n) = \Omega(n \lg n)$$

$$\therefore T(n) = \Theta(n \lg n)$$

Be careful when using asymptotic notation

P.T. $T(n) = 4T(n/4) + n$ is $T(n) = O(n)$

we may assume $T(n) \leq cn$

Using Induction

$\therefore T(n) \leq 4(c \cdot (n/4)) + n$

$\leq nc + n$

$= O(n)$ ..... Wrong

Wrong because in induction process we havent shown the exact form i.e. $T(n) \leq nc$

$\therefore$ It is not correct soln

Example 4-1.5 - ST. the soln to $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ is $O(n \lg n)$.

$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$

Assume that $T(n) = 1$ $\forall n$ $0 < n < 35$

Now we need show that.

$\forall$ some $n_0 > 0$ and some fixed const $c > 0$

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

Let $n_0 = 35$

If $T(n) \leq c((n-34)\lg n - n$

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

because $c(n-34)\lg n - n \leq cn\lg n$.

Base case: Let $n = 35$

$$T(35) = 2T(34) + 35 = 2 + 35 = 37$$

and $c(n-34)\lg n - n = c\lg 35 - 35$

$$\geq 37$$

When $c \geq 72/\lg 35$

Inductive step - Suppose $T(k) \leq c((k-34)\lg k - k$

$* \; n_0 \leq k < n$

$\therefore T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$

$\leq 2(c \cdot \lfloor n/2 \rfloor + 17 - 34)\lg(\lfloor n/2 \rfloor + 17) - n$ →

$$T(n) \leq 2T(n/2 + 17) + n$$

$$\leq 2c(n/2 - 34)\lg\left(\frac{n}{2} - 34\right) - \frac{n}{2} + n$$

$\leq 2(c\lfloor n/2 \rfloor - 17)\lg(\lfloor n/2 \rfloor + 17) - 2n + n$

$= c(n-34)\lg(\lfloor n/2 \rfloor + 17) - n$

$< c(n-34)\lg n - n(\lfloor n/2 \rfloor + 17 < n)$

$\therefore \quad f(n) \in O(n \lg n)$

Example 4.1-6 Solve the recurrence $T(n) = 2T(\sqrt{n}) + 1$ by making a change of variables.

$$T(n) = 2T(\sqrt{n}) + 1$$

Put $m = \lg n$

$\therefore \quad n = 2^m$ — ①

$\therefore$ eqn ① becomes

$T(2^m) = 2T(2^{m/2}) + 1$ — ②

Now eqn ② becomes in terms of m.

$T(m) = 2T(m/2) + 1$ — ③

Using substitution method,

Guess $T(m) \leq c \cdot \lg(m-b)$ for some +ve const c & b.

$\therefore$ Now we have to P.T. $T(m) \leq c \lg(m-b)$

Let us assume it holds for $\frac{m}{2}$

$\therefore \quad T(m) = T(m/2) + 1$

$\therefore \quad T(m) \leq c \lg\left(\frac{m}{2} - b\right) + 1$

$< c \lg\left(\frac{m}{2} - b + 1\right) + 1$

$$= c \lg \left(\frac{\left(\frac{m}{2} - b + 1\right) \times 2}{2} \right) + 1$$

$$= c \lg \left(\frac{m - 2b + 2}{2} \right) + 1$$

$$= c \lg (m - 2b + 2) - c \lg 2 + 1$$

$$= c \lg (m - 2b + 2) + 1$$

$$< c \lg (m - b)$$

$\therefore$ It is $O(\lg m)$.

Now putting back the values of m

$$m = \lg n$$

$\therefore$ $\boxed{T(n) = (\lg \lg m)}$

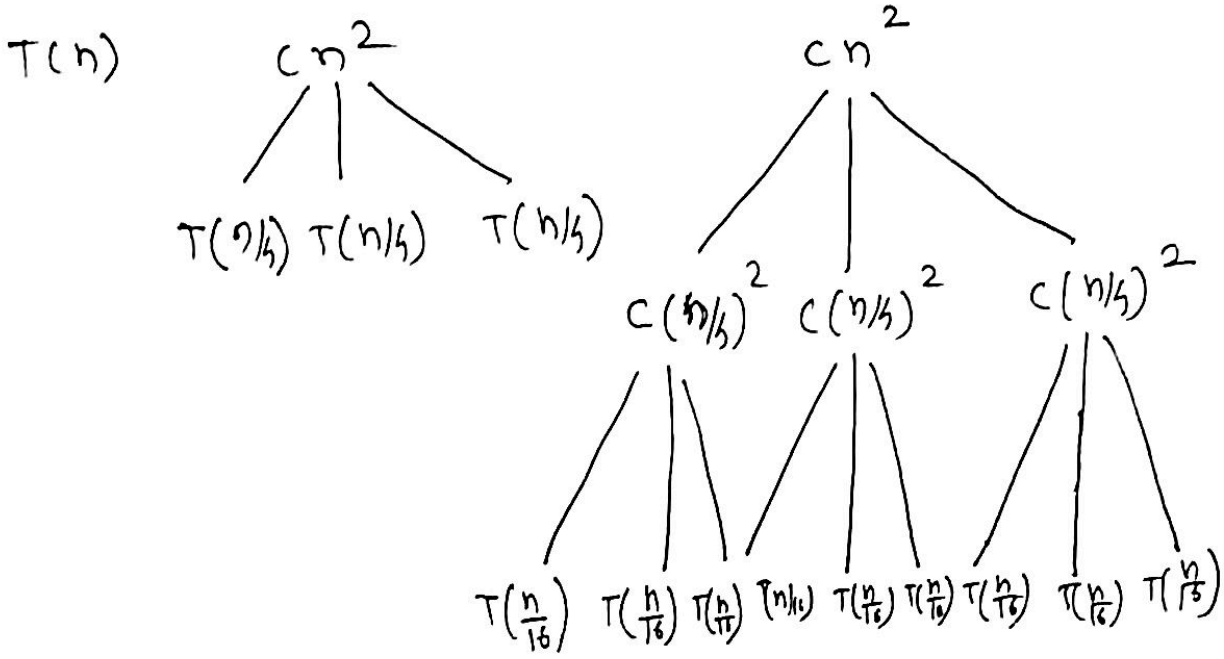
Recursion-Tree Method

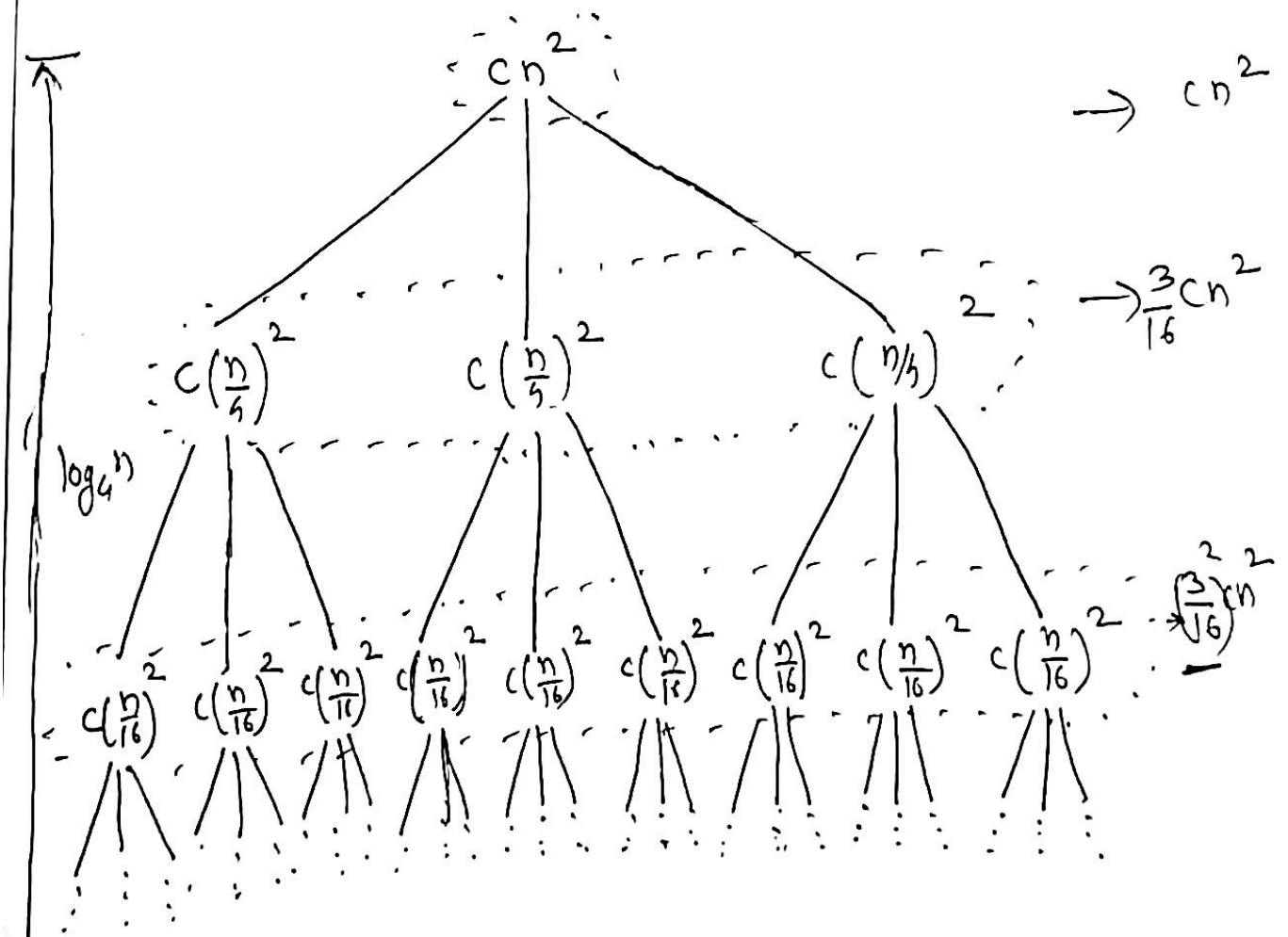
In this method, each node represents the cost of a single subproblem somewhere in the set of recursive function invocations.

We sum the costs within each level of the tree to obtain a set of per-level costs. and then we sum all the per-level costs to determine the total costs of all levels of the recursion.

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

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





T(1) T(1) T(1) T(1) T(1) . T(1) . T(1) T(1) . . . T(1) → $\theta(n^{\log_4 3})$

$n^{\log_4 3}$

Total $O(n^2)$

At depth i, the size of subproblem $= \frac{n}{4^i}$

Level	no. of node at level	Cost of node	Cost of level
0	1	cn^2	cn^2
i	3	$c(1/4)^2 n^2$	$(\frac{3}{16})cn^2$
2	9	$c(\frac{1}{16})^2 n^2$	$(\frac{3}{16})^2 cn^2$
3	27	$c(\frac{1}{64})^2 n^2$	$(\frac{3}{16})^3 cn^2$
⋮	⋮	⋮	⋮

add it.

$$\therefore T(n) = cn^2 + \frac{3}{16}cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n-1} cn^2 + \Theta(n^{\log_4 3})$$

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

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

To find the upper bound we can set it to

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

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

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

The lower bound is when it call for the first time it is still $cn^2 = \Omega(n^2)$

We can prove same thing with substitution method.

Let the guess be $T(n) \leq dn^2 = O(n^2)$

$\therefore$ s.t. $T(n) \leq 3T(\lfloor n/4 \rfloor) + cn^2$

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

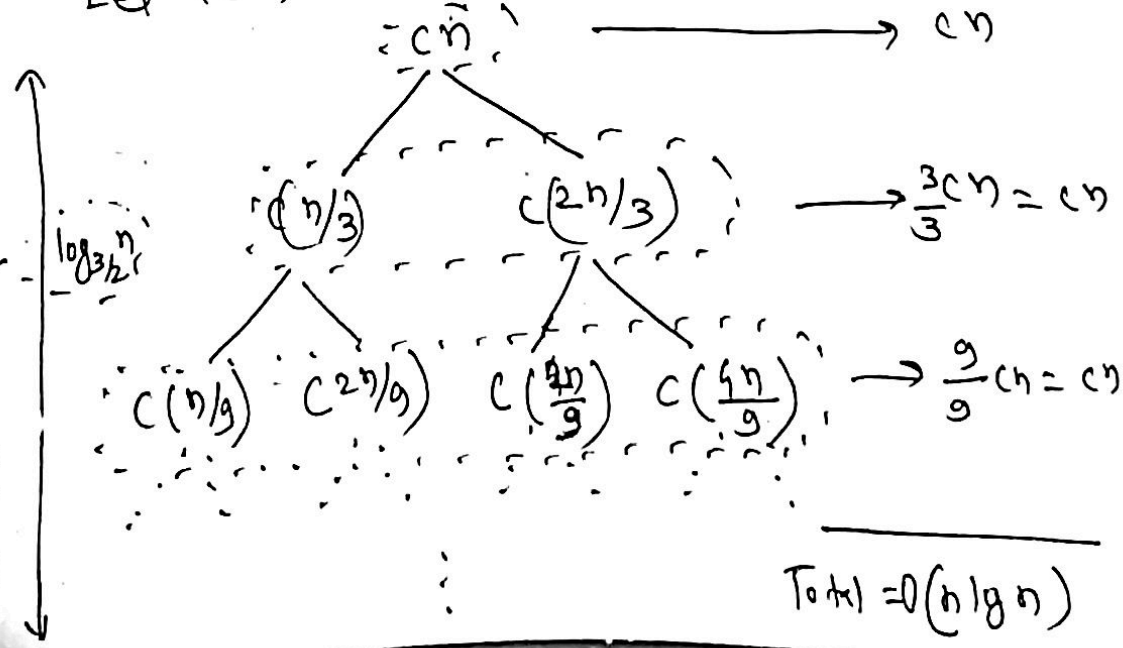
$\leq 3d(n/4)^2 + cn^2$

$= \frac{3}{16} dn^2 + cn^2$

$\leq dn^2$

Example $T(n) = T(n/3) + T(2n/3) + O(n)$

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



$$T(n) = \sum_{i=0}^{\log_{3/2} n} cn = O\left(cn \log_{3/2} n\right) = O(n \lg n)$$

[redacted]

$$= cn \log_{3/2} n = cn \log_{\frac{3}{2}\times\frac{2}{2}} n = cn \log_{3/4} + cn \log_2 n$$

The recursion tree is not complete binary tree but [redacted]

$$= \text{[redacted]} = O(n \lg n)$$

[redacted]

Substitution method for verification for upper bound

S.T. the upper bound is $O(n \lg n)$

Now P.T. $T(n) \leq dn \lg n$ where $d = +ve$ const.

$$T(n) \leq T(n/3) + T(2n/3) + cn$$

$$\leq \left(d \frac{n}{3} \lg \frac{n}{3}\right) + \left(d \frac{2n}{3} \cdot \log \frac{2n}{3}\right) + cn$$

$$\leq \left(\frac{dn}{3} \lg n - \frac{dn}{3} \lg 3\right) + \left(d \frac{2n}{3} \cdot \log n - d \frac{2n}{3} \lg \frac{3}{2}\right) + cn$$

$$= \text{[redacted]}$$

$$= dn \lg n - \frac{dn}{3} \lg 3 - d \frac{2n}{3} \lg \frac{3}{2} + cn$$

$$= dn \lg n - d\left(\frac{n}{3} \lg 3 + \frac{2n}{3} \lg \frac{3}{2}\right) + cn$$

$$= dn\lg n - d\left(\frac{n}{3}\lg 3 + \frac{2n}{3}\lg 3 - \frac{2n}{3}\lg 2\right) + c(n)$$

$$= dn\lg n - dn\left(\lg 3 - \frac{2}{3}(\log 3 - \lg 2)\right) + c(n)$$

$$= dn\lg n - dn\left(\lg 3 - \frac{2}{3}\lg 3 - \frac{2}{3}\lg 2\right) + c(n)$$

$$= dn\lg n - dn\left(\frac{1}{3}\lg 3 - \frac{2}{3}\lg 2\right) + c(n)$$

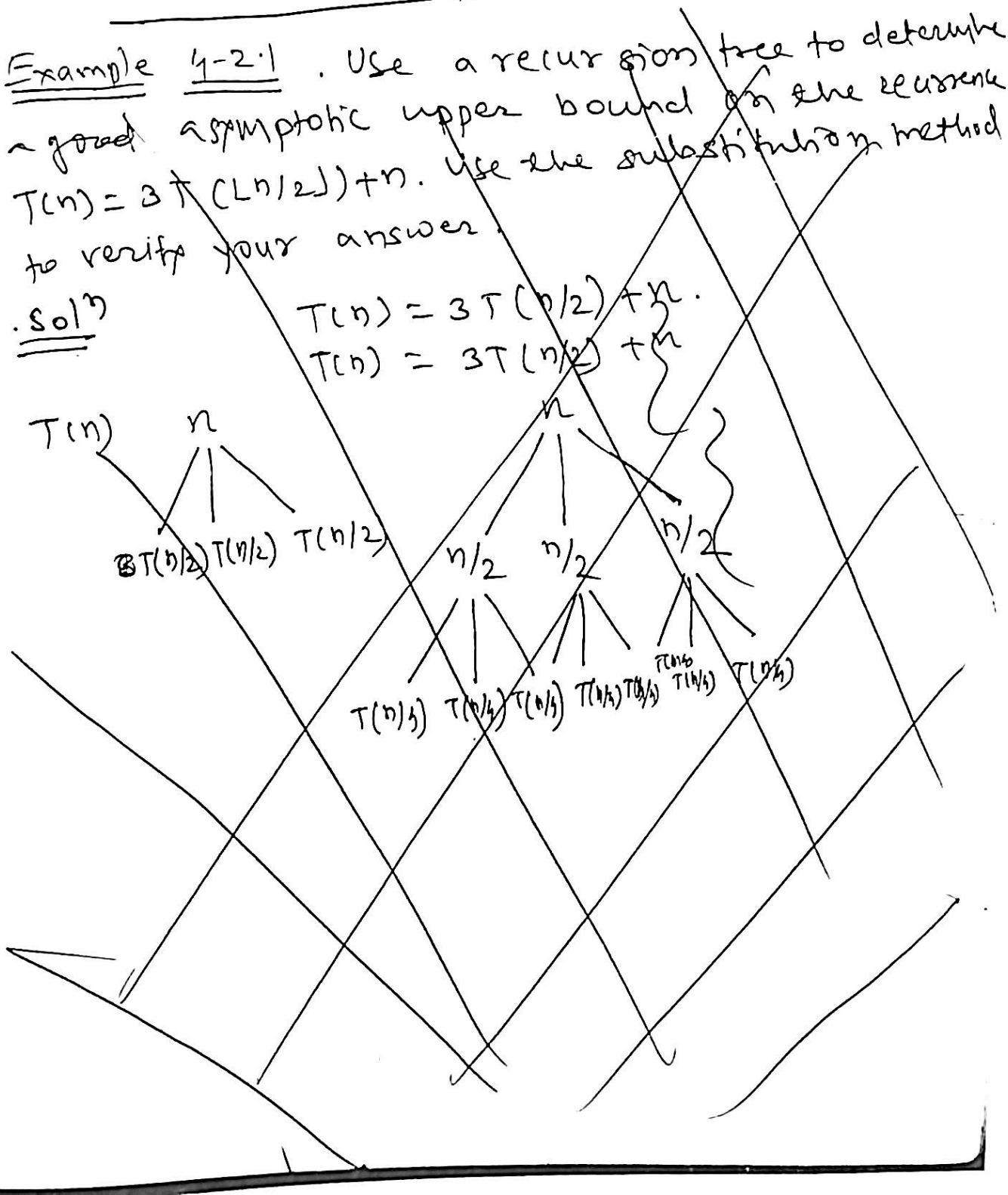
$$\leq dn\lg n$$

Example 4-2.1 . Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 3T(\lfloor n/2 \rfloor) + n$. Use the substitution method to verify your answer.

Soln

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

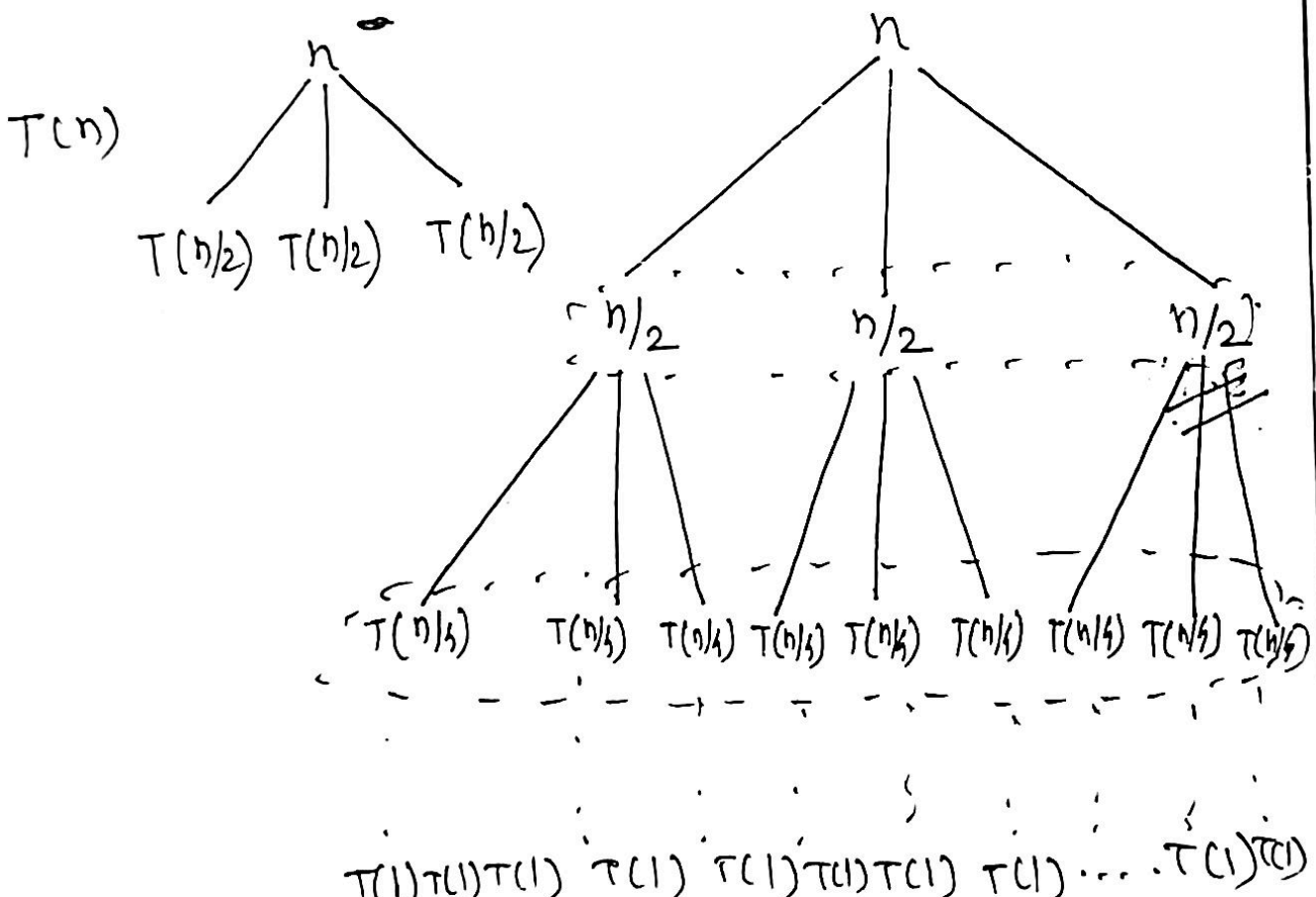
$T(n) = 3T(n/2) + n$



Example 4-2-1 Use a recursion method to determine a good asymptotic upper bound on the recurrence $T(n) = 3T(\lfloor n/2 \rfloor) + n$. Use the substitution methode to verify your answer.

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

$$\therefore T(n) = 3T(n/2) + n.$$



$$\therefore T(n) = \sum_{i=0}^{\log_2 n - 1} (3/2)^i n + \theta\left(n^{\log_2 3}\right)$$

$$= n \sum_{i=0}^{\lg n - 1} (3/2)^i + \theta\left(n^{\lg 3}\right)$$

$$= n \frac{(3/2)^{\lg n} - 1}{(3/2 - 1)} + \theta\left(n^{\lg 3}\right)$$

$$= 2n\left[(3/2)^{\lg n} - 1\right] + \theta(n^{\lg 3})$$

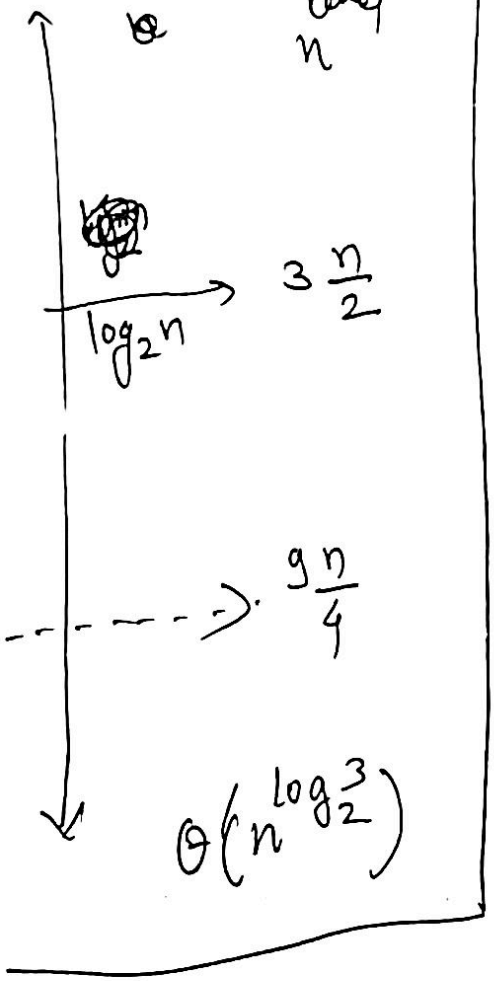
$$= 2\left(n(3/2)^{\lg n} - n\right) + \theta(n^{\lg 3})$$

$$= 2 \times \frac{3^{\lg n}}{(2^{\lg n})} - 2n + \theta(n^{\lg 3})$$

$$= 2 \cdot 3^{\lg n} - 2n + \theta(n^{\lg 3})$$

$$= 2n^{\lg 3} - 2n + \theta(n^{\lg 3})$$

$$= \theta(n^{\lg 3})$$



We can prove it by substitution method,

$$T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor^{\lg 3} - c \lfloor n/2 \rfloor$$

i.e. $\boxed{T(n) = c\, n^{\lg 3} - cn.}$

P.T. $T(n) \leq 3T(n/2) + n$

$$\leq 3c(n/2)^{\lg 3} - \frac{cn}{2} + n$$

$$\leq 3c \frac{n^{\lg 3}}{2^{\lg 3}} - \frac{cn}{2} + n$$

$$\leq cn^{\lg 3} - cn + n$$

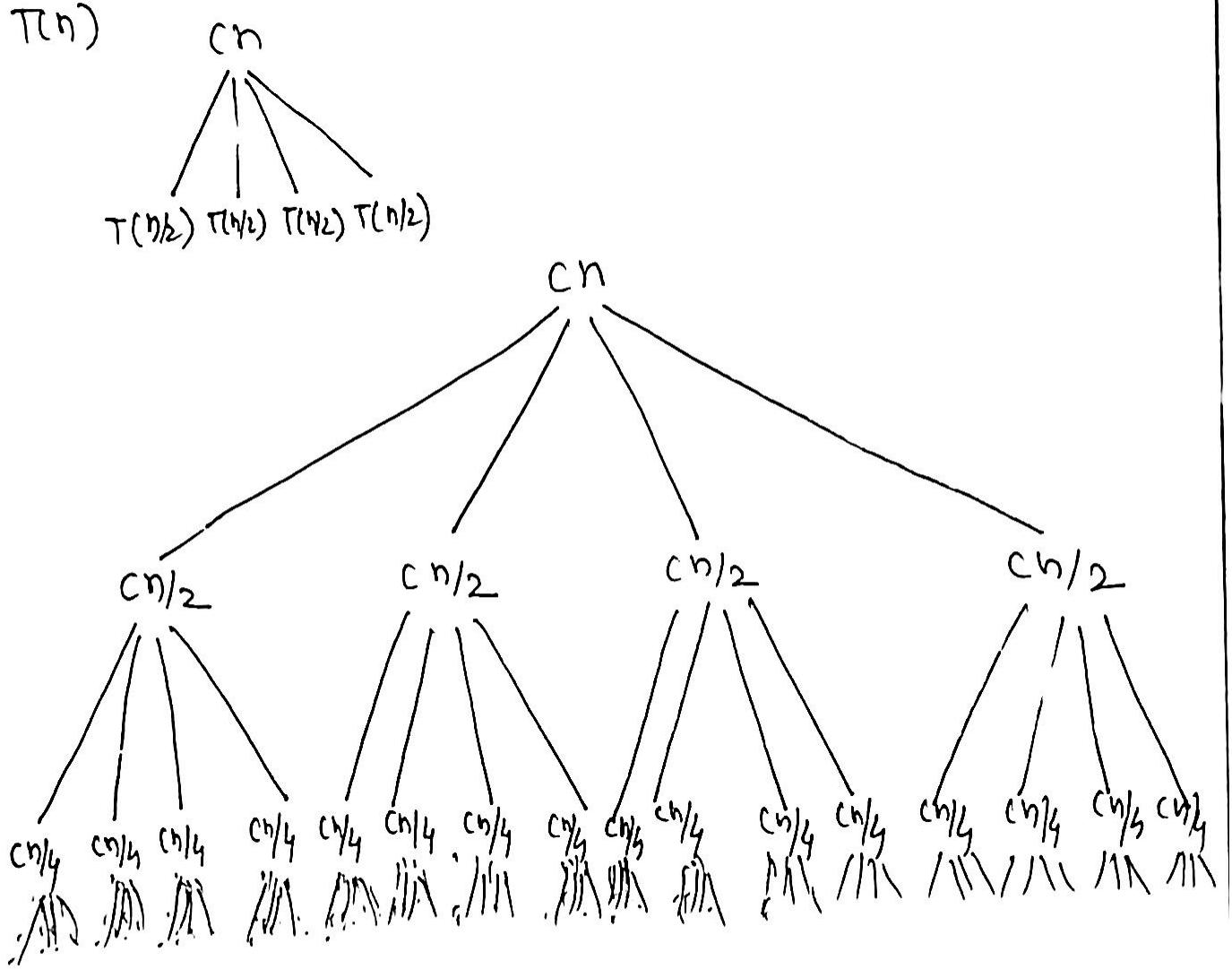
$$\leq cn^{\lg 3}$$

It holds for $c \geq 2$

Example 4.2-3 Draw the recursion tree for $T(n) = 4T(\lfloor n/2 \rfloor) + cn$, where c is a constant, and provide a tight asymptotic bound on et's soln. verify your bound by the substitution method.

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

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



T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1) .... T(1) T(1)

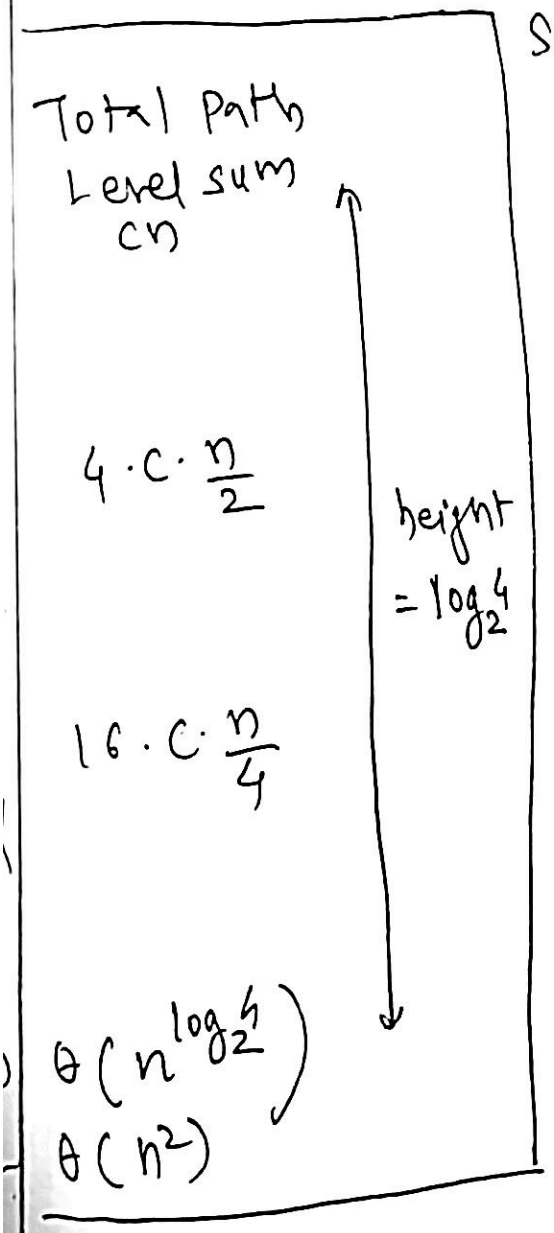
$n^{\log_2 4}$

$$\therefore T(n) = \sum_{i=0}^{\log_2 n - 1} 4^i \cdot c \cdot \frac{n}{2^i} + \theta(n^2)$$

$$= nc \sum_{i=0}^{\log_2 n - 1} 2^i + \theta(n^2)$$

$$= cn \frac{2^{\log_2 n} - 1}{2-1} + \theta(n^2)$$

$$= \theta(n^2)$$



Substitution method Proof.

Let $T(n) = cn^2 - cn$

P.T. $T(n) \leq 4T(n/2) + cn$

$$\leq 4 \cdot c \cdot \frac{n^2}{4} - \frac{4nc}{2} + cn$$

$$\leq cn^2 - 2nc + cn$$

$$\leq cn^2 + n(c) - cn$$

$$\leq cn^2 - cn$$

Hence proved.

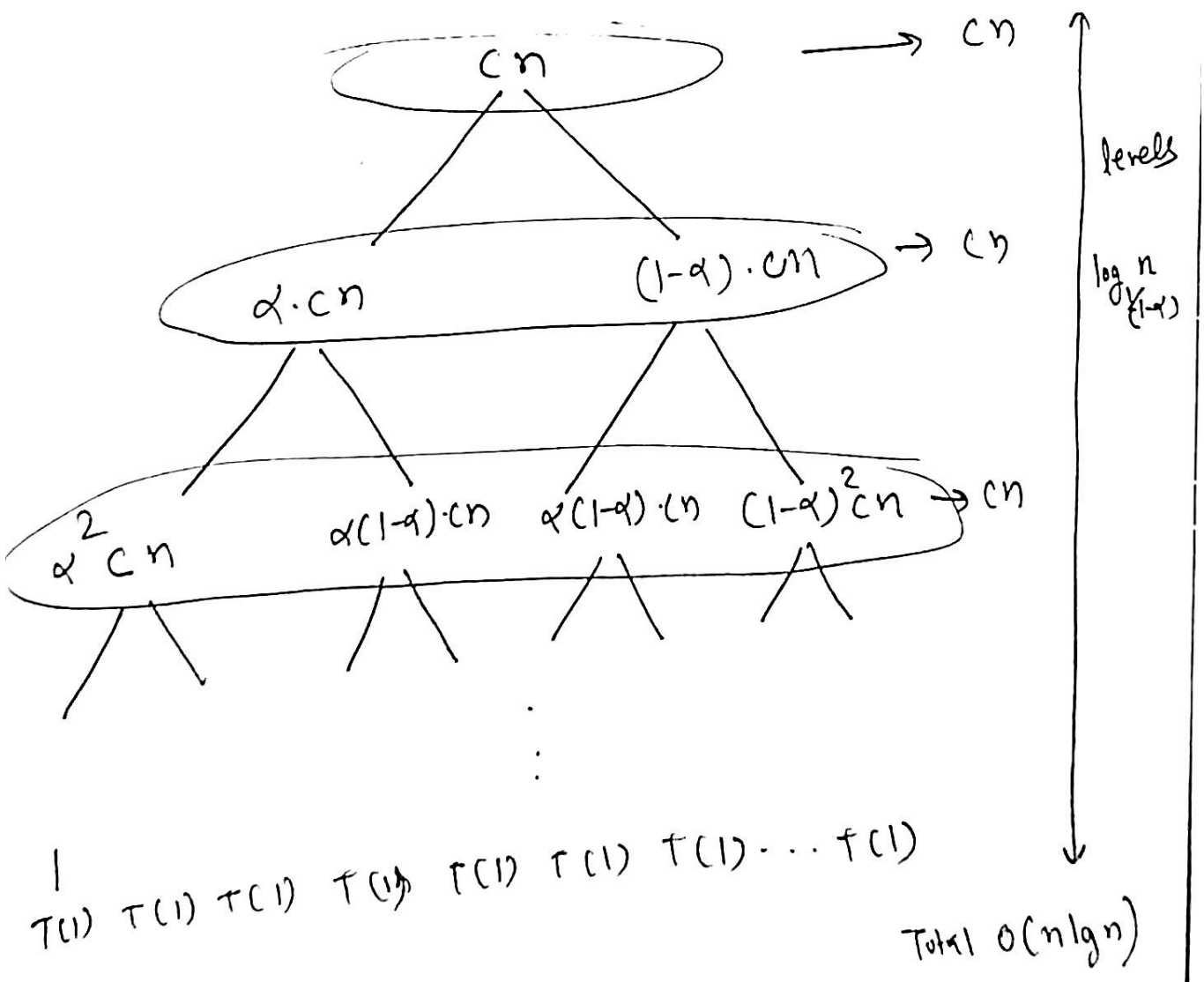
Example 4.2-5 Use a recursion tree to give an asymptotically tight bound solution to the recurrence $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$, where α is a constant in the range $0 < \alpha < 1$ and $c > 0$ is also a constant.

Soln

$$T(n) = T(n/3) + T(2n/3) + cn \quad \text{--- (1)}$$

Eqn (1) is same eqn with $\alpha = 1/3$ so that $1-\alpha = \frac{2}{3}$

$$T(n) = T(\alpha n) + T((1-\alpha)\cdot n) + cn$$



Let $\alpha \geq 1-\alpha$

$\therefore 2\alpha \geq 1$

$\alpha \geq 1/2 \quad \therefore \frac{1}{2} \leq \alpha < 1$

Recursion tree is ~~complete~~ full tree. each level has cn

So we have lower bound $\Omega(kn \cdot \log_{\frac{1}{1-\alpha}} n)$

$= \Omega(n \cdot \lg n)$

Similarly it upper bound is $O(n \lg n)$

$\therefore \boxed{T(n) = \Theta(n \lg n)}$

Substitution method

Guess $T(n) = \Theta(n \lg n) = cn \lg n$

$$T(n) = T(\alpha n) + T((1-\alpha)n) + cn$$

$$\leq d\alpha n \lg(\alpha n) + d(1-\alpha) \cdot n \lg((1-\alpha)n) + cn$$

$$= d\alpha n \lg\alpha + d\alpha n \lg n + d(1-\alpha) n \lg(1-\alpha) + d(1-\alpha) n \lg n + cn$$

$$= d\alpha n \lg\alpha + d\alpha n \lg n + d(1-\alpha) n \lg(1-\alpha) + dn \lg n - d\alpha n \lg n + cn$$

$$= dn \lg n + dn\left(\alpha \lg\alpha + (1-\alpha) \cdot \lg(1-\alpha)\right) + cn$$

$$\leq dn \lg n$$

to check for upper & lower bounds

$dn(\alpha \lg\alpha + (1-\alpha)\lg(1-\alpha)) + cn < 0$

$d(\alpha \lg\alpha + (1-\alpha)\lg(1-\alpha)) < -c$

$$d \geq \frac{-c}{\alpha \log\alpha + (1-\alpha)\cdot \lg(1-\alpha)}$$ (for upper bound)

$$d \leq \frac{c}{-\alpha \log\alpha - (1-\alpha)\lg(1-\alpha)}$$

$\therefore T(n) = \Theta(n \lg n)$

Master Th^m (More general)

$T(n) = aT(n/b) + f(n)$ | $a \geq 1$ $f(n) > 0$, $b > 1$

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

case 1 - $f(n) = O(n^{\log_b a - \epsilon})$ for $\epsilon > 0$

(i.e. $f(n)$ is polynomially smaller than $n^{\log_b a}$)

so/n — $T(n) = \Theta(n^{\log_b a})$

f(n) is polynomially smaller than $n^{\log_b a}$

case 2 - $f(n) = \Theta(n^{\log_b a} \cdot \lg^k n)$ where $k \geq 0$

so/n $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$

i.e. f(n) is polylog factor of $n^{\log_b a}$, but not smaller

case 3 - $f(n) = \Omega(n^{\log_b a + \epsilon})$ for $\epsilon > 0$ &

$af(n/b) \leq cf(n)$

$c \leq 1$

$T(n) = \Theta(f(n))$

f(n) is polynomially greater than $n^{\log_b a}$

Master Method

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function and let $T(n)$ be defined on the non-negative by the recurrence

$$T(n) = a\,T(n/b) + f(n)$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then $T(n) = \theta(n^{\log_b a})$
2. If $f(n) = \theta(n^{\log_b a})$ then $T(n) = \theta(n^{\log_b a} \lg n)$
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ then if $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \theta(f(n))$.

Proof of Master Theorem

The proof is divided into three parts.

First n is assumed to be power of some $b > 1$.

(I) Three parts:
- (i) Solve the recurrence as summation using recursion-tree.
- (ii) Find bounds on the summation
- (iii) Puts (i) & (ii) together to prove a version of master th^m for the case in which n is an exact power of b.

(II) Extend (I) to allow ceiling & floors.

(I) Master theorem with n as powers of b.

Let $T(n) = aT(n/b) + f(n)$ —— (1)

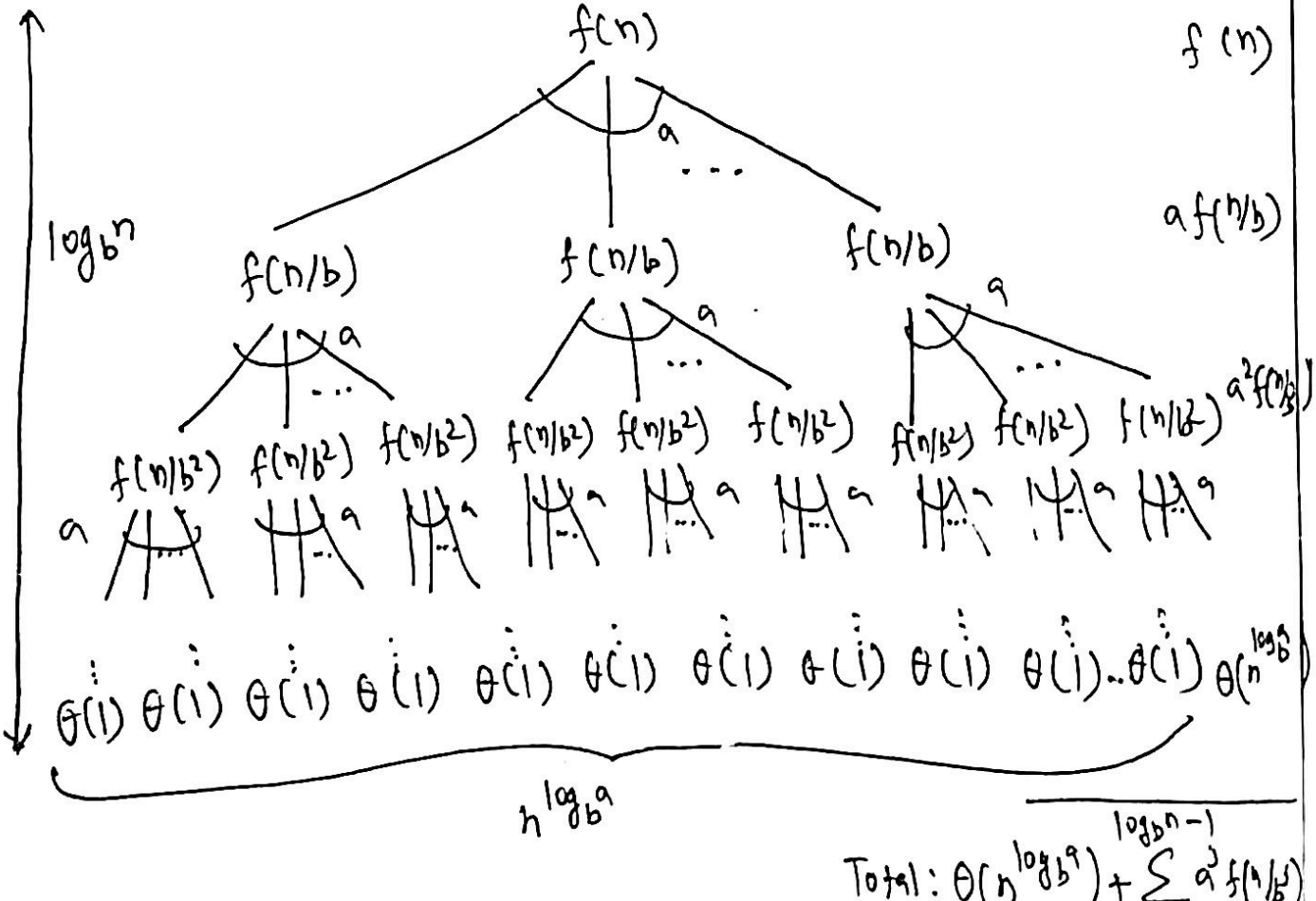
(i) Solve recurrence as summation using recursion-tree

Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be a non negative function defined on exact powers of b. Define $T(n)$ on exact powers of b as

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ aT(n/b) + f(n) & \text{if } n = b^i \end{cases}$$

where i = +ve integer

Recursion-tree



Total: $\Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$

$$\therefore T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) \quad \text{——②}$$

(ii) Find bounds on summation

Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be a non-negative function defined on exact powers of b. A function $g(n)$ defined over exact powers of b by

$$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

can then be bounded as follows.

a) If $f(n) = O(n^{\log_b a - \epsilon})$ for some const $\epsilon > 0$ then $g(n) = \Theta(n^{\log_b a})$

b) If $f(n) = \Theta(n^{\log_b a})$ then $g(n) = \Theta(n^{\log_b a} \lg n)$

c) If $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all $n \geq b$ then $g(n) = \Theta(f(n))$

Proof We have $f(n) = O(n^{\log_b a - \epsilon})$

From eq ② (a)

$$g(n) = O\left(\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a - \epsilon} \right)$$

we solve this equation

$$\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a - \epsilon} = n^{\log_b a - \epsilon} \sum_{j=0}^{\log_b n - 1} \left(\frac{a b^\epsilon}{b^{\log_b a}}\right)^j$$

$$= n^{\log_b a - \epsilon} \sum_{j=0}^{\log_b n - 1} \left(\frac{b^\epsilon a}{a}\right)^j$$

$$= n^{\log_b a - \epsilon} \sum_{j=0}^{\log_b n - 1} (b^\epsilon)^j$$

$$= n^{\log_b a - \epsilon} \left(\frac{b^{\epsilon \log_b n} - 1}{b^\epsilon - 1} \right)$$

$$= n^{\log_b a - \epsilon} \times \left(\frac{n^\epsilon - 1}{b^\epsilon - 1} \right)$$

$$\therefore \quad g(n) = O\left(n^{\log_b a}\right)$$

(b) $$g(n) = \theta \left(\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j} \right)^{\log_b a} \right)$$

$$\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j} \right)^{\log_b a} = n^{\log_b a} \sum_{j=0}^{\log_b n - 1} \left(\frac{a}{b^{\log_b a}} \right)^j$$

$$= n^{\log_b a} \sum_{j=0}^{\log_b n - 1} \left(\frac{a}{a} \right)^j$$

$$= n^{\log_b a} \log_b n$$

$$g(n) = \theta \left(n^{\log_b a} \log_b n \right)$$

$$= \theta \left(n^{\log_b a} \lg n \right)$$

(c) $$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

We assumed in case-3 that

$$a\, f(n/b) \le c\, f(n)$$

$$f(n/b) \le \frac{c}{a} f(n)$$

$$\therefore \quad g(n) = \sum_{j=0}^{\log_b n - 1} a^j f\left(n/b^j\right)$$

$$= \sum_{j=0}^{\log_b n - 1} a^j \left(\frac{c}{a}\right)^j f(n)$$

$$= f(n) \sum_{j=0}^{\log_b n - 1} c^j$$

$$\le f(n) \sum_{j=0}^{\infty} c^j$$

$$= f(n) \left(\frac{1}{1-c}\right) \quad \Big| \text{ where } c < 1$$

$$= O(f(n))$$

(iii) Putting together (i) & (ii) Eqn ⊖ becomes

a) $T(n) = \Theta\left(n^{\log_b a}\right) + \Theta\left(n^{\log_b a}\right)$

$= \Theta\left(n^{\log_b a}\right)$

b) $T(n) = \Theta\left(n^{\log_b a}\right) + \Theta\left(n^{\log_b a} \lg n\right)$

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

c) $T(n) = \Theta\left(n^{\log_b a}\right) + \Theta(f(n))$

$= \Theta(f(n))$.

(II) Extend (I) for floor and ceiling.

We have

$$T(n) = aT(\lceil n/b \rceil) + f(n) \quad \text{... (find upper bound)}$$

or

$$T(n) = aT(\lfloor n/b \rfloor) + f(n) \quad \text{(find lower bound)}$$

They are similar we will find later

$\therefore$ n,

$\lceil n/b \rceil$

$\lceil \lceil n/b \rceil / b \rceil$,

$\lceil \lceil \lceil n/b \rceil / b \rceil / b \rceil$

$\vdots$

Let us denote j^{th} element in sequence by n_j

$$n_j = \begin{cases} n & \text{if } j = 0 \\ \lceil n_{j-1}/b \rceil & \text{if } j > 0 \end{cases}$$

$n_0 \leq n$

$n_1 \leq \frac{n}{b} + 1$

$n_2 \leq \frac{n}{b} + \frac{1}{b} + 1$

$n_3 \leq \frac{n}{b^3} + \frac{1}{b^2} + \frac{1}{b} + \frac{1}{b^0}$

$\therefore n_j \leq \frac{n}{b^j} + \sum_{i=0}^{j-1} \frac{1}{b^i}$

$< \frac{n}{b^j} + \sum_{i=0}^{\infty} \frac{1}{b^i}$

$= \frac{n}{b^j} + \frac{b}{b-1}$

let $j = \lfloor \log_b n \rfloor$

$\therefore n_{\lfloor \log_b n \rfloor} < \frac{n}{b^{\lfloor \log_b n \rfloor}} + \frac{b}{b-1}$

$\leq \frac{n}{b^{\log_b n - 1}} + \frac{b}{b-1}$

$= \frac{n}{n/b} + \frac{b}{b-1}$

$= b + \frac{b}{b-1}$

$= O(1)$

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j)$$

$$g(n) = \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j)$$

$\vdots$

(It is not full proof)
check comment at-91.

Example - Use master theorem to solve following

① $T(n) = 9T(n/3) + n$

$a = 9 \quad b = 3$

$f(n) = n$

$n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$

Now $f(n) = O(n^{\log_3 9 - \epsilon}) = O(n^{2-\epsilon}) = n$

where $\epsilon = 1$

$\therefore \boxed{T(n) = \Theta(n^2)}$. case-1

② $T(n) = T(2n/3) + 1$

$a = 1 \quad b = 3/2 \quad f(n) = 1$

$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

Now $f(n) = 1 = O(1)$ Case-2

$\therefore T(n) = O(n^0 \cdot \lg n)$

$= O(\lg n)$

③ $T(n) = 3T(n/4) + n \cdot \lg n$

$a = 3 \quad b = 4 \quad f(n) = n \lg n$

$n^{\log_4 3} = O(n^{0.793})$

$n \lg n = f(n) = \Omega(n^{\log_4 3 + \epsilon})$

Where $\epsilon = 0.2$ } case 3

for case 3 { we have to show that the regularity condition holds for $f(n)$.

for large n

$$a f(n/b) = 3 \cdot \left(\frac{n}{4}\right) \cdot \lg(n/4)$$

$$\leq 3/4 \; n \cdot \lg n$$

$$= c f(n)$$

where $c = 3/4$

$\therefore T(n) = \Theta(n \cdot \lg n)$.

Examples from class

① $T(n) = 5T(n/2) + \Theta(n^2)$

$a = 5 \quad b = 2$

$n^{\log_2 5}$ vs n^2

Since $\log_2 5 - \epsilon = 2$ for some const.

case-1 $\Rightarrow T(n) = \Theta(n^{\log_2 5})$ $\epsilon > 0$

(2) $T(n) = 27T(n/3) + \Theta(n^3 \lg n)$ $\quad f(n) = \Theta(n^3 \lg n)$

$a = 27 \quad b = 3$

$n^{\log_b a} = n^{\log_3 27} = n^3$ vs. $n^3 \lg n$

<u>case 2</u>

$f(n) = n^3 \lg n = \Theta(n^3)$

$\therefore \quad T(n) = \Theta(n^3 \cdot \lg^2 n)$

(3) $T(n) = 5T(n/2) + \Theta(n^3)$

$a = 5 \quad b = 2$

$n^{\log_b a} = n^{\log_2 5}$ Vs n^3

$\therefore \log_2 5 + \epsilon = 3$

$af(n/b) = 5(n/2)^3 = \frac{5n^3}{8} \leq cn^3$

$c = 5/8 < 1$

$\therefore T(n) = \Theta(n^3)$

(4) $T(n) = 27T(n/3) + \Theta(n^3/\lg n)$

$a = 27 \quad b = 3$

$n^{\log_b a} = n^{\log_3 27} = n^3$ Vs $\frac{n^3}{\lg n} = n^3 \lg^{-1} n$

$f(n) = n^3 \cdot \lg^{-1} n \neq \Theta(n^3 \lg^k n)$ for $k > 0$

$\therefore$ Can not use master th^m

Exercise 4-3.1

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

$a = 4 \quad b = 2$

$n^{\log_2 4} = n^2$ Vs $n = O(n^{2-\epsilon})$

$\therefore \quad T(n) = \Theta(n^2)$ | case-1

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

$a = 4 \quad b = 2$

$n^{\log_2 4} = n^2$ vs n^2

$\therefore \quad T(n) = \Theta(n^2 \lg n)$ | case-2

~~case~~ 3

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

$a = 4 \quad b = 2$

$n^{\log_2 4} = n^2$ Vs n^3

Since $n^3 = \Omega(n^{2+\epsilon})$ $\epsilon = 1$

regularity cond?

$af(n/b) \leq cf(n)$

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

$\frac{4}{8} n^3 \leq cn^3$

$\therefore \quad T(n) = \Theta(n^3)$ c), 1/2

Example 4.3-2

Algorithm A — Complexity $T(n) = 7T(n/2) + n^2$

—"— A' — Complexity $T'(n) = aT'(n/4) + n^2$

What is the largest value of a for A' is asymptotically faster than A.

① $T(n) = 7T(n/2) + n^2$

$a = 7 \quad b = 2$

$n^{\log_b a} = n^{\log_2 7}$ Vs n^2

$\log_2 7 - \epsilon = 2$ case 1

$\epsilon > 0$

$\therefore T(n) = \Theta(n^{\log_2 7})$

② $T'(n) = aT'(n/4) + n^2$

$a = a \quad b = 4$

$n^{\log_b a} = n^{\log_4 a}$ Vs n^2 $\Theta(n^{\log_4 a})$

$\log_4 a - \epsilon = 2$

$\log_2 2 > \log_2 4$

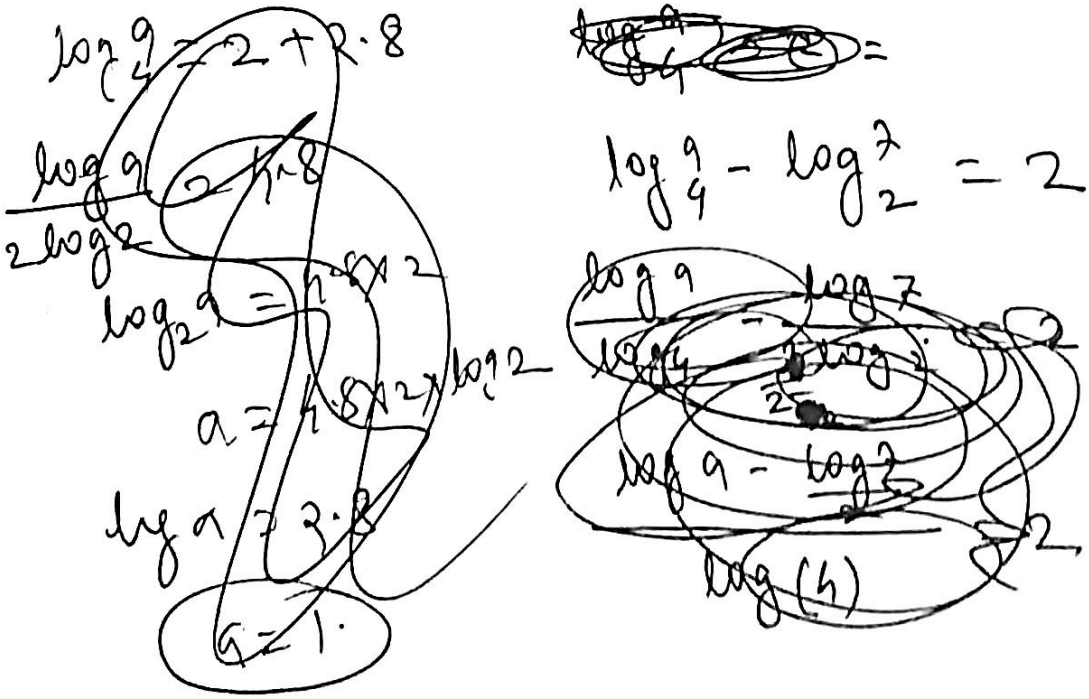
$\log_4 a - \log_2 7 = 2$ $\frac{\log 4}{\log 2} > \frac{\log 4}{2\log 2}$

$2\log 7 > \log a$

$\log 7^2 > \log a$

$7^2 > a$

$49 > a$



If $a = 49$

$$O(n^{\log_4 49}) = O(n^{\log_2 7})$$

we want $O(n^{\log_4 a})$ to be smaller

$\therefore$ $a = 50$ ✓

4.3-3 Use the master method to show that the soln to the B.S.T. $T(n) = T(n/2) + \Theta(1)$ is $T(n) = \Theta(\lg n)$.

$$T(n) = T(n/2) + \Theta(1)$$

$$a = 1 \quad b = 2$$

$$n^{\log_b a} = n^{\log_2 1} = n^0 \quad \text{Vs} \quad \Theta(1) = n^0$$

case 2

$\therefore$ $T(n) = \Theta(\lg n)$

4.3-4 Can the master method be applied to the recurrence, $T(n) = 4T(n/2) + n^2 \lg n$? Why or why not? Give an asymptotic upper bound for this recurrence.

$$T(n) = 4T(n/2) + n^2 \lg n$$

$$a = 4 \quad b = 2$$

$n^{\log_b a} = n^{\log_2 4} = n^2$ Vs $n^2 \lg n$

Master's thm can be applied since $n^2 \lg n$ is poly log factor of n^2.

$\therefore T(n) = n^2 \lg^2 n$

Exercise 4-1 Recurrences

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible and justify your answers.

(a) $T(n) = 2T(n/2) + n^3$

$a = 2 \quad b = 2$

$\therefore n^{\log_b a} = n^{\log_2 2} = n^1 = n$ Vs n^3

$\therefore 1 + \epsilon = 3 \quad | \quad \epsilon = 2$

case-3

regularity condn

$af(n/b) \leq c f(n)$

$2 f(n/2) \leq c f(n)$

$2 \frac{n^3}{8} \leq c n^3$

$c \geq 1/4 \quad c \leq 1$

$\therefore T(n) = \Theta(n^3)$

(b) $T(n) = T(9n/10) + n$

$a = 1 \quad b = 10/9.$

$n^{\log_b a} = n^{\log_{10/9} 1} = n^0 = 1 = n^0 \text{ vs } n^1$

$\epsilon = 1$ | case 3

$a f(n/b) \leq c f(n)$

$1 \left(\frac{n}{10} \times 9\right) \leq c n$

$c \geq \frac{9}{10} \quad | \quad c < 1$

But

$\therefore \quad T(n) = \theta(n)$

(c) $T(n) = 16 T(n/4) + n^2$

$a = 16 \quad b = 4$

$n^{\log_b a} = n^{\log_4 16} = n^2 \text{ vs } n^2$

$\therefore \quad f(n) = n^2$ | case-2

$\therefore \quad T(n) = \theta(n^2 \lg n)$

(d) $T(n) = 7T(n/3) + n^2$

$a = 7 \quad b = 3$

$n^{\log_b a} = n^{\log_3 7} \text{ vs } n^2$ | case-3

$\log_3 7 \; \epsilon = 2$

$\therefore T(n) = \Omega(n^{\log_3 2} + 1) = \Omega(n^2)$

regularity condn

$af(n/b) \leq cf(n)$

$2\left(\frac{n}{3}\right)^2 \leq cn^2$

$\frac{2}{9}n^2 \leq cn^2$

$c \geq \frac{2}{9}$ | But $c < 1$

(e) $T(n) = 7T(n/2) + n^2$

$a = 7 \quad b = 2$

$n^{\log_2 7}$ Vs n^2

$\log_2 7 - \epsilon = 2$ | case-1

$\therefore T(n) = \theta(n^{\log_2 7})$

(f) $T(n) = 2T(n/4) + \sqrt{n}$

$a = 2 \quad b = 4$

$n^{\log_4 2} = n^{1/2}$

$f(n) = \sqrt{n} = n^{1/2}$

Vs $n^{1/2}$

$0 + \epsilon = 1/2$ case-3

$T(n) = \theta(n^{1/2})$

regularity condn

$af(n/b) \leq cf(n)$

$2\frac{\sqrt{n}}{2} \leq c\sqrt{n}$ $c < 1$

(f) $T(n) = 2T(n/4) + \sqrt{n}$

$a = 2 \quad b = 4$

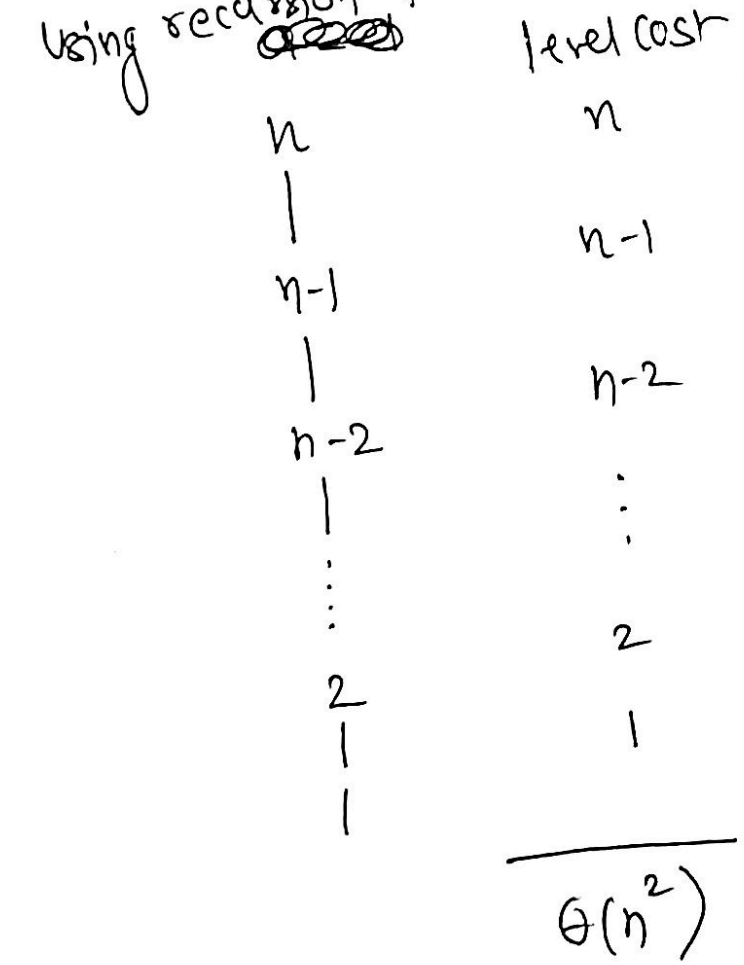
$n^{\log_4 2} = n^{1/2} \quad vs \quad \sqrt{n} = n^{1/2}$

case-2

$\therefore \quad T(n) = \Theta(\sqrt{n} \cdot \lg n)$

(g) $T(n) = T(n-1) + n$

Using recursion tree



$\Theta(n^2)$

First we P.T.

lower bound

$T(n) = \Omega(n^2)$

Let $T(n) \geq cn^2$

$\therefore T(n) = T(n-1) + n$

$\geq c(n-1)^2 + n$

$= cn^2 - 2cn + c + n$

$\geq cn^2$

if This zero

i.e. $n(1-2c) + c \geq 0$

upper bound

$T(n) = O(n^2)$

we again get

$T(n) \leq cn^2 - 2cn + c + n$

$\therefore T(n) = \Omega(n^2)$

$= O(n^2)$

$\therefore$

$T(n) = \Theta(n^2)$

(7) $T(n) = T(\sqrt{n}) + 1$

Let $m = \log_2 n = \lg n$

$\therefore \quad 2^m = n$

$\therefore$ New Recurrence will be

$S(m) = T(2^m)$

i.e. $T(2^m) = T(2^{m/2}) + 1$

$\therefore \quad S(m) = S(m/2) + 1$

Solving using master th^m.

$a = 1 \quad b = 2 \quad f(n) = 1 = n^0$

$\therefore \quad n^{\log_b a} = n^{\log_2 1} = n^0 \text{ Vs } n^0$

case-2

$\therefore$

$S(m) = \theta(1 \cdot \lg m)$

$S(m) = \theta(\lg m)$

$\therefore \boxed{T(n) = \theta(\lg \lg n)}$

Exercise 4-4 More recurrence examples

(a) $T(n) = 3T(n/2) + n \lg n$

$a = 3 \quad b = 2 \quad f(n) = n \lg n$

$n^{\log_b a} = n^{\log_2 3} \cong n^{1.585}$ Vs $f(n) = n \lg n$

$\therefore n \lg n = O(n^{\lg 3 - \epsilon})$ | case-1

$\therefore T(n) = \Theta(n^{\log_2 3}) = \Theta(n^{\lg 3})$

(b) $T(n) = 5T(n/5) + n/\lg n$

$a = 5 \quad b = 5 \quad f(n) = n \cdot \lg^{-1} n$

$n^{\log_5 5} = n^1$ Vs $f(n) = n^1 \lg^{-1} n$

But $n^1 \lg^{-1} n \neq \Theta(n^1 \lg^k n) \quad k \geq 0$

We can not solve this using recurrence relation.

(c) $T(n) = 4T(n/2) + n^2\sqrt{n}$

$a = 4 \quad b = 2 \quad f(n) = n^{5/2}$

$n^{\log_2 4} = n^{\log_2 4} = n^2$ Vs $n^{5/2}$

$2 + \epsilon = 5/2$ | case-3

$\therefore T(n) = \Theta(n^{5/2})$

regularity condn

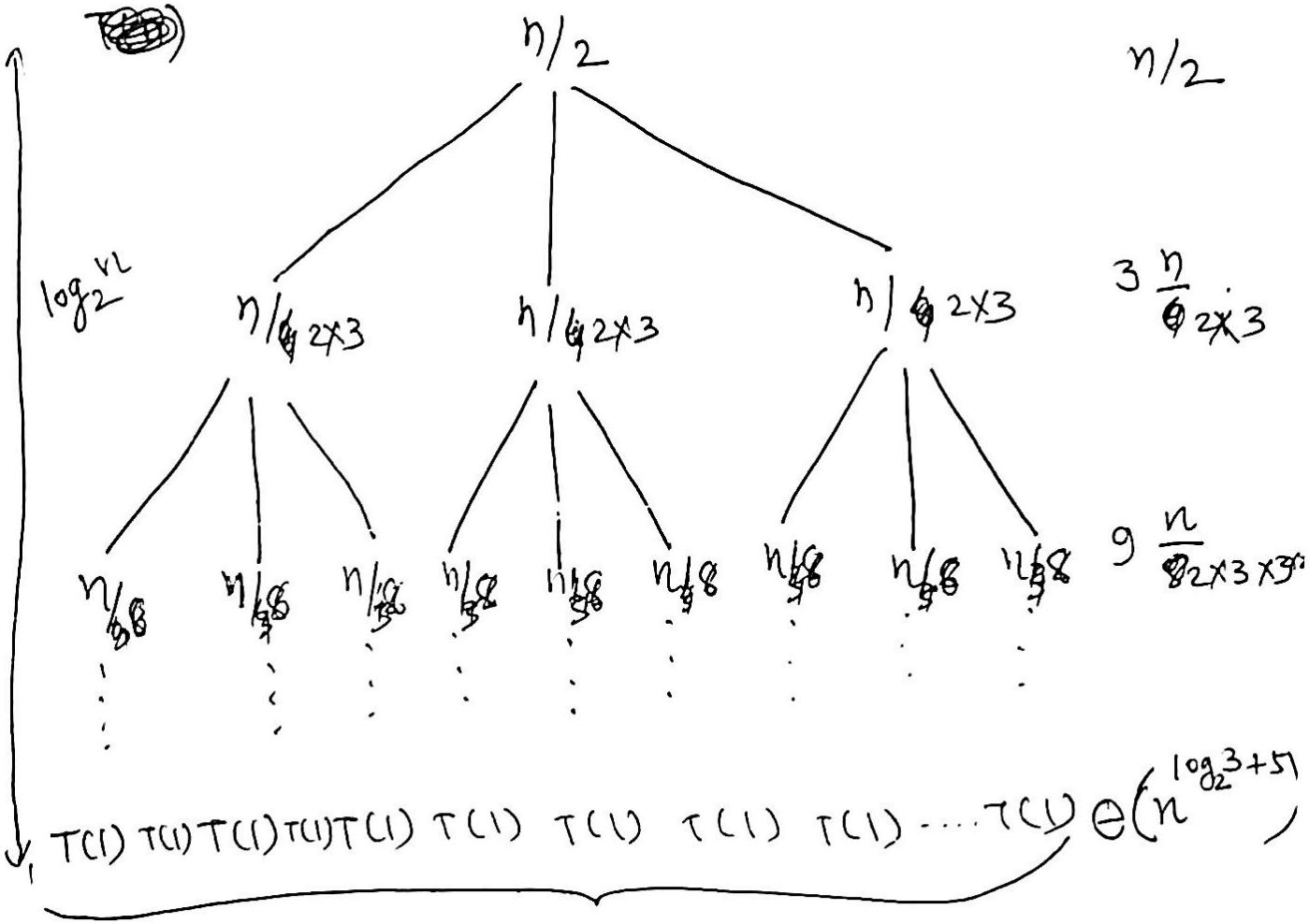
$a f(n/b) \leq c f(n)$

$4 \frac{n^2}{4} \frac{\sqrt{n}}{\sqrt{2}} \leq c n^2 \sqrt{n}$

$c \geq \frac{1}{\sqrt{2}}$

(d) $T(n) = 3T(n/3+5) + n/2$

We can not solve this using master th^m.



$$T(n) = \sum_{i=0}^{\log_2 n - 1} 3^i \frac{n}{2^i} + \Theta(n^{\log_2 3 + 5})$$

$$= n \sum_{i=0}^{\log_2 n - 1} \left(\frac{3}{2}\right)^i + \Theta(n^{\log_2 3 + 5})$$

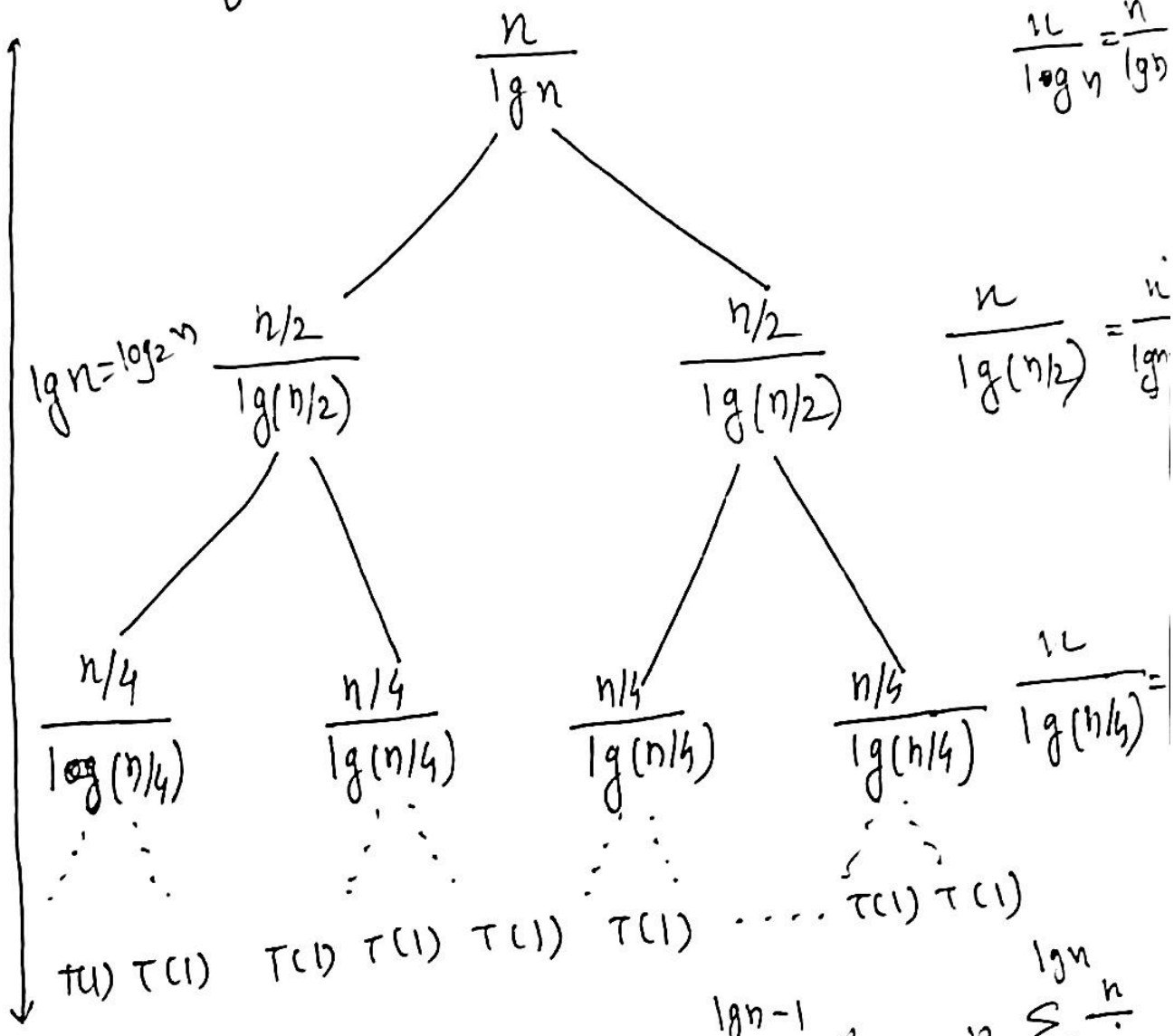
$$= n$$

(e) $T(n) = 2T(n/2) + n/\lg n$

we can't apply case-2

∴ Using recursion tree.

Level costs.



$\frac{n}{\log n} = \frac{n}{\lg n}$

$\frac{n}{\lg(n/2)} = \frac{n}{\lg n}$

$\frac{n}{\lg(n/4)} =$

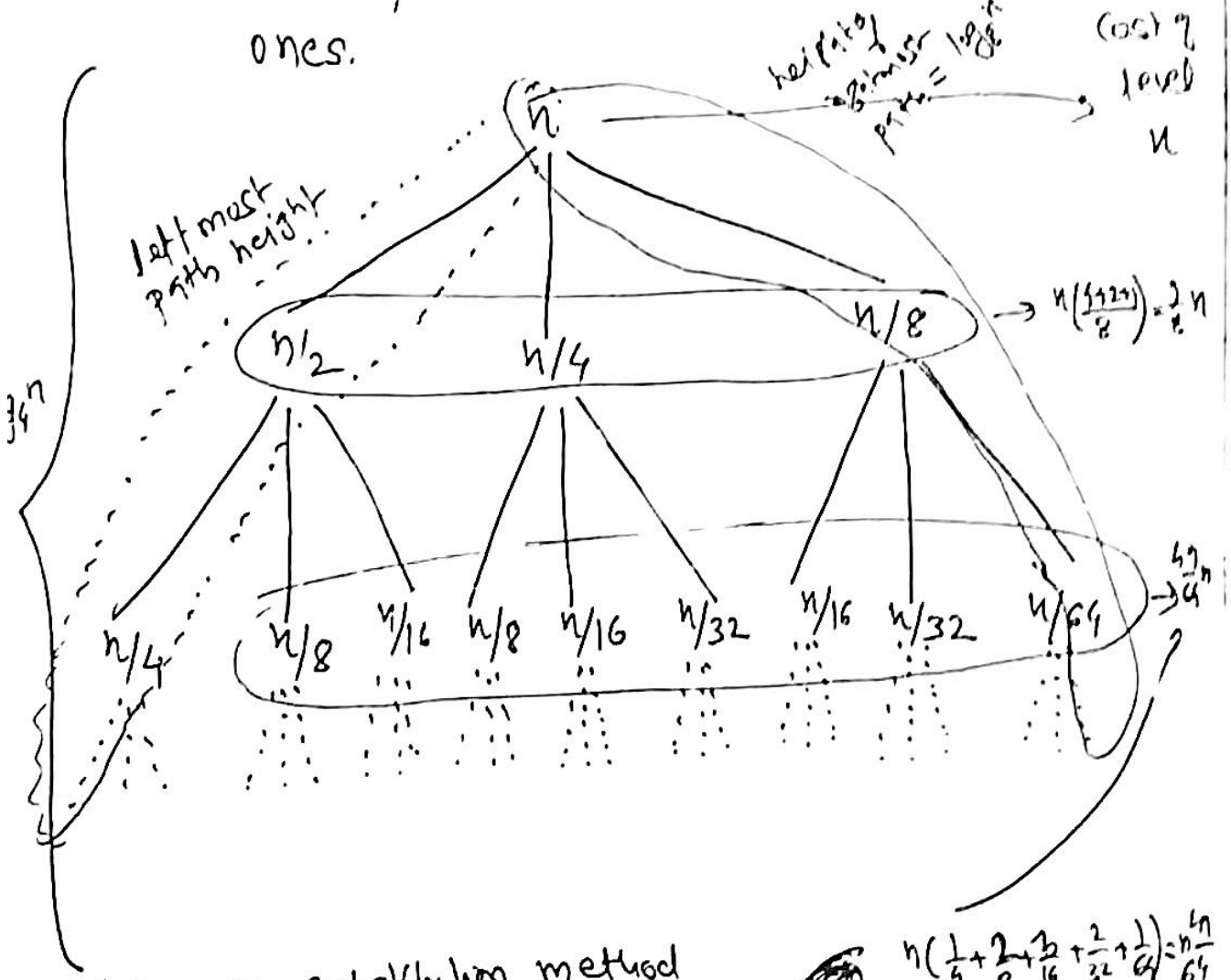
Sum of all levels $= \sum_{i=0}^{\lg n - 1} \frac{n}{\lg n - i} = n \sum_{i=1}^{\lg n} \frac{1}{i}$

$= n \sum_{i=1}^{\lg n} 1/i = n\,\Theta(\lg \lg n)$

$= \Theta(n \lg \lg n)$

(f) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

We can not use the master theorem directly as it consists of more than one ones.



$n\left(\frac{1}{1} + \frac{2}{8} + \frac{3}{16} + \frac{2}{32} + \frac{1}{64}\right) = \frac{49}{64} n$

$= \frac{7^2}{8^2} n$

We use substitution method to prove that $T(n) = O(n)$

Let $T(n) = nc$

$T(n) = T(n/2) + T(n/4) + T(n/8) + n$

$\leq \frac{cn}{2} + \frac{cn}{4} + \frac{cn}{8} + n$

$= \frac{4+2+1}{8} cn + n$

$= \frac{7}{8} cn + n$

$\leq cn$ if $c \geq 8$

$\therefore \sum_{i=1}^{\log n} \left(\frac{7}{8}\right)^i n = \Theta(n)$

we can easily show

$T(n) = \Omega(n)$

As $T(n) = O(n)$ & $T(n) = \Omega(n)$

$\therefore T(n) = \Theta(n)$.

(g) $T(n) = T(n-1) + 1/n$

Again we can not solve this recurrence using Master-thm.

This is harmonic series

$T(n) = H_n$ where

$H_n = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

We have Base case $T(1) = 1 = H_1$ —①

Inductive step $T(n-1) = H_{n-1}$

$\therefore T(n) = T(n-1) + 1/n$

$= H_{n-1} + 1/n$

$= H_n$

We know $H_n = \Theta(\lg n)$

$\therefore T(n) = \Theta(\lg n)$

(h) $T(n) = T(n-1) + \lg n$

We guess that

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

① Upper bound proof:-

we P.T.

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

Our inductive hypothesis

$T(n) \leq cn\lg n$ for some constt, c

$$T(n) = T(n-1) + \lg n$$

$$\leq c(n-1)\lg(n-1) + \lg n$$

$$= cn\lg(n-1) - c\lg(n-1) + \lg n$$

$$\leq cn\lg(n-1) - c\lg(n/2) + \lg n$$

$\lg(n-1) \geq \lg(n/2)$ for $n \geq 2$

$$= cn\lg(n-1) - c\lg n + c + \lg n$$

$$< cn\lg n - c\lg n + c + \lg n$$

$$\leq cn\lg n$$

$-c\lg n + c + \lg n \leq 0$

$c \leq (c-1)\lg n$

$\lg n \geq c/(c-1)$

This works for $c = 2$ & $n \geq 4$

(ii) To prove lower bound that is

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

$T(n) \geq cn\lg n + dn$ for constt c & d.

$$T(n) = T(n-1) + \lg n$$

$\geq c(n-1)\lg(n-1) + d(n-1) + \lg n$

$= cn\lg(n-1) - c\lg(n-1) + dn - d + \lg n$

$\geq cn\lg(n/2) - c\lg(n-1) + dn - d + \lg n$

$\downarrow$ $(\because \lg(n-1) \geq \lg(n/2)$ for $n \geq 2)$

$= cn\lg n - cn - c\lg(n-1) + dn - d + \lg n$

$\geq cn\lg n$

$-cn - c\lg(n-1) + dn - d + \lg n \geq 0$

$-cn - c\lg(n-1) + dn - d + \lg(n-1) \geq 0$

$(d-c)n \geq (c-1)\lg(n-1) + d$

This works for $c=1$ $d=2$ for $n \geq 2$

$\therefore T(n) = O(n\lg n)$ & $T(n) = \Omega(n\lg n)$

$\therefore T(n) = \Theta(n\lg n)$

① $T(n) = T(n-2) + 2\lg n$

Guess $T(n) = \Theta(n\lg n)$

① show the upper bound.

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

$T(n) = T(n-2) + 2\lg n$

$\leq c(n-2)\cdot\lg(n-2) + 2\lg n$

$\leq c(n-2)\lg n + 2\lg n$

$= (cn - 2c + 2)\lg n$

$= cn\lg n + (2-2c)\lg n$

$\leq cn\lg n$ if $c > 1$

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

(ii) To show lower bound

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

We will show that

$T(n) \geq cn\lg n + dn$ for consts c and d.

We assume $n \geq 4$

It implies

(i) $\lg(n-2) \geq \lg(n/2)$

(ii) $n/2 \geq \lg n$

& (iii) $\frac{n}{2} \geq 2$.

$T(n) \geq c(n-2)\lg(n-2) + d(n-2) + 2\lg n$

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

$[\because -\lg n < -\lg(n-2)]$

$= cn\lg(n-2) - 2(c-1)\lg n + dn - 2d$

$\geq cn\lg(n/2) - 2(c-1)\lg n + dn - 2d$

$= cn\lg n - cn - 2(c-1)\lg n + dn - 2d$

$\geq cn\lg n$

≥ 0

$d \geq 2(2c-1)$

$\frac{d}{2} \geq 2c-1 = c + (c-1)$

$d \geq c + (c-1) + \frac{d}{2}$ ——— ④ × n

$nd \geq cn + (c-1)n + \frac{dn}{2}$

$dn \geq cn + 2(c-1)\frac{n}{2} + \frac{dn}{2}$

Using ② & ③

$\therefore dn \geq cn + 2(c-1)\lg n + 2d$

$\therefore T(n) = \Omega(n \lg n)$

$\therefore T(n) = \Theta(n \lg n)$

$\log_2 16 = 4$ $\quad \log_b \left(b^{\log_b q}\right)$

④ Anti log

$q = 16$ $\quad \log_b q \quad \log_{10} b$

Anti $\log_b(1)$ $\quad \frac{\log q}{\log b} \quad \frac{\log b}{\log a}$