1. First use the iteration method to solve the recurrence, draw the recursion tree to analyze.

$$T(n) = T(\tfrac{n}{8}) + T(\tfrac{n}{3}) + 3n$$

Then use the substitution method to verify your solution.

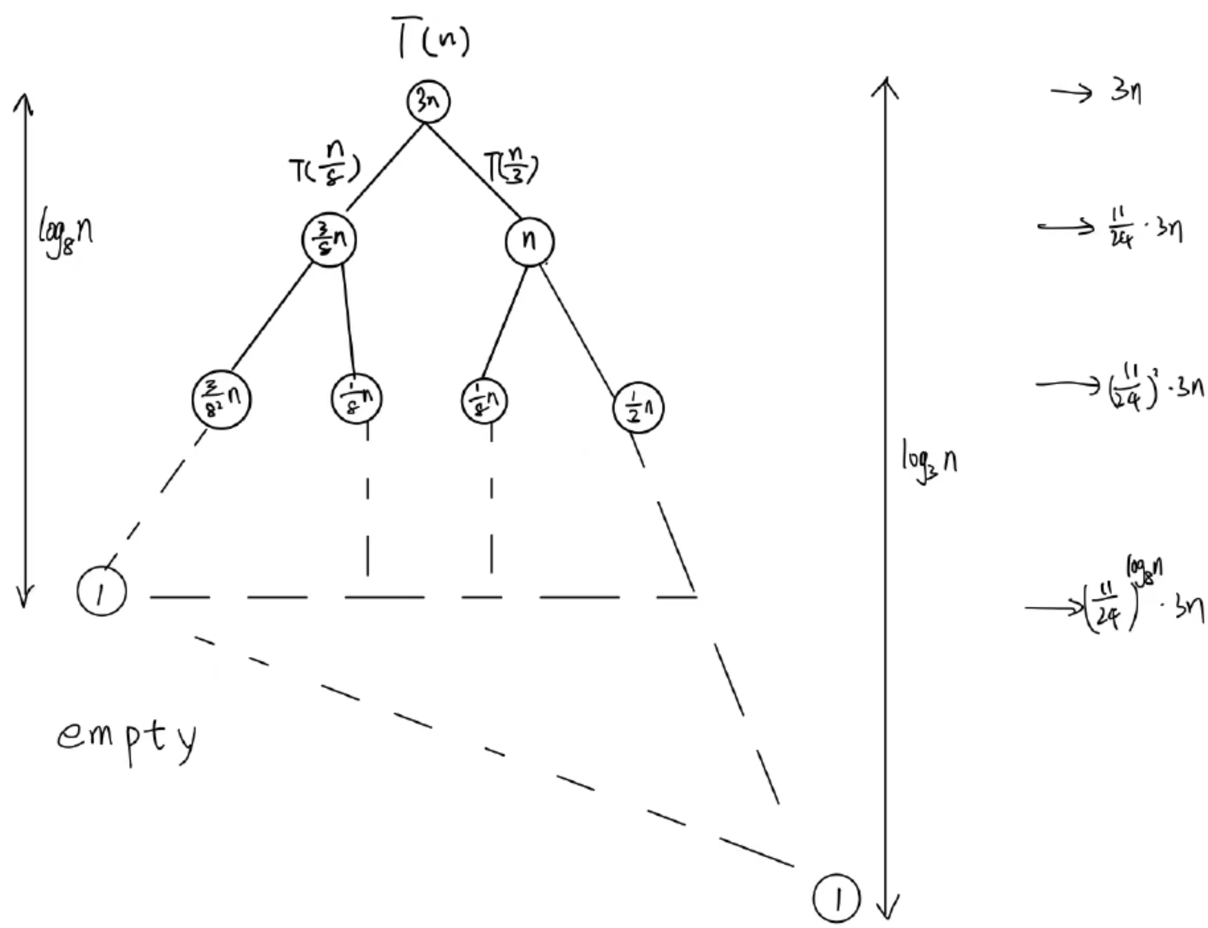

Figure 1: asymmetric and inbalanced recursion tree

divide the problem size by 8 reach the bottom the fastest at depth $log_8 n$, whereas the recursions that divide the problem size by 3 reach the bottom at depth $log_3 n$.

Other recursions that keep dividing the problem size by combinations of 8 and 3 are in the middle. Note that the cost at each depth is reduced by a factor of $\frac{1}{8}+\frac{1}{3}=\frac{11}{24}$. In other words, the merging cost at depth k is $3n * \frac{11}{24}^{k}$. Then we can bound $T(n)$ from above and below by

$$upper = 3n(1+\frac{11}{24}+\frac{11}{24}^{2}+\frac{11}{24}^{3}+\cdots+\frac{11}{24}^{log_3 n}) \tag{1}$$

$$lower = 3n(1+\frac{11}{24}+\frac{11}{24}^{2}+\frac{11}{24}^{3}+\cdots+\frac{11}{24}^{log_8 n}) \tag{2}$$

$$lower \leq T(n) \leq upper \tag{3}$$

As $n \to \infty$, both the upper and lower bounds tend to $\frac{72}{13}n$, which implies $T(n)=\Theta(n)$

Substitution method:

The upper bound:
IH: $T(k) \leq dk$ for all $k<n$

$$T(n) = T(\frac{n}{8})+T(\frac{n}{3})+3n \leq d\frac{n}{8}+d\frac{n}{3}+3n \tag{4}$$

$$d\frac{n}{8}+d\frac{n}{3}+3n \leq dn \tag{5}$$

$$d \geq \frac{72}{13} \tag{6}$$

If we set $d \geq \frac{72}{13}, T(n) \leq dn \Rightarrow T(n)=\Omega(n)$

The lower bound:
IH: $T(k) \geq ck$ for all $k<n$

$$T(n) = T(\frac{n}{8})+T(\frac{n}{3})+3n \geq c\frac{n}{8}+c\frac{n}{3}+3n \tag{7}$$

$$c\frac{n}{8}+c\frac{n}{3}+3n \geq cn \tag{8}$$

$$c \leq \frac{72}{13} \tag{9}$$

If we set $c \leq \frac{72}{13}, T(n) \geq cn \Rightarrow T(n)=O(n)$
Bound by the upper and the lower, we can get $T(n)=\Theta(n)$

2.

$T(1) = 1 > 0 = d \times 1 \times \log_2 1$, so, we set the boundary as 2. Our base is $T(2) \le d * 2\,(\log_2 2)^2$.

Induction Hypothesis: If for all k < n we have $T(k) \le dk\,(\log_2 k)^2$.

$T(n) = 2T\left(\frac{n}{2}\right) + cn\log_2 n \le 2d\frac{n}{2}(\log_2 \frac{n}{2})^2 + cn\log_2 n = dn(\log_2 n - 1)^2 + cn\log_2 n$

$= dn(\log_2 n)^2 - 2dn\log_2 n + dn + cn\log_2 n.$

If we set $-2dn\log_2 n + dn + cn\log_2 n \le 0$, formula above is smaller than $dn\,(\log_2 n)^2$,

$\leftrightarrow\ (2\log_2 n - 1)nd \ge cn\log_2 n$

$\leftrightarrow\ (2\log_2 n - 1)d \ge c\log_2 n$

$\leftrightarrow\ d \ge \frac{c\log_2 n}{2\log_2 n - 1} = c\frac{1}{2-\frac{1}{\log_2 n}},\ c\frac{1}{2-\frac{1}{\log_2 n}}$ monotone decrease from 2 to $+\infty$,

$c\frac{1}{2-\frac{1}{\log_2 n}} \le c\frac{1}{2-\frac{1}{\log_2 2}} = c$. That is, if we set $d \ge c,\ T(n) \le dn\,(\log_2 n)^2$

Therefore, $T(n) = O(n\,(\log_2 n)^2)$.

3.

$$T(n) = 3T(\sqrt{n}) + logn^2$$

set $m = logn$, we have,

$$T(2^m) = 3T(2^{\frac{m}{2}}) + m^2$$

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

$$S(m) = 3S(\frac{m}{2}) + m^2$$

Use master method: $a = 3, b = 2, f(m) = m^2$
In case 3: $af(m/b) \leq cf(m)$ for some $c \leq 1$ and all sufficiently large n

$$3(\frac{m}{2})^2 \leq cm^2$$
$$\frac{3}{4}m^2 \leq cm^2$$

Then we can use the master method case 3 in this recurrence:

$$S(m) = \Theta(m^2)$$
$$T(n) = \Theta((logn)^2)$$

4. You have three algorithms to a problem and you do not know their efficiency, but fortunately, you find the recurrence formulas for each solution, which are shown as follows:

A: $T(n) = 2T(\frac{n}{2}) + \theta(n)$ = 2T(n/2) + theta(n) = kT(n/k) + theta(n) = k*(kT(n/k/k) + n/k)+ n = k*...*kT(1) + n + … + n= k^(logk(n)) + n*logk(n) = theta(nlogn)

B: $T(n) = 2T(\frac{9n}{10}) + \theta(n)$

C:$T(n) = 2T(\frac{n}{2}) + \theta(n^2)$

Please give the running time of each algorithm (In θ notation), and which of your algorithms is the fastest (You probably can do this without a calculator)?

For algorithm A: $T(n) = 2T(\frac{n}{2}) + \theta(n)$

a = 2, b = 2, $f(n) = \theta(n)$, $d = 1, log_b a = log_2 2 = 1 = d$, so $T(n) = \theta(n^{log_b a} * logn) = \theta(nlogn)$

For algorithm B: $T(n) = 2T(\frac{9n}{10}) + \theta(n)$

a = 2, b = $\frac{10}{9}$,$f(n) = \theta(n)$, $d = 1$, $log_b a = log_{\frac{10}{9}} 2 > 1 = d$, so $T(n) = \theta(n^{log_b a}) = \theta(n^{log_{\frac{10}{9}} 2})$

For algorithm C: $T(n) = 2T(\frac{n}{2}) + \theta(n^2)$

a =2, b =2, $f(n) = \theta(n^2)$, d =2, $log_b a = log_2 2 = 1 < 2 = d$, so

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

The solution A is the fastest.