3-Sum

input: $a_1, \ldots, a_n, c$

output: find i, j, k s.t. $a_i + a_j + a_k = c$ or report "none"

1: enumerate i, j, k and check.

$O(n^3)$

2: $\underbrace{c - a_i - a_j}_{} = a_k$

$$\begin{array}{|c|} \hline a_1 \\ \hline a_2 \\ \hline \vdots \\ \hline a_n \\ \hline \end{array}$$

$a_1 \le a_2 \ldots \le a_n$

① sort once $O(n \log n)$

② enumerate i, j $O(n^2)$

③ perform binary search for a_k $O(\log n)$

time $= O(n^2 \log n + n \log n)$
$= O(n^2 \log n)$

3: $a_i + a_j = \underbrace{c - a_k}_{\text{generate } k}$ let $c - a_k = b$

2-sum problem $\exists\, i, j$ s.t. $a_i + a_j = b$

Claim: sorted, 2-sum solved in $O(n)$ time

$\Rightarrow$ time $= O(\underset{b}{n} \cdot \underset{\text{2-sum}}{n} + \underset{\text{sort}}{n \log n}) = \underline{\underline{O(n^2)}}$ ← total

① sort once $O(n \log n)$

② 2-sum $O(n)$

③ check b $O(n)$

How is 2-Sum $O(n)$

1	3	6	8	9	11	14	20

L R

$b = 19$

initially $L = 1$, $R = n$

$$\begin{cases} \text{if } (a_L + a_R = b): \text{ done} \\ \text{if } (a_L + a_R > b): \text{ decrease } R \text{ by one} \\ \text{if } (a_L + a_R < b): \text{ increase } L \text{ by one} \end{cases}$$

repeat until $L > R$

Need to prove

$O(n)$ in sorted array

Merge Sort

input: $a_1, \dots, a_n$ $\quad n = 2^k$

output: sorted

$T(n)$: time to sort n numbers

$$T(n) = \underset{\text{split}}{2T\left(\tfrac{n}{2}\right)} + \underset{\text{merge}}{cn}$$

$$T(n) \qquad + cn$$

$$T\left(\tfrac{n}{2}\right) + \tfrac{cn}{2} \quad T\left(\tfrac{n}{2}\right) + \tfrac{cn}{2} \qquad + cn$$

$$T\left(\tfrac{n}{4}\right) + \tfrac{cn}{4}\, T\left(\tfrac{n}{4}\right) \quad T\left(\tfrac{n}{4}\right) \quad T\left(\tfrac{n}{4}\right) \qquad + cn$$

$$T(1) \quad \cdots \qquad + cn$$

$\log_2 n$ levels

$\frac{n}{2^i} = 1$, $n = 2^i$, $i = \log_2 n$

$\therefore$ total time $n \log n$

hypothesis: $T(n) = cn\log_2 n$

induction: $T(m) = 2T(\frac{m}{2}) + cm$

$= 2(\frac{cm}{2}\log_2 \frac{m}{2}) + cm$

$= cm(\log_2 m - 1) + cm$

$= cm\log_2 m$

Why is this wrong?

$\downarrow$

hypothesis: $T(n) = O(n) = cn$

induction: $T(m) = 2T(\frac{m}{2}) + O(m)$

$= 2(\frac{cm}{2}) + cm$

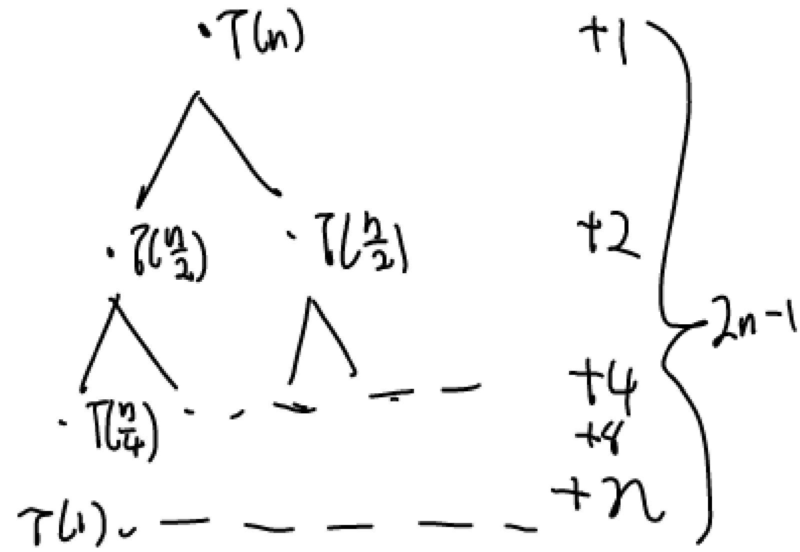
$= 2cm$

$= O(m)$ ~~X~~ because constant gets doubled every level.

Side note

Constant needs to be independant of n.

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



hypothesis : $T(n) = cn - 1$

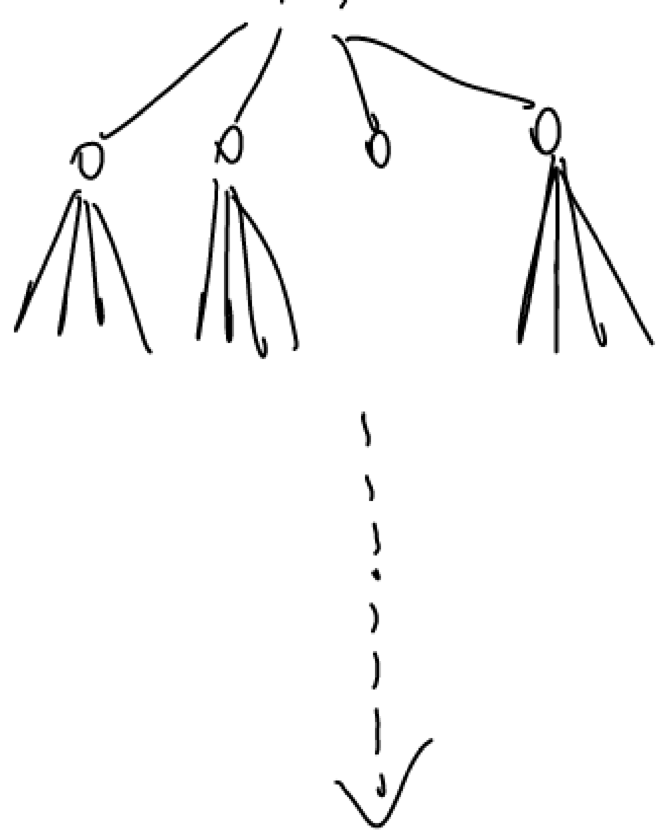
induction : $T(m) = 2T\left(\frac{m}{2}\right) + 1$

$$= 2\left(\frac{cm}{2} - 1\right) + 1$$

$$= cm - 1 \quad \square$$

$T(n) = a \cdot T\left(\frac{n}{b}\right) + n^c \qquad a > 0,\ b \geq 1,\ c \geq 0$ constants

$T(n)$



size $\frac{n}{b}$ — a subproblems

size $\frac{n}{b^2}$ — a^2 subproblems

$\frac{n}{b^3}$ — a^3 subproblems

$\log_b n$ subproblems

$1 - - - - - - - a^{\log_b n} = n^{\log_b a}$ subprobs

$$\frac{n}{b^i} = 1 \Rightarrow n = b^i \Rightarrow \log n = \log b^i \Rightarrow i = \frac{\log n}{\log b} \Rightarrow i = \log_b n$$

levels $= \log_b n$

work

$$n^c = n^c$$

$$a\left(\frac{n}{b}\right)^c = n^c\left(\frac{a}{b^c}\right)$$

$$a^2\left(\frac{n}{b^2}\right)^c = n^c\left(\frac{a}{b^c}\right)^2$$

$$a^3\left(\frac{n}{b^3}\right)^c = n^c\left(\frac{a}{b^c}\right)^3$$

$$\vdots$$

$$a^{\log_b n}(1)^c = n^{\log_b a}$$

ratio $= \frac{a}{b^c} = r$

sum $= n^c(1 + r + r^2 + \dots + r^{\log_b n})$

$\hookrightarrow \frac{1}{1-r}$

case 1: $r = 1$

$\Rightarrow O(n^c \cdot \log n)$

case 2: $r < 1$

$\Rightarrow O(n^c)$

case 3: $r > 1$

$\Rightarrow O(n^{\log_b a})$

master theorem

$$T(n) = T\left(\frac{2n}{3}\right) + T\left(\frac{n}{3}\right) + n$$