$p_1 = 2$

$p_2 = 3$

$p_3 = 5$ $\qquad p_1 < p_2 < p_3$

Thm: $\forall n \geq 2$ there exists a unique decomposition of n into prime factors $\quad n = \prod_{i=1}^{\infty} p_i n^i$,

n_1, n_2, n_i $\qquad n_1 = 4$

$n_2 = 12$

$A_i = \prod_{J=1}^{i} p_J n^J$

$N_{i+1} = \prod_{J=1}^{i+1} p_J n^J$

$n_1 =$ you hit 1 n_1 times.

$n_2 =$ " $n_2 - n_1$ times.

$n_i =$ " $n_i - n_{i-1}$ times.

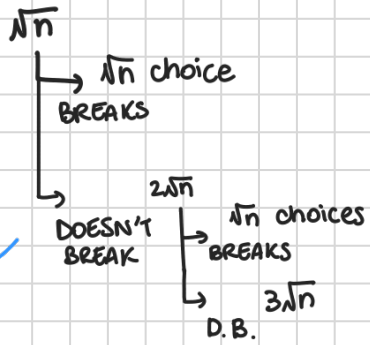
$\times\times\times\times\times\times\times\times\times\times\times\times$

$12 = 2^2 \cdot 3^1$

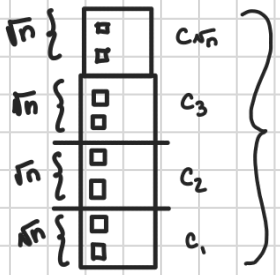
Algorithms

DATE: 12/4/2022

1 PHONE
2 PHONE
3 PHONE



$O\left(\frac{n}{t}+t\right)$

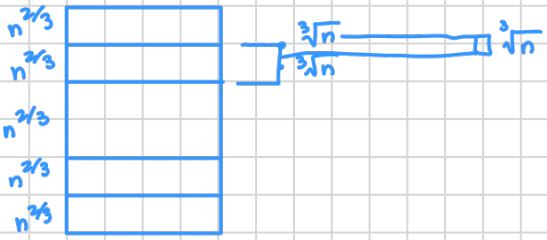


1 PHONE → $O(n)$
2 = → $O(\sqrt{n})$
3 = → $O(\sqrt[3]{n^2})$
K = → $O(K\sqrt[K]{n})$

K-1

$$\frac{n}{2^{K-1}}$$

NOTE: we throw from the last floor of the chunk



$K = \log n$ PHONES → $O(K\sqrt[K]{n})$

$$\sqrt[K]{n} = n^{1/K} = \left(2^{\log_2 n}\right)^{1/K} =$$

$$= \left(2^{\log_2 n}\right)^{\frac{1}{\log_2 n}} = 2$$

$$O\left(\log_2 n \sqrt[\log_2 n]{n}\right) = O(\log_2 n)$$

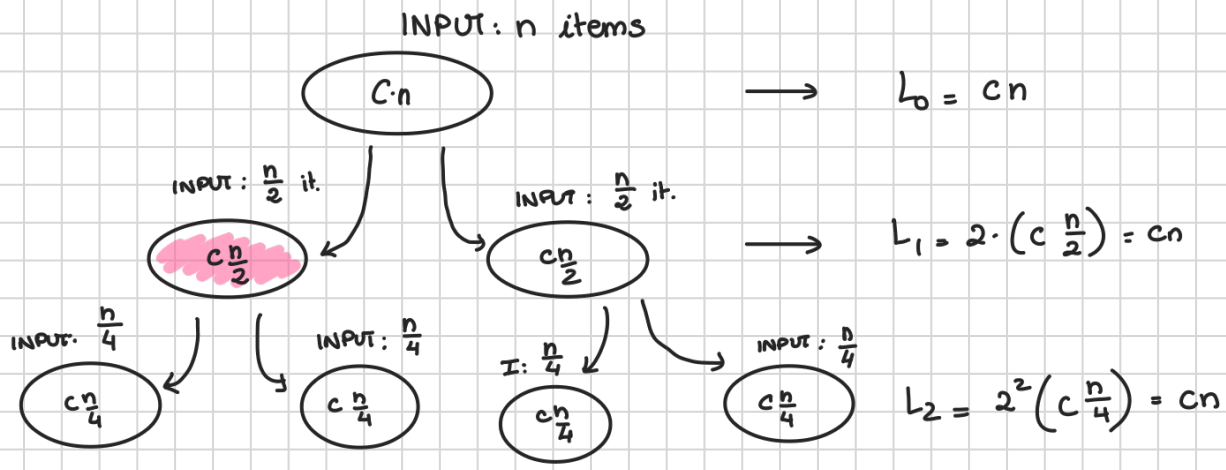
We looked at MERGESORT on $n = 2^t$, for t positive integer, elements.

We proved that the runtime of MERGESORT is $T(n)$, and it satisfies:

$$\exists\, c > 0, \quad T(n) \leq \begin{cases} 2T\left(\frac{n}{2}\right) + c_n & \forall n > 2 \\ T(n) \leq c & n \in \{0, 1, 2\} \end{cases}$$

APPROACH 1

Unroll the recurrence by looking at its first few steps.



In the i^{TH} level, there will be 2^i calls. The generic call of the ith level is going to take time $\frac{cn}{2^i}$ (plus whatever its own calls will cost).

The total runtime for level "i" is going to be

$$2^i \cdot \left(\frac{cn}{2^i}\right) = cn = O(n)$$

Since there are $\log_2 n$ levels, the total runtime is going to be $O(n \log n)$

APPROACH 2

HOW TO BOUND $T(n)$?

① Guess a particular bound ($T(n) \leq n^2$, $T(n) \leq n$).

Suppose we believe (and would like to check) that $T(n) \leq a \cdot n \log_2 n$, for some constant $a > 0$.

R2

$$\exists\, c > 0, \quad T(n) \leq \begin{cases} 2T\left(\frac{n}{2}\right) + c_n & \forall n > 2 \\ T(n) \leq c & n \in \{0, 1, 2\} \end{cases}$$

The case $T(2)$ clearly holds, if $2a \geq c$,

$T(2) \leq c$

$T(n) \leq a n \log_2 n$

We want to claim that $T(2) \leq a\, 2 \log_2 2 = 2a$.
The base case holds.

For $n=2$, then $T(n)$ should be at most c.

Suppose, now, that $n \geq 3$. By induction, we know that $T(m) \leq a \cdot m \log_2 m$. $\forall m \leq n-1$.

We want to prove the inequality for n.

$$T(n) \overset{R_2}{\leq} 2T\left(\frac{n}{2}\right) + c_n$$

$$\text{I.H.} \rightarrow \leq 2\left(a \frac{n}{2} \log_2 \frac{n}{2}\right) + c_n$$

(Induction Hypothesis)

$$= an(\log_2(n) - 1) + c_n$$

$$= an \log_2 n - an + cn$$

$$= an \log_2 n + (c-a)n$$

$$a \geq c \rightarrow \leq an \log_2 n \quad \checkmark$$

L: Thus, if $a \geq c$, we have that $T(n) \leq an \log_2 n$.

C: $T(n) \leq cn \log_2 n$

R2

$$\exists\, c > 0, \quad T(n) \leq \begin{cases} 2T\left(\frac{n}{2}\right) + c_n & \forall n > 2 \\ c & n \in \{0, 1, 2\} \end{cases}$$

$$\begin{array}{r} 12345 + \\ 67193 = \\ \hline 79538 \end{array}$$

$$\begin{array}{r} 100110 + \\ 11011 = \\ \hline 1000001 \end{array}$$

THE RUNTIME OF THIS ALGORITHM IS $O(n)$ IF YOU SUM UP TWO NUMBERS OF n DIGITS EACH

$$\begin{array}{r} {\scriptstyle 1\ 2} \\ 324 \times \\ 156 = \\ \hline 1944 \\ 1620 \\ 324 \\ \hline 50.544 \end{array}$$

RSA encryption algorithm example.

THE RUNTIME OF THIS ALGORITHM IS $O(n^2)$ IF YOU MULTIPLY TWO NUMBER OF n DIGITS EACH.

The divide et impera algorithm that we'll be seeing to improve the time complexity of the multiplication isn't the optimal one, but it still has a better efficiency than the actual one.

It's based on something called fast Fourier transform and will decrease the exponent from $O(n^2)$ to $O(n^{1.5 \text{ or } 1.6})$.

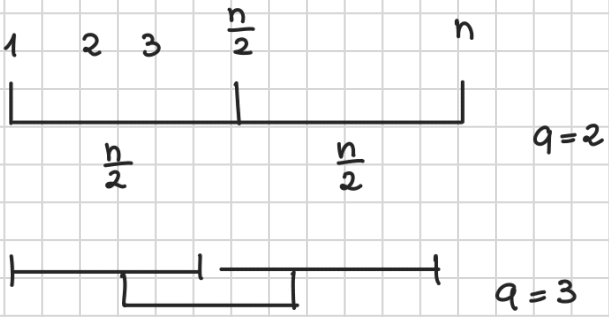
R2

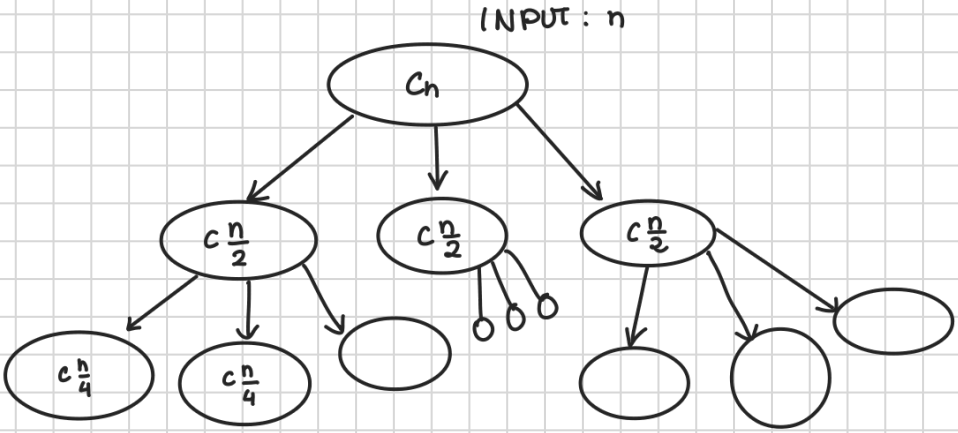
$$\exists\, c>0, \quad T(n) \leq \begin{cases} 2T\left(\frac{n}{2}\right) + c_n & \forall n > 2 \\ c & n \in \{0,1,2\} \end{cases}$$

R2

$$\exists\, c>0, \quad T_q(n) \leq \begin{cases} q \cdot T_q\left(\frac{q}{2}\right) + cn & \forall n > 2 \\ c & \text{OTHERWISE} \end{cases}$$

$q \geq 3$





It appears that in level i we have 3^i many calls and the runtime per call at level i is $c \cdot \frac{n}{2^i}$.

The proper cost at level i is $L_i \leq \frac{q}{i} \cdot c\frac{n}{2^i} = \left(\frac{q}{2}\right)^i \cdot c \cdot n$

The total runtime

$$\sum_{i=0}^{\log n} L_i \leq \sum_{i=0}^{\log n} \left(\left(\frac{q}{2}\right)^i \cdot cn\right)$$

Recall that $\sum_{i=0}^{t} \alpha^i = \alpha^0 + \alpha^1 + \alpha^2 + \ldots + \alpha^t = \frac{\alpha^{t+1} - 1}{\alpha - 1}$

In our case, $\alpha = \frac{q}{2}$ and $t = \log_2 n$ thus,

$$\sum_{i=0}^{\log_2 n} L_i \leq cn \sum_{i=0}^{\log_2 n} \left(\frac{q}{2}\right)^i = cn \frac{\left(\frac{q}{2}\right)^{\log_2 n + 1} - 1}{\frac{q}{2} - 1} =$$

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

$$= \frac{2cn}{q-2} \left(\frac{q}{2}\right)^{\log_2 n} \frac{q}{2}$$

$$(c^x)^y = n^{xy} = (c^y)^x$$

$$= \frac{2cn}{q-2} \left(2^{\log_2 \frac{q}{2}}\right)^{\log_2 n} \frac{q}{2}$$

$$= \frac{2cn}{q-2} \left(2^{\log_2 n}\right)^{\log_2 \frac{q}{2}} \frac{q}{2}$$

$$= \frac{2cn}{q-2} n^{\log_2 \frac{q}{2}} \frac{q}{2}$$

$$= \frac{2c}{q-2} n^{1 + (\log_2(q) - 1)} \frac{q}{2}$$

$$= \frac{qc}{q-2} n^{\log_2 q} = O(n^{\log_2 q})$$

C: The solution to R_q is the $T_q(n) \leq O(n^{\log_2 q})$