

Divide & Conquer

REDUCTION / RECURSION

reducing
input size
by 1

What is recursion if not solving ^{smaller} pieces of a problem until the whole thing is solved? ...

Q Contains ($A[0, \dots, n]$) // Does A contain x?

→ If the array is empty,
return false

else:

if ($A[0] = x$):

return true

else:

return contains ($A[1:n], x$)

let $T(n)$ be runtime
of the function...
 $T(0) = \Theta(1)$

$$\begin{aligned} T(n) &\leq \Theta(1) + T(n-1) \\ &\leq \Theta(1) + \Theta(1) + T(n-2) \\ &\leq \Theta(1) + \Theta(1) + \Theta(1) \\ &\quad + T(n-3) \dots \end{aligned}$$

How do we formally show such an algorithm is correct?

- we can't use an EG

- can we use induction? YES, let's see

Proof by Induction

o Base Case ($n=0$)

It works for an empty list.

o Inductive Step → show it works for

Revisiting the Multiplication Problem

eg

$$\begin{array}{r} 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \\ \times 8 \ 7 \ 6 \ 5 \ 4 \ 3 \ 2 \ 1 \\ \hline \end{array}$$

1	2	3	4	5	6	7	8
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-

$$\begin{array}{r} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \times \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \hline \end{array}$$

Quadratic
 $\rightarrow \underline{\underline{\Theta(n^2)}}$

n rows

$2n$ b/c of the last row (shifted by $n-1$ zeros the right)

$$+ \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---}$$

$\leq 2n$ (upper bound)

[Revisions in pink for the better algorithm]

Break it down instead of doing one big calculation
 BOX ALGORITHM

New Approach

Ex. 2

$$\begin{array}{r} 1 \ 2 \\ \times 3 \ 4 \end{array} = \begin{array}{l} (1 \times 10 + 2) \\ (3 \times 10 + 4) \end{array} \rightarrow \begin{array}{l} 3 \times 10^3 + (3 \times 2 \times 10) + 1 \times 10 + 8 \\ 3 \times 10^2 + (3 \times 2 \times 10) + 8 \\ 3 \times 10^2 + 10 \times 10 + 8 \end{array}$$

2 4 6 8

$$\begin{array}{r} \overbrace{a \quad b}^{n \text{ total digits}} \\ \hline c \quad d \end{array} \begin{array}{l} = a \times 10^{n/2} + b \\ = c \times 10^{n/2} + d \end{array}$$

(i) $a \times 10^n + (ad + bc) \times 10^{n/2} + bd$

even better

(ii) $(a+b)(c+d) = \underline{ac} + \underline{ad + bc} + \underline{bd}$

to substitute this in subtract $ac + bd$ from (i)

does it help?
 [run tests]

Problem: The algorithm takes about n^2 time as well. $n^2/4$ in 4 places

Alg-1

To multiply two n -digit #s x & y

if $n=1$:

just do it

otherwise

split into halves, recurse

and put them back together

like this $\rightarrow a \times c \times 10^n + (bctad)10^{n/2} + bd$

What's the runtime?

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

$$T(n) = \Theta(n) + 4T(n/2) + \Theta(n)$$

$$= 4T(n/2) + \Theta(n)$$

$$= 4(4T(n/4) + \Theta(n/2)) + \Theta(n)$$

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

★ Induction (Before)

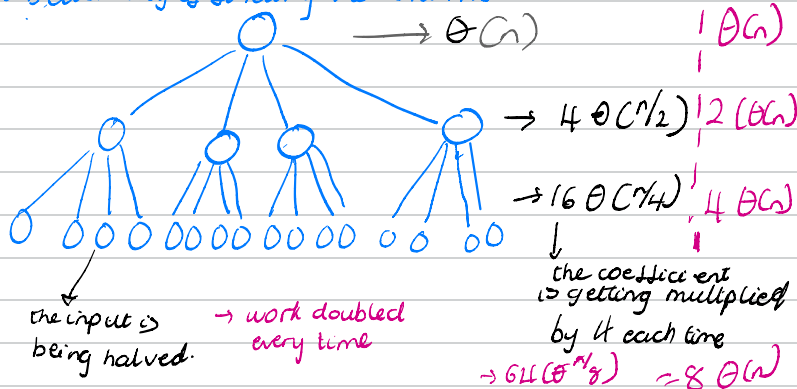
For a particular n , it works for $n+1$.

★ Induction (Now)

If it works for all sizes up to a particular n , it works for $n+1$

Let's find a better way of finding the runtime

linear
in the
size of
the input



$\rightarrow \log(n)$ - inverse operator of exponent

$\rightarrow$ Height of the tree - $\log_2(n)$

$\rightarrow$ sum up all the work in the tree

Total Work

$$\begin{aligned} & \Theta(n) + 2\Theta(n) + 4\Theta(n) + \dots + 2^{\log_2(n)}\Theta(n) \\ &= \sum_{i=0}^{\log_2(n)} 2^i \Theta(n) \\ &= \Theta(n) \sum_{i=0}^{\log_2(n)} 2^i \end{aligned}$$

$$(a10^{n/2} + b) (c10^{n/2} + d)$$

$$a10^n + (bc + ad)10^{n/2} + b$$

$$C(n) = a10^n + (bc + ad)10^{n/2} + b$$

o Karatsuba Algorithm

if $n \leq 1$

just do it.

otherwise:

compute $u = atb$
compute $v = ctd$ } $\Theta(n)$

recursive calls

recurse on ac, bd, uv } $3T(n/2)$

compute $bc + ad = uv - Cactbd$
put it in all together

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

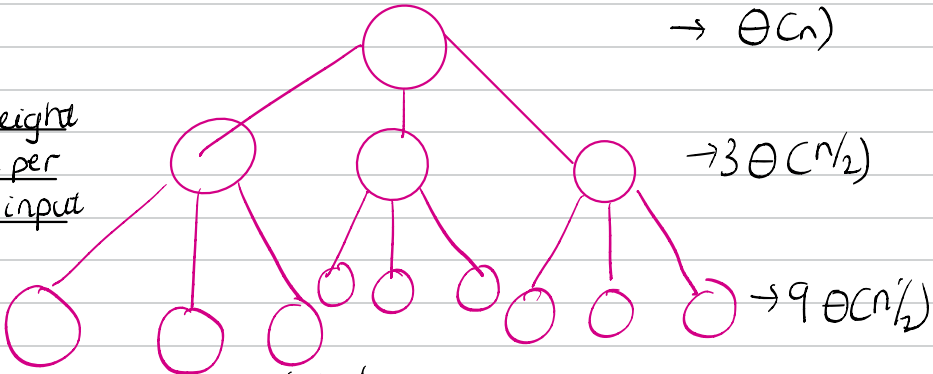
$$T(n) = 3T(n/2) + \Theta(n)$$

$$\Theta(n)$$

$$\Theta(n)$$

$$\rightarrow \Theta(n)$$

- o still $\log_2 n$ height
- o still the work per call is linear in input size



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

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

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

$= n^{1.585}$

□ Log Properties

$$\rightarrow a^{\log_b c} \rightarrow \log_b c = \frac{\log_a c}{\log_a b}$$

$$(a^b)^c = a^{bc}$$

Application to Previous Problem

$$\left(\frac{3}{2}\right)^{\log_2 n} = \frac{3^{\log_2 n}}{2^{\log_2 n}} = \frac{3^{\log_2 n}}{n}$$

$$= 3^{\frac{\log_2 n}{\log_2 2}}$$

$$= \left(3^{\log_2 n}\right)^{1/\log_2 2}$$

$$= n^{1/\log_2 2}$$