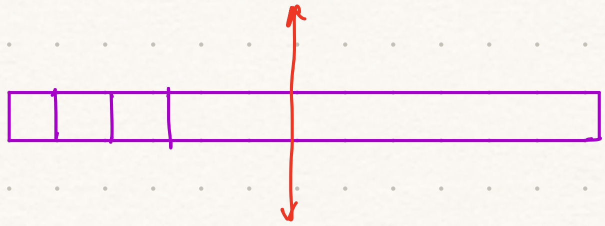
Divide and Conquer

Prev examples: Merge sort.



Sort (A)

Sort ($A_{\leq 1/2}$)   Sort ($A_{>1/2}$)

$$T(n) \leq 2^{\lceil \log_2 n \rceil} \cdot T(1) + c \cdot n \cdot \lceil \log_2 n \rceil$$

Paradigm:

- "Break" your problem into disjoint parts and these form a "smaller" instance of the same problem
- Put together solutions of "smaller" problems

$$T(n) = 2 \cdot T(n/2) + c \cdot n.$$

$$= 2\left[2T\left(\frac{n}{4}\right) + c \cdot \frac{n}{2}\right] + c \cdot n$$

$$= 2^2 T\left(\frac{n}{2^2}\right) + c \cdot [n + n]$$

$$= 2^i \cdot T\left(\frac{n}{2^i}\right) + c \cdot n \cdot i$$

$$i \leq \lceil \log_2 n \rceil$$

Integer Multiplication.

We are given 2 natural numbers a, b.

<u>Want</u>: <u>$a \times b$</u>.

binary repr.

Input size: $\overset{k}{\log_2 a}$, $\overset{l}{\log_2 b}$

Trivial complexity:

a: $2^0 \times 1 + 2^1 \times 1 + 2^2 \times 0 + 2^3 \times 1 + 2^4 \times 0$

b: $2^0 \times 0 + 2^1 \times 1 + 2^2 \times 1 + 2^3 \times 0 + 2^4 \times 0$

MSB   LSB

+1 +1

a:	0	1	0	1	1
b:	0	0	1	1	0
	1	0	0	0	1

$2^0 \times 1 + 2^4 \times 1$

$O(\max\{k, \ell\})$.

$a = \sum_{i=0}^{k-1} a_i \cdot 2^i$ $\quad$ $b = \sum_{j=0}^{\ell-1} b_j \cdot 2^j$

Note that each a_i, b_j are 0, 1 bits.

$$a \times b = \underbrace{\left(\sum_{i=0}^{k-1} a_i \cdot 2^i\right)\left(\sum_{j=0}^{\ell-1} b_j \cdot 2^j\right)}_{k\ell \text{ terms.}}$$

$k, \ell \approx n$

Complexity $\sim n^2$

Polynomial Mult.

$P(z) = \sum_{i=0}^{d} a_i z^i$ | $Q(z) = \sum_{j=0}^{d} b_j \cdot z^j$.

$\hookrightarrow P(z) \cdot Q(z)$.

"Algo for polynomial mult. $\Rightarrow$ Algo for Integer mult. w/ some overhead"

Matrix Mult.

$C = A \cdot B$

$$\underset{n \times n}{\overset{A}{\begin{bmatrix} & & \\ & a_{ij} & \\ & & \end{bmatrix}}} \underset{n \times n}{\overset{B}{\begin{bmatrix} & & \\ & b_{k\ell} & \\ & & \end{bmatrix}}}$$

$$C_{st} = \sum_u a_{su} \cdot b_{ut}$$

$= \langle$ Row s in A, Col t in B $\rangle$

$O(n^3)$ to compute all of C.

($O(n)$ per entry of C).

(worst case/ brute-force).

(Assume that n is a power of 2.)

Karatsuba's Integer Mult.

$a = (a_0, \ldots, a_{n-1})$ (LSB … MSB) $\quad$ $b = (b_0, \ldots, b_{n-1})$ (LSB … MSB)

$a = \sum_{i=0}^{n-1} a_i \cdot 2^i$ $\quad$ $b = \sum_{j=0}^{n-1} b_j \cdot 2^j$

$$= \underbrace{\left(\sum_{i=0}^{n/2-1} a_i \cdot 2^i\right)}_{A_0} + 2^{n/2}\underbrace{\left(\sum_{i'=0}^{\frac{n}{2}-1} a_{i'+\frac{n}{2}} \cdot 2^{i'}\right)}_{A_1} \qquad = \underbrace{\sum_{j=0}^{\frac{n}{2}-1} b_j \cdot 2^j}_{B_0} + 2^{n/2}\underbrace{\left(\sum_{j'=0}^{\frac{n}{2}-1} b_{j'+\frac{n}{2}} \cdot 2^{j'}\right)}_{B_1}$$

$$= A_0 + (A_1 \cdot 2^{n/2}) \qquad = B_0 + (B_1 \cdot 2^{n/2})$$

$$a \cdot b = (A_0 + A_1 \cdot 2^{n/2})(B_0 + B_1 \cdot 2^{n/2}) = \overbrace{A_0 B_0}^{C_1} + \overbrace{(A_1 B_0 + A_0 B_1)}^{C_2} \cdot 2^{n/2} + \underbrace{A_1 \cdot B_1}_{C_3} \cdot 2^n$$

Obs: A_0, A_1, B_0, B_1 are integers of bit complexity $\le \frac{n}{2}$.

$$T(n) = 4 \cdot T\left(\frac{n}{2}\right) + O(n) = n^{\log_2 4} \simeq n^2$$

Can we cut down on the no. of $\frac{n}{2}$-sized mult.

$$\underbrace{(A_0 + A_1)}\,\underbrace{(B_0 + B_1)} \qquad \underbrace{A_0 B_0}_{C_1} \qquad \underbrace{A_1 B_1}_{C_3}.$$

$$= \underbrace{A_0 B_0 + A_1 B_1 + (A_0 B_1 + A_1 B_0)}_{C^*}$$

$$\hookrightarrow A_0 B_1 + A_1 B_0 = C^* - C_1 - C_3$$

$\frac{n}{2}+1 \leftarrow$ $A_0 + A_1$, $B_0 + B_1$ could have $\frac{n}{2}+1$ bits.

$$T(n) = 3 \cdot T\left(\frac{n}{2}+1\right) + O(n). \qquad \sim n^{\log_2 3 + \sim}$$

Increasing no. of additions/sub to optimize # of mult.

mult of $\frac{n}{2}$ bit nos:

$A_0 \cdot B_0 \leftarrow C_1 \leftarrow$ ① mult

$A_1 \cdot B_1 \leftarrow C_3 \leftarrow$ ② mult

$A_0 + A_1 \leftarrow \tilde{A} \leftarrow$ ① add

$B_0 + B_1 \leftarrow \tilde{B} \leftarrow$ ② add

$(\tilde{A} \cdot \tilde{B}) \leftarrow \tilde{A} \cdot \tilde{B} \leftarrow$ ③ mult

mult. of $\le \frac{n}{2}+1$ bit numbers.

$$C_2 = \underbrace{\tilde{A} \cdot \check{B} - C_1 - C_3}$$

③ sub/add

$$C_1, \; C_2, \; C_3 \longrightarrow \left.\underbrace{C_1 + \underbrace{C_2 \cdot 2^{n/2}}_{} + \underbrace{C_3 \cdot 2^{n}}_{}}_{\text{Add}}\right\} O(n)$$

Bit shifts

Nearest power of 2 is at most

$$n_0 \longrightarrow 2n_0$$

$$\left.\begin{aligned} a &= \tilde{A}_0 + \tilde{A}_1 \cdot 2^{n/3} + \tilde{A}_2 \cdot 2^{2n/3} \\ b &= \tilde{B}_0 + \tilde{B}_1 \cdot 2^{n/3} + \tilde{B}_2 \cdot 2^{2n/3} \end{aligned}\right\}$$

$\tilde{A}_i, \tilde{B}_j$ are $\frac{n}{3}$ bit no.s.

Brute force $\rightarrow n^{\log_3 9}$

$$T(n) = 9 \cdot T\left(\frac{n}{3}\right) + O(n).$$

Brief look into Strassen's matrix mult:

$$\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}_{n\times n} \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}_{n\times n} = \begin{bmatrix} A_1B_1 + A_2B_3 & A_1B_2 + A_2B_4 \\ A_3B_1 + A_4B_3 & A_3B_2 + A_4B_4 \end{bmatrix}$$

$A_i, B_j \leftarrow$ are of size $\frac{n}{2} \times \frac{n}{2}$.

$$T(n) = 8 \cdot T\left(\frac{n}{2}\right) + O(n^2) \sim O\left(n^{\log_2 8}\right) \underset{\sim}{= cn^3}$$

Strassen gave a procedure with 7 matrix mult of $\frac{n}{2} \times \frac{n}{2}$ mat. at the expense of matrix additions. $\longrightarrow n^{\log_2 7} \sim n^{2.81}$

$\mapsto$ "Laser method" + "tensor alg" w/ Coppersmith-Winograd tensor.

$\hookrightarrow \omega = 2.3 \; - - -$

Alhussein Fawzi, Deepmind } + Quanta article.