CSCE 750
8/24/2023

https://cse.sc.edu/~fenner/csce750 (my page)

Analysis of Algorithms

Multiplying two n-bit numbers

$$A = a_{n-1} a_{n-2} \cdots a_1 a_0 \qquad a_i \in \{0,1\}$$
$$= \sum_{i=0}^{n-1} a_i 2^i$$

$$B = b_{n-1} \cdots b_0 = \sum_{i=0}^{n-1} b_i 2^i$$

Suppose $n = 2^k$ (some k)

$$AB = \left(\sum_{i=0}^{n-1} a_i 2^i\right)\left(\sum_{j=0}^{n-1} b_j 2^j\right) = \sum_{i,j} a_i b_j 2^{i+j}$$

$$= \sum_{\ell=0}^{2n-2} 2^\ell \left(\sum_{i+j=\ell} a_i b_j\right) = \sum_{\ell} 2^\ell \left(\sum_{i=0}^{\ell} a_i b_{\ell-i}\right)$$

individual bit mults: $\sum_{\ell=0}^{2n-2} (\ell+1) =$ order of n^2

Divide & conquer method ($n = 2^k$)

$A_{low} = a_{\frac{n}{2}-1} \cdots a_0$ $\quad B_{low} = b_{\frac{n}{2}-1} \cdots b_0$

$A_{high} = a_{n-1} \cdots a_{n/2}$ $\quad B_{high} = b_{n-1} \cdots b_{\frac{n}{2}}$

$$A = 2^{\frac{n}{2}} A_{high} + A_{low}$$

$$B = 2^{n/2} B_{high} + B_{low}$$

$$AB = \left(2^{\frac{n}{2}} A_{high} + A_{low}\right)\left(2^{n/2} B_{high} + B_{low}\right)$$

$$= 2^n \overbrace{A_{high} B_{high}}^{A_h B_h} + 2^{n/2}\left(\overbrace{A_{high} B_{low}}^{A_h B_\ell} + \overbrace{A_{low} B_{high}}^{A_\ell B_h}\right) + \overbrace{A_{low} B_{low}}^{A_\ell B_\ell}$$

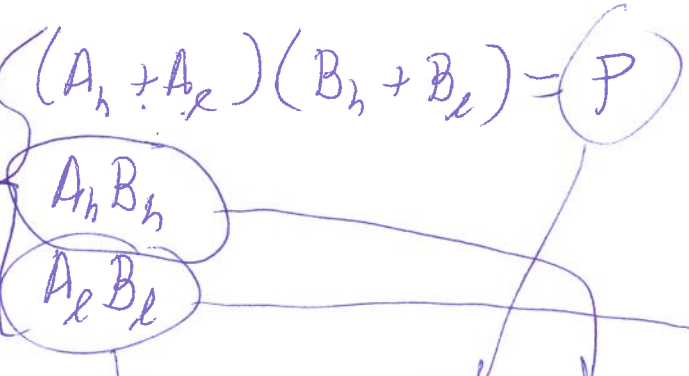
Recursively compute $A_{high} B_{high}, \ldots, A_{low} B_{low}$

4 recursive calls on numbers half the size

h = high
ℓ = low

$$(A_h + A_\ell)(B_h + B_\ell) = A_h B_h + A_h B_\ell + A_\ell B_h + A_\ell B_\ell = P$$

Recursively multiply: $(A_h + A_\ell)(B_h + B_\ell) = P$



$$AB = 2^n A_h B_h + A_\ell B_\ell + 2^{n/2}\left(P - A_h B_h - A_\ell B_\ell\right)$$

Runs in time order of $n^{\log_2 3}$

Asymptotic notation

$\mathbb{N} := \{0, 1, 2, \ldots\}$

Functions $f, g : \mathbb{N} \to \mathbb{R}$

We say that f is eventually positive (nonnegative) if

$$\exists n_0 \; \forall n \geq n_0, \; f(n) > 0 \quad (\geq)$$

Suppose f, g are eventually positive. (from now on)

Say that $f \in \Theta(g)$ if

$$\exists C \in \mathbb{R}, \exists n_0 \; \forall n \geq n_0,$$

$$f(n) \leq C g(n)$$

Actually, $\Theta(g) = \{f : f$ is eventually positive & $\exists C \; \exists n_0 \; \forall n \geq n_0, \; f(n) \leq C g(n)\}$

We say $f(n) = O(g(n))$ to mean $f \in \Theta(g)$

Small inputs — don't care because any algo will be good enough

constant factors — don't care because actual run times are implementation-dependent & don't say anything about the algo itself.

Examples: $f(n) = 5n^2 + 3n - 6 = O(n^2)$

Proof: $f(n) = 5n^2 + 3n - 6 \leq 5n^2 + 3n^2$

$= 8n^2 \leq cn^2$ (any $c \geq 8$, $n_0 \geq 0$)

$f(n) = 3n - 6 \stackrel{?}{=} O(n^2)$

true because $f(n) \leq 3n^2$ $(\forall n)$

Def: $f \in \Omega(g)$ $\left[f = \Omega(g) \right]$

iff $\exists c > 0, \exists n_0, \forall n \geq n_0, f(n) \geq cg(n)$

$(c \in \mathbb{R})$

$f(n) = 5n^2 + 3n - 6 \stackrel{?}{=} \Omega(n^2)$

Proof: $f(n) = 5n^2 + 3n - 6 \geq 5n^2 - 6$

$= 4n^2 + \underbrace{(n^2 - 6)}_{\geq 0 \text{ if } n^2 \geq 6, \text{ True if } n \geq 3} \geq \underbrace{4n^2}_{c}$ for all $n \geq \underbrace{3}_{n_0}$

Can neglect lower degree terms in a polynomial of n for O and Ω

& can neglect the leading coefficient for O & Ω.

$f(n) = 5n - 3 \stackrel{?}{=} \Omega(n^2)$ no

Def. $f \in \Theta(g)$ means $f \in O(g)$ & $f \in \Omega(g)$.

"f is asymptotically similar to g"

equivalence relation

Fact: $f \in \Theta(g) \iff g \in \Omega(f)$

~~**Def:** $f \in o(g)$ ("little o")~~

~~means~~