Lecture 1 Karatsuba's algorithm

Sect 4.3

January 12, 2021 3:44 PM

$98|76 < 10^4$
$a_1 \quad a_2$

Assume $0 < a, b < B^n$ and $n = 2^k$ for simplicity.

Let $a = a_1 B^{n/2} + a_2$ $b = b_1 \cdot B^{n/2} + b_2$ where $0 \le a_1, a_2, b_1, b_2 < B^{n/2}$

$$a \times b = (a_1 B^{n/2} + a_2) \cdot (b_1 B^{n/2} + b_2)$$

$$= a_1 \cdot b_1 \cdot B^n + (a_1 \cdot b_2 + a_2 \cdot b_1) B^{n/2} + a_2 \cdot b_2$$

4 mults. of integers of $\frac{1}{2}$ as long and 3 +. and 2 shifts.

Algorithm is "do this recursively until $n=1$, $a=0$, $b=0$."

a $87|00$
$a_1 \quad a_2$

Example: $a = 87|65 = 87 \cdot 10^2 + 65$

$b = 65|43 = 65 \cdot 10^2 + 43$

$$a \times b = 87 \cdot 65 \cdot 10^4 + (87 \cdot 43 + 65 \cdot 65) \cdot 10^2 + 65 \cdot 43.$$

$$8|7 \cdot 6|5 = 8 \cdot 6 \cdot 10^2 + (8 \cdot 5 + 7 \cdot 6) \cdot 10 + 7 \cdot 5$$

$$= 4800 + (40 + 42) \cdot 10 + 35$$

$$= 4800 + 820 + 35 = \cdots$$

Let $T(n)$ be the cost of multiplying $a \times b$ of length $n = 2^k$ digits.

4 mults — half size — work for 3+ and 2 shifts.

$$T(n) \le 4T(n/2) + c \cdot n \quad \text{for } n > 1 \text{ and } T(1) = d.$$

1° $T(n) \le 4 \cdot \cancel{T(\tfrac{n}{2})} + cn$

$4^1 \cancel{T(n/2)} \le 4(4 \cdot T(\tfrac{n}{4}) + c \cdot n/2) = 4^2 \cdot \cancel{T(n/4)} + 2cn$

$4^2 \cancel{T(n/4)} \le 4^2(4 \cdot T(\tfrac{n}{8}) + c \cdot n/4) = \cancel{4^3 T(n/8)} + 4cn$

$\vdots$

$(2^{k-1}) \cdot 2 \cdot 2^{k-1} \cdot c$

$\cancel{4^{k-1} T(2)} \le 4^{k-1}(4T(1) + c \cdot 2) = \cancel{4^k T(1)} + 4^{k-1} \cdot 2 c = 2^{k-1} \cdot n \cdot c$

$\cancel{4^k T(1)} = 4^k d$

$+$

$$T(n) \le cn + 2cn + 4cn + \dots + 2^{k-1} cn. + 4^k d$$

$$= cn \left[\underbrace{1 + 2 + 4 + \dots + 2^{k-1}}\right] + (2^k)^2 d$$

$$= cn(2^k - 1) = cn^2 - cn. + n^2 d$$

$$= cn(2^k - 1) = cn^2 - cn + n^2 d$$
$$2^k = n$$
$$= (c+d)n^2 - cn$$
$$\in O(n^2).$$

This is not Karatsuba's alg. It's no better than before.

Karatsuba's idea.

Consider $a \times b = a_1 b_1 \cdot B^n + (a_1 b_2 + a_2 b_1) \cdot B^{n/2} + a_2 b_2$

$$= a_1 \cdot b_1 B^n + [(a_1 - a_2)(b_2 - b_1) + a_1 b_1 + a_2 b_2] \cdot B^{n/2} + a_2 b_2$$

There are 3 distinct multiplications of length $n/2$ plus some + and - and shifts which cost linear in n.

Now

$$T(n) \le 3T(n/2) + cn \text{ for } n > 1 \text{ and } T(1) = d.$$

Excercise: Show that $T(n) = (2c+d)n^{\log_2 3 = 1.585} - 2cn \in O(n^{1.585})$

Show that $\frac{T(2n)}{T(n)} \sim 3.$

So doubling the length of a and b tripples the time.

Example:

$a = 27|65$ (a_1, a_2)

$b = 43|65$ (b_1, b_2)

$$a \times b = 27 \cdot 43 \cdot 10^4 + \left[\overbrace{(27-65) \cdot (65-43)}^{-38 \cdot 22} + 27 \cdot 43 + 65 \cdot 65 \right] \cdot 10^2 + 65 \cdot 65$$

$$-(3|8 \cdot 2|2) =$$

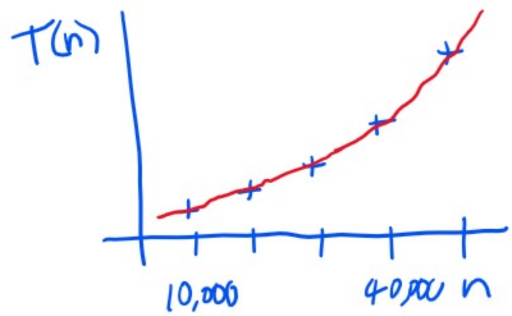
Suppose the cost of an algorithm is $T(n)$ and theoretically

$T(n) = c_1 \cdot n^2 + c_2 \cdot n + c_3 \in O(n^2)$.

How can we experimentally verify this?

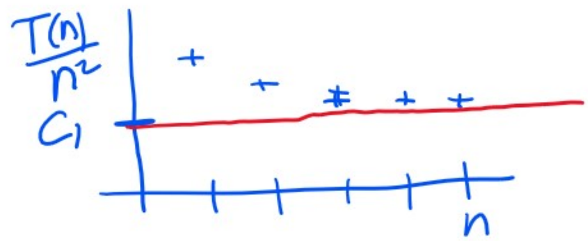
Time it for large $n = 10{,}000,\ 20{,}000,\ 30{,}000,\ 40{,}000,\ 50{,}000.$

Time it for large $n = 10{,}000, 20{,}000, 30{,}000, 40{,}000, 50{,}000.$



$$\frac{T(n)}{n^2} = \frac{c_1 n^2 + c_2 n + c_3}{n^2} = c_1 + c_2/n + c_3/n^2$$

$$\lim_{n\to\infty} T(n)/n^2 = c_1$$



Best way: $$\lim_{n\to\infty} \frac{T(2n)}{T(n)} = \frac{c_1(2n)^2 + c_2(2n) + c_3}{c_1 n^2 + c_2 n + c_3} = \underline{4}$$