

# CS 314 : Divide + Conquer

Note Title

2/5/2010

## Announcements

- HW posted - due (written) next Friday at start of class

## Multiplying two numbers

Let's say we have an n-digit number

(Aside - how big can that number be?)

$$\leq 10^n$$

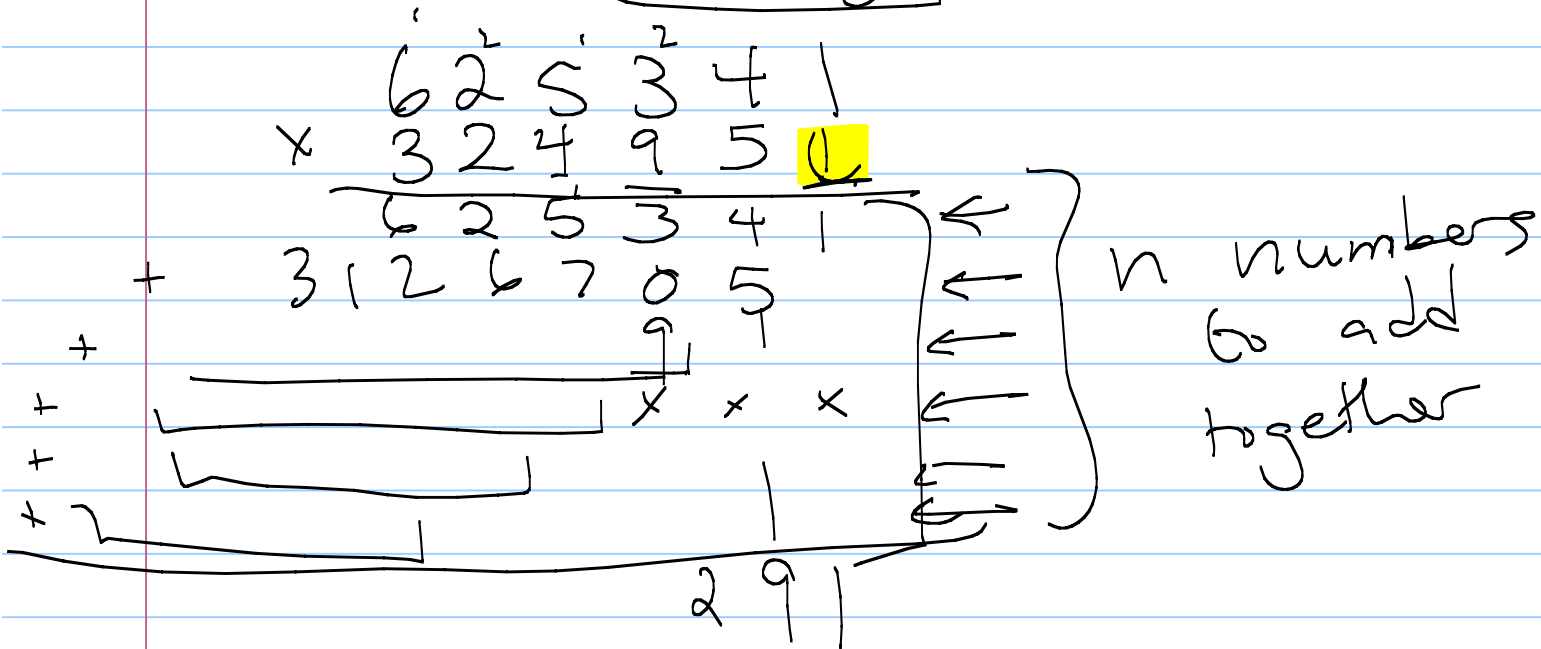
$$\begin{array}{r} \phantom{x} \overset{1}{6} \overset{2}{3} \overset{3}{5} 2 \overset{4}{1} 9 \leftarrow \\ x \phantom{0000} 4 \leftarrow \\ \hline 2540876 \end{array}$$

if I give you a number  $\leq n$ , how many bits does it take to represent?

$$\log_2 n$$

$O(n)$  <sup>1 digit</sup> multiplications  
 $O(n)$  additions

What about 2 n-digit numbers?



$O(n^2)$  1-digit

How would we code that algorithm?

Two nested for loops

$$O(n^2)$$

More efficient: Recursion!

$\underbrace{\text{XXXXXX}}_{10^4}$

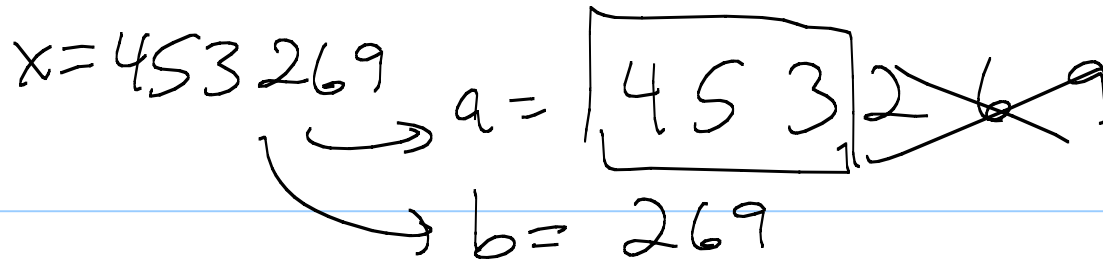
$$\begin{aligned} & \overbrace{(10^m a + b)(10^m c + d)} \\ &= 10^{2m} \underbrace{ac} + 10^m \underbrace{(bc + ad)} + \underbrace{bd} \end{aligned}$$

Ex:  $4563 \times 2729$   
 $= (10^2 \cdot \underbrace{45} + \underbrace{63}) (10^2 \cdot \underbrace{27} + \underbrace{29})$   
                   $\parallel$            $\parallel$                    $\parallel$            $\parallel$   
                  a          b                  c          d

Instead of calculating 1 big multiplication,  
compute  $\underline{a \cdot c}$ ,  $\underline{bc}$ ,  $\underline{ad}$ , &  $\underline{b \cdot d}$

Is this better? ←

Pseudocode



```
MULTIPLY(x, y, n):  
  if n = 1  
    return x · y  
  else  
    m ← ⌊n/2⌋  
    a ← ⌊x/10m⌋; b ← x mod 10m  
    d ← ⌊y/10m⌋; c ← y mod 10m  
    e ← MULTIPLY(a, c, m)  
    f ← MULTIPLY(b, d, m)  
    g ← MULTIPLY(b, c, m)  
    h ← MULTIPLY(a, d, m)  
    return 102me + 10m(g + h) + f
```

Running time: to add #5

$$R(n) = 4R\left(\frac{n}{2}\right) + O(n)$$
$$R(1) = 1$$

Exercise: use Master Thm

$$R(n) = O(n^2)$$

A trick: (Karatsuba 1962)

bc + ad can be computed using only one multiplication!

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

we already  
compute these!

$$\begin{aligned} &= ac + bd - [ac - bc - ad + bd] \\ &= bc + ad \end{aligned}$$

New + improved pseudo code:

FASTMULTIPLY(x, y, n):

if  $n = 1$

return  $x \cdot y$

else

$m \leftarrow \lceil n/2 \rceil$

$a \leftarrow \lfloor x/10^m \rfloor$ ;  $b \leftarrow x \bmod 10^m$

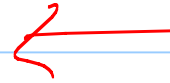
$d \leftarrow \lfloor y/10^m \rfloor$ ;  $c \leftarrow y \bmod 10^m$

$e \leftarrow \text{FASTMULTIPLY}(a, c, m)$

$f \leftarrow \text{FASTMULTIPLY}(b, d, m)$

$g \leftarrow \text{FASTMULTIPLY}(a - b, c - d, m)$

return  $10^{2m}e + 10^m(e + f - g) + f$





What's this running time?

$$T(n) = 3 T\left(\frac{n}{2}\right) + \underbrace{O(n)}$$

$$T(1) = O(1)$$

↑  
additions & subtractions

$$T(n) = O\left(n^{\log_2 3}\right) = O\left(n^{\uparrow 1.58\dots}\right)$$

# Exponentiation

How do we compute  $a^n$ ?

$$a^n = \underbrace{a \cdot a \cdot a \cdot a \cdots a}_n$$

$O(n)$  multiplications  
↑

Naive algorithm:

SLOWPOWER(a, n):

$x \leftarrow a$

for  $i \leftarrow 2$  to  $n$

$x \leftarrow x \cdot a$

return  $x$

Running time? (# of multiplications)

$O(n)$   
↑

Faster idea:

$$a^n = a^{\lfloor n/2 \rfloor} \cdot a^{\lceil n/2 \rceil}$$

$$a^6 = a^3 \cdot a^3$$

$$a^{1001} = a^{500} \cdot a^{501}$$

Pseudocode:

```
FASTPOWER(a, n):  
  if n = 1  
    return a  
  else  
    x ← FASTPOWER(a, [n/2])  
    if n is even  
      return x · x  
    else  
      return x · x · a
```

FastMultiply(x, x)

Running time?

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

$$T(1) = 1$$

$$\Rightarrow T(n) = O(\log_2 n)$$

$$= 2 \log_2 n$$