### Divide and Conquer — Multiplying Large Integers

School Method

X

 $\begin{array}{c} 981 \\ \underline{1234} \end{array}$ 

3 9 2 4

2 9 4 3

1962

981

1 2 1 0 5 5 4

This takes  $O(n^2)$  time to multiply two n digit numbers.

Exercise: Time to multiply an n digit number by an m digit number is O(nm).

# Assignment Project Exam Help

Divide and Conquer

https://powcoder.com

Easiest when both numbers had sand earlier provise der 981 to 0981.

$$09|81 \times 12|34$$
 $00 \times 12 \times 108_{-1}$ 
 $01 \times 12 \times 108_{-1}$ 
 $01 \times 12 \times 12 \times 108_{-1}$ 
 $01 \times 12 \times 12 \times 108_{-1}$ 
 $01 \times 12 \times 12 \times 108_{-1}$ 

$$recorse!$$
 $e.g., 0/9 \times 1/2$ 
 $shift$ 
 $0 \times 1 \times 2 \times 0$ 
 $0 \times 2 \times 1 \times 0$ 
 $9 \times 1 \times 1 \times 1 \times 0$ 
 $9 \times 1 \times 1 \times 1 \times 0$ 
 $9 \times 1 \times 1 \times 1 \times 0$ 
 $9 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 1 \times 1 \times 0$ 
 $1 \times 1 \times 0$ 
 $1 \times 1 \times 0$ 
 $1 \times 1 \times 0$ 
 $1 \times 1 \times 0$ 
 $1 \times 1 \times 1$ 

$$T(n) = 4T(n/2) + O(n)$$

Apply Master Method: 
$$T(n) = aT(\frac{n}{k}) + cn^k$$

$$a = 4 \quad b = 2 \quad k = 1 \quad \text{compare } a \quad \text{to } b^k$$

$$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^2)$$

$$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^2)$$
No gain so far!

Idea: avoid one of the four multiplications:

https:// $\overline{p}$ owcoder.com $^{10^4}wy + 10^2(wz + xy) + xz$ 

We don't need wz and xy. We just need wz + xy.

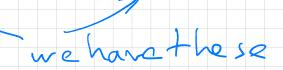
Consider

## Add WeChat powcoder

$$(w + x) \times (y + z) = wy + (wz + xy) + xz$$

Algorithm:

$$p = wy$$
 $q = xz$ 
 $r = (w + x) \times (y + z)$ 
 $\mathbf{return} \ 10^4 p + 10^2 (r - p - q) + q$ 



Now we get:

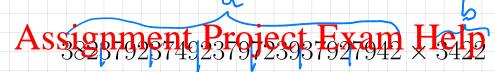
$$T(n) = 3T(n/2) + O(n)$$
  $a = 3$   $b = 2$   $k = 1$   $a = 3 > b^k = 2$ 

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3})$$
 Note:  $\log_2 3 \approx 1.585$ 

This algorithm is was discovered by Karatsuba in 1960.

Practical issues:

1. What about number of different length? E.g., a with n digits, b with m digits,  $m \gg m$ 



- (a) Break a into O(n/m) thurs:  $\phi$  positive dechem
- (b) Multiply each chunk by b.
- (c) Add up all products, taking in the chart provided

Cost:  $O((n/m)m^{\log_2 3})$ , or  $O(nm^{0.585})$ 

2. Which base to use? In practice: base 2<sup>64</sup>

Number is stored as an array of 64-bit integers (unsigned long):

$$a = a_0 + a_1 2^{64} + a_2 (2^{64})^2 + \dots + a_{n-1} (2^{64})^{n-1} \longrightarrow A = a_0 \mid a_1 \mid \dots \mid a_{n-1} \mid a_{n-$$

3. Asymptotically faster methods for larger n.

Schönhage & Strassen (1971):  $O(n(\log n)(\log\log n))$  (used in practice)

recent breakthrough (2019): O(nlown

### Multiplying Matrices

Problem: multiply two  $n \times n$  matrices (count operations  $\{+, -, \times\}$  from domain of entries) Standard method is  $O(n^3)$ 

Divide and Conquer: divide into submatrice of size n/2

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\frac{\textbf{Assignment Project Example Loss}_{A_{21}B_{11} + A_{22}B_{21} | A_{21}B_{12} + A_{22}B_{22} }}{A_{21}B_{11} + A_{22}B_{21} | A_{21}B_{12} + A_{22}B_{22} } ]$$

$$T(n) = 8T(n/2) + O(n^2)$$
 https://powcoder.com >  $b^k = 4$ 

$$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^3)$$
 Add WeChat powcoder

So far, no progress.

Strassen's algorithm: (1969)

- like idea for integer multiplication
- get by with 7 subproblems instead of 8 (tricky!)

$$T(n) = 7T(n/2) + O(n^2)$$
  $a = 7$   $b = 2$   $k = 2$   $a = 7 > b^k = 4$ 

$$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 7})$$
 Note:  $\log_2 7 \approx 2.808$ 

Again, there are asymptotically faster methods, but they are not considered to be practical.

#### The Centrality of Matrix Multiplication

Suppose two  $n \times n$  matrices can be multiplied using  $O(n^{\omega})$ :  $2 \le \omega \le 3$ .

Many problems can be solved in time  $O(n^{\omega})$ :

solving Ax = b computing  $\det A$  computing  $A^{-1}$ .

Many problems are at least as difficult as matrix multiplication.

Example: Reduction of triangular matrix inversion to matrix multiplication. Compute the inverse Assignment Project Exam Help

Divide and Conquer: decompose 
$$T_{\text{into blocks of size }n/2}$$
.

$$T = \begin{bmatrix} T_1 & U \\ A \text{dd} & WeChat powcoder} \end{bmatrix}$$

$$T = \begin{bmatrix} T_1 & U \\ A \text{dd} & WeChat powcoder} \end{bmatrix}$$

$$T(n) = 2T(n/2) + O(n^{\omega}) \quad a = 2 \quad b = 2 \quad k = \omega \quad a = 2 < b^k = 2^{\omega} \ge 4$$

$$T(n) \in \Theta(n^{\omega})$$

Example: Reduction of matrix multiplication to triangular matrix inversion.

$$\begin{bmatrix} I_n & A & & \\ & I_n & B \\ & & I_n \end{bmatrix}^{-1} = \begin{bmatrix} I_n & -A & AB \\ & I_n & -B \\ & & I_n \end{bmatrix}$$