CMPS 3120

Algorithm Analysis

Dr. Chengwei Lei
CEECS
California State University, Bakersfield
Multiplication of Large Integers

Consider the problem of multiplying two (large) \( n \)-digit integers represented by arrays of their digits such as:

\[
A = 12345678901357986429 \quad B = 87654321284820912836
\]

The grade-school algorithm:

\[
\begin{array}{cccc}
& a_1 & a_2 & \ldots & a_n \\
\times & b_1 & b_2 & \ldots & b_n \\
\hline
& (d_{10}) & (d_{11}) & d_{12} & \ldots & d_{1n} \\
& (d_{20}) & d_{21} & d_{22} & \ldots & d_{2n} \\
& \vdots & \vdots & \vdots & \ddots & \vdots \\
& (d_{n0}) & d_{n1} & d_{n2} & \ldots & d_{nn}
\end{array}
\]

Efficiency: \( n^2 \) one-digit multiplications
First Divide-and-Conquer Algorithm

A small example: $A \ast B$ where $A = 2135$ and $B = 4014$

$A = (21 \cdot 10^2 + 35), \quad B = (40 \cdot 10^2 + 14)$

So, $A \ast B = (21 \cdot 10^2 + 35) \ast (40 \cdot 10^2 + 14)$

$= 21 \ast 40 \cdot 10^4 + (21 \ast 14 + 35 \ast 40) \cdot 10^2 + 35 \ast 14$

In general, if $A = A_1 A_2$ and $B = B_1 B_2$ (where $A$ and $B$ are $n$-digit, $A_1$, $A_2$, $B_1$, $B_2$ are $n/2$-digit numbers),

$A \ast B = A_1 \ast B_1 \cdot 10^n + (A_1 \ast B_2 + A_2 \ast B_1) \cdot 10^{n/2} + A_2 \ast B_2$

Recurrence for the number of one-digit multiplications $M(n)$:

$M(n) = 4M(n/2), \quad M(1) = 1$

Solution: $M(n) = n^2$
Second Divide-and-Conquer Algorithm

\[ A \ast B = A_1 \ast B_1 \cdot 10^n + (A_1 \ast B_2 + A_2 \ast B_1) \cdot 10^{n/2} + A_2 \ast B_2 \]

The idea is to decrease the number of multiplications from 4 to 3:

\[ (A_1 + A_2) \ast (B_1 + B_2) = A_1 \ast B_1 + (A_1 \ast B_2 + A_2 \ast B_1) + A_2 \ast B_2, \]

i.e., \((A_1 \ast B_2 + A_2 \ast B_1) = (A_1 + A_2) \ast (B_1 + B_2) - A_1 \ast B_1 - A_2 \ast B_2,
\]

which requires only 3 multiplications at the expense of \((4-1)\) extra add/sub.

Recurrence for the number of multiplications \(M(n)\):

\[ M(n) = 3M(n/2), \quad M(1) = 1 \]

Solution: \(M(n) = 3 \log_2 n = n \log_2 3 \approx n^{1.585} \)
General Divide-and-Conquer Recurrence

\[ T(n) = aT(n/b) + f(n) \quad \text{where} \quad f(n) \in \Theta(n^d), \quad d \geq 0 \]

**Master Theorem:**

- If \( a < b^d \), \( T(n) \in \Theta(n^d) \)
- If \( a = b^d \), \( T(n) \in \Theta(n^d \log n) \)
- If \( a > b^d \), \( T(n) \in \Theta(n^{\log_b a}) \)

*Note: The same results hold with \( \Omega \) instead of \( \Theta \).*
Example of Large-Integer Multiplication

2135 \times 4014
Strassen’s Matrix Multiplication

Strassen observed [1969] that the product of two matrices can be computed as follows:

\[
\begin{bmatrix}
C_{00} & C_{01} \\
C_{10} & C_{11}
\end{bmatrix}
= \begin{bmatrix}
A_{00} & A_{01} \\
A_{10} & A_{11}
\end{bmatrix}
\times
\begin{bmatrix}
B_{00} & B_{01} \\
B_{10} & B_{11}
\end{bmatrix}
= \begin{bmatrix}
M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\
M_2 + M_4 & M_1 + M_3 - M_2 + M_6
\end{bmatrix}
\]
Formulas for Strassen’s Algorithm

\[ M_1 = (A_{00} + A_{11}) \times (B_{00} + B_{11}) \]
\[ M_2 = (A_{10} + A_{11}) \times B_{00} \]
\[ M_3 = A_{00} \times (B_{01} - B_{11}) \]
\[ M_4 = A_{11} \times (B_{10} - B_{00}) \]
\[ M_5 = (A_{00} + A_{01}) \times B_{11} \]
\[ M_6 = (A_{10} - A_{00}) \times (B_{00} + B_{01}) \]
\[ M_7 = (A_{01} - A_{11}) \times (B_{10} + B_{11}) \]
Analysis of Strassen’s Algorithm

If \( n \) is not a power of 2, matrices can be padded with zeros.

Number of multiplications:

\[
M(n) = 7M(n/2), \quad M(1) = 1
\]

Solution: \( M(n) = 7^\log_2 n = n^{\log_7 7} \approx n^{2.807} \) vs. \( n^3 \) of brute-force alg.

Algorithms with better asymptotic efficiency are known but they are even more complex.
General Divide-and-Conquer Recurrence

\[ T(n) = aT(n/b) + f(n) \quad \text{where } f(n) \in \Theta(n^d), \quad d \geq 0 \]

**Master Theorem:**
- If \( a < b^d \), \( T(n) \in \Theta(n^d) \)
- If \( a = b^d \), \( T(n) \in \Theta(n^d \log n) \)
- If \( a > b^d \), \( T(n) \in \Theta(n^{\log_b a}) \)

**Note:** The same results hold with \( O \) instead of \( \Theta \).
Closest-Pair Problem by Divide-and-Conquer
closest-pair problem by divide-and-conquer

step 1: divide the points given into two subsets $P_l$ and $P_r$ by a vertical line $x = m$ so that half the points lie to the left or on the line and half the points lie to the right or on the line.
Closest Pair by Divide-and-Conquer (cont.)

Step 2  Find recursively the closest pairs for the left and right subsets.

Step 3  Set \( d = \min\{d_l, d_r\} \)

We can limit our attention to the points in the symmetric vertical strip \( S \) of width \( 2d \) as possible closest pair. (The points are stored and processed in increasing order of their \( y \) coordinates.)

Step 4  Scan the points in the vertical strip \( S \) from the lowest up.

For every point \( p(x,y) \) in the strip, inspect points in the strip that may be closer to \( p \) than \( d \). There can be no more than 5 such points following \( p \) on the strip list!
Efficiency of the Closest-Pair Algorithm

Running time of the algorithm is described by

\[ T(n) = 2T(n/2) + M(n), \text{ where } M(n) \in O(n) \]

By the Master Theorem (with \( a = 2, b = 2, d = 1 \))

\[ T(n) \in O(n \log n) \]
General Divide-and-Conquer Recurrence

\[ T(n) = aT(n/b) + f(n) \quad \text{where } f(n) \in \Theta(n^d), \quad d \geq 0 \]

**Master Theorem:**
- \( a < b^d \), \( T(n) \in \Theta(n^d) \)
- \( a = b^d \), \( T(n) \in \Theta(n^d \log n) \)
- \( a > b^d \), \( T(n) \in \Theta(n^{\log_b a}) \)

**Note:** The same results hold with \( \Omega \) instead of \( \Theta \).