MCS-375: Algorithms: Analysis and Design Handout #DC2
San Skulrattanakulchai
Gustavus Adolphus College December 3, 2013

Integer Multiplication

Let 2 n-bit positive integers x and y be given. We wish to find their product xy. Standard multiplication algorithm takes $\Theta(n^2)$ bit operations. We will present an algorithm that beats this bound.

Gauss once made the following observation: Computing the product of two complex numbers,

$$(a + bi)(c + di) = (ac - bd) + (bc + ad)i,$$

seems to involve four real-number multiplications: ac, bd, bc, and ad. Nevertheless, it can be achieved with three: ac, bd, and $(a + b)(c + d)$. This is because

$$bc + ad = (a + b)(c + d) - ac - bd.$$

Gauss' observation is the basis for the following divide-and-conquer integer multiplication algorithm.

Input: Nonnegative integers x and y, each n bits long
Output: $x \cdot y$

MULTIPLY(x, y) {
  if $n = 1$ **then return** $x * y$
  $x_1 \leftarrow$ leftmost $\lfloor n/2 \rfloor$ bits of x; $x_0 \leftarrow$ rightmost $\lceil n/2 \rceil$ bits of x
  $y_1 \leftarrow$ leftmost $\lfloor n/2 \rfloor$ bits of y; $y_0 \leftarrow$ rightmost $\lceil n/2 \rceil$ bits of y
  $p_1 \leftarrow$ MULTIPLY(x_1, y_1)
  $p_2 \leftarrow$ MULTIPLY(x_0, y_0)
  $p_3 \leftarrow$ MULTIPLY$(x_1 + x_0,\ y_1 + y_0)$
  return $p_1 * 2^{2\lceil n/2 \rceil} + (p_3 - p_1 - p_2) * 2^{\lceil n/2 \rceil} + p_2$
}

The procedure's worst-case running time $T(n)$ satisfies the recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ n + 3T(n/2) & \text{if } n > 1. \end{cases}$$

By the Master Theorem, $T(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.59})$, which is $o(n^2)$.

Remarks.

1. Multiplying an n-bit number z by a perfect power of two, say 2^k, where k is $O(n)$, takes time $O(n)$ since it can be done simply by shifting the bits of z.
2. In practice, instead of $n = 1$, use the base case that corresponds to the natural word size of the processor (say $n = 32$).
3. An even faster multiplication algorithm exists and is based on the FFT.