Algorithms – CMSC-27200/37000
Divide and Conquer: *The Karatsuba–Ofman algorithm*
(multiplication of large integers)

The Karatsuba–Ofman algorithm provides a striking example of how the "Divide and Conquer" technique can achieve an asymptotic speedup over an ancient algorithm.

The classroom method of multiplying two n-digit integers requires $\Theta(n^2)$ digit operations. We shall show that a simple recursive algorithm solves the problem in $\Theta(n^\alpha)$ digit operations, where $\alpha = \log_2 3 \approx 1.58$. This is a considerable improvement of the asymptotic order of magnitude of the number of digit-operations.

We describe the procedure in *pseudocode.*

Procedure $KO(X, Y)$

Input: X, Y: n-digit integers.
Output: the product $X * Y$.
Comment: We assume $n = 2^k$, by prefixing X, Y with zeros if necessary.

1. **if** $n = 1$ **then** use multiplication table to find $T := X * Y$
2. **else** split X, Y in half:
3. $X =: 10^{n/2} X_1 + X_2$
4. $Y =: 10^{n/2} Y_1 + Y_2$
5. *Comment:* X_1, X_2, Y_1, Y_2 each have $n/2$ digits.
6. $U := KO(X_1, Y_1)$
7. $V := KO(X_2, Y_2)$
8. $W := KO(X_1 - X_2, Y_1 - Y_2)$
9. $Z := U + V - W$
10. $T := 10^n U + 10^{n/2} Z + V$
11. *Comment:* So $U = X_1 * Y_1$, $V = X_2 * Y_2$, $W = (X_1 - X_2) * (Y_1 - Y_2)$, and therefore $Z = X_1 * Y_2 + X_2 * Y_1$. Finally we conclude that $T = 10^n X_1 * Y_1 + 10^{n/2}(X_1 * Y_2 + X_2 * Y_1) + X_2 * Y_2 = X * Y$.

12. **return** T

Analysis. This is a *recursive* algorithm: during execution, it calls smaller instances of itself.

Let $M(n)$ denote the number of *digit-multiplications* (line 1) required by the Karatsuba–Ofman algorithm when multiplying two n-digit integers ($n = 2^k$). In lines 6,7,8 the procedure calls itself three times on $n/2$-digit integers; therefore

$$M(n) = 3M(n/2). \tag{1}$$

This equation is a simple *recurrence* which we may solve directly as follows. Applying equation (1) to $M(n/2)$ we obtain $M(n/2) = 3M(n/4)$; therefore $M(n) = 9M(n/4)$. Continuing similarly we see that $M(n) = 27M(n/8)$, and it follows by induction on i that for every i ($i \leq k$),

$$M(n) = 3^i M(n/2^i).$$

Setting $i = k$ we find that $M(n) = 3^k M(n/2^k) = 3^k M(1) = 3^k$. Notice that $k = \log n$ (base 2 logarithm), therefore $\log M(n) = k \log 3$ and hence $M(n) = 2^{\log M(n)} = 2^{k \log 3} = (2^k)^{\log 3} = n^{\log 3}$.

It would seem that we reduced the number of digit-multiplications to $n^{\log 3}$ at the cost of an increased number of additions (lines 9, 10). Appearances are deceptive: actually, the procedure achieves similar savings in terms of the total number of digit-operations (additions as well as multiplications).

To see this, let $T(n)$ be the total number of digit-operations (additions, multiplications, bookkeeping (copying digits, maintaining links)) required by the Karatsuba–Ofman algorithm. Then

$$T(n) = 3T(n/2) + O(n) \tag{2}$$

where the term $3T(n/2)$ comes, as before, from lines 6,7,8; the additional $O(n)$ term is the number of digit-additions required to perform the additions and subtractions in lines 9 and 10. The $O(n)$ term also includes bookkeeping costs.

We shall learn later how to analyse recurrences of the form (2). It turns out that the additive $O(n)$ term does not change the order of magnitude, and the result will still be

$$T(n) = \Theta(n^{\log 3}) \approx \Theta(n^{1.58}). \tag{3}$$