Karatsuba’s Algorithm

Object: To multiply two (roughly) \( n \)-digit numbers.

Numbers: Numbers are represented as arrays \( A[0\cdots N] \) with \( A[I] \) is the I-th digit, starting the count at the right with \( A[0] \). Thus 904 is represented as \( A[0] = 4, A[1] = 0, A[2] = 9 \). (We’ll do examples in decimal but in the computer it is usually in binary.)

Addition: Given \( A[0\cdots N], B[0\cdots N] \) we find their sum \( C[0\cdots (N + 1)] \) by the “standard” method learned at age eight.

\[ \text{CARRY}=0 \text{ (*initialization*)} \]

\[
\text{FOR I}= 0 \text{ TO N+1}
\]

\[
C[I]=A[I]+B[I]+\text{CARRY}
\]

\[
\text{IF } C[I] \leq 9 \text{ do}
\]

\[
\text{CARRY} = 0
\]

\[
\text{ELSE do (*C[I] \geq 10*)}
\]

\[
\text{CARRY} =1
\]

\[
C[I] = C[I] - 100
\]

This takes a single pass and is a \( \Theta(n) \) algorithm. Subtraction (a good exercise!) is similar, and also a linear time, that is \( \Theta(n) \), algorithm.

The idea of Karatsuba’s Algorithm is to take two \( n \)-digit numbers \( \alpha, \beta \) and to cut them (thinking of them as strings of digits) in half, writing (assume \( n \) even for convenience)

\[
\alpha = 10^{n/2} x + y
\]

\[
\beta = 10^{n/2} z + w
\]

We want the product \( \gamma = \alpha \beta \) and

\[
\gamma = 10^n (xz) + 10^{n/2} (xw + yz) + yw
\]

(Note that multiplying by \( 10^n \) or \( 10^{n/2} \) is not “real” multiplication but just string manipulation and so is very quick.) It looks like we want to do four multiplications \( xz, xw, yz, yw \) of \( n/2 \)-digit numbers. But here is the clever idea:

1. Find \( xz \). (multiply two \( n/2 \)-digit numbers)
2. Find \( yw \). (multiply two \( n/2 \)-digit numbers)
3. Find \( x + y \). (Addition, time \( \Theta(n) \).
4. Find \( z + w \). (Addition, time \( \Theta(n) \).
5. Find \( (x + y)(z + w) \) (multiply two \( n/2 \)-digit\(^1 \) numbers)

\(^1\)maybe one more digit but that will have negligible affect
6. Find \( xw + yz = (x + y)(z + w) - xz - yw \) (Two subtractions, time \( \Theta(n) \))

7. Put parts together to get \( \gamma \) (Two additions, time \( \Theta(n) \))

The calls to multiply the smaller numbers are done **recursively**. Letting \( T(n) \) be the time for Karatsuba’s algorithm the key point is that there are only *three* recursive calls and an “overhead” of \( \Theta(n) \). So we have the recursion

\[
T(n) = 3T(n/2) + \Theta(n)
\]

From the Master Theorem

\[
T(n) = \Theta(n^{\log_3 4}) = \Theta(n^{1.58...})
\]

where the exact exponent is \( \frac{\log 3}{\log 2} \). So this is better (of course, in this course by “better” we mean faster for \( n \) large) than the normal \( \Theta(n^2) \) algorithm learned at age eleven.