11.3
Faster multiplication: Karatsuba's Algorithm

A Trick of Gauss

Carl Friedrich Gauss: 1777–1855 "Prince of Mathematicians"

Observation: Multiply two complex numbers: $(a + bi)$ and $(c + di)$

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

How many multiplications do we need?

Only 3! If we do extra additions and subtractions.
Compute $ac, bd, (a + b)(c + d)$. Then $(ad + bc) = (a + b)(c + d) - ac - bd$

A Trick of Gauss

Carl Friedrich Gauss: 1777–1855 "Prince of Mathematicians"

Observation: Multiply two complex numbers: $(a + bi)$ and $(c + di)$

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

How many multiplications do we need?

Only 3! If we do extra additions and subtractions.
Compute $ac, bd, (a + b)(c + d)$. Then $(ad + bc) = (a + b)(c + d) - ac - bd$

A Trick of Gauss

Carl Friedrich Gauss: 1777–1855 "Prince of Mathematicians"

Observation: Multiply two complex numbers: $(a + bi)$ and $(c + di)$

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

How many multiplications do we need?

Only 3! If we do extra additions and subtractions.
Compute $ac, bd, (a + b)(c + d)$. Then $(ad + bc) = (a + b)(c + d) - ac - bd$

Gauss technique for polynomials

$p(x) = ax + b$ and $q(x) = cx + d.$

$$p(x)q(x) = acx^2 + (ad + bc)x + bd.$$

$$p(x)q(x) = acx^2 + ((a + b)(c + d) - ac - bd)x + bd.$$

Gauss technique for polynomials

$p(x) = ax + b$ and $q(x) = cx + d$.

$$p(x)q(x) = acx^2 + (ad + bc)x + bd.$$

$$p(x)q(x) = acx^2 + \big((a + b)(c + d) - ac - bd\big)x + bd.$$

Improving the Running Time

$$bc = b(x)c(x) = (b_L x + b_R)(c_L x + c_R)$$

Improving the Running Time

$$
\begin{aligned}
bc = b(x)c(x) &= (b_L x + b_R)(c_L x + c_R) \\
&= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R
\end{aligned}
$$

Improving the Running Time

$$
\begin{aligned}
bc &= b(x)c(x) = (b_L x + b_R)(c_L x + c_R) \\
&= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R \\
&= (b_L * c_L)x^2 + \Big((b_L + b_R) * (c_L + c_R) - b_L * c_L - b_R * c_R\Big)x + b_R * c_R
\end{aligned}
$$

Improving the Running Time

$$\begin{aligned} bc &= b(x)c(x) = (b_L x + b_R)(c_L x + c_R) \\ &= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R \\ &= (b_L * c_L)x^2 + \Big((b_L + b_R) * (c_L + c_R) - b_L * c_L - b_R * c_R\Big)x + b_R * c_R \end{aligned}$$

Recursively compute only $b_L c_L, b_R c_R, (b_L + b_R)(c_L + c_R)$.

Improving the Running Time

$$
\begin{aligned}
bc &= b(x)c(x) = (b_L x + b_R)(c_L x + c_R) \\
&= b_L c_L x^2 + (b_L c_R + b_R c_L)x + b_R c_R \\
&= (b_L * c_L)x^2 + \Big((b_L + b_R) * (c_L + c_R) - b_L * c_L - b_R * c_R\Big)x + b_R * c_R
\end{aligned}
$$

Recursively compute only $b_L c_L, b_R c_R, (b_L + b_R)(c_L + c_R)$.

Time Analysis

Running time is given by

$$T(n) = 3T(n/2) + O(n) \qquad T(1) = O(1)$$

which means $T(n) = O(n^{\log_2 3}) = O(n^{1.585})$

State of the Art

Schönhage-Strassen 1971: $O(n \log n \log \log n)$ time using Fast-Fourier-Transform (FFT)

Martin Fürer 2007: $O(n \log n 2^{O(\log^* n)})$ time

Conjecture

There is an $O(n \log n)$ time algorithm.

THE END

...

(for now)