Faster Polynomial Multiplication via Discrete Fourier Transforms

Alexey Pospelov

Computer Science Department, Saarland University

Cluster of Excellence Multimodal Computing and Interaction

June 14th, 2011

Polynomial multiplication

Given

$$a(x) = a_0 + a_1 x + \cdots + a_n x^n, \quad b(x) = b_0 + b_1 x + \cdots + b_n x^n,$$

Compute

$$c(x) = c_0 + c_1 x + \cdots + c_{2n} x^{2n} = a(x)b(x).$$

Polynomial multiplication

Given

$$a(x) = a_0 + a_1 x + \cdots + a_n x^n, \quad b(x) = b_0 + b_1 x + \cdots + b_n x^n,$$

Compute

$$c(x) = c_0 + c_1 x + \cdots + c_{2n} x^{2n} = a(x)b(x).$$

For all $0 \leq i \leq 2n$, compute

$$c_i = \begin{cases} a_0 b_i + a_1 b_{i-1} + \cdots + a_i b_0, & 0 \leq i \leq n, \\ a_{i-n} b_n + a_{i-n+1} b_{n-1} + \cdots + a_n b_{i-n}, & n < i \leq 2n. \end{cases}$$

In what model?

- Arithmetic circuits with binary "$+$", "$-$", "$\cdot$"
- Each binary gate has unit cost
- No divisions
- Constants from the field available at no cost
- Inputs are the coefficients of the polynomials to be multiplied
- Outputs are the coefficients of the product polynomial
- Interested in a circuit for degree n polynomial multiplication of the minimal size

History and state of the art

School method: $O(n^2)$

Karatsuba 1960: $O(n^{\log_2 3}) = O(n^{1.585})$

Toom 1963: $n^{1+O(1/\sqrt{\log n})} = O(n^{1+\epsilon})$, for any fixed $\epsilon > 0$

- Over infinite fields

History and state of the art

School method: $O(n^2)$

Karatsuba 1960: $O(n^{\log_2 3}) = O(n^{1.585})$

Toom 1963: $n^{1+O(1/\sqrt{\log n})} = O(n^{1+\epsilon})$, for any fixed $\epsilon > 0$

- Over infinite fields

Schönhage-Strassen 1971: $O(n \log n \log \log n)$

- Didn't work for fields of char $= 2$

History and state of the art

School method: $O(n^2)$

Karatsuba 1960: $O(n^{\log_2 3}) = O(n^{1.585})$

Toom 1963: $n^{1+O(1/\sqrt{\log n})} = O(n^{1+\epsilon})$, for any fixed $\epsilon > 0$

- Over infinite fields

Schönhage-Strassen 1971: $O(n \log n \log \log n)$

- Didn't work for fields of char $= 2$

Schönhage 1977: $O(n \log n \log \log n)$, over field of char $= 2$

Kaminski 1988, Cantor-Kaltofen 1991: $O(n \log n \log \log n)$, over arbitrary **algebras**

Over $\mathbb{C}$ or $\mathbb{R}$: $O(n \log n)$

All general lower bounds: $\Omega(n)$.

Multiplication in $O(n \log n)$

Given

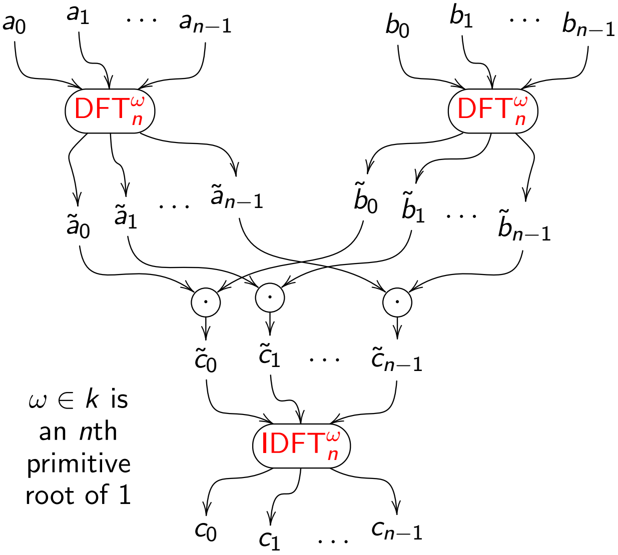
$$a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1},$$
$$b(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1},$$

Compute

$$c(x) = c_0 + c_1 x + \cdots + x_{n-1} x^{n-1} = a(x) b(x) \pmod{x^n - 1}.$$

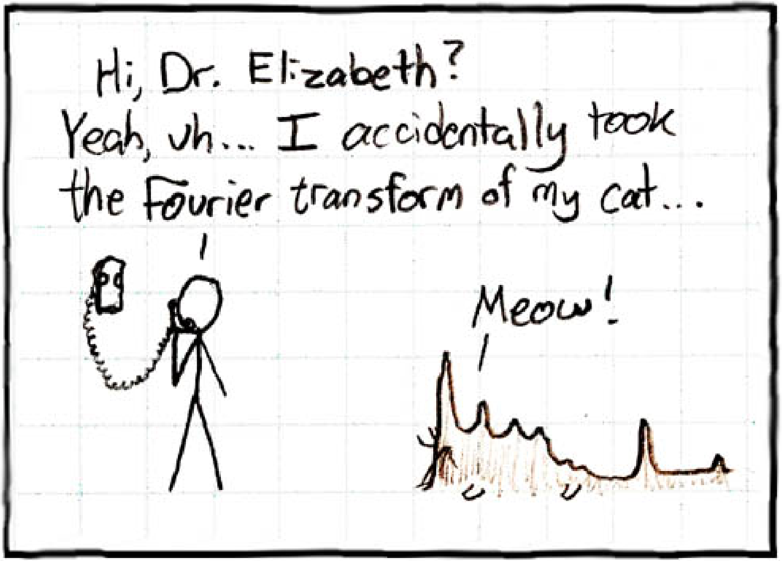
(Can always choose a larger n and pad polynomials with zeroes to reduce the ordinary polynomial multiplication to the product in $k[x]/(x^n - 1)$.)

Multiplication in $O(n \log n)$



$\omega \in k$ is an nth primitive root of 1

Discrete Fourier transform



Discrete Fourier transform

- Maps a degree $n-1$ polynomial to its values at n distinct nth roots of unity:

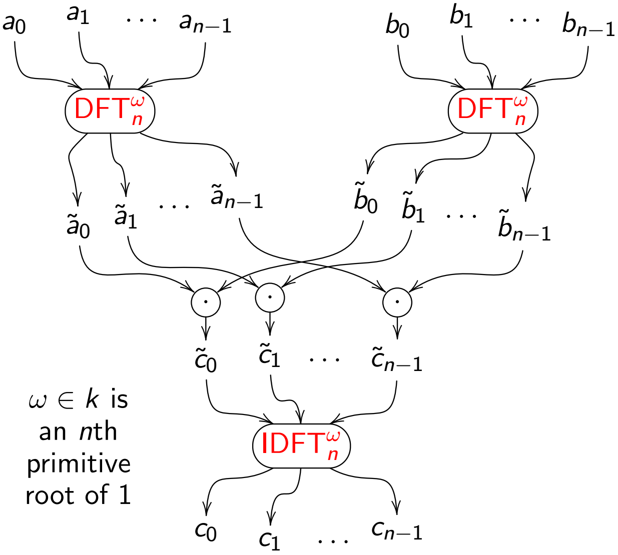
$$\tilde{a}_i := a(\omega^i) = \sum_{j=0}^{n-1} a_j \omega^{ij}, \qquad 0 \le i \le n-1$$

$$\mathrm{DFT}_n^{\omega} : (a_0, a_1, \ldots, a_{n-1}) \mapsto (\tilde{a}_0, \tilde{a}_1, \ldots, \tilde{a}_{n-1})$$

  (ω is a primitive nth root of unity)
- Linear transform: $\mathrm{DFT}_n^{\omega} : k[x] \to k^n$
- Isomorphism: $\mathrm{DFT}_n^{\omega} : k[x]/(x^n - 1) \to k^n$
- Can be *often* computed in $O(n \log n)$
- The inverse isomorphism is almost a DFT again:

$$\frac{1}{n} \mathrm{DFT}_n^{\omega^{n-1}} : (\tilde{a}_0, \tilde{a}_1, \ldots, \tilde{a}_{n-1}) \mapsto (a_0, a_1, \ldots, a_{n-1})$$

Discrete Fourier transform



$\omega \in k$ is an nth primitive root of 1

Discrete Fourier transform

- $L_k(n)$: the complexity of degree n polynomial multiplication over a field k
- $D_k(n)$: the complexity of computing length n DFT over k

$$L_k(n) \leq 3D_k(n) + 2n = O(n \log n).$$

Note: we need roots of unity.

What if roots of unity are not available?

What if roots of unity are not available?

Attach them!

What if roots of unity are not available?

Attach them!

- Switch from the field k to its algebraic extension $\mathcal{A}_m$ where roots of unity of sufficiently large order exist.

What if roots of unity are not available?

Attach them!

- Switch from the field k to its algebraic extension $\mathcal{A}_m$ where roots of unity of sufficiently large order exist.
- More precisely: take a (ring) extension $\mathcal{A}_m$ of k of degree m over k with a 2ℓth root of unity $\omega \in \mathcal{A}_m$:
  - For example,
$$\mathcal{A}_m = k[x]/p_m(x),$$
  - $p_m(x) \in k[x]$ is a polynomial of degree m,
  - $p_m(x)$ vanishes on $\omega_{2\ell}$,
  - ($\omega_{2\ell}$ is a primitive 2ℓth root of unity in the algebraic closure of the field k.)

What if roots of unity are not available?

Attach them!

- Switch from the field k to its algebraic extension $\mathcal{A}_m$ where roots of unity of sufficiently large order exist.
- More precisely: take a (ring) extension $\mathcal{A}_m$ of k of degree m over k with a 2ℓth root of unity $\omega \in \mathcal{A}_m$:
  - For example,
  $$\mathcal{A}_m = k[x]/p_m(x),$$
  - $p_m(x) \in k[x]$ is a polynomial of degree m,
  - $p_m(x)$ vanishes on $\omega_{2\ell}$,
  - ($\omega_{2\ell}$ is a primitive 2ℓth root of unity in the algebraic closure of the field k.)

How can it help?

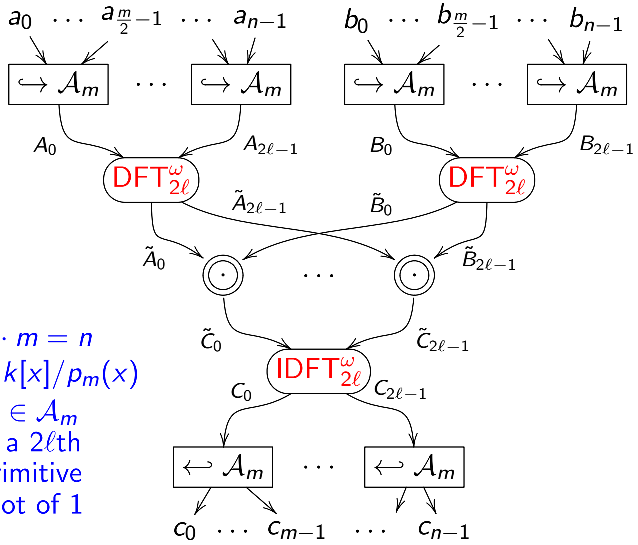
What if roots of unity are not available?

Attach them!

- Switch from the field k to its algebraic extension $\mathcal{A}_m$ where roots of unity of sufficiently large order exist.
- More precisely: take a (ring) extension $\mathcal{A}_m$ of k of degree m over k with a 2ℓth root of unity $\omega \in \mathcal{A}_m$:
  - For example,
  $$\mathcal{A}_m = k[x]/p_m(x),$$
  - $p_m(x) \in k[x]$ is a polynomial of degree m,
  - $p_m(x)$ vanishes on $\omega_{2\ell}$,
  - ($\omega_{2\ell}$ is a primitive 2ℓth root of unity in the algebraic closure of the field k.)

How can it help? See next slide.

Fast polynomial multiplication



$2\ell \cdot m = n$

$\mathcal{A}_m = k[x]/p_m(x)$

$\omega \in \mathcal{A}_m$
is a 2ℓth
primitive
root of 1

Fast polynomial multiplication

In this case

$$L_k(n) \leq 2\ell L_k(m) + 3D_{\mathcal{A}_m}(n) \cdot \text{complexity of aritrhmetics in } \mathcal{A}_m + \text{cost of embedding and unembedding in } \mathcal{A}_m$$

Our contribution #1:

- Formalize this kind of algorithms
- The relation between m and 2ℓ is a barrier for the algorithm's performance
- This relation depends heavily on the field properties
- The cost of the DFT can usually be made $O(n \log n)$
- Embedding and unembedding run usually in linear time, e.g., if $p_m(x)$ is sparse

Does it work?

Yes!!!

Schönhage-Strassen 1971: $\ell = m$

Schönhage 1977: $3\ell = 2m$ (+ a little trick)

Kaminski 1988: $\ell = \phi(m)$ (Euler's totient function)

Cantor-Kaltofen 1991: $\ell = m$ (and $\mathcal{A}_m$ is *a little* more complicated than $k[x]/p_m(x)$)

Slow fields

Slow fields

Recall:

$$L_k(n) \leq 2\ell L_k(m) + 3D_{\mathcal{A}_m}(n) \cdot \text{complexity of arithmetics in } \mathcal{A}_m + \text{cost of embedding and unembedding in } \mathcal{A}_m$$

Ideally we want m to be small and ℓ to be large.

Slow fields

Recall:

$$L_k(n) \leq 2\ell L_k(m) + 3D_{\mathcal{A}_m}(n) \cdot \text{complexity of arithmetics in } \mathcal{A}_m + \text{cost of embedding and unembedding in } \mathcal{A}_m$$

Ideally we want m to be small and ℓ to be large.

Definition

For a field k, and n, s.t. $\text{char}\, k \nmid n$, let $f_k(n)$ be $[k(\omega_n) : k]$, the *degree function* of k.

Slow fields

Recall:

$$L_k(n) \leq 2\ell L_k(m) + 3D_{\mathcal{A}_m}(n) \cdot \text{complexity of arithmetics in } \mathcal{A}_m + \text{cost of embedding and unembedding in } \mathcal{A}_m$$

Ideally we want m to be small and ℓ to be large.

Definition

For a field k, and n, s.t. char $k \nmid n$, let $f_k(n)$ be $[k(\omega_n) : k]$, the *degree function* of k.

Our contribution #2:

- If $f_k(n) = o(\log \log n)$ for some *not too sparse set of n* then k is *fast* and $L_k(n) = o(n \log n \log \log n)$
- If $f_k(n) = \Omega(n^{1-\epsilon})$ for any fixed $\epsilon > 0$, then k is *slow* and any algorithm of that kind runs in $\Omega(n \log n \log \log n)$

More details

- To attach an ℓth root of unity we need an extension of degree at least $f_k(\ell)$
- The degree of the polynomial is then $\sim \ell \cdot f_k(\ell)$
- For the least solution i_0 of $i \cdot f_k(i) \geq n$, $f_k^{\surd}(n) := f_k(i_0)$
- The number of recursive steps is at least the number of $f_k^{\surd}(f_k^{\surd}(\cdots f_k^{\surd}(n)\cdots))$, until the value becomes $O(1)$
- This superposition depth will be denoted $(f_k^{\surd})^*(n)$
- The cost of all steps on a single recursion level is determined by the complexity of the DFTs, and is $\Theta(n \log n)$

More details

- To attach an ℓth root of unity we need an extension of degree at least $f_k(\ell)$
- The degree of the polynomial is then $\sim \ell \cdot f_k(\ell)$
- For the least solution i_0 of $i \cdot f_k(i) \geq n$, $f_k^{\surd}(n) := f_k(i_0)$
- The number of recursive steps is at least the number of $f_k^{\surd}(f_k^{\surd}(\cdots f_k^{\surd}(n)\cdots))$, until the value becomes $O(1)$
- This superposition depth will be denoted $(f_k^{\surd})^*(n)$
- The cost of all steps on a single recursion level is determined by the complexity of the DFTs, and is $\Theta(n \log n)$
- The total cost is estimated as

$$\Omega(n \log n) \cdot (f_k^{\surd})^*(n)$$

“Lower bound”

For the rational field $\mathbb{Q}$, for all n

$$f_{\mathbb{Q}}(n) = \phi(n) \geq c \cdot \frac{n}{\log \log n},$$

and

$$(f_{\mathbb{Q}}^{\sqrt{}})^{*}(n) = \Omega(\log \log n).$$

“Lower bound”

For the rational field $\mathbb{Q}$, for all n

$$f_{\mathbb{Q}}(n) = \phi(n) \geq c \cdot \frac{n}{\log\log n},$$

and

$$(f_{\mathbb{Q}}^{\sqrt{}})^*(n) = \Omega(\log\log n).$$

Complexity of any DFT-based multiplication algorithm is then

$$\Omega(n \log n \log\log n).$$

It follows that over $\mathbb{Q}$ we need another kind of an algorithm.

Summary

- Uniform treatment of all known asymptotically fastest polynomial multiplication algorithms w.r.t. the total complexity
- A way to improve the total complexity upper bounds over certain fields
- Impossibility to improve Schönhage-Strassen over any fields (and rings or algebras) of characteristic 0
- In particular, no light at the end of the tunnel for polynomial multiplication over $\mathbb{Q}$

Summary

- Uniform treatment of all known asymptotically fastest polynomial multiplication algorithms w.r.t. the total complexity
- A way to improve the total complexity upper bounds over certain fields
- Impossibility to improve Schönhage-Strassen over any fields (and rings or algebras) of characteristic 0
- In particular, no light at the end of the tunnel for polynomial multiplication over $\mathbb{Q}$
- Over fields of positive characteristic,

Thank you for attention!