Big integer multiplication

Svyatoslav Covanov
Supervisors: Jérémie Detrey and Emmanuel Thomé

Team CARAMBA

June 29, 2016

1. Fast Fourier Transform
2. Fürer
3. Using generalized Fermat primes

Naive multiplication

How to multiply two N-bit integers a and b ?

Naive multiplication

How to multiply two N-bit integers a and b ?

Schoolbook multiplication: $O(N^2)$ bit complexity.

Karatsuba:

- $O(N^{\log_2 3})$ bit complexity.
- Transformation of integers into polynomials.

Multiplying integer using polynomials

Input: 2 numbers a and b of N bits.
Output: 2 polynomials $A = \sum_i a_i x^i$ and $B = \sum_i b_i x^i$ of degree $n - 1$.

$$a = a_0 + 2^k \times a_1 + \cdots + a_{n-1} \times 2^{(n-1)k} = A(2^k)$$
$$b = b_0 + 2^k \times b_1 + \cdots + b_{n-1} \times 2^{(n-1)k} = B(2^k)$$

Multiplying integer using polynomials

Input: 2 numbers a and b of N bits.
Output: 2 polynomials $A = \sum_i a_i x^i$ and $B = \sum_i b_i x^i$ of degree $n-1$.

$$a = a_0 + 2^k \times a_1 + \cdots + a_{n-1} \times 2^{(n-1)k} = A(2^k)$$
$$b = b_0 + 2^k \times b_1 + \cdots + b_{n-1} \times 2^{(n-1)k} = B(2^k)$$

- $\mathcal{R}$ is a commutative ring.
- $\begin{array}{l} A \longrightarrow \tilde{A} \in \mathcal{R}[x] \\ B \longrightarrow \tilde{B} \in \mathcal{R}[x] \end{array}$.
- $C \longrightarrow \tilde{C} = \tilde{A} \cdot \tilde{B}$ is injective:

$$\forall j, |c_j| = |\sum_{i=0}^{j} a_i \cdot b_{j-i}| < (j+1) \cdot 2^{2k} \leq n \cdot 2^{2k}.$$

- We choose $2n$ distinct points w_i of $\mathcal{R}$.
- Computation of $A(w_i)$ and $B(w_i)$: equivalent to the product

$$\begin{pmatrix} 1 & w_0 & \dots & w_0^{2n-1} \\ 1 & w_1 & \dots & w_1^{2n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & w_{2n-1} & \dots & w_{2n-1}^{2n-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} A(w_0) \\ \vdots \\ A(w_i) \\ \vdots \\ A(w_{2n-1}) \end{pmatrix}.$$

- Pointwise products $A(w_i) \cdot B(w_i) = C(w_i)$.
- Lagrange interpolation of C from the $2n$ points $A(w_i) \cdot B(w_i)$:

$$\begin{pmatrix} 1 & w_0 & \dots & w_0^{2n-1} \\ 1 & w_1 & \dots & w_1^{2n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & w_{2n-1} & \dots & w_{2n-1}^{2n-1} \end{pmatrix}^{-1} \cdot \begin{pmatrix} A(w_0)B(w_0) \\ \vdots \\ A(w_{2n-1})B(w_{2n-1}) \end{pmatrix}.$$

Discrete Fourier Transform (DFT)

If $\mathcal{R}$ is a ring containing a $2n$-th principal root of unity ω:
let

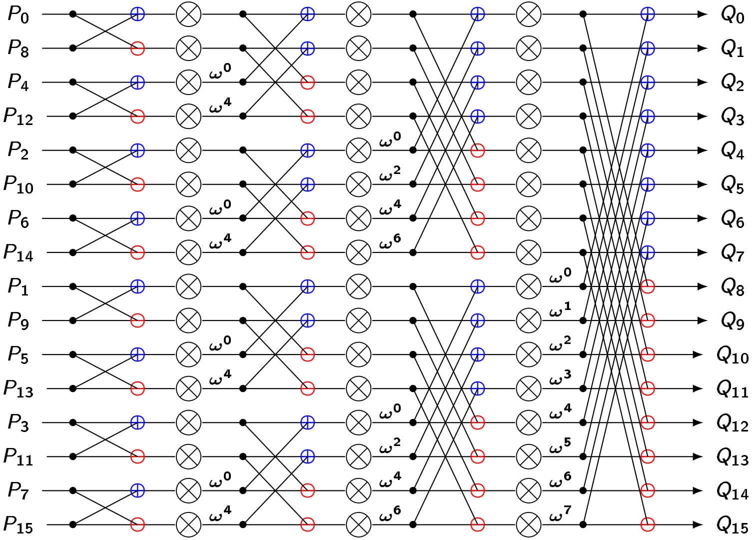
$$M_{2n}(\omega) = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{2n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2n-1} & \dots & (\omega^{2n-1})^{2n-1} \end{pmatrix}.$$

The root ω is said to be a $2n$-th principal root of unity if

$$\forall i \in [1, 2n-1], \sum_{j=0}^{2n-1} \omega^{ij} = 0.$$

FFT(A, ω, $2n$)

if $n = 2$ **then**
  return $A_0 + A_1 + X(A_0 - A_1)$
end if
$A_{even} \leftarrow (A_{2i})_i$
$A_{odd} \leftarrow (A_{2i+1})_i$
$\hat{A}_{even} \leftarrow \text{FFT}(A_{even},\ \omega^2,\ n)$ ▷ $\hat{A}_{even} = \sum_{i \in [0,n-1]} A_{even}(\omega^{2i})X^i$
$\hat{A}_{odd} \leftarrow \text{FFT}(A_{odd},\ \omega^2,\ n)$ ▷ $\hat{A}_{odd} = \sum_{i \in [0,n-1]} A_{odd}(\omega^{2i})X^i$
$\hat{A} \leftarrow \hat{A}_{odd}(X) + \hat{A}_{even}(\omega X) + X^n \cdot (\hat{A}_{odd}(X) - \hat{A}_{even}(\omega X))$
return $\hat{A}$



$\Rightarrow 2n = 16$ points, $\log(2n) = 4$ levels, $n(\log(2n) - 1) = 24$ multiplications.

Choice of the ring

1. N: # bits of the integers that we multiply
2. $n - 1$: degree of the polynomials A and B used to represent a and b
3. k: # bits used to encode the coefficients of A and B: $a = A(2^k)$, $b = B(2^k)$ and $n \cdot k = N$.

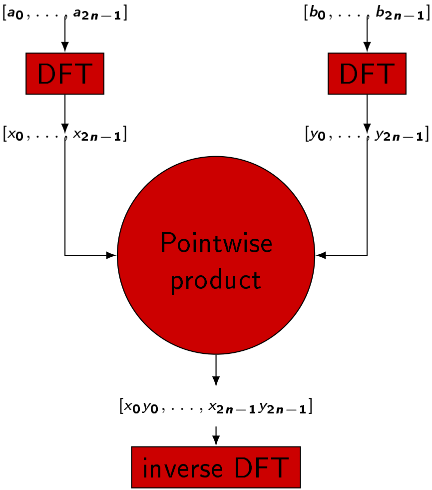
Choice of the ring

1. N: # bits of the integers that we multiply
2. $n-1$: degree of the polynomials A and B used to represent a and b
3. k: # bits used to encode the coefficients of A and B: $a = A(2^k)$, $b = B(2^k)$ and $n \cdot k = N$.

Examples: (Schönhage-Strassen algorithms)

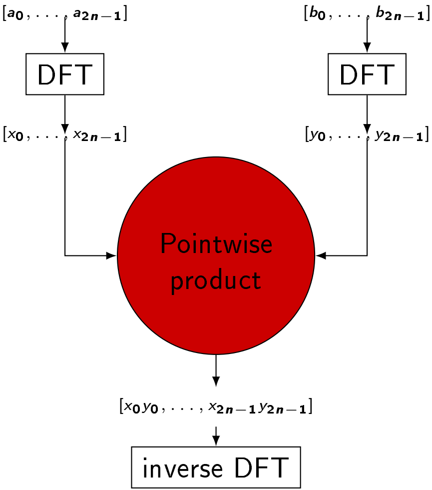
- $\mathcal{R} = \mathbb{C}$: $\omega = \exp(i\pi/n)$, provided that we allow enough precision.
- $\mathcal{R} = \mathbb{Z}/(2^e+1)\mathbb{Z}$: $\omega = 2^j$ is a $2e/j$-th principal root of unity.

Complex Case



- $O(n \log n)$ expensive multiplications during the FFT
- $2n$ expensive multiplications during the pointwise product

Modular Case



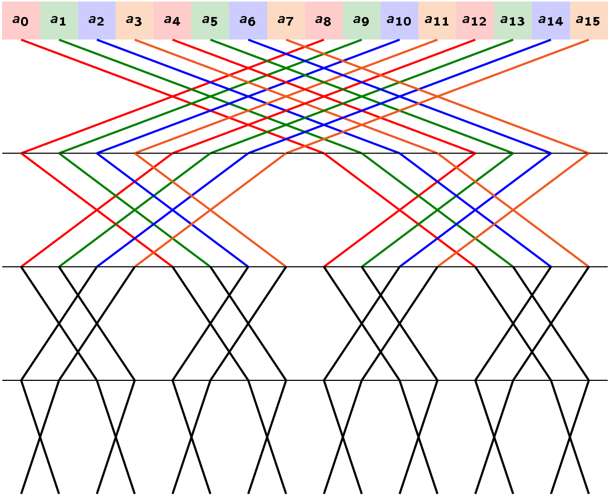
- $O(n \log n)$ trivial multiplications during the FFT
- $2n$ expensive multiplications during the pointwise product

Some remarks

Case	Degree	Mult. by a root	Recursion	Complexity
$\mathbb{C}$	$O(N/\log N)$	expensive	$O(\log N)$	$N \ \log N \ \log\log N \cdots 2^{O(\log^* N)}$
$\mathbb{Z}/(2^e+1)\mathbb{Z}$	$O(\sqrt{N})$	cheap	$O(\sqrt{N})$	$N \ \log N \ \log\log N$

In $\mathbb{C}$, computing an FFT in $\{1, -1, i, -i\}$ is quite easy. But less obvious for superior orders...

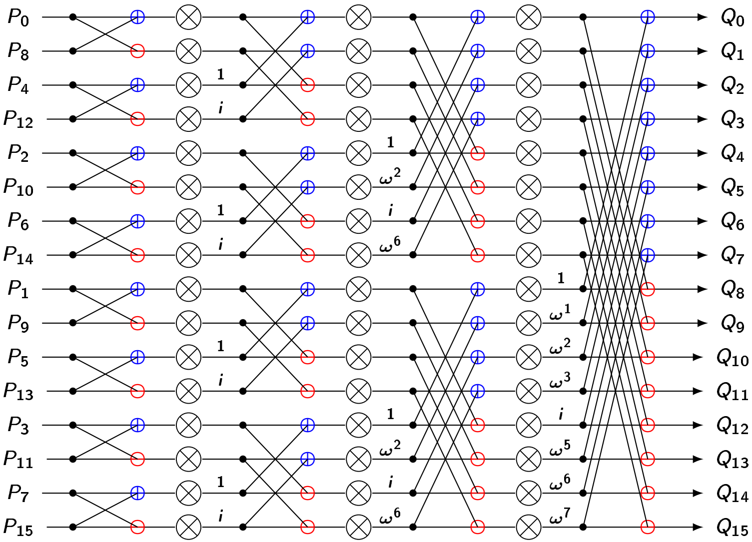
$4 \cdot \mathrm{DFT}(4)$:



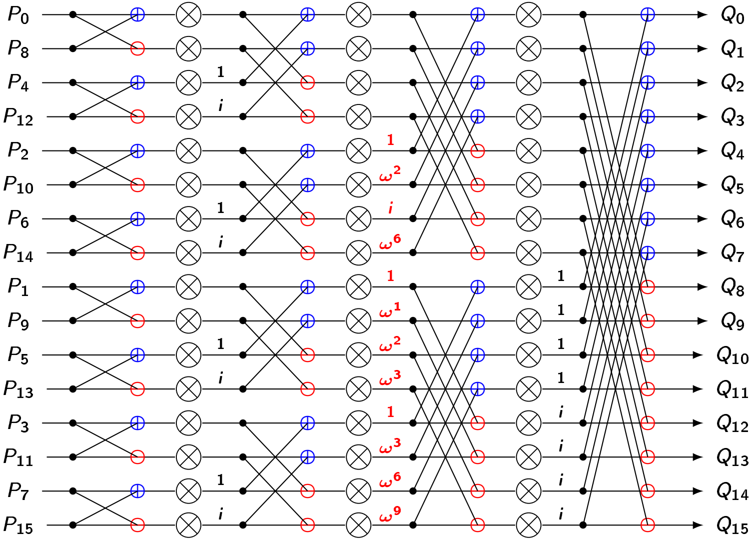
Matrix point of view:

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix} \cdot M_4^T(\omega^4)$$

An example in Complex Field: radix-2 FFT



An example in Complex Field: radix-4 FFT



Fürer's algorithm

$$\mathrm{DFT}(2P\cdot(n/P)) = 2P\cdot\mathrm{DFT}(n/P) + \text{Twiddle factors} + (n/P)\cdot\mathrm{DFT}(2P).$$

- $\mathcal{R}$ is the ring $\mathcal{R} = \mathbb{C}[x]/(x^P+1)$ (P divides n).
  $\Rightarrow$There exists a $2n$-th root of unity ρ such that $\rho^{n/P} = x$.
- Computation of $2n$-point DFT with radix-$2P$ FFT ($2P \approx \log N$).
- $\log_{2P} 2n$ levels of recursion:

$$\underbrace{\log_{2P}(2n)}_{\text{nb. of levels}} \cdot \underbrace{2n}_{\text{mult. per level}} \cdot \underbrace{\mathcal{M}_{\mathcal{R}}}_{\text{cost of a mult. in } \mathcal{R}}$$

expensive multiplications.

Case	Degree	Mult. by a root	Recursion	Complexity
$\mathbb{C}$	$O(N/\log N)$	expensive	$O(\log N)$	$N \log N \log\log N \cdots 2^{O(\log^* N)}$
$\mathbb{Z}/(2^e+1)\mathbb{Z}$	$O(\sqrt{N})$	cheap	$O(\sqrt{N})$	$N \log N \log\log N$
$\mathbb{C}[x]/(x^P+1)$	$O(N/\log^2 N)$	it depends	$O(\log^2 N)$	$N \log N\, 2^{O(\log^* N)}$

In 2014, Harvey, Lecerf and Van Der Hœven proved that the exact complexity is

$$N\ \log N\ 16^{\log^* N}.$$

With Bluestein's Chirp transform, they reach unconditionally $N\ \log N\ 8^{\log^* N}$.

By using a conjecture on Mersenne primes, they even have $N\ \log N\ 4^{\log^* N}$.

Number-theoretic transform

1. N: # bits of the integers that we multiply
2. $n - 1$: degree of the polynomials A and B used to represent a and b
3. k: # bits used to encode the coefficients of A and B: $a = A(2^k)$ and $b = B(2^k)$

Instead of computing FFT over $\mathbb{C}$, we can choose $\mathcal{R} = \mathbb{Z}/q\mathbb{Z}$.
The prime q must satisy $2n \mid q - 1$ (there exists a $2n$-th principal root of unity).
A choice of q such that $\log q = O(\log N)$ is optimal.

A Fürer-like number theoretic transform

- q is chosen such that $q = r^P + 1$: this is a generalized Fermat prime.
  Conjecturally, there exists r such that $r < P \cdot (\log P)^2 \Rightarrow \log_2 q \approx P \log P$.
- There exists ρ a $2n$-th root of unity in $\mathbb{Z}/q\mathbb{Z}$ such that $\rho^{n/P} = r$.

A Fürer-like number theoretic transform

- q is chosen such that $q = r^P + 1$: this is a generalized Fermat prime.
  Conjecturally, there exists r such that $r < P \cdot (\log P)^2 \Rightarrow$ $\log_2 q \approx P \log P$.
- There exists ρ a $2n$-th root of unity in $\mathbb{Z}/q\mathbb{Z}$ such that $\rho^{n/P} = r$.
- $x \in \mathbb{Z}/q\mathbb{Z}$ and $y \in \mathbb{Z}/q\mathbb{Z}$ are represented by polynomials over $\mathbb{Z}$:
  $$X(r) = x_0 + x_1 \cdot r + x_2 \cdot r^2 \cdots x_{P-1} \cdot r^{P-1}$$
  and
  $$Y(r) = y_0 + y_1 \cdot r + y_2 \cdot r^2 \cdots y_{P-1} \cdot r^{P-1}.$$
- Computation of $x \cdot y$: we choose $Q = O(\log \log P)$ and we represent x and y in radix r^Q.
  $\Rightarrow$ We get $\tilde{X}$ and $\tilde{Y}$ polynomials modulo $X^{P/Q} + 1$ with coefficients $\leq r^Q$.
  $\Rightarrow$ We compute a P/Q-points FFT.

Some estimations

	Schönhage-Strassen algorithm		
bitsize	nb. mult.	mult. bitsize	estimated time (s)
2^{30}	2^{16}	$\approx 2^{16}$	9.96
2^{36}	2^{18}	$\approx 2^{18}$	$2.60 \cdot 10^2$
2^{40}	2^{21}	$\approx 2^{21}$	$2.36 \cdot 10^4$
2^{46}	2^{24}	$\approx 2^{24}$	$2.17 \cdot 10^6$
2^{50}	2^{26}	$\approx 2^{26}$	$4.10 \cdot 10^7$
2^{56}	2^{29}	$\approx 2^{29}$	$2.94 \cdot 10^9$

	Generalized Fermat primes			
bitsize	nb. mult.	prime	KS. bitsize	estimated time (s)
2^{30}	$2^{24} \cdot 13$	$562^{32} + 1$	800	$3.57 \cdot 10$
2^{36}	$2^{30} \cdot 16$	$562^{32} + 1$	800	$3.35 \cdot 10^3$
2^{40}	$2^{34} \cdot 19$	$562^{32} + 1$	800	$6.26 \cdot 10^4$
2^{46}	$2^{40} \cdot 22$	$884^{32} + 1$	800	$4.64 \cdot 10^6$
2^{50}	$2^{44} \cdot 25$	$884^{32} + 1$	800	$7.91 \cdot 10^7$
2^{56}	$2^{50} \cdot 28$	$884^{32} + 1$	800	$5.67 \cdot 10^9$

- nb. mult.: $2n \cdot (3 \cdot \lceil \log_{2P} 2n \rceil + 1)$.

Conclusion

Avoiding the padding due to a modular ring and the Kronecker substitution improves on the complexity of the algorithm: we reach $N \log N \cdot 4^{\log^* N}$.

The complexity is conjectural: related to "Hypothesis H" and lower bounds on r such that $P(r)$ is prime for a polynomial P.

In practice, we do not expect this algorithm to improve on Schönhage-Strassen for sizes $\leq 2^{40}$ bits.

It is possible to improve the arithmetic in $\mathbb{Z}/q\mathbb{Z}$ by choosing $q = b^P + 1$ with a special b (sparse?): a lot of generalized Fermat primes.