Multiplication algorithms

Svyatoslav Covanov

Team CARAMBA

February 25, 2018

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 $n-1$.

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

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

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

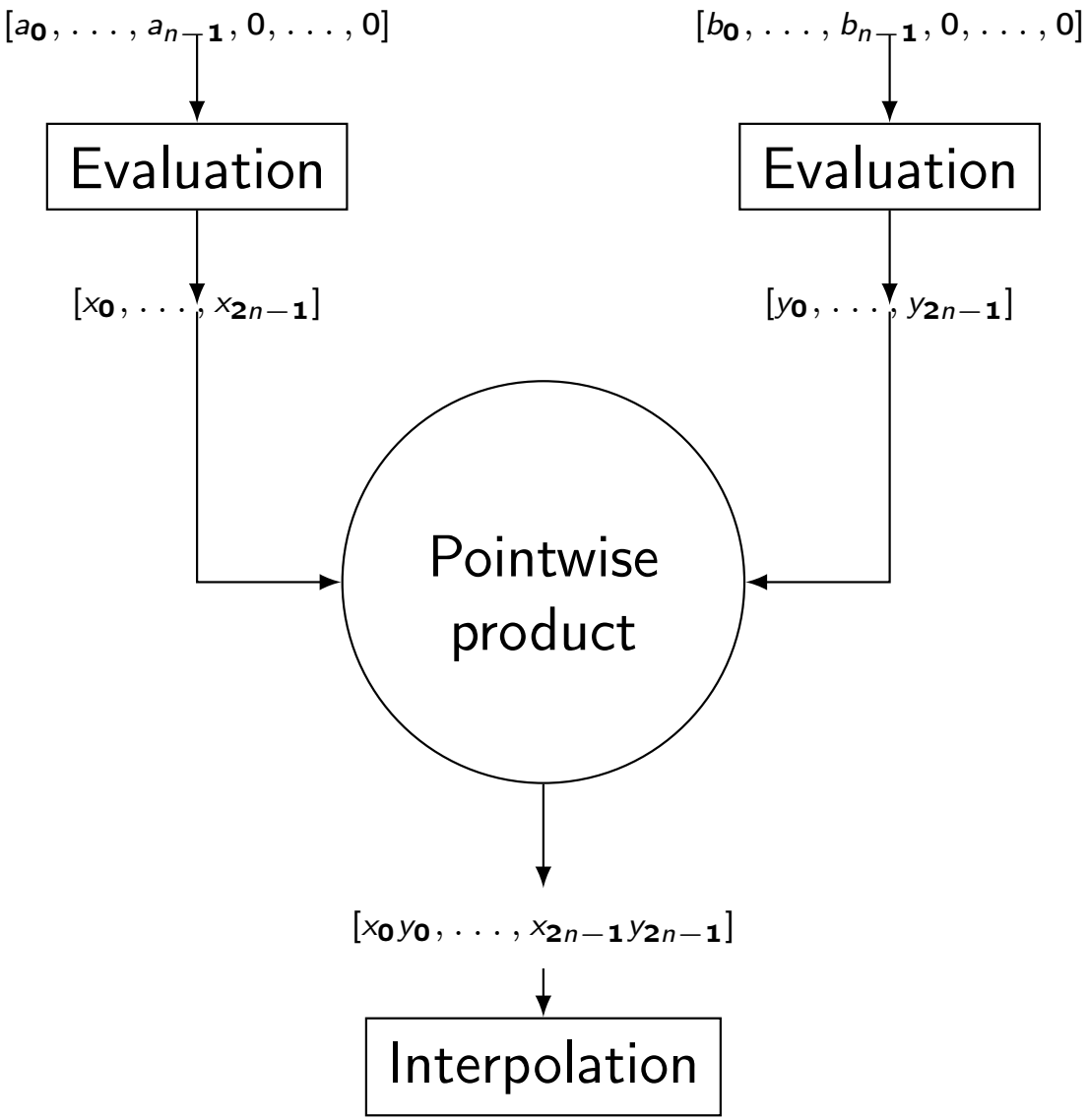
- We choose $2n-1$ distinct points w_i of $\mathcal{R}$.
- Computation of $A(w_i)$ and $B(w_i)$: equivalent to the product

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

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

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

Evaluation-Interpolation scheme



Discrete Fourier Transform (DFT)

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

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

For $A \in \mathcal{R}[x]$,

$$\begin{pmatrix} A(1) \\ A(\omega) \\ \vdots \\ A(\omega^{2n-1}) \end{pmatrix}$$

is the discrete Fourier transform of A.

Definition

Let $\mathcal{R}$ be a **ring** containing a $2n$-th root of unity ω. The root ω is said to be a $2n$-th principal root of unity if

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

Weaker notion: primitive root of unity if

$$\forall i \in [1, 2n-1], \omega^i \neq 1.$$

Primitive and principal is the same thing on a field.

If $\mathcal{R}$ contains a $2n$-th principal root of unity ω, then

$$M_{2n}(\omega)^{-1} = \frac{1}{2n} M_{2n}(\omega^{-1}).$$

$\Rightarrow$ An efficient algorithm for the evaluation gives an efficient algorithm for the interpolation...

FFT(A, ω, $2n$)

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

Choice of the ring

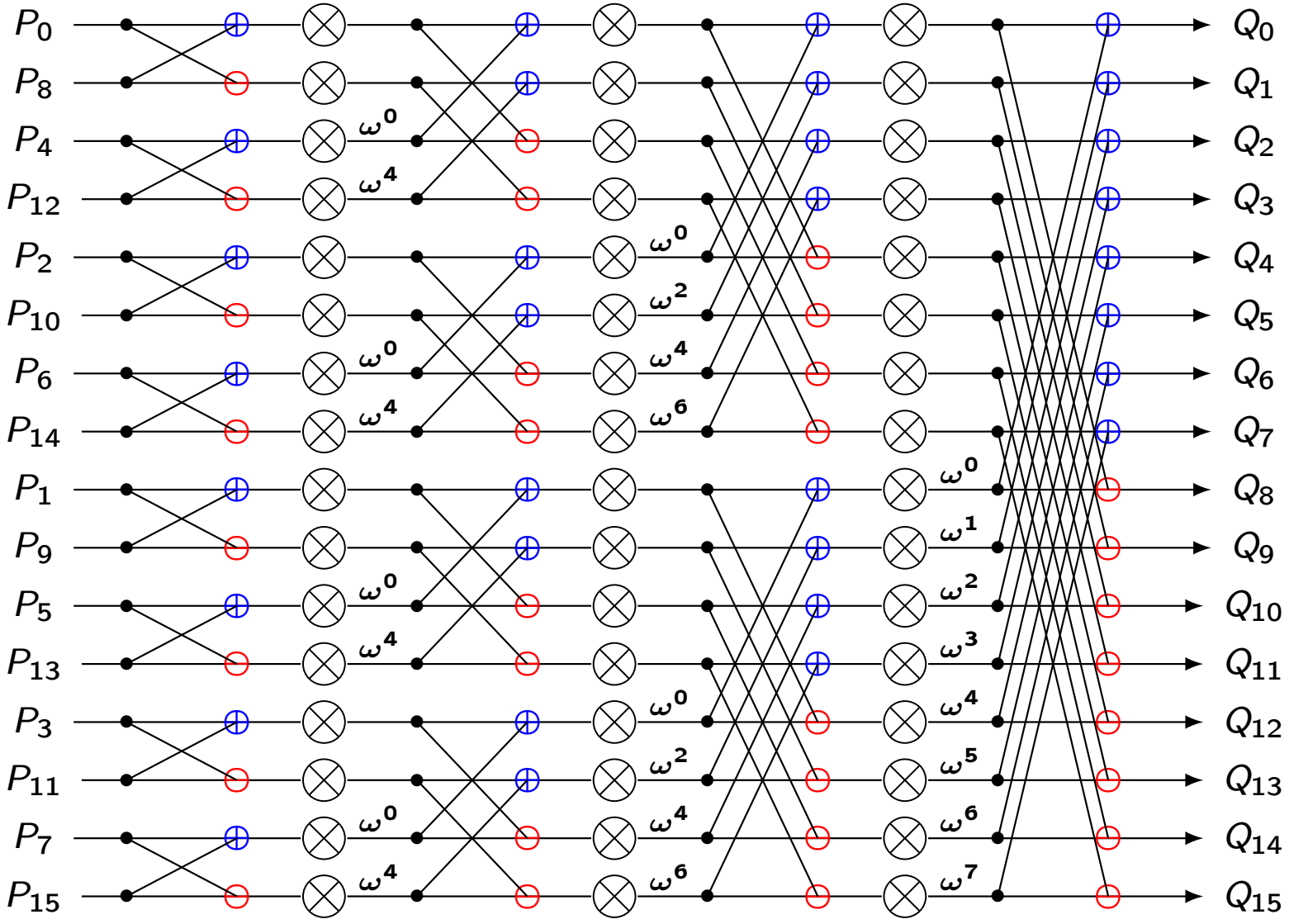
1. N: # bits of the integers that we multiply
2. $n-1$: degree of the polynomials A and B used to represent a and b
3. k: # bits used to encode the coefficients of A and B: $a = A(2^k)$, $b = B(2^k)$ and $n \cdot k = N$.

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.



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

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...

Cooley-Tukey

- $2n$-point DFT computed with radix-2 FFT:

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

- $2n$-point DFT computed with radix-4 FFT:

$$4 \cdot \mathrm{DFT}(n/2) + \text{Twiddle factors} + n/2 \cdot \mathrm{DFT}(4).$$

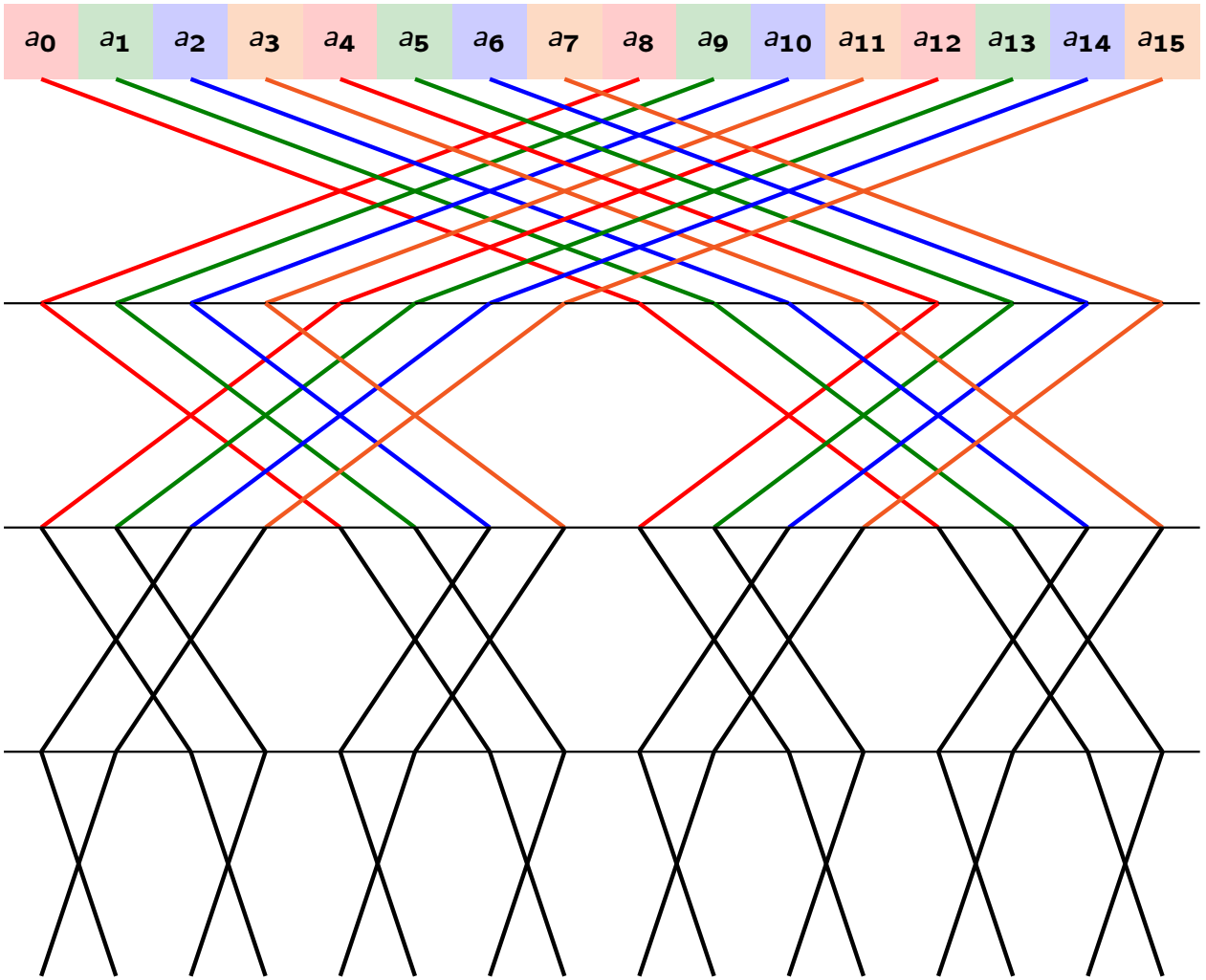
- $2n$-point DFT computed with radix-$2m$ FFT ($2m$ divides $2n$):

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

$$\mathrm{DFT}(mn) = m \cdot \mathrm{DFT}(n) + \text{Twiddle factors} + n \cdot \mathrm{DFT}(m).$$

Radix-4 Cooley-Tukey

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

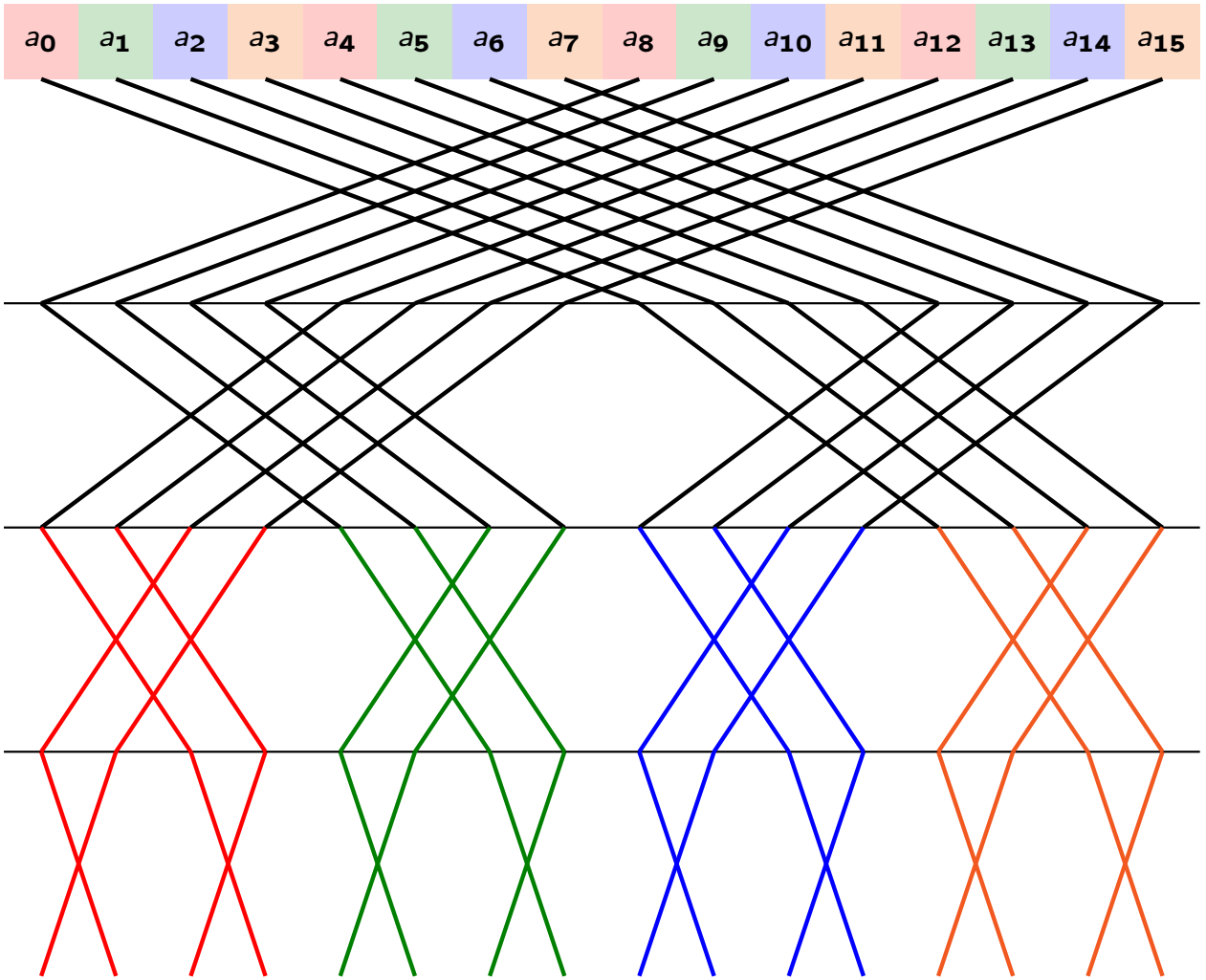


Matrix point of view:

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

Radix-4 Cooley-Tukey

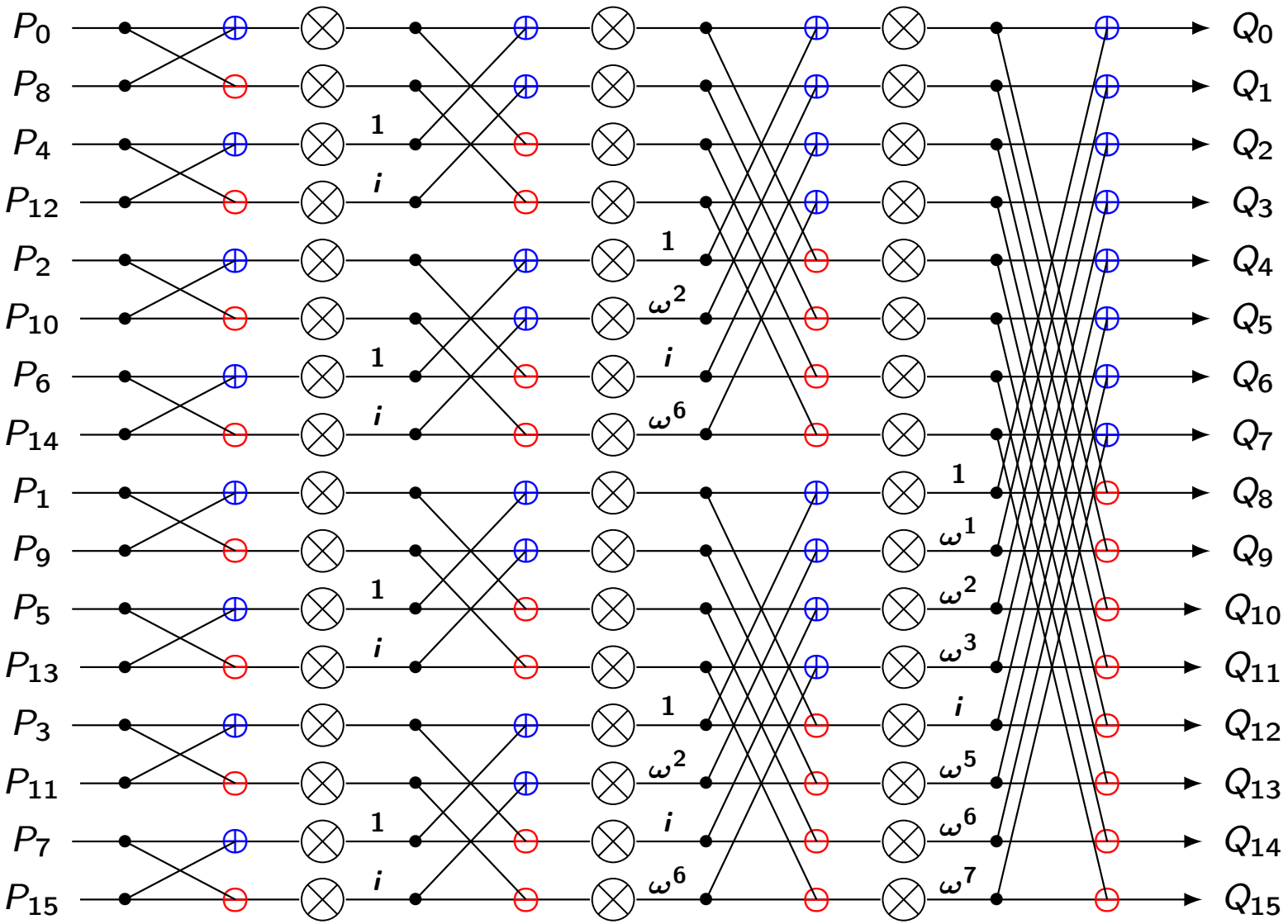
$4 \cdot \mathrm{DFT}(4)$:



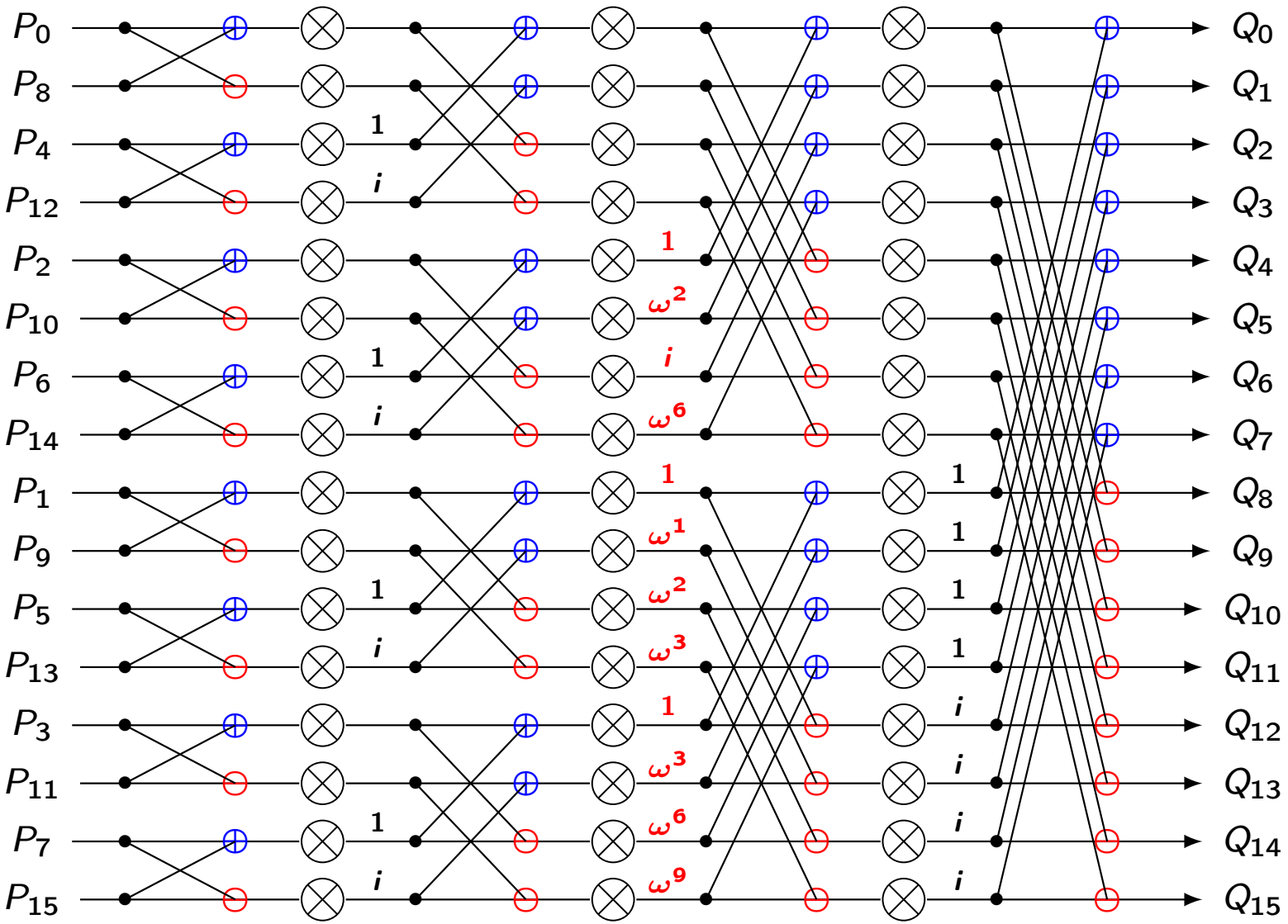
Matrix point of view:

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

An example in Complex Field: radix-2 FFT



An example in Complex Field: radix-4 FFT



Fürer's algorithm

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

Fürer's algorithm

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

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.

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.

We cut the N-bit integers in pieces of size $k \approx \frac{1}{2}\log q$:

$$\log n + 2k \leq \log q.$$

$\Rightarrow \mathcal{M}_{\mathcal{R}}^{KS} \geq \mathcal{M}_{2k}$.

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 $r < P \cdot (\log P)^2 \Rightarrow \log_2 q \approx P \log P$.
- Let ρ be 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 $r < P \cdot (\log P)^2 \Rightarrow \log_2 q \approx P \log P$.
- Let ρ be a $2n$-th root of unity in $\mathbb{Z}/q\mathbb{Z}$ such that $\rho^{n/P} = r$.

Working in radix r is like working with "polynomials" of degree P whose coefficients are bounded by r:

$$\mathcal{M}_{\mathcal{R}} \leq \mathcal{M}_{\mathbb{Z}_P[X]}.$$

Steps of the algorithm

- Find a prime $q = r^{\log N} + 1$ sufficiently large for multiplying integers of size N.
- Cut the integers a and b into pieces of size $k = O(\log N \log\log N)$, that are the coefficients of A and B.
- Represent the pieces as elements of $\mathbb{Z}/q\mathbb{Z}$ in radix r.
- Compute the FFT, the componentwise product, the inverse FFT.
- Switch from radix r to the regular representation of elements of $\mathbb{Z}/q\mathbb{Z}$.
- Transform the polynomial $C = A \cdot B$ into an integer c by evaluating it at 2^k.

Comparison of complexities

Using generalized Fermat primes we get the following data:

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	$O(\log^2 N)$	$N\ \log N\ 16^{\log^* N}$
Mersenne prime and Bluestein	$O(N/(\log N^{\log\log N}))$	it depends	$O(\log N^{\log\log N})$	$N\ \log N\ 4^{\log^* N}$
$\mathbb{Z}/(r^P+1)\mathbb{Z}$	$O(N/(\log N\ \log\log N))$	it depends	$O(\log N\ \log\log N)$	$N\ \log N\ 4^{\log^* N}$

Short product example

How to multiply two polynomials $A = a_0 + a_1 X + a_2 X^2$ and $B = b_0 + b_1 X + b_2 X^2$ modulo X^3 ?

Short product example

How to multiply two polynomials $A = a_0 + a_1X + a_2X^2$ and $B = b_0 + b_1X + b_2X^2$ modulo X^3 ?

$A \cdot B = a_0b_0 + (a_1b_0 + a_0b_1)X + (a_2b_0 + a_1b_1 + a_0b_2)X^2$

1. **Naive multiplication:**
   - $\pi_0 = \boldsymbol{a_0b_0}$, $\pi_1 = \boldsymbol{a_1b_0}$, $\pi_2 = \boldsymbol{a_0b_1}$, $\pi_3 = \boldsymbol{a_1b_1}$, $\pi_4 = \boldsymbol{a_2b_1}$ and $\pi_5 = \boldsymbol{a_1b_2}$.
   - We have $A \cdot B = \pi_0 + (\pi_1 + \pi_2)X + (\pi_3 + \pi_4 + \pi_5)X^2$.

2. **Optimal formula**:
   - $\pi_0 = \boldsymbol{a_0b_0}$, $\pi_1 = \boldsymbol{a_1b_1}$, $\pi_2 = \boldsymbol{a_2b_2}$, $\pi_3 = \boldsymbol{(a_0 + a_1)(b_0 + b_1)}$ and $\pi_4 = \boldsymbol{(a_0 + a_2)(b_0 + b_2)}$.
   - We have $A \cdot B = \pi_0 + (\pi_3 - \pi_0 - \pi_1)X + (\pi_1 + \pi_4 - \pi_0 - \pi_2)X^2$.

   The **bilinear rank** is equal to 5.

Matrix formalism

$$c_0 = \begin{pmatrix} a_0 & a_1 & a_2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = a_0 b_0$$

$$c_1 = \begin{pmatrix} a_0 & a_1 & a_2 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = a_1 b_0 + a_0 b_1$$

$$c_2 = \begin{pmatrix} a_0 & a_1 & a_2 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = a_2 b_0 + a_1 b_1 + a_0 b_2$$

Matrix representation of formulae:

$$\underbrace{\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}}_{a_0b_0},\ \underbrace{\begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}}_{a_1b_1}\ \underbrace{\begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix}}_{a_2b_2}\ \underbrace{\begin{pmatrix}1&1&0\\1&1&0\\0&0&0\end{pmatrix}}_{(a_0+a_1)(b_0+b_1)}\ \underbrace{\begin{pmatrix}1&0&1\\0&0&0\\1&0&1\end{pmatrix}}_{(a_0+a_2)(b_0+b_2)}$$

Decomposition with rank-one matrices:

$$\underbrace{\begin{pmatrix}0&1&0\\1&0&0\\0&0&0\end{pmatrix}}_{c_1} = \begin{pmatrix}1&1&0\\1&1&0\\0&0&0\end{pmatrix} - \begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix} - \begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}$$

$$\underbrace{\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}}_{c_2} = \begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix} + \begin{pmatrix}1&0&1\\0&0&0\\1&0&1\end{pmatrix} - \begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix} - \begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix}$$

Problem to be solved

Let K be a finite field. Let T (the target) be a subspace of $\mathcal{M}_{n,n}(K)$ of dimension ℓ.
Let $\mathcal{G}$ be the set of matrices of rank one in $\mathcal{M}_{n,n}(K)$.
Problem to be solved: Find all free families $\mathcal{F} \subset \mathcal{G}$ of minimal size such that $T \subset \mathrm{Span}(\mathcal{F})$.

Definition

Let $r \geq 0$ and $n \geq 0$. We denote by $\mathscr{S}_r$ the set of all subspaces $V \subset \mathcal{M}_{n,n}$ such that there exists $\{g_0, \ldots, g_{r-1}\}$ a free family of $\mathcal{G}$ satisfying $V = \mathrm{Span}(g_0, \ldots, g_{r-1})$.

Restatement:

1. Find minimal r such that $\mathscr{S}_r$ contains subspaces V s.t. $T \subset V$;
2. Enumerate the bases of rank-one matrices for subspaces $V \in \mathscr{S}_r$ s.t. $T \subset V$.

Short Product Example

For the short product modulo X^3, we have

$$T = \mathrm{Span}\left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \right),$$

$K = \mathbb{F}_2$, $\ell = n = 3$ and $r = 5$.

Naive algorithm

Enumerate all subspaces $V \in \mathscr{S}_r$ and keep those which contain T.

Complexity: $\#\mathscr{S}_r \leq \binom{\#\mathcal{G}}{r}$. For $\ell = 3$ and $K = \mathbb{F}_2$, we have $\#\mathscr{S}_5 = 157,535 \ll 1,906,884 = \binom{49}{5}$.

Incomplete basis improvement

Theorem

Let T be a subspace of dimension ℓ of $\mathcal{M}_{n,n}$, let $r \geq \ell$ be an integer. For any $V \in \mathscr{S}_r$, such that $T \subset V$, there exists $W \in \mathscr{S}_{r-\ell}$ such that $T \oplus W = V$.

Incomplete basis improvement: compute all the vector spaces $V = T + W$ for $W \in \mathscr{S}_{r-\ell}$ and keep those which are in $\mathscr{S}_r$.

Complexity: $\#\mathscr{S}_{r-\ell} \leq \binom{\#\mathcal{G}}{r-\ell}$. For $\ell = 3$, $\#\mathscr{S}_2 = 980 \ll 157,535$.

Automorphisms

We consider the action of pairs (P, Q) (P and Q in GL_n) on $M \in \mathcal{M}_{n,n}$:

$$M \circ (P, Q) = P^T \cdot M \cdot Q.$$

Let $\mathrm{Stab}(T)$ be the group of (P, Q) such that

$$\forall M \in T,\ M \circ (P, Q) \in T.$$

The group action can be used with all the previous algorithms: we compute $\mathscr{S}_r / \mathrm{Stab}(T)$ or $\mathscr{S}_{r-\ell} / \mathrm{Stab}(T)$.

Intermediate strategies

The two previous strategies are two extreme cases of a mixed strategy.

Let $k \geq 0$. For all $W \in \mathscr{S}_{r-\ell+k}$ such that $\dim(W \cap T) = k$, compute the vector spaces $V = T + W$ and keep those which are in $\mathscr{S}_r$.

Notation

For an integer $d \geq 0$ and a subspace $F \subset \mathcal{M}_{n,n}$, we denote by $\mathcal{C}_d(F)$ the set of subspaces $W \in \mathscr{S}_d$ such that $F \subset W$.

Intermediate strategies

The two previous strategies are two extreme cases of a mixed strategy.

Let $k \geq 0$. For all $W \in \mathscr{S}_{r-\ell+k}$ such that $\dim(W \cap T) = k$, compute the vector spaces $V = T + W$ and keep those which are in $\mathscr{S}_r$.

Notation

For an integer $d \geq 0$ and a subspace $F \subset \mathcal{M}_{n,n}$, we denote by $\mathcal{C}_d(F)$ the set of subspaces $W \in \mathscr{S}_d$ such that $F \subset W$.

- Naive algorithm: compute $\mathcal{C}_r(\emptyset)$;
- Incomplete basis improvement: compute $\mathcal{C}_{r-\ell}(\emptyset)$;
- General case: given g subspaces $F_0, \ldots, F_{g-1}$ of T of dimensions $k_0, \ldots, k_{g-1}$, compute $\mathcal{C}_{r-\ell+k_0}(F_0), \ldots, \mathcal{C}_{r-\ell+k_{g-1}}(F_{g-1})$.

Example for the short product

Notation

For an integer $\ell \geq 0$, we denote by T_ℓ the subspace of $\mathcal{M}_{\ell,\ell}$ such that $T_\ell = \mathsf{Span}(c_0, \ldots, c_{\ell-1})$, where the c_i's are the coefficients of the short product modulo X^ℓ.

Theorem

Let $V \in \mathscr{S}_r$ containing T_ℓ. There exist $\sigma \in \mathsf{Stab}(T_\ell)$ and $W \in \mathcal{C}_{r-\ell+2}(\mathsf{Span}(c_{\ell-1}, c_{\ell-2}))$ such that $V = T_\ell + W \circ \sigma$.

For $\ell = 3$:

approach	covering set	cardinality
Naive approach	$\mathcal{C}_5(\emptyset)$	$157,535$
BDEZ '12	$\mathcal{C}_2(\emptyset)$	980
New approach	$\mathcal{C}_4(\mathsf{Span}(c_2, c_1))$	12

Statement of the problem

Algorithmic problem: enumerate a set of the form

$$\mathcal{C}_{r-\ell+k}(\mathrm{Span}(\phi_0,\ldots,\phi_{k-1}))/\,\mathrm{Stab}(T)\cap\mathrm{Stab}(\mathrm{Span}(\phi_0,\ldots,\phi_{k-1})),$$

where T is a subspace of $\mathcal{M}_{n,n}$ of dimension ℓ and the ϕ_i's are elements of T.

Steps:

- precompute $\mathscr{S}_{r-\ell+k}/\,\mathrm{GL}_n\times GL_n$,
- deduce $\mathcal{C}_{r-\ell+k}(\mathrm{Span}(\phi_0,\ldots,\phi_{k-1}))/\,\mathrm{Stab}(\{\phi_0,\ldots,\phi_{k-1}\})$ and
- apply the the quotient $\mathrm{Stab}(\{\phi_0,\ldots,\phi_{k-1}\})/\,\mathrm{Stab}(T)\cap\mathrm{Stab}(\{\phi_0,\ldots,\phi_{k-1}\})$.

Remark: we obtain $\mathcal{C}_{r-\ell+k}(\mathrm{Span}(\phi_0,\ldots,\phi_{k-1}))/\,\mathrm{Stab}(T)\cap\mathrm{Stab}(\{\phi_0,\ldots,\phi_{k-1}\})$, slightly larger than the targeted set.

Technical aspects

How to compute
$\mathcal{C}_{r-\ell+k}(\mathrm{Span}(\phi_0, \ldots, \phi_{k-1}))/\mathrm{Stab}(\{\phi_0, \ldots, \phi_{k-1}\})$?

Algorithm:

- for all $W \in \mathscr{S}_{r-\ell+k}/\mathrm{GL}_n \times GL_n$, enumerate all the tuples $(\psi_0, \ldots, \psi_{k-1})$ such that $\psi_i \in W$ and $\mathrm{rk}(\psi_i) = \mathrm{rk}(\phi_i)$;
- compute $\sigma \in \mathrm{GL}_n \times \mathrm{GL}_n$ such that $\{\psi_0, \ldots, \psi_{k-1}\} \circ \sigma = \{\phi_0, \ldots, \phi_{k-1}\}$ (computational group theory, Weierstrass-Kronecker theory...).

Covering sets on examples

Covering set for the short product modulo X^5:

- $\mathcal{C}_8(\mathrm{Span}(c_3, c_4))$.

Covering sets for the product of matrices 3×2 by 2×3 (the coefficients are denoted by $c_{i,j}$):

- $\mathcal{C}_7(\mathrm{Span}(c_{0,0} + c_{1,1} + c_{2,2}))$;
- $\mathcal{C}_8(\mathrm{Span}(c_{0,0} + c_{1,1}, c_{0,0} + c_{2,2}))$;
- $\mathcal{C}_8(\mathrm{Span}(c_{0,0} + c_{1,1}, c_{0,1} + c_{2,2}))$;
- $\mathcal{C}_8(\mathrm{Span}(c_{0,0} + c_{1,1}, c_{2,2}))$;
- $\mathcal{C}_9(\mathrm{Span}(c_{0,0}, c_{1,1}, c_{2,2}))$.

Timings

We have timings on a single core 3.3 GHz Intel Core i5-4590.

product	time (s)	nb. of solutions
ShortProduct_4	3.0	$1,440$
ShortProduct_5	$2.4 \cdot 10^3$	$146,944$

Table: Computation of decompositions of the short product.

product	time (s)	nb. of solutions
2×3 by 3×2	$4.1 \cdot 10^6$	$1,096,452$
3×2 by 2×3	$3.0 \cdot 10^6$	$7,056$

Table: Computation of decompositions of the matrix product.

Conclusion

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

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.

We obtain interesting speed-up for symmetric bilinear maps such as matrix product and short product compared to implementations of BDEZ.

How to push computations further: possible to decompose matrix product 3×3 by 3×3?