Computational Algebra. Lecture 5 Multipoint evaluation and interpolation

M.Eulàlia Montoro

16 de marzo de 2022

Today

- **Multipoint Evaluation** Given $n = 2^k$, $f \in R[x]$ of degree$< n$, and $u_0, \ldots, u_{n-1} \in R$, compute

$$\chi(f) = (f(u_0), \ldots, f(u_{n-1}))$$

- **Interpolation** Given $n = 2^k$, $u_0, \ldots, u_{n-1} \in R$ such that $u_i - u_j$ is a unit, and $v_0, \ldots, v_{n-1}$, compute $f \in R[x]$ of degree$< n$ with

$$\chi(f) = (f(u_0), \ldots, f(u_{n-1})) = (v_0, \ldots, v_{n-1})$$

Recall

- Evaluating a polynomial $f \in \mathbb{F}[x]$ of degree$< n$ at n distinct points $u_1, \dots, u_n \in F$ can be performed with $O(n^2)$ operations in $\mathbb{F}$.
- Computing an interpolating polynomial at these points can be performed with $O(n^2)$ operations in $\mathbb{F}$.

-If R supports the FFT and $u_i = \omega^i :\longrightarrow O(n \log n)$.
-Today: A similar bound.

Binary Tree

Definition

Let R be a commutative ring, $r = 2^k$ and $m_0, \ldots, m_{r-1} \in R[x]$ monic and non constant polynomials. We define the **binary tree of products** $M_{i,j}$ **as**

$$M_{0,j} = m_j,\ j = 0, \ldots, 2^k - 1$$

$$M_{i+1,j} = M_{i,2j} \cdot M_{i,2j+1},\ j = 0, \ldots, 2^{k-i} - 1$$

Binary Tree

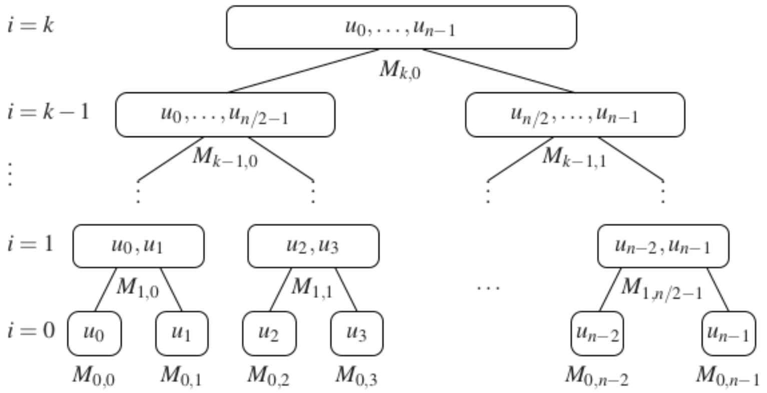

FIGURE 10.1: Subproduct tree for the multipoint evaluation algorithm.

Binary Tree

Input: Monic nonconstant moduli $m_0, \ldots, m_{r-1} \in R[x]$, $r = 2^k$, $k \in \mathbb{N}$.

Output: $M_{i,j}$, $0 \leq i \leq k, 0 \leq j \leq 2^{k-i}$

```
1. for $j = 0, \ldots, r-1$ do $M_{0,j} \longleftarrow m_j$
2. for $i = 1, \ldots, k$ do
3. for $j = 0, \ldots, 2^{k-i} - 1$ do $M_{i,j} \longleftarrow M_{i-1,2j} \cdot M_{i-1,2j+1}$
```

Lemma

The above algorithm correctly computes all subproducts $M_{i,j} \in R[x]$ and takes $M(n) \log r$ operations ($n = \sum \deg m_i$).

FSR with precomputation

The computation of $M_{i,j}$ can be regarded as a precomputation stage for the fast multipoint evaluation algorithm that we are going to present now:

FAST SIMULTANEOUS REDUCTION WITH PRECOMPUTATION

FSR with precomputation

Input: Monic nonconstant moduli $m_0, \dots, m_{r-1} \in R[x]$, with R a commutative ring with 1, $r = 2^k$, $k \in \mathbb{N}, f \in R[x]$ of degree less than $n = \sum_{i=0}^{r-1} \deg(m_i)$, and the polynomials $M_{i,j}$ from the binary tree of products.

Output: f rem m_0, ..., f rem m_{r-1}.

```
1. If $r = 1$ then return $f$
2. $r_0 \leftarrow f$ rem $M_{k-1,0}$, $r_1 \leftarrow f$ rem $M_{k-1,1}$
3. call the algorithm recursively to compute
   $r_0$ rem $m_0, \dots, r_0$ rem $m_{r/2-1}$
4. call the algorithm recursively to compute
   $r_1$ rem $m_{r/2}, \dots, r_1$ rem $m_{r-1}$.
5. return $r_0$ rem $m_0, \dots, r_0$ rem $m_{r/2-1}$, $r_1$ rem $m_{r/2}, \dots, r_1$ rem $m_{r-1}$
```

FSR with precomputation

Theorem

The algorithm works correctly and takes at most $O(n)\log r$ operations in R

Demostración.

We prove the correctness by induction on k.

- If $k = 0$ we only have m_0 and since $\deg f \leq \deg m_0$, then the algorithm returns $f \; rem \; m_0 = f$.

□

Evaluation. Correctness

Demostración.

Assume true for $k-1$. Now $M_{k-1,0} = m_0 \cdot \ldots \cdot m_{\frac{r}{2}-1}$ and $M_{k-1,1} = m_{\frac{r}{2}} \cdot \ldots \cdot m_{r-1}$.

- $r_0 = f \; rem \; m_0 \cdot \ldots \cdot m_{\frac{r}{2}-1}$ and $r_1 = f \; rem \; m_{\frac{r}{2}} \cdot \ldots \cdot m_{r-1}$
- And the correctness follow from $(f \; mod \; ka) \; mod \; a = f \; mod \; a$

$\square$

Linear Combination

The following divide-and-conquer algorithm is the core of the fast interpolation algorithm

Linear Combination

Consider the following algorithm:

Input: Monic nonconstant moduli $m_0, \dots, m_{r-1} \in R[x]$, with R a commutative ring with 1, $r = 2^k$, $k \in \mathbb{N}$, $c_0, \dots, c_{r-1} \in R[x]$ with $\deg(c_i) < \deg(m_i)$ for all i, and the polynomials $M_{i,j}$ from the binary tree of products.

Output: The polynomial $f = \sum_{i=0}^{r-1} c_i \frac{m}{m_i} \in R[x]$, where $m = m_0 \cdots m_{r-1}$.

1. If $r = 1$ then return c_0
2. call the algorithm recursively to compute $r_0 = \sum_{i=0}^{r/2-1} c_i \frac{M_{k-1,0}}{m_i}$
3. call the algorithm recursively to compute $r_1 = \sum_{i=r/2+1}^{r-1} c_i \frac{M_{k-1,1}}{m_i}$
4. return $M_{k-1,1} r_0 + M_{k-1,0} r_1$.

Linear Combination(proof)

- $k = 0$, then $r = 1$ and the algorithm returns c_0
- Assume true for $k-1$, then

$$M_{k-1,1}r_0 + M_{k-1,0}r_1 =$$

$$= \sum_{i=0}^{r/2-1} c_i \frac{M_{k-1,0}M_{k-1,1}}{m_i} + \sum_{i=r/2+1}^{r-1} c_i \frac{M_{k-1,1}M_{k-1,0}}{m_i} = \sum_{i=0}^{r-1} c_i \frac{M_{k,0}}{m_i}$$

- **Cost is** $O(M(n)\log r)$.

Homework

You are to trace and implement the integer analogues of the two algorithms seen today in class. To do this, let $m_0 = 23$, $m_1 = 24$, $m_2 = 25$, and $m_3 = 29$.

1. Check that the moduli are pairwise coprime.
2. Compute the binary tree of products.
3. Compute $300000 \bmod m_i$ for $0 \leq i < 3$, using your implementation.
4. Let $c_0 = 5$, $c_1 = 2$, $c_2 = 1$ and $c_3 = 22$. Compute $f \in \mathbb{Z}$ as the outcome of the linear combination algorithm, using your implementation.