Chapter 10 Fast polynomial evaluation and interpolation

1. P1. (Multipoint evaluation) Given $n = 2^k$ for some $k \in \mathbb{N}$, $f \in R[x]$ of degree less than n, and $u_0, \ldots, u_{n-1} \in R$, compute

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

2. P2. (Interpolation) Given $n = 2^k$ for some $k \in \mathbb{N}$, $u_0, \ldots, u_{n-1} \in R$ such that $u_i - u_j$ is a unit for $i \neq j$, and $v_0, \ldots, v_{n-1} \in R$, compute $f \in R[x]$ of degree less than n with

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

Main tool: Divide-and-conquer

10.1 Fast multipoint evaluation

We let $m_i = x - u_i$, for $0 \leq i \leq k$ and $0 \leq j < 2^{k-i}$

$$M_{i,j} = m_{j \cdot 2^i} \cdot m_{j \cdot 2^i + 1} \cdots m_{j \cdot 2^i + (2^i - 1)} = \prod_{0 \leq l < 2^i} m_{j \cdot 2^i + l}$$

Algorithm 10.3 Building up the subproduct tree

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

Output: The polynomials $M_{i,j}$ for $0 \leq i \leq k$ and $0 \leq j < 2^{k-i}$

1. for $0 \leq j < r$ do $M_{0,j} \longleftarrow m_j$

2. for $1 \leq i \leq k$ do
   for $0 \leq j < 2^{k-i}$ do $M_{i,j} \longleftarrow M_{i-1,2j} \cdot M_{i-1,2j+1}$

Algorithm 10.5 Going down the subproduct tree

Input: $f \in R[x]$ of degree less than $n = 2^k$, $k \in \mathbb{N}$. $u_0, \ldots, u_{n-1} \in R$ and $M_{i,j}$.

Output: $f(u_0), \ldots, f(u_{n-1}) \in R$.

1. if $n = 1$ then return f.

2. $r_0 \longleftarrow f \,\mathrm{rem}\, M_{k-1,0}$, $r_1 \longleftarrow f \,\mathrm{rem}\, M_{k-1,1}$

3. call the algorithm recursively to compute $r_0(u_0), \ldots, r_0(u_{n/2-1})$

4. call the algorithm recursively to compute $r_1(u_{n/2}), \ldots, r_1(u_{n-1})$

5. return $r_0(u_0), \ldots, r_0(u_{n/2-1})$, $r_1(u_{n/2}), \ldots, r_1(u_{n-1})$

Theorem 1. *Algorithm 10.5 works correctly and takes at most $O(M(n)\log n)$ operations in R.*

Proof: Let $q_0 = f\,\mathsf{quo}M_{k-1,0}$ and $q_1 = f\,\mathsf{quo}M_{k-1,1}$. Then

$$f(u_i) = \begin{cases} q_0(u_i)M_{k-1,0}(u_i) + r_0(u_i) & = & r_0(u_i)(0 \le i < \frac{n}{2}) \\ q_1(u_i)M_{k-1,1}(u_i) + r_1(u_i) & = & r_1(u_i)(\frac{n}{2} \le i < n) \end{cases}$$

$$T(2^k) = 2T(2^{k-1}) + 2D(2^{k-1}) \le 2k \cdot D(2^{k-1}) \le D(n)\log n$$

Corollary 1. *Evaluation of a polynomial in $R[x]$ of degree less than n at n points in R can be performed using $O(M(n)\log n)$ operations in R.*

10.2 Fast interpolation

Algorithm 10.9 Linear combination for linear moduli

Input: $u_0, \ldots, u_{n-1}, c_0, \ldots, c_{n-1} \in R$, where $n = 2^k$ for some $k \in \mathbb{N}$, and the polynomials $M_{i,j}$.

Output: $\sum_{0 \le i < n} c_i \frac{m}{x-u_i} \in R[x]$, where $m = (x-u_0)\cdots(x-u_{n-1})$.

1. if $n = 1$ then return c_0
2. call the algorithm recursively to compute $r_0 = \sum_{0 \le i < n/2} c_i \frac{M_{k-1,0}}{x-u_i} \in R[x]$
3. call the algorithm recursively to compute $r_1 = \sum_{n/2 \le i < n} c_i \frac{M_{k-1,1}}{x-u_i} \in R[x]$
4. return $M_{k-1,1} r_0 + M_{k-1,0} r_1$

Algorithm 10.11 Fast interpolation

Input: $u_0, \ldots, u_{n-1} \in R$ such that $u_i - u_j$ is a unit for $i \neq j$, and $v_0, \ldots, v_{n-1} \in R$, where $n = 2^k$ for some $k \in \mathbb{N}$.

Output: The unique polynomial $f \in R[x]$ of degree less than n such that $f(u_i) = v_i$ for $0 \leq i < n$.

1. call algorithm 10.3 with input $x - u_0, \ldots, x - u_{n-1}$ to compute $M_{i,j}$
2. $m \longleftarrow M_{k,0}$
   call Algorithm 10.5 to evaluate $m'(u_i) = \frac{m}{x-u_i} \mid_{x=u_i} = \frac{1}{s_i}, i = 0, \ldots, n-1$.
3. call Algorithm 10.9 with input $u_0, \ldots, u_{n-1}, v_0 s_0, \ldots, v_{n-1} s_{n-1}$ and $M_{i,j}$.

Corollary 2. *Algorithm 10.11 solves the interpolation problem over a commutative ring* R *using at most* $O(M(n)\log n)$ *operations in* R.

10.3 Fast Chinese remaindering

$$\begin{array}{rcl} \chi : R[x]/ < m > & \longrightarrow & R[x]/ < m_0 > \times \cdots \times R[x]/ < m_{r-1} > \\ f & \longmapsto & (f \mod m_0, \ldots, f \mod m_{r-1}) \end{array}$$

Algorithm 10.14 Fast simultaneous reduction with precomputation

Input: Monic nonconstant moduli $m_0, \ldots, m_{r-1} \in R[x]$, where $r = 2^k$ with $k \in \mathbb{N}$, $f \in R[x]$ of degree less than $n = \sum_{0 \le i < r} \deg m_i$, and $M_{i,j}$

Output: $f \operatorname{rem} m_0, \ldots, f \operatorname{rem} m_{r-1} \in R[x]$

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

Theorem 2. *Algorithm 10.14 works correctly and takes at most* $O(M(n) \log r)$ *operations in* R.

Algorithm 10.16 Simultaneous inverse computation

Input: Monic nonconstant moduli $m_0, \ldots, m_{r-1} \in R[x]$ such that $\mathrm{res}(m_i, m_j) \in R^\times$ for $i \neq j$, where $r = 2^k$ with $k \in \mathbb{N}$, and $m = m_0 \cdots m_{r-1}$

Output: $s_0, \ldots, s_{r-1} \in R[x]$ with $s_i \frac{m}{m_i} \equiv 1 \mod m_i$ and $\deg_{s_i} < \deg_{m_i}$ for all i.

1. Compute $m$remm_i^2 for all i
2. for $0 \leq i < r$ compute $\frac{m}{m_i}$remm_i
3. for $0 \leq i < r$ compute $s_i \in R[x]$ with $\deg s_i < \deg m_i$ and $s_i \cdot (\frac{m}{m_i}$rem$m_i) \equiv 1 \mod m_i$.
4. return $s_0, \ldots, s_{r-1}$

Algorithm 10.22 Fast Chinese Remainder Algorithm

Input: Monic nonconstant moduli $m_0, \ldots, m_{r-1} \in R[x]$ such that $\operatorname{res}(m_i, m_j) \in R^\times$ for $i \neq j$, where $r = 2^k$ with $k \in \mathbb{N}$, and $v_0, \ldots, v_{r-1} \in R[x]$ with $\deg v_i < \deg m_i$ for all i.

Output: The unique polynomial $f \in R[x]$ of degree less than $n = \sum_{0 \leq i < r} \deg m_i$ such that $f \equiv v_i \mod m_i$ for $0 \leq i < n$.

1. Compute $M_{i,j}$
2. Compute polynomials $s_i \in R[x]$ with $s_i \frac{m}{m_i} \equiv 1 \mod m_i$ and $\deg s_i < \deg m_i$.
3. Compute $c_0 = v_0 s_0 \operatorname{rem} m_0, \ldots, c_{r-1} = v_{r-1} s_{r-1} \operatorname{rem} m_{r-1}$
4. Return $f = \sum_{0 \leq i < r} c_i \frac{m}{m_i}$

Theorem 3. *Given* $m_0, \ldots, m_{r-1} \in F[x]$ *monic and pairwise coprime, where* F *is a field, and* $v_0, \ldots, v_{r-1} \in F[x]$ *with* $\deg v_i < \deg m_i$ *for all* i*, we can compute the unique solution* $f \in F[x]$ *of degree less than* $n = \sum_{0 \leq i < r} \deg m_i$ *of the Chinese Remainder Problem*

$$f \equiv v_i \mod m_i \text{ for } 0 \leq i < r$$

using $O(M(n) \log n)$ *operations in* F