SFU SIMON FRASER UNIVERSITY

Implementing Kaltofen and Yagati's fast transposed Vandermonde solver

Hyukho Kwon, Michael Monagan

Department of Mathematics
Simon Fraser University

July 23, 2024

Outline

1. Transposed Vandermonde System
2. Fast Division
3. Fast Multipoint Evaluation
4. Fast Transposed Vandermonde Solver
5. C Benchmark

Transposed Vandermonde System of Equations

Let F be a field. Let $a = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} \in F[x]$ where all a_is are unknown. Given $u_1, u_2, \ldots, u_n \in F$, let $b_i = a(u_i)$ for $1 \leq i \leq n$.

Let $u_i = \alpha^{i-1}$ for some $\alpha \in F$ where $\alpha^i \neq \alpha^j$ for all $i \neq j$. Then

$$b_i = a(u_i) = \sum_{j=0}^{n-1} a_j (\alpha^{i-1})^j = \sum_{j=0}^{n-1} a_j (\alpha^j)^{i-1} = \sum_{j=0}^{n-1} a_j (u_{j+1})^{i-1} \text{ for } 1 \leq i \leq n.$$

Problem: Interpolate a with $u_1, u_2, \ldots, u_n$ and $b_1, b_2, \ldots, b_n$.

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ u_1 & u_2 & u_3 & \cdots & u_n \\ u_1^2 & u_2^2 & u_3^2 & \cdots & u_n^2 \\ \vdots & \vdots & \vdots & & \vdots \\ u_1^{n-1} & u_2^{n-1} & u_3^{n-1} & \cdots & u_n^{n-1} \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}}_{\mathbf{b}}$$

Transposed Vandermonde System of Equations (continued)

We call $U\mathbf{a} = \mathbf{b}$ a transposed Vandermonde system of equations.

Methods	# ops in F	space
Gaussian Elimination	$O(n^3)$	$O(n^2)$
Zippel's method [6]	$O(n^2)$	$O(n)$
Kaltofen & Yagati's method [4]	$O(M(n)\log n)$	$O(n\log n)$

$M(n)$ is the number of field operations for polynomial multiplication with two polynomials at most n in $F[x]$.

Goal: an algorithm to solve transposed Vandermonde system which does $O(n\log^2 n)$ field operations in F.

Fast multiplication

The in-place recursive FFT from Law and Monagan [5] which does exactly $\frac{1}{2} n \log_2 n$ multiplications.

- An array W of size n of the powers of ω needed for all recursive calls
- The two bit-reversal permutations cancellation.

The three primes method: Let $f, g \in \mathbb{Z}_p[x]$. To compute $f \times g \in \mathbb{Z}_p[x]$

- Choose three Fourier primes p_1, p_2, p_3 such that

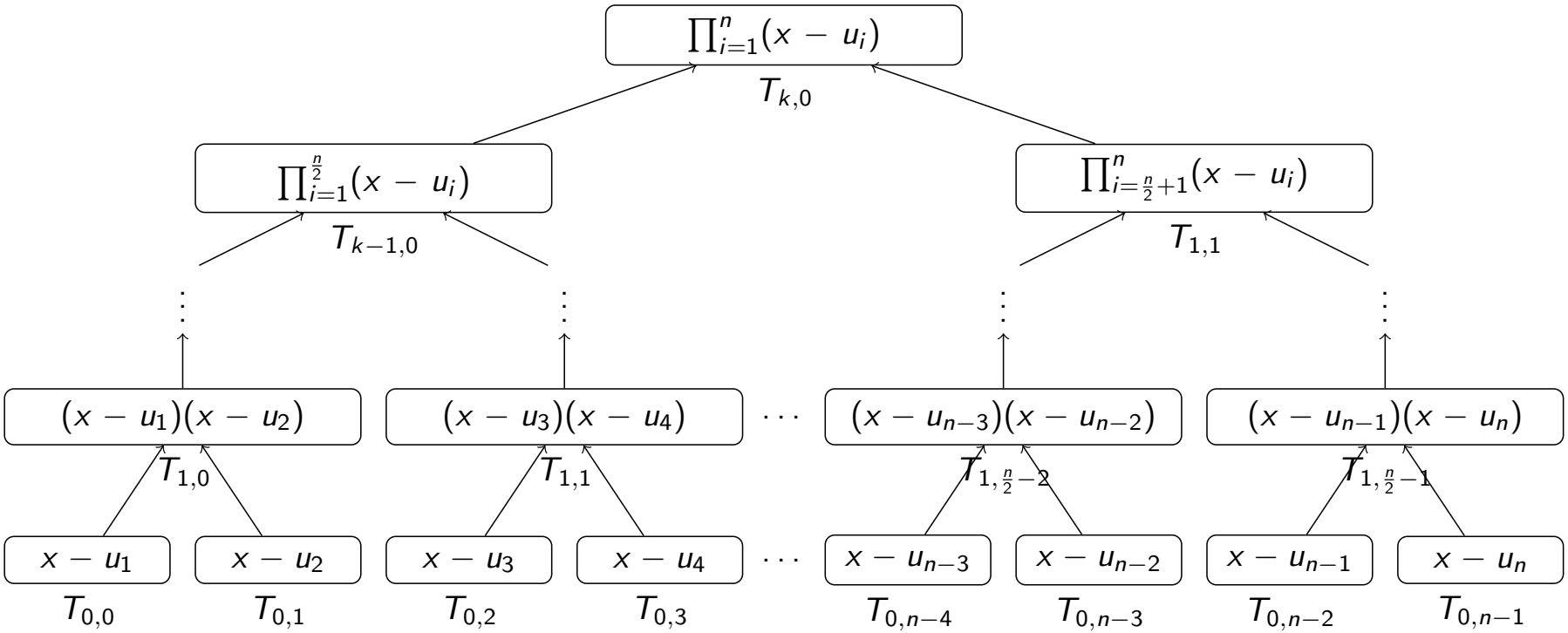
$$\prod_{i=1}^{3} p_i > (p-1)^2 \min(1 + \deg(f), 1 + \deg(g)).$$

- Use the FFT to multiply $f \times g \mod p_i$ for $i = 1, 2, 3$.
- Recover the integer coefficients in $f \times g$ before reduction mod p by Chinese remaindering.

Fast multiplication does $M(n) = 27 n \log_2 n + O(n)$ field operations in $\mathbb{Z}_p[x]$ [5].

Product Tree

Kaltofen and Yagati's method utilized Borodin and Munro's product tree [1] for polynomial multipoint evaluation algorithm with distinct evaluation points $u_1, u_2, \ldots, u_n \in F$ where $n = 2^k$ for some $k \in \mathbb{N}$.



Newton Inversion

Let $f, g \in \mathbb{Z}_p[x]$ such that $\deg(f) = m < 2n$ and $\deg(g) = n$ to compute r, q satisfying $f = g \cdot q + r$ where $\deg(r) < \deg(g)$ or $r = 0$

Newton inversion finds the inverse of the divisor polynomial to an order $m - n + 1$ approximation, i.e. $g^{-1} \mod x^{m-n+1}$.

Theorem (Theorem 9.2 [2])

Assume $h = \sum_{i=0}^{n} h_i x^i \in \mathbb{Z}_p[x]$ *and* $h_0 \neq 0$. *Let*

$$\begin{cases} y_0 = h_0^{-1} \\ y_i = 2y_{i-1} - h \cdot y_{i-1}^2 \mod x^{2^i} \quad \forall i \geq 1 \end{cases}$$

Then, for all $i \geq 0$,

$$h \cdot y_i \mod x^{2^i} = 1.$$

Newton inversion does at most $3M(n) + O(n)$ field operations in $\mathbb{Z}_p[x]$ when $m = 2n - 1$.

Newton Inversion with the Middle Product

In 2004, Hanrot, Quercia, and Zimmerman introduced an alternative formula for Newton inversion [3].

$$y_k = 2y_{k-1} - h \cdot y_{k-1}^2 = y_{k-1} + y_{k-1} \cdot (1 - h \cdot y_{k-1} \mod x^{2^k}) \mod x^{2^k}$$

Assuming $n = 2^k$, $h \cdot y_{k-1} \mod x^{\frac{n}{2}} = 1$. Then

$$h \cdot y_{k-1} \bmod x^n = 1 + \underbrace{m_0 x^{\frac{n}{2}} + \cdots + m_{\frac{n}{2}-1} x^{n-1}}_{= mp \cdot x^{\frac{n}{2}}} + \underbrace{a_0 x^n + \cdots + a_{\frac{n}{2}-2} x^{\frac{3n}{2}-2}}_{= a \cdot x^n} \bmod x^n$$

$$= 1 + mp \cdot x^{\frac{n}{2}}$$

The polynomial mp is called the middle product. Thus,

$$y_k = y_{k-1} + (y_{k-1} \cdot (-mp)) \cdot x^{\frac{n}{2}} \mod x^n.$$

Using the FFT of size n, Newton inversion costs at most $2M(n) + O(n)$.

Computing the FFT on y_{k-1} once reduces the cost to $\frac{5}{3}M(n) + O(n)$.

Fast Division

Definition

Let $f \in \mathbb{Z}_p[x]$ be a polynomial such that $f = f_0 + f_1 x + \cdots + f_{d-1} x^{d-1} + f_d x^d$ with $f_d \neq 0$. The reciprocal polynomial of f is

$$\widehat{f}(x) = x^d f\left(\frac{1}{x}\right) = f_0 x^n + f_1 x^{d-1} + \cdots + f_{d-1} x + f_d.$$

Theorem

Let $\mathbb{Z}_p$ be a field. Suppose $f, g \in \mathbb{Z}_p[x]$ are polynomials such that $\deg(f) = m$ and $\deg(g) = n$, where $m \geq n$. Let $r, q \in \mathbb{Z}_p[x]$ satisfy $f = g \cdot q + r$ where $r = 0$ or $\deg(r) < \deg(g)$. Then,

$$\widehat{q} = \widehat{f} \cdot (\widehat{g})^{-1} \mod x^{m-n+1}$$

Fast Division Algorithm

Input: Polynomials $f, g \in F[x]$ where $g \neq 0$.
Output: The remainder and quotient $r, q \in F[x]$ satisfying $f = g \cdot q + r$ where $r = 0$ or $\deg(r) < \deg(g)$

1: $m \leftarrow \deg(f)$
2: $n \leftarrow \deg(g)$
3: **if** $m < n$ **then return** $f, 0$ **end if**
4: $s \leftarrow m - n + 1$ // $\deg(q) = m - n$
5: $c \leftarrow \text{NIwithMP}(\widehat{g} \mod x^s, s)$ // $c = (\widehat{g})^{-1} \mod x^s$ ..... $I(n) \leq \frac{5}{3}M(n) + O(n)$
6: $e \leftarrow (\widehat{f} \mod x^s) \cdot c$ using Fast multiplication ..... $M(n)$
7: $\widehat{q} \leftarrow e \mod x^s$ // $\widehat{q} = \sum_{i=0}^{m-n} q_i^* x^i$
8: $q \leftarrow \sum_{i=0}^{m-n} q_{m-n-i}^* x^i$
9: $M \leftarrow g \cdot q$ using Fast multiplication ..... $M(n)$
10: $r \leftarrow f - M$
11: **return** r, q

This algorithm does at most $\frac{11}{3}M(n) + O(n)$ arithmetic operations in F when $\deg(f) = 2n - 1$.

Building the Product Tree

Let $f \in F[x]$ such that $\deg(f) = n - 1 < n$. Assume we have n distinct evaluation points $u_0, u_1, \ldots, u_{n-1}$.

Borodin and Munro presented the product tree

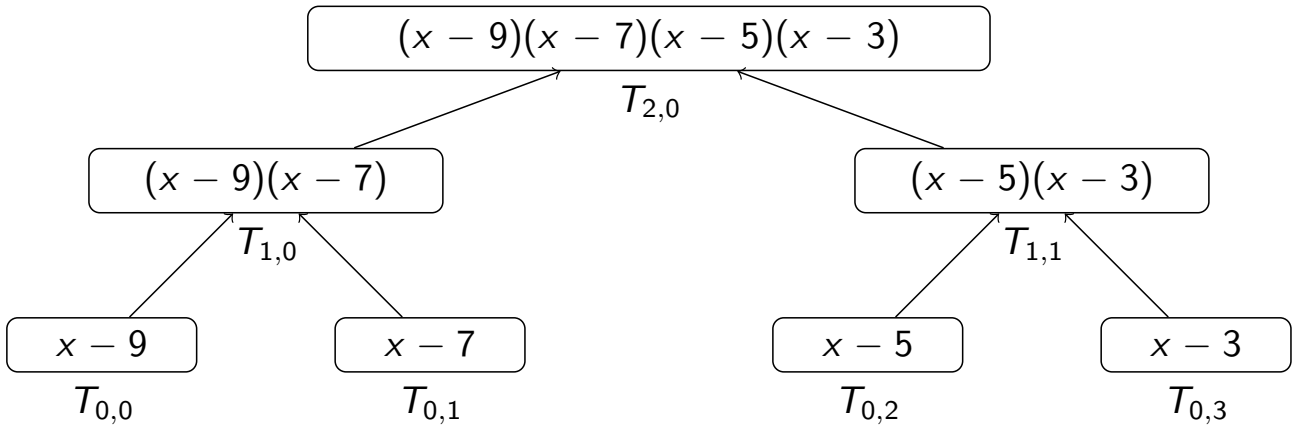


Figure: The product tree with evaluation points 9,7,5, and 3 in $\mathbb{Z}_{97}[x]$

Building the product tree (BuPT) does $M(n)\log_2 n + O(n \log n)$ field operations in $\mathbb{Z}_p$.

Optimizing the FFT, the cost reduces to $\frac{1}{2}M(n)\log_2 n + O(n \log n)$ field operations in $\mathbb{Z}_p$.

Dividing Down the Product Tree

Let $f = 1 + 2x + 3x^2 + 4x^3 \in \mathbf{Z}_{97}[x]$ to be evaluated.

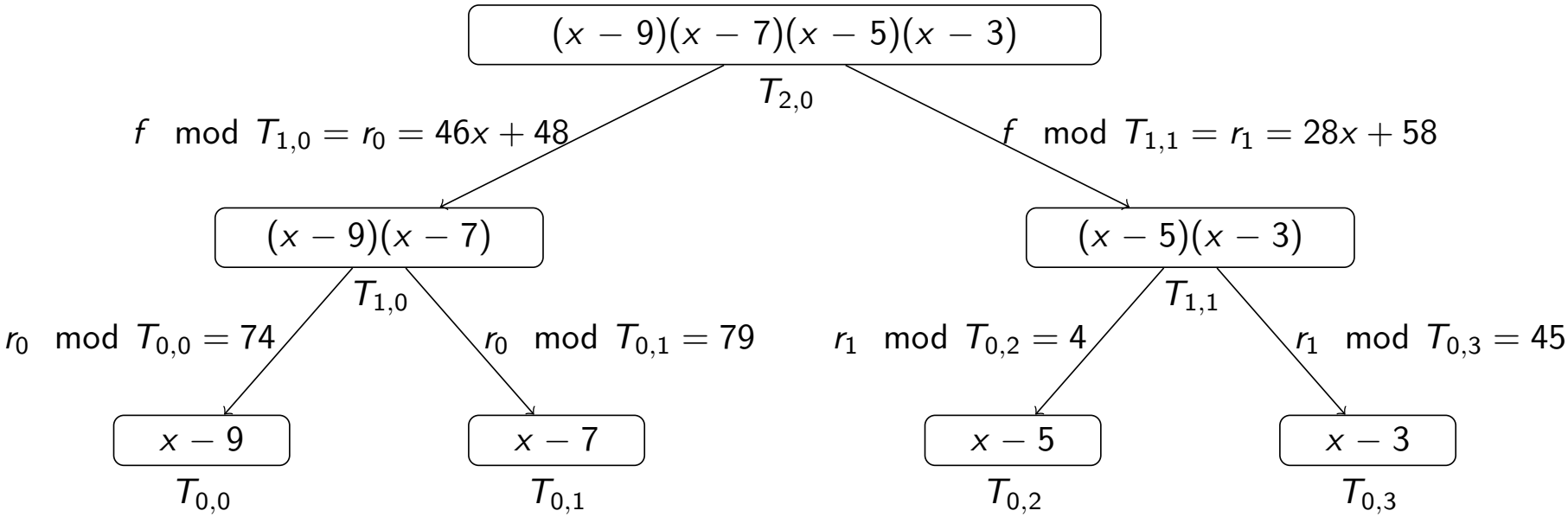


Figure: Dividing down the product tree to evaluate $f = 1 + 2x + 3x^2 + 4x^3 \in \mathbf{Z}_{97}[x]$

$f(9) = 74$, $f(7) = 79$, $f(5) = 4$, and $f(3) = 45$.

Dividing down the product tree (DDPT) does $\frac{11}{3} M(n) \log_2 n + O(n \log n)$ field operations in $\mathbb{Z}_p$.

Fast Multipoint Evaluation Algorithm

Input: $n = 2^k$ for some $k \in \mathbb{N}$, $f \in F[x]$ of degree less than n, and $u_0, u_1, \ldots, u_{n-1} \in F$.
Output: $f(u_0), f(u_1), \ldots, f(u_{n-1}) \in F$

1: $T \leftarrow \mathrm{BUPT}(n, u_0, u_1, \ldots, u_{n-1})$ ..... $B(n) \leq \frac{1}{2} M(n) \log_2 n + O(n \log n)$
2: $f(u_0), f(u_1), \ldots, f(u_{n-1}) \leftarrow \mathrm{DDPT}(n, f, T)$ ..... $C(n) \leq \frac{11}{3} M(n) \log_2 n + O(n \log n)$
3: **return** $f(u_0), f(u_1), \ldots, f(u_{n-1})$

This algorithm does at most $\frac{25}{6} M(n) \log_2 n + O(n \log n)$ field operations in $\mathbb{Z}_p$.

Remark: We can observe that DDPT is $\frac{22}{3}$ times as expensive as BUPT.

Zippel's Transposed Vandermonde Solver

Recall the following $n \times n$ transposed Vandermonde system of equations

$$\underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ u_1 & u_2 & \cdots & u_n \\ \vdots & \vdots & & \vdots \\ u_1^{n-1} & u_2^{n-1} & \cdots & u_n^{n-1} \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{\mathbf{b}}$$

In Zippel's method, define U^{-1}

$$U^{-1} = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,n} \\ \vdots & \vdots & & \vdots \\ s_{n,1} & s_{n,2} & \cdots & s_{n,n}. \end{bmatrix}$$

Let $p_i(x) = s_{i,1} + s_{i,2}x + \cdots + s_{i,n}x^{n-1}$.

Zippel's Transposed Vandermonde Solver (continued)

Since $U^{-1}U = I$ and I is the $n \times n$ identity matrix,

$$\begin{bmatrix} s_{i,1} & s_{i,2} & \cdots & s_{i,n} \end{bmatrix} \begin{bmatrix} 1 \\ u_j \\ \vdots \\ u_j^{n-1} \end{bmatrix} = p_i(u_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Define $M = \prod_{i=1}^{n}(x - u_i)$ and $q_i(x) = M/(x - u_i)$.

Set $p_i(x) = q_i(u_i)^{-1} \cdot q_i(x)$ to recover i-th row entries of U^{-1}.

Compute the dot product of i-th row of U^{-1} and $\mathbf{b}$ to solve the transposed Vandermonde system.

This method does $O(n^2)$ field operations in $\mathbb{Z}_p$ and uses space for $O(n)$ elements of $\mathbb{Z}_p$.

Kaltofen & Yagati's Fast Transposed Vandermonde Solver

Since $p_i(x) = q_i(u_i)^{-1} \cdot q_i(x)$, $M = q_i(u_i) \cdot (x - u_i) \cdot p_i(x)$.

Define $D = b_n x + b_{n-1} x^2 + \cdots + b_1 x^n$. Let $H = M \cdot D = \sum_{i=0}^{2n-1} h_i x^{i+1}$.

$$H/(x - u_i) = q_i(u_i) \cdot p_i(x) \cdot D.$$

The coefficient of x^n in $H/(x - u_i)$ is

$$q_i(u_i) \cdot (s_{i,1} \cdot b_1 + s_{i,2} \cdot b_2 + \cdots + s_{i,n} \cdot b_n) = q_i(u_i) \cdot a_{i-1}.$$

In $H/(x - z)$, the coefficient of x^n is

$$v(z) = h_n + h_{n+1} z + \cdots + h_{2n-1} z^{n-1}.$$

Then $v(u_i) = q_i(u_i) \cdot a_{i-1}$. Also,

$$M'(x) = (x - u_i) \cdot q_i'(x) + q_i(x)$$

so $M'(u_i) = q_i(u_i)$. Thus, $a_{i-1} = v(u_i)/M'(u_i)$.

Fast Transposed Vandermonde Solver Algorithm

Input: $n = 2^k$ for some $k \in \mathbb{N}$, $u_1, u_2, \ldots, u_n \in \mathbb{Z}_p$, which compose the transposed Vandermonde matrix U and $\mathbf{b} = [b_1, b_2, \ldots, b_n] \in \mathbb{Z}_p^n$

Output: $\mathbf{a} = [a_0, a_1, \ldots, a_{n-1}] \in \mathbb{Z}_p^n$ satisfying $U\mathbf{a} = \mathbf{b}$

1: $T \leftarrow \text{BUPT}(n,\ u_1, u_2, \ldots, u_n)$ ..... $B(n) \leq \frac{1}{2}M(n)\log_2 n + O(n \log n)$
2: $M \leftarrow T_{k,0}$ from T
3: $D \leftarrow b_n x + b_{n-1}x^2 + \cdots + b_1 x^n$
4: $H \leftarrow M \cdot D$ using fast multiplication // $H = \sum_{i=0}^{2n-1} h_i x^{i+1}$ ..... $M(n)$
5: $Q \leftarrow \sum_{i=0}^{n-1} h_{n+i} z^i$ // Q = the coefficient of x^n in $H/(x - z)$
6: $q_1, q_2, \ldots, q_n \leftarrow \text{DDPT}(n, Q, T)$ // $q_i = Q(u_i)$ .. $C(n) \leq \frac{11}{3}M(n)\log_2 n + O(n \log n)$
7: Differentiate M
8: $r_1, r_2, \ldots, r_n \leftarrow \text{DDPT}(n, M', T)$ // $r_i = M'(u_i)$. $C(n) \leq \frac{11}{3}M(n)\log_2 n + O(n \log n)$
9: **for** i from 1 to n **do**
10: $\quad t \leftarrow r_i^{-1}$
11: $\quad a_{i-1} \leftarrow t \cdot q_i$
12: **end for**
13: **return** $[a_0, a_1, \ldots, a_{n-1}]$

This algorithm does $\frac{53}{6}M(n)\log_2 n + O(n \log n) \in O(M(n) \log n)$ field operations in $\mathbb{Z}_p$.

Case 1: $p = 116 \cdot 2^{55} + 1$

n	FastTVS					ZippelTVS	speedup	Maple	speedup
	BuPT	InvTree	DDPT1	DDPT2	Total				
2^6	0.046	-	0.046	0.039	0.195	0.1389	0.71	3.4	17.4
2^7	0.086	-	0.107	0.098	0.380	0.4879	1.28	8.6	22.6
2^8	0.150	-	0.254	0.238	0.808	1.9039	2.35	20.8	25.7
2^9	0.363	-	0.693	0.674	2.065	7.4640	3.61	63.0	30.5
2^{10}	0.875	0.600	1.890	1.877	5.811	30.826	5.30	113.2	19.5
2^{11}	2.020	2.417	5.070	5.008	15.775	116.84	7.40	270.0	17.1
2^{12}	4.755	7.529	12.307	12.268	39.444	469.64	11.90	608.0	15.4
2^{13}	11.146	20.556	29.566	29.270	95.765	1,868	19.50	1,321	13.8
2^{14}	25.901	53.099	71.091	70.580	231.55	7,456	32.19	3,025	13.1
2^{15}	60.151	131.30	166.15	166.46	546.52	29,986	54.86	7,190	13.2
2^{16}	131.23	314.56	380.02	376.77	1,249.3	120,292	96.28	16,455	13.2
2^{17}	339.89	746.70	867.30	863.48	2,914.9	478,912	164.3	69,705	23.9
2^{18}	663.01	1,747.1	1,961.8	1,955.2	6,529.8	1,929,776	295.5	97,667	15.0

Table: CPU timings in *ms* for solving $n \times n$ transposed Vandermonde systems over the prime field $\mathbb{Z}_p$ with $p = 116 \cdot 2^{55} + 1$

Case 2: $p = 14411518807585859$

n	FastTVS					ZippelTVS	speed up
	BuPT	InvTree	DDPT1	DDPT2	total		
2^6	0.043	-	0.042	0.040	0.247	0.1509	0.61
2^7	0.059	-	0.145	0.144	0.545	0.4869	0.89
2^8	0.276	-	0.343	0.340	1.355	1.9060	1.40
2^9	0.899	-	0.857	0.843	3.463	7.4809	2.16
2^{10}	2.615	-	2.388	2.398	9.195	29.702	3.23
2^{11}	6.685	-	7.510	7.374	25.353	116.44	4.59
2^{12}	16.044	-	25.265	25.139	73.946	470.04	6.35
2^{13}	38.754	80.014	76.846	76.688	288.00	1865.1	6.47
2^{14}	93.628	212.80	213.94	214.83	768.44	7478.1	9.73
2^{15}	214.23	510.61	541.49	540.96	1,875.7	29,763	15.86
2^{16}	497.72	1,237.4	1,343.2	1,354.4	4,576.6	119,478	26.10
2^{17}	1,111.9	2,890.1	3,199.1	3,210.3	10,716	488,369	45.57
2^{18}	2,494.2	6,632.9	7,480.6	7,470.6	24,725	1,953,252	78.99

Table: CPU timings in *ms* for solving $n \times n$ transposed Vandermonde system over $\mathbb{Z}_p$ with $p = 14411518807585859 < 2^{57}$

References

[1] Allan Borodin and Ian Munro. "Evaluating polynomials at many points". In: *Information Processing Letters* 1.2 (1971), pp. 66–68.

[2] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. eng. United States: Cambridge University Press, 2013.

[3] Guillaume Hanrot, Michel Quercia, and Paul Zimmermann. "The Middle Product Algorithm I. Speeding up the division and square root of power series". eng. In: *Applicable algebra in engineering, communication and computing* 14.6 (2004), pp. 415–438.

[4] Erich Kaltofen and Lakshman Yagati. "Improved sparse multivariate polynomial interpolation algorithms". eng. In: *ISSAC*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 467–474.

[5] Marshall Law and Michael Monagan. "A parallel implementation for polynomial multiplication modulo a prime". eng. In: *Proceedings of the 2015 International Workshop on parallel symbolic computation*. ACM, 2015, pp. 78–86.

References (continued)

[6] Richard Zippel. “Interpolating polynomials from their values”. eng. In: *Journal of symbolic computation* 9.3 (1990), pp. 375–403.