Counting Perfect Matchings and Hamiltonian Cycles Faster

Baitian Li
Columbia University, New York, NY, USA

Abstract

We show that the hafnian of a symmetric $2n \times 2n$ matrix of poly(n)-bit integers (which counts the number of perfect matchings of a $2n$-vertex graph) and the number of Hamiltonian cycles of an n-vertex directed graph can be computed in time $2^{n-\Omega(\sqrt{n})}$, improving and generalizing an earlier algorithm of Björklund, Kaski, and Williams (Algorithmica 2019) that runs in time $2^{n-\Omega\left(\sqrt{n/\log\log n}\right)}$.

A key tool of our approach is the design of a data structure that supports fast evaluation of high-order derivatives of hafnian and Hamiltonian cycles, which integrates with the new approach on multivariate multipoint evaluation by Bhargava, Ghosh, Guo, Kumar, and Umans (FOCS 2022, JACM 2024).

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases permanent, hafnian, Hamiltonian cycle, Kakeya sets

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.138

Category Track A: Algorithms, Complexity and Games

Funding *Baitian Li*: Supported in part by NSF Grant CCF-2238221, a Packard Foundation Fellowship, and a Columbia SEAS Presidential Fellowship.

Acknowledgements The work was done when the author was an undergraduate student at Tsinghua University. The author would like to thank Josh Alman and anonymous referees for helpful comments on earlier drafts.

1 Introduction

Given an $n \times n$ matrix A over a commutative ring R, the R-Permanent is defined by

$$\operatorname{per} A = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)},$$

where S_n denotes the symmetric group on $[n]$, i.e., permutations of $\{1, \ldots, n\}$. Similarly, R-HamCycles is defined as

$$\operatorname{hc} A = \sum_{\substack{\sigma \in S_n \\ c(\sigma)=1}} \prod_{i=1}^{n} A_{i,\sigma(i)},$$

where $c(\sigma)$ denotes the number of cycles in σ.

The permanent and Hamiltonian cycles are two fundamental problems in computer science. The problem of deciding whether a given graph has a Hamiltonian cycle is one of Karp's 21 NP-complete problems [19]. Valiant proved that over the integers, computing the permanent is #P-complete, even if the entries of the matrix are restricted to 0 and 1 [24], and counting Hamiltonian cycles is also #P-complete [25].

Ryser's formula [23] shows that the permanent can be computed with $O(n2^n)$ arithmetic operations. It remains a prominent open problem whether the permanent can be computed with arithmetic circuits of size less than 2^n, as mentioned by Knuth [21, Exercise 4.6.4.11].

Indeed, beyond the confines of arithmetic operations, faster algorithms for computing the permanent have emerged. Bax and Franklin [1] gave an algorithm that computes the 01-permanent in $2^{n-\Omega(n^{1/3}/\log n)}$ expected time. For dense instances over finite fields and integers, Björklund [7] introduced a framework based on self-reduction and tabulation, achieving a running time of $2^{n-\Omega\left(\sqrt{n/\log n}\right)}$. Björklund, Kaski, and Williams [11] refined the tabulation step via Kakeya sets, obtaining an improved running time $2^{n-\Omega\left(\sqrt{n/\log\log n}\right)}$.

1.1 Our results

In this paper, we further improve the algorithm of Björklund, Kaski, and Williams [11], removing the $\log\log n$ term in the exponent. We also show how to extend the complexity bound for permanent to a natural extension called hafnian.

For a $2n \times 2n$ symmetric matrix A over a commutative ring R, the R-Hafnian of A is defined as

$$\operatorname{haf}(A) = \sum_{\sigma \in P_{2n}} \prod_{(i,j)\in\sigma} A_{i,j},$$

where P_{2n} is the family of partitions of $[2n]$ into n pairs. The permanent of an $n \times n$ matrix can be reduced to the hafnian of a $2n \times 2n$ matrix via the following basic relation:

$$\operatorname{per}(A) = \operatorname{haf}\begin{pmatrix} 0 & A \\ A^{\mathsf{T}} & 0 \end{pmatrix}.$$

Previously the best known algorithm for hafnian ran in time $\tilde{O}(2^n)$ (for a $2n \times 2n$ matrix), first developed by Björklund [5] and alternatively by Cygan and Pilipczuk [15].

▶ **Theorem 1.** *There is an algorithm that computes the permanent* $\operatorname{per}(A)$ *of a given matrix* $A \in \mathbb{F}_q^{n\times n}$ *in time* $2^{n-\Omega(\sqrt{n})}q^{O(1)}$. *The same bound holds for hafnian* $\operatorname{haf}(A)$ *of a given symmetric matrix* $A \in \mathbb{F}_q^{2n\times 2n}$, *and for computing Hamiltonian cycles* $\operatorname{hc}(A)$ *of a given matrix* $A \in \mathbb{F}_q^{n\times n}$.

The Chinese remainder theorem and a simple estimate of prime products yield the following corollary for integer-valued matrices.

▶ **Corollary 2.** *Given a* $2n \times 2n$ *symmetric matrix with integer entries having absolute values bounded by* M, *we can compute* $\operatorname{haf}(A)$ *(or* $\operatorname{per}(A)$*) in time* $2^{n-\Omega(\sqrt{n})}(\log M)^{O(1)}$. *The same type of bound holds for computing Hamiltonian cycles* $\operatorname{hc}(A)$ *of a given* $n \times n$ *integer matrix.*

1.2 Related works

Multivariate Multipoint Evaluation. Our algorithm is inspired by progress in multivariate multipoint evaluation. Kedlaya and Umans [20] introduced a tabulation-based approach (combined with the Chinese remainder theorem) that later became a key ingredient in fast polynomial composition and factorization. More recently, a sequence of works [4, 2, 3] developed the use of Hasse derivatives and Hermite interpolation to extract more information per evaluation point. We adapt these ideas to our sparse tabulation framework.

Permanents. There exist faster algorithms for computing the permanent in other settings. For sparse matrices, Cygan and Pilipczuk [15] gave a $2^{n-\Omega(n/d)}$ time algorithm, where d is the average degree of non-zero entries per row. Björklund and Williams [12] gave a $2^{n-\Omega(n/d^{3/4})}$ time algorithm for d-regular bipartite graphs, and a $2^{n-\Omega(n/r)}$ time algorithm that runs over

a finite ring with r elements. Björklund, Husfeldt, and Lyckberg [9] gave a $2^{n-\Omega(n/(p\log p))}$ time algorithm for computing the permanent modulo a prime power $p^{\lfloor \lambda n/p \rfloor}$, for any constant $\lambda < 1$.

Hamiltonian cycles. There exist faster algorithms for counting Hamiltonian cycles in other settings as well. Björklund, Kaski, and Koutis [10] gave an $O((2-\delta)^n)$-time algorithm for counting Hamiltonian cycles modulo moderate prime powers. In the general setting, it is somewhat surprising that our *counting* algorithm also yields the fastest known algorithm for *deciding* Hamiltonicity. This differs from the case of permanents: the support of the permanent corresponds to perfect matchings in bipartite graphs, whose existence can be decided in polynomial time. Faster decision algorithms are known in special cases, including Björklund's $O(1.66^n)$-time algorithm for undirected graphs [6] and the $O(1.888^n)$-time algorithm of Cygan, Kratsch, and Nederlof for directed bipartite graphs [14].

1.3 Technical overview

For simplicity, we first sketch the case of computing the permanent.

Our improvement comes from combining three ideas:

1. *Reduce to smaller instances.* We reduce the computation on an $n \times n$ matrix to many instances on $k \times k$ matrices. Taking k around $\sqrt{n}$ is what creates room for an improvement in the exponent, provided we can answer the reduced instances fast. This kind of reduction ("self-reduction") for permanents and Hamiltonian cycles was introduced by Björklund [7]. More concretely, for a parameter k, the reduction produces about $2^{n-k}\,\mathrm{poly}(n)$ instances on $k \times k$ matrices.
2. *Tabulate only on a sparse set of points.* A direct lookup table for all smaller matrices would be far too large. Instead, we employ the fact that permanent is a low-degree polynomial – we tabulate only on a carefully chosen sparse subset of points with the following key property: for every query point, there exists a low-degree univariate curve passing through it whose other points all lie inside the tabulated subset. Then we can recover the value at the query point by interpolating along that curve. (Over finite fields, such subsets are called *Kakeya sets*. Björklund, Kaski, and Williams [11] were the first to leverage this idea for multivariate polynomial multipoint evaluation in our setting, and we build on their approach.)
3. *Make each tabulated point more informative.* Interpolating from plain point evaluations is limited by how many points we see on a curve. We use the recent idea from Bhargava, Ghosh, Guo, Kumar, and Umans [2, 3] to enrich each tabulated point with additional local information (captured via suitable high-order derivatives), and then use Hermite interpolation to recover higher-degree information along the curve. This is what allows the sparse tabulation approach to only require smaller Kakeya sets, thus working at the larger subproblem sizes we need. (We refer to this task as *high-order derivative evaluation*.)

Our main technical contribution is a dynamic programming algorithm that makes the required derivative access efficient for the permanent, and we develop analogous data structures for hafnian and Hamiltonian cycles. More concretely, for any constant $\epsilon > 0$ and a parameter k, our data structure tabulates over a Kakeya set of size $2^{O_\epsilon(k^2)}$[1] and takes $O(2^{\epsilon k})$ time to evaluate one $k \times k$ permanent, so the total time complexity is $O(2^{n-\Omega(k)} + 2^{O(k^2)})$. Balancing the savings from self-reduction with the costs of tabulation yields the final running time $2^{n-\Omega(\sqrt{n})}$.

[1] Here $O_\epsilon(\cdot)$ means that the hidden constant factor depends on ϵ.

The same high-level framework extends to hafnian and Hamiltonian cycles. For hafnian, we derive a suitable self-reduction by modifying components of Björklund's algorithm [5]. For Hamiltonian cycles, we design an efficient derivative-evaluation data structure based on a determinant characterization [13].

1.4 Discussion

With the tabulation of information on Kakeya sets in $k \times k$ dimensional space, our algorithm essentially computes the hafnian and Hamiltonian cycles in $2^{n-\Omega(k)}n^{O(1)}$ time. It seems that a better construction of a smaller Kakeya set of size $2^{o(k^2)}$ would lead to a faster algorithm. However, the resolution of the finite field Kakeya conjecture [16, 17] rules out such possibilities, showing that the size of a Kakeya set is at least $\Omega(\delta^{k^2})$ when the degree of the curve is not greater than q/δ, which corresponds to the regime of our application. Thus, the current construction is essentially optimal for our purposes.

It seems that we have reached a limit with the current approach of self-reduction and Kakeya sets. It remains open whether the techniques on perturbing Ryser's formula, which work well for sparse permanents [12] and modulo p^k permanents [9], can be adapted to dense permanents to achieve a faster algorithm.

2 Preliminaries

2.1 Notation

We use $\tilde{O}(f(n))$ to denote $O(f(n)\operatorname{poly}\log(f(n)))$.

Bold symbols like $\boldsymbol{x}$ denote vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$.

For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

We use Iverson's bracket notation. Let P be a logical proposition, we let $[\![P]\!]$ be 1 if P is true and 0 otherwise.

With $A \sqcup B$, we denote the disjoint union of two sets A and B.

For an $n \times m$ matrix A, for subsets $S \subseteq [n]$ and $T \subseteq [m]$, we use $A_{S,T}$ to denote the submatrix of A with rows indexed by S and columns indexed by T.

Let $\binom{n}{\downarrow m}$ denote the partial sum of binomial coefficients, i.e.,

$$\binom{n}{\downarrow m} = \sum_{0 \le i \le m} \binom{n}{i}.$$

2.2 Inequality for binomials

We need the estimate of the partial sum of binomials, see [18, Lemma 3.13] for a proof.

▶ **Lemma 3.** *Consider* $0 < \alpha < 1/2$. *Then we have*

$$\binom{n}{\downarrow \alpha n} \le 2^{n\,\mathrm{H}(\alpha)},$$

where $\mathrm{H}(\alpha) = -\log_2(\alpha^\alpha(1-\alpha)^{1-\alpha})$ *is the binary entropy function.*

2.3 Hermite interpolation

We need the following lemma for Hermite interpolation, see [26, Section 5.6] for a proof.

▶ **Lemma 4.** *Let $f(t) \in \mathbb{F}[t]$ be a polynomial of degree less than d, and m distinct points $\tau_1, \ldots, \tau_m$ in $\mathbb{F}$, with multiplicities $e_1, \ldots, e_m$ positive integers such that $e_1 + \cdots + e_m = d$. Given the remainder polynomials $r_i = f \bmod (t - \tau_i)^{e_i}$ for each $i \in [m]$, then*

- *f is uniquely determined by these r_i,*
- *moreover, the coefficients of f can be recovered in $\mathrm{poly}(d)$ many $\mathbb{F}$-operations, given the coefficients of r_i as input.*

In particular, our algorithm uses the case where those distinct points are the entire finite field $\mathbb{F}_q$, and $e_i = r$ for all i.

▶ **Corollary 5.** *Let $f(t)$ be a polynomial of degree less than qr. Given the coefficients of $f \bmod (t - \alpha)^r$ for all $\alpha \in \mathbb{F}_q$, then the coefficients of f can be recovered in $\mathrm{poly}(qr)$ $\mathbb{F}_q$-operations.*

2.4 Multimodular reduction

Our algorithm uses the Chinese remainder theorem to reduce the problem to small finite fields.

▶ **Theorem 6.** *Let $p_1, \ldots, p_n$ be distinct primes, and $a_1, \ldots, a_n$ be integers such that $0 \le a_i < p_i$. Let $M = p_1 \cdots p_n$. Then there exists a unique integer a in the range $0 \le a < M$ such that $a \equiv a_i \pmod{p_i}$ for every $i \in [n]$. Moreover, a can be computed in time $\mathrm{poly}(\log M)$.*

See [26, Section 10.3] for a proof.

We also need an estimate on the product of primes.

▶ **Lemma 7.** *For an integer $N \ge 2$, we have*

$$\prod_{\substack{\text{prime } p \\ p \le 16 \log N}} p > N.$$

See [20, Lemma 2.4] for a proof.

3 Common framework

In this section, we set up the common framework for computing hafnians and counting Hamiltonian cycles.

3.1 Self reduction

We borrow the self-reduction lemma of Hamiltonian cycles from [7, Lemma 4].

▶ **Lemma 8.** *Suppose $|\mathbb{F}| \ge k^2+1$, given a matrix $A \in \mathbb{F}^{n \times n}$, one can compute $m = 2^{n-k} n^{O(1)}$ instances $a_i \in \mathbb{F}, F_i \in \mathbb{F}^{k \times k}$ such that*

$$\mathrm{hc}(A) = \sum_{i=1}^{m} a_i \, \mathrm{hc}(F_i).$$

Furthermore, the computation of these instances takes $2^{n-k} n^{O(1)}$ $\mathbb{F}$-operations.

3.2 Kakeya set

We borrow the definition and construction of Kakeya sets mentioned in [11].

▶ **Definition 9.** *A set $K \subseteq \mathbb{F}_q^m$ is said to be a* Kakeya set *of degree u, if for every $a_1, \ldots, a_m \in \mathbb{F}_q$, there exists degree-$u$ polynomials $g_1, \ldots, g_m$, such that the degree u coefficient of g_i is a_i, and the set*

$$\{(g_1(\tau), \ldots, g_m(\tau)) : \tau \in \mathbb{F}_q\}$$

is a subset of K.

▶ **Theorem 10.** *Let u be a positive integer such that $u+1$ divides $q-1$. Then there is a Kakeya set K of degree u in $\mathbb{F}_q^m$ of size at most*

$$\left(\frac{q-1}{u+1}+1\right)^{m+1}.$$

Such K can be constructed in time $|K| \cdot \mathrm{poly}(q)$ and for each point $\boldsymbol{a} = (a_1, \ldots, a_m) \in \mathbb{F}_q^m$, the coefficients of the corresponding polynomials $g_1, \ldots, g_m$ can be computed in time $\mathrm{poly}(u, m)$.

For the convenience of the reader, we provide the construction below, which is originally from [22].

Proof. Since $u+1$ divides $q-1$, we have $q \equiv 1 \pmod{u+1}$, thus $u+1$ is coprime with q, so $u+1$ is invertible in $\mathbb{F}_q$. For each point $\boldsymbol{a} = (a_1, \ldots, a_m) \in \mathbb{F}_q^m$, we consider the polynomials given by

$$\begin{aligned} g_i(\tau) &= \left(\frac{a_i}{u+1} + \tau\right)^{u+1} - \tau^{u+1} \\ &= \sum_{k=0}^{u} \binom{u+1}{k} \left(\frac{a_i}{u+1}\right)^{u-k+1} \tau^k \end{aligned}$$

The $u+1$-th degree coefficient of g_i cancels out, and the u-th degree coefficient is

$$\binom{u+1}{1} \frac{a_i}{u+1} = a_i.$$

So these polynomials satisfy the leading monomial condition of Definition 9, and one can easily compute the coefficients of g_i in polynomial time, through the explicit expression given above.

Since $u+1$ divides $q-1$, from basic finite field theory, the set $T = \{x^{u+1} : x \in \mathbb{F}_q\}$ consists of $0 \in \mathbb{F}_q$ and roots of unity of order $d = (q-1)/(u+1)$, so $|T| = d+1$. Let K be the set of points

$$(u_1 - v, \ldots, u_m - v) : u_i \in T, v \in T.$$

Since each u_i and v can take $d+1$ values, we have $|K| \leq (d+1)^{m+1}$. Moreover, for each $\tau \in \mathbb{F}_q$, we have $g_i(\tau) = u_i - v$ for $u_i = (a_i/(u+1) + \tau)^{u+1}$ and $v = \tau^{u+1}$. This shows that K is indeed a Kakeya set of degree u, and satisfies the size bound. It is also straightforward to see that the construction can be done in time $|K| \cdot \mathrm{poly}(q)$. ◀

3.3 High-order derivative evaluation

▶ **Definition 11.** *Let P be a polynomial over m indeterminates. We call the following operation a* derivative evaluation *of P at $\boldsymbol{a} \in \mathbb{F}_q^m$ up to* order r *(r-order evaluation): Given a polynomial vector $\boldsymbol{f}(t) = (f_1(t), \ldots, f_m(t))$, where each $f_i(t) \in \mathbb{F}_q[t]$ is a polynomial with degree less than r, and $\boldsymbol{f}(0) = \boldsymbol{a}$. Compute the coefficients of the polynomial $P(\boldsymbol{f}(t)) \bmod t^r$.*

This terminology comes from the intuition in characteristic zero. In that case, computing $P(\boldsymbol{f}(t)) \bmod t^r$ is equivalent to computing all the derivatives of $P(\boldsymbol{f}(t))$ up to order r.

We rephrase the idea of [2, 3] to reveal information from derivatives.

▶ **Theorem 12.** *Let P be a homogeneous degree k polynomial over m indeterminates, b be a positive integer such that $q \equiv 1 \pmod b$. Let $u = (q-1)/b - 1$ and $r = \lceil k/b \rceil$. Let K be a Kakeya set of degree u, with an oracle that supports r-order evaluation query at any point of K.*

Then given any point $\boldsymbol{a}$ and the associated curve $\boldsymbol{C}_{\boldsymbol{a}}(t) = (g_1(t), \ldots, g_m(t))$, we can compute $P(\boldsymbol{a})$ with q oracle queries, and $\mathrm{poly}(k, q)$ arithmetic operations over $\mathbb{F}_q$.

Proof. By the definition of Kakeya sets, it is guaranteed that $\boldsymbol{C}_{\boldsymbol{a}}(\tau) \in K$ for all $\tau \in \mathbb{F}_q$. The polynomial $P(\boldsymbol{C}_{\boldsymbol{a}}(t))$ is of degree ku. Write $P(x_1, \ldots, x_m)$ with

$$P(x_1, \ldots, x_m) = \sum_{\substack{i_1, \ldots, i_m \in \mathbb{N} \\ i_1 + \cdots + i_m = k}} p_{i_1, \ldots, i_m} x_1^{i_1} \cdots x_m^{i_m},$$

since $g_i(t) = a_i t^u + O(t^{u-1})$, we have

$$\begin{aligned}
P(\boldsymbol{C}_{\boldsymbol{a}}(t)) &= \sum_{\substack{i_1, \ldots, i_m \in \mathbb{N} \\ i_1 + \cdots + i_m = k}} p_{i_1, \ldots, i_m} g_1(t)^{i_1} \cdots g_m(t)^{i_m} \\
&= \sum_{\substack{i_1, \ldots, i_m \in \mathbb{N} \\ i_1 + \cdots + i_m = k}} p_{i_1, \ldots, i_m} (a_1 t^u + O(t^{u-1}))^{i_1} \cdots (a_m t^u + O(t^{u-1}))^{i_m} \\
&= \sum_{\substack{i_1, \ldots, i_m \in \mathbb{N} \\ i_1 + \cdots + i_m = k}} p_{i_1, \ldots, i_m} (a_1^{i_1} \cdots a_m^{i_m} t^{ku} + O(t^{ku-1})) \\
&= P(\boldsymbol{a}) t^{ku} + O(t^{ku-1}),
\end{aligned}$$

from which we have that the coefficient of t^{ku} in $P(\boldsymbol{C}_{\boldsymbol{a}}(t))$ is $P(\boldsymbol{a})$.

By the choice of u, we have $ku = k((q-1)/b - 1) < qk/b \le qr$. Let $Q(t) = P(\boldsymbol{C}_{\boldsymbol{a}}(t))$. If we are given $Q(t) \bmod (t-\tau)^r$ for each $\tau \in \mathbb{F}_q$, by Hermite interpolation (Lemma 4), we can recover Q in $\mathrm{poly}(qr)$ operations. So the problem reduces to computing $P(\boldsymbol{C}_{\boldsymbol{a}}(t)) \bmod (t-\tau)^r$ for each $\tau \in \mathbb{F}_q$.

In order to compute $Q(t) \bmod (t-\tau)^r$, one can write $Q(t) = R(t) + (t-\tau)^r D(t)$ where $\deg R < r$, then $R(t)$ is the desired result. Thus we have $Q(t+\tau) = R(t+\tau) + t^r D(t+\tau)$, so we can compute $Q(t+\tau) \bmod t^r$, and then reveal $Q(t) \bmod (t-\tau)^r$ by substituting $t \leftarrow t-\tau$. The conversion of coefficients only takes $\mathrm{poly}(r)$ arithmetic operations over $\mathbb{F}_q$. Thus we only need to compute $P(\boldsymbol{C}_{\boldsymbol{a}}(t+\tau)) \bmod t^r$ for each $\tau \in \mathbb{F}_q$. This is exactly an r-order evaluation of P at $\boldsymbol{C}_{\boldsymbol{a}}(\tau)$. ◀

4 Self reduction for hafnian

In this section, we show that it is enough to take a truncation of Björklund's algorithm [5] to obtain a self-reduction algorithm for the hafnian.

4.1 Technical ingredients from Björklund's algorithm

First, we list the technical ingredients we borrow from Björklund.

We first recap a basic concept introduced in [5, Section 3.1]. For a commutative ring R, the *set-partition algebra* $R[U_m]$ is defined as follows. Intuitively, U_m may be viewed as a *partial* semigroup whose elements are the subsets of $[m]$, with the operation given by disjoint union. Thus, the product of two subsets $U, V \subseteq [m]$ is defined precisely when U and V are disjoint. The algebra $R[U_m]$ is then the R-algebra associated with this partial semigroup, analogous to the group algebra $R[G]$ associated with a group G.

More explicitly, every element $r \in R[U_m]$ can be written uniquely as a formal sum

$$r = \sum_{X \subseteq [m]} r_X [X].$$

where $r_X \in R$. Addition in $R[U_m]$ is defined componentwise, while multiplication is given by

$$r \cdot s = \sum_{X \subseteq [m]} \left(\sum_{Y \sqcup Z = X} r_Y s_Z \right) [X],$$

where the inner sum ranges over all ordered decompositions of X as a disjoint union $Y \sqcup Z$.

We identify $R = R[U_0]$ and the inclusion $R[U_m] \subseteq R[U_{m+1}]$ via the inclusion $[m] \subset [m+1]$. Computationally, an element $r \in R[U_m]$ can be stored as the 2^m coefficients $r_X \in R$ where X goes through all subsets of $[m]$. The multiplication in $R[U_m]$ is known as the *subset convolution*, which can be done in $\tilde{O}(2^m)$ R-operations [8].

Then, Björklund [5, Section 3.2] introduced a sequence of matrices $B^{(i)} \in R[U_i]^{(2n-2i)\times(2n-2i)}$ (for $0 \le i \le n$) starting with $B^{(0)} = A$, together with a sequence called *squeeze factors* $\beta^{(i)} \in R[U_i]$ (for $1 \le i \le n$). We will not need the precise definition of these matrices and factors, but we invoke the following two facts.

▶ **Lemma 13** (Björklund [5, Lemma 4]). *For every $X \subseteq [i-1]$ where $i \ge 1$, the following relation holds:*

$$(\operatorname{haf} B^{(i-1)})_X = (\beta^{(i)} \cdot \operatorname{haf} B^{(i)})_{X \sqcup \{i\}}.$$

The above lemma has an efficient algorithmic counterpart.[2]

▶ **Lemma 14** (Björklund [5, Section 3.4]). *There is an algorithm that given $B^{(i-1)}$, computes $B^{(i)}$ and $\beta^{(i)}$ in $2^i n^{O(1)}$ R-operations.*

4.2 Self reduction via inclusion-exclusion

We take several steps to obtain a self-reduction for hafnian. For $1 \le \ell \le n$, we denote $b^{(\ell)}$ to be the prefix product $b^{(\ell)} = \beta^{(1)} \cdots \beta^{(\ell)}$, by repeatedly applying Lemma 13, we obtain

$$\operatorname{haf} A = (\operatorname{haf} B^{(0)})_{\varnothing} = (b^{(\ell)} \cdot \operatorname{haf} B^{(\ell)})_{[\ell]}. \tag{1}$$

Then the next step involves extracting the ideas of ranked Möbius transform and inversion from the subset convolution algorithm [8, Section 2].

[2] Lemma 14 is implicit in Björklund's original paper. In the original text Björklund merely stated how to sequentially compute $B^{(i)}$ and $\beta^{(i)}$ for all $1 \le i \le n$ and gave a time complexity analysis, however, in the last paragraph of [5, Section 3.4], the analysis of a single squeeze step is given.

We consider the ranked Möbius transform, for which for every $X \subseteq [\ell]$, we introduce the polynomial $\hat{b}_X^{(\ell)} \in R[T]$ defined as

$$\hat{b}_X^{(\ell)} = \sum_{Y \subseteq X} b_Y^{(\ell)} T^{|Y|}.$$

Similarly, we define $\hat{B}_X^{(\ell)} \in R[T]^{(2n-2\ell)\times(2n-2\ell)}$ as

$$\hat{B}_X^{(\ell)} = \sum_{Y \subseteq X} B_Y^{(\ell)} T^{|Y|}.$$

For a polynomial $f \in R[T]$, let $[T^d]f$ denote the coefficient of T^d in f. We first prove a general statement and then apply it to equation (1).

▶ **Lemma 15.** *For a polynomial $F \in R[Y_1, \ldots, Y_m]$ and $y_1, \ldots, y_m \in R[U_\ell]$, for every $X \subseteq [\ell]$, let*

$$\hat{y}_{i,X} = \sum_{Y \subseteq X} y_{i,Y} T^{|Y|},$$

and $\hat{f}_X = F(\hat{y}_{1,X}, \ldots, \hat{y}_{m,X})$, let its Möbius inversion be

$$f_X = \sum_{Y \subseteq X} (-1)^{|X|-|Y|} \hat{f}_Y,$$

then we have

$$F(y_1, \ldots, y_m)_X = [T^{|X|}] f_X.$$

Proof. By linearity, it suffices to prove the case when F is a monomial. Furthermore, we can without loss of generality, assume that F is a monomial of the form

$$F(Y_1, \ldots, Y_m) = Y_1 \cdots Y_m.$$

In this case, we have

$$F(y_1, \ldots, y_m)_X = (y_1 \cdots y_m)_X = \sum_{X_1 \sqcup \cdots \sqcup X_m = X} y_{1,X_1} \cdots y_{m,X_m}.$$

On the other hand, since $\{\hat{y}_{i,X}\}_X$ is the Möbius transform of $\{y_{i,X} T^{|X|}\}_X$, and $\{f_X\}_X$ is the Möbius inversion of $\{\hat{f}_X\}_X$, the basic property of the Möbius transform gives us

$$f_X = \sum_{X_1 \cup \cdots \cup X_m = X} y_{1,X_1} \cdots y_{m,X_m} T^{|X_1|+\cdots+|X_m|}.$$

Since $X_1 \cup \cdots \cup X_m = X$, we have $|X_1| + \cdots + |X_m| \geq |X|$ and the equality is attained when $X_1, \ldots, X_m$ form a partition of X. Thus, we have

$$[T^{|X|}] f_X = \sum_{X_1 \sqcup \cdots \sqcup X_m = X} y_{1,X_1} \cdots y_{m,X_m}. \qquad \blacktriangleleft$$

In equation (1), treating $b^\ell \cdot \operatorname{haf} B^{(\ell)}$ as a polynomial in b^ℓ and all entries of $B^{(\ell)}$, we have the following corollary.

▶ **Corollary 16.** *For every $X \subseteq [\ell]$ define $\hat{h}_X^{(\ell)}, h_X^{(\ell)}$ by $\hat{h}_X^{(\ell)} = \hat{b}_X^{(\ell)} \cdot \operatorname{haf}(\hat{B}_X^{(\ell)})$ and its Möbius inversion*

$$h_X^{(\ell)} = \sum_{Y \subseteq X} (-1)^{|X|-|Y|} \hat{h}_Y^{(\ell)},$$

then we have

$$\operatorname{haf} A = [T^\ell] h_{[\ell]}^{(\ell)}.$$

Now we are ready to give the self-reduction algorithm for hafnian.

▶ **Theorem 17.** *Let $\mathbb{F}_q$ be a finite field with $q \geq (k+1)(n-k)+1$. There is an algorithm that takes a symmetric matrix $A \in \mathbb{F}_q^{2n \times 2n}$ as input, outputs $m = 2^{n-k} n^{O(1)}$ instances, consisting of $a_i \in \mathbb{F}_q$ and symmetric matrices $F_i \in \mathbb{F}_q^{2k \times 2k}$ such that*

$$\operatorname{haf}(A) = \sum_{i=1}^{m} a_i \operatorname{haf}(F_i).$$

This algorithm also runs in $2^{n-k} n^{O(1)}$ $\mathbb{F}_q$-operations.

Proof. Consider the following algorithm.

1. Iteratively compute $B^{(i)}$ and $\beta^{(i)}$ for $1 \leq i \leq n-k$ via Lemma 14.
2. Compute $b^{(n-k)} = \beta^{(1)} \cdots \beta^{(n-k)}$ via fast subset convolution [8].
3. Compute $\hat{b}_X^{(n-k)}$ and entries of $\hat{B}_X^{(n-k)}$ for all $X \subseteq [n-k]$, via fast Möbius transform [8, Section 2.2].
4. Let $D = (k+1)(n-k)$. Since $q \geq D+1$, by Lagrange interpolation, it is possible to choose points $\alpha_0, \ldots, \alpha_D \in \mathbb{F}_q$ and coefficients $\gamma_0, \ldots, \gamma_D \in \mathbb{F}_q$ such that $[T^{n-k}] f(T) = \sum_{i=0}^{D} \gamma_i f(\alpha_i)$ hold for every polynomial $f(T)$ of degree at most D. For every $X \subseteq [n-k]$ and $0 \leq i \leq D$, compute the pair of a scalar and a matrix over $\mathbb{F}_q$:

   $$\left((-1)^{n-k-|X|} \gamma_i \hat{b}_X^{(n-k)}(\alpha_i), \hat{B}_X^{(n-k)}(\alpha_i)\right)$$

   and they are the instances we need.

We now explain the correctness of the algorithm. By Corollary 16, we have

$$\begin{aligned}
\operatorname{haf} A &= [T^{n-k}] h_{[n-k]}^{(n-k)} \\
&= [T^{n-k}] \sum_{X \subseteq [n-k]} (-1)^{n-k-|X|} \hat{h}_X^{(n-k)} \\
&= \sum_{X \subseteq [n-k]} (-1)^{n-k-|X|} [T^{n-k}] \hat{h}_X^{(n-k)} \\
&= \sum_{X \subseteq [n-k]} (-1)^{n-k-|X|} [T^{n-k}] \hat{b}_X^{(n-k)} \cdot \operatorname{haf}(\hat{B}_X^{(n-k)}).
\end{aligned}$$

Since $\operatorname{haf}(\hat{B}_X^{(n-k)})$ is a polynomial of degree k in the entries of $\hat{B}_X^{(n-k)}$, $\hat{b}_X^{(n-k)} \cdot \operatorname{haf}(\hat{B}_X^{(n-k)})$ is a polynomial of degree $\leq D$ in T, so we have

$$\begin{aligned}
&\sum_{\substack{X \subseteq [n-k] \\ 0 \leq i \leq D}} (-1)^{n-k-|X|} \gamma_i \hat{b}_X^{(n-k)}(\alpha_i) \cdot \operatorname{haf}(\hat{B}_X^{(n-k)}(\alpha_i)) \\
= &\sum_{X \subseteq [n-k]} (-1)^{n-k-|X|} [T^{n-k}] \hat{b}_X^{(n-k)} \cdot \operatorname{haf}(\hat{B}_X^{(n-k)})
\end{aligned}$$

correctly computes $\operatorname{haf} A$. Each step of the algorithm requires $2^{n-k} n^{O(1)}$ $\mathbb{F}_q$-operations, and outputs $2^{n-k}(D+1) = 2^{n-k} n^{O(1)}$ many instances, each is a tuple consisting of a scalar in $\mathbb{F}_q$ and a matrix in $\mathbb{F}_q^{2k \times 2k}$. ◀

5 Data structure for hafnian

▶ **Lemma 18.** *For a commutative ring R and symmetric matrices $A, B \in R^{2n\times 2n}$, we have*

$$\operatorname{haf}(A+B) = \sum_{\substack{S\subseteq[2n]\\ |S|\equiv 0 \pmod 2}} \operatorname{haf}(B_{S,S}) \operatorname{haf}(A_{[2n]\setminus S,[2n]\setminus S}).$$

Proof. We give a combinatorial proof. The hafnian $\operatorname{haf}(A+B)$ takes the summation over perfect matchings of the complete graph K_{2n} with the product of edge weights. By expanding the product of $(A+B)_{i,j}$, this is equivalent to coloring each selected edge with one of two colors A and B, and taking the product of the weights of edges with the selected color. Hence we can first determine the vertices whose matching edges have color A and B respectively, say vertices colored by B form the set S (whose cardinality must be even). Then the contribution of such a coloring is $\operatorname{haf}(B_{S,S}) \operatorname{haf}(A_{[2n]\setminus S,[2n]\setminus S})$. ◀

▶ **Theorem 19.** *Given a symmetric matrix $A \in \mathbb{F}^{2k\times 2k}$ and a positive integer r, we can precompute in time $\tilde{O}\left(\binom{2k}{\downarrow 2r} \cdot 2^k\right)$, and answer the r-order evaluation of hafnian at A in time $\tilde{O}\left(\binom{2k}{\downarrow 2r}\right)$. Both are measured in $\mathbb{F}$-operations.*

Proof. We write $F(t) = A + B(t)$, where $B(t)$ has no constant term.

Note that when $|S| \geq 2r$, the term $\operatorname{haf}(B_{S,S})$ does not contribute to the result. Let $f(S) = \operatorname{haf}(B_{S,S})$, these $f(S)$ can be computed via dynamic programming, described as follows.

For the base case, we have $f(\varnothing) = 1$.

For $0 < |S| < 2r$ and $|S| \equiv 0 \pmod 2$, let s be a member of S, by enumerating the matching vertex v of s, we have

$$f(S) = \sum_{v\in S\setminus\{s\}} B_{s,v} f(S\setminus\{s,v\}).$$

After computing all $f(S)$, we can compute

$$\operatorname{haf}(A+B) = \sum_{\substack{S\subseteq[2n]\\ |S|\equiv 0 \pmod 2\\ |S|<2r}} f(S) g_S,$$

where $g_S = \operatorname{haf}(A_{[2n]\setminus S})$ can be precomputed via Björklund's algorithm in time $\tilde{O}(2^k)$. The precomputation time is $\binom{2k}{\downarrow 2r} \cdot \tilde{O}(2^k) = \tilde{O}\left(\binom{2k}{\downarrow 2r} \cdot 2^k\right)$, and each query takes time $\tilde{O}\left(\binom{2k}{\downarrow 2r}\right)$. ◀

Note that when $r = \alpha k$ for some $0 < \alpha < 1/2$, by Lemma 3, precomputation takes time $\tilde{O}(2^{(1+2\,\mathrm{H}(\alpha))k})$, and each query takes time $\tilde{O}(2^{2\,\mathrm{H}(\alpha)k})$.

6 Data structure for Hamiltonian cycles

In [13] they considered that Hamiltonian cycles can be counted as spanning trees with restricted degree and used it to count undirected Hamiltonian cycles in time exponential of treewidth. We give a directed version.

Let $\sigma \in S_n$ be a permutation, let P_σ denote the permutation matrix associated with σ, such that $(P_\sigma)_{i,j} = [\![j = \sigma(i)]\!]$.

▶ **Lemma 20.** *For a permutation $\sigma \in S_n$,*

$$\det\left((I - P_\sigma)_{[n]\setminus\{1\},[n]\setminus\{1\}}\right) = [\![c(\sigma) = 1]\!].$$

Proof. Consider a directed graph G with directed edges $(i, \sigma(i))$, then $L = I - P_\sigma$ is exactly the Laplacian of the graph G. By the directed version of matrix tree theorem, $\det(L_{[n]\setminus\{1\},[n]\setminus\{1\}})$ is the number of directed spanning trees rooted at vertex 1. When $c(\sigma) = 1$, then clearly there is exactly one spanning tree, otherwise there is no spanning tree. Thus we can conclude the claimed equality. ◀

Therefore, we use the above characterization of Hamiltonian cycles to help computing HamCycles.

▶ **Theorem 21.** *Given a matrix $A \in \mathbb{F}^{k\times k}$ and a positive integer r, we can precompute in $\tilde{O}\left(\binom{k}{\downarrow r}4^k\right)$, and answer the r-order evaluation of Hamiltonian cycles polynomial at A in time $\tilde{O}\left(\binom{k}{\downarrow r}^3\right)$. Both are measured in $\mathbb{F}$-operations.*

Proof. By the definition of HamCycles and Lemma 20, we have

$$\mathrm{hc}(A) = \sum_{\sigma\in S_k}\left(\prod_{i=1}^{k} A_{i,\sigma(i)}\right)\det\left((I - P_\sigma)_{[k]\setminus\{1\},[k]\setminus\{1\}}\right).$$

We also expand the determinant by the Leibniz formula, i.e.,

$$\det\left((I - P_\sigma)_{[k]\setminus\{1\},[k]\setminus\{1\}}\right) = \sum_{\substack{\tau\in S_k\\ \tau(1)=1}} \mathrm{sgn}(\tau)\prod_{i=2}^{k}(I - P_\sigma)_{i,\tau(i)}.$$

Combining the above two equations, and interpret $\mathrm{sgn}(\tau)$ as $(-1)^{\mathrm{inv}(\tau)}$, where $\mathrm{inv}(a)$ denotes the number of inversions for a sequence a, we have

$$\begin{aligned}\det\left((I - P_\sigma)_{[k]\setminus\{1\},[k]\setminus\{1\}}\right) &= \sum_{\substack{\sigma,\tau\in S_k\\ \tau(1)=1}} \mathrm{sgn}(\tau)\left(\prod_{i=1}^{k} A_{i,\sigma(i)}\right)\left(\prod_{i=2}^{k}(I - P_\sigma)_{i,\tau(i)}\right)\\ &= \sum_{\substack{\sigma,\tau\in S_k\\ \tau(1)=1}} (-1)^{\mathrm{inv}(\tau)}\left(\prod_{i=1}^{k} A_{i,\sigma(i)}\right)\left(\prod_{i=2}^{k}[\![i = \tau(i)]\!] - [\![\sigma(i) = \tau(i)]\!]\right).\end{aligned}$$

Now consider dynamic programming. For $S \subseteq [k]\setminus\{1\}, T \subseteq [k]$ and say $s = |S| = |T|$, let $f(S, T)$ only counts in the last s values of σ and τ, with domain $\{\tau(k-s+1), \ldots, \tau(k)\} = S$ and $\{\sigma(k-s+1), \ldots, \sigma(k)\} = T$, and the inversions of τ in the last s values are counted, i.e.,

$$f(S,T) = \sum_{\sigma,\tau}(-1)^{\mathrm{inv}(\tau)}\left(\prod_{i=k-s+1}^{k} A_{i,\sigma(i)}\right)\left(\prod_{i=k-s+1}^{k}[\![i = \tau(i)]\!] - [\![\sigma(i) = \tau(i)]\!]\right). \tag{2}$$

We let $a \leftarrow^+ b$ denote $a \leftarrow a + b$ for simplicity in describing the updating rules. The base case is simply $f(\varnothing, \varnothing) = 1$, and for each $s < k - 1$, we use the computed values of $f(S, T)$ with $|S| = |T| = s$ to compute $f(S, T)$ with $|S| = |T| = s + 1$ by the following rules. Let $i = k - s$. For each $j \notin T$, we can choose $\sigma(i)$ to be j, then there are two choices of $\tau(i)$:

- If $i \notin S$, update with

$$f(S \cup \{i\}, T \cup \{j\}) \leftarrow^{+} (-1)^{\mathrm{inv}(i,S)} A_{i,j} f(S,T),$$

denoting the choice that the contribution of term $[\![i = \tau(i)]\!]$ in equation (2).

- If $j \notin S$, update with

$$f(S \cup \{j\}, T \cup \{j\}) \leftarrow^{+} (-1)^{1+\mathrm{inv}(j,S)} A_{i,j} f(S,T),$$

denoting the choice that the contribution of term $[\![\sigma(i) = \tau(i)]\!]$ in equation (2).

Here $\mathrm{inv}(v, S)$ means the number of elements $x \in S$ such that $v > x$.

Finally, we have the choice of $\sigma(1)$, thus

$$\mathrm{hc}(A) = \sum_{i=1}^{k} A_{1,i} f([k] \setminus \{1\}, [k] \setminus \{i\}).$$

This dynamic programming takes $\tilde{O}(4^k)$, which is slower than the usual one, but its dependence on the rows of A is explicitly graded by s, so is useful for our purpose.

Now suppose the first j rows are left undetermined, we can first preprocess all the $f(S,T)$ for $s \le k - j$ in time $\tilde{O}(4^k)$, since their value does not depend on the first j rows. Then for each query, i.e., given the first j rows, can be computed in time

$$\tilde{O}\left(\sum_{i=1}^{j} \binom{k}{i}\binom{k}{i-1}\right) = \tilde{O}\left(\binom{k}{\downarrow j}^2\right).$$

Write $F = A + B(t)$, where $B(t)$ has no constant term, by the multilinearity on rows of $\mathrm{hc}(\cdot)$, we have

$$\mathrm{hc}(F(t)) = \mathrm{hc}(A + B) = \sum_{S \subseteq [k]} \mathrm{hc}(\mathrm{rep}_S(A, B)),$$

where $\mathrm{rep}_S(A, B)$ denote the matrix obtained by replacing the rows indexed in S of A by those rows of B. The terms $|S| \ge r$ do not contribute to the result. For each $|S| < r$, we can reorder the rows and columns simultaneously to make S be the first $|S|$ rows, and use the above dynamic programming to do precomputation and handle queries.

There are $\binom{k}{\downarrow r}$ ways to choose S, so the precomputation needs $\tilde{O}\left(\binom{k}{\downarrow r} 4^k\right)$ time, and $\tilde{O}\left(\binom{k}{\downarrow r}^3\right)$ for each query. ◀

Note that when $r = \alpha k$ for some $0 < \alpha < 1/2$, by Lemma 3, precomputation takes time $\tilde{O}(2^{(2+\mathrm{H}(\alpha))k})$, and each query takes time $\tilde{O}(2^{3\,\mathrm{H}(\alpha)k})$.

7 The algorithms

We first prove Theorem 1 under some restrictions, and then remove the restrictions by bootstrapping the results.

▶ **Lemma 22.** *Let q satisfy $q \ge n^2 + 1$ and $q \equiv 1 \pmod b$, where $b \ge 10$. There is an algorithm that computes the hafnian $\mathrm{haf}(A)$ of a given symmetric matrix $A \in \mathbb{F}_q^{2n \times 2n}$ in time $2^{n - \delta_b \sqrt{n}} q^{O(1)}$, for some $\delta_b > 0$.*

Proof. Let $\theta = \sqrt{\log(1.9)/\log(1+b)}$ and $k = \lfloor \theta\sqrt{n} \rfloor$, and consider the following algorithm.

1. First compute the Kakeya set by Theorem 10 over k^2 variables of degree $u = (q-1)/b - 1$.
2. Precompute the data structure for r-order evaluation for $r = \lceil k/b \rceil$ at each point of K.
3. Use the self-reduction of hafnian (Theorem 17) to reduce the problem to $m = 2^{n-k} n^{O(1)}$ instances of size $2k \times 2k$.
4. For each instance, use Theorem 12 to compute the hafnian.

Then we analyze the time complexity. In the precomputation phase, by Theorem 10, the size of Kakeya set is $(\frac{q-1}{u+1} + 1)^{k^2+1} \leq (b+1)^{\theta^2 n + 1} = O(1.9^n)$, and by Theorem 19, each data structure takes $2^{O(k)}$ time to precompute, so the total time of the first two steps is $1.9^{n+O(\sqrt{n})} q^{O(1)}$.

The data structure can answer r-order evaluation in time $\tilde{O}(2^{2\,\mathrm{H}(\alpha)k})$. Here we have $\alpha = 1/b \leq 0.1$, hence $2\,\mathrm{H}(\alpha) \leq 2\,\mathrm{H}(0.1) < 0.94$, the total time in last two steps is

$$2^{n-k} n^{O(1)} \cdot O(2^{0.94k}) q^{O(1)} = 2^{n-0.06k} q^{O(1)} = 2^{n-0.06\theta\sqrt{n}} q^{O(1)}.$$

In conclusion, we have $\delta_b = 0.06\theta$ satisfies the requirement. ◀

▶ **Lemma 23.** *Let q satisfy $q \geq n^2 + 1$ and $q \equiv 1 \pmod b$, where $b \geq 17$. There is an algorithm that computes Hamiltonian cycles $\mathrm{hc}(A)$ of a given matrix $A \in \mathbb{F}_q^{n \times n}$ in time $2^{n-\delta_b\sqrt{n}} q^{O(1)}$, for some $\delta_b > 0$.*

Proof. The algorithm is similar to the proof of Lemma 22, with replacing the data structure for Hamiltonian cycles instead of hafnian.

By Theorem 21, the data structure can answer r-order evaluation of Hamiltonian cycles in time $\tilde{O}(2^{3\,\mathrm{H}(\alpha)k})$, where $3\,\mathrm{H}(\alpha) \leq 3\,\mathrm{H}(1/17) < 0.97$.

Then the total time in the last two steps is

$$2^{n-k} n^{O(1)} \cdot O(2^{0.97k}) q^{O(1)} = 2^{n-0.03k} q^{O(1)} = 2^{n-0.03\theta\sqrt{n}} q^{O(1)}.$$

In conclusion, we have $\delta_b = 0.03\theta$ satisfies the requirement. ◀

7.1 Proof of Theorem 1

To prove Theorem 1, we only need to remove the conditions of Lemma 22 and Lemma 23 on q that $q \geq n^2 + 1$ and $q \equiv 1 \pmod b$ for some fixed modulus b.

Note that for some integer ℓ, we can embed $\mathbb{F}_q$ into a larger finite field $\mathbb{F}_{q^\ell}$. We only need to satisfy $q^\ell \geq n^2 + 1$ and $q^\ell \equiv 1 \pmod b$. When q is coprime with b, taking $\ell = \varphi(b)$ is enough to satisfy the second condition, where φ is the Euler totient function. Taking ℓ as the smallest multiple of $\varphi(b)$ such that $q^\ell > n^2$, we have $q^\ell \leq q^{\varphi(b)} n^2$.

For the hafnian, since q is a prime power, it must be coprime with either $b = 10$ or $b = 11$. For Hamiltonian cycles, q must be coprime with either $b = 17$ or $b = 18$. Therefore, we have $q^\ell = q^{O(1)} n^2$ since we only consider finite possibilities for b.

Therefore, by invoking the algorithms in Lemma 22 and Lemma 23 through the finite field $\mathbb{F}_{q^\ell}$, we can compute the hafnian and Hamiltonian cycles in time $2^{n-\Omega(\sqrt{n})} (q^\ell)^{O(1)} = 2^{n-\Omega(\sqrt{n})} q^{O(1)}$.

To actually support the computation in the finite field $\mathbb{F}_{q^\ell}$, we need to find an irreducible polynomial f and identify $\mathbb{F}_{q^\ell}$ as $\mathbb{F}_q[t]/(f)$. We can enumerate the polynomials of degree ℓ over $\mathbb{F}_q$ and test whether they satisfy the conditions. By [26, Theorem 14.37], the time complexity of testing irreducibility is $\mathrm{poly}(\ell, \log q)$. The time required to find an irreducible polynomial is $O(q^\ell \,\mathrm{poly}(\log q^\ell))$, so this is not a bottleneck. ◀

7.2 Proof of Corollary 2

The absolute value of $\mathrm{haf}(A)$ and $\mathrm{hc}(A)$ is trivially bounded by $C = (2n)!M^n$. Let $p_1, \ldots, p_r$ be distinct prime numbers such that $D := \prod_i p_i > 2C + 1$. Then if we can compute $\mathrm{haf}(A)$ and $\mathrm{hc}(A)$ modulo D, the values of $\mathrm{haf}(A)$ and $\mathrm{hc}(A)$ are uniquely determined.

By the Chinese remainder theorem, we only need to compute $\mathrm{haf}(A)$ and $\mathrm{hc}(A)$ modulo p_i for each i, and then combine them to get the result modulo D.

By Lemma 7, the primes not greater than $16 \log D = O(n \log M)$ have their product greater than D. So we only need to compute $\mathrm{haf}(A)$ and $\mathrm{hc}(A)$ over finite fields $\mathbb{F}_q$ with $p = O(n \log M)$. By Theorem 1, we can compute them in time $2^{n-\Omega(\sqrt{n})}p^{O(1)} = 2^{n-\Omega(\sqrt{n})}(\log M)^{O(1)}$. There are $O(n \log M)$ instances to compute. Since the product of the chosen primes has $O(n \log M)$ bits, by Theorem 6, it takes $\mathrm{poly}(n \log M)$ time to combine them, which is not a bottleneck. So the total time is $2^{n-\Omega(\sqrt{n})}(\log M)^{O(1)}$. ◀

References

1 Eric Bax and Joel Franklin. A finite-difference sieve to count paths and cycles by length. *Inform. Process. Lett.*, 60(4):171–176, 1996. doi:10.1016/S0020-0190(96)00159-7.

2 Vishwas Bhargava, Sumanta Ghosh, Zeyu Guo, Mrinal Kumar, and Chris Umans. Fast multivariate multipoint evaluation over all finite fields. In *Proceedings of the 63rd annual IEEE symposium on foundations of computer science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 221–232. Los Alamitos, CA: IEEE Computer Society, 2022. doi:10.1109/FOCS54457.2022.00028.

3 Vishwas Bhargava, Sumanta Ghosh, Zeyu Guo, Mrinal Kumar, and Chris Umans. Fast multivariate multipoint evaluation over all finite fields. *J. ACM*, 71(3):Art. 22, 32, 2024. doi:10.1145/3652025.

4 Vishwas Bhargava, Sumanta Ghosh, Mrinal Kumar, and Chandra Kanta Mohapatra. Fast, algebraic multivariate multipoint evaluation in small characteristic and applications. In *Proceedings of the 54th annual ACM SIGACT symposium on theory of computing, STOC '22, Rome, Italy June 20–24, 2022*, pages 403–415. New York, NY: Association for Computing Machinery (ACM), 2022. doi:10.1145/3519935.3519968.

5 Andreas Björklund. Counting perfect matchings as fast as Ryser. In *Proceedings of the twenty-third annual acm-siam symposium on discrete algorithms*, pages 914–921. SIAM, 2012. doi:10.1137/1.9781611973099.73.

6 Andreas Björklund. Determinant sums for undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

7 Andreas Björklund. Below all subsets for some permutational counting problems. In *15th Scandinavian symposium and workshops on algorithm theory, SWAT 2016, Reykjavik, Iceland, June 22–24, 2016. Proceedings*, page 11. Wadern: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. Id/No 17. doi:10.4230/LIPIcs.SWAT.2016.17.

8 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 67–74, 2007. doi:10.1145/1250790.1250801.

9 Andreas Björklund, Thore Husfeldt, and Isak Lyckberg. Computing the permanent modulo a prime power. *Inform. Process. Lett.*, 125:20–25, 2017. doi:10.1016/j.ipl.2017.04.015.

10 Andreas Björklund, Petteri Kaski, and Ioannis Koutis. Directed Hamiltonicity and out-branchings via generalized Laplacians. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 91:1–91:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2017.91.

11 Andreas Björklund, Petteri Kaski, and Ryan Williams. Generalized Kakeya sets for polynomial evaluation and faster computation of fermionants. *Algorithmica*, 81(10):4010–4028, 2019. doi:10.1007/s00453-018-0513-7.

12 Andreas Björklund and Ryan Williams. Computing permanents and counting hamiltonian cycles by listing dissimilar vectors. In *46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 25, 14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.25.

13 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inform. and Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

14 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):Art. 12, 46, 2018. doi:10.1145/3148227.

15 Marek Cygan and Marcin Pilipczuk. Faster exponential-time algorithms in graphs of bounded average degree. *Inform. and Comput.*, 243:75–85, 2015. doi:10.1016/j.ic.2014.12.007.

16 Zeev Dvir. On the size of Kakeya sets in finite fields. *J. Amer. Math. Soc.*, 22(4):1093–1097, 2009. doi:10.1090/S0894-0347-08-00607-3.

17 Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to Kakeya sets and mergers. *SIAM J. Comput.*, 42(6):2305–2328, 2013. doi:10.1137/100783704.

18 Fedor V. Fomin and Dieter Kratsch. *Exact exponential algorithms*. Texts Theor. Comput. Sci., EATCS Ser. Berlin: Springer, 2010. doi:10.1007/978-3-642-16533-7.

19 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, The IBM Research Symposia Series, pages 85–103. Plenum, New York-London, 1972. doi:10.1007/978-1-4684-2001-2_9.

20 Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, 2011. doi:10.1137/08073408X.

21 Donald E. Knuth. *The art of computer programming. Vol. 2: Seminumerical algorithms*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1969.

22 Gerd Mockenhaupt and Terence Tao. Restriction and Kakeya phenomena for finite fields. *Duke Math. J.*, 121(1):35–74, 2004. doi:10.1215/S0012-7094-04-12112-8.

23 Herbert John Ryser. *Combinatorial mathematics*, volume No. 14 of *The Carus Mathematical Monographs*. Mathematical Association of America, distributed by John Wiley and Sons, Inc., New York, 1963. doi:10.5948/UPO9781614440147.

24 L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

25 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979. doi:10.1137/0208032.

26 Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, third edition, 2013. doi:10.1017/CBO9781139856065.