Counting All Lattice Rectangles in the Square Grid in Near-Linear Time

Dmitry Babichev
Paris, France

Sergey Babichev
Paris, France

Abstract

We study the exact counting problem for all lattice rectangles contained in the square $[0, n) \times [0, n)$, including non-axis-parallel ones. Starting from the standard parametrization by a primitive direction (u, v) and two side lengths, we derive a sequence of exact algorithms of complexity $O(n^2)$, $O(n^{3/2} \log n)$, $O(n^{4/3} \log n)$, and finally $O(n \log^3 n)$. The main idea behind the near-linear algorithm is to reduce the geometric summation to a constant-size family of weighted floor sums closed under Euclidean-style affine and reciprocal transformations, and hence evaluable in $O(\log n)$ time per query. The intermediate algorithms expose the structural reductions leading to this final kernel and provide independent cross-checks for the implementation.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms; Mathematics of computing → Discrete mathematics

Keywords and phrases Lattice rectangles, grid enumeration, floor sums, Möbius inversion

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.26

Related Version *Full Version*: `https://arxiv.org/abs/2604.22456`

Supplementary Material *Software (Source Code)*: `https://github.com/flykiller/lattice-rectangles`, archived at `swh:1:dir:b09ae306bd57299279c92e1d9250fcc4a52ed131`

1 Introduction

We study the following problem.

Problem

Count all rectangles whose vertices lie in the integer lattice and are contained in the square $[0, n) \times [0, n)$, including non-axis-parallel rectangles.

Counting lattice configurations in planar grids is a classical topic at the interface of combinatorics, geometry of numbers, and lattice-point enumeration. In this paper we study an exact counting problem for lattice rectangles. From an algorithms perspective, this is a compact test case for a common phenomenon in exact evaluation of structured integer sequences: a direct geometric summation has too many primitive objects, but after divisor-sum rearrangements and floor-sum reciprocity it collapses to a small number of Euclidean one-dimensional kernels.

The resulting counting function is the OEIS sequence A085582 [8]. Geometrically, it extends the classical axis-parallel rectangle count to arbitrary lattice orientations in the same $n \times n$ grid of points. Arithmetically, each non-axis-parallel family is governed by a primitive direction (u, v), which brings in coprimality, divisor sums, and floor-sum recurrences. Computationally, the OEIS entry already collects exact values, tables, and scripts, so faster exact algorithms are useful both for extending the data and for testing asymptotic predictions.

Table 1 Summary of the algorithms discussed in the paper.

Method	Main idea	Time	Authors	Kernel
Baseline	primitive-direction enumeration	$O(n^2)$	Radcliffe	none
Square-root	small/large split; coprime prefixes	$O(n^{3/2}\log n)$	this paper	coprime prefixes
Cubic-root	dual parametrization; six moments	$O(n^{4/3}\log n)$	this paper	six-moment weighted floor-sum
Final	Möbius inversion; weighted floor sums	$O(n\log^3 n)$	this paper	ten-moment weighted floor-sum

The classical split appears in the OEIS decomposition $A085582(n) = A000537(n-1) + A113751(n)$, where $A000537(n-1) = \binom{n}{2}^2$ is the axis-parallel count and $A113751(n)$ records the non-axis-parallel contribution [8]. Our goal is to show that the full sequence is not merely a table of values, but a structured counting problem admitting several increasingly efficient exact algorithms.

We are not aware of prior work on this exact enumeration problem in the present algorithmic form. Our main contribution is an algorithmic reduction from geometric enumeration to a constant-size family of weighted Euclidean floor-sum kernels, which yields the complexity improvements proved below. For related background on floor-sum reciprocity, elementary number-theoretic estimates, and lattice-point enumeration, see [4, Section 3.5] and [5, 3, 2, 12]. The use of primitive lattice directions also connects the problem to the broader literature on primitive lattice points in planar domains and polygonal counting problems [6, 9]. The last line of work is algorithmically close in spirit: for example, Pawlewicz and Pătraşcu obtain sublinear-time algorithms for Farey order statistics and related primitive-lattice-point counting in polygons [9].

Our main contributions are as follows: We derive three progressively faster exact algorithms beyond the quadratic baseline, with complexities $O(n^{3/2}\log n)$, $O(n^{4/3}\log n)$, and $O(n\log^3 n)$. In particular, an exact $O(n\log^3 n)$ algorithm achieves near-linear time. Methodologically, the final algorithm is obtained by reducing the original geometric counting problem to a constant-size recursively closed family of weighted floor-sum kernels; the Euclidean affine and reciprocal transformations preserve this family, so each outer query can be evaluated in $O(\log n)$ arithmetic operations. See Table 1 for more details.

The intermediate algorithms are included because they expose the structural reductions leading to the final near-linear method: the $O(n^{3/2}\log n)$ split introduces the direction/side-length duality, the $O(n^{4/3}\log n)$ algorithm isolates a finite floor-moment state space, and the final $O(n\log^3 n)$ algorithm emerges from the resulting recursively closed weighted floor-sum kernel. A full version contains the longer ten-moment transition identities, a two-term asymptotic expansion, larger exact-value tables, and the full ten-moment coefficient list.

Throughout the paper we count arithmetic operations in a standard RAM model. In the implementation, all loop indices and intermediate affine parameters are stored in `std::int128_t`, which is exact throughout the benchmarked range reported in Section 8. Accordingly, the stated running times should be read as arithmetic-operation bounds rather than bit-complexity bounds for arbitrarily large integers. The recursive floor-sum kernels have $O(1)$ state size; the remaining memory usage comes from the preprocessing tables.

2 Preliminaries

We use the following standard arithmetic functions. The Möbius function is $\mu(1) = 1$, $\mu(m) = 0$ if a square prime divides m, and otherwise $\mu(m) = (-1)^r$, where r is the number of distinct prime divisors of m. Euler's totient $\varphi(m)$ counts residues coprime to m; $\tau(m)$ is the divisor-counting function, and $\omega(m)$ is the number of distinct prime divisors. The Möbius identity used to remove coprimality is $\mathbf{1}_{\gcd(r,s)=1} = \sum_{d\mid r,\ d\mid s} \mu(d)$, hence, for any function f, $\sum_{1\leqslant s\leqslant X,\ \gcd(r,s)=1} f(s) = \sum_{d\mid r} \mu(d) \sum_{m\leqslant X/d} f(dm)$. We also use the elementary estimates $\sum_{m\leqslant X} 1/m = O(\log X)$, $\sum_{m\leqslant X} \tau(m)/m^2 = O(\log X)$, and $\sum_{m\leqslant X} 2^{\omega(m)}/m = O(\log^2 X)$. The floor-sum kernels below are evaluated by Euclidean recursion: after reducing affine parameters modulo the denominator, a reciprocal step exchanges numerator and denominator.

3 The standard parametrization

This section fixes the basic parametrization and separates the easy axis-parallel and boundary contributions from the primitive non-axis-parallel part.

Let (u, v) be a primitive lattice direction, so $u \geqslant v > 0$ and $\gcd(u, v) = 1$. The orthogonal direction is $(-v, u)$. A rectangle with side lengths $a, b \geqslant 1$ in these two directions is generated by $a \cdot (u, v)$ and $b \cdot (-v, u)$, hence its axis-aligned bounding box has side lengths $au + bv$ and $av + bu$. Therefore it fits into the half-open square $[0, n) \times [0, n)$ if and only if

$$au + bv \leqslant n, \qquad av + bu \leqslant n. \tag{1}$$

For such a quadruple (u, v, a, b), the number of placements equals

$$(n - au - bv)(n - av - bu). \tag{2}$$

We split $F(n) = F_0(n) + F_1(n)$. Here $F_0(n)$ collects the axis-parallel orientations together with the boundary direction $(1, 1)$, while $F_1(n)$ contains all primitive directions with $0 < v < u$. The axis-parallel term is the classical rectangular-grid count, the direction $(1, 1)$ is the simplest genuinely tilted family, and all remaining work is concentrated in the primitive non-axis-parallel directions. These cases are separated to avoid double counting and to isolate the boundary direction $(1, 1)$. The axis-parallel rectangles contribute $\binom{n}{2}^2 = n^2(n-1)^2/4$. For the boundary direction $(1, 1)$, a rectangle generated by $a(1, 1)$ and $b(-1, 1)$ has bounding-box side length $a+b$, hence it contributes $(n-a-b)^2$ placements; summing $\sum_{a,b\geqslant 1,\ a+b\leqslant n}(n-a-b)^2$ gives $n(n-1)^2(n-2)/12$. Hence

$$F_0(n) = \frac{n^2(n-1)^2}{4} + \frac{n(n-1)^2(n-2)}{12}.$$

For $F_1(n)$, by symmetry in a, b, set $x := \max(a, b)$ and $y := \min(a, b)$, so $x \geqslant y \geqslant 1$. The constraints become $xu + yv \leqslant n$ and $xv + yu \leqslant n$, and the second inequality is redundant because $x \geqslant y$ and $u > v$. Thus the two orderings of (a, b) collapse into one term; swapping (u, v) with (v, u) only produces the reflected direction, excluded by the convention $u > v > 0$. Thus we get

$$F_1(n) = \sum_{\substack{u>v>0,\ x\geqslant y\geqslant 1\\ \gcd(u,v)=1,\ xu+yv\leqslant n}} \mathcal{W}(x, y)\,(n - xu - yv)(n - xv - yu), \tag{3}$$

where $\mathcal{W}(x, y) = 2$ if $x = y$ and $\mathcal{W}(x, y) = 4$ if $x > y$. This identity will be the common starting point for the reparametrizations below.

4 The classical quadratic algorithm

We record the natural $O(n^2)$ baseline obtained by sweeping over primitive directions and summing admissible side-length pairs [8, 10].

Direction sweep

Fix a primitive direction (u, v) with $u > v$. For each admissible y, the variable x runs through an interval determined by Equation (3). A straightforward implementation computes the contribution of one primitive direction in time $O(n/u)$, or in time $O(n/u \cdot \log n)$ if one performs coprimality tests online and evaluates the inner arithmetic progressions without the closed-form simplifications.

The number of primitive pairs with first coordinate u is $\varphi(u) + O(1)$, where φ is Euler's totient function, so using the classical estimate $\sum_{u \leqslant n} \frac{\varphi(u)}{u} = O(n)$ we get:

$$\sum_{u \leqslant n} \varphi(u) \frac{n}{u} = O(n^2), \qquad \sum_{u \leqslant n} \varphi(u) \frac{n \log n}{u} = O(n^2 \log n),$$

▶ **Proposition 1** (classical baseline). *The rectangle-counting problem admits*

- *a direct implementation of complexity $O(n^2 \log n)$ if primitivity and inner summation are handled naively, and*
- *an $O(n^2)$ implementation after standard preprocessing of primitive directions and elementary arithmetic simplifications.*

Algorithmically, the classical baseline sweeps over primitive directions (u, v) and, for each such direction, over the admissible values of the secondary side length parameter; for fixed (u, v, y), the remaining interval in x is summed explicitly. We state this baseline explicitly because every later improvement modifies exactly one bottleneck of this sweep. Compare the standard divisor-summation and hyperbola-method viewpoints in [7, 11]; a more implementation-oriented version is recorded in Appendix A. For readers coming from the OEIS entry, this is also the closest formalization of the straightforward computation behind the existing tables and scripts [10].

5 A square-root decomposition: $O(n^{3/2} \log n)$

Let $B := \lfloor \sqrt{n} \rfloor$. We split the set of primitive directions into two regions

$$\mathcal{D}_{\text{small}} = \{(u, v) : \gcd(u, v) = 1,\ u > v > 0,\ u \leqslant B\},$$

$$\mathcal{D}_{\text{large}} = \{(u, v) : \gcd(u, v) = 1,\ u > v > 0,\ u > B\}.$$

5.1 Small directions

For $(u, v) \in \mathcal{D}_{\text{small}}$, we keep the classical parametrization (u, v, x, y). The contribution of one direction is computed in time $O(n/u \cdot \log n)$, and hence the total cost is

$$\sum_{u \leqslant B} \varphi(u) \frac{n \log n}{u} = O(nB \log n) = O(n^{3/2} \log n).$$

5.2 Large directions

If $(u, v) \in \mathcal{D}_{\text{large}}$, then $u > B$, hence every admissible rectangle has $x \leqslant n/u < n/B$. Therefore we switch the order of summation and use the parameterization from Section 3. Restricting the formula for $F_1(n)$ to $\mathcal{D}_{\text{large}}$, we obtain

$$F_{1,\text{large}}(n) = \sum_{\substack{(u,v)\in\mathcal{D}_{\text{large}} \\ x\geqslant y\geqslant 1,\ xu+yv\leqslant n}} \mathcal{W}(x,y)\,(n-xu-yv)(n-xv-yu).$$

Fix u, x, y and set

$$v_{\max} := \min\left(u-1, \left\lfloor \frac{n-xu}{y} \right\rfloor\right).$$

Then the inner summation runs over all $v \in [1, v_{\max}]$ with $\gcd(u, v) = 1$, and the placement factor is a quadratic polynomial in v:

$$(n-xu-yv)(n-xv-yu) = A_0(u;x,y) + A_1(u;x,y)\,v + A_2(x,y)\,v^2,$$

where

$$A_0(u;x,y) = (n-xu)(n-yu), \qquad A_1(u;x,y) = -(x(n-xu)+y(n-yu)), \qquad A_2(x,y) = xy.$$

Hence for fixed u, x, y we need only the three coprime prefix sums

$$C_j(u;X) := \sum_{\substack{1\leqslant v\leqslant X \\ \gcd(u,v)=1}} v^j, \qquad j \in \{0,1,2\}.$$

Indeed, applying the Möbius identity from Section 2 gives $\mathbf{1}_{\gcd(u,v)=1} = \sum_{d\mid u,\ d\mid v} \mu(d)$, and therefore

$$C_j(u;X) = \sum_{d\mid u} \mu(d) \sum_{m\leqslant X/d} (dm)^j.$$

Consequently the v-sum becomes

$$\sum_{\substack{1\leqslant v\leqslant v_{\max} \\ \gcd(u,v)=1}} (n-xu-yv)(n-xv-yu) = A_0C_0(u;v_{\max}) + A_1C_1(u;v_{\max}) + A_2C_2(u;v_{\max}).$$

So the whole large-direction contribution is

$$F_{1,\text{large}}(n) = \sum_{u>B} \sum_{x\leqslant n/u} \sum_{y\leqslant x} \mathcal{W}(x,y)\big(A_0C_0(u;v_{\max}) + A_1C_1(u;v_{\max}) + A_2C_2(u;v_{\max})\big).$$

For each fixed u, the number of pairs (x, y) is $O((n/u)^2)$, and one coprime-prefix query is answered in $O(\tau(u))$ time from the squarefree divisors of u. Summing over $u > B$ gives

$$\sum_{u>B} O\left(\tau(u)\left(\frac{n}{u}\right)^2\right) = O\left(n^2 \sum_{u>B} \frac{\tau(u)}{u^2}\right) = O\left(\frac{n^2 \log n}{B}\right).$$

▶ **Theorem 2** (square-root decomposition). *Choosing $B = \lfloor\sqrt{n}\rfloor$ and treating small directions and large directions by dual parametrizations yields an algorithm of complexity $O(n^{3/2}\log n)$.*

At the algorithmic level, one chooses the threshold $B = \lfloor\sqrt{n}\rfloor$, evaluates the small-direction range $u \leqslant B$ by the one-direction routine from the previous section, and evaluates the complementary range by the dual sweep in the (x, y) variables. Appendix A records a fuller pseudocode version of this decomposition.

6 A cubic-root decomposition: $O(n^{4/3} \log n)$

The next refinement follows the same philosophy as the square-root split, but with a stronger one-direction kernel. Let

$$B := \left\lfloor n^{2/3} \right\rfloor .$$

We again split the set of primitive directions at the threshold $u = B$.

6.1 Small directions: one primitive pair in $O(\log n)$

Fix a primitive direction (u, v) with

$$u > v \geqslant 1, \qquad \gcd(u, v) = 1.$$

Denote by $c_{u,v}(n)$ its total contribution in the parametrization of (3):

$$c_{u,v}(n) := \sum_{\substack{x \geqslant y \geqslant 1 \\ xu+yv \leqslant n}} \mathcal{W}(x, y)\,(n - xu - yv)(n - xv - yu),$$

We sum over x, with y as the inner variable. For fixed x, the admissible values of y satisfy

$$1 \leqslant y \leqslant \min(x, M(x)), \text{ where } M(x) = \left\lfloor \frac{n - xu}{v} \right\rfloor$$

Thus the x-range splits at the transition point $x = \frac{n}{u+v}$. Therefore we obtain two segments:

$$I_1 = \left[1, \left\lfloor \frac{n}{u+v} \right\rfloor\right], \qquad I_2 = \left[\left\lfloor \frac{n}{u+v} \right\rfloor + 1, \left\lfloor \frac{n}{u} \right\rfloor\right].$$

On the first segment I_1, the bound coming from $xu + yv \leqslant n$ is inactive, so $1 \leqslant y \leqslant x$. On the second segment I_2, the active upper bound is $1 \leqslant y \leqslant \left\lfloor \frac{n-xu}{v} \right\rfloor$. Thus $c_{u,v}(n)$ is the sum of two segment contributions.

On I_1, we separate the diagonal $y = x$ from the strict range $1 \leqslant y < x$. The diagonal contribution is computable in $O(1)$ time.

On I_2, the upper bound for y is $M(x)$, and since $M(x) < x$ throughout this segment, we always have $x > y$, hence $\mathcal{W}(x, y) = 4$. Expanding

$$(n - xu - yv)(n - xv - yu) = P_0(x) + P_1(x)y + P_2 y^2,$$

where $P_0(x)$ is quadratic in x, $P_1(x)$ is linear in x, and P_2 is constant, and then carrying out the inner summation over y, we obtain a polynomial in x and $M(x)$. More precisely, the total contribution of I_2 is a fixed linear combination of the six floor moments

$$\sum_{x \in I_2} M(x), \quad \sum_{x \in I_2} xM(x), \quad \sum_{x \in I_2} x^2 M(x), \quad \sum_{x \in I_2} M(x)^2, \quad \sum_{x \in I_2} xM(x)^2, \quad \sum_{x \in I_2} M(x)^3. \tag{4}$$

To formalize the required queries, for integers $N \geqslant 0$, $m \geqslant 1$, and $a, b \geqslant 0$, define the six-moment kernel

$$\mathcal{H}_{p,q}(N; m, a, b) := \sum_{t=0}^{N-1} t^p \left\lfloor \frac{at + b}{m} \right\rfloor^q, \quad (p, q) \in \{(0,1), (1,1), (2,1), (0,2), (1,2), (0,3)\}. \tag{5}$$

The interval sums appearing in (4) have negative slope in the summation variable, so before invoking the kernel we first reverse the index.

▶ **Lemma 3** (sign reversal for floor moments). *Let $m \geqslant 1$, $u \geqslant 0$, $B \in \mathbb{Z}$, and $L \leqslant R$. Put $N := R - L + 1$. Then for every $p \geqslant 0$ and $q \geqslant 1$,*

$$\sum_{x=L}^{R} x^p \left\lfloor \frac{B - ux}{m} \right\rfloor^q = \sum_{j=0}^{p} (-1)^j \binom{p}{j} R^{p-j} \mathcal{H}_{j,q}(N; m, u, B - uR).$$

Proof. Let $x = R - t$, where $0 \leqslant t \leqslant R - L$. Then

$$\left\lfloor \frac{B - ux}{m} \right\rfloor = \left\lfloor \frac{B - uR + ut}{m} \right\rfloor, \qquad x^p = (R - t)^p = \sum_{j=0}^{p} (-1)^j \binom{p}{j} R^{p-j} t^j.$$

Substituting and exchanging the finite sums gives the claim. ◀

In particular, each of the six sums in (4) is a linear combination of six-moment kernel values with nonnegative slope $a = u$; these six direct states are already recursively closed, since the affine and reciprocal Euclidean steps do not increase the total weighted degree $p + q$, so the family with $q \geqslant 1$ and $p + q \leqslant 3$ is stable. The common recursive structure is summarized in Appendix B. The reciprocal step used below is analogous to the classical conjugation/reciprocity viewpoint in integer-partition decompositions [1].

▶ **Lemma 4** (affine closure of the six-moment kernel). *Each affine normalization step expresses a state $\mathcal{H}_{p,q}(N; m, a, b)$ as a linear combination of states $\mathcal{H}_{p',q'}(N; m, a', b')$ inside the same six-state family, together with ordinary polynomial sums.*

Proof. Write the floor argument as $q_1 t + q_0 + g(t)$ after removing the easy Euclidean quotients. Expanding $(q_1 t + q_0 + g(t))^q$ shows that every nontrivial term still has the form $t^{p'} g(t)^{q'}$ with $(p', q') \in \{(0,1), (1,1), (2,1), (0,2), (1,2), (0,3)\}$, while the terms with $q' = 0$ are ordinary polynomial sums. The coefficient identities are obtained by this expansion; the full expanded list is left to the implementation and full version. ◀

▶ **Lemma 5** (reciprocal closure of the six-moment kernel). *Under the reciprocal Euclidean step, each state of the six-moment family is expressible as a linear combination of states of the same family and polynomial sums.*

Proof. Write

$$\mathcal{H}_{p,q}(N; m, a, b) = \sum_{x=0}^{N-1} x^p y(x)^q, \qquad y(x) := \left\lfloor \frac{ax + b}{m} \right\rfloor,$$

and assume $0 \leqslant a, b < m$. Put

$$Y := \left\lfloor \frac{a(N-1) + b}{m} \right\rfloor.$$

Instead of summing by columns indexed by x, we regroup the same staircase region by horizontal levels $t = 0, 1, \ldots, Y - 1$. The condition $y(x) \geqslant t + 1$ is equivalent to

$$x \geqslant \left\lfloor \frac{mt + (m - b - 1)}{a} \right\rfloor + 1.$$

Hence the transposed staircase is encoded by the reciprocal floor function

$$g(t) := \left\lfloor \frac{mt + (m - b - 1)}{a} \right\rfloor, \qquad 0 \leqslant t < Y,$$

which is exactly the same construction with the roles of a and m interchanged. When one rewrites the sums over the horizontal strips, the lower bounds contribute only polynomial weights in t, so every term becomes a linear combination of moments

$$\sum_{t=0}^{Y-1} t^{p'} g(t)^{q'} = \mathcal{H}_{p',q'}(Y; a, m, m-b-1),$$

with $(p', q') \in \{(0,1),(1,1),(2,1),(0,2),(1,2),(0,3)\}$, together with ordinary power sums. Thus the same six-state family is preserved under the reciprocal step, and the larger Euclidean parameter decreases from m to $a < m$. The coefficient identities are obtained by this transposition; the full expanded list is left to the implementation and full version. ◀

▶ **Corollary 6** (evaluation of the six-moment kernel). *For fixed integers (n, m, a, b), the six moments in (4) can be computed in time $O(\log n)$ by an Euclidean recursion. The recursive system is closed: no moments outside the family (4) are generated.*

Proof. By Lemmas 4 and 5, the recursion alternates between two Euclidean-style operations: affine normalization replaces (a, b) by their residues modulo m, and the reciprocal step then swaps the active pair (m, a) to (a, m) with $a < m$. Thus after each nontrivial cycle the larger Euclidean parameter strictly decreases, exactly as in the ordinary Euclidean algorithm. The recursion therefore has depth $O(\log m) = O(\log n)$, and each step performs only $O(1)$ arithmetic operations on the constant-size family of moments. Hence the six-moment kernel is evaluable in $O(\log n)$ time. ◀

On the first segment, all required quantities reduce to polynomial sums, hence are computable in $O(1)$ time. On the second segment, Lemma 3 and Corollary 6 gives an $O(\log n)$ evaluation.

▶ **Corollary 7.** *For a fixed primitive direction (u, v) with $u \geqslant v \geqslant 1$, the contribution $c_{u,v}(n)$ can be computed in time $O(\log n)$.*

Since the number of primitive pairs with $u \leqslant B$ is $O(B^2)$, the total cost of the small part is $O(B^2 \log n) = O(n^{4/3} \log n)$.

6.2 Large directions

We now consider the complementary region $u > B = n^{2/3}$. Geometrically, this is completely analogous to the large-direction analysis in Section 5. The difference is only that the cutoff is now $B = n^{2/3}$ and that, instead of the square-root sweep, we evaluate the resulting one-dimensional sums by coprime prefix sums.

$$\sum_{u>B} O\left(\tau(u)\left(\frac{n}{u}\right)^2\right) = O\left(n^2 \sum_{u>B} \frac{\tau(u)}{u^2}\right) = O\left(\frac{n^2 \log n}{B}\right) = O(n^{4/3} \log n).$$

Thus the large-direction part matches the cost of the small-direction part, but it uses a different kernel: coprime prefix sums of orders $0, 1, 2$, not floor moments.

▶ **Theorem 8** (cubic-root decomposition). *The lattice-rectangle counting problem admits an algorithm of complexity $O(n^{4/3} \log n)$.*

Fix $B = \lfloor n^{2/3} \rfloor$. For $u \leqslant B$, each primitive direction is reduced to $O(1)$ queries to the six-moment kernel; for $u > B$, the dual parametrization is handled by the coprime prefix sums C_0, C_1, C_2. The main text proves the reduction and the resulting complexity bound, while Appendix A and Appendix B record the corresponding implementation details and explicit kernel transitions.

7 The final reduction to weighted floor sums: $O(n \log^3 n)$

We now turn to the fastest algorithm. This section explains how the quantities $R_{u,y,d}$ arise and how they reduce to weighted floor sums. We first isolate the required weighted floor-sum queries and derive the global complexity conditionally on an $O(\log n)$ query bound; this temporary assumption is discharged below by proving that the relevant kernel is recursively closed and evaluable in $O(\log n)$ time.

7.1 From primitive directions to Möbius inversion

We start from the contribution of the non-axis-parallel directions, restricting to $u > v \geqslant 1$. The coprimality condition is removed by the Möbius identity stated in Section 2. Writing $v = dt$ with $d \mid u$, $t \geqslant 1$, and $dt < u$, we obtain

$$F_1(n) = \sum_{u=2}^{n} \sum_{\substack{d \mid u \\ \mu(d) \neq 0}} \mu(d) \sum_{1 \leqslant t < u/d} \sum_{\substack{x \geqslant y \geqslant 1 \\ xu + y\,dt \leqslant n}} \mathcal{W}(x, y)\,(n - xu - y\,dt)(n - x\,dt - yu).$$

Now fix u, y, and d. The remaining inner contribution depends only on these three outer parameters, so we define

$$R_{u,y,d} := \sum_{1 \leqslant t < u/d} \sum_{\substack{x \geqslant y \geqslant 1 \\ xu + y\,dt \leqslant n}} \mathcal{W}(x, y)\,(n - xu - y\,dt)(n - x\,dt - yu). \tag{6}$$

Since necessarily $1 \leqslant y \leqslant \lfloor n/u \rfloor$, the whole sum becomes

$$F_1(n) = \sum_{u=2}^{n} \sum_{1 \leqslant y \leqslant \lfloor n/u \rfloor} \sum_{d \mid u} \mu(d)\, R_{u,y,d}. \tag{7}$$

7.2 How many outer triples?

Let $T(n)$ be the number of admissible outer triples (u, y, d), where $1 \leqslant y \leqslant \lfloor n/u \rfloor$ and $d \mid u$ is squarefree. For fixed u, the number of squarefree divisors of u equals $2^{\omega(u)}$, where $\omega(u)$ is the number of distinct prime divisors of u. Thus

$$T(n) = \sum_{u \leqslant n} 2^{\omega(u)} \left\lfloor \frac{n}{u} \right\rfloor.$$

For the algorithmic analysis we only need the upper bound $T(n) = O(n \log^2 n)$. Indeed, this upper bound follows from $\lfloor n/u \rfloor \leqslant n/u$, which gives $T(n) \leqslant n \sum_{u \leqslant n} 2^{\omega(u)}/u$. Next, using $2^{\omega(m)} = \sum_{d \mid m} \mu^2(d)$, where $\mu^2(d)$ is the indicator of squarefree integers, we obtain

$$\sum_{u \leqslant n} \frac{2^{\omega(u)}}{u} = \sum_{d \leqslant n} \frac{\mu^2(d)}{d} \sum_{m \leqslant n/d} \frac{1}{m}.$$

Now $\sum_{m \leqslant X} 1/m = O(\log X)$ and $\sum_{d \leqslant n} \mu^2(d)/d = O(\log n)$, so $\sum_{u \leqslant n} 2^{\omega(u)}/u = O(\log^2 n)$ and therefore $T(n) = O(n \log^2 n)$. Consequently, if each set $R_{u,y,d}$ can be processed in time $O(\log n)$ then the total running time is $O(n \log^3 n)$. The remainder of this section proves exactly this $O(\log n)$ evaluation bound by showing that all required weighted floor sums belong to a finite recursively closed Euclidean kernel.

7.3 Weighted floor-sum reduction

The purpose of this subsection is to isolate the one-dimensional arithmetic kernel behind the final algorithm. The explicit affine and reciprocal transition identities for this kernel are somewhat lengthy; here we state the kernel, explain why it is the right one, and derive the global $O(n \log^3 n)$ bound from the closure argument summarized in Appendix B.

Fix u, y, d. By definition,

$$R_{u,y,d} = \sum_{t<u/d} \sum_{\substack{x\geqslant y\geqslant 1 \\ xu+y\,dt\leqslant n}} \mathcal{W}(x,y)\,(n - xu - y\,dt)(n - x\,dt - yu).$$

For each fixed $x \geqslant y$, the variable t runs over

$$1 \leqslant t \leqslant T(x), \qquad T(x) := \min\left(\left\lfloor \frac{u-1}{d} \right\rfloor, \left\lfloor \frac{n-ux}{yd} \right\rfloor\right).$$

Write

$$T_0 := \left\lfloor \frac{u-1}{d} \right\rfloor, \qquad G(x) := \left\lfloor \frac{n-ux}{yd} \right\rfloor.$$

Then the summation over x naturally splits into two zones:

- the *capped zone*, where $G(x) \geqslant T_0$ and therefore $T(x) = T_0$;
- the *active-floor zone*, where $1 \leqslant G(x) < T_0$ and therefore $T(x) = G(x)$.

The split point is determined by the inequality $n - ux \geqslant ydT_0$, that is,

$$x \leqslant X_0 := \left\lfloor \frac{n - ydT_0}{u} \right\rfloor.$$

Hence the capped zone contributes only polynomial sums in x, while the active-floor zone is exactly where the weighted floor-sum kernel appears. More precisely,

$$R_{u,y,d} = \sum_{y\leqslant x\leqslant X_0} \mathcal{W}(x,y)\, S_x^{\mathrm{cap}} + \sum_{x>X_0} \mathcal{W}(x,y)\, S_x^{\mathrm{act}}, \quad \text{where}$$

$$S_x^{\mathrm{cap}} := \sum_{1\leqslant t\leqslant T_0} (n-xu-y\,dt)(n-xdt-yu), \qquad S_x^{\mathrm{act}} := \sum_{1\leqslant t\leqslant G(x)} (n-xu-y\,dt)(n-xdt-yu).$$

In the capped zone the inner sum is a polynomial in x because the upper limit T_0 is constant, and in the active-floor zone the same expansion reduces everything to moments of $G(x) = \lfloor (n-ux)/(yd) \rfloor$. Expanding the summand as a quadratic polynomial in t, we obtain

$$(n - xu - y\,dt)(n - xdt - yu) = P_0(x) + P_1(x)t + P_2(x)t^2, \quad \text{where}$$

$$P_0(x) = (n-xu)(n-yu), \qquad P_1(x) = -d\big(y(n-yu) + x(n-xu)\big), \qquad P_2(x) = xyd^2.$$

Using the identities

$$\sum_{t=1}^{T} 1 = T, \qquad \sum_{t=1}^{T} t = \frac{T(T+1)}{2}, \qquad \sum_{t=1}^{T} t^2 = \frac{T(T+1)(2T+1)}{6},$$

we see that each inner sum is a polynomial in x and in its upper limit T_0 or $G(x)$. Thus the direct expansion expresses $R_{u,y,d}$ through the following seven basic weighted floor sums:

$$\sum T(x), \quad \sum xT(x), \quad \sum x^2T(x),$$
$$\sum T(x)^2, \quad \sum xT(x)^2, \quad \sum x^2T(x)^2, \quad \sum xT(x)^3.$$

Now $\mathcal{W}(x,y) = 4 - 2\,\mathbf{1}_{x=y}$, so the diagonal contribution $x = y$ contributes only a constant-size correction.

▶ **Lemma 9** (direct weighted floor-sum reduction). *For fixed (u, y, d), the quantity $R_{u,y,d}$ is a linear combination of the seven basic sums displayed above over the active-floor zone $x > X_0$, together with ordinary polynomial sums in x coming from the capped zone $y \leqslant x \leqslant X_0$ and the diagonal correction $x = y$.*

The seven direct states are not closed under the Euclidean transitions. We therefore enlarge them to the ten-moment family from Appendix B. For integers $N \geqslant 0$, $m \geqslant 1$, and $a, b \geqslant 0$, define

$$\mathcal{H}_{p,q}(N; m, a, b) := \sum_{x=0}^{N-1} x^p \left\lfloor \frac{ax + b}{m} \right\rfloor^q, \qquad q \geqslant 1, \quad p \geqslant 0, \quad p + q \leqslant 4. \tag{8}$$

Equivalently, the state space is indexed by

$$(p, q) \in \{(0,1), (1,1), (2,1), (3,1), (0,2), (1,2), (2,2), (0,3), (1,3), (0,4)\}.$$

As in the six-state kernel discussed earlier, the affine and reciprocal Euclidean steps do not increase the total weighted degree $p + q$; since the seven direct queries all satisfy $p + q \leqslant 4$, their closure is therefore contained in the ten-state family above, while the five cases with $q = 0$ are ordinary monomial sums treated separately. Since the floor term has negative slope in x, we first reverse the index. Thus for any interval $[L, R]$,

$$\sum_{x=L}^{R} x^p \left\lfloor \frac{N - ux}{yd} \right\rfloor^q = \sum_{j=0}^{p} (-1)^j \binom{p}{j} R^{p-j} \mathcal{H}_{j,q}(R - L + 1; yd, u, N - uR).$$

▶ **Lemma 10** (affine closure of the weighted kernel). *For the evaluation of (8), each affine normalization step expresses a state $\mathcal{H}_{p,q}(N; m, a, b)$ as a linear combination of states $\mathcal{H}_{p',q'}(N; m, a', b')$ with $q' \geqslant 1$, $p' \geqslant 0$, $p' + q' \leqslant 4$, together with polynomial sums.*

▶ **Lemma 11** (reciprocal closure of the weighted kernel). *Under the reciprocal Euclidean step, each state of the extended family is expressible as a linear combination of states of the same family and polynomial sums.*

▶ **Corollary 12** (evaluation of the weighted kernel). *The ten-moment kernel (8) is evaluable in $O(\log n)$ time.*

The proof follows the same Euclidean-recursion scheme as in Lemmas 4 and 5 and Corollary 6. The affine step expands $(Ax + B + g(x))^q$, and the reciprocal step transposes the same staircase by horizontal levels; neither operation increases the weighted degree $p + q$. Thus the enlarged ten-state family is closed, and the Euclidean parameter decreases after each reciprocal step. The explicit coefficient formulas are mechanical extensions of the six-moment identities in Appendix B; they are omitted from this conference version and are included in the full version and implementation.

▶ **Corollary 13** (evaluation of outer contributions). *Each outer contribution $R_{u,y,d}$ is computable in time $O(\log n)$.*

Proof. By Lemma 9, each $R_{u,y,d}$ reduces to a constant number of shifted instances of the ten-moment kernel (8) plus polynomial sums. By Corollary 12, each such kernel query costs $O(\log n)$ time, while the polynomial-zone contribution is evaluated in $O(1)$ time. ◀

Combining the $O(\log n)$ evaluation of each outer contribution $R_{u,y,d}$ with the $O(n \log^2 n)$ bound on the number of admissible outer triples yields the final complexity bound.

Table 2 Sample exact values for powers of two.

k	$F(2^k)$	k	$F(2^k)$
1	1	19	172327747508964813792096
2	44	20	29077428688555984332023 44
3	1192	21	48931868439876126051425552
4	27128	22	82143761565179367519866 9752
5	564120	23	13759445380252558103053449112
6	11114080	24	23001422256138720967944581 6240
7	211224480	25	38380371046198672101121968 14232
8	3914221216	26	63933546372113490066412405897360
9	71182606216	27	10633359851249499413058636860 97296
10	1275797150128	28	17659763652737469299382592232330696
11	22602804487208	29	29289842469561056449421585791 2343064
12	396685572297544	30	48518500951587460955614854515923 36296
13	6907621416632376	31	80277206323003614389748671287223855080
14	119492377263166968	32	13267969779754764030926892868629865 16504
15	2055404973525169560	33	21906538476526319541299023010218991588136
16	35182910663019639384	34	36134920488712027208952304224982184057 1528
17	599669468453524178752	35	59551007063971108112609226598124911316624 32
18	10182597857710132553464	36	98057826153604756744005601368029402514221504

▶ **Theorem 14** (final complexity). *The lattice-rectangle counting problem admits an exact* $O(n \log^3 n)$ *algorithm.*

Proof. There are $O(n \log^2 n)$ admissible triples (u, y, d), and each corresponding contribution $R_{u,y,d}$ is computable in $O(\log n)$ time by Corollary 13. Summing over all outer triples proves the stated bound. ◀

Algorithmically, the final method iterates over the admissible outer triples (u, y, d), evaluates each contribution $R_{u,y,d}$ by the recursive weighted floor-sum kernel, and accumulates it with weight $\mu(d)$. The main text proves the reduction and the complexity bound; Appendix A gives pseudocode, while the full ten-moment transition identities are included in the full version and in the accompanying implementation.

8 Experiments

All four implementations were benchmarked in single-threaded C++ on one core of an Intel i7-13700 at 5.2 GHz, compiled with `clang++` 20.1.8 and `-O3 -march=native`, using `std::int128_t` throughout the tested range. The plots report wall-clock seconds for $n = 2^{13}, \ldots, 2^{30}$; each point is the median of repeated runs, with relative variation below 0.5%. The exact values in Table 2 were produced by the fastest implementation; small entries were cross-checked against the slower algorithms.

Comparison of algorithms

Figure 1 shows close agreement with the predicted asymptotic bounds. The log–log plots are nearly linear with the expected slopes, and the normalized runtimes remain stable or slowly decrease for the faster algorithms, indicating entry into the asymptotic regime.

In the tested range, the $O(n \log^3 n)$ implementation overtakes the $O(n^{4/3} \log n)$ one near the largest inputs and has the smallest normalized runtime among the nontrivial exact methods.

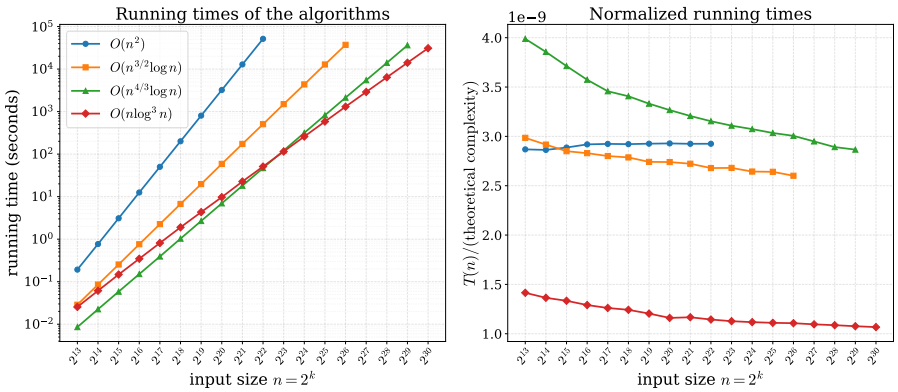

Figure 1 Wall-clock running times of the four algorithms on inputs $n = 2^{13}, \ldots, 2^{30}$ (left) and after normalization by the theoretical complexity (right).

9 Summary and future work

We obtained exact algorithms of complexity $O(n^2)$ for the classical primitive-direction sweep, $O(n^{3/2} \log n)$ for the square-root decomposition, $O(n^{4/3} \log n)$ for the cubic-root decomposition with floor moments, and $O(n \log^3 n)$ for the final weighted floor-sum reduction, where the weighted floor-sum family from Section 7 admits an $O(\log n)$ Euclidean evaluation. The key structural feature of the last two stages is the collapse of the geometric summation to one-dimensional arithmetic kernels: a six-state floor-moment kernel in the $O(n^{4/3} \log n)$ algorithm and a ten-state Euclidean kernel in the final $O(n \log^3 n)$ reduction. The full version contains implementation-level simplifications and the two-term asymptotic expansion.

Natural directions include further reducing the complexity, extending the methods to other lattice objects and higher dimensions, and evaluating $F(k)$ over $[1, n]$ in total complexity better than $O(n^2)$. A genuinely sublinear algorithm for a single value $F(n)$ would likely require a different idea: even in the final formulation there are $\Theta(n)$ natural outer scales to account for, so an $\Omega(n)$-type barrier appears heuristically unavoidable unless one finds additional global cancellations or a new transform-level description of the count.

References

1 George E. Andrews and Kimmo Eriksson. *Integer Partitions*. Cambridge University Press, 2004.

2 Matthias Beck and Sinai Robins. Dedekind sums: A combinatorial-geometric viewpoint. In *Unusual Applications of Number Theory*, volume 64 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 25–35. American Mathematical Society, Providence, RI, 2004. doi:10.1090/dimacs/064/04.

3 Matthias Beck and Sinai Robins. *Computing the Continuous Discretely: Integer-Point Enumeration in Polyhedra*. Undergraduate Texts in Mathematics. Springer, New York, second edition, 2015. doi:10.1007/978-1-4939-2969-6.

4 Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science, 2nd Ed.* Addison-Wesley, Reading, MA, second edition, 1994. URL: https://www-cs-faculty.stanford.edu/%7Eknuth/gkp.html.

5 Godfrey H. Hardy and Edward M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, Oxford, sixth edition, 2008.

6 Martin N. Huxley and Werner Georg Nowak. Primitive lattice points in convex planar domains. *Acta Arithmetica*, 76:271–283, 1996.

7 Hugh L. Montgomery and Robert C. Vaughan. *Multiplicative Number Theory I: Classical Theory*, volume 97 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2007. doi:10.1017/CBO9780511618314.

8 OEIS Foundation Inc. A085582: Number of rectangles (orthogonal or not) with corners on an $n \times n$ grid of points. https://oeis.org/A085582, 2026. Accessed: April 2026.

9 Jakub Pawlewicz and Mihai Pătraşcu. Order statistics in the Farey sequences in sublinear time and counting primitive lattice points in polygons. *Algorithmica*, 55:271–282, 2009. doi:10.1007/s00453-008-9221-z.

10 D. Radcliffe. Table and Python script for OEIS A085582. Linked from https://oeis.org/A085582, 2026. Accessed: April 2026.

11 Gérald Tenenbaum. *Introduction to Analytic and Probabilistic Number Theory*, volume 163 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, third edition, 2015. doi:10.1090/gsm/163.

12 Don Zagier. Higher dimensional Dedekind sums. *Mathematische Annalen*, 202:149–172, 1973. doi:10.1007/BF01425212.

A Algorithmic appendix: expanded pseudocode

This appendix records more explicit procedural versions of the four algorithmic stages. The aim is not to specify low-level implementation details, but to make the control flow of each algorithm and the interfaces between the geometric summation and the one-dimensional kernels transparent. Throughout this appendix, the three directions $(0,1)$, $(1,0)$, and $(1,1)$ are excluded from the loops and are handled separately through the closed-form term $F_0(n)$.

Algorithm 1 Classical quadratic baseline.

Initialize $ans \leftarrow 0$.
for each primitive (u,v) with $u > v > 0$ **do**
  determine $y_{\min} \leqslant y \leqslant y_{\max}$ from Equation (1)
  for $y = y_{\min}, \dots, y_{\max}$ **do**
    determine the induced interval $x_{\min}(y) \leqslant x \leqslant x_{\max}(y)$
    if nonempty, add the closed polynomial sum of (2) over x
  end for
end for
return $F_0(n) + ans$.

Algorithm 2 Square-root decomposition.

Set $B \leftarrow \lfloor \sqrt{n} \rfloor$ and $ans \leftarrow 0$.
for each primitive (u,v) with $u \leqslant B$ and $u > v > 0$ **do**
  evaluate $c_{u,v}(n)$, defined in Section 6.1, by the baseline routine
  add the resulting small-direction contribution to ans
end for
for $u = B+1, \dots, n$ **do**
  for each (x,y) with $1 \leqslant y \leqslant x \leqslant \lfloor n/u \rfloor$ **do**
    set $v_{\max} \leftarrow \min\{u-1, \lfloor (n-xu)/y \rfloor\}$
    accumulate the primitive contribution on $1 \leqslant v \leqslant v_{\max}$ by coprime prefix sums
  end for
end for
return $F_0(n) + ans$.

The squarefree divisor lists used in the final algorithm can be generated once by a sieve for the smallest prime factor together with the recursive construction of all squarefree divisors of each u. Their total size up to n is $\sum_{u \leqslant n} 2^{\omega(u)} = O(n \log n)$, so this preprocessing does not affect the final $O(n \log^3 n)$ complexity bound.

Algorithm 3 Cubic-root decomposition.

Set $B \leftarrow \lfloor n^{2/3} \rfloor$ and $ans \leftarrow 0$.
for each primitive (u, v) with $u \leqslant B$ and $u > v > 0$ **do**
  rewrite $c_{u,v}(n)$ as a fixed linear combination of six moments
  evaluate the required values $\mathbf{H}(\cdot;\cdot,\cdot,\cdot)$ by Appendix B
  add the resulting small-direction contribution to ans
end for
for $u = B+1, \ldots, n$ **do**
  set $x_{\max} \leftarrow \max\{x \in \mathbb{Z}_{\geqslant 0} : xu < n\}$
  for each (x, y) with $1 \leqslant y \leqslant x \leqslant x_{\max}$ **do**
    set $m \leftarrow 2$ if $x = y$, and $m \leftarrow 4$ otherwise
    set $v_{\max} \leftarrow \min\{u-1, \max\{v \in \mathbb{Z}_{\geqslant 0} : yv < n - xu\}\}$
    evaluate (C_0, C_1, C_2) on $1 \leqslant v \leqslant v_{\max}$ by coprime prefix sums
    add $m\big((n-xu)(n-yu)C_0 - (x(n-xu) + y(n-yu))C_1 + xyC_2\big)$
  end for
end for
return $F_0(n) + ans$.

Algorithm 4 Final weighted-floor-sum algorithm.

Precompute the squarefree divisor lists $D(u) = \{(d, \mu(d)) : d \mid u,\ d \text{ squarefree}\}$ for all $u \leqslant n$.
Initialize $ans \leftarrow 0$.
for $u = 2, \ldots, n$ and each y with $1 \leqslant y \leqslant \lfloor n/u \rfloor$ **do**
  for each $(d, \mu(d)) \in D(u)$ with $d < u$ **do**
    form the Möbius-expanded quantity $R_{u,y,d}$ from (6)
    split $R_{u,y,d}$ into its polynomial and weighted floor-sum parts
    evaluate the weighted queries by the ten-state version of Appendix B
    add $\mu(d) R_{u,y,d}$ to ans
  end for
end for
return $F_0(n) + ans$.

B Euclidean floor-moment kernels

We record the common recursive structure behind the six-moment kernel used in Section 6 and the ten-moment kernel used in Section 7. For integers $n \geqslant 0$, $m \geqslant 1$, and $a, b \geqslant 0$, put

$$\mathcal{H}_{p,q}(n; m, a, b) = \sum_{x=0}^{n-1} x^p \left\lfloor \frac{ax+b}{m} \right\rfloor^q .$$

In the cubic-root algorithm we need the six states with $q \geqslant 1$ and $p + q \leqslant 3$, namely $(0,1), (1,1), (2,1), (0,2), (1,2), (0,3)$. In the final algorithm we use the ten states with $q \geqslant 1$ and $p + q \leqslant 4$, listed in (8).

The base case $a = 0$ is immediate because the floor term is constant. For the affine step, write $a = Am + a'$ and $b = Bm + b'$, where $0 \leqslant a', b' < m$. Then

$$\left\lfloor \frac{ax+b}{m} \right\rfloor = Ax + B + \left\lfloor \frac{a'x + b'}{m} \right\rfloor ,$$

and expanding $(Ax + B + g(x))^q$ expresses every $\mathcal{H}_{p,q}(n; m, a, b)$ as a linear combination of moments with parameters $(n; m, a', b')$ and ordinary power sums $\sum_{x<n} x^r$. The total weighted degree $p + q$ never increases.

After this reduction assume $0 \leqslant a, b < m$ and $a > 0$. Let

$$Y = \left\lfloor \frac{a(n-1)+b}{m} \right\rfloor, \qquad g(t) = \left\lfloor \frac{mt + (m-b-1)}{a} \right\rfloor, \quad 0 \leqslant t < Y.$$

Transposing the staircase $y \leqslant \lfloor (ax+b)/m \rfloor$ and summing by horizontal levels rewrites each moment for $(n; m, a, b)$ as a linear combination of moments for $(Y; a, m, m-b-1)$ and power sums in n and Y. Again the total weighted degree is preserved. Thus each nontrivial reciprocal step decreases the Euclidean parameter from m to $a < m$, while all computations remain inside the same constant-size state space.

Consequently both the six-state and ten-state kernels are evaluable in $O(\log m)$ arithmetic operations. For the six-state kernel, the affine formulas are short enough to include here. Let

$$\mathbf{B} = \mathbf{H}(n; m, a', b') = (b_{01}, b_{11}, b_{21}, b_{02}, b_{12}, b_{03}), \qquad P_r = \sum_{x=0}^{n-1} x^r.$$

Then

$$\begin{aligned}
\mathcal{H}_{0,1} &= b_{01} + AP_1 + BP_0,\\
\mathcal{H}_{1,1} &= b_{11} + AP_2 + BP_1,\\
\mathcal{H}_{2,1} &= b_{21} + AP_3 + BP_2,\\
\mathcal{H}_{0,2} &= b_{02} + 2Ab_{11} + 2Bb_{01} + A^2P_2 + 2ABP_1 + B^2P_0,\\
\mathcal{H}_{1,2} &= b_{12} + 2Ab_{21} + 2Bb_{11} + A^2P_3 + 2ABP_2 + B^2P_1,\\
\mathcal{H}_{0,3} &= b_{03} + 3Ab_{12} + 3Bb_{02} + 3A^2b_{21} + 6ABb_{11} + 3B^2b_{01}\\
&\quad + A^3P_3 + 3A^2BP_2 + 3AB^2P_1 + B^3P_0.
\end{aligned}$$

For the reciprocal step, let

$$Y := \left\lfloor \frac{a(n-1)+b}{m} \right\rfloor.$$

If $Y = 0$, then all six moments vanish. Otherwise put

$$\mathbf{G} = \mathbf{H}(Y; a, m, m-b-1) = (g_{01}, g_{11}, g_{21}, g_{02}, g_{12}, g_{03}), \qquad Q_r := \sum_{t=0}^{Y-1} t^r.$$

Staircase transposition gives

$$\begin{aligned}
\mathcal{H}_{0,1}(n; m, a, b) &= P_0Y - Q_0 - g_{01},\\
\mathcal{H}_{1,1}(n; m, a, b) &= \frac{2P_1Y - g_{01} - g_{02}}{2},\\
\mathcal{H}_{2,1}(n; m, a, b) &= \frac{6P_2Y - g_{01} - 3g_{02} - 2g_{03}}{6},\\
\mathcal{H}_{0,2}(n; m, a, b) &= P_0Y^2 - Q_0 - g_{01} - 2Q_1 - 2g_{11},\\
\mathcal{H}_{1,2}(n; m, a, b) &= \frac{2P_1Y^2 - g_{01} - g_{02} - 2g_{11} - 2g_{12}}{2},\\
\mathcal{H}_{0,3}(n; m, a, b) &= P_0Y^3 - Q_0 - g_{01} - 3Q_1 - 3g_{11} - 3Q_2 - 3g_{21}.
\end{aligned}$$

These identities are obtained by summing horizontal strips between the two complementary staircases. They also make the algorithmic closure explicit: all right-hand-side non-polynomial terms are moments for the reciprocal instance $(Y; a, m, m-b-1)$, while all other terms are ordinary power sums.

The affine step reduces (a, b) modulo m, and the reciprocal step replaces $(n; m, a, b)$ by $(Y; a, m, m - b - 1)$, so the active modulus strictly decreases from m to $a < m$. Hence the recursion has Euclidean depth $O(\log m)$, and each step performs only $O(1)$ arithmetic operations on the stored moments and the power sums.

The ten-state expansion is the same calculation with the larger family $q \geqslant 1$, $p + q \leqslant 4$. We omit only the expanded coefficient table in this conference version; it is included in the full version and in the accompanying implementation.

C Implementation notes and verification

The implementation follows the reductions above literally, but uses two constant-factor simplifications in the final weighted-floor kernel. First, the affine and reciprocal transition formulas are hard-coded as straight-line arithmetic rather than constructed symbolically at run time. Second, the two most common affine subcases, namely $B = 0$ and then $A = 1$, are handled separately; these cases occur frequently in the Euclidean recursion and remove a large number of temporary products.

The four algorithms are kept in the repository as separate implementations. This is useful both for benchmarking and for validation: the quadratic implementation checks the small range, the square-root and cubic-root implementations check intermediate sizes, and the final implementation is then compared against all earlier methods on their overlapping ranges. The scripts also reproduce the timing plot in Figure 1.

For reproducibility, the code separates the mathematical kernels from the drivers enumerating the outer parameters. The same weighted-floor routine is used by the cubic-root and final implementations with different state families, enabling tests of the Euclidean recursion independently of the surrounding geometry. All benchmarked runs use exact integer arithmetic.