

Deterministic enumeration of primes in $N(\log \log N)^{1+o(1)}$ bit operations

24 October 2026

Abstract

We give a deterministic algorithm that lists the primes below N in increasing order in $N(\log \log N)^{1+o(1)}$ bit operations on a multitape Turing machine. The algorithm uses Harvey’s reconstruction algorithm for odd square-primes, together with his direct tests and conversion to primes. Its candidate for a block is correct when the block contains at most a prescribed number of odd square-primes, but the output does not identify blocks where this condition fails. We certify blocks through a nonnegative sieve majorant. Its block sums are rarely large, and we compute upper bounds for those sums whose total excess is small. A mean-square estimate controls the number of blocks with large exact sums. Logarithmic binning and convolution compute one-sided bounds for all the sums in $O(N)$ bit operations. The error comes from products near block boundaries, and a modular count bounds its total. Blocks of length about $(\log N)^{18}$ then leave only $O(N)$ bit operations of direct testing. The result answers Harvey’s question about unconditional deterministic enumeration in $N(\log N)^{o(1)}$ bit operations.

1 Introduction

The problem is to list every prime below an input bound N , in increasing order. We measure the cost in bit operations on a multitape Turing machine. A machine in this model has a fixed number of tapes, and each tape head moves one position at a time. Thus an algorithm must pay for moving between distant positions as well as for arithmetic. A sieve that is fast when memory can be accessed at arbitrary locations need not have the same cost in this model.

Harvey’s unconditional deterministic algorithm takes $N(\log N)^{1/2}(\log \log N)^{1+o(1)}$ bit operations [5, Theorem 1.2]. He also gives a deterministic algorithm with the smaller bound $N(\log \log N)^{1+o(1)}$, whose success on every input follows from a conjecture about square-primes in short intervals [5, Theorem 1.4 and Conjecture 7.1]. We obtain this smaller bound without the conjecture.

Theorem 1.1. *There is a deterministic algorithm on a multitape Turing machine that, given an integer $N \geq 3$, enumerates all primes below N in increasing order in*

$$N(\log \log N)^{1+o(1)}$$

bit operations.

The algorithm uses Harvey’s existing methods to reconstruct a sparse set of integers, test the integers in selected blocks, and convert the result to primes. Our task is to select those blocks both safely and cheaply. The distinction between these two requirements is central: a proof that few blocks are difficult does not by itself tell an algorithm which blocks they are.

The proof uses estimates for divisor sums, periodic counting, and integer convolution. We develop the counting arguments below, including the correlation calculation in Section 4.1 and the implementation of convolution in Section 5.6. The specialized number-theoretic inputs are the three algorithmic theorems stated in Section 2 and the mean estimate for the sieve weights in Lemma 3.1. We also use the standard integer-multiplication bound stated in Section 5. We use these results through their stated hypotheses and conclusions; their internal proofs are not needed to follow the argument here. Appendix A specifies the weights, and Appendix B supplies the elementary numerical routines. The additional decoding result in Appendix C is separate from the proof of Theorem 1.1.

1.1 What the reconstruction algorithm guarantees

An integer is a *square-prime* if it is m^2p , where $m \geq 1$ is an integer and p is prime. For example, $45 = 3^2 \cdot 5$ is a square-prime, and a prime itself is a square-prime with $m = 1$. We will first recover the odd square-primes. Their increasing list can be converted to the increasing prime list in $O(N)$ bit operations [5, Proposition 7.13].

This intermediate set comes from Harvey’s generating-function identity [5, Theorem 2.6]. The identity initially records, modulo 2, the number of representations $n = m^2p^\ell$, with m odd, p an odd prime, and $\ell \geq 1$. Harvey corrects the terms with $\ell \geq 2$ to obtain the indicator of the odd square-primes [5, Section 6.1]. The representation m^2p is unique when it exists: in the prime factorization, p is the unique prime with an odd exponent. Removing one factor p leaves a square, which determines m . For instance, $27 = 3^2 \cdot 3$ is also covered by this description.

Divide $[0, N)$ into consecutive blocks

$$I_r = [rT, (r+1)T) \cap \mathbb{Z}, \quad 0 \leq r < N/T,$$

where T divides N . A block contains exactly T integers. Let w_r be its number of odd square-primes, and let $a^r \in \{0, 1\}^T$ be their indicator vector. Thus a^r has w_r entries equal to one.

Harvey’s core algorithm takes a threshold R and returns a candidate vector c^r for every block. Its guarantee is

$$w_r \leq R \implies c^r = a^r, \quad \text{and always} \quad \text{wt}(c^r) \leq R.$$

Here wt means the number of ones. If $w_r > R$, the candidate has too few ones to be correct. The algorithm need not report that failure. In particular, looking at the candidate’s weight does not tell us whether it can be trusted.

For the overview, write $L \asymp \log N$; the exact bit-length convention and the harmless enlargement of the input bound are specified in Section 2. When T is bounded by a fixed power of L , the core costs

$$O\left(\left(1 + \frac{RL}{T}\right) N(\log L)^{1+o(1)}\right).$$

Consequently we want R to be of order T/L . This keeps the factor $1 + RL/T$ bounded and gives the running time in Theorem 1.1. We are free to choose a longer polylogarithmic block length without changing that conclusion, provided we increase R in the same proportion.

Our blocks are called *intervals* in Harvey’s paper; his word *block* refers to an object inside the core algorithm. The formal statement that we use is Theorem 2.1 below.

1.2 Certifying most blocks and repairing the rest

We keep a candidate only after establishing $w_r \leq R$. This is what it means to *certify* a block. We mark every other block and test its positive integers directly. A marked block need not have more than R odd square-primes: our test for acceptance is allowed to be conservative.

Harvey's bulk test, stated as Theorem 2.2, finds the square-primes in any list $\mathcal{R}$ of distinct positive integers below N . It costs

$$O(N) + |\mathcal{R}|L^{11.5+o(1)}.$$

We will arrange that the marked blocks contain $O(N/L^{12})$ integers. The cost of testing them is then

$$O(N) + O\left(\frac{N}{L^{12}}L^{11.5+o(1)}\right) = O(N) + O\left(NL^{-1/2+o(1)}\right) = O(N).$$

This explains the exponent 12: it leaves room above the exponent 11.5 in the direct test.

We therefore need two things from the marking procedure. Every unmarked block must satisfy $w_r \leq R$, and the marked blocks must have small total length. Neither requirement asks us to compute w_r exactly. An exact count would give an acceptance test for the core's candidate; the upper bounds below supply such a test without determining the count.

1.3 Why upper bounds with errors are enough

We use three quantities for each block:

$$\begin{aligned} w_r &: \text{ the true number of odd square-primes,} \\ W_r &: \text{ an exact sum of nonnegative weights,} \\ U_r &: \text{ a computed upper bound for that sum.} \end{aligned}$$

The weight construction will give $w_r \leq W_r$, except on a small set of blocks that we mark separately. The computation will give $W_r \leq U_r$ on every block. We mark a block whenever $U_r > R$. Thus, on every unmarked block,

$$w_r \leq W_r \leq U_r \leq R,$$

which is exactly the condition needed to keep the candidate.

An upper bound can be safe but too large to be useful. To prevent it from marking too many blocks, we control the sum of the errors $U_r - W_r$. Each error is nonnegative, so no large positive error can be hidden by a negative one elsewhere.

Here is the elementary counting argument that joins the two parts of the proof. Choose a sufficiently large fixed constant C_1 and put

$$R = \left\lceil \frac{2C_1T}{L} \right\rceil.$$

If $U_r > R$, then at least one of the following must hold:

$$W_r > \frac{C_1T}{L}, \quad U_r - W_r > \frac{C_1T}{L}.$$

Otherwise their sum would be at most $2C_1T/L \leq R$. Every block of the second kind accounts for more than C_1T/L in the nonnegative total error. Hence

$$T \#\{r : U_r > R\} \leq T \#\{r : W_r > C_1T/L\} + \frac{L}{C_1} \sum_r (U_r - W_r). \quad (1.1)$$

Multiplication by T turns a number of blocks into their total length.

We will prove

$$T \#\{r : W_r > C_1 T/L\} = O(N/L^{12}), \quad \sum_r (U_r - W_r) = O(N/L^{13}).$$

The first estimate concerns the exact weights. The second concerns the computation. Substitution in (1.1) gives $T \#\{r : U_r > R\} = O(N/L^{12})$, since multiplying the second estimate by L/C_1 gives the same order. The separately marked exceptions to $w_r \leq W_r$ also have total length $o(N/L^{12})$, as shown in the next subsection. Thus all marked blocks fit within the repair budget.

Once the estimate for large exact weights holds, a computed bound may be poor on an individual block. It is enough that every computed bound is an upper bound and their total excess is small.

1.4 Constructing the weights

We first need a nonnegative function that detects primes without assigning too much weight to other integers. An upper-bound sieve supplies coefficients $\lambda_e \in \{-1, 0, 1\}$ such that

$$\theta(q) = \sum_{e|q} \lambda_e \geq 0 \quad (q \geq 1).$$

Only squarefree $e \leq Z$ occur, and their prime factors are below y . Here squarefree means that no square of a prime divides e . The parameters Z and y are small fixed powers of N . Also $\lambda_1 = 1$. If $q = p > y$ is prime, only $e = 1$ contributes, so $\theta(p) = 1$.

The average weight is small. More precisely, with $\nu = \sum_e \lambda_e/e$, we have, for every real $x \geq 0$,

$$\sum_{1 \leq q \leq x} \theta(q) = \nu x + O(Z), \quad 0 \leq \nu = O(1/L).$$

The coefficient ν describes the average over long intervals; the error $O(Z)$ must still be allowed for on a finite interval. This weight does not distinguish primes from all composites. If p and q are primes exceeding y , then $\theta(pq) = 1$ as well: the only supported divisor of pq is 1. The sieve supplies an upper bound whose average is small, rather than an exact primality test. Although its coefficients can have either sign, the resulting value $\theta(q)$ is always nonnegative.

To detect square-primes, remove each possible odd square divisor and apply θ to the quotient:

$$W(n) = \sum_{\substack{m \geq 1, m^2 | n \\ m \text{ odd}}} \theta(n/m^2) \quad (n \geq 1), \quad W_r = \sum_{\substack{n \in I_r \\ n \geq 1}} W(n).$$

If $n = m^2 p$ is an odd square-prime with $p > y$, then m is odd, and the term for this m is one. Since the other terms are nonnegative, $W(n) \geq 1$. It follows that W_r bounds the number of such square-primes in the block.

The exception is $p \leq y$: the sieve may give $\theta(p) = 0$. We enumerate these square-primes separately and mark their blocks. There are at most

$$\sqrt{N} \sum_{p \leq y} p^{-1/2} \leq \sqrt{N} \sum_{j \leq y} j^{-1/2} = O(\sqrt{Ny})$$

pairs (m, p) to consider. Multiplying by T bounds the total length of their blocks, even if we count the same block several times. Our later choice has $y \leq N^{1/1000}$. Hence

$$\frac{T\sqrt{Ny}}{N/L^{12}} \leq TL^{12}N^{-999/2000} \rightarrow 0,$$

because T is bounded by a fixed power of $L \asymp \log N$. Thus these blocks have total length $o(N/L^{12})$. This is the exceptional set excluded when writing $w_r \leq W_r$ above.

The divisor-sum representation serves two purposes. The next subsection uses it to estimate how often the block sums are large. For the computation, expanding θ writes a contribution to W_r as a product $n = m^2 ek \in I_r$, with sign λ_e . Section 1.6 groups the factors by logarithmic size and uses convolution to count these groups. A group can cross a block boundary, so its counts give upper bounds U_r for the exact sums W_r , with a controlled total excess.

1.5 Why large exact sums are rare

There are two reasons why knowing the mean is not enough here. First, subtracting the summatory estimate for θ at the ends of an interval of length T gives $\nu T + O(Z)$. Our choice $Z = \lfloor N^{1/1000} \rfloor$ eventually makes the allowed error scale Z much larger than T/L , since T is only a fixed power of $\log N$. Thus that estimate does not bound each short interval well enough. Second, even a total-weight bound of order N/L would allow order N/T blocks to exceed a threshold of order T/L . Their total length would be of order N , whereas we need $O(N/L^{12})$.

We therefore estimate how much block sums can deviate from their mean. The squared deviations are nonnegative, and each large deviation uses a definite part of their total. The average in this argument is over positions of an interval; the algorithm itself uses no random choices.

We separate the sum defining W according to whether $m \leq D$ or $m > D$, where $D = L^{12}$. The first part is the *truncated majorant*:

$$W_{\leq D}(n) = \sum_{\substack{1 \leq m \leq D \\ m \text{ odd}, m^2 | n}} \theta(n/m^2), \quad W_r^{\leq D} = \sum_{\substack{n \in I_r \\ n \geq 1}} W_{\leq D}(n).$$

The remaining part, $W(n) - W_{\leq D}(n)$, is its *tail*.

The cutoff is needed because the mean-square estimate depends on the largest divisor index. On expanding $\theta(n/m^2)$, an index has the form $m^2 e$, with $e \leq Z$. Restricting m to $m \leq D$ bounds this index by $Q = D^2 Z$. For windows of length proportional to T and a range of endpoints of length N , Lemma 4.1 bounds the total squared deviation by

$$O(NT(1 + \log Q)^3 + Q^4).$$

Here $\log Q = O(L)$. The choices $D = L^{12}$ and $Z \leq N^{1/1000}$ give

$$\frac{Q^4}{N} \leq N^{-249/250} L^{96} = o(TL^3).$$

Thus dividing by N gives an average squared deviation $O(TL^3)$. Without the cutoff, divisor indices can be as large as order N , for which the Q^4 term gives no useful bound.

The mean parameter for an interval of length T is $T\nu_D$, where

$$\nu_D = \nu \sum_{\substack{1 \leq m \leq D \\ m \text{ odd}}} \frac{1}{m^2} \leq 2\nu = O(1/L).$$

We choose C_1 large enough that even $3T\nu_D \leq C_1 T/(4L)$; this leaves a gap of order T/L below the threshold $C_1 T/(2L)$ for the longer windows used below. A mean-square bound then limits

the number of blocks that cross it: each such block must account for a definite amount of squared deviation. Section 4.2 proves

$$T \# \{r : W_r^{\leq D} > C_1 T / (2L)\} \ll \frac{NL^5}{T}.$$

The Q^4 error has been absorbed here using the fixed parameters. After averaging over interval positions, the squared-deviation estimate has size $O(TL^3)$. A deviation of size T/L has square $(T/L)^2$. Thus the proportion of such positions is bounded by a constant multiple of

$$\frac{TL^3}{(T/L)^2} = \frac{L^5}{T}.$$

Longer blocks therefore reduce this exceptional proportion.

To pass from moving intervals to fixed blocks, the proof uses windows of length $3T$. Every I_r is contained in $(x - 3T, x]$ for all $x \in [(r + 1)T, (r + 2)T]$. The proof checks that the divisor-sum extension remains nonnegative at the ends of the range. Thus an exceptional block forces a deviation greater than $C_1 T / (4L)$ throughout this endpoint interval. These endpoint intervals have disjoint interiors and length T , so the measure bound for moving positions gives the displayed total-length bound for fixed blocks.

The tail needs only a bound for its total weight. Writing $n = m^2 q$, each $m > D$ contributes at most $O(N/(m^2 L) + Z)$ by the summatory estimate for θ . There are at most $\sqrt{N}$ possible m , and $\sum_{m>D} m^{-2} \leq 1/D$. Section 4.3 therefore proves

$$\sum_{1 \leq n < N} (W(n) - W_{\leq D}(n)) \ll \frac{N}{DL} + Z\sqrt{N} = O(N/(DL)).$$

The last step uses $Z\sqrt{N} \leq N^{501/1000} = o(N/L^{13})$ and $D = L^{12}$. A block whose tail exceeds $C_1 T / (2L)$ uses at least this much of the total. There are consequently $O(N/(DT))$ such blocks, and their total length is $O(N/D)$.

If $W_r > C_1 T / L$, either its truncated part or its tail exceeds $C_1 T / (2L)$. Combining the two estimates gives

$$T \# \{r : W_r > C_1 T / L\} \ll \frac{NL^5}{T} + \frac{N}{D}. \quad (1.2)$$

For $D = L^{12}$ and $T \geq L^{18}$, this is $O(N/L^{12})$. This establishes the first requirement in (1.1). It remains to compute suitable upper bounds without spending too much time.

1.6 Counting products without listing them

Recall that $\lambda_e \in \{-1, 0, 1\}$ and that $\lambda_e = 0$ unless $e \leq Z$ is squarefree. Expand the divisor sums and write $n = m^2 ek$. The result is

$$W_r = \sum_{(d, \sigma) \in \mathcal{S}} \sigma \# \{k \geq 1 : dk \in I_r\},$$

where

$$\mathcal{S} = \{(m^2 e, \lambda_e) : m \geq 1 \text{ odd}, \lambda_e \neq 0, m^2 e \leq N\}.$$

This list has $O(\sqrt{NZ})$ entries; Section 3 proves the bound by summing at most $\sqrt{N/e}$ choices of m for each e . The entries give the divisors and their signs; they do not list all the products. That distinction matters for bit complexity: the entry $d = 1$ alone would produce about N

products, and writing them as explicit binary integers would cost $\Omega(NL)$ bits: at least a fixed positive proportion are at least $N/2$, so each of those entries needs $\Omega(L)$ bits.

For a given divisor d , only multipliers $k < N/d$ can contribute below N . That varying cutoff prevents us from using the same multiplier histogram for every d . We first allow all pairs with $1 \leq k \leq N$, so the two factors can be counted independently. After counting these pairs, we use their product locations to keep only contributions that can lie below N . This deliberate enlargement is why the endpoint N , as well as the interior block boundaries, must be included in the error analysis.

We group each factor by the size of its logarithm. Let $\eta = L^4$ and $K = \lceil N/\eta \rceil$. A logarithmic bin has width about $1/K$ on the $\log_2$ scale. Near a value x , this corresponds to an ordinary width of order x/K : the ratio of its endpoints is $2^{1/K}$, and $2^{1/K} - 1$ is of order $1/K$. Near N , the uncertainty is therefore of order $N/K \leq \eta$. There are $O(KL)$ bin thresholds below N , and each numerical evaluation will cost $O(L^3)$ bit operations. Their total cost is $O(NL^4/\eta)$. This is the reason for choosing $\eta = L^4$: it makes the numerical work $O(N)$ while keeping the boundary uncertainty small enough for the marking argument.

We count how many multipliers k lie in each bin, and how many positive and negative divisor entries lie in each bin. These histograms store counts, rather than lists of their objects. For the multipliers, the main count in a bin is the difference between its consecutive integer thresholds; only an initial $O(K)$ multipliers are assigned individually. The shorter divisor list can be processed one entry at a time. Section 5.1 proves that the rounded thresholds give an exact partition with the required location bounds.

The identity

$$\log_2(dk) = \log_2 d + \log_2 k$$

makes multiplication compatible with addition of bin indices. If A_i and B_j are two factor counts, their convolution has entry $\sum_{i+j=\ell} A_i B_j$. It counts the pairs whose bin indices add to ℓ . We perform this operation separately for the positive and negative divisor entries. To compute all the coefficients together, we pack each histogram into an integer, using $O(L)$ binary positions for each count. Enough space between counts prevents carries between the convolution coefficients, so integer multiplication produces the whole convolution. Section 5.6 proves this construction and accounts for sorting and sequential tape scans.

The numerical computation is allowed to move a factor to a nearby bin. Its chosen bin must satisfy a proved bound on the logarithmic index, and the factor must be counted exactly once. The resulting pair counts are therefore exact for the partition actually used; we do not need to recover an ideal partition with exact real thresholds. Uncertainty concerns where the products lie within the interval associated with their sum of indices.

The certified construction in Section 5 gives enclosures of width at most 64η whenever they meet $[0, N)$. Enclosures outside this range do not contribute to the block sums. Since this is less than T , an enclosure lies inside one block or crosses one block boundary. In the first case, its signed count can be assigned to that block. In the second case, we assign its positive count to every block in $[0, N)$ that it meets and subtract its negative count from none of them. At the endpoint N , only one retained block is involved.

For example, consider an enclosure crossing an interior block boundary and containing seven positive pairs and three negative pairs. Assigning seven to each block is safe, whatever the unknown split of the products. Subtracting three from either block would not be safe: all three negative products might belong to the other one. The total true contribution from this bin is $7 - 3 = 4$, while the two bounds receive 14. Thus this bin contributes an excess of 10. We do not try to remove this excess by locating the products more precisely. We bound how

often such ambiguity can occur.

Every ambiguous product is near a multiple of T . The choice of parameters ensures that $(d, T) \leq 2$ for every divisor entry, where (d, T) denotes the greatest common divisor. Indeed, T is a power of two, m is odd, and e is squarefree. For fixed d , every $T/(d, T)$ consecutive multipliers k make the products dk visit each allowed residue modulo T exactly once. There are at least $T/2$ such residues. Only a small fraction of these residues are near a boundary. The condition is essential: if $T \mid d$, every product dk lies on a boundary.

Only enclosures meeting $[0, N)$ can cause excess. Their width bound forces every contributing product to be less than $N + 64\eta < 2N$, even if it lies beyond the endpoint N . Each pair adds at most two units of excess: a positive pair is assigned to at most two blocks, and an uncertain negative pair is omitted from at most its one actual block. Thus, for a fixed d , it suffices to count $k \leq 2N/d$ whose products are within 64η of a multiple of T .

Put $g = (d, T) \leq 2$. In one period of length T/g , the allowed residues are the multiples of g , and only $O(\eta/g + 1) = O(\eta)$ are near a boundary. There are $O(Ng/(dT))$ complete periods. A final incomplete period has no repeated residue, so it contributes at most the same $O(\eta)$ favorable residues. This gives $O(N\eta/(dT) + \eta)$ pairs for this divisor. Summing their bounded charges gives

$$\sum_r (U_r - W_r) \ll \frac{N\eta}{T} \sum_{(d,\sigma) \in \mathcal{S}} \frac{1}{d} + \eta|\mathcal{S}|.$$

Lemma 3.2 gives $\sum_{(d,\sigma) \in \mathcal{S}} 1/d = O(L)$ and $|\mathcal{S}| = O(\sqrt{NZ})$. Consequently $T \geq \eta L^{14}$ makes the first term $O(N/L^{13})$. For the second, recall that $Z = \lfloor N^{1/1000} \rfloor$ and $\eta = L^4$; hence

$$\eta|\mathcal{S}| = O\left(N^{1/2+1/2000}L^4\right) = o(N/L^{13}).$$

It follows that

$$\sum_r (U_r - W_r) = O(N/L^{13}),$$

which is the total-excess requirement in (1.1).

Each packed integer has $O(KL)$ count slots of $O(L)$ bits, hence length $b = O(KL^2) = O(N/L^2)$ bits. The Schönhage–Strassen bound $O(b \log b \log \log b)$, stated in Section 5, therefore gives $O((N/L^2)L \log L) = O(N \log L/L)$ bit operations for the two multiplications. The histogram construction and the two integer multiplications take $O(N)$ bit operations for these parameters, as checked in Section 6. The exact threshold construction, endpoint cases, and tape costs are proved in Section 5.

1.7 The algorithm and its relation to earlier work

The algorithm can now be stated in order. Its parameters are fixed precisely in Section 2.

1. Choose the parameters and enlarge the input bound to a multiple of T .
2. Compute the sieve coefficients λ_e and the signed divisor list $\mathcal{S}$.
3. Compute the upper bounds $U_r \geq W_r$ by convolution.
4. Mark the blocks with $U_r > R$ and the blocks containing an odd square-prime m^2p with $p \leq y$.
5. Run Harvey’s core and keep its candidates on the unmarked blocks.

6. Test the positive integers in the marked blocks and keep their odd square-primes.
7. Merge the results into the increasing list of odd square-primes, convert this list to primes, and discard primes at least the original input bound.

Section 6 verifies the hypotheses of the imported algorithms and accounts for every step. The core takes $N(\log \log N)^{1+o(1)}$ bit operations; the remaining steps together take $O(N)$. The exponent 18 in the block length is a sufficient choice arising from the error and cost estimates. It is not an optimized exponent.

For context, before Harvey's work the best bound in this model was $O(N \log N)$, due to Sergeev [8], following Schönhage, Grotfeld and Vetter [9] and Farach-Colton and Tsai [2]. Harvey surveys these results in [5, Sections 1.3 and 1.6]. Alongside his deterministic algorithms, he gives a Las Vegas algorithm with expected running time $N(\log \log N)^{2+o(1)}$ [5, Theorem 1.3]; a Las Vegas algorithm uses randomness but returns only a correct answer.

Harvey already identifies $R \asymp T/L$ as the desired scale for the core [5, Section 1.7]. The difficulty is to recognize enough blocks at that scale. His unconditional deterministic method uses rough integers. An integer is Y -rough if it has no prime factor at most Y , and a square- Y -rough integer is $m^2 n$ with n Y -rough. These include the square-primes whose prime part exceeds Y .

For integers $2 \leq Y \leq \sqrt{N}$, his generalization of Sergeev's algorithm constructs the *indicator bit array* of the Y -rough integers below N in $O(N \log Y)$ bit operations [5, Theorem 10.1]. From that array he obtains the increasing list of odd square- Y -rough integers in $O(N \log N / \log Y)$ bit operations [5, Proposition 10.7], and counts them by interval. The array and the explicit list have different output costs. Increasing Y reduces the excess of rough integers over primes, but makes the array more expensive to construct.

Harvey's sieve estimate for tuples controls the rejected intervals [5, Sections 8 and 10.2]. His parameter choice has $T \asymp \log Y \asymp L^{1/2} \log L$ and $R \asymp \log L$. For these values the factor in the core's cost is

$$1 + \frac{RL}{T} \asymp L^{1/2}.$$

Thus the core costs $N L^{1/2} (\log L)^{1+o(1)}$, while constructing the rough-integer array costs $O(N L^{1/2} \log L)$. These are the competing costs; converting the array to the square-rough list costs the smaller $O(N L^{1/2} / \log L)$.

The present method keeps Harvey's square-prime reduction, reconstruction, direct tests, and conversion to primes. It replaces the procedure for deciding which blocks to mark. The weights have mean $O(1/L)$, the longer blocks make their large sums rare, and the computation allows controlled overestimation instead of listing the products. These three facts permit the threshold $R \asymp T/L$ without the earlier trade-off.

2 Preliminaries and parameters

All algorithms use a fixed number of Turing-machine tapes, and all running times count bit operations. In particular, reading or writing an integer with b binary digits costs $O(b)$ operations; it is not one unit-cost operation. This distinction will matter when we compare counting products with writing down all of them.

We write $\log$ for the natural logarithm and $\log_2$ for the base-two logarithm. The weight $\text{wt}(v)$ of a binary vector is its number of ones. For a condition E , the indicator $\mathbf{1}_E$ is 1 when E holds and 0 otherwise. For nonnegative quantities, $f \ll g$ means the same as $f = O(g)$: a

constant multiple of g bounds f for all sufficiently large inputs. The notation $f \asymp g$ means that both $f \ll g$ and $g \ll f$ hold, while $f = o(g)$ means that $f/g \rightarrow 0$. Constants may depend on the fixed sieve parameter s introduced below, but not on N . All parameter inequalities are understood to hold for sufficiently large N . Testing each candidate integer by trial division handles the finitely many smaller inputs.

Divisors are positive. In divisor sums and product counts, the indices m, e, d, k also range over positive integers unless a different range is specified.

In a running-time upper bound, the exponent $1 + o(1)$ allows a quantity that tends to 1 as $N \rightarrow \infty$. In particular, the bound in Theorem 1.1 is at most $O_\varepsilon(N(\log \log N)^{1+\varepsilon})$ for every fixed $\varepsilon > 0$. This notation does not assert an exact linear-time bound.

The notation below is our own. When comparing with [5], only N, T, R, a^r, c^r and wt retain their meanings from that paper.

2.1 Reconstruction, repair, and conversion

The proof uses three algorithms from Harvey. We state their complete input and output guarantees here, so that the new argument can be followed without using the details of their implementations.

For $T \mid N$, recall that $I_r = [rT, (r+1)T) \cap \mathbb{Z}$. Let $a^r \in \{0, 1\}^T$ be the indicator vector of the odd square-primes in I_r : its entry at position t is one precisely when $rT + t$ is an odd square-prime. Thus $\text{wt}(a^r) = w_r$. The first algorithm produces a candidate for each of these unknown vectors.

Theorem 2.1 ([5, Theorem 6.1]). *Given positive integers N, T, R with $T \mid N$ and $R \leq T/2$, a deterministic algorithm returns, in block order, vectors $c^r \in \{0, 1\}^T$, $0 \leq r < N/T$, satisfying*

$$\text{wt}(c^r) \leq R, \quad w_r \leq R \implies c^r = a^r.$$

For every fixed $C > 0$, if $\log T < C \log \log N$, its running time is

$$O_C \left(\left(1 + \frac{R \log N}{T} \right) N (\log \log N)^{1+o(1)} \right). \quad (2.1)$$

There are two different guarantees in this theorem. The running-time bound holds for every admissible input. On a particular block, the candidate is guaranteed to be correct when $w_r \leq R$. When $w_r > R$, the candidate has at most R ones and therefore cannot be correct; the theorem does not promise an error signal. This is why we must certify a block before using its candidate.

The core places no arithmetic restriction on T beyond $T \mid N$. Our later choice that T is a power of two serves a different purpose: it will ensure that the divisors used in the convolution have greatest common divisor at most 2 with T (Lemma 3.2).

The second algorithm supplies a way to repair the blocks that we cannot certify. It tests an arbitrary list of integers, rather than requiring us to test every integer below N .

Theorem 2.2 ([5, Proposition 9.5]). *Given an integer $N \geq 1$ and a list $\mathcal{R}$ of distinct integers in $[1, N)$, a deterministic algorithm returns, in increasing order, all square-primes in $\mathcal{R}$. Its running time is*

$$O(N) + |\mathcal{R}|(\log N)^{11.5+o(1)}. \quad (2.2)$$

The test returns square-primes of either parity. We keep only the odd ones when repairing our blocks. The term $O(N)$ is paid once for the whole list; the remaining cost is proportional to its length. Thus a list of size $O(N/L^{12})$ is affordable, because $L \asymp \log N$ and $12 > 11.5$.

The last algorithm turns the resulting complete square-prime list into the desired prime list.

Theorem 2.3 ([5, Proposition 7.13]). *Given the increasing list of all odd square-primes below N , where $N \geq 2$, a deterministic algorithm computes the increasing list of all primes below N in $O(N)$ bit operations.*

The completeness requirement matters: this theorem takes every odd square-prime below N , not just the ones found on unmarked blocks. The repair step is what supplies the missing part of that list.

We will also switch between indicator vectors and sorted lists, and merge or sort lists on tapes. The bounds we use are [5, Lemmas 3.1–3.3]:

- A binary vector v of length n can be converted to the sorted list of its nonzero positions, or conversely, in $O(n + \text{wt}(v) \log(2+n))$ operations. A scan costs $O(n)$, including maintenance of a binary position counter: its j th bit changes only once per 2^j increments, so the total counter work is $O(n)$. Writing the positions of the ones costs $O(\text{wt}(v) \log(2+n))$.
- Two sorted lists with n records in total, each of b bits, can be merged in $O(bn)$ operations. Comparing and copying one record costs $O(b)$.
- Such a list can be sorted in $O(bn \log(2+n))$ operations, provided comparisons cost $O(b)$. Merge sort uses $O(\log(2+n))$ passes, each of which reads and writes $O(bn)$ bits.

The use of $\log(2+n)$ makes these statements valid for empty lists and one-element lists as well.

2.2 Choice of parameters

We now fix the parameters used throughout the proof. Their exact values make the later estimates concrete; the reasons for the powers of L are collected in Table 1.

Let N_0 be the input bound, and put $L = \lceil \log_2(2N_0) \rceil$. Thus L measures the number of bits needed for integers of size about N_0 . Choose the block length T to be the least power of two that is at least L^{18} , and set

$$N = T \left\lceil \frac{N_0}{T} \right\rceil.$$

The enlarged bound N is a multiple of T , so every block has the same length. We enumerate up to N and discard primes at least N_0 at the end. We keep the original value of L .

The enlargement does not change the asymptotic size of the problem. Indeed, $0 \leq N - N_0 < T < 2L^{18}$, whereas $2N_0 \leq 2^L < 4N_0$. For sufficiently large N_0 , we have $T \leq N_0$, and hence $N < 2N_0 \leq 2^L$. In particular,

$$T \mid N, \quad N \leq 2^L \leq 4N, \quad L \asymp \log N.$$

All subsequent uses of N refer to this enlarged bound.

The cutoff D limits the square multiplier m in the part treated by a mean-square estimate. The sieve level Z limits the divisor indices in the sieve, and the sifting limit y specifies which small prime factors enter the sieve. The parameter η controls the width of the product enclosures used by the convolution algorithm. We take

$$D = L^{12}, \quad \eta = L^4, \quad Z = \lfloor N^{1/1000} \rfloor, \quad y = \lfloor Z^{1/s} \rfloor, \quad (2.3)$$

where s is a sufficiently large fixed positive integer. The threshold for accepting a block is

$$R = \left\lceil \frac{2C_1 T}{L} \right\rceil, \quad (2.4)$$

where we take $C_1 = 200000s$. The factor 2 leaves room for two contributions: the exact majorant sum and the excess in its computed upper bound. The large fixed constant places the threshold above the mean of the majorant; Section 4.2 checks this explicitly. Neither this constant nor the exponent 18 is claimed to be optimal.

All these parameters are computed exactly by integer arithmetic in time polynomial in L . For example, Z is the integer 1000th root of N , and y is the integer s th root of Z . Binary search finds the largest integer z for which $z^{1000} \leq N$, and then the largest integer u for which $u^s \leq Z$. Each search uses $O(L)$ comparisons. The exponents 1000 and s are fixed, so the powers have $O(L)$ bits and can be computed in polynomial time.

We will need a lower bound for $\log y$ as well as an upper bound for y . For sufficiently large N , the two quantities being rounded down are at least two, and hence

$$\frac{1}{2}N^{1/1000} \leq Z \leq N^{1/1000}, \quad \frac{1}{2}Z^{1/s} \leq y \leq Z^{1/s}.$$

Taking logarithms in the two inequalities gives

$$\frac{\log N}{1000s} - \left(1 + \frac{1}{s}\right) \log 2 \leq \log y \leq \frac{\log N}{1000s}. \quad (2.5)$$

In particular, $\log y \geq (\log N)/(2000s)$ for sufficiently large N . Here s is fixed: the subtracted term is a constant, while $\log N$ grows.

The repair algorithm explains the choice of 12: its per-integer cost has exponent $11.5 + o(1)$, so a marked length of N/L^{12} is small enough. The large-square-divisor tail contributes a marked length $O(N/D)$, so we take $D = L^{12}$.

The choices of η and T balance two costs. The convolution construction groups factors on the logarithmic scale, with $K = \lceil N/\eta \rceil$ bins per unit of base-two logarithm. Factors range from 1 to N , so their logarithms occupy an interval of length $O(L)$. There are therefore $O(KL) = O(NL/\eta)$ bin thresholds. Computing an exponential threshold to absolute error at most $1/4$ costs $O(L^3)$ bit operations by Lemma B.1; this is the accuracy used in the enclosure construction. The numerical work is consequently

$$O\left(\frac{NL}{\eta} L^3\right) = O\left(\frac{NL^4}{\eta}\right).$$

Taking $\eta = L^4$ makes this cost $O(N)$. Increasing η would reduce the number of thresholds, but would also widen the product enclosures and increase their boundary errors.

To see the resulting condition on T , recall that a divisor in the signed list has the form $d = m^2 e$, with m odd and e squarefree. Since T is a power of two, $(d, T) \leq 2$. Lemma 5.2 bounds the excess weight by twice the number of products $dk \leq 2N$ within 64η of a block boundary; the range up to $2N$ includes enclosures crossing the final boundary N . For one d , there are $2N/d + O(1)$ candidate products. Section 5.5 uses $(d, T) \leq 2$ to show that only an $O(\eta/T)$ proportion in each complete residue period can be near a boundary, with a separate error for the final incomplete period. Summing the complete-period terms gives a contribution to the excess bound of at most a constant multiple of

$$\frac{N\eta}{T} \sum_d \frac{1}{d} = \frac{N\eta}{T} V \ll \frac{N\eta L}{T}.$$

The bound $V \ll L$ is proved in Lemma 3.2.

Consider a block whose exact majorant sum is at most $C_1 T/L$. If its computed upper bound exceeds $R \geq 2C_1 T/L$, the overestimate must exceed $C_1 T/L$. Marking such an additional block adds T positions to the repair list. Thus converting excess weight to marked length multiplies the bound by L/C_1 . To keep the result within $O(N/L^{12})$, it is enough that

$$\frac{N\eta L^2}{T} \leq \frac{N}{L^{12}}, \quad \text{or equivalently} \quad T \geq \eta L^{14} = L^{18}.$$

The two factors of L have different sources: the reciprocal sum over divisors and the conversion from excess weight to marked length. This choice also meets the weaker requirement $T \geq L^{17}$ from the mean-square estimate in Section 4.

Table 1 records these inequalities for later use; Sections 4–6 prove the estimates behind them, including the rounding and incomplete-period errors omitted from the preceding scale calculation. Here $\mathcal{R}$ is the set tested in Step 6. The error terms involving Z in those estimates are $o(N/L^c)$ for every fixed c .

Requirement	Sufficient inequality	Choice or consequence
Linear repair cost	$ \mathcal{R} \ll N/L^{12}$	$12 > 11.5$
Square-divisor tail	$N/D \leq N/L^{12}$	$D = L^{12}$
Variance contribution	$NL^5/T \leq N/L^{12}$	$T \geq L^{17}$
Numerical evaluations	$(NL/\eta)L^3 = O(N)$	$\eta = L^4$
Excess of computed bounds	$L(N\eta L/T) \leq N/L^{12}$	$T \geq \eta L^{14}$
Core complexity	$R \asymp T/L, \log T = O(\log L)$	our $T \asymp L^{18}$

Table 1: Constraints on the parameters. We choose the final block length to bound the excess of the computed bounds.

For reference, the main parameters and counts are summarized below.

Symbol	Meaning
N, L	enlarged bound and bit-length parameter, $N \leq 2^L \leq 4N$
T, R	block length and weight threshold
w_r, W_r, U_r	odd-square-prime count, majorant sum, and computed bound
D	cutoff on m in the square-divisor sum, retaining $m \leq D$
Z, y	sieve level and sifting limit, with $y = \lfloor Z^{1/s} \rfloor$
S, S, V	signed divisor list, its length, and the sum of its reciprocals $1/d$
η, K, J	bin parameter, bins per factor of two, number of bins
s, β	sieve parameters
ν, ν_D	mean values of θ and of the part of W with $m \leq D$
C_1	constant in the weight threshold R
Q	upper bound on divisor indices in Lemma 4.1
$\mathcal{R}$	distinct positive integers in the marked blocks

The values $\beta = 30$ (Appendix A) and $Z = \lfloor N^{1/1000} \rfloor$ follow the choices in [7, Section 2]. An optional range of other exponents for Z is discussed at the end of Appendix A, after the estimates that constrain it have been proved.

3 The sieve majorant

A *majorant* is an upper bound given by a function: here we want a nonnegative function that is at least one at every odd square-free whose prime part exceeds y . We first construct a weight θ for integers with no small prime factor, then apply it after removing square divisors.

Write $P(y) = \prod_{p < y} p$. The condition $(n, P(y)) = 1$ means that no prime below y divides n . Its exact indicator is

$$\mathbf{1}_{(n, P(y))=1} = \prod_{p < y} (1 - \mathbf{1}_{p|n}).$$

If one of these primes divides n , one factor is zero; otherwise every factor is one. Expanding the product gives inclusion–exclusion: subtract multiples of each prime, add back multiples of each pair of distinct primes, and continue through all subsets. Each subset contributes a divisor term, with sign determined by the number of primes in the subset.

These divisor indices can be much larger than the allowed cutoff Z . Simply deleting all large indices can destroy the upper-bound property. For example, sieving by 2 and 3 gives

$$1 - \mathbf{1}_{2|n} - \mathbf{1}_{3|n} + \mathbf{1}_{6|n}.$$

If we keep only indices at most 5, the last term disappears. The value at $n = 6$ becomes $1 - 1 - 1 = -1$, below the true indicator 0. The beta-sieve selects terms in a way that avoids this failure. Its divisor indices stay below Z , but the remaining sum is still an upper bound. Appendix A gives the selection rule and proves this property.

We also need to know how much weight this upper bound assigns on average. For comparison, the exact indicator above is periodic with period $P(y)$. Every divisor index e in its expansion divides $P(y)$, so exactly $P(y)/e$ integers in $\{1, \dots, P(y)\}$ are divisible by e . Averaging the expansion therefore replaces each divisibility indicator by $1/e$. Expanding the resulting product shows that the proportion surviving the exact sieve is

$$\prod_{p < y} \left(1 - \frac{1}{p}\right).$$

The imported sieve theorem says that the selected upper-bound weights have mean at most twice this proportion. Thus shortening the divisor sum does not increase its mean by more than a constant factor.

We use the following standard sieve result. Its pointwise inequality is proved from the explicit coefficients in Appendix A. Its estimate for the mean is the fundamental lemma of the upper beta-sieve, which is an external input to this paper. Recall that an integer is squarefree if no prime square divides it.

Lemma 3.1 (Upper-bound sieve weights). *Given a sufficiently large fixed integer s , we can compute coefficients $\lambda_e \in \{-1, 0, 1\}$, supported on squarefree divisors of $P(y)$ with $e \leq Z$, such that $\lambda_1 = 1$ and*

$$\theta(n) := \sum_{e|n} \lambda_e \geq \mathbf{1}_{(n, P(y))=1} \geq 0 \quad (n \geq 1). \quad (3.1)$$

The mean value of θ satisfies

$$0 \leq \nu := \sum_{e \leq Z} \frac{\lambda_e}{e} \ll \frac{1}{L}. \quad (3.2)$$

The coefficients can be computed in $O(Z^2 L^4)$ bit operations.

Proof. Take the upper beta-sieve coefficients with $\beta = 30$, level Z , and sifting limit y , as defined in Appendix A. Their construction gives (3.1), the usual upper-bound inequality [3, (6.26)–(6.27)]. The fundamental lemma [3, (6.31)–(6.33), Lemma 6.8] gives

$$\prod_{p < y} \left(1 - \frac{1}{p}\right) \leq \nu \leq 2 \prod_{p < y} \left(1 - \frac{1}{p}\right)$$

for the divisibility frequencies $1/e$, when $\log Z / \log y$ exceeds a sufficiently large constant depending on the fixed β . Our choice $y = \lfloor Z^{1/s} \rfloor$ gives $\log Z / \log y \geq s$ for $y > 1$, so choosing the fixed integer s sufficiently large meets this condition. These statements, with the same coefficients, are also stated in [7, (14), (28)].

We can bound the product without an asymptotic formula for primes. Expand each factor of $\prod_{p < y} (1 - 1/p)^{-1}$ as a geometric series. Every integer $j < y$ has all its prime factors below y , so the expansion contains its reciprocal. Therefore

$$\prod_{p < y} \left(1 - \frac{1}{p}\right)^{-1} \geq \sum_{1 \leq j < y} \frac{1}{j} \geq \log y.$$

It follows that $\nu \leq 2 / \log y$. By (2.5), $\log y \geq (\log N) / (2000s)$ for large N . Since s is fixed and $L \asymp \log N$, this gives $\nu \ll 1/L$. Trial division and the defining integer inequalities give the stated construction cost, as detailed in Appendix A. $\square$

The coefficients can have either sign, but the weight $\theta(n)$ is nonnegative for every positive integer n . This distinction will be needed twice: nonnegativity lets us compare sums over nested intervals, while the signed divisor expansion makes convolution possible.

The number ν is the mean of θ in the following precise sense. Expanding the divisor sum gives, for every real $x \geq 0$,

$$\sum_{1 \leq n \leq x} \theta(n) = \nu x + O(Z). \quad (3.3)$$

Indeed,

$$\sum_{1 \leq n \leq x} \theta(n) = \sum_{e \leq Z} \lambda_e \left\lfloor \frac{x}{e} \right\rfloor = x \sum_{e \leq Z} \frac{\lambda_e}{e} + O\left(\sum_{e \leq Z} |\lambda_e|\right).$$

Each replacement of $\lfloor x/e \rfloor$ by x/e contributes an error of absolute value at most one, and there are at most Z coefficients. Thus the $O(Z)$ error is uniform, including when $x < Z$. For fixed sieve parameters, dividing by x and letting $x \rightarrow \infty$ shows that the average of θ tends to ν .

This average does not by itself give a useful estimate on our short blocks. Subtracting (3.3) at $a + T$ and a , for $a \geq 0$, gives

$$\sum_{a < n \leq a+T} \theta(n) = \nu T + O(Z).$$

The main term is $O(T/L)$. Recall that $Z \asymp N^{1/1000}$ and $T \asymp L^{18}$; the error allowed here is therefore much larger:

$$\frac{Z}{T/L} \asymp \frac{N^{1/1000}}{L^{17}} \longrightarrow \infty.$$

The difficulty is the accumulated error from counting multiples of many divisors. The global mean estimate alone cannot certify an individual block. Section 4 instead bounds the number of blocks whose weights exceed the threshold, by estimating the average squared deviation over intervals.

The weight also assigns mass to some composites. If p and q are distinct primes above y , the only supported divisor of pq is 1, so $\theta(pq) = 1$. The same reasoning gives $\theta(p) = 1$. Thus the weight retains every sufficiently large prime, but does not distinguish such a prime from a product of two large primes. The estimates on its mean and deviations control this extra weight.

To pass from primes to square-primes, apply θ to each quotient left after removing an odd square divisor. Define

$$W(n) = \sum_{\substack{m \geq 1, m^2 | n \\ m \text{ odd}}} \theta(n/m^2) \quad (n \geq 1), \quad W_r = \sum_{\substack{n \in I_r \\ n \geq 1}} W(n). \quad (3.4)$$

We sum only over odd m , for two reasons. The square-primes we need to bound are odd, so their m is odd. More importantly, when m is odd every divisor $m^2 e$ in the list below satisfies $(m^2 e, T) \leq 2$ (Lemma 3.2). We need this bound to count products near block boundaries in Section 5. If $n = m^2 p$ is an odd square-prime with $p > y$, the term indexed by this m is $\theta(p) = 1$. Indeed, the only divisors of p are 1 and p , and $\lambda_p = 0$ because supported indices have all their prime factors below y . The term $\lambda_1 = 1$ remains. Every other term in $W(n)$ is nonnegative, so $W(n) \geq 1$. This argument does not cover $p \leq y$; their blocks are marked separately in the algorithm.

Expanding θ gives the identity

$$W(n) = \sum_{\substack{m \geq 1 \\ m^2 | n}} \sum_{\substack{\text{odd } e | n/m^2}} \lambda_e = \sum_{\substack{m \geq 1 \text{ odd}, e \leq Z \\ m^2 e | n}} \lambda_e.$$

The condition $e | n/m^2$ has become $m^2 e | n$. Hence, for $n < N$, $W(n)$ is a signed count of divisors from the fixed list

$$\mathcal{S} = \{(m^2 e, \lambda_e) : m \geq 1 \text{ odd}, e \leq Z, \lambda_e \neq 0, m^2 e \leq N\}. \quad (3.5)$$

Each divisor appears once. To see this, factor d into primes. In an identity $d = m^2 e$ with e squarefree, a prime divides e exactly when its exponent in d is odd. Thus e is determined by d , and then $m = \sqrt{d/e}$ is determined too. No coprimality between m and e is needed. For $1 \leq n < N$,

$$W(n) = \sum_{(d, \sigma) \in \mathcal{S}} \sigma \mathbf{1}_{d|n}. \quad (3.6)$$

There are two sizes associated with this list. Its length measures how many divisor records the algorithm must store. Each record (d, σ) also represents many products $d, 2d, 3d, \dots$; their number up to N is $\lfloor N/d \rfloor$. For the later boundary count, we will therefore need the sum of the frequencies $1/d$ as well as the number of records.

Lemma 3.2. *The list $\mathcal{S}$ has length $S \ll \sqrt{NZ}$. The sum of its reciprocals is*

$$V := \sum_{(d, \sigma) \in \mathcal{S}} \frac{1}{d} \ll L.$$

Every divisor in the list satisfies $(d, T) \leq 2$.

Proof. For a fixed e , the condition $m^2 e \leq N$ permits at most $\sqrt{N/e}$ values of m . Counting all such values, including even ones, only enlarges the count. Thus

$$S \leq \sqrt{N} \sum_{e \leq Z} e^{-1/2} \ll \sqrt{NZ}.$$

For the reciprocal sum, dropping the size restriction on m gives

$$V \leq \sum_{e \leq Z} \frac{1}{e} \sum_{m \geq 1} \frac{1}{m^2} \ll 1 + \log Z \ll L.$$

Here the harmonic sum is $O(1 + \log Z)$, and $\sum_{m \geq 1} m^{-2}$ converges. Since m is odd and e is squarefree, $m^2 e$ contains at most one factor of 2. The gcd assertion follows because T is a power of two. $\square$

In particular, the number of product pairs represented by the list, if all products up to N were written out, would satisfy

$$NV - S \leq \sum_{(d, \sigma) \in \mathcal{S}} \left\lfloor \frac{N}{d} \right\rfloor \leq NV.$$

Each floor differs from N/d by less than one, and there are S records. This count ignores signs: opposite signs do not save the cost of generating two records. Hence the bound on S does not justify writing out all represented products. The single entry $d = 1$ already represents N products. A literal list of these integers requires $\Theta(N \log N)$ bits, beyond the intended running time. Section 5 counts groups of these pairs without listing the products.

4 Few blocks have large majorant sums

Section 3 gives a nonnegative weight $W(n)$, and W_r is its sum on block I_r . We now prove that the blocks with $W_r > C_1 T/L$ have total length $O(N/L^{12})$. This is a statement about how many such blocks there are. The next section will explain how to find a suitable set of blocks to mark.

A bound for the total weight does not suffice. Indeed, (3.3) gives

$$\begin{aligned} \sum_r W_r &= \sum_{\substack{m \leq \sqrt{N} \\ m \text{ odd}}} \sum_{1 \leq q < N/m^2} \theta(q) \\ &\ll \frac{N}{L} \sum_{m \geq 1} \frac{1}{m^2} + Z\sqrt{N} \ll \frac{N}{L}. \end{aligned}$$

Here the change of variables is $n = m^2 q$; including even m only increases the upper bound. The series $\sum m^{-2}$ converges, and $Z\sqrt{N} \leq N^{501/1000} = o(N/L)$. Since every block with $W_r > C_1 T/L$ uses more than $C_1 T/L$ of this total, this estimate gives only

$$T \#\{r : W_r > C_1 T/L\} \leq \frac{L}{C_1} \sum_r W_r = O(N).$$

Increasing the fixed constant C_1 can improve a constant here, but cannot supply the factor L^{-12} that the repair budget requires. We need information about how the weight is distributed among blocks.

We split the square-divisor sum at $m = D$. For $m \leq D$, we bound the average squared deviation of an interval sum from its mean. A large block sum then forces a large squared deviation, and the mean-square bound limits how many such blocks can occur. For $m > D$, the corresponding series is $\sum_{m > D} m^{-2} \leq 1/D$. This will give a total tail weight $O(N/(DL))$, smaller by the factor $1/D$ needed for a first-moment bound to control its exceptional blocks.

The cutoff also makes the mean-square argument possible. Expanding $\theta(n/m^2)$ produces divisor indices m^2e , with $e \leq Z$. For $m \leq D$ these indices are at most $Q = D^2Z$. The lemma below has an error Q^4 , which comes from incomplete periods of pairs of divisor counts. We will keep track of this error and verify that our choices of D and Z make it affordable.

4.1 Variance of a bounded divisor sum

The estimate below applies to any divisor sum with bounded coefficients and bounded divisor indices. It does not require sieve weights. The centering term has a simple meaning: multiples of e occur with frequency $1/e$, so a sum $\sum_{e|n} \gamma_e$ has mean $\sum_e \gamma_e/e$. Over an interval of length H , the corresponding mean term is therefore $H \sum_e \gamma_e/e$.

Mean-square estimates of this kind are classical. Matomäki [7, Proposition 5.1] gives a formula with an explicit error for a closely related integral, using the same floor-error functions as below. She compares it with [3, Proposition 6.25]. We include the proof of the upper bound we need, so that its dependence on the largest divisor index is explicit.

Lemma 4.1. *Let $Q \geq 1$ be an integer. Suppose that the real coefficients γ_e satisfy $|\gamma_e| \leq 1$ for $1 \leq e \leq Q$ and $\gamma_e = 0$ for $e > Q$. Define*

$$f(n) = \sum_{e|n} \gamma_e \quad (n \in \mathbb{Z}), \quad \bar{f} = \sum_{e \leq Q} \frac{\gamma_e}{e},$$

where every positive integer divides zero. If $H \geq 1$ and $\mathcal{I}$ is a finite real interval of length X , then

$$\int_{\mathcal{I}} \left| \sum_{x-H < n \leq x} f(n) - H\bar{f} \right|^2 dx \ll XH(1 + \log Q)^3 + Q^4. \quad (4.1)$$

The implied constant is absolute.

Proof. For a positive integer d , put

$$E_d(x) = \left\lfloor \frac{x}{d} \right\rfloor - \left\lfloor \frac{x-H}{d} \right\rfloor - \frac{H}{d}.$$

The difference of floors counts the multiples of d in $(x-H, x]$; this remains true when the interval contains negative integers or zero. The term H/d is the mean count. Thus, after interchanging finite sums,

$$\sum_{x-H < n \leq x} f(n) - H\bar{f} = \sum_{d \leq Q} \gamma_d E_d(x).$$

To compute the error, write $x/d = j + u$ and $H/d = q + \alpha$, where j, q are integers and $0 \leq u, \alpha < 1$. Then

$$\left\lfloor \frac{x}{d} \right\rfloor - \left\lfloor \frac{x-H}{d} \right\rfloor = q - \lfloor u - \alpha \rfloor = q + \mathbf{1}_{u < \alpha}.$$

Indeed, $u - \alpha$ lies in $(-1, 1)$, and its floor is -1 exactly when $u < \alpha$. Subtracting $H/d = q + \alpha$ gives $E_d(x) = \mathbf{1}_{u < \alpha} - \alpha$. Adding d to x leaves u unchanged, so E_d has period d . During one period, u runs through $[0, 1)$ at constant speed. Thus E_d is $1 - \alpha$ for a proportion α of the period and $-\alpha$ for the rest. Its mean is $\alpha(1 - \alpha) + (1 - \alpha)(-\alpha) = 0$, and its absolute value is at most one.

Using only $|E_d(x)| \leq 1$ would give $|\sum_d \gamma_d E_d(x)| \leq Q$, and hence an integral bound XQ^2 . In our application below, $H = 3T \asymp (\log N)^{18}$, while $Q = D^2Z$ grows as a fixed positive power

of N times a power of $\log N$. Thus XQ^2 is much larger than the main term $XH(1 + \log Q)^3$ in (4.1). Instead, expand the square before estimating:

$$\int_{\mathcal{I}} \left| \sum_{d \leq Q} \gamma_d E_d(x) \right|^2 dx = \sum_{d, e \leq Q} \gamma_d \gamma_e \int_{\mathcal{I}} E_d(x) E_e(x) dx. \quad (4.2)$$

The coefficients and errors are real, and all sums are finite. Thus our task is to estimate the average product of the counting errors for two divisors.

Fix d, e , put $g = (d, e)$, and write $d = ga$, $e = gb$, so that a and b are coprime. The product $E_d E_e$ has period $P = \text{lcm}(d, e) = gab$. Define its average over a period by

$$\Gamma(d, e) = \frac{1}{P} \int_0^P E_d(x) E_e(x) dx.$$

Shifting the endpoint by g moves through a positions within a period of E_d and b positions within a period of E_e . For example, with $d = 6$ and $e = 10$, shifts by $g = 2$ return to the same position after three shifts for the first function and five for the second. Over fifteen shifts, all $3 \cdot 5$ pairs of positions occur. In general, coprimality of a, b will give the same pairing. This is why we separate the endpoint into a position $t \in [0, g)$ and an integer number of shifts by g .

First fix $t \in [0, g)$. We claim that

$$\sum_{q=0}^{a-1} E_d(t + qg) = E_g(t). \quad (4.3)$$

To see this, consider the counts before subtracting their means. A multiple dk in $(t + qg - H, t + qg]$ becomes, after subtracting qg , the multiple $g(ak - q)$ in $(t - H, t]$. As k runs through the integers and q through $0, \dots, a-1$, the integer $ak - q$ takes every integer value exactly once. Indeed, for a prescribed integer r , choose the unique q in this range with $q \equiv -r \pmod{a}$; then $k = (r + q)/a$ is forced. Summing the counts therefore counts each multiple of g in $(t - H, t]$ once. The means also add correctly: $a(H/d) = H/g$. Subtracting them proves (4.3). The same argument gives

$$\sum_{q=0}^{b-1} E_e(t + qg) = E_g(t).$$

Next let q run through $0, \dots, ab-1$. By periodicity, $E_d(t + qg)$ depends only on q modulo a , and $E_e(t + qg)$ depends only on q modulo b . Every pair of these residues occurs exactly once. For completeness, if two values of q gave the same pair, their difference would be divisible by both a and b , hence by ab because a and b are coprime. In the given range the two values must be equal. There are ab values and ab residue pairs, so all pairs occur. It follows that

$$\begin{aligned} \sum_{q=0}^{ab-1} E_d(t + qg) E_e(t + qg) &= \left(\sum_{i=0}^{a-1} E_d(t + ig) \right) \left(\sum_{j=0}^{b-1} E_e(t + jg) \right) \\ &= E_g(t)^2. \end{aligned}$$

Partition $[0, P)$ into the ab intervals $qg + [0, g)$ and use $x = t + qg$ on the q th interval. The

preceding identity gives

$$\begin{aligned}
\int_0^P E_d(x)E_e(x) dx &= \sum_{q=0}^{ab-1} \int_0^g E_d(t+qg)E_e(t+qg) dt \\
&= \int_0^g \sum_{q=0}^{ab-1} E_d(t+qg)E_e(t+qg) dt \\
&= \int_0^g E_g(t)^2 dt.
\end{aligned}$$

Dividing by the full period length P therefore gives

$$\Gamma(d, e) = \frac{1}{P} \int_0^g E_g(t)^2 dt.$$

It remains to evaluate this one square. Write $H/g = \lfloor H/g \rfloor + \alpha$, with $0 \leq \alpha < 1$. The floor calculation above, with $d = g$ and $u = t/g$, shows that for $0 \leq t < g$, the interval $(t - H, t]$ contains $\lfloor H/g \rfloor + 1$ multiples of g when $t < g\alpha$, and $\lfloor H/g \rfloor$ otherwise. Thus E_g takes the values $1 - \alpha$ and $-\alpha$ in proportions α and $1 - \alpha$. When $\alpha = 0$, the first range is empty and E_g is identically zero. Its mean square over $[0, g)$ is therefore

$$\alpha(1 - \alpha)^2 + (1 - \alpha)\alpha^2 = \alpha(1 - \alpha).$$

Since $g/P = g^2/(de)$, we obtain the exact formula

$$\Gamma(d, e) = \frac{g^2}{de} \alpha(1 - \alpha) \geq 0. \quad (4.4)$$

Because $0 \leq \alpha < 1$ and $\alpha \leq H/g$, this formula gives

$$0 \leq \Gamma(d, e) \leq \frac{g^2}{de} \alpha \leq \frac{H(d, e)}{de}. \quad (4.5)$$

The two errors can therefore have a small average product even though each can have absolute value close to one.

Now return to $\mathcal{I}$. Every interval of length P has the same integral of $E_d E_e$. Split $\mathcal{I}$ into complete periods and one remaining interval $\mathcal{J}$ of length less than P . The complete periods contribute their length times $\Gamma(d, e)$. Consequently

$$\begin{aligned}
\int_{\mathcal{I}} E_d(x)E_e(x) dx &= X\Gamma(d, e) + R_{d,e}, \\
R_{d,e} &= \int_{\mathcal{J}} (E_d(x)E_e(x) - \Gamma(d, e)) dx.
\end{aligned}$$

We have $|E_d E_e| \leq 1$ and $|\Gamma(d, e)| \leq 1$, so

$$|R_{d,e}| \leq 2|\mathcal{J}| \leq 2P = 2\text{lcm}(d, e).$$

Each pair uses its own period $\text{lcm}(d, e) \leq de \leq Q^2$. We do not need to use a common period for all of $E_1, \dots, E_Q$, which would be $\text{lcm}(1, \dots, Q)$.

Substitute this expression into (4.2). The coefficients can have either sign; taking absolute values of their contributions gives the upper bound

$$\begin{aligned} \int_{\mathcal{I}} \left| \sum_{d \leq Q} \gamma_d E_d(x) \right|^2 dx &\leq X \sum_{d,e \leq Q} |\gamma_d \gamma_e| |\Gamma(d,e)| \\ &\quad + \sum_{d,e \leq Q} |\gamma_d \gamma_e| |R_{d,e}| \\ &\leq XH \sum_{d,e \leq Q} \frac{(d,e)}{de} + 2 \sum_{d,e \leq Q} \text{lcm}(d,e). \end{aligned}$$

For the second sum, $\text{lcm}(d,e) \leq de$, hence

$$\sum_{d,e \leq Q} \text{lcm}(d,e) \leq \sum_{d,e \leq Q} de = \left(\sum_{d \leq Q} d \right)^2 = O(Q^4).$$

There are at most Q^2 pairs, each with an incomplete-period error of size at most $O(Q^2)$. This is the source of the Q^4 term.

For the first sum, group the pairs by their greatest common divisor. The representation $d = ga$, $e = gb$ gives a bijection between pairs $d, e \leq Q$ with $(d,e) = g$ and coprime pairs $a, b \leq Q/g$. Their summand is $g/((ga)(gb)) = 1/(gab)$. Therefore

$$\begin{aligned} \sum_{d,e \leq Q} \frac{(d,e)}{de} &= \sum_{g \leq Q} \frac{1}{g} \sum_{\substack{a,b \leq Q/g \\ (a,b)=1}} \frac{1}{ab} \\ &\leq \sum_{g \leq Q} \frac{1}{g} \left(\sum_{a \leq Q/g} \frac{1}{a} \right)^2. \end{aligned}$$

The inequality drops the coprimality restriction from a nonnegative sum; the resulting double sum factors into two harmonic sums. For $u \geq 1$, comparison with the integral of $1/t$ gives $\sum_{1 \leq a \leq u} 1/a \leq 1 + \log u$. Thus each inner harmonic sum is at most $1 + \log Q$, and

$$\sum_{d,e \leq Q} \frac{(d,e)}{de} \leq (1 + \log Q)^2 \sum_{g \leq Q} \frac{1}{g} \leq (1 + \log Q)^3.$$

The three logarithmic factors come from the two reduced divisors a, b and their common factor g . Substituting the two sum bounds proves (4.1). $\square$

4.2 The truncated majorant

We apply Lemma 4.1 to the part of W with $m \leq D$. The task is to bound the total length of blocks on which this part alone exceeds $C_1 T/(2L)$, half the threshold for the full sum. To use the lemma, we must write this part as one divisor sum with coefficients of absolute value at most one. Grouping the original sieve terms could in principle add several coefficients together; the squarefree support will prevent that. We must also turn a large sum on one fixed block into a lower bound for an integral over moving endpoints.

For positive integers n , define

$$W_{\leq D}(n) = \sum_{\substack{1 \leq m \leq D, \, m \text{ odd} \\ m^2 | n}} \theta(n/m^2), \quad W_r^{\leq D} = \sum_{\substack{n \in I_r \\ n \geq 1}} W_{\leq D}(n).$$

Every summand is nonnegative by Lemma 3.1, so $W_{\leq D}(n) \geq 0$. Let

$$\mathcal{E} = \{r : W_r^{\leq D} > C_1 T / (2L)\}$$

be the set of blocks that exceed the threshold for this part.

Divisor coefficients and mean. Expanding $\theta(n/m^2)$ gives

$$W_{\leq D}(n) = \sum_{\substack{1 \leq m \leq D, \\ m^2 | n}} \sum_{\substack{m \text{ odd} \\ e | n/m^2}} \lambda_e = \sum_{d | n} \gamma_d, \quad \gamma_d := \sum_{\substack{1 \leq m \leq D, \\ m \text{ odd}, \\ m^2 e = d}} \lambda_e.$$

The change of index uses the equivalence $e | n/m^2$ if and only if $m^2 e | n$, once $m^2 | n$. Here λ_e is the original sieve coefficient, indexed by e . The new coefficient γ_d collects all terms with the same combined divisor $d = m^2 e$. These are the coefficients to which we will apply Lemma 4.1; its divisor index is now called d rather than e .

A term with $\lambda_e \neq 0$ has squarefree e and satisfies $|\lambda_e| \leq 1$, by Lemma 3.1. This squarefree restriction makes the representation $d = m^2 e$ unique. To see why, write the exponent of a prime in d as $2b + \varepsilon$, where $\varepsilon \in \{0, 1\}$. That prime must occur to exponent b in m and exponent ε in e . Thus at most one summand in γ_d can be nonzero. The restrictions that m be odd and $m \leq D$ can remove that summand, but cannot introduce another one. Consequently $|\gamma_d| \leq 1$. Also $m \leq D$ and $e \leq Z$ imply $d = m^2 e \leq D^2 Z$. We have therefore proved

$$W_{\leq D}(n) = \sum_{d | n} \gamma_d, \quad |\gamma_d| \leq 1, \quad \gamma_d = 0 \ (d > Q), \quad Q = D^2 Z. \quad (4.6)$$

In particular, the support condition in Lemma 4.1 follows from the construction; it is not an additional assumption about the sieve weights.

The mean parameter in that lemma is

$$\begin{aligned} \nu_D &:= \sum_{d \leq Q} \frac{\gamma_d}{d} = \sum_{\substack{1 \leq m \leq D \\ m \text{ odd}}} \sum_{e \leq Z} \frac{\lambda_e}{m^2 e} \\ &= \left(\sum_{e \leq Z} \frac{\lambda_e}{e} \right) \left(\sum_{\substack{1 \leq m \leq D \\ m \text{ odd}}} \frac{1}{m^2} \right) = \nu \sum_{\substack{1 \leq m \leq D \\ m \text{ odd}}} \frac{1}{m^2}. \end{aligned}$$

These are finite sums, and every pair in them satisfies $m^2 e \leq D^2 Z = Q$. The factorization therefore loses no terms. The proof of Lemma 3.1 gives $0 \leq \nu \leq 2/\log y$. Using

$$\sum_{m \geq 1} \frac{1}{m^2} \leq 1 + \int_1^\infty \frac{dt}{t^2} = 2,$$

we obtain $0 \leq \nu_D \leq 4/\log y \ll 1/L$.

From an exceptional block to a large deviation. The mean-square lemma integrates over a moving endpoint x , whereas $\mathcal{E}$ concerns sums on fixed blocks. A large value at a single endpoint would give no lower bound for the integral, since a point has length zero. We instead enlarge the moving window so that it contains a fixed block for a whole interval of endpoints of length T . Nonnegativity will then make the moving sum at least as large as that block sum throughout this interval. Different blocks will supply endpoint intervals with disjoint interiors.

For a block $I_r = [rT, (r+1)T) \cap \mathbb{Z}$, where $0 \leq r < N/T$ and $T \mid N$, take the endpoint interval

$$\mathcal{J}_r = [(r+1)T, (r+2)T].$$

For every $x \in \mathcal{J}_r$, the interval $(x-3T, x]$ contains all of I_r . Its left endpoint is at most $(r-1)T < rT$, and its right endpoint is at least $(r+1)T$. In particular, the open left endpoint of the moving interval does not exclude the first integer of the block. The interval $(x-3T, x]$ contains the integers being summed; $\mathcal{J}_r$ specifies the allowed positions of its endpoint.

Near the beginning of the range, these moving intervals can include nonpositive integers. We therefore extend $W_{\leq D}$ to all of $\mathbb{Z}$ by the divisor sum in (4.6). Setting the weight to zero at nonpositive integers would generally destroy that divisor-sum formula, so it would no longer satisfy the lemma's hypotheses. We must check that the extension by the formula is still nonnegative. For negative n , divisibility gives the same value as at $|n|$. At zero, every positive integer divides zero, so

$$W_{\leq D}(0) = \sum_d \gamma_d = \#\{1 \leq m \leq D : m \text{ odd}\} \sum_{e \in \mathbb{Z}} \lambda_e = \#\{1 \leq m \leq D : m \text{ odd}\} \theta(P(y)) \geq 0.$$

The last equality holds because every supported e divides $P(y)$; the last inequality is the nonnegativity of θ at the positive integer $P(y)$. The extension is consequently nonnegative everywhere. The moving intervals at the other end may extend beyond N , but $W_{\leq D}$ is defined and nonnegative at those positive integers as well. Thus the containment above gives

$$\sum_{x-3T < n \leq x} W_{\leq D}(n) \geq W_r^{\leq D} \quad (x \in \mathcal{J}_r).$$

We need the mean $3T\nu_D$ to be at most $C_1T/(4L)$, leaving a positive deviation when $r \in \mathcal{E}$. This is why we chose the fixed constant $C_1 = 200000s$. The bound $2^L \leq 4N$ gives

$$L \leq \frac{\log N + \log 4}{\log 2} \leq 2 \log N$$

for sufficiently large N . The calculation in (2.5), which accounts for both floors in $Z = \lfloor N^{1/1000} \rfloor$ and $y = \lfloor Z^{1/s} \rfloor$, gives

$$\log y \geq \frac{\log N}{2000s}.$$

Together with $\nu_D \leq 4/\log y$, these inequalities yield

$$12L\nu_D \leq \frac{48L}{\log y} \leq 192000s < C_1, \quad \text{so} \quad 3T\nu_D \leq \frac{C_1T}{4L}.$$

Define the deviation

$$G(x) = \sum_{x-3T < n \leq x} W_{\leq D}(n) - 3T\nu_D.$$

If $r \in \mathcal{E}$, then for every $x \in \mathcal{J}_r$,

$$G(x) > \frac{C_1T}{2L} - \frac{C_1T}{4L} = \frac{C_1T}{4L}.$$

Applying the mean-square estimate. Each endpoint interval $\mathcal{J}_r$ has length T . The intervals have disjoint interiors and lie in $[T, N+T]$; their shared endpoints do not affect the integrals. Hence

$$T \#\mathcal{E} \left(\frac{C_1T}{4L} \right)^2 \leq \sum_{r \in \mathcal{E}} \int_{\mathcal{J}_r} |G(x)|^2 dx \leq \int_T^{N+T} |G(x)|^2 dx.$$

This is the step that turns a mean-square bound into a count of exceptional blocks. Each such block contributes length T times the square of the displayed lower bound.

Apply Lemma 4.1 with

$$f = W_{\leq D}, \quad \bar{f} = \nu_D, \quad Q = D^2 Z, \quad H = 3T, \quad \mathcal{I} = [T, N + T], \quad X = N.$$

We checked the coefficient and support conditions in (4.6); Q is an integer and $H \geq 1$. The lemma bounds the last integral by $O(NT(1 + \log Q)^3 + Q^4)$. Here the active divisor bound is $Q = D^2 Z$, with $D = L^{12}$ and $Z \leq N^{1/1000}$. Recall also that $N \leq 2^L$, so $\log N \leq L \log 2$. Hence

$$\begin{aligned} \log Q &= 2 \log D + \log Z \leq 24 \log L + \frac{\log N}{1000} \leq 24 \log L + \frac{L \log 2}{1000} = O(L), \\ Q^4 &= D^8 Z^4. \end{aligned}$$

Divide the integral bound by $(C_1 T / (4L))^2$; C_1 is fixed. The two terms become, up to fixed constants,

$$NTL^3 \frac{L^2}{T^2} = \frac{NL^5}{T}, \quad D^8 Z^4 \frac{L^2}{T^2} = \frac{Z^4 D^8 L^2}{T^2}.$$

Thus

$$T \# \mathcal{E} \ll \frac{NL^5}{T} + \frac{Z^4 D^8 L^2}{T^2}. \quad (4.7)$$

The first term decreases as T grows: a block must accumulate weight of order T/L to be exceptional, while the mean square grows only linearly with the window length. The second term is the incomplete-period error from the lemma. We will compare both with N/L^{12} after adding the tail bound.

4.3 The square-divisor tail

On block r , the omitted part is

$$B_r := W_r - W_r^{\leq D} = \sum_{\substack{n \in I_r \\ n \geq 1}} \sum_{\substack{m > D, \, m \text{ odd} \\ m^2 | n}} \theta(n/m^2) \geq 0.$$

Here a bound on the total weight $\sum_r B_r$ will be enough. Every block with a large tail uses up at least $C_1 T / (2L)$ of that nonnegative total.

The blocks partition $[0, N)$. Write $n = m^2 q$ and interchange the finite sums to get

$$\sum_r B_r = \sum_{\substack{D < m \leq \sqrt{N} \\ m \text{ odd}}} \sum_{1 \leq q < N/m^2} \theta(q) \leq \sum_{D < m \leq \sqrt{N}} \sum_{1 \leq q \leq N/m^2} \theta(q).$$

For the inequality we included even m and allowed equality at the upper endpoint. Both changes add only nonnegative terms. Different pairs (m, q) can give the same integer n ; this is appropriate, since $W(n)$ also includes every eligible square divisor separately.

Recall that (3.3) gives $\sum_{1 \leq q \leq x} \theta(q) = \nu x + O(Z)$ uniformly in x , with $0 \leq \nu \ll 1/L$. For every retained m , we have $x = N/m^2 \geq 1$, so

$$\sum_{1 \leq q \leq N/m^2} \theta(q) \ll \frac{N}{m^2 L} + Z.$$

There are at most $\sqrt{N}$ values of m , and $\sum_{m>D} m^{-2} \leq \int_D^\infty t^{-2} dt = 1/D$. Summing the bound yields

$$\sum_r B_r \ll \frac{N}{DL} + Z\sqrt{N} \ll \frac{N}{DL}. \quad (4.8)$$

The second inequality uses $Z \leq N^{1/1000}$, $D = L^{12}$, and $L \asymp \log N$:

$$\frac{Z\sqrt{N}}{N/(DL)} \leq N^{-499/1000} L^{13} \longrightarrow 0.$$

Thus the error Z in each individual sum remains affordable even after summing over all m .

Let $\mathcal{F} = \{r : B_r > C_1 T/(2L)\}$. Nonnegativity gives

$$\frac{C_1 T}{2L} \#\mathcal{F} \leq \sum_{r \in \mathcal{F}} B_r \leq \sum_r B_r \ll \frac{N}{DL}.$$

This is Markov's inequality in its elementary counting form: divide the available total by the amount required for each member of the exceptional set. We obtain

$$\#\mathcal{F} \ll \frac{N}{DT}, \quad T\#\mathcal{F} \ll \frac{N}{D}. \quad (4.9)$$

The two estimates now give the promised bound for the full majorant.

Proposition 4.2. *With $C_1 = 200000s$, we have*

$$T \#\{r : W_r > C_1 T/L\} \ll N/L^{12}.$$

Proof. Recall that $\mathcal{E} = \{r : W_r^{\leq D} > C_1 T/(2L)\}$ is the exceptional set for the truncated part. For every block, $W_r = W_r^{\leq D} + B_r$. If neither term exceeds $C_1 T/(2L)$, their sum does not exceed $C_1 T/L$. Hence

$$\{r : W_r > C_1 T/L\} \subseteq \mathcal{E} \cup \mathcal{F}.$$

The total length of this union is at most $T\#\mathcal{E} + T\#\mathcal{F}$. Substituting (4.7) and (4.9) gives

$$T \#\{r : W_r > C_1 T/L\} \ll \frac{NL^5}{T} + \frac{Z^4 D^8 L^2}{T^2} + \frac{N}{D}. \quad (4.10)$$

Use $D = L^{12}$, $Z \leq N^{1/1000}$, and $T \geq L^{18}$ in each term:

$$\begin{aligned} \frac{NL^5}{T} &\leq \frac{N}{L^{13}}, \\ \frac{Z^4 D^8 L^2}{T^2} &\leq N^{4/1000} L^{8 \cdot 12 + 2 - 2 \cdot 18} = N^{1/250} L^{62}, \\ \frac{N}{D} &= \frac{N}{L^{12}}. \end{aligned}$$

The middle term contains a positive power of N , but its power is strictly below one. Relative to the required bound, it has size

$$\frac{N^{1/250} L^{62}}{N/L^{12}} = N^{-249/250} L^{74} \longrightarrow 0,$$

because a fixed power of $\log N$ grows more slowly than any positive power of N . The sum of the three terms is consequently $O(N/L^{12})$. $\square$

5 Computing block bounds by convolution

The previous section bounds the number of blocks with large majorant sums. To decide which candidates to keep, we must now compute upper bounds for those sums. Recall their product-counting form:

$$W_r = \sum_{(d,\sigma) \in \mathcal{S}} \sigma \# \{k \geq 1 : dk \in I_r\}.$$

Evaluating every divisor entry in every block would require at least $\Omega(N^{3/2}/T)$ operations: the entries with $e = 1$ alone include $\Omega(\sqrt{N})$ odd squares, and there are N/T blocks. Listing the products is also too costly. The entry $d = 1$ alone produces $N - 1$ products below N . Of these, $\Omega(N)$ are at least $N/2$ and each needs $\Omega(L)$ binary digits. Writing the list therefore takes $\Omega(NL)$ bits. We need to count products in groups without locating each one separately.

There is first a restriction to remove. For a fixed divisor d , the products below N use only multipliers $k < N/d$. This range depends on d , so it does not give a common collection of multipliers that can be combined with every divisor. We instead form all pairs

$$(d, \sigma) \in \mathcal{S}, \quad 1 \leq k \leq N.$$

Every required pair is included, since $dk < N$ and $d \geq 1$ imply $k < N$. Divisors and multipliers can now be grouped independently. The extra products, which can be as large as N^2 , will be discarded according to their locations. Only uncertainty near the cutoff N can make these extra pairs affect a block bound.

To choose the groups, first suppose that exact logarithms were available. For an integer $K \geq 1$, assign a factor u to the bin with index $\lfloor K \log_2 u \rfloor$. Bin j is then the interval

$$[2^{j/K}, 2^{(j+1)/K}).$$

If d lies in bin i and k in bin j , then

$$2^{(i+j)/K} \leq dk < 2^{(i+j+2)/K}.$$

Thus the sum $i + j$ determines a short interval containing the product. Let A_i count the divisors in bin i , and B_j the multipliers in bin j , for the moment ignoring signs. Since the factors can be chosen independently, there are exactly $A_i B_j$ pairs from these two bins. Adding over the pairs of bins with $i + j = \ell$ gives

$$C_\ell = \sum_{i+j=\ell} A_i B_j.$$

This is the convolution of the two histograms. Each pair has one index sum, although the intervals belonging to neighboring sums may overlap. Thus C_ℓ counts the pairs assigned to its interval; it does not count every product that happens to lie there. We will compute the entire convolution by one integer multiplication.

The bin width explains the choice of K . If $x = 2^{j/K}$, then the next threshold is $x2^{1/K}$, at distance

$$x(2^{1/K} - 1) = O(x/K).$$

The product intervals above have the same scale of width, since $2^{2/K} - 1 = O(1/K)$. Near the upper bound N , we therefore expect an uncertainty of order N/K . We choose a permissible uncertainty $1 \leq \eta \leq N$ and take $K = \lceil N/\eta \rceil$. There are $O(L)$ units of binary logarithm between 1 and N , with K bins per unit, so the number of thresholds is $O(KL)$. Counting the integers between consecutive thresholds avoids computing N separate logarithms.

These scales also explain the computational constraint on η . Appendix B supplies each numerical approximation in $O(L^3)$ bit operations. With $O(KL) = O(NL/\eta)$ thresholds, the numerical work is

$$O((NL/\eta)L^3) = O(NL^4/\eta).$$

The choice $\eta = L^4$ makes this work linear in N . Increasing η would reduce the number of thresholds, but would also widen the product intervals and leave more uncertainty about which blocks contain their products. This is why the computation must be analyzed together with its excess error.

The algorithm cannot assume exact logarithms or exact real thresholds as primitive operations. The construction below will instead assign each factor to a nearby bin and prove a bounded error in its logarithmic index. The assigned bins need not agree with the ideal bins just described. What matters is that every factor is assigned once and that every resulting product lies in the interval assigned to its pair of bins. Positive and negative divisor entries will be counted separately so that a product's unknown location can only increase the computed upper bounds.

We state the construction for a general signed divisor list. This separates the computation from the particular sieve weights that produced our list.

Let $\mathcal{S}$ be a list of S pairs (d, σ) with $1 \leq d \leq N$ and $\sigma \in \{-1, 1\}$; repeated entries are allowed. Write

$$F_r = \sum_{(d, \sigma) \in \mathcal{S}} \sigma \# \{k \geq 1 : dk \in I_r\}, \quad V = \sum_{(d, \sigma) \in \mathcal{S}} \frac{1}{d}.$$

Here each occurrence of a divisor in the list is counted separately, even if another occurrence has the same value or the opposite sign. For the list in (3.5), $F_r = W_r$. For a general signed list, F_r need not be nonnegative; the construction will still give an upper bound. For positive integers N, T, η , set

$$K = \lceil N/\eta \rceil, \quad J = K(L + 1),$$

where $N \leq 2^L \leq 4N$ as in Section 2. Each histogram of individual factors will use the indices $0, \dots, J - 1$; the extra K bins beyond KL leave room for rounding. Adding two indices gives $2J - 1$ possible product bins. We will prove that every product enclosure meeting $[0, N)$ has width at most 64η . The numerical constant allows both the binning errors and outward rounding of the endpoints.

Proposition 5.1 (Computed block bounds). *Assume $T \mid N$, $S \leq N^2$, $(d, T) \leq 2$ for every entry, and*

$$16 \leq \eta, \quad 1024\eta \leq T \leq N.$$

A deterministic multitape algorithm computes integers $U_r \geq F_r$ for all $0 \leq r < N/T$, with

$$\sum_{0 \leq r < N/T} (U_r - F_r) \ll \frac{N\eta}{T} V + \eta S. \quad (5.1)$$

The implied constant is absolute. The running time is

$$O((J + S)L^3 + \mathbf{M}(O(JL))), \quad (5.2)$$

where $\mathbf{M}(b)$ is a deterministic bound for multiplying b -bit integers. The Schönhage–Strassen bound $O(b \log b \log \log b)$ suffices for our parameters [10]. Harvey and van der Hoeven proved $\mathbf{M}(b) = O(b \log b)$ in the same multitape model [6].

The two terms in (5.1) have different sources. For one divisor d , there are about N/d relevant multiples. The residue count below bounds the proportion near a block boundary

by $O(\eta/T)$, giving $O(N\eta/(dT))$ after a complete number of residue periods. Summing this contribution produces $(N\eta/T)V$. There is also one possibly incomplete period for each list entry; its contribution is $O(\eta)$, producing the term ηS . The hypothesis $(d, T) \leq 2$ prevents the multiples of a divisor from being concentrated on the block boundaries.

We first construct the histograms and upper bounds, then prove this residue estimate, and finally account for the bit operations. Throughout, we count pairs (d, k) with their list multiplicities: different pairs can have the same product.

5.1 Divisor and multiplier histograms

We need three histograms: one for positive divisor entries, one for negative divisor entries, and one for the multipliers $1, \dots, N$. Every factor must be assigned once. The divisor index will differ from $K \log_2 d$ by at most 1, and the multiplier index will differ from $K \log_2 k$ by at most 2. Adding these two errors will later give a product enclosure. The two kinds of factors need not use the same exact bin edges: the bound on the sum of their index errors is what will control their product.

The divisor list already contains one record per entry, so we allow one approximate logarithm for each of its S records. There are N multipliers, however, and one numerical evaluation for each would be too costly. For most multipliers we will instead compute approximate integer thresholds and count all the integers between successive thresholds by subtraction.

The thresholds need not reproduce the ideal logarithmic partition. For example, replacing a threshold at $a + 1/5$ by the integer a changes the bin containing the integer a . It does not lose that integer or count it twice, provided neighboring bins use the same threshold. We let the computed thresholds define the partition and prove a location guarantee for that partition. This avoids having to decide on which side of every exact real threshold an integer lies.

A bounded error in a threshold does not always give a bounded error in its logarithmic index. By integrating the derivative of $\log_2 x$, for $x > 1$ and $|\Delta| \leq 1$ we have

$$K|\log_2(x + \Delta) - \log_2 x| \leq \frac{K|\Delta|}{(x - 1) \log 2}.$$

Thus an additive error of one can have a large effect when x is small compared with K . If $x > 3K$ and $K \geq 1024$, then $\log 2 \geq 1/2$ gives

$$\frac{K}{(x - 1) \log 2} < \frac{2K}{3K - 1} < 1.$$

We will assign the first $4K$ multipliers individually and use approximate thresholds only above that cutoff. The margin between $3K$ and $4K$ will ensure that the threshold immediately below each remaining multiplier still exceeds $3K$. Handling $O(K)$ multipliers individually fits within the $O(KL)$ numerical evaluations already allowed for thresholds.

The hypotheses of Proposition 5.1 imply $N \geq 1024\eta \geq 16384$, $1024 \leq K \leq N$, and

$$4K \leq 4(N/\eta + 1) \leq N/4 + 4 \leq N.$$

We use the certified routines of Lemma B.1 for real approximations. Its hypotheses $N \geq 3$, $1 \leq K \leq N$, and $N \leq 2^L \leq 4N$ are thus satisfied.

For each divisor entry, approximate $K \log_2 d$ within $1/4$ and round to the nearest integer, breaking ties downward. Call the resulting index $i(d)$. The logarithm routine applies since

$1 \leq d \leq N$. The approximation error is at most $1/4$, and rounding adds at most $1/2$. In particular,

$$|i(d) - K \log_2 d| \leq 1. \quad (5.3)$$

Let A_i^+ and A_i^- count the positive and negative entries with index i , respectively.

Construct the multiplier histogram B_j as follows.

- (i) For $1 \leq k \leq 4K$, assign an index by the same approximation and rounding rule as for divisors. Since $4K \leq N$, these logarithm inputs also satisfy Lemma B.1. This uses $O(K)$ evaluations.
- (ii) For each $0 \leq j \leq J$, compute an integer $\tilde{\xi}_j$ within one of $2^{j/K}$, by approximating within $1/4$ and rounding to the nearest integer, again breaking ties downward. These exponential inputs satisfy $|j| \leq J = K(L+1) \leq 2K(L+1) + 8$, as required by Lemma B.1. Set

$$\xi_0 = \tilde{\xi}_0, \quad \xi_j = \max(\xi_{j-1}, \tilde{\xi}_j) \quad (1 \leq j \leq J).$$

This is a single forward scan, keeping only the previous maximum. It gives $\xi_j = \max_{0 \leq h \leq j} \tilde{\xi}_h$, so the thresholds are non-decreasing. They still approximate $2^{j/K}$ within one: the current term gives $\xi_j \geq \tilde{\xi}_j \geq 2^{j/K} - 1$, while every term in the maximum satisfies $\tilde{\xi}_h \leq 2^{h/K} + 1 \leq 2^{j/K} + 1$.

- (iii) Replace thresholds outside the remaining multiplier range by the nearer endpoint of that range. In formulas, set

$$\xi'_j = \min(N+1, \max(4K+1, \xi_j)).$$

For $0 \leq j < J$, assign the $\xi'_{j+1} - \xi'_j$ integers in $[\xi'_j, \xi'_{j+1})$ to bin j . In the forward scan, retain ξ'_j until the next threshold is computed, then emit this difference as the bin's count.

The last intervals partition the integers in $[4K+1, N+1)$. Indeed, the modified thresholds remain non-decreasing, the initial one is $4K+1$, and the final one is $N+1$. The endpoint claims follow from $\xi_0 \leq 2$ and $2^{J/K} = 2^{L+1} \geq 2N$. Repeated thresholds simply give empty bins. The difference $\xi'_{j+1} - \xi'_j$ counts a whole bin at once; we do not visit its integers individually.

It remains to prove that this partition gives the promised logarithmic indices. Put $\zeta_j = 2^{j/K}$ for the exact thresholds, and consider a multiplier k assigned to bin j by the threshold rule. We have $\xi'_j \leq k < \xi'_{j+1}$ and $4K+1 \leq k \leq N$. Replacing an endpoint by $4K+1$ or $N+1$ cannot place such a k on the other side of the original endpoint. More explicitly, $k \geq \xi'_j$ implies $k \geq \xi_j$: clipping at $N+1$ would make this bin empty, while clipping at $4K+1$ only increases its left endpoint. For the right endpoint, clipping down to $N+1$ can only strengthen $k < \xi_{j+1}$, while clipping up to $4K+1$ would leave no eligible $k < \xi'_{j+1}$. Thus $k < \xi'_{j+1}$ implies $k < \xi_{j+1}$. Consequently,

$$\zeta_j - 1 \leq k < \zeta_{j+1} + 1.$$

The upper inequality gives $k - 1 < \zeta_{j+1} = \zeta_j 2^{1/K}$. Since $k \geq 4K+1$,

$$\zeta_j > (k-1)2^{-1/K} \geq 4K 2^{-1/K} > 3K.$$

For the last inequality, $K \geq 1024$ implies $2^{1/K} \leq 2^{1/4} < 4/3$. Thus both ζ_j and ζ_{j+1} exceed $3K$, where the logarithmic sensitivity bound above applies. The lower endpoint gives

$$K \log_2 k \geq K \log_2(\zeta_j - 1) > K \log_2 \zeta_j - 1 = j - 1.$$

The upper endpoint gives

$$K \log_2 k < K \log_2(\zeta_{j+1} + 1) < K \log_2 \zeta_{j+1} + 1 = j + 2.$$

In particular, every multiplier assigned by the threshold rule satisfies

$$|j - K \log_2 k| \leq 2. \quad (5.4)$$

The individually assigned multipliers satisfy the stronger error bound (5.3). The direct assignments cover $1, \dots, 4K$, and the threshold intervals partition $4K + 1, \dots, N$. Adding these two collections of counts therefore gives a histogram containing every multiplier exactly once, with (5.4) valid for each assignment.

All factor indices lie in $[0, J)$. For an individual assignment, $0 \leq K \log_2 u \leq KL$ and the numerical approximation is within $1/4$. Rounding to the nearest integer cannot give a negative index and gives an index at most $KL + 1 < J$. The threshold rule uses only $0 \leq j < J$ by construction.

5.2 Product histograms and enclosures

The factor histograms are now available. We use convolution to count pairs with each sum of bin indices, then turn that approximate logarithmic location into an integer interval containing every product counted in the bin. These enclosures will determine the possible destination blocks.

Compute the ordinary integer convolutions

$$C^+ = A^+ * B, \quad C^- = A^- * B. \quad (5.5)$$

For example, the coefficient with index ℓ is

$$C_\ell^+ = \sum_{\substack{0 \leq i, j < J \\ i+j=\ell}} A_i^+ B_j.$$

The product in each summand counts all ways to choose one positive divisor entry in bin i and one multiplier in bin j . Thus C_ℓ^+ is the number of pairs (d, k) , with $(d, +1)$ in the list and $1 \leq k \leq N$, whose bin indices sum to ℓ . The count C_ℓ^- is the same with $(d, -1)$. Each pair occurs exactly once in its corresponding convolution. Since both factor histograms have indices in $[0, J)$, the product indices satisfy $0 \leq \ell \leq 2J - 2$.

For a pair assigned to factor bins i and j , put $\ell = i + j$. Equations (5.3) and (5.4) give

$$|\ell - K \log_2(dk)| \leq |i - K \log_2 d| + |j - K \log_2 k| \leq 1 + 2 = 3.$$

In particular, every such product lies between $2^{(\ell-3)/K}$ and $2^{(\ell+3)/K}$. We allow an extra unit in the index, then round the resulting interval outwards. This gives integer endpoints x_ℓ^-, x_ℓ^+ satisfying

$$\begin{aligned} 2^{(\ell-4)/K} - 2 &\leq x_\ell^- \leq 2^{(\ell-4)/K}, \\ 2^{(\ell+4)/K} &\leq x_\ell^+ \leq 2^{(\ell+4)/K} + 2. \end{aligned} \quad (5.6)$$

The exponential inputs $\ell - 4$ and $\ell + 4$ have absolute value at most $2J + 2 \leq 2K(L + 1) + 8$, so Lemma B.1 applies to both endpoints. They can be computed by outward rounding. Specifically, if $\hat{x}_\ell^-$ and $\hat{x}_\ell^+$ approximate $2^{(\ell-4)/K}$ and $2^{(\ell+4)/K}$ within $1/4$, set

$$x_\ell^- = \lfloor \hat{x}_\ell^- - \frac{1}{4} \rfloor, \quad x_\ell^+ = \lceil \hat{x}_\ell^+ + \frac{1}{4} \rceil.$$

We now convert the multiplicative enclosure into a bound on its ordinary width. This bound is needed only when the enclosure meets $[0, N)$: all other bins can be discarded. Put

$$a_\ell = 2^{(\ell-4)/K}.$$

If the enclosure meets $[0, N)$, then $x_\ell^- < N$. The lower-endpoint bounds in (5.6) give

$$a_\ell \leq x_\ell^- + 2 < N + 2.$$

The upper exponential threshold is $a_\ell 2^{8/K}$. Hence

$$\begin{aligned} x_\ell^+ - x_\ell^- &\leq (a_\ell 2^{8/K} + 2) - (a_\ell - 2) \\ &= a_\ell(2^{8/K} - 1) + 4 \\ &\leq (N + 2)(2^{8/K} - 1) + 4. \end{aligned}$$

For $K \geq 1024$, setting $t = 8 \log 2/K$ gives $e^t \leq 2$ and $e^t - 1 \leq te^t < 16/K$. Thus

$$x_\ell^+ - x_\ell^- \leq 16N/K + 32/K + 4 \leq 64\eta < T, \quad (5.7)$$

using $N/K \leq \eta$, $K \geq 1024$, $\eta \geq 16$, and $T \geq 1024\eta$. Each relevant enclosure therefore meets at most two blocks: meeting three blocks would require reaching across the whole middle block of length T . The restriction to enclosures meeting $[0, N)$ matters. A product near N^2 can have a much wider enclosure, but such a bin is irrelevant unless its lower endpoint reaches back into the output range.

5.3 Assigning signed counts to blocks

We now use the enclosures to construct U_r and prove $U_r \geq F_r$ for every block. The issue is how to treat a bin whose enclosure meets two blocks: its products' exact locations are unknown, and positive and negative counts require different rules.

If an enclosure lies wholly inside one block, we know where every product in that bin belongs, so we use its signed count exactly. If the enclosure crosses a boundary, a positive pair may belong to either neighboring block. Adding its count to every block that the enclosure meets is safe. A negative pair needs the opposite treatment: subtracting its count from a block would be safe only if we knew that the product belonged there. We therefore subtract nothing for an uncertain negative pair.

Consider an enclosure crossing an interior boundary, with seven positive pairs and three negative pairs. The following table shows one possible split of the products. The algorithm does not know this split; it knows only the two totals.

	Left block	Right block	Total
Positive pairs	5	2	7
Negative pairs	2	1	3
True signed contribution	3	1	4
Contribution to the upper bounds	7	7	14
Excess	4	6	10

The bounds remain safe for every possible split. For instance, all seven positive pairs could lie on the left and all three negative pairs on the right; subtracting three from the left bound would then make it too small. Although the individual excesses depend on the split, their sum is always $14 - (7 - 3) = 10$. There is also a pairwise way to see this total. Each of the

seven positive pairs is counted in two blocks instead of its one actual block, adding one unit of excess. Each of the three negative pairs is omitted from its actual block, also adding one unit. We will bound the total excess by counting pairs whose enclosures cross a boundary.

At the exterior boundary N , only the last block is retained. The product itself may lie on either side of N , so there are four possibilities for a pair in a bin crossing that boundary:

Sign	Product location	True contribution	Assigned contribution	Excess
+1	below N	1	1	0
+1	at least N	0	1	1
-1	below N	-1	0	1
-1	at least N	0	0	0

Here the assigned contribution follows the same rule: retain the positive count and omit the negative count. This is how the extra products introduced at the start of the section can affect a bound. A bin wholly beyond N contributes nothing. At the left endpoint 0, the rule is the same, except that $dk \geq 1$ for every pair; an enclosure extending left of 0 can only affect the first block.

Formally, for $0 \leq r < N/T$, set

$$U_r = \sum_{\ell: [x_\ell^-, x_\ell^+] \cap I_r \neq \emptyset} C_\ell^+ - \sum_{\ell: [x_\ell^-, x_\ell^+] \cap \mathbb{Z} \subseteq I_r} C_\ell^-. \quad (5.8)$$

The first condition is equivalent to $x_\ell^+ \geq rT$ and $x_\ell^- \leq (r+1)T - 1$; the second is equivalent to $x_\ell^- \geq rT$ and $x_\ell^+ \leq (r+1)T - 1$.

Consider one positive pair (d, k) . If $dk \in I_r$, its enclosure meets I_r , so it is counted in U_r . Any additional inclusion increases the bound. A negative pair is subtracted only when its whole enclosure lies in I_r , which implies $dk \in I_r$. Consequently, the contribution of each pair to U_r is at least its contribution to F_r . Summing over pairs proves

$$U_r \geq F_r. \quad (5.9)$$

5.4 Charging the excess to boundaries

The preceding subsection established the upper-bound property. We next identify which pairs can make the inequality strict and bound how much each pair can add. This reduces the total excess to a count of products near block boundaries, including 0 and N .

Lemma 5.2 (Excess comes from products near boundaries). *Every pair with a strictly positive contribution to $\sum_r (U_r - F_r)$ has the following properties.*

- (i) *Its enclosure crosses a boundary in $T\mathbb{Z}$, possibly 0 or N .*
- (ii) *Its product lies within 64η of that boundary and is at most $2N$.*
- (iii) *Its total contribution to the excess is at most two.*

Consequently,

$$\sum_r (U_r - F_r) \leq 2 \sum_{(d, \sigma) \in \mathcal{S}} \#\{k \geq 1 : dk \leq 2N, \text{dist}(dk, T\mathbb{Z}) \leq 64\eta\}. \quad (5.10)$$

Here $\text{dist}(x, T\mathbb{Z})$ is the distance from x to the nearest integer multiple of T .

Proof. If an enclosure lies wholly within one block, both signs contribute exactly. If it meets none of the blocks, its product lies outside the range $[0, N)$ and contributes to neither side. Thus an excess requires an enclosure that meets the range $[0, N)$ and crosses a boundary of a block. At the ends, this includes an enclosure reaching to the left of 0 or to the right of $N - 1$, proving (i).

Recall from (5.7) that an enclosure meeting $[0, N)$ has width at most $64\eta < T$. Also, $1024\eta \leq T \leq N$ gives $64\eta \leq N/16$. Let the crossed boundary be $b \in T\mathbb{Z}$. Since the endpoints are integers, $x_\ell^- \leq b - 1$ and $x_\ell^+ \geq b$. Both b and the product lie in $[x_\ell^-, x_\ell^+]$, so their distance is at most $x_\ell^+ - x_\ell^- \leq 64\eta$. The enclosure also meets $[0, N)$; hence its products are less than $N + 64\eta < 2N$. This proves (ii), including products at or beyond N .

For a positive pair, let t be the number of retained blocks met by its enclosure, so $t \leq 2$. Its assigned total is t , while its true total is 1 if $dk < N$ and 0 otherwise. If $dk < N$, its actual block is among those t blocks. Thus its excess is nonnegative and at most two. For a negative pair, an enclosure contained in a retained block gives the exact contribution -1 . Otherwise the assigned contribution is zero; the excess is one if the actual product is in the range and zero if it is outside. This proves (iii) for both signs.

Summing over the pairs with positive excess gives (5.10). We have enlarged the set being counted on its right side: it includes all $k \geq 1$ satisfying $dk \leq 2N$, even when $k > N$. Every pair with positive excess belongs to this larger set by (ii), and all terms in the count are nonnegative. $\square$

5.5 Counting products near a boundary

Lemma 5.2 reduced the excess to the count in (5.10). We now bound that count for each divisor d by following dk through its residues modulo T , then sum over the list to prove (5.1).

Fix one divisor entry d . There are

$$M_d = \left\lfloor \frac{2N}{d} \right\rfloor$$

possible multipliers in (5.10). To count the favorable ones, we need the period of the residue sequence $d, 2d, 3d, \dots$ modulo T . Put $g = (d, T) \leq 2$ and $q_d = T/g$. Since $(d/g, q_d) = 1$, multiplication by d/g permutes the residue classes modulo q_d . Indeed, if it sends two classes to the same class, then q_d divides $(d/g)(k_1 - k_2)$, and coprimality implies that q_d divides $k_1 - k_2$. Thus the map is one-to-one and hence onto the finite set of classes. Multiplying back by g , it follows that $dk \bmod T$ visits each multiple of g exactly once in every q_d consecutive values of k .

The favorable residues are zero, the multiples of g between 1 and 64η , and their negatives modulo T . The positive and negative ranges do not overlap, since $64\eta < T/2$. Their number is therefore

$$h_d = 2 \left\lfloor \frac{64\eta}{g} \right\rfloor + 1.$$

Write $M_d = a_d q_d + r_d$, with $0 \leq r_d < q_d$. Each of the a_d complete periods contains exactly h_d favorable residues. The final r_d positions contain no repeated residue, so they contain at most h_d favorable ones. Hence the count for this divisor is at most

$$(a_d + 1)h_d \leq \left(\frac{2Ng}{dT} + 1 \right) \left(\frac{128\eta}{g} + 1 \right) = \frac{256N\eta}{dT} + \frac{2Ng}{dT} + \frac{128\eta}{g} + 1.$$

Here $a_d \leq M_d/q_d \leq 2Ng/(dT)$. The first two terms come from complete periods. Since $g \leq 2$ and $\eta \geq 16$, their sum is $O(N\eta/(dT))$. The last two terms account for the incomplete period

and are $O(\eta)$ because $g \geq 1$. We have obtained

$$\#\{k \geq 1 : dk \leq 2N, \text{dist}(dk, T\mathbb{Z}) \leq 64\eta\} \ll \frac{N\eta}{dT} + \eta.$$

Summing this estimate in (5.10) gives

$$\sum_r (U_r - F_r) \ll \frac{N\eta}{T} \sum_{(d,\sigma) \in \mathcal{S}} \frac{1}{d} + \eta \sum_{(d,\sigma) \in \mathcal{S}} 1 = \frac{N\eta}{T} V + \eta S,$$

which is (5.1).

The bound on g is essential in the complete-period terms. If g were as large as T , the term $2Ng/(dT)$ could be $2N/d$. This happens when T divides d : every product dk then lies exactly on a block boundary, and there is no small exceptional proportion to exploit.

5.6 Bit complexity

We have proved both $U_r \geq F_r$ and the total excess bound. It remains to implement the construction within (5.2): we account for sorting the factor records, performing the two convolutions, collecting the block contributions, and evaluating the logarithms and exponential thresholds.

A histogram is initially stored as a list of records, rather than an array to which we make distant tape writes. Each divisor entry produces a record containing its bin index, sign, and count 1. Each of the first $4K$ multipliers produces its bin index and count 1. Sorting the divisor records by index and sign brings equal entries together; a forward scan adds their counts and produces A^+ and A^- . Sorting the direct multiplier records by index and adding equal indices produces their contribution to B .

The threshold counts for the remaining multipliers are already produced in increasing index order. Merge this stream with the sorted direct multiplier counts: copy the smaller-index record, or add the two counts when the indices agree. This produces B without visiting any of the integers represented by a threshold count. During threshold construction, each new value uses only the previous maximum and previous clipped threshold, so that construction also requires only forward scans apart from its numerical subroutines.

Factor counts are at most S or N , and every convolution coefficient counts at most $SN \leq N^3$ pairs. They therefore occupy $O(L)$ bits. The same holds for indices and endpoints: indices are at most $2J$, while the largest exponent used for an endpoint is at most $2L + 2 + 2/K$. Even endpoints above N thus have $O(L)$ binary digits. There are $O(S + J)$ records, with $S \leq N^2$ and $J = K(L + 1) = O(NL)$, so $\log(S + J) = O(L)$. Comparing two $O(L)$ -bit records costs $O(L)$. The sorting bound recalled in Section 2, with $n = O(S + J)$ and $b = O(L)$, therefore gives $O((S + J)L \log(S + J)) = O((S + J)L^2)$ for these operations.

It remains to explain how integer multiplication computes a whole convolution. Each coefficient counts at most $SN \leq N^3$ pairs. Choose the base $b = 2^{4L+4}$ and form the integers

$$P^\pm = \sum_{i=0}^{J-1} A_i^\pm b^i, \quad Q = \sum_{j=0}^{J-1} B_j b^j.$$

The input coefficients fit in base b , since $A_i^\pm \leq S \leq N^2$ and $B_j \leq N$. The distributive law gives the exact identity

$$P^\pm Q = \sum_{\ell=0}^{2J-2} \left(\sum_{\substack{0 \leq i, j < J \\ i+j=\ell}} A_i^\pm B_j \right) b^\ell = \sum_{\ell=0}^{2J-2} C_\ell^\pm b^\ell.$$

Every coefficient is smaller than b , since $SN \leq N^3 \leq 2^{3L} < 2^{4L+4}$. Therefore this expression is already the base- b expansion of the product: no carry passes from one coefficient to the next. Each digit occupies exactly $4L + 4$ binary positions, so a sequential scan reads off the coefficients.

To build these packed integers from the sorted records, scan the bin indices $0, 1, \dots, J - 1$ in order alongside each record list. At an index present in the list, write its count in $4L + 4$ binary positions, least significant bit first and padded with zeros. At a missing index, write $4L + 4$ zero bits. This emits the base- b digits from least to most significant, using $O(L)$ work per index, including the index counter. A multiplication routine requiring the opposite order can be supplied by reading the completed tape from its right end to its left end and copying each bit to a fresh tape from left to right. Moving to the right end first, if necessary, and this copying each take linear time. Packing therefore costs $O(JL)$ and gives integers of $O(JL)$ bits. If multiplication returns the most significant bit first, reverse its output by the same procedure before unpacking. Then pad the high-order end with zeros to exactly $(2J - 1)(4L + 4)$ bits. Multiplication may omit leading zeros or return only one zero bit for a zero product; this padding restores all $2J - 1$ coefficient positions in either case. Reading successive chunks of $4L + 4$ bits then gives the coefficients in increasing index order and costs $O(JL)$, including padding. Keeping positive and negative counts separate is what makes all these digits nonnegative.

This method is called Kronecker substitution [4, Corollary 8.27]. Farach-Colton and Tsai use it for prime enumeration [2, Section 3.2]; see also [5, Section 1.6]. The two multiplications consequently cost $O(M(O(JL)))$ bit operations.

For a product enclosure that meets $[0, N)$, its first and last possible block indices are

$$\left\lfloor \frac{\max(x_\ell^-, 0)}{T} \right\rfloor, \quad \left\lfloor \frac{\min(x_\ell^+, N - 1)}{T} \right\rfloor.$$

Since $x_\ell^+ - x_\ell^- \leq 64\eta < T$ for every enclosure meeting $[0, N)$, these indices are equal or consecutive. A constant number of divisions and comparisons of $O(L)$ -bit integers finds them in $O(L^2)$ time per bin. We then write the positive contribution for each possible block r . We write the negative contribution only if

$$x_\ell^- \geq rT \quad \text{and} \quad x_\ell^+ \leq (r + 1)T - 1,$$

which is the containment test in (5.8) applied to the original endpoints. Equal clipped block indices alone do not suffice: clipping can hide an enclosure's extension beyond 0 or N . There are only $O(1)$ records per product bin, because an enclosure meets at most two blocks. Sort these $O(J)$ records by block index, then scan each equal-index run and add its signed contributions. The sort costs $O(JL \log J) = O(JL^2)$, and the scan costs $O(JL)$. Thus we obtain the block bounds without repeatedly moving to distant positions on a tape.

To write every U_r in block order, scan the indices $0, \dots, N/T - 1$ alongside these sums. Output a sum when its index is reached and zero when there is no record for that block. The counter and output each use $O(L)$ bits, so this costs $O((N/T)L)$. It is absorbed in the preceding bounds because $N/T \leq K/1024 \leq J$. Every intermediate sum has $O(L)$ bits: there are at most SN pairs in total, each positive pair contributes to at most two blocks, and each negative pair is subtracted from at most one. In particular, even before cancellation, the sum of the absolute values of the emitted contributions is at most $2SN$.

There are $O(J + S)$ logarithm and exponential evaluations: one for each divisor, $O(K)$ for the individually assigned multipliers, and $O(J)$ for thresholds and product enclosures.

Appendix B, Lemma B.1, gives absolute error at most $1/4$ in $O(L^3)$ time per evaluation. Its parameter hypotheses are $N \geq 3$, $1 \leq K \leq N$, and $N \leq 2^L \leq 4N$, all established above. The logarithm inputs are integers $u \in [1, N]$; the exponential inputs are $2^{j/K}$ with $|j| \leq 2K(L+1) + 8$, as checked when the thresholds and enclosures were introduced. Thus the total numerical cost is $O((J+S)L^3)$.

The factor-index error bounds, followed by the outward rounding in (5.6), ensure that every counted product remains in its enclosure. The signed assignment rule therefore still gives $U_r \geq F_r$ with these numerical approximations. Combining the numerical cost with the sorting, packing, output, and multiplication costs gives

$$O((J+S)L^3 + M(O(JL))),$$

which is (5.2) and completes the proof of Proposition 5.1.

6 Proof of the main theorem

We now apply the two main estimates to the same sieve majorant. Proposition 4.2 says that few blocks have a large exact sum W_r . Proposition 5.1 computes upper bounds $U_r \geq W_r$ and bounds the sum of their errors. A block can be marked because its exact sum is large, or because the computed bound overestimates it by too much. We count these two possibilities separately. This will prove that the repair inputs occupy only $O(N/L^{12})$ positions.

6.1 The marked blocks

First we check that the sieve list satisfies the hypotheses of Proposition 5.1. Its entries, defined in (3.5), have $1 \leq d \leq N$ and signs ± 1 . Lemma 3.2 gives

$$S \ll N^{1/2+1/2000} \leq N^2, \quad V \ll L, \quad (d, T) \leq 2$$

for sufficiently large N . The condition $T \mid N$ holds by construction. The remaining size conditions follow from

$$\eta = L^4 \geq 16, \quad \frac{T}{\eta} \geq L^{14} \geq 1024, \quad T < 2L^{18} \leq N.$$

Section 2 established $N \leq 2^L \leq 4N$. Finally, the divisor expansion of W identifies the signed count F_r in the proposition with W_r . Thus the proposition computes an upper bound $U_r \geq W_r$ for every block.

Write $\Delta_r = U_r - W_r$ for the overestimation on block r . Each Δ_r is nonnegative. Substituting the list bounds and $\eta = L^4$, $T \geq L^{18}$ in (5.1) gives

$$\sum_r \Delta_r \ll \frac{N\eta L}{T} + \eta\sqrt{NZ} \ll \frac{N}{L^{13}} + N^{1/2+1/2000}L^4 \ll \frac{N}{L^{13}}. \quad (6.1)$$

The first term is $NL^5/T \leq N/L^{13}$. For the other term, division by the desired bound gives

$$\frac{N^{1/2+1/2000}L^4}{N/L^{13}} = N^{-999/2000}L^{17} \rightarrow 0.$$

The fixed negative power of N dominates the power of L .

We mark a block when $U_r > R$, where $R = \lceil 2C_1T/L \rceil$. Such a block must satisfy at least one of

$$W_r > \frac{C_1T}{L} \quad \text{or} \quad \Delta_r > \frac{C_1T}{L}.$$

Indeed, if both quantities were at most C_1T/L , their sum U_r would be at most $2C_1T/L \leq R$. The first possibility is controlled by Proposition 4.2. For the second, every block being counted contributes more than C_1T/L to the nonnegative sum in (6.1). Consequently

$$\frac{C_1T}{L} \#\{r : \Delta_r > C_1T/L\} \leq \sum_r \Delta_r.$$

Multiplying the number of blocks by their length T , and adding the two bounds, yields

$$\begin{aligned} T \#\{r : U_r > R\} &\leq T \#\{r : W_r > C_1T/L\} + \frac{L}{C_1} \sum_r (U_r - W_r) \\ &\ll N/L^{12}. \end{aligned} \tag{6.2}$$

Thus the computed bounds leave only the allowed number of positions for repair. No estimate for an individual error Δ_r was needed.

There is one additional reason to mark a block. The bound $W(m^2p) \geq 1$ proved above applies to odd square-primes with $p > y$. To cover the remaining cases, we also mark every block containing an odd square-prime with $p \leq y$. Enumerate the odd primes $p \leq y$ and, for each one, the positive odd integers m with $m^2p < N$. The number of pairs is at most

$$\sqrt{N} \sum_{p \leq y} p^{-1/2} \leq \sqrt{N} \sum_{1 \leq j \leq y} j^{-1/2} \ll \sqrt{Ny}. \tag{6.3}$$

For a fixed prime p , the condition $m^2p < N$ gives $m < \sqrt{N/p}$, so there are at most $\sqrt{N/p}$ possible positive m . Counting the even ones as well can only increase this bound. Replacing the primes by all integers at most y gives the displayed sum, and

$$\sum_{1 \leq j \leq y} j^{-1/2} \leq 1 + \int_1^y t^{-1/2} dt = 2\sqrt{y} - 1.$$

Each pair causes at most one block of length T to be marked. Since $y \leq N^{1/(1000s)}$ and $T < 2L^{18}$, the ratio

$$\frac{T\sqrt{Ny}}{N/L^{12}} \ll L^{30} N^{-1/2+1/(2000s)}$$

tends to zero. Hence these additional blocks have total length $o(N/L^{12})$.

For each enumerated pair, write its block index $\lfloor m^2p/T \rfloor$. Sort these indices, merge them with the already ordered indices satisfying $U_r > R$, and remove adjacent repetitions. Let $\mathcal{R}$ contain every positive integer below N in the resulting marked blocks. The blocks are disjoint, so

$$|\mathcal{R}| \ll N/L^{12}. \tag{6.4}$$

Zero is excluded because the repair theorem takes positive inputs.

An unmarked block now contains no odd square-prime with prime part at most y . Every odd square-prime in that block therefore contributes at least one to W_r ; the other terms of the majorant are nonnegative. Also, the block was not marked by its computed bound. We have proved

$$w_r \leq W_r \leq U_r \leq R \quad \text{on every unmarked block.} \tag{6.5}$$

This is exactly the condition under which the core's candidate is correct.

6.2 Correctness

The core requires $T \mid N$ and a positive integer threshold $R \leq T/2$. Divisibility holds by construction, and for sufficiently large N ,

$$\frac{R}{T} \leq \frac{2C_1}{L} + \frac{1}{T} < \frac{1}{2}.$$

Theorem 2.1 therefore applies in Step 5. On every unmarked block, (6.5) gives $w_r \leq R$, so $c^r = a^r$. Thus every candidate that we keep gives exactly the odd square-primes in its block. We make no claim about candidates on marked blocks, and do not use them.

In Step 6, the input list $\mathcal{R}$ consists of distinct integers in $[1, N)$. The distinctness follows from removing repeated block indices and then listing the integers in the disjoint blocks. Theorem 2.2 returns all square-primes in this list, in increasing order. Discarding its even outputs leaves exactly the odd square-primes in the marked blocks, still in increasing order.

Scan the kept candidates in increasing block order, and within each block in increasing position order. This gives an increasing list of the odd square-primes in the unmarked blocks. Merge it with the odd repair output. The two lists come from disjoint sets of blocks, and every block belongs to one of those sets. Their merge is therefore the complete increasing list of odd square-primes below N .

This is the input required by Theorem 2.3, and $N \geq N_0 \geq 3$ also satisfies that theorem's size condition. Step 7 consequently returns the primes below N , including 2. Since $N \geq N_0$, discarding primes at least N_0 gives precisely the primes below the original input bound, in increasing order.

6.3 Running time

We account for all seven steps, including the cost of writing lists. The parameters s and C_1 are fixed throughout. Recall the scales used in the estimates below:

$$\begin{aligned} L &\asymp \log N, & \eta &= L^4, & L^{18} &\leq T < 2L^{18}, \\ Z &\leq N^{1/1000}, & y &\leq N^{1/(1000s)}. \end{aligned}$$

We repeatedly use that a fixed power of $\log N$ is smaller than any fixed positive power of N for sufficiently large N .

Parameters and divisor list (Steps 1 and 2). The integer parameter calculations take $L^{O(1)}$ bit operations, as described in Section 2. Appendix A constructs the sieve coefficients in $O(Z^2 L^4)$ time. This is $o(N)$ because $Z^2 L^4 / N \leq N^{-998/1000} L^4 \rightarrow 0$.

Read the coefficients in increasing order of e . When $\lambda_e \neq 0$, start at $m = 1$, increase m by two, and write $(m^2 e, \lambda_e)$ for as long as $m^2 e \leq N$. Stop this inner loop at its first product above N , then continue to the next coefficient. Every successful iteration writes one entry of $\mathcal{S}$. There are S such iterations and at most one unsuccessful final test per coefficient, so the number of iterations and scan steps is $O(Z + S)$.

The counters and products have $O(L)$ bits. Two elementary multiplications compute $m^2 e$ in $O(L^2)$ time, and comparison, counter updates, and writing the record fit within the same bound. The loop uses a fixed number of working tapes and writes its output sequentially. Its total cost is therefore

$$O((Z + S)L^2) = O(SL^2) = o(N).$$

For the middle equality, $\lambda_1 = 1$ supplies one entry for every odd $m \leq \sqrt{N}$, so $S \gg \sqrt{N} \geq Z$ for large N . For the last equality, Lemma 3.2 gives $SL^2 / N \ll N^{-999/2000} L^2 \rightarrow 0$.

Computed bounds (Step 3). The histogram length is

$$J = \lceil N/\eta \rceil (L + 1) = O(N/L^3), \quad S \ll N^{1/2+1/2000}.$$

Each histogram coefficient is stored in a digit of $O(L)$ bits, and there are J digits. The packed integers in the multiplication step consequently have $b = O(JL) = O(N/L^2)$ bits. Since this length is at most a fixed power of N , we have $\log b = O(L)$ and $\log \log b = O(\log L)$. The Schönhage–Strassen bound therefore gives

$$M(b) = O(b \log b \log \log b) = O\left(\frac{N}{L^2} L \log L\right) = O(N \log L/L).$$

The other term in (5.2) is $(J + S)L^3$: there are $O(J + S)$ numerical evaluations, each costing $O(L^3)$, and the sorting and scanning costs are smaller. Substituting the values of J and S gives

$$O\left(N + N^{1/2+1/2000}L^3 + \frac{N \log L}{L}\right) = O(N). \quad (6.6)$$

Marked blocks and repair inputs (Step 4). Each computed bound has $O(L)$ bits, so a sequential scan comparing all of them with R costs $O((N/T)L) = o(N)$.

For the small-prime exceptions, test each integer from 2 to y by trial division. Testing all potential proper divisors uses at most y^2 divisions in total. Each division has $O(L)$ -bit operands and costs $O(L^2)$, giving $O(y^2 L^2)$ time to find the primes.

For each odd prime found, loop over odd m until $m^2 p \geq N$. For every pair below N , write the block index $\lfloor m^2 p/T \rfloor$. Let E be the number of these pairs; (6.3) gives $E = O(\sqrt{Ny})$. There are E successful iterations and at most y stopping tests. Computing the product, comparing it with N , and dividing by T takes $O(L^2)$ per iteration. This stage costs $O((E + y)L^2)$.

The resulting E block indices have $O(L)$ bits each. Sorting them costs $O(EL \log(2 + E))$ by the sorting bound in Section 2. Since $E = O(\sqrt{Ny}) = O(N)$, $\log(2 + E) = O(L)$, and sorting costs $O(EL^2)$. All three stages therefore fit within

$$O((y^2 + \sqrt{Ny})L^2) = o(N).$$

Indeed, $y^2 \leq N^{1/(500s)}$ and $\sqrt{Ny} \leq N^{1/2+1/(2000s)}$. Both exponents are strictly below one, so multiplying by the fixed power L^2 still gives $o(N)$.

The indices obtained from the computed bounds are already in order. Merge them with the sorted small-prime indices and remove adjacent repetitions. This costs $O((N/T + \sqrt{Ny})L) = o(N)$. Finally, scan the distinct marked block indices in order. For each index r , form rT by a binary shift, since T is a power of two, and write the positive integers from rT through $(r + 1)T - 1$ by successive increments. This skips every unmarked gap. The shift, each increment, and each output record cost $O(L)$; each marked block contains at least one positive integer for large N . Thus the cost is $O(|\mathcal{R}|L) = o(N)$ by (6.4).

The core and its kept output (Step 5). The correctness proof checked the core's input conditions. For its running-time bound we also need

$$\frac{R \log N}{T} \leq \frac{2C_1 \log N}{L} + \frac{\log N}{T} = O(1), \quad \log T = 18 \log \log N + O(1).$$

Thus $\log T < C \log \log N$ for any fixed $C > 18$ and sufficiently large N . Theorem 2.1 gives $N(\log \log N)^{1+o(1)}$ bit operations.

To discard the marked candidates, scan the candidate array together with the ordered list of marked block indices, replacing those candidates by zero. This is a sequential operation on N bits and costs $O(N)$; the $O(L)$ bookkeeping per block contributes only $O((N/T)L) = o(N)$. The resulting binary array has length N and at most

$$\frac{N}{T}R = O(N/L)$$

ones. The vector-to-list bound in Section 2 therefore converts it to an increasing list in $O(N + (N/L) \log N) = O(N)$ time. In particular, the explicit list does not introduce an extra factor of $\log N$ into the total cost.

Repair (Step 6). Theorem 2.2 and (6.4) give

$$O(N) + |\mathcal{R}|L^{11.5+o(1)} = O(N) + O(NL^{-1/2+o(1)}) = O(N).$$

The output contains at most $|\mathcal{R}|$ integers. Filtering out the even ones costs $O(|\mathcal{R}|L) = o(N)$.

Merge and conversion (Step 7). The kept list has $O(N/L)$ entries and the repair list has $O(N/L^{12})$ entries, each of $O(L)$ bits. Together they occupy $O(N)$ bits. Their merge costs $O(N)$ by [5, Lemma 3.2], and Theorem 2.3 converts the complete list to primes in $O(N)$ time. Restricting the output to primes below N_0 takes one further sequential scan.

All seven steps use a fixed number of tapes; their working tapes can be reused between steps. Their combined cost is $N(\log \log N)^{1+o(1)}$. Finally, $N = N_0 + O(L^{18})$ and $L \asymp \log N_0$, so $N/N_0 \rightarrow 1$. The same running-time bound therefore holds with the original input bound N_0 in place of N . For the finitely many inputs below the fixed size needed by the estimates, use the ordinary enumeration algorithm specified in Section 2. This proves Theorem 1.1 for every input in its stated range.

Remark 6.1 (Other polynomial costs for repair). The particular exponent 11.5 does not determine the final bound. Suppose instead that the bulk test cost is $O(N) + |\mathcal{R}|L^{a+o(1)}$, with a fixed. Choose a fixed integer $b > \max(a, 0)$ and retain $L = \lceil \log_2(2N_0) \rceil$. Put $D = L^b$, take T to be the least power of two at least L^{b+6} , and recompute

$$N = T \left\lceil \frac{N_0}{T} \right\rceil, \quad R = \left\lceil \frac{2C_1 T}{L} \right\rceil.$$

Retain the definitions $\eta = L^4$, $Z = \lfloor N^{1/1000} \rfloor$, and $y = \lfloor Z^{1/s} \rfloor$, with the same fixed s and C_1 . Thus $L^{b+6} \leq T < 2L^{b+6}$ and $T \mid N$. Since $T = o(N_0)$, we again have $N/N_0 \rightarrow 1$ and $N \leq 2^L \leq 4N$ for sufficiently large inputs.

The desired marked length is now $O(N/L^b)$, so that the factor L^b saved in the number of repair inputs exceeds their individual cost $L^{a+o(1)}$. The tail estimate still applies with this D , because

$$\frac{Z\sqrt{N}}{N/(DL)} \leq N^{-499/1000} L^{b+1} \rightarrow 0.$$

It therefore gives marked length $O(N/D) = O(N/L^b)$. The truncated estimate also retains its form: for $Q = D^2 Z$,

$$\log Q = 2b \log L + \log Z \leq 2b \log L + \frac{\log N}{1000} = O(L).$$

Consequently the derivation of (4.10) remains valid for these new parameters.

The choice of T comes from the computed excess: its principal contribution to marked length is $O(L \cdot N\eta L/T) = O(NL^6/T)$, which is at most $O(N/L^b)$ when $T \geq L^{b+6}$.

The two principal terms in (4.10) are now

$$\frac{NL^5}{T} = O(N/L^{b+1}), \quad \frac{N}{D} = N/L^b.$$

The remaining term in (4.10) satisfies

$$\frac{Z^4 D^8 L^2}{T^2} \leq N^{1/250} L^{6b-10}, \quad \frac{N^{1/250} L^{6b-10}}{N/L^b} = N^{-249/250} L^{7b-10} \rightarrow 0.$$

Thus it is $o(N/L^b)$ for every fixed b . For the computed excess, (5.1) gives

$$\sum_r (U_r - W_r) \ll \frac{N\eta L}{T} + \eta S \ll \frac{N}{L^{b+1}} + N^{1/2+1/2000} L^4 \ll \frac{N}{L^{b+1}}.$$

The ratio of the second term to the first scale is $N^{-999/2000} L^{b+5} \rightarrow 0$. The marked-length argument multiplies this error by L/C_1 , so its contribution is $O(N/L^b)$. The small-prime blocks have total length $O(T\sqrt{Ny})$, and

$$\frac{T\sqrt{Ny}}{N/L^b} \ll L^{2b+6} N^{-1/2+1/(2000s)} \rightarrow 0.$$

Hence $|\mathcal{R}| \ll N/L^b$, and repair costs

$$O(N) + O(NL^{a-b+o(1)}) = O(N).$$

Writing the repair inputs takes $O(|\mathcal{R}|L) = O(N)$ as well, since $b \geq 1$. The definitions $\eta = L^4$, $K = \lceil N/\eta \rceil$, and $J = K(L+1)$ retain the same form, so $J = O(N/L^3)$. Moreover,

$$\frac{T}{\eta} \geq L^{b+2} \rightarrow \infty, \quad T < 2L^{b+6} \leq N$$

for sufficiently large N . Thus the convolution's size conditions still hold; its list and gcd conditions are unchanged, and its running time remains $O(N)$.

For the core, the new threshold still satisfies

$$\frac{R}{T} \leq \frac{2C_1}{L} + \frac{1}{T} < \frac{1}{2}, \quad \frac{R \log N}{T} = O(1).$$

Also $\log T = (b+6) \log \log N + O(1)$, so we may use any fixed $C > b+6$ in Theorem 2.1. Its running time is again $N(\log \log N)^{1+o(1)}$. The kept list still has $O(N/L)$ entries, and the repair list has $O(N/L^b)$ entries, also $O(N/L)$ because $b \geq 1$. Their merge and conversion therefore remain $O(N)$. Finally, $0 \leq N - N_0 < T < 2L^{b+6} = o(N_0)$, so the same bound holds for the original input size N_0 .

7 Concluding remarks

The algorithm reaches the running time of Harvey's conditional algorithm by changing how its output is certified. The computed upper bounds need not be close to the exact majorant sums on every block. Their nonnegative total excess over those sums is enough to control the number of additional blocks sent to repair. With the chosen block length, both certification

and repair take $O(N)$ bit operations; the core accounts for the remaining factor in the running time.

Appendix C gives an optional refinement: once the exact compressed vectors are available, decoding the unmarked blocks can also be done in $O(N)$ bit operations. This refinement is not used in the proof of Theorem 1.1, which uses Harvey’s core unchanged.

The proof does not establish an $O(N \log \log N)$ running time. The bound $N(\log \log N)^{1+o(1)}$ permits an additional factor $(\log \log N)^{o(1)}$, which need not be bounded as N increases. Removing such factors from Harvey’s three main bounds is his question (4) [5, Section 1.4]. The certification and repair developed here already take $O(N)$ time; the unresolved factor remains in the cost of the imported core.

There are also space and practical limitations. As written, the algorithm stores the full N -bit candidate array. It does not give the segmented algorithm using $O(N^{1-\varepsilon})$ bits of memory for a fixed $0 < \varepsilon < 1$ asked for in Harvey’s question (1). The parameter choices and estimates apply only for very large inputs, so this is not a practical prime-enumeration algorithm.

Finally, a deterministic algorithm is a special case of a Las Vegas algorithm: it can ignore its random bits. Theorem 1.1 therefore also meets the running-time target in Harvey’s question (3). Separately, the crowded-block estimate appears sufficient to improve his probabilistic algorithm to the same expected running time, retaining its randomized acceptance and global verification steps without computing the convolution certificate. This adaptation is not developed here.

A Sieve coefficients

This appendix gives the coefficients used in Lemma 3.1, proves their pointwise upper-bound property, and checks that they can be computed within the required time. The estimate for their mean is the cited fundamental lemma of sieve theory.

The example in Section 3 showed why deleting individual terms from inclusion–exclusion can give a negative weight. We will instead delete groups whose total sign is nonpositive. To form such groups, order the chosen primes from largest to smallest and stop at the first failed test. We will test only at odd positions, where the prefix has negative inclusion–exclusion sign. Allowing all choices of smaller primes either cancels this sign to zero or leaves it negative. Removing the whole group can therefore only increase the sum. The proof below makes this grouping precise.

The cutoff tests must also keep the mean small. For this we use the upper beta-sieve with $\beta = 30$, level Z , and sifting limit y . Write a squarefree integer as a product of its distinct prime factors in decreasing order:

$$e = p_1 \cdots p_\omega, \quad y > p_1 > \cdots > p_\omega,$$

and set

$$\lambda_e = \begin{cases} (-1)^\omega, & p_1 \cdots p_j p_j^\beta < Z \text{ for every odd } j \leq \omega, \\ 0, & \text{otherwise.} \end{cases} \quad (\text{A.1})$$

The empty product gives $\lambda_1 = 1$. Set $\lambda_e = 0$ if e is not squarefree or has a prime factor at least y . In the nonzero case the sign $(-1)^\omega$ is the usual inclusion–exclusion sign; the inequalities decide which terms to keep. The factor p_j^β requires a prefix to lie further below Z than the condition $p_1 \cdots p_j < Z$ would require. The imported mean estimate below applies to this selection rule with fixed $\beta = 30$; the upper-bound argument alone does not supply that mean estimate. This is the construction of upper beta-sieve weights in [3, Section 6.4], with the conventions used for ρ_e^+ in [7, Section 2, preceding (14)].

Why the support lies below Z . Suppose $\lambda_e \neq 0$. If ω is odd, the condition with $j = \omega$ gives $ep_\omega^\beta < Z$, so $e < Z$. If $\omega \geq 2$ is even, use the condition with $j = \omega - 1$:

$$p_1 \cdots p_{\omega-1} p_{\omega-1}^\beta < Z.$$

Since $p_\omega < p_{\omega-1}^\beta$, replacing the last power by p_ω gives $e < Z$ again. The remaining case is $e = 1$, which also satisfies $e < Z$ for the large inputs under consideration.

Why the divisor sum is an upper bound. Recall the notation

$$P(y) = \prod_{p < y} p, \quad \theta(n) = \sum_{e|n} \lambda_e.$$

Fix $n \geq 1$, and let A be the set of primes below y that divide n . Every squarefree divisor of n with prime factors below y is the product of one subset of A . Full inclusion–exclusion gives

$$\sum_{B \subseteq A} (-1)^{|B|} = \begin{cases} 1, & A = \emptyset, \\ 0, & A \neq \emptyset, \end{cases} = \mathbf{1}_{(n, P(y))=1}.$$

The sum $\theta(n)$ keeps only those subsets whose decreasing prime lists pass every test in (A.1). To prove that this increases the sum, we group the omitted subsets according to their first failed test.

Such a test occurs at an odd index j . Fix its prefix $p_1 > \cdots > p_j$: all earlier tests pass, and $p_1 \cdots p_j p_j^\beta \geq Z$. Every omitted subset with this first failed prefix consists of the prefix together with an arbitrary subset of

$$A_{<p_j} = \{p \in A : p < p_j\}.$$

Conversely, adjoining any such subset leaves the same first failed test, because that test uses only the prefix. The total sign of this group is therefore

$$(-1)^j \sum_{C \subseteq A_{<p_j}} (-1)^{|C|} = - \sum_{C \subseteq A_{<p_j}} (-1)^{|C|} \leq 0.$$

The last sum is 1 if $A_{<p_j}$ is empty. Otherwise choose one prime in $A_{<p_j}$ and pair each subset not containing it with the subset obtained by adding it; the two signs cancel. Thus in that case the last sum is zero. For example, if there are two available smaller primes, the four extensions of an odd prefix have signs $-, +, +, -$, whose sum is zero. Every omitted subset belongs to exactly one of these groups. Removing groups with nonpositive total sign can only increase the full inclusion–exclusion sum. Consequently

$$\theta(n) \geq \mathbf{1}_{(n, P(y))=1} \geq 0.$$

This is the upper-bound property in [3, (6.26)–(6.27)] and [7, (14)].

The mean estimate supplied by sieve theory. The fundamental lemma, in the form used here, states that for these coefficients and fixed $\beta = 30$,

$$\prod_{p < y} \left(1 - \frac{1}{p}\right) \leq \sum_e \frac{\lambda_e}{e} \leq 2 \prod_{p < y} \left(1 - \frac{1}{p}\right)$$

once $\log Z / \log y$ is sufficiently large for this fixed β [3, (6.31)–(6.33), Lemma 6.8]. The density $1/e$ appears because the proportion of integers divisible by e is $1/e$. The product on the two

sides is the exact survivor density of full inclusion–exclusion, computed in Section 3. The inequality therefore says that the deletions needed to restrict the divisor indices increase the mean by at most a factor of two. This estimate is the non-elementary sieve input; the pointwise argument above does not by itself bound the mean sharply enough. Our choice $y \leq Z^{1/s}$ makes $\log Z / \log y \geq s$ for $y > 1$, so a sufficiently large fixed s meets its hypothesis. The same estimate appears in [7, (28)], with $Z = X^{1/1000}$ and $y = X^\delta$.

Computing the coefficients. Recall that $Z \leq N \leq 2^L$ and $L \asymp \log N$, so every integer at most N has $O(L)$ bits. For each integer $e \leq Z$, keep a current remainder, initially e , and try the candidate divisors $2, 3, \dots, e$ in order. On a successful division, record the divisor and its multiplicity while dividing it out repeatedly. Testing all candidates up to e takes at most e failed divisions; there are at most $\log_2 e$ successful divisions because each reduces the remainder by a factor of at least two. A successful candidate is prime: any smaller prime factor would already have been removed. Thus this procedure factors e in $O(Z)$ divisions of $O(L)$ -bit integers. Elementary division takes $O(L^2)$ bit operations, giving $O(ZL^2)$ per integer e .

A repeated prime factor makes e nonsquarefree, and a factor at least y makes it ineligible; in either case write $\lambda_e = 0$. Otherwise there are at most L distinct factors. They were found in increasing order, so a backwards scan reads them in the decreasing order needed in (A.1). Maintain the prefix product and, at each odd position j , compute p_j^β and compare the product with Z . Since $\beta = 30$ is fixed, each such test uses a bounded number of multiplications. Its largest operand is at most $N^{\beta+1}$, hence has $O(L)$ bits. At $O(L^2)$ bit operations per multiplication, all $O(L)$ tests take $O(L^3)$ time for this e . The factor record occupies $O(L^2)$ bits, so reading, reversing, and comparing its entries fits within this bound. If a cutoff test fails, write $\lambda_e = 0$; if they all pass, write $(-1)^\omega$, where ω is the number of prime factors. For $e = 1$ there are no factors or tests, and the output is 1.

The total for $e = 1, \dots, Z$, including writing the coefficients, is therefore

$$O(Z^2 L^2 + ZL^3 + Z).$$

This is bounded by the claimed $O(Z^2 L^4)$. These operations use only counters and factor records on a fixed number of tapes. The product $P(y)$ is only notation; the algorithm never needs to write it down.

Remark A.1 (Other choices of sieve level). The proof uses $Z = \lfloor N^{1/1000} \rfloor$ to give one definite algorithm. The same argument allows $Z = \lfloor N^c \rfloor$ for any fixed rational $0 < c < 1/4$, with a larger constant C_1 if necessary. Constants in this variation may depend on the fixed c . Keep $D = L^{12}$, $L^{18} \leq T < 2L^{18}$, and $y = \lfloor Z^{1/s} \rfloor$, with the same sufficiently large fixed s ; as before, $L \asymp \log N$. Indeed, the divisor bound for the truncated majorant becomes $Q = D^2 Z$, so the remainder in (4.7) is at most

$$\frac{Q^4 L^2}{T^2} \ll N^{4c} L^{8 \cdot 12 + 2 - 2 \cdot 18} = N^{4c} L^{62}.$$

Relative to the required marked length N/L^{12} , this is $N^{4c-1} L^{74}$, which tends to zero because $4c - 1 < 0$. This explains the restriction $c < 1/4$ for this estimate.

For the other Z -dependent terms, the earlier bounds become

$$\begin{aligned} Z^2 L^4 &\leq N^{2c} L^4 && \text{(coefficient construction),} \\ Z\sqrt{N} &\leq N^{c+1/2} && \text{(tail error),} \\ S &\ll \sqrt{NZ} \leq N^{(1+c)/2} && \text{(divisor-list length).} \end{aligned}$$

The small-prime terms y^2 and $\sqrt{Ny}$ are no larger than Z^2 and $\sqrt{NZ}$ because $y \leq Z$. All remaining multipliers in these terms are fixed powers of L . Each displayed exponent of N is strictly below one, so each such term is $o(N/L^A)$ for every fixed $A > 0$.

The rounded choices give $\log y = (c/s) \log N + O_s(1)$. For the threshold comparison, recall that the truncated mean satisfies $\nu_D \leq 4/\log y$ and that we require $12L\nu_D < C_1$. Now $L/\log y = O(s/c)$, so increasing the fixed C_1 if necessary preserves that inequality.

Writing $c = a/b$ with fixed positive integers a, b , the integer $Z = \lfloor N^{a/b} \rfloor$ is characterized by $Z^b \leq N^a < (Z+1)^b$. Use binary search on the integers in $[0, N]$, which contains Z because $c < 1$. At each step compare the b th power of the midpoint with N^a and retain the half containing the largest integer z with $z^b \leq N^a$. There are therefore $O(\log N) = O(L)$ comparisons. Since a and b are fixed, all powers involved have $O(L)$ bits and can be computed by a fixed number of elementary multiplications. The resulting time is polynomial in L . This is why the optional variation is stated for rational c .

B Certified logarithms and exponential thresholds

The algorithm in Section 5 uses logarithms to assign bin indices and exponentials to enclose products. It allows an absolute error of $1/4$ in each numerical evaluation. The histogram entries and the resulting block bounds are still exact integers: the numerical error only changes the intervals used to locate the products.

We give elementary routines based on power series. The first step is to reduce the argument to a bounded interval, where the series converges quickly. We then control three errors separately: rounding the argument, rounding the arithmetic in the finite sum, and omitting the rest of the series. Finally we allow for the scaling that restores the original argument. Faster routines are available [1], but these suffice for the required bit bound.

Lemma B.1. *Let N, K, L be integers with $N \geq 3$, $1 \leq K \leq N$, and $N \leq 2^L \leq 4N$. The following quantities can be computed deterministically in $O(L^3)$ bit operations each:*

- (i) *an approximation to $K \log_2 u$ of absolute error at most $1/4$, for an integer $1 \leq u \leq N$;*
- (ii) *an approximation to $2^{j/K}$ of absolute error at most $1/4$, for an integer j with $|j| \leq 2K(L+1) + 8$.*

The precision and stopping rules can be chosen from L alone.

Proof. The accuracy needed inside a routine is determined by its final scaling. The logarithm will be multiplied by $K \leq 2^L$. For the exponential, write $j = vK + r$ with $0 \leq r < K$. Then

$$2^{j/K} = 2^v \exp((r/K) \log 2).$$

The input bound gives $|j/K| \leq 2L + 10$. Since $2L + 10$ is an integer, its floor v also satisfies $|v| \leq 2L + 10$. Thus the exponential's last multiplication can amplify an error by as much as 2^{2L+10} . It is enough to make the errors before these scalings exponentially small in L .

We use the target error and the working parameters

$$\delta = 2^{-4L-40}, \quad n_0 = 4L + 40, \quad p_0 = 20L + 100, \quad \epsilon = 2^{-p_0}.$$

Each series will use n_0 terms, and each rounded number will have p_0 fractional binary digits. We will prove an error at most δ for each unscaled logarithm and at most 8δ for the exponential before multiplication by 2^v . The factors 2^L and 2^{2L+10} will then leave errors below $1/4$.

To specify the computation, let

$$\text{rd}(x) = 2^{-p_0} \lfloor 2^{p_0} x \rfloor \quad (x \geq 0).$$

This rounds down and gives $0 \leq x - \text{rd}(x) < \epsilon$. For a nonnegative rational $x = A/B$, the rounded value is obtained by dividing the integers $2^{p_0} A$ and B and keeping the integer quotient. In particular, this rule implements rounded products and quotients of stored numbers using integer arithmetic. During the series calculations, all stored numbers have denominator 2^{p_0} , so their additions are exact. Rounding down also keeps a nonnegative number below any known upper bound for its exact value; no separate clipping operation is needed.

The logarithm series and its argument. Let v be the position of the leading binary digit of u , starting with position zero. Reading the binary representation therefore gives v without evaluating a logarithm, and

$$2^v \leq u < 2^{v+1}, \quad t = u/2^v \in [1, 2).$$

Set

$$z = \frac{t-1}{t+1} = \frac{u-2^v}{u+2^v}, \quad 0 \leq z < \frac{1}{3}.$$

The second expression shows how to compute $\hat{z} = \text{rd}(z)$ by one integer division. Both z and $\hat{z}$ lie in $[0, 1/3]$, and their difference is at most ϵ .

Integrating the geometric series $2/(1-z^2) = 2 \sum_{h \geq 0} z^{2h}$ gives

$$\log t = \log \frac{1+z}{1-z} = 2 \sum_{h=0}^{\infty} \frac{z^{2h+1}}{2h+1}.$$

We also need $\log 2$. The same formula computes it by taking $z = 1/3$. The following error estimates apply to both uses, including this endpoint. Write

$$S(z) = 2 \sum_{h=0}^{n_0-1} \frac{z^{2h+1}}{2h+1}.$$

Its omitted tail is bounded by

$$2 \sum_{h=n_0}^{\infty} \frac{z^{2h+1}}{2h+1} \leq \frac{2z^{2n_0+1}}{1-z^2} \leq 3 \cdot 9^{-n_0}.$$

Here the factor 9^{-n_0} comes from $z^2 \leq 1/9$; this is why only $O(L)$ terms are needed for an error exponentially small in L .

Computing the logarithm series and bounding its error. Compute a rounded square $\hat{q} = \text{rd}(\hat{z}^2)$, then form the odd powers by

$$\hat{p}_0 = \hat{z}, \quad \hat{p}_{h+1} = \text{rd}(\hat{q} \hat{p}_h) \quad (0 \leq h < n_0 - 1).$$

Thus $\hat{p}_h$ approximates $\hat{z}^{2h+1}$. We have $0 \leq \hat{q} \leq \hat{z}^2 \leq 1/9 < 1/4$ and $|\hat{q} - \hat{z}^2| \leq \epsilon$. If $e_h = |\hat{p}_h - \hat{z}^{2h+1}|$, adding and subtracting $\hat{q} \hat{z}^{2h+1}$ gives

$$\begin{aligned} e_{h+1} &\leq \epsilon + |\hat{q}| e_h + |\hat{q} - \hat{z}^2| |\hat{z}|^{2h+1} \\ &\leq \frac{1}{4} e_h + 2\epsilon. \end{aligned}$$

The three terms are the new product-rounding error, the propagated power error, and the error in the stored multiplier $\hat{q}$. Since $e_0 = 0$, induction gives $e_h \leq 3\epsilon$: substituting this bound on the right gives at most $11\epsilon/4 < 3\epsilon$.

The computed sum is

$$\hat{S} = 2 \sum_{h=0}^{n_0-1} \text{rd} \left(\frac{\hat{p}_h}{2h+1} \right).$$

For each term, the power error contributes at most 3ϵ , and the division contributes at most ϵ . The outer factor 2 therefore gives

$$|\hat{S} - S(\hat{z})| \leq 8n_0\epsilon.$$

This accounts for the arithmetic with the rounded argument held fixed. To account for replacing z by $\hat{z}$, differentiate the finite polynomial:

$$S'(x) = 2 \sum_{h=0}^{n_0-1} x^{2h} \leq \frac{2}{1-x^2} \leq \frac{9}{4} < 3 \quad (0 \leq x \leq 1/3).$$

The mean value theorem consequently gives $|S(\hat{z}) - S(z)| \leq 3\epsilon$. Adding this error and the tail bound yields

$$|\hat{S} - \log t| \leq (8n_0 + 3)\epsilon + 3 \cdot 9^{-n_0} \leq 16n_0\epsilon + 3 \cdot 9^{-n_0}.$$

The same estimate holds for the computation of $\log 2$. For our parameters it is at most δ . Indeed, for $L \geq 2$, $n_0 = 4L + 40 \leq 2^{L+4}$ and $9 > 8$, so

$$16n_0\epsilon + 3 \cdot 9^{-n_0} \leq 2^{-19L-92} + 3 \cdot 2^{-12L-120} \leq 2^{-4L-40} = \delta.$$

Restoring the binary logarithm. Let $a = \log t$, $b = \log 2$, and let $\hat{a}, \hat{b}$ be the approximations just computed. The first term of the series for b and the geometric-series upper bound give

$$\frac{2}{3} \leq b \leq \frac{2}{3} \sum_{h=0}^{\infty} 9^{-h} = \frac{3}{4}.$$

Since $|\hat{b} - b| \leq \delta$, we have $1/2 < \hat{b} < 1$. Thus division by $\hat{b}$ does not amplify the error greatly. More precisely, $0 \leq a \leq b$ gives

$$\begin{aligned} \left| \frac{\hat{a}}{\hat{b}} - \frac{a}{b} \right| &\leq \frac{|\hat{a} - a|}{\hat{b}} + \frac{a|b - \hat{b}|}{b\hat{b}} \\ &\leq 4\delta. \end{aligned}$$

Compute $\text{rd}(\hat{a}/\hat{b})$, add the exact integer v , and multiply by the integer K . These last two operations are exact on the stored dyadic numbers. The result approximates $K(v + a/b) = K \log_2 u$ with error at most

$$K(4\delta + \epsilon) \leq 5 \cdot 2^L \delta = 5 \cdot 2^{-3L-40} < \frac{1}{4}.$$

This proves (i).

The exponential series and its argument. Use the Euclidean division $j = vK + r$ introduced above, with $0 \leq r < K$, and put $t = r/K$. This also handles negative j : $v = \lfloor j/K \rfloor$, and the remainder is still nonnegative. We must approximate $\exp(z)$ for $z = t \log 2 \in [0, 1]$, then multiply by 2^v .

Compute $\hat{t} = \text{rd}(r/K)$, compute $\hat{b}$ as above, and set $\hat{z} = \text{rd}(\hat{t}\hat{b})$. Since $0 \leq \hat{t} \leq t < 1$ and $0 < \hat{b} < 1$, the computed argument lies in $[0, 1]$. Its error satisfies

$$\begin{aligned} |\hat{z} - tb| &\leq \epsilon + |\hat{t} - t| |\hat{b}| + t|\hat{b} - b| \\ &\leq 2\epsilon + \delta. \end{aligned}$$

The terms of $\exp(\hat{z}) = \sum_{h \geq 0} \hat{z}^h/h!$ can be obtained successively by multiplying the preceding term by $\hat{z}$ and dividing by its new index. With rounding at each operation, the actual computation is

$$\hat{c}_0 = 1, \quad \hat{m}_h = \text{rd}(\hat{c}_{h-1}\hat{z}), \quad \hat{c}_h = \text{rd}(\hat{m}_h/h) \quad (1 \leq h < n_0).$$

Let $e_h = |\hat{c}_h - \hat{z}^h/h!|$. The multiplication error is divided by h , while the subsequent division adds its own error. Consequently

$$e_h \leq \frac{|\hat{z}|}{h} e_{h-1} + \frac{\epsilon}{h} + \epsilon \leq \frac{e_{h-1}}{h} + 2\epsilon.$$

Here $e_0 = 0$, so $e_1 \leq 2\epsilon$. For $h \geq 2$, the bound $e_{h-1} \leq 4\epsilon$ gives $e_h \leq 4\epsilon$ again. Thus summing the n_0 computed terms introduces at most $4n_0\epsilon$ of arithmetic error.

The argument error is separate. The derivative of $\exp(x)$ is at most 3 on $[0, 1]$, so replacing z by $\hat{z}$ changes its value by at most $3(\delta + 2\epsilon)$. For the omitted tail, $0 \leq \hat{z} \leq 1$ gives

$$\sum_{h=n_0}^{\infty} \frac{\hat{z}^h}{h!} \leq \frac{1}{n_0!} \sum_{r=0}^{\infty} (n_0 + 1)^{-r} \leq \frac{2}{n_0!} \leq 2^{2-n_0} = 4\delta.$$

In the last step, every factor in $2 \cdot 3 \cdots n_0$ is at least 2, so $n_0! \geq 2^{n_0-1}$. The three error contributions therefore sum to at most

$$4n_0\epsilon + 3(\delta + 2\epsilon) + 4\delta = (4n_0 + 6)\epsilon + 7\delta \leq 8\delta.$$

The last inequality uses $(4n_0 + 6)\epsilon \leq 16n_0\epsilon \leq \delta$, already established in the logarithm estimate. Finally multiply the computed sum by 2^v . This is an exact binary shift; if $v < 0$, keep the additional fractional digits. Since $|v| \leq 2L + 10$, the resulting absolute error is at most

$$2^{2L+10} \cdot 8\delta = 8 \cdot 2^{-2L-30} < \frac{1}{4}.$$

This proves (ii).

Bit cost. The preceding formulas describe integer algorithms. Their input integers have $O(L)$ bits: this includes j , since $|j| \leq 2K(L+1) + 8$ and $K \leq 2^L$. Each stored dyadic number has $p_0 = O(L)$ fractional bits. Before the final scaling, the powers, series terms, and series sums are bounded by absolute constants; adding v in the logarithm routine introduces only $O(\log L)$ integer bits. Multiplying by K introduces at most L more bits. In the exponential routine, the final shift changes the number of integer or fractional bits by at most $2L + 10$. Hence every operand and every output has $O(L)$ bits. Intermediate integer products and

the scaled numerators used for rounding also have $O(L)$ bits, since they at most add a fixed number of these bit lengths.

Reading the leading digit, Euclidean division, and forming the initial rational arguments require a fixed number of scans and integer arithmetic operations. Each requested output then uses at most two series evaluations, each with $n_0 = O(L)$ terms and a fixed number of arithmetic operations per term. Evaluate the terms in index order, keeping only the current term, its index, the fixed multiplier, and a running sum. These occupy a fixed number of $O(L)$ -bit registers; no array of all the terms is needed. Elementary multiplication and division on $O(L)$ -bit integers cost $O(L^2)$ bit operations; addition, comparison, and shifts cost no more. The total is therefore $O(L) \cdot O(L^2) = O(L^3)$, on a fixed number of tapes. All bounds above apply for $L \geq 2$, which follows from $N \geq 3$. $\square$

C An optional linear-time decoding step

The main proof uses Harvey's core as stated in Theorem 2.1. This appendix concerns one stage inside that core: recovering the occupied positions of a block from its compressed vector. Our marking rule already guarantees that each unmarked block has at most R occupied positions. We use that guarantee to batch the lookups in the decoder, so that the decoding and its explicit output together cost $O(N)$ bit operations.

We first state the additional facts about Harvey's representation and decoder that this refinement uses. Put

$$h = \lceil \log_2(T + 1) \rceil.$$

Harvey works in a field $\mathbb{F}$ with 2^h elements, represented as $\mathbb{F}_2[z]/(P)$, where P is an irreducible polynomial of degree h over the two-element field $\mathbb{F}_2$. Here irreducible means that P cannot be factored into two nonconstant polynomials of smaller degree over $\mathbb{F}_2$. A field element has a unique representative

$$c_0 + c_1z + \cdots + c_{h-1}z^{h-1}, \quad c_i \in \{0, 1\},$$

so it is stored by the h bits $c_0, \dots, c_{h-1}$. Addition uses coefficientwise addition modulo two. Multiplication forms the polynomial product and reduces it modulo P ; elementary polynomial arithmetic performs this in $O(h^2)$ bit operations.

The representation includes an element α of multiplicative order $2^h - 1$: this is the least positive exponent j for which $\alpha^j = 1$. Harvey gives a deterministic construction of P and α in time polynomial in T [5, Section 5.1, Lemmas 4.5 and 4.7]. Since $2^h - 1 \geq T$, the elements α^{-t} for $0 \leq t < T$ are distinct. Indeed, an equality for two different positions would give $\alpha^{t-u} = 1$ for a nonzero integer $t - u$ of absolute value less than T , contradicting this order.

Harvey's compression map κ represents a binary vector with T positions by $2R$ elements of $\mathbb{F}$, or $2hR$ bits [5, Section 5.1]. We use the following decoding guarantee. If the input is exactly $\kappa(a)$ and a has at most R ones, the first two steps of his decoder compute the polynomial

$$g(Y) = \prod_{t: a_t=1} (Y - \alpha^{-t})$$

and all its roots in $R(\log T)^{5+o(1)}$ bit operations [5, proof of Theorem 5.5]. For an empty vector this is the constant polynomial 1, with no roots. In every case, the resulting roots give the occupied positions indirectly: the root α^{-t} identifies position t . What remains is to carry out these lookups efficiently on sequential tapes.

Proposition C.1. *Use the parameters of Section 2. The relations used below are*

$$T \mid N, \quad N \leq 2^L \leq 4N, \quad L^{18} \leq T < 2L^{18}, \quad R = O(T/L).$$

Suppose that the exact compressed vectors $\kappa(a^r)$ and one bit indicating whether each block is marked are given in block order. If $w_r \leq R$ on every unmarked block, then all odd square-primes in those blocks can be listed in increasing order in $O(N)$ bit operations.

Proof. Harvey’s algorithm produces the exact compressed vectors in block order before decoding [5, Proposition 6.20]. Their production is not part of the decoding cost in this proposition. For each unmarked block, the input is exactly $\kappa(a^r)$ and $\text{wt}(a^r) = w_r \leq R$. The decoding guarantee therefore supplies precisely its roots. Harvey’s final recompression test checks a candidate when this guarantee is unavailable; here the low-weight condition has already been proved, so the test can be omitted. A marked block contributes no roots to this procedure.

A common lookup table. Construct the table of pairs

$$(\alpha^{-t}, t), \quad 0 \leq t < T,$$

using the same field representation and the same α as the compression map. The distinctness proved above ensures that each recovered root has exactly one matching position. Sort the table by the h -bit encoding of its first component, using lexicographic order on bit strings. This order is only for searching the table; it need not respect the field operations.

The construction has a polynomial cost in T . After constructing the field, compute α^{-1} once. For example, $\alpha^{-1} = \alpha^{2^h-2}$. To compute this power, successive squarings give $\alpha, \alpha^2, \alpha^4, \dots$; multiply those powers whose exponents correspond to one bits in the binary expansion of $2^h - 2$. This exponent has h bits, so there are $O(h)$ field multiplications, or $O(h^3)$ bit operations. Starting from 1, successive multiplication by this inverse then generates the T table entries in $O(Th^2)$ bit operations. Each record has $O(\log T)$ bits because both h and the bit length of t are $O(\log T)$. Sorting the table costs $O(T(\log T)^2)$ by the record-sorting bound in Section 2. Since $T < 2L^{18}$, the field construction, table generation, and sorting together take $L^{O(1)}$ time.

Why we use batches of L blocks. The table occupies $O(T \log T)$ bits. Scanning it once for each of the N/T blocks would cost $O(N \log T)$, exceeding the desired $O(N)$ decoding bound. A scan can instead answer the queries from several blocks at once, provided those queries have first been sorted in the same order as the table.

Suppose temporarily that we use B blocks per batch. There are at most $\lceil N/(BT) \rceil$ batches, so the total table-scan cost becomes

$$O\left(\left(\frac{N}{BT} + 1\right) T \log T\right) = O\left(\frac{N \log T}{B} + T \log T\right).$$

A batch contributes at most BR roots. Since $R = O(T/L)$, the choice $B = L$ keeps this number $O(T)$, comparable to the size of the lookup table, and reduces the leading scan cost to $O(N \log T/L) = o(N)$. We therefore use batches of L consecutive blocks, allowing a shorter last batch. Marked blocks retain their places in this division into batches but supply no queries.

Looking up and ordering the positions. Attach to each recovered root the block's index within its batch. This index uses $O(\log L) = O(\log T)$ bits, so each query record has $O(\log T)$ bits. Sort these records by the binary encoding of the root. Scan them together with the sorted lookup table, and append the matching position t to each query record. If the current table key is smaller than the current query key, advance the table head. At equality, append the position and advance only the query head. Every query key occurs in the table, so these comparisons reach its matching entry. A root can occur in several different blocks; keeping the table head fixed at equality lets the same entry answer all those queries. Thus one scan answers every query in the batch, including repeated roots.

The records now contain both a local block index and a position within that block. Sort them by the local block index first, and by the position second. This restores the order in which the integers must be output.

There are at most $LR = O(T)$ query records per batch. Sorting records of $O(\log T)$ bits therefore costs

$$O(LR \log T \log(2 + LR)) = O(LR(\log T)^2)$$

for each of the two sorts. The lookup scan, including the return scan that resets the table head for the next batch, costs $O(T \log T)$. The query list is no longer than $O(T)$ records, so its scan fits within the same bound. The total lookup and sorting cost over all batches is consequently

$$\begin{aligned} O\left(\left(\frac{N}{LT} + 1\right)(LR(\log T)^2 + T \log T)\right) \\ = O\left(\frac{NR}{T}(\log T)^2 + \frac{N}{L} \log T\right) + L^{O(1)}. \end{aligned}$$

The polynomial remainder includes the possible last partial batch; its size is bounded by the cost of one full batch, with $LR = O(T)$ and $T = L^{O(1)}$.

Root computation over the at most N/T unmarked blocks adds $O((NR/T)(\log T)^{5+o(1)})$. Combining this with the lookup costs and the one-time table construction gives

$$O\left(\frac{NR}{T}(\log T)^{5+o(1)} + \frac{NR}{T}(\log T)^2 + \frac{N}{L} \log T\right) + L^{O(1)}.$$

Here $R/T = O(1/L)$ and $\log T = O(\log L)$. Dividing each of the three main terms by N gives, respectively,

$$O\left(\frac{(\log L)^{5+o(1)}}{L}\right), \quad O\left(\frac{(\log L)^2}{L}\right), \quad O\left(\frac{\log L}{L}\right),$$

all tending to zero. Also $L^{O(1)} = o(N)$. The internal decoding work is therefore $o(N)$.

We must still read the supplied data and write the result. Reading all $2hR$ compressed bits and the marking bit for every block, including marked blocks, costs

$$O\left(\frac{N}{T}(R \log T + 1)\right) = O\left(\frac{N \log L}{L} + \frac{N}{T}\right) = o(N).$$

The batches consist of consecutive blocks, and the final sort within each batch orders the records by block and then by position. Retain the global index b of the first block of the current batch, initially 0, and increase it by L between batches. A record with local block index i belongs to global block $r = b + i$. Form r only when emitting the record, at cost $O(L)$; the records used by the two sorts therefore retain their $O(\log T)$ -bit size. For this record with position t , output the integer $rT + t$. Because $0 \leq t < T$, these integers are increasing within a block and every integer from block r precedes every integer from block $r + 1$. Processing the batches in order gives the required global order.

There are at most $(N/T)R = O(N/L)$ output positions. Computing $rT + t$ need not involve a multiplication for each record. Maintain a current global block index and its offset, initially 0 and 0. Before emitting a record from a later block, advance the index by one and the offset by T until that block is reached, passing through empty and marked blocks as well. Then one addition of t gives the output integer. Forming r , adding t to the offset, and writing the result's $O(L)$ bits cost $O(L)$ per position, for a total of $O(N)$. The offset advances at most once per block, and the batch start changes at most once per batch. Maintaining these counters and offsets therefore costs $O((N/T)L) = o(N)$ in total. Adding these output costs to the internal decoding work proves the stated $O(N)$ bound. $\square$

Use of AI assistance

An earlier version of this manuscript was produced largely with AI assistance and extensively revised under human direction. This exposition was rewritten and cross-checked with AI assistance. These checks do not constitute formal verification.

References

- [1] R. P. Brent and P. Zimmermann, *Modern Computer Arithmetic*, Cambridge Monographs on Applied and Computational Mathematics **18**, Cambridge University Press, 2011. doi:10.1017/CBO9780511921698.
- [2] M. Farach-Colton and M. Tsai, *On the complexity of computing prime tables*, in Algorithms and Computation, Lecture Notes in Computer Science **9472**, Springer, Heidelberg, 2015, 677–688. doi:10.1007/978-3-662-48971-0_57.
- [3] J. Friedlander and H. Iwaniec, *Opera de Cribro*, American Mathematical Society Colloquium Publications **57**, American Mathematical Society, 2010. doi:10.1090/coll/057.
- [4] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, 3rd ed., Cambridge University Press, Cambridge, 2013. doi:10.1017/CBO9781139856065.
- [5] D. Harvey, *Faster enumeration of primes*, arXiv:2606.22851v1, 22 June 2026. <https://arxiv.org/abs/2606.22851v1>. Theorem and proposition numbers in this manuscript refer to this version.
- [6] D. Harvey and J. van der Hoeven, *Integer multiplication in time $O(n \log n)$* , Annals of Mathematics (2) **193** (2021), no. 2, 563–617. doi:10.4007/annals.2021.193.2.4.
- [7] K. Matomäki, *Almost primes in almost all very short intervals*, Journal of the London Mathematical Society (2) **106** (2022), 1061–1097. doi:10.1112/jlms.12592. The equation numbers cited here refer to arXiv:2012.11565v3, 18 January 2022.
- [8] I. S. Sergeev, *On the complexity of computing prime tables on the Turing machine*, Prikladnaya Diskretnaya Matematika, no. 1 (2016), 86–91 (Russian). English translation: *On the complexity of computing prime tables on a Turing machine*, arXiv:1604.01154, 2016. <https://arxiv.org/abs/1604.01154>.
- [9] A. Schönhage, A. F. W. Grotefeld and E. Vetter, *Fast Algorithms: A Multitape Turing Machine Implementation*, Bibliographisches Institut, Mannheim, 1994.
- [10] A. Schönhage and V. Strassen, *Schnelle Multiplikation großer Zahlen*, Computing **7** (1971), 281–292. doi:10.1007/BF02242355.