

Computing $\binom{N}{R} \bmod m$ for Arbitrary Composite Moduli and $N \leq 10^{18}$: A Prime-Power Decomposition Engine with Two Memory Regimes and Division-Free Arithmetic

Vadik Malik, Sarthak Vashishtha, Rudr Pratap

Departments of Computer Science & Engineering (CSE), Raj Kumar Goel Institute of Technology (RKGIT)
Ghaziabad, India
vadik.core@gmail.com

Abstract—Binomial coefficients modulo an integer are a primitive of combinatorial counting, yet the two textbook methods fail at scale: the Pascal recurrence costs $\Theta(NR)$ time, and the factorial-table method costs $\Theta(N)$ memory, requires a prime modulus, and requires $N < m$. We present a complete, provably correct engine for $\binom{N}{R} \bmod m$ for arbitrary $m < 2^{62}$ (composite moduli included) and $N < 2^{64}$. The engine factors $m = \prod p_i^{k_i}$, evaluates each residue through the decomposition $\binom{N}{R} \equiv p^e F_p(N) F_p(R)^{-1} F_p(N-R)^{-1} \pmod{p^k}$ —where F_p is the p -free part of the factorial and e is the Kummer carry count—and recombines by the Chinese Remainder Theorem. A periodicity lemma derived from Gauss’s generalization of Wilson’s theorem compresses the state space of Pascal’s triangle modulo p^k to p^k prefix products, yielding two regimes: a *table regime* with $O(\log_p N)$ multiplications per query after $O(\min(p^k, N_{\max}))$ preprocessing, and a *direct regime* with $O(R + \log_p N)$ multiplications for $R < p^k$ and $O(1)$ memory. The hot loops are division-free: Montgomery arithmetic exploits the identity $\bar{i} + 1 = \bar{i+1}$ so that consecutive-integer products cost one multiply and one add; moduli 2^k reduce by a bit-mask. Four modular-inverse strategies are compared, including Fermat–Hensel lifting with $\lceil \log_2 k \rceil$ Newton steps and an inverse-prefix table built with a single inversion. Correctness is established by proofs and validated by 3,139,684 oracle checks plus 3,004 exact big-integer comparisons; results are bit-identical on two platforms (Windows/GCC 16/12th-gen Intel and Linux/GCC 11/Xeon). Measured per-query costs at $N \approx 10^{18}$ are 1.4–6.9 μ s for composite m and scale linearly in $\log N$ at 15–36 ns per base- p digit; Montgomery arithmetic is 2.0–3.1 $\times$ faster than 128-bit modulo; Hensel inversion is up to 20 $\times$ faster than the extended Euclidean algorithm at 2^{40} ; and the Kummer short-circuit resolves 95–99.998% of uniformly random queries in $O(\log N)$ time, a rate predicted exactly by Fine’s theorem. We also report the honest wall: for a prime $p \approx 10^9$ with $R \approx N/2$, no table fits in memory and a query costs 3.4–4.5 s.

Index Terms—binomial coefficients, Lucas’ theorem, Kummer’s theorem, prime-power moduli, Chinese Remainder Theorem, Montgomery multiplication, modular inverse, Hensel lifting, algorithm engineering

I. INTRODUCTION

The residue $\binom{N}{R} \bmod m$ appears wherever counting is done in modular arithmetic—lattice-path enumeration, inclusion–exclusion sums, coefficient extraction in polynomial identities, and the inner loops of dynamic programs over combinatorial

structures. Three techniques constitute the standard toolkit, and each hits a wall well before $N = 10^9$:

- 1) **Pascal’s recurrence** $\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}$ costs $\Theta(NR)$ time. At $N = 10^9$, $R = N/2$ that is 5×10^{17} additions—several CPU-years (Section VI-F).
- 2) **Factorial tables with Fermat inverses**, $\binom{N}{R} \equiv N! (R!)^{-1} ((N-R)!)^{-1}$, cost $\Theta(N)$ time and memory and require a prime modulus with $N < m$. At $N = 10^9$ the table alone is 4–8 GB.
- 3) **Lucas’ theorem** [1] handles $N \geq p$ for prime p by reducing to base- p digits, but says nothing about composite moduli or prime powers.

Composite moduli are the difficult case. Writing $m = \prod p_i^{k_i}$ and recombining by the Chinese Remainder Theorem is straightforward; computing $\binom{N}{R} \bmod p^k$ for $k \geq 2$ is not. Factorials are not invertible modulo p^k , Wilson’s theorem changes form for 2^k , and the exponent of p dividing the binomial coefficient must be tracked exactly. The mathematics is classical—Kummer [2], Legendre [3], Gauss [6], and, for prime powers, Davis–Webb [4] and Granville [5]—but a *unified, measured, memory-aware* implementation covering all of m with proven cost bounds is, to our knowledge, not available in the literature.

This paper makes the following contributions.

- **A unified algorithm** (Theorem 5) for $\binom{N}{R} \bmod m$, arbitrary $m < 2^{62}$, $N < 2^{64}$, that reduces exactly to Lucas’ theorem when $k = 1$ and to the classical factorial table when $N < p$, with the Kummer criterion used as an $O(\log_p N)$ short-circuit that decides $p^k \mid \binom{N}{R}$ before any multiplication.
- **Two complementary regimes with proven bounds** (Theorem 7): a table regime costing $O(\log_p N)$ multiplications per query after tabulating $\min(p^k, N_{\max} + 1)$ prefix products, and a direct regime costing $O(R + \log_p N)$ multiplications for $R < p^k$ with $O(1)$ memory, enabled by a range identity (Lemma 6) that makes the cost sensitive to R rather than to N .
- **Division-free arithmetic engineering**: Montgomery

multiplication [8] with an incremental-form identity that turns products of consecutive integers into one multiply and one add per unit; bit-mask reduction for 2^k ; 32-bit table storage when $p^k < 2^{32}$; and four modular-inverse strategies, including Fermat–Hensel lifting and an inverse-prefix table constructed with one inversion via Montgomery’s batch trick [9].

- **A reproducible two-platform experimental study** (five experiments, deterministic seeds, machine-independent checksums) and a **validation protocol** of 3.1 million oracle checks against Pascal’s triangle, Lucas’ theorem, Gauss’s theorem, and exact big-integer arithmetic.
- **An honest characterization of the limits:** the p^k memory/time wall, scheduler-induced measurement anomalies, and the design choices whose benefit is platform-dependent.

The scope is deliberately narrow: pure algorithmic number theory and its engineering. No new theorem is claimed; the contribution is the synthesis, the proofs of cost, the implementation, and the measurements.

II. RELATED WORK

Classical congruences. Lucas [1] proved that for prime p , $\binom{N}{R} \equiv \prod_i \binom{N_i}{R_i} \pmod{p}$ where N_i, R_i are base- p digits. Kummer [2] showed that the exponent of p in $\binom{N}{R}$ equals the number of carries when adding R and $N - R$ in base p , which follows from Legendre’s formula $v_p(x!) = \sum_{j \geq 1} \lfloor x/p^j \rfloor$ [3]. Fine [24] counted the binomial coefficients in row N that are non-zero modulo p as $\prod_i (N_i + 1)$; we use his theorem to predict the measured frequency of the Kummer short-circuit (Section VI-B). Babbage [7] proved $\binom{2p-1}{p-1} \equiv 1 \pmod{p^2}$, the first prime-power congruence, which we use as a large-modulus test vector in the form $\binom{p^2}{p} \equiv p \pmod{p^2}$.

Prime-power moduli. Davis and Webb [4] gave a Lucas-type theorem modulo p^k ; Granville [5] gave the definitive treatment of binomial coefficients modulo prime powers, expressing $\binom{N}{R} \pmod{p^k}$ through products of p -free factorials of digit blocks and the generalized Wilson theorem. Our Theorem 5 is the computational form of this theory; Gauss’s generalization of Wilson’s theorem [6, art. 78] supplies the sign W_{p^k} that appears in our periodicity lemma. Howard [25] studied the number of binomial coefficients divisible by a fixed power of a prime, the $k \geq 2$ analogue of Fine’s count.

Modular arithmetic. Montgomery’s reduction [8] replaces division by a modulus with multiplication by a precomputed constant and a shift; his simultaneous-inversion trick [9] computes n inverses with one inversion and $3n$ multiplications, which we specialize to prefix products. The average number of division steps of Euclid’s algorithm, $\approx (12 \ln 2 / \pi^2) \ln n$, is analysed by Knuth [10, §4.5.3]; Newton/Hensel lifting of inverses is treated by von zur Gathen and Gerhard [11] and Shoup [12].

Factorization and primality. Miller [13] and Rabin [14] introduced the strong-pseudoprime test; Sorenson and Webster [15] proved that the twelve prime bases up to 37 give a deterministic test below 3.3×10^{24} , covering all 64-bit integers.

Pollard’s rho method [16] with Brent’s cycle detection [17] factors the cofactor left by trial division. Garner [18] gave the mixed-radix formulation of the Chinese Remainder Theorem.

Sub-linear factorials. Strassen [19] and Pollard [21] showed that $N! \pmod{p}$ can be computed in $O(\sqrt{N} \text{ polylog})$ ring operations by splitting the product into $\sqrt{N}$ polynomial evaluations; Bostan, Gaudry, and Schost [20] gave the best-known algorithm. This is the only known route past the p^k wall reported in Section VIII and is our principal item of future work.

Practice. The competitive-programming literature [23] documents the factorial-table method and Lucas’ theorem as folklore, typically for a fixed prime such as $10^9 + 7$. We are not aware of a published implementation for arbitrary composite m with measured, regime-aware behaviour, which is the gap this paper fills.

III. MATHEMATICAL FOUNDATIONS

Throughout, p is a prime, $k \geq 1$, $n = p^k$, and $0 \leq R \leq N$. We write $v_p(x)$ for the exponent of p in x , $s_p(x)$ for the base- p digit sum of x , and

$$U_p(y) = \prod_{\substack{1 \leq i \leq y \\ p \nmid i}} i, \quad F_p(x) = \frac{x!}{p^{v_p(x!)}} \\ W_n = \prod_{u \in (\mathbb{Z}/n\mathbb{Z})^\times} u \pmod{n}.$$

$F_p(x)$ is the p -free part of $x!$ and W_n is the product of all units modulo n .

Theorem 1 (Legendre [3]): $v_p(x!) = \sum_{j \geq 1} \lfloor x/p^j \rfloor = \frac{x - s_p(x)}{p - 1}$.

Theorem 2 (Kummer [2]): $v_p \binom{N}{R} = \frac{s_p(R) + s_p(N - R) - s_p(N)}{p - 1}$, which equals the number of carries when R and $N - R$ are added in base p .

Proof sketch: Apply Theorem 1 to $N!$, $R!$, $(N - R)!$. Each carry in the base- p addition reduces the digit sum by exactly $p - 1$. ■

In algorithmic form: with $x_j = \lfloor N/p^j \rfloor$, $y_j = \lfloor R/p^j \rfloor$, $z_j = \lfloor (N - R)/p^j \rfloor$, the quantity $x_j - y_j - z_j \in \{0, 1\}$ is the carry out of digit $j - 1$, and $e = \sum_{j \geq 1} (x_j - y_j - z_j)$.

Theorem 3 (Gauss [6]): $W_n \equiv -1 \pmod{n}$ if $n = 4$, $n = p^k$, or $n = 2p^k$ for an odd prime p (and trivially for $n = 2$); otherwise $W_n \equiv 1 \pmod{n}$.

Corollary 3.1: For $n = p^k$: $W_n = -1$ if p is odd or $n \in \{2, 4\}$; $W_n = +1$ if $n = 2^k$ with $k \geq 3$.

Proof sketch: Pair each unit with its inverse; the product of all units equals the product of the self-inverse units, the solutions of $u^2 \equiv 1$. For cyclic $(\mathbb{Z}/p^k\mathbb{Z})^\times$ (p odd) these are ± 1 , with product -1 . For 2^k , $k \geq 3$, the group is $C_2 \times C_{2^{k-2}}$ and the four solutions $\pm 1, \pm(1 + 2^{k-1})$ have product 1. ■

Lemma 4 (Periodicity of unit prefix products): For $n = p^k$ and every $y \geq 0$,

$$U_p(y) \equiv W_n^{\lfloor y/n \rfloor} U_p(y \pmod{n}) \pmod{n}. \quad (1)$$

Proof: Partition $\{1, \dots, y\}$ into $q = \lfloor y/n \rfloor$ complete blocks $B_t = \{tn+1, \dots, (t+1)n\}$ and the tail $\{qn+1, \dots, y\}$. Reduction modulo n maps each B_t bijectively onto $\{1, \dots, n\}$ (with $(t+1)n \mapsto n \equiv 0$), and since $p \mid n$, $p \nmid i \iff p \nmid (i \bmod n)$. Hence the p -free elements of B_t reduce to exactly the units modulo n , whose product is W_n . The tail reduces to $\{1, \dots, y - qn\}$ and contributes $U_p(y \bmod n)$. ■

Theorem 5 (Prime-power decomposition): Let $n = p^k$ and $e = v_p\left(\frac{N}{R}\right)$. Then

- (a) $F_p(x) = \prod_{j \geq 0} U_p(\lfloor x/p^j \rfloor)$;
- (b) $F_p(x)$ is a unit modulo n ;
- (c) $\left(\frac{N}{R}\right) \equiv p^e F_p(N) F_p(R)^{-1} F_p(N-R)^{-1} \pmod{n}$;
- (d) $\left(\frac{N}{R}\right) \equiv 0 \pmod{n}$ if and only if $e \geq k$.

Proof: (a) Separate the multiples of p in $x!$:

$$x! = \left(\prod_{i \leq x, p \nmid i} i \right) \prod_{t=1}^{\lfloor x/p \rfloor} (pt) = U_p(x) p^{\lfloor x/p \rfloor} \lfloor x/p \rfloor!.$$

Iterating on $\lfloor x/p \rfloor!$ collects $p^{\sum_{j \geq 1} \lfloor x/p^j \rfloor} = p^{v_p(x!)}$ by Theorem 1, and the remaining factor is $\prod_{j \geq 0} U_p(\lfloor x/p^j \rfloor)$. (b) Each $U_p(\cdot)$ is a product of integers coprime to p . (c) $\left(\frac{N}{R}\right) = N!/(R!(N-R)!) = p^{v_p(N!) - v_p(R!) - v_p((N-R)!)} F_p(N)/(F_p(R)F_p(N-R))$; the exponent is e by Theorem 1, and the quotient of p -free parts is an integer coprime to p (it equals $\left(\frac{N}{R}\right)p^{-e}$), so it may be evaluated modulo n with the inverses guaranteed by (b). (d) $p^e \cdot (\text{unit}) \equiv 0 \pmod{p^k}$ iff $e \geq k$. ■

Corollary 5.1 (Lucas' theorem recovered): For $k = 1$, $U_p(y \bmod p) = (y \bmod p)!$ and $W_p = -1$. If $e = 0$ the sign factors $W_p^{x_{j+1}}, W_p^{-y_{j+1}}, W_p^{-z_{j+1}}$ cancel level by level (no carries), and (c) becomes $\left(\frac{N}{R}\right) \equiv \prod_j \binom{N_j}{R_j} \pmod{p}$; if $e \geq 1$ both sides vanish.

Lemma 6 (Range identity): For $0 \leq R \leq N$,

$$\frac{F_p(N)}{F_p(N-R)} = \prod_{j \geq 0} \prod_{\substack{\lfloor (N-R)/p^j \rfloor < i \leq \lfloor N/p^j \rfloor \\ p \nmid i}} i, \quad (2)$$

an integer coprime to p .

Proof: From Theorem 5(a), $F_p(N)/F_p(N-R) = \prod_j U_p(\lfloor N/p^j \rfloor)/U_p(\lfloor (N-R)/p^j \rfloor)$ and $U_p(b)/U_p(a) = \prod_{a < i \leq b, p \nmid i} i$ for $a \leq b$. ■

The range at level j has length $\lfloor N/p^j \rfloor - \lfloor (N-R)/p^j \rfloor \in \{\lfloor R/p^j \rfloor, \lfloor R/p^j \rfloor + 1\}$; this is what makes the direct regime's cost depend on R rather than on N .

Chinese Remainder Theorem. For pairwise coprime $n_1, \dots, n_t$ with $m = \prod n_i$ and $M_i = m/n_i$, the unique $x \bmod m$ with $x \equiv x_i \pmod{n_i}$ is

$$x \equiv \sum_{i=1}^t x_i M_i (M_i^{-1} \bmod n_i) \pmod{m}. \quad (3)$$

IV. ALGORITHM AND IMPLEMENTATION

A. Pipeline

Given m , the engine (i) factors $m = \prod_{i=1}^t p_i^{k_i}$, (ii) instantiates one *prime-power solver* per $p_i^{k_i}$, choosing an

Algorithm 1 Digit walk (Legendre/Kummer)

Input: N, R with $R \leq N - R$; prime p ; exponent k

```

1:  $L \leftarrow 0$ ;  $e \leftarrow 0$ ;  $x \leftarrow N$ ;  $y \leftarrow R$ ;  $z \leftarrow N - R$ 
2: while  $x > 0$  do
3:    $a[L] \leftarrow x$ ;  $b[L] \leftarrow y$ ;  $c[L] \leftarrow z$ ;  $L \leftarrow L + 1$ 
4:    $x \leftarrow \lfloor x/p \rfloor$ ;  $y \leftarrow \lfloor y/p \rfloor$ ;  $z \leftarrow \lfloor z/p \rfloor$ 
5:    $e \leftarrow e + (x - y - z) \quad \triangleright$  carry out of digit  $L - 1$ ,
    $\in \{0, 1\}$ 
6:   if  $e \geq k$  then return ZERO  $\triangleright p^k \mid \left(\frac{N}{R}\right)$ : short-circuit
7:   end if
8: end while
9: return  $(a, b, c, L, e)$ 
```

arithmetic backend and a regime, and (iii) precomputes the CRT coefficients $M_i^{-1} \bmod p_i^{k_i}$ by the extended Euclidean algorithm. A query (N, R) is answered by t solver calls and one CRT recombination.

B. Factorization Front End

Trial division by 2 and odd $d \leq 10^6$ removes all prime factors up to 10^6 . The cofactor, if greater than one, is either prime or has all prime factors above 10^6 ; it is tested by the deterministic Miller–Rabin test with the twelve bases 2, 3, ..., 37 [13]–[15] and, if composite, split by Pollard–Brent rho [16], [17] in expected $O(m^{1/4})$ multiplications. For the moduli in this study, factorization took 2–120 μ s.

C. The Digit Walk and the Kummer Short-Circuit

Every query first enforces $R \leftarrow \min(R, N - R)$ and then executes Algorithm 1, which computes the three base- p quotient chains and the exponent e of Theorem 2 simultaneously, aborting as soon as $e \geq k$. The walk costs $L = \lfloor \log_p N \rfloor + 1$ iterations, each with three integer divisions by p (shifts when $p = 2$). Because the exponent is accumulated from the least significant digit upward, the short-circuit fires as early as the k -th carry.

D. Regime A — Prefix Table

For a prime power $n = p^k$ and a bound $N_{\max}$ on the queries, the solver tabulates $T[r] = U_p(r) \bmod n$ for $0 \leq r < S$, where $S = \min(n, N_{\max} + 1)$. By Lemma 4, $U_p(y) \equiv W_n^{\lfloor y/n \rfloor} T[y \bmod n]$ for every y , so with Theorem 5(a),

$$F_p(x) \equiv \prod_{j=0}^{L-1} W_n^{\lfloor x_j/n \rfloor} T[x_j \bmod n] \pmod{n}, \quad (4)$$

where $x_j = \lfloor x/p^j \rfloor$. Since $W_n = \pm 1$ (Corollary 3.1), the sign is a parity test on $\lfloor x_j/n \rfloor$ —no exponentiation. When $N_{\max} < n$ the table is truncated to $N_{\max} + 1$ entries and all indices satisfy $x_j \leq N \leq N_{\max}$; in that case, for $k = 1$, $T[r] = r! \bmod p$ and the solver is the classical factorial table.

The implementation processes runs of $p - 1$ consecutive units between multiples of p , so no divisibility test is executed per element. The inverse table uses $T[r-1]^{-1} = T[r]^{-1} \cdot r$ for $p \nmid r$, i.e., Montgomery's batch-inversion trick [9] specialised to prefix products.

Algorithm 2 Table construction (backend form; $\otimes, \oplus$ are the backend's multiply and add)

```

1:  $T[0] \leftarrow \bar{1}; acc \leftarrow \bar{1}; \bar{i} \leftarrow \bar{1} \quad \triangleright \bar{i} = \text{form}(i) \text{ for } i = 1$ 
2: for  $i \leftarrow 1$  to  $S - 1$  do
3:   if  $p \nmid i$  then  $acc \leftarrow acc \otimes \bar{i}$ 
4:   end if
5:    $T[i] \leftarrow acc$ 
6:    $\bar{i} \leftarrow \bar{i} \oplus \bar{1} \quad \triangleright \text{form}(i + 1) = \text{form}(i) + \text{form}(1)$ 
7: end for
8: optional inverse table (one inversion in total):
9:  $TI[S - 1] \leftarrow acc^{-1}$ 
10: for  $r \leftarrow S - 1$  down to  $1$  do
11:    $TI[r - 1] \leftarrow (p \nmid r) ? TI[r] \otimes \bar{r} : TI[r]$ 
12: end for

```

Algorithm 3 Range product $\prod_{a < i \leq b, p \nmid i} i \bmod n$

```

1: if  $b \leq a$  then return  $\bar{1}$ 
2: end if
3:  $q \leftarrow \lfloor (b - a)/n \rfloor; acc \leftarrow W_n^q; i \leftarrow a + qn + 1 \quad \triangleright q$ 
   complete cycles
4:  $\bar{i} \leftarrow \text{form}(i \bmod n)$ 
5: while  $i \leq b$  do
6:   if  $p \nmid i$  then  $acc \leftarrow acc \otimes \bar{i}$ 
7:   end if
8:    $\bar{i} \leftarrow \bar{i} \oplus \bar{1}; i \leftarrow i + 1 \triangleright$  wraps modulo  $n$  automatically
9: end while
10: return  $acc$ 

```

Query. After Algorithm 1 returns, the solver computes

$$\text{num} = \prod_{j < L} \hat{T}(a_j), \quad \text{den} = \prod_{j < L} \hat{T}(b_j) \hat{T}(c_j),$$

$$\binom{N}{R} \equiv p^e \text{num} \cdot \text{den}^{-1} \pmod{n},$$

where $\hat{T}(y) = W_n^{\lfloor y/n \rfloor} T[y \bmod n]$. With the inverse table, den^{-1} is accumulated directly from $\hat{T}(\cdot)$ and no inversion is performed.

E. Regime B — Direct Range Products

Without a table, the solver evaluates Lemma 6 and $F_p(R) = \prod_j U_p(b_j)$ by Algorithm 3. The remaining range after the complete cycles has length $< n$, so Algorithm 3 costs $\min(b - a, n) + O(1)$ multiplications. The query is

$$\text{num} = \prod_{j < L} \text{Range}(c_j, a_j), \quad \text{den} = \prod_{j < L} \text{Range}(0, b_j),$$

$$\binom{N}{R} \equiv p^e \text{num} \cdot \text{den}^{-1} \pmod{n}.$$

F. Cost Bounds

Theorem 7 (Cost): Let $n = p^k$, $R \leq N/2$, $L = \lfloor \log_p N \rfloor + 1$, and count backend multiplications.

(A) *Table regime* with $S = \min(n, N_{\max} + 1)$: preprocessing costs S multiplications and S words ($2S$ with the inverse

table). A query costs $3L$ table reads, $3L$ multiplications, $3L$ integer divisions by p , and one modular inversion (none with the inverse table), plus $O(1)$.

(B) *Direct regime*: $O(1)$ words. A query costs

$$T_B(N, R) = O\left(L + \sum_{j=0}^{L-1} \min(\lfloor R/p^j \rfloor + 1, n)\right), \quad (5)$$

which is $O(L + R)$ when $R < n$ and $O(L + n(1 + \log_p(R/n)))$ when $R \geq n$, plus one inversion.

Proof: (A) is immediate from Section IV-D. (B) By Lemma 6 the numerator range at level j has length $\lfloor R/p^j \rfloor$ or $\lfloor R/p^j \rfloor + 1$ and the denominator range has length $\lfloor R/p^j \rfloor$; Algorithm 3 costs $\min(\text{length}, n) + O(1)$ for each. If $R < n$ the lengths sum to $\leq 2Rp/(p-1) + 2L \leq 4R + 2L$. If $R \geq n$, the $\lfloor \log_p(R/n) \rfloor + 1$ levels with $R/p^j \geq n$ cost n each and the remaining levels form a geometric series bounded by $2np/(p-1)$. ■

The preprocessing of Regime A and the worst case of Regime B are therefore both $\Theta(n)$ multiplications for a single prime power; Section VI-C measures their ratio.

G. Arithmetic Backends

Three backends implement the interface $\{\otimes, \oplus, \ominus, \text{form}, \text{form}^{-1}\}$ and are selected per prime power at compile time (templates).

Montgomery (odd $n < 2^{63}$). Values are kept in form $\bar{a} = a \cdot 2^{64} \bmod n$. With $n' = -n^{-1} \bmod 2^{64}$,

$$\text{REDC}(t) = \frac{t + ((t \bmod 2^{64}) n' \bmod 2^{64}) n}{2^{64}}, \quad (6)$$

which satisfies $\text{REDC}(t) \equiv t \cdot 2^{-64} \pmod{n}$; then $\bar{a} \otimes \bar{b} = \text{REDC}(\bar{a} \bar{b})$, followed by one conditional subtraction; for $n < 2^{63}$ and $\bar{a}, \bar{b} < n$ the intermediate sum is below 2^{128} [8]. The constant n' is obtained from $x_0 = 1$ by six Newton steps $x \leftarrow x(2 - nx) \bmod 2^{64}$, each doubling the number of correct bits. The key engineering identity is that the form is additive: $\bar{i} + \bar{1} = \bar{i} \oplus \bar{1}$. Algorithms 2 and 3 therefore never convert integers into Montgomery form inside the loop; each unit costs one REDC multiply and one modular add, and the loop contains no division instruction.

Bit-mask ($n = 2^k$, $k \leq 61$). Reduction is AND with $2^k - 1$; unsigned wrap-around makes $(a \cdot b) \bmod 2^{64}$ exact modulo 2^k . Division by $p = 2$ in Algorithm 1 becomes a shift.

Naive (baseline). $(a \cdot b) \bmod n$ computed with a 128-bit product and a 128-by-64-bit modulo (compiled to the `__umodti3` runtime routine).

Table entries are stored as 32-bit words when $n < 2^{32}$, halving memory; Montgomery-form values are bounded by n , so no information is lost.

H. Modular-Inverse Strategies

One inversion per prime power per query is needed for den^{-1} . Four strategies are implemented and compared.

1) **Euler**. $a^{-1} = a^{\varphi(n)-1}$ with $\varphi(p^k) = p^{k-1}(p-1)$: $\approx 1.5 \log_2 n$ multiplications by square-and-multiply.

- 2) **Fermat–Hensel.** $x_0 = a^{p-2} \bmod p$ ($\approx 1.5 \log_2 p$ multiplications in a Montgomery context modulo p), then $\lceil \log_2 k \rceil$ Newton steps $x \leftarrow x(2 - ax) \bmod n$ (two multiplications each). Correctness: if $ax \equiv 1 \pmod{p^j}$ then $ax(2 - ax) = 1 - (1 - ax)^2 \equiv 1 \pmod{p^{2j}}$. For $k = 1$ this degenerates to Fermat’s inverse and coincides with strategy 1.
- 3) **Extended Euclid.** $\approx 0.843 \ln n$ division steps on average [10, §4.5.3]; requires leaving and re-entering Montgomery form (two multiplications).
- 4) **Inverse-prefix table** (Regime A only). Zero inversions per query; one inversion and S multiplications at build time; S extra words.

I. Recombination and Special Cases

The residues are combined by the CRT formula of Section III with precomputed $y_i = M_i^{-1} \bmod n_i$; each term $(x_i y_i \bmod n_i) M_i$ is below $m < 2^{62}$, so the sum needs no 128-bit arithmetic. Edge cases: $R > N \Rightarrow 0$; $m = 1 \Rightarrow 0$; $R \in \{0, N\} \Rightarrow 1 \bmod m$.

J. Validation Protocol

Correctness was established by proof (Section III) and tested against independent oracles, on all nine combinations of regime, inverse strategy, and backend: (i) exact Pascal rows modulo m for $N \leq 130$ and 40 moduli including 2^{40} , 3^{20} , 5^{16} , $10^{18} + 3$ and the 62-bit prime $2^{62} - 57$ (3,112,920 checks); (ii) cross-regime agreement on random (N, R) with N up to 1.8×10^{19} (20,000 checks); (iii) Lucas’ theorem evaluated independently for eleven primes (6,600 checks); (iv) Gauss’s theorem checked against the tabulated $T[n - 1]$ for 68 prime powers; (v) 96 hand-verified constants including $\binom{p^2}{p} \equiv p \pmod{p^2}$ for $p = 10^6 + 3$ [5], [7]; and (vi) 3,004 comparisons against Python’s exact big-integer `math.comb` for random moduli up to 10^{12} , $N \leq 10^5$ with arbitrary R , and $N \leq 1.8 \times 10^{19}$ with $R \leq 60$. All 3,142,688 checks passed on both platforms. In addition, every benchmark run emits a 64-bit checksum of its results; all 71 + 30 checksums agree across the two platforms.

V. EXPERIMENTAL METHODOLOGY

A. Platforms

Table I lists the two platforms. All code is single-threaded C++17; timings use `std::chrono::steady_clock`. Each configuration was run in a fresh process.

B. Query Generation and the Kummer Short-Circuit

Queries are generated by a seeded SplitMix64 generator (seed 20260916) with $N \sim U[N_{\min}, N_{\max}]$ and one of four R -selection modes:

- **random:** $R \sim U[0, N]$ —the realistic workload;
- **small:** $R \sim U[0, 1000]$;
- **fullpath:** $R = 0$ —the unique R that produces *no* carry in *any* base, so every prime-power solver executes its complete path. In Regime A the arithmetic is identical

TABLE I
EXPERIMENTAL PLATFORMS

	Platform A (primary)	Platform B (replication)
CPU	Intel 12th-gen Core, Alder Lake (CPUID family 6, model 154), hybrid P/E cores, 12 logical CPUs	Intel Xeon @ 2.90 GHz (Ice Lake-SP class), 2 vCPUs
OS	Windows 11 (build 26200)	Linux 6.18 (Amazon Linux 2023)
Compiler	GCC 16.1.0 (MSYS2), <code>-O3 -march=native -std=c++17</code>	GCC 11.5.0, same flags
Python	3.13.5	3.9.25
Memory metric	table bytes (exact); peak RSS unavailable	table bytes and peak RSS

for every R that does not short-circuit (Theorem 7(A)), so this mode measures the R -independent upper bound;

- **worst:** for a prime p and $N \geq p$, R is built digit-wise as $R_0 = N_0$ and $R_j = \lfloor N_j/2 \rfloor$ ($j \geq 1$) whenever some N_j ($j \geq 1$) is odd (which guarantees $R < N/2$ and no carry); this maximises the range lengths of Algorithm 3. For $N < p$, $R = \lfloor N/2 \rfloor$.

The distinction matters because, with uniformly random R , the expected number of base- p carries grows with the number of digits, so the Kummer short-circuit terminates the overwhelming majority of prime-power solves at $N \approx 10^{18}$ (Section VI-B). Every run reports the number of short-circuited prime-power solves and the number of zero results; a preliminary version of this study that lacked these counters was misinterpreting short-circuits as arithmetic speed.

C. Experiments

E1 — Scaling in N . Composite $m \in \{720720 = 2^4 3^2 5 \cdot 7 \cdot 11 \cdot 13, 10^9 = 2^9 5^9, 53,343,360,000 = 2^{10} 3^5 5^4 7^3\}$, $N \sim U[N/2, N]$ for $N = 10^6, \dots, 10^{18}$, 2×10^5 queries, modes fullpath and random. **E2 — Regime crossover.** Prime $m = p \in \{1009, 10007, 100003, 1000003, 10^7+19, 10^8+7, 10^9+7\}$ at $N = 10^9$: table build time and memory, table query time (fullpath), direct worst-case query time, direct random query time, and small- R direct time at $N = 10^9$ and $N \sim U[10^{17}, 10^{18}]$. **E3 — Modular inverses.** (a) Micro-benchmark: 10^6 random units for ten prime powers, three strategies, with $a \cdot a^{-1} \equiv 1$ verification. (b) End-to-end: Regime A, $N \sim U[10^{17}, 10^{18}]$, 5×10^5 fullpath queries, four strategies. **E4 — Reduction strategy.** Montgomery/bit-mask versus the naive backend on table construction (arithmetic loop timed separately from allocation and page faults) and on the direct worst case. **E5 — Baseline.** The $O(NR)$ Pascal recurrence for $\binom{N}{N/2} \bmod 720720$.

The complete grid runs in 73 s (A) and 53 s (B) from a single command; the harness, seeds, and expected checksums are released with the code.

TABLE II
VALIDATION SUMMARY (IDENTICAL ON BOTH PLATFORMS)

Suite / oracle	Checks	Failures
Pascal rows, $N \leq 130$, 40 moduli, 9 configurations	3,112,920	0
Cross-regime consistency, $N \leq 1.8 \times 10^{19}$	20,000	0
Lucas' theorem, 11 primes, $N \leq 10^{18}$	6,600	0
Gauss's generalized Wilson theorem	68	0
Known constants (incl. Babbage congruence)	96	0
Python <code>math.comb</code> , random $m \leq 10^{12}$	3,004	0
Cross-platform result checksums	101	0

TABLE III
E1: FULL-PATH COST (FULLPATH MODE, μs PER QUERY)

m	$N_{\max}$	L	Platform A	Platform B
720720	10^6	61	1.568	1.559
720720	10^9	90	2.165	2.183
720720	10^{12}	120	2.747	2.833
720720	10^{15}	151	3.561	3.491
720720	10^{18}	181	6.919 [†]	4.126
$10^9 = 2^9 5^9$	10^6	29	1.194	0.571
10^9	10^9	43	1.773	0.787
10^9	10^{12}	57	2.199	1.003
10^9	10^{15}	72	2.830	1.230
10^9	10^{18}	86	3.264	1.434
$2^{10} 3^5 5^4 7^3$	10^6	49	1.185	1.195
$2^{10} 3^5 5^4 7^3$	10^9	73	1.771	1.674
$2^{10} 3^5 5^4 7^3$	10^{12}	97	2.180	2.168
$2^{10} 3^5 5^4 7^3$	10^{15}	122	5.372 [†]	2.654
$2^{10} 3^5 5^4 7^3$	10^{18}	146	6.842 [†]	3.129

[†]Off-trend on Platform A; see Section VIII.

VI. RESULTS

A. Correctness

Table II summarises the validation of Section IV-J; every check passed on both platforms.

B. E1 — Scaling in N

Let $L(N) = \sum_i (\lfloor \log_{p_i} N \rfloor + 1)$ be the total number of base- p_i digits processed per query (Theorem 7(A)); values in Table III are at $N = 0.75 N_{\max}$, the mean of the query range.

Least-squares fits of $t = t_0 + cL$ give, on Platform B, $c = 21.4$ ns/digit and $t_0 = 0.26 \mu\text{s}$ for $m = 720720$; $c = 15.2$ ns/digit, $t_0 = 0.14 \mu\text{s}$ for $m = 10^9$; and $c = 19.9$ ns/digit, $t_0 = 0.22 \mu\text{s}$ for $m = 2^{10} 3^5 5^4 7^3$ —the $\Theta(\log N)$ scaling of Theorem 7(A) with a per-digit constant that is smaller when one of the two primes is 2 (shifts instead of divisions). On Platform A, excluding the three flagged rows, the fits are 21.9, 36.3, and 20.7 ns/digit; the model predicts 4.16, 2.73, and 3.23 μs for the flagged rows against the measured 6.92, 5.37, and 6.84 μs . The intercept t_0 is the fixed cost of the t inversions and the CRT recombination (six inversions for 720720).

With uniformly random R (Table IV), the per-query cost is *flat or decreasing* in N because the probability that fewer than k carries occur among $\lfloor \log_p N \rfloor + 1$ digits decays exponentially with the digit count; for $m = 720720$ (all $k_i \leq 4$) only

TABLE IV
E1: REALISTIC WORKLOAD (RANDOM MODE); FREQUENCIES ARE DETERMINISTIC AND IDENTICAL ON BOTH PLATFORMS

m	$N_{\max}$	A (μs)	B (μs)	Solves short-circuited	Zero results
720720	10^6	0.319	0.290	95.58 %	76.45 %
720720	10^9	0.296	0.249	99.37 %	96.25 %
720720	10^{12}	0.291	0.241	99.91 %	99.43 %
720720	10^{15}	0.290	0.239	99.986 %	99.92 %
720720	10^{18}	0.507 [†]	0.240	99.998 %	99.99 %
10^9	10^6	0.955	0.458	26.50 %	0.00 %
10^9	10^9	1.102	0.509	48.81 %	8.86 %
10^9	10^{12}	1.030	0.460	71.29 %	43.96 %
10^9	10^{15}	0.809	0.368	86.53 %	73.17 %
10^9	10^{18}	0.765	0.292	94.58 %	89.16 %
$2^{10} 3^5 5^4 7^3$	10^6	0.733	0.693	57.96 %	11.20 %
$2^{10} 3^5 5^4 7^3$	10^9	0.517	0.478	87.77 %	59.30 %
$2^{10} 3^5 5^4 7^3$	10^{12}	0.817	0.361	97.54 %	90.53 %
$2^{10} 3^5 5^4 7^3$	10^{15}	0.765	0.328	99.53 %	98.12 %
$2^{10} 3^5 5^4 7^3$	10^{18}	0.749	0.321	99.91 %	99.65 %

2×10^{-5} of prime-power solves reach the arithmetic at $N \approx 10^{18}$. For $m = 10^9$ the exponents $k = 9$ delay the effect. For prime moduli the frequency is predicted exactly by Fine's theorem [24]: row N of Pascal's triangle has $\prod_i (N_i + 1)$ entries not divisible by p , hence a uniformly random R escapes the short-circuit with probability $\prod_i (N_i + 1)/(N + 1)$. For $N = 10^9$ this predicts 6.71%, 4.85%, 70.00%, and 99.70% non-zero results for $p = 1009, 10007, 100003, 1000003$; the E2 random runs measured 6.4% (32/500), 4.4% (22/500), 68.0% (136/200), and 100% (50/50).

C. E2 — Regime Crossover for Prime Moduli at $N = 10^9$

Table V supports three observations. (i) The table query is essentially independent of p : 0.09–0.16 μs while the table fits in cache and 0.13–0.27 μs for the 400 MB table, where the three random reads per digit miss all caches. (ii) Building the table costs $\Theta(p)$ multiplications, and so does one worst-case direct query (Theorem 7); measured, the build costs $0.85 \times$ one worst-case query on Platform B at every scale (5.8/6.8, 58/68, 581/682) and 0.36 – $0.50 \times$ on Platform A. The table therefore pays for itself after a *single* worst-case query, or after ≈ 2 random- R queries at $p = 10^6 + 3$. (iii) At $p = 10^9 + 7$ the table would occupy 4 GB and was not built; the worst-case direct query executes $\approx 10^9$ Montgomery multiplications in 4.49 s (A) and 3.41 s (B), i.e., 4.5 and 3.4 ns per multiplication—the arithmetic runs at hardware speed, and the cost is intrinsic to the $\Theta(p^k)$ bound.

Table VI shows the cost identical at $N = 10^9$ and $N \approx 10^{18}$, confirming the $O(R + \log_p N)$ bound of Theorem 7(B): the $\log_p N$ term (two or three digits) is negligible next to $\approx 2R$ multiplications.

D. E3 — Modular Inverses

Table VII is consistent with the operation counts of Section IV-H. For 2^k , bit-mask arithmetic and Newton lifting make Hensel the fastest method by a wide margin ($20 \times / 16 \times$ over Euclid at 2^{40}). For odd p with large k ($5^9, 3^{20}$), Hensel replaces $1.5 \log_2 n$ Montgomery multiplications

TABLE V
E2: TABLE REGIME VERSUS DIRECT REGIME AT $N = 10^9$ (PLATFORM A / PLATFORM B)

p	Table entries (bytes)	Build, arith. (ms)	Build, alloc. (ms)	Table query (μ s)	Direct, worst-case	Direct, random R (zero results)
1009	1,009 (4.0 KB)	0.008 / 0.006	0.001 / 0.002	0.160 / 0.097	12.1 / 5.32 μ s	0.305 / 0.239 μ s (93.6%)
10007	10,007 (40 KB)	0.051 / 0.058	0.022 / 0.014	0.160 / 0.149	48.2 / 37.2 μ s	2.09 / 1.51 μ s (95.6%)
100003	100,003 (400 KB)	0.440 / 0.577	0.221 / 0.117	0.086 / 0.095	707 / 513 μ s	239 / 170 μ s (32.0%)
1000003	1,000,003 (4.0 MB)	4.47 / 5.81	0.92 / 1.26	0.108 / 0.115	8.96 / 6.81 ms	5.05 / 3.60 ms (0%)
10^7+19	10,000,019 (40 MB)	45.0 / 58.1	8.6 / 13.3	0.132 / 0.135	89.6 / 68.2 ms	—
10^8+7	100,000,007 (400 MB)	519 / 581	130 / 211	0.272 / 0.126	1460 / 682 ms	—
10^9+7	1,000,000,007 (4.0 GB), not built	—	—	—	4489 / 3411 ms	—

TABLE VI
E2: DIRECT REGIME, SMALL R ($R \leq 1000$), $p = 10^9 + 7$, 20,000 QUERIES

N	A (μ s)	B (μ s)	Short-circuits
10^9	4.65	3.59	0
$U[10^{17}, 10^{18}]$	4.61	3.59	0

TABLE VII
E3A: NS PER INVERSE OVER 10^6 RANDOM UNITS (A / B); BOLD MARKS THE FASTEST STRATEGY ON EACH PLATFORM

$n = p^k$	k	Euler $a^{\varphi(n)-1}$	Fermat–Hensel	Ext. Euclid
2^{10}	10	6.72 / 8.67	5.39 / 6.27	51.2 / 45.7
3^5	5	25.6 / 34.3	42.4 / 56.0	61.7 / 53.0
7^3	3	29.6 / 39.6	47.9 / 59.4	59.4 / 54.2
5^9	9	86.5 / 95.5	59.3 / 70.3	107.0 / 94.4
10^7+19	1	91.5 / 106.6	89.5 / 106.5	113.4 / 101.5
3^{20}	20	126.9 / 142.1	62.5 / 72.5	141.8 / 127.5
2^{40}	40	33.0 / 34.4	7.57 / 8.60	154.1 / 138.7
10^9+7	1	127.2 / 136.0	123.1 / 136.1	137.3 / 122.2
999,999,999,989	1	171.0 / 182.0	166.0 / 182.0	169.3 / 153.1
$(10^6+3)^2$	2	169.9 / 181.5	109.7 / 162.9	169.6 / 152.7

by $1.5 \log_2 p + 2 \lceil \log_2 k \rceil$ and wins by 1.4–2.0 $\times$; for small k with small p (3^5 , 7^3) the fixed overhead of the mod- p context and the conversions makes Euler faster. For $k = 1$, Hensel is Fermat, and the verdict between Fermat and Euclid is platform-dependent: Euclid is 10–16% faster on Platform B and 8–11% slower on Platform A. Every inverse was verified ($a \cdot a^{-1} \equiv 1$).

For moduli with many small prime powers (Table VIII) the inversions are a minor component ($\leq 15\%$) of a cost dominated by the $L \approx 150$ –180 digit steps, and the differences between strategies are within scheduler noise on Platform A. For a single large prime power, the inverse-prefix table removes the inversion entirely and reduces the per-query cost by 21–57% at the price of doubling the table (80 MB).

E. E4 — Reduction Strategy

On Platform A (Table IX) the naive backend is 3.0–3.1 $\times$ slower in every odd-modulus loop and 6.7 $\times$ slower than the bit-mask; on Platform B the direct loop shows a 2.0 $\times$ gap but the table-construction loop only 1.1 $\times$. The table-construction loop performs one store per element, and on Platform B it runs at 5.8 ns/element with Montgomery versus 3.4 ns/multiplication in the store-free direct loop—the build

TABLE VIII
E3B: END-TO-END EFFECT OF THE INVERSE STRATEGY (REGIME A, $N \sim U[10^{17}, 10^{18}]$, 5×10^5 FULLPATH QUERIES, μ S PER QUERY, A / B)

m	Inv./query	Euler	Hensel	Euclid	Inverse table
720720	6	5.33 / 3.96	4.73 / 4.04	5.69 / 4.08	5.60 / 3.87
$2^{10}3^55^47^3$	4	4.06 / 3.00	3.48 / 3.07	3.87 / 3.12	3.99 / 2.86
10^7+19	1	0.436 / 0.271	0.298 / 0.251	0.309 / 0.241	0.189 / 0.199

TABLE IX
E4: MONTGOMERY / BIT-MASK VERSUS NAIVE 128-BIT MODULO (A / B)

Workload	Montg./bit-mask	Naive %	Ratio
Build loop, $p = 10^6+3$ (ms)	4.42 / 5.81	13.72 / 6.41	3.10 / 1.10
Build loop, $p = 10^7+19$ (ms)	44.5 / 58.1	138.1 / 63.9	3.10 / 1.10
Build loop, $p = 10^8+7$ (ms)	459 / 581	1374 / 639	2.99 / 1.10
Build loop, $n = 2^{26}$ (ms)	68.8 / 74.2	458 / 214	6.66 / 2.88
Direct worst case, $p = 10^8+7$ (ms)	886 / 683	2717 / 1373	3.07 / 2.01

is store-bound there, which hides most of the arithmetic difference. Separating allocation from arithmetic was essential: page-faulting 400 MB costs 130–211 ms, comparable to a third of the arithmetic.

F. E5 — Baseline

The recurrence runs at 0.29–0.37 ns per addition (vectorised), so extrapolating to $N = 10^9$ gives 5×10^{17} additions, or 4.6–5.8 CPU-years, plus a 4 GB row; the engine answers the same query in $\approx 2 \mu$ s from a 244-byte table (Table X).

VII. DISCUSSION

Where the time goes. In Regime A, a query is a digit walk. The fitted 15–22 ns per base- p digit on Platform B decomposes into three 64-bit divisions by a run-time divisor ($\approx 3 \times 6$ ns on this core), three table reads and three Montgomery multiplications; the digit walk for $p = 2$ costs less because divisions become shifts. Replacing the divisions by multiplication with a precomputed reciprocal would remove the dominant term and is the most promising micro-optimisation left.

The short-circuit is the common case. For random queries the Kummer test is not an optimisation but the main event: at $N \approx 10^{18}$ it answers essentially every prime-power solve in $O(\log N)$ integer operations without touching the table. Any benchmark of binomial coefficients modulo small prime powers that samples R uniformly is, in effect, benchmarking

TABLE X
E5: $O(NR)$ PASCAL RECURRENCE VERSUS THE ENGINE,
 $\binom{N}{N/2} \bmod 720720$ (A / B)

N	Pascal (ms)	Engine (ms)	Speed-up
10^3	0.210 / 0.263	0.034 / 0.009	$6\times / 28\times$
10^4	17.7 / 14.6	0.027 / 0.008	$660\times / 1,800\times$
4×10^4	292.7 / 229.9	0.041 / 0.018	$7,100\times / 12,800\times$

carry counting; reporting the short-circuit frequency should be standard practice.

Choosing a regime. Theorem 7 and Table V give a simple rule: tabulate whenever $4 \min(p^k, N_{\max} + 1)$ bytes are available, because the table costs about one worst-case query to build and then answers queries in $\approx 0.1\text{--}0.3 \mu\text{s}$. Otherwise use the direct regime, whose cost is governed by $\min(R, N - R)$: microseconds for small R , seconds for $R \approx N/2$ at $p \approx 10^9$.

Classical methods as special cases. With $N_{\max} < p$ and $k = 1$, the truncated table $T[r] = r! \bmod p$ is the factorial table of the folklore method, and the Euler strategy is Fermat’s inverse; the engine’s $N_{\max}$ parameter therefore subsumes the classical method with the same memory. With $k = 1$ and no table, Algorithm 3 evaluates the digit binomials of Lucas’ theorem. Nothing is lost by using the general engine.

Platform dependence. The two platforms agree on every result bit and on every qualitative conclusion, but not on two quantitative ones: the Montgomery-versus-naive gap ($3.0\times$ vs $2.0\times$ in the direct loop) and the Fermat-versus-Euclid ordering for prime moduli, which flips. Both hinge on the relative cost of 64-bit division on the core (and, on Platform A, on which core type the scheduler chose). The robust conclusions are those that do not depend on division speed: Montgomery and bit-mask arithmetic never lose, and Hensel lifting wins decisively for 2^k and for large k .

VIII. LIMITATIONS

We list every limitation we are aware of, in decreasing order of importance.

- 1) **The p^k wall.** For a prime power $p^k > 10^8$ the table needs > 400 MB, and at $p^k \approx 10^9$ it does not fit (4 GB). The direct regime then costs $\Theta(\min(R, p^k))$ multiplications per level: 3.4–4.5 s for $p = 10^9 + 7$, $R \approx N/2$. This is not an implementation weakness; it is the $\Theta(p^k)$ bound of Theorem 7, and the only known way past it is sub-linear factorial evaluation [19]–[21], which we did not implement. For moduli with a prime-power factor near 10^{18} and large $\min(R, N - R)$, the engine is infeasible.
- 2) **Single-run measurements.** Each configuration was timed once (aggregating $1\text{--}5 \times 10^5$ queries where the per-query cost is small). No repetitions, confidence intervals, or core pinning were used. Platform A is a hybrid P/E-core laptop under Windows; three E1 rows (Table III, †) are 1.7–2.2 $\times$ off the linear trend that fits the other twelve rows, the E3b ordering for $m = 720720$ is not

monotone in the expected way, and the worst-case direct query at $p = 10^8 + 7$ took $2.1\times$ Platform B’s time while the $p = 10^9 + 7$ query took $1.3\times$. We attribute these to scheduling onto efficiency cores and report them unmodified.

- 3) **fullpath measures $R = 0$.** It is the only R that avoids the short-circuit for every prime simultaneously and its arithmetic path is identical to that of any non-short-circuited R in Regime A, but one of the three table reads per digit ($T[0]$) is always cache-resident. For large tables the fullpath figures are therefore slightly optimistic; for tables that fit in cache the effect is nil.
- 4) **Division-bound digit walk.** Per-digit cost is dominated by three 64-bit divisions by a run-time p . Precomputed-reciprocal division was not implemented.
- 5) **Memory reporting.** Table sizes are exact, but peak resident set size is only available on Platform B; on Windows it was not measured.
- 6) **Table construction is not arithmetic-bound on all cores.** On Platform B the build loop is store-bound (5.8 ns/element versus 3.4 ns/multiplication), so the Montgomery advantage in *construction* is only $1.1\times$ there; it is $3\times$ on Platform A. We do not know which behaviour is more typical.
- 7) **Hensel lifting has no advantage for $k = 1$** (it degenerates to Fermat) and loses to Euler for small k with small p because of fixed overheads.
- 8) **Range of m and N .** $m < 2^{62}$ (Montgomery requires $n < 2^{63}$ and the CRT sum must fit in 64 bits); $N < 2^{64} - 1$. Trial division to 10^6 plus Pollard–Brent rho handles all such m , but rho’s running time is probabilistic.
- 9) **Empirical, not formal, verification of the code.** The mathematics is proved; the implementation is tested—exhaustively for $N \leq 130$, by exact big integers for $N \leq 10^5$ (arbitrary R) and $N \leq 1.8 \times 10^{19}$ ($R \leq 60$), and by cross-regime consistency and Lucas’ theorem elsewhere. No machine-checked proof of the code exists.
- 10) **Single-threaded, no SIMD.** All numbers are single-core. Batch queries are embarrassingly parallel; we did not exploit this.
- 11) **Reduction alternatives not benchmarked.** Barrett reduction and a 32-bit-specialised Montgomery variant (for $p^k < 2^{32}$, where a $32 \times 32 \rightarrow 64$ multiply suffices) might narrow or widen the gaps of Table IX.

IX. CONCLUSION AND FUTURE WORK

We presented a complete engine for binomial coefficients modulo arbitrary composite moduli at $N \leq 10^{18}$, built on classical theorems of Legendre, Kummer, Gauss, Lucas, and Granville, with proven cost bounds, division-free arithmetic, and a validation protocol of 3.1 million checks. Two regimes cover the memory spectrum: a prefix table answering queries in $\Theta(\log N)$ steps—1.4–6.9 μs at $N \approx 10^{18}$ for the composite moduli studied—and a direct method whose cost follows $\min(R, N - R)$ rather than N . The experiments quantified

three effects that are easy to misjudge: the Kummer short-circuit dominates random workloads (predicted by Fine's theorem), a prefix table costs about as much as one worst-case direct query, and the benefit of Montgomery arithmetic and the choice of inversion algorithm depend on the cost of integer division on the target core—while Hensel lifting for 2^k and large k wins everywhere.

Future work, in order of expected impact: (i) sub-linear evaluation of p -free factorials modulo p^k by the Bostan–Gaudry–Schost method [20], which would replace the $\Theta(p^k)$ wall by $O(\sqrt{p^k} \text{ polylog})$ and make $p \approx 10^9$ moduli practical without tables; (ii) precomputed-reciprocal division in the digit walk; (iii) Barrett and 32-bit Montgomery variants; (iv) repeated, core-pinned measurements with confidence intervals on a wider set of microarchitectures; and (v) a machine-checked correctness proof of the prime-power solver.

Reproducibility. The engine (header-only C++17), the benchmark harness, the Python oracle, and the one-command driver that produced every number in this paper are released with deterministic seeds; the 101 result checksums allow any reader to confirm bit-exact agreement before comparing timings.

REFERENCES

- [1] É. Lucas, “Théorie des fonctions numériques simplement périodiques,” *Amer. J. Math.*, vol. 1, pp. 184–240, 289–321, 1878.
- [2] E. E. Kummer, “Über die Ergänzungssätze zu den allgemeinen Reciprocitätsgesetzen,” *J. Reine Angew. Math.*, vol. 44, pp. 93–146, 1852.
- [3] A.-M. Legendre, *Essai sur la théorie des nombres*, 2nd ed. Paris, France: Courcier, 1808.
- [4] K. S. Davis and W. A. Webb, “Lucas’ theorem for prime powers,” *European J. Combin.*, vol. 11, no. 3, pp. 229–233, 1990.
- [5] A. Granville, “Arithmetic properties of binomial coefficients I: Binomial coefficients modulo prime powers,” in *Organic Mathematics (Burnaby, BC, 1995)*, CMS Conf. Proc., vol. 20. Providence, RI, USA: Amer. Math. Soc., 1997, pp. 253–276.
- [6] C. F. Gauss, *Disquisitiones Arithmeticae*. Leipzig, Germany: Fleischer, 1801, art. 78.
- [7] C. Babbage, “Demonstration of a theorem relating to prime numbers,” *Edinburgh Philos. J.*, vol. 1, pp. 46–49, 1819.
- [8] P. L. Montgomery, “Modular multiplication without trial division,” *Math. Comp.*, vol. 44, no. 170, pp. 519–521, 1985.
- [9] P. L. Montgomery, “Speeding the Pollard and elliptic curve methods of factorization,” *Math. Comp.*, vol. 48, no. 177, pp. 243–264, 1987.
- [10] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 3rd ed. Reading, MA, USA: Addison-Wesley, 1997.
- [11] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2013.
- [12] V. Shoup, *A Computational Introduction to Number Theory and Algebra*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- [13] G. L. Miller, “Riemann’s hypothesis and tests for primality,” *J. Comput. Syst. Sci.*, vol. 13, no. 3, pp. 300–317, 1976.
- [14] M. O. Rabin, “Probabilistic algorithm for testing primality,” *J. Number Theory*, vol. 12, no. 1, pp. 128–138, 1980.
- [15] J. Sorenson and J. Webster, “Strong pseudoprimes to twelve prime bases,” *Math. Comp.*, vol. 86, no. 304, pp. 985–1003, 2017.
- [16] J. M. Pollard, “A Monte Carlo method for factorization,” *BIT*, vol. 15, no. 3, pp. 331–334, 1975.
- [17] R. P. Brent, “An improved Monte Carlo factorization algorithm,” *BIT*, vol. 20, no. 2, pp. 176–184, 1980.
- [18] H. L. Garner, “The residue number system,” *IRE Trans. Electron. Comput.*, vol. EC-8, no. 2, pp. 140–147, 1959.
- [19] V. Strassen, “Einige Resultate über Berechnungskomplexität,” *Jahresber. Deutsch. Math.-Verein.*, vol. 78, no. 1, pp. 1–8, 1976.
- [20] A. Bostan, P. Gaudry, and É. Schost, “Linear recurrences with polynomial coefficients and application to integer factorization and Cartier–Manin operator,” *SIAM J. Comput.*, vol. 36, no. 6, pp. 1777–1806, 2007.
- [21] J. M. Pollard, “Theorems on factorization and primality testing,” *Proc. Cambridge Philos. Soc.*, vol. 76, no. 3, pp. 521–528, 1974.
- [22] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, 6th ed. Oxford, U.K.: Oxford Univ. Press, 2008.
- [23] “Binomial coefficients,” *cp-algorithms*. [Online]. Available: <https://cp-algorithms.com/combinatorics/binomial-coefficients.html>
- [24] N. J. Fine, “Binomial coefficients modulo a prime,” *Amer. Math. Monthly*, vol. 54, no. 10, pp. 589–592, 1947.
- [25] F. T. Howard, “The number of binomial coefficients divisible by a fixed power of a prime,” *Proc. Amer. Math. Soc.*, vol. 37, no. 2, pp. 358–362, 1973.