Division by Two Cycles in Binary and Explicit Representations

Nuutti Kuosa

25 November 2026

hexagon:2611.01091v1

Abstract

Let A be a permutation with two cycles and B a target permutation. We ask whether B is isomorphic to the synchronous product AX for some quotient X, how to construct one, and how many quotients there are up to isomorphism. Suppose the cycle lengths in A are 2 and an odd $b > 1$. If the input lists cycle lengths and multiplicities in binary, deciding existence is NP-complete and counting quotient classes is #P-complete under parsimonious polynomial-time reductions. Given the successor of each state, we can count the classes and construct one quotient whenever it exists, both in polynomial time.

For arbitrary cycle lengths a, b of A, put $g = \gcd(a, b)$. In either representation we construct, in deterministic polynomial time, a new target and a divisor with cycle lengths $gd, g(d+1)$, $d \geq 2$. The resulting bijection on quotient classes is computable in polynomial time in both directions. For each fixed g, varying d gives the same two completeness results for binary lists. With $d = 2$ fixed and g varying, one binary quotient can be found in polynomial time whenever it exists. Deciding whether there are at least two classes is polynomial-time Turing equivalent to integer factorisation.

For two arbitrary divisor lengths, division from successor tables is decidable in polynomial time if and only if Boolean satisfiability on m-bit formulas is decidable in deterministic time $m^{O(\log \log m)}$. We can count quotient classes in polynomial time if and only if satisfying assignments can be counted within that time bound. Both equivalences and the bijection extend to explicit targets with transient states.

1 Can two consecutive cycles divide a permutation?

Write C_t for a directed cycle of length t. We add permutations by taking their disjoint union and multiply them by advancing both coordinates one step at a time. We identify isomorphic permutations. In the product $C_a C_t$, each pair returns to its starting point after $\operatorname{lcm}(a, t)$ steps. Dividing the at states by this orbit length gives the product rule

$$C_a C_t = \gcd(a, t)\, C_{\operatorname{lcm}(a,t)}.$$

The smallest consecutive lengths greater than one are two and three. Consider the division problem

$$(C_2 + C_3)X = C_3 + 2C_6.$$

The right side has fifteen states, so X must have three. Its product with C_2 uses six states and has only even cycle lengths. It must therefore account for one C_6. On three states only $X = C_3$ can produce that length. Now look at what remains for $C_3 X$. The required $C_3 + C_6$ can be obtained from $X = C_1 + C_2$, whereas the choice already forced gives $C_3 C_3 = 3C_3$. The two parts of the product, which we call the *projections*, are individually possible. No single quotient realises both. The input gives only their union B. If both projections contain cycles of the same length, their multiplicities add in B.

Doré et al. give a polynomial-time division algorithm for connected quotients [11, Theorem 3.13]. Their *unroll* starts at each periodic state and follows its predecessors to form a rooted tree. For a disconnected quotient they ask which periods to choose when rolling these

trees back into cycles. They can also construct the two extreme rolls, which group the copies of each recurring tree pattern into as few or as many components as possible [11, Algorithm 5.10 and Section 6, p. 14]. For permutations this question takes a simple form. Each state has exactly one predecessor, so every permutation on n states unrolls to n identical one-way infinite paths [11, Remark 2.3]. Grouping those paths into cycles amounts to choosing a partition of n. In the example above, 3, $2+1$ and $1+1+1$ all give the required unroll. The prescribed product rules out all three. Changing the target to $3C_2 + C_3 + C_6$ gives a useful comparison. Of the same three partitions, only $2+1$ works, so the quotient is $C_1 + C_2$ [22, Section 2].

How much does it help to know that the two lengths are consecutive? For coprime lengths, every instance can be put in this form. The same holds for the reduced lengths of any pair. The new divisor retains the original greatest common divisor.

In the *binary model* we give each distinct cycle length together with its positive multiplicity, using self-delimiting binary fields of size linear in their bit lengths. In the *explicit model* we give the successor of every state. A single cycle of length 2^k takes $O(k)$ bits in a binary list and 2^k entries in a successor table. The total number of states in the divisor and target is $N = |A|+|B|$, so the tables take $O(N \log(2N))$ bits. Until Corollary 8.2, A, B are nonempty permutations, and we ask whether $AX = B$ for some X with any number of cycles. Allowing arbitrary finite self-maps X gives the same question. A transient state of X would give a transient state in the product, and every state of B is periodic. Corollary 8.2 returns to the case where the target itself has transient states.

Theorem A (Consecutive lengths after removing the gcd)**.** *Let $A = C_a + C_b$ and $g = \gcd(a, b)$. In either input representation one can construct, in deterministic polynomial time, a divisor $A' = C_{gd} + C_{g(d+1)}$ with $d \geq 2$ and a target B'. The quotient classes of $AX = B$ and $A'Y = B'$ are in bijection. Both directions of the bijection are computable in time polynomial in the input and the supplied quotient. The integer g may vary with the input. In the explicit model the two output tables have $O(N^4)$ states in total.*

The first step in the proof recovers a piece of information hidden by the product. Whenever both projections can contribute at the same length, their individual contributions there are forced by their numbers of states. We recover this split both before and after changing the divisor lengths. It determines which part of the target to use in each direction. Section 3.1 gives the map.

Write $\mathcal{D}_2$ for the explicit decision problem with $A = C_a + C_b,\ a, b > 1,\ \gcd(a, b) = 1$. The lengths are part of the input. Write SAT for Boolean satisfiability and QP for deterministic quasipolynomial time, meaning time $\exp(O((\log(m+2))^c))$ for some constant $c \geq 1$ on m-bit inputs.

Theorem 1.1 (Two representations)**.** *For divisors $C_2 + C_b$ with $b > 1$ odd and part of the input, binary division is NP-complete, and counting quotient isomorphism classes is $\#\mathsf{P}$-complete under parsimonious polynomial-time reductions. On the same family, explicit division admits a deterministic polynomial-time algorithm that gives the exact count and constructs one quotient whenever one exists.*

For the full explicit problem $\mathcal{D}_2$, the following statements hold.

(i) *If SAT has a deterministic polynomial-time Turing reduction to $\mathcal{D}_2$, then $\mathsf{P} = \mathsf{NP}$.*

(ii) *$\mathcal{D}_2 \in \mathsf{QP}$ if and only if $\mathsf{NP} \subseteq \mathsf{QP}$.*

(iii) *$\mathcal{D}_2 \in \mathsf{P}$ if and only if SAT on m-bit formulas is decidable in deterministic time $m^{O(\log \log m)}$. A polynomial-time decision algorithm for $\mathcal{D}_2$ also suffices to construct one quotient as a successor table whenever one exists.*

Consequently, $\mathcal{D}_2$ is NP-intermediate under polynomial-time many-one reductions exactly when SAT has no deterministic $m^{O(\log \log m)}$-time algorithm. If $\mathsf{NP} \not\subseteq \mathsf{QP}$, then $\mathcal{D}_2$ also lies outside QP.

For $C_2 + C_b$, group quotient lengths t by $R = \operatorname{lcm}(2, b, t)$. The C_b projection gives length $v = \operatorname{lcm}(b, t)$. Since $R = \operatorname{lcm}(2, v)$, only R and $R/2$ are possible. A dynamic program tracks their totals. The $2 : b$ ratio of projection sizes gives the other totals it needs (Section 2). The running time is polynomial in the number $N = |A| + |B|$ of states. A short binary list can describe exponentially many states. Section 7 encodes each triple in the digits of a cycle length and gives a bijection between exact covers and quotient classes. Such a reduction is called *parsimonious*.

To understand the threshold in part (iii), start with a division input on N states. Each block becomes a SAT question of bit length ℓ with $(\log \ell)(\log \log \ell) = O(\log N)$. The proposed SAT running time is therefore polynomial in N. For the reverse implication we turn a formula into a cubic graph, encode a proper edge-colouring by a weighted table and realise its cells as quotient cycles. The costliest step is writing the two successor tables. Appendix D bounds this cost by $m^{O(\log \log m)}$ on every formula.

Part (i) concerns reductions that themselves run in polynomial time. A many-one reduction produces one instance. A Turing reduction can adapt its next query to the answers already received. Even in this latter model a reduction from SAT would allow us to decide SAT recursively on shorter formulas. The short-query principle of Allender and Koucký then gives polynomial time [1, Theorem 5.1(3)]. Thus if SAT misses the threshold in (iii), explicit division is NP-intermediate: it belongs to NP, lies outside P and is not NP-complete.

For the representation contrast alone, Sections 2 and 7 give the algorithm and the reduction. For binary lists in the normal form, varying d already gives NP-completeness at $g = 1$. Holding $d = 2$ fixed and varying g makes one quotient easy to construct. Proposition C.3 explains why finding out whether there is a second is equivalent to factoring integers. For successor tables, Corollary 8.1 gives the SAT and counting thresholds for arbitrary pairs of lengths. The earlier NP-completeness proof allows the number of divisor cycles to grow [20, p. 14].

Readers interested in the period-selection question can follow the proof of the normal form in Section 3.1 directly to the extension to rooted trees in Corollary 8.2.

Relation to known algorithms. For permutations given by cycle lists, Porreca and Rolland's binary algorithm applies when the shortest divisor length divides every other one. They ask about extending this condition to several seeds [26, Theorem 42 and Section 7]. Theorem 1.1 gives the complexity of the binary problem when two incomparable lengths are allowed. Crespelle, Phan and Richard describe quotients recursively and work out a division with divisor $C_2 + C_3$ [8, Theorem 3.8 and Example 3.14]. Here the two lengths vary with the input and we track the time needed to write their successor tables. In the explicit model, Bridoux et al. split an instance into independent smaller ones and rescale their quotient lengths. Their decomposition generally gives a list of instances [5, Lemmas 2 and 8]. Theorem A gives a single instance with consecutive reduced lengths and the original divisor gcd. Their counting theorem gives polynomial time whenever the number of divisors of $\operatorname{lcm}(a, b)$ stays bounded [5, Theorem 1]. Two distinct primes, or the squares of two distinct primes, satisfy that condition. To see the difference when the bound grows, take $a = 2$ and let b be the product of $k \geq 2$ distinct odd primes. Then $2b$ has 2^{k+1} divisors, so the exponent in their bound grows with k. The dynamic program below keeps just two running totals for every such b. For two cycles of lengths a, b, divide both lengths by $g = \gcd(a, b)$. Theorem 2.2 gives polynomial-time explicit counting when one of these two reduced lengths has a bounded number of divisors. If both divisor counts are bounded, Proposition B.2 constructs one binary quotient in polynomial time. Thus for every divisor $C_{2g} + C_{3g}$ we can find a quotient efficiently from binary lists, although the exact count can itself take superpolynomially many bits (Proposition C.2).

Gadouleau and Johnson solve the cycle division equation with multiplicities taken modulo two [15, Theorem 3.9]. Our binary lists retain every digit of each multiplicity. For example, $C_2 + C_3$ divides itself. Adding two fixed points to the target preserves its parity data but makes

division impossible, since every cycle of the product has length divisible by two or three.

Ascone, Porreca and Rolland prove hardness for a linear equation in three unknowns and a constant right-hand side [2, p. 2]. Here only X is unknown. Given B alone, one may instead seek two connected factors [21]. A unique quotient can itself use many different cycle lengths when the number of divisor lengths grows [22]. With two coprime divisor cycles we can list all candidate lengths in polynomial time. It is the choice of multiplicities that encodes a colouring.

2 Solving the blocks of an explicit input

Read the cycle lengths of B from its successor table. Write β_ℓ for the number of cycles of length ℓ, taking zero for an absent length. A quotient must have $n = |B|/(a + b)$ states, so a nonintegral n gives a negative answer. Its unknowns are the integers x_t counting cycles of length t, with $1 \le t \le n$ and $0 \le x_t \le n$. To account for every copy of C_ℓ, the product rule requires

$$\sum_{\operatorname{lcm}(a,t)=\ell} \gcd(a,t)x_t + \sum_{\operatorname{lcm}(b,t)=\ell} \gcd(b,t)x_t = \beta_\ell. \tag{1}$$

Write this equation for every length that can occur in either sum, including a zero right side wherever B forbids that length. Reject if some prescribed length has no possible contribution. When the two projections produce the same length, their coefficients add and the sum is at most $a + b$. These equations also account for the size of X. Multiplying the equation at ℓ by ℓ and summing gives $(a+b)\sum_t tx_t = |B|$. Rolland describes the admissible component types and their multiplicity equations for general finite systems [27, Propositions 4.40–4.41]. His witness algorithm is fixed-parameter tractable in the number of distinct target component types and the shortest divisor period [27, Theorem 4.43]. For $C_2 + C_b$, the count below allows arbitrarily many distinct target lengths and any odd $b > 1$. The cycle product rule supplies the two contributions of each unknown.

Which equations can influence one another? Take $a = 2, b = 3$ again and now allow $n \ge 5$. A C_3 contributes to lengths 6 and 3. A C_5 contributes to lengths 10 and 15. Taking the lcm with 6 gives the common value 6 for the first pair and 30 for the second. This value stays fixed along every chain of variables and the equations in which they occur. In particular, no such chain can connect the two pairs.

Lemma 2.1 (Short queries)**.** *For any two divisor cycles, explicit division reduces in polynomial time to the conjunction of at most N SAT queries. For sufficiently large N, each query has bit length* $\exp(O(\log N/\log\log N))$.

Proof. Let $D = \operatorname{lcm}(a, b)$. Give variable x_t the label $R = \operatorname{lcm}(D, t)$. Whenever it occurs in the equation at length ℓ, that equation satisfies $\operatorname{lcm}(D, \ell) = R$. Taking the lcm of t first with a or b and then with D is the same as taking it directly with D. Thus equations with different labels use disjoint sets of variables. There are at most n nonempty blocks. We build them in polynomial time by running through $t = 1, \dots, n$ and recording its two contributions.

To bound a single block, notice that every length t or ℓ in it divides its label R, and $R \le abn \le N^3$. Write $\tau(R)$ for the number of positive divisors of R. The block has at most $\tau(R)$ variables and at most $\tau(R)$ equations. Its coefficients, right sides and bounds take $O(\log(2N))$ bits apiece. Even a dense matrix, with rows and columns numbered locally, needs only $O((1 + \tau(R))^2 \log(2N))$ bits.

The elementary divisor estimate needed here is $\log \tau(R) = O(\log N/\log\log N)$. Set $z = \log(N^3)$. In the factorisation $R = \prod p^{e_p}$, each exponent is at most $z/\log 2$. Primes up to $\sqrt{z}$ therefore contribute at most $\sqrt{z}\log(1 + z/\log 2)$ to $\log \tau(R)$. The other primes have exponent sum at most $2z/\log z$, and $e + 1 \le 2^e$. The two estimates give the stated bound.

Feasibility of the resulting bounded integer system belongs to NP, since one can guess its multiplicities and check the equations. A fixed polynomial reduction to SAT raises the bit length

of a block to at most a constant power, which preserves the claimed bound. Every block must be feasible, and solutions to separate blocks can be joined by taking all their multiplicities. □

Proof of Theorem 1.1(i). Let an oracle algorithm decide SAT using $\mathcal{D}_2$ in polynomial time. On an m-bit formula it asks at most $m^{O(1)}$ questions, each of polynomial length. Simulate it in order. Answer a malformed table or a divisor outside the specified class negatively. For each valid question we have $N \le m^{O(1)}$. Lemma 2.1 gives a conjunction of at most N SAT queries, each of length at most $\lfloor\sqrt{m}\rfloor$ for sufficiently large m. Evaluate the conjunction recursively and return its truth value before the simulation continues.

Below a fixed input size solve SAT by exhaustive search. Both the number of recursive calls along the simulated computation and the work to produce them are polynomial in m. If $T(m)$ is the worst running time on inputs of at most m bits, some constant c gives $T(m) \le m^c(1+T(\lfloor\sqrt{m}\rfloor))$ for large m. For any $K > 2c$, substituting the inductive bound $T(k) = O(k^K)$ on shorter inputs gives $O(m^{c+K/2}) = O(m^K)$. Thus SAT would belong to P. This includes a many-one reduction as the one-query case. □

When the divisor has more cycles. The short-query argument uses only a bound on $D = \operatorname{lcm}\{a_i\}$. For a divisor $\sum_i \alpha_i C_{a_i}$, give x_t the same label $R = \operatorname{lcm}(D, t)$. Each possible output $\operatorname{lcm}(a_i, t)$ has that label, and each matrix coefficient is at most $\sum_i \alpha_i a_i = |A|$. If $D \le N^c$ for a fixed c, then $R \le N^{c+1}$, so the same divisor estimate gives the same short-query bound, with constants depending on c. This includes any fixed bound on the number of distinct divisor lengths, with arbitrary multiplicities. The block split is also implicit in the general decomposition of Bridoux et al. [5, Lemma 2].

Counting a block from the reduced ratio of the two lengths

Consider $A = C_2 + C_b$. In a block labelled R, a length r coming from the C_b projection satisfies $\operatorname{lcm}(r, 2) = R$. Thus $r = R$ or $r = R/2$. We can remember the totals for these lengths even if the other projection produces many different lengths. For general a, b, a common factor actually reduces the number of possible row lengths. This is why the reduced ratio $a : b$ enters the running time.

Write a *row* for the output of C_a and a *column* for the output of C_b. We can determine how many target cycles each must supply before choosing any quotient cycles. Write $D = \operatorname{lcm}(a, b)$, and group the target cycles of length ℓ by $R = \operatorname{lcm}(D, \ell)$. Both projections of a quotient cycle of length t belong to the group with $R = \operatorname{lcm}(D, t)$. Reject a target length divisible by neither a nor b. In the group B_R, only the length R can occur in both projections: any such length is divisible by D and hence equals its label. The support of a block is the set of lengths occurring in it. Let β_ℓ denote the target multiplicity, taking $\beta_R = 0$ if R is absent. The number of quotient states in the group must be $n_R = |B_R|/(a+b)$. Reject if it is nonintegral. The projection from C_a has an_R states, so its contribution at the only shared length is forced to be

$$m_R = \frac{an_R - \sum_{\ell \in \operatorname{supp} B_R,\, \ell \ne R,\, a \mid \ell} \ell \beta_\ell}{R}. \tag{2}$$

Reject unless m_R is an integer between zero and β_R. Assign m_R copies at R to the row projection from C_a and $\beta_R - m_R$ to the column projection from C_b. Every other multiplicity goes to its unique projection. We now have separate nonnegative margins, including when $a = b$.

These are necessary margins. We still have to realise both of them by one quotient. Their uniqueness also follows from classical fixed-point counts. If $F_j(Z)$ counts the fixed points of the jth iterate, then $F_j(AX) = F_j(A)F_j(X)$. For $a \mid j$, the row projection has $aF_j(B)/F_j(A)$ fixed points; for $a \nmid j$ it has none. These counts determine its cycles by divisor inversion [12, §2, (2-7-1)–(2-7-2)]. The local formula (2) needs only the lengths in one block.

Theorem 2.2 (Counting from the reduced ratio)**.** *Let $A = C_a + C_b$ and B be given by successor tables. Put $g = \gcd(a, b)$ and $K = \min\{\tau(a/g), \tau(b/g)\}$, where $\tau(d)$ counts the positive divisors of d. There is a deterministic algorithm that counts all quotient isomorphism classes and, when the count is positive, constructs the successor table of one quotient. Each layer of its dynamic program for a block has at most $(N+1)^K$ states, indexed by at most K row totals. The full algorithm takes $(N+1)^{K+O(1)}$ bit operations. The lengths a, b may have common factors.*

Proof. Read the cycles of both permutations. Form the equations and the independent blocks of Lemma 2.1, including zero targets and rejecting a nonintegral quotient size or an unreachable target length. Interchange a, b if necessary so that $\tau(b/g) = K$. Both divisor counts can be computed by trial division up to N. For $1 \le t \le n$, call $r = \operatorname{lcm}(a, t)$ its row and $c = \operatorname{lcm}(b, t)$ its column in the block $R = \operatorname{lcm}(a, b, t)$. Since $\operatorname{lcm}(r, b) = R$,

$$R/r = b/\gcd(r, b).$$

Since $a \mid r$, the denominator is a multiple of g, and R/r divides b/g. For fixed R this divisor determines r. Thus there are at most $\tau(b/g) = K$ row lengths. Interchanging the two axes likewise bounds the number of column lengths by $\tau(a/g)$.

Use (2) after ordering the two axes. Write ρ_r for the number of cycles required from row r and γ_c for the number required from column c. A row and a column of the same numerical length have separate quotas. A positive quota with no candidate variable on its side makes the block infeasible.

Let P be the set of candidate rows. Process one column at a time in any fixed order, fixing an order of variables within each column. Keep the number s_r already supplied by each row. During column c, let z be its current count and let J be the completed columns. If the processed quotient cycles use u states, the two projections have used au and bu states. Therefore

$$b \sum_{r \in P} r s_r = abu = a \left(\sum_{d \in J} d\gamma_d + cz \right).$$

The sum over completed columns is fixed at this step, so the row counters determine z.

For a candidate period t try each multiplicity $x_t = 0, \dots, \lfloor n/t \rfloor$. Add $\gcd(a, t)x_t$ to its row and $\gcd(b, t)x_t$ to its current column. Discard a transition if it exceeds either quota. At the end of the column require $z = \gamma_c$, and after the last column require $s_r = \rho_r$ for every row. This rule also applies when $r = c$.

Two prefixes at the same step with the same row counters have identical possible completions. Starting the zero state with count one, add the counts of all transitions reaching the same state. Each full vector (x_t) follows one path through the prescribed variable order and specifies one isomorphism class. Multiply the final counts of the independent blocks. The identity following (1) then gives the required size n without a separate counter for states.

For reconstruction, keep one predecessor and its chosen x_t for each reached state at each step. Reading back from the accepted state in each block supplies all multiplicities. Writing that many disjoint cycles gives a successor table on exactly n states. The number of distinct row vectors per layer is at most $\prod_{r \in P}(\rho_r + 1) \le (N+1)^K$. There are at most $n+1$ choices per variable and n variables altogether. All partial counts and products over completed blocks are bounded by $(n+1)^n$, the number of unrestricted choices in $\{0, \dots, n\}^n$. Integer arithmetic and comparing states thus cost polynomially many bit operations per transition. Sorting transitions by destination merges them deterministically within the claimed bound, including predecessor storage and reconstruction. □

For example, take $(C_2 + C_3)X = 6C_2 + 6C_3 + 5C_6$. Every quotient length divides six. Take the C_2 projection as the rows, of lengths two and six. While the algorithm fills its first column, of length three, the balance of states gives $s_3 = s_2 + 3s_6$. The two row counters thus determine

all three totals. To see the resulting count, observe that the targets at two and three require $x_1 + 2x_2 = x_1 + 3x_3 = 6$, so x_1 is either six or zero. At the shared length six, the equation is $x_2 + x_3 + 5x_6 = 5$. Formula (2) assigns two of the five cycles of length six to the row and three to the column. If $x_1 = 6$, one quotient cycle C_6 supplies all five copies through the two projections. If $x_1 = 0$, the three copies of C_2 and two copies of C_3 together supply those same five cycles. The count is two. In neither case is there a further choice of how to assign the five target cycles to the projections.

There is also a route to the decision bound through integer programming. In the matrix with separate row and column quotas, cut the ordered list of variables between two positions. The prefix and suffix column spans can overlap on the rows P and at most the column c currently being filled. Every vector y in their intersection satisfies the same linear balance as the product columns, $b\sum_{r\in P} r y_r = ac\, y_c$ when c is present. Thus the intersection has dimension at most $|P| \le K$. This supplies a path decomposition of width at most $K+1$ in the convention of Fomin et al. Their Theorem 6 gives a feasibility algorithm for this order, building on Cunningham and Geelen's approach to integer programming [9, 14]. The direct recurrence also counts classes and records a witness.

3 Reading a table from a product of cycles

Suppose $\gcd(a, b) = 1$. If a quotient contains a t-cycle, its two contributions to B have lengths $\ell = \operatorname{lcm}(a, t)$ and $\ell' = \operatorname{lcm}(b, t)$. These two lengths determine t through the identity

$$\gcd(\ell, \ell') = t.$$

Indeed, at each prime at least one of a, b has exponent zero, so the smaller of the two output exponents is the exponent in t. With k distinct lengths in B, at most k^2 gcds have to be tried, allowing a length to be paired with itself. What remains is to choose how often each candidate occurs.

Here is a product in which those multiplicities can be read as a table. Take $A = C_6 + C_{35}$. Use rows indexed by the divisors $5, 7$ of 35 and columns indexed by the divisors $2, 3$ of 6. In the cell with indices u, v, put the candidate quotient cycle C_{uv}.

	$v = 2$	$v = 3$	length contributed by C_6
$u = 5$	C_{10}	C_{15}	30
$u = 7$	C_{14}	C_{21}	42
length contributed by C_{35}	70	105	

Take one copy in each diagonal cell. The 10-cycle contributes $2C_{30}$ to its row and $5C_{70}$ to its column. The 21-cycle contributes $3C_{42}$ and $7C_{105}$. Thus

$$(C_6 + C_{35})(C_{10} + C_{21}) = 2C_{30} + 3C_{42} + 5C_{70} + 7C_{105}.$$

Reading from the right, the required multiplicity of C_{30} is a row sum. A copy in its first cell counts twice and a copy in its second cell counts three times. Down a column it is the row index that supplies the weight.

Lemma 3.1 (Two families of target lengths)**.** *Let $a, b > 1$ be coprime. Choose distinct proper divisors u_i of b and distinct proper divisors v_j of a. For nonnegative integers R_i, S_j, put*

$$A = C_a + C_b, \qquad B = \sum_i R_i C_{au_i} + \sum_j S_j C_{bv_j}.$$

Quotient classes correspond bijectively to nonnegative integer tables $x = (x_{ij})$ satisfying

$$\sum_j v_j x_{ij} = R_i \quad \text{for each row } i, \qquad \sum_i u_i x_{ij} = S_j \quad \text{for each column } j. \tag{3}$$

The corresponding quotient is $X = \sum_{i,j} x_{ij} C_{u_i v_j}$.

Proof. Consider any cycle C_t in a quotient. Its contribution from C_a has length divisible by a. This rules out every bv_j, since $\gcd(a,b) = 1$ and v_j is a proper divisor of a. Thus this length is au_i. Similarly the contribution from C_b has length bv_j. Taking the gcd recovers $t = u_i v_j$. The factors u_i, v_j are recovered from t by taking its gcd with b, a, respectively. Every cycle of every quotient therefore belongs to exactly one cell. Conversely $C_a C_{u_i v_j} = v_j C_{au_i}$ and $C_b C_{u_i v_j} = u_i C_{bv_j}$. The row and column lengths are all distinct, so adding the contributions gives exactly (3). □

3.1 Separating the two projections

In the last example a useful change of lengths is already visible: $6 \cdot 6 - 35 = 1$. Replacing $6, 35$ by $36, 35$ carries the same diagonal quotient $C_{10}+C_{21}$ to the target $2C_{180}+3C_{252}+5C_{70}+7C_{105}$. The two row lengths were multiplied by six. The column lengths and the four required multiplicities stayed the same.

For an arbitrary input the first difficulty is to identify the rows. For instance, $(C_2+C_3)(C_1+C_3) = C_2 + 4C_3 + C_6$. The C_6 comes from C_2C_3. If we called it a column because its length is divisible by three, changing $2, 3$ to $4, 3$ would give $C_4 + 4C_3 + C_6$. Its 22 states cannot be a product with a seven-state divisor. The total number of states in each projection tells us which assignment is possible.

Proof of Theorem A. Put $D = \operatorname{lcm}(a, b)$. Group the observed lengths ℓ by their common label $R = \operatorname{lcm}(D, \ell)$. Both images of a quotient cycle of length t belong to the block B_R with $R = \operatorname{lcm}(D, t)$. The split in (2) either detects an impossible target or gives its required row and column multiplicities in every block. Write $n_R = |B_R|/(a + b)$ for the required quotient size there.

Choosing the new divisor. Write $U = a/g$, $V = b/g$. We need positive P, Q with $UP - VQ = 1$ and $VQ \geq 2$. The $(6, 35)$ example used $P = 6, Q = 1$. For a uniform construction take $1 \leq P_0 \leq V$ with $UP_0 \equiv 1 \pmod{V}$, taking $P_0 = 1$ if $V = 1$, and put $Q_0 = (UP_0 - 1)/V$. Set $P = P_0 + 2V$, $Q = Q_0 + 2U$ and

$$a' = Pa, \qquad b' = Qb, \qquad D' = \operatorname{lcm}(a', b') = PQD.$$

Then $a'/g = b'/g + 1$, $b'/g \geq 2$, and $\gcd(P, Q) = \gcd(P, V) = \gcd(Q, U) = 1$. Writing $\nu_p(t)$ for the exponent of p in t, for every prime $p \mid P$ we have $\nu_p(a) \geq \nu_p(b)$, and symmetrically for Q.

Transforming a block. If a change of divisor length introduces a prime power already forced by the block label, the quotient length must change with it. The factors below record precisely those forced changes. Let F_R be the set of primes dividing R/D. Let P_R and Q_R be the products of the full prime powers of P and Q, respectively, supported on F_R. Empty products are one. The mapping on quotient lengths in this block is $t \mapsto y = P_R Q_R t$. Form B' by the following rules:

	new length	new multiplicity
row of length r and multiplicity k	$PQ_R r$	$P_R k$
column of length c and multiplicity k	$QP_R c$	$Q_R k$.

Add multiplicities when the new lengths coincide. For example, take $A = 2C_1$ and $B = 2C_6$. Each projection must supply one C_6. Here $P = 3, Q = 2, R = 6$, so the quotient C_6 becomes C_{36}. Its two contributions are $3C_{36}$ and $2C_{36}$, giving the single observed length in the new target $5C_{36}$.

For $\operatorname{lcm}(D, t) = R$, the transformation follows from the identities

$$\operatorname{lcm}(a', y) = PQ_R \operatorname{lcm}(a, t), \quad \gcd(a', y) = P_R \gcd(a, t),$$
$$\operatorname{lcm}(b', y) = QP_R \operatorname{lcm}(b, t), \quad \gcd(b', y) = Q_R \gcd(b, t).$$

Indeed at $p \in F_R$ the exponent in t is its exponent in R, strictly greater than that in either original axis. Raising it by $\nu_p(PQ)$ keeps it above both new axes. At $p \notin F_R$ the exponent in t is at most that in D. If $p \mid P$ then $\nu_p(D) = \nu_p(a)$, so increasing the exponent in a multiplies the row length by the p-part of P and leaves its multiplicity and the whole column contribution unchanged. The case $p \mid Q$ is symmetric; elsewhere the exponents are unchanged. Thus every original solution has the stated image.

Recovering every quotient. Every target length transformed in block R has new label $S = PQR$. At primes in F_R its original exponent is the one in R, and the rules add $\nu_p(PQ)$. At all other primes its new exponent is at most the one in $D' = PQD$. Thus the new label identifies the original block.

Which row and column can meet? A common image length is divisible by both new axes, so it equals its own label S. If row r and column c meet, then $PQ_R r = QP_R c = PQR$, giving

$$r = (Q/Q_R)R, \qquad c = (P/P_R)R.$$

Both r and c divide R. Hence $r = c = R$, $P_R = P$ and $Q_R = Q$. Conversely, under these conditions positive quotas on both sides at R do meet at S.

Put $E = P_R Q_R$ in any block. The prescribed image rows have $a' E n_R$ states and the columns have $b' E n_R$ states. Thus any quotient has $E n_R$ states in this block. Formula (2) applied to the new axes forces the prescribed share at S, even if one side contributes zero.

Now take a cycle of length y in any quotient of B'. Its label $\operatorname{lcm}(D', y) = S$ determines R. By the forced split its two projection lengths are $PQ_R r$ and $QP_R c$ for some $r, c \mid R$. Since y divides both lengths,

$$y \mid R \gcd(PQ_R, QP_R) = ER,$$

where the equality uses $\gcd(P/P_R, Q/Q_R) = 1$. For each $p \in F_R$, the exponent in S exceeds that in D', so $\nu_p(y) = \nu_p(R) + \nu_p(E)$. Hence $E \mid y$, and $t = y/E$ divides R with the full exponent of R at every $p \in F_R$. At all other primes D and R have the same exponent. Thus $\operatorname{lcm}(D, t) = R$. In a block where the two sides meet, this inverse simply divides every quotient length by PQ.

The four identities above now recover the contributions of C_t. Within a block each side scales its lengths by a fixed factor, so dividing its image multiplicities by P_R on the row side and by Q_R on the column side recovers all the old margins. The decoded cycles therefore give $AX = B$. Each keeps its multiplicity, and mapping t forward gives back y. The maps are inverse on quotient classes.

Bit complexity. Let L be the length of the binary cycle lists. Extended Euclid computes P_0, and $P \le 3V$, $Q \le 3U$. No factoring of a binary axis or target period is needed. For example,

$$P_R = \gcd(P, (R/D)^{\lceil \log_2(2P) \rceil})$$

is computed by modular powering, and similarly for Q_R. There are only polynomially many periods and blocks. The displayed formulas use polynomial-bit integers throughout. A quotient can be mapped or decoded in time polynomial in the input and its own list. On a rejected input output $B' = C_1$ with the new divisor, which admits no quotient.

On explicit inputs let $N = |A| + |B|$. Reading their cycles gives $L = O(N \log(2N))$. A new row's number of states is its old number times $PP_R Q_R$, and a column's number is its old number times $QP_R Q_R$. Equivalently, if the margin split passed,

$$|B'| = (Pa + Qb) \sum_R P_R Q_R \, n_R.$$

Thus the two output permutations have at most $PQ \max(P, Q) N = O(N^4)$ states, independently of feasibility. Writing their successor tables and, when requested, expanding the cycles of a witness take polynomial bit time. □

With just one divisor length, each candidate contributes to a single target length. One can solve the resulting equations separately in polynomial time, even in binary [10, Theorems 3 and 4]. Appendix B gives the criterion and a short certificate for general binary division. With two lengths the same entry x_{ij} must help meet both a row sum and a column sum. We now use that requirement to make several occurrences of an edge colour agree.

4 How a table records an edge-colouring

Take a nonempty simple cubic graph G'. To colour its edges with three colours, we must give the two ends of each edge the same colour, and use each colour once at each vertex. We will make a row choose a colour and let the columns check who else has chosen it.

We want each element to occur in exactly three triples. Starting with one triple (v, w, e) per edge $e = vw$ nearly works. Each vertex occurs three times and each edge just once. Give every vertex v three copies v_0, v_1, v_2, one per layer. Within layer ℓ all three incident edges use the same v_ℓ. Keep a single element e for the edge across all layers. The triples are now

$$h = (v_\ell, w_\ell, e), \qquad \ell = 0, 1, 2.$$

Each element t, whether it is a vertex copy or an edge, belongs to exactly three triples. Require the triples through t to have all three colours. In layer 0 this gives an edge-colouring of G'. Conversely, any proper edge-colouring κ gives each triple (v_ℓ, w_ℓ, e) the colour $\kappa(e) + \ell \bmod 3$. Thus the extra layers cost nothing for existence. They will matter when we count the tables in Section 7.

A row chooses a colour. In (3) a row sum uses the *column* weights. We want the row for $h = (v_\ell, w_\ell, e)$ to select one column for each of its elements, all of the same colour. Give each element t a symbol B_t for its base weight, and give each vertex v a symbol D_v. Initially treat different symbols as independent. Define

$$\Delta_{v_\ell} = D_v, \qquad \Delta_e = -D_v - D_w \quad (e = vw), \qquad X_{t,c} = B_t + \sigma_c \Delta_t, \qquad (\sigma_0, \sigma_1, \sigma_2) = (0, 1, 3).$$

Suppose the row selects its three X-columns at colours i, j, c, respectively. After subtracting its target $\sum_{t \in h} B_t$, the excess weight is

$$(\sigma_i - \sigma_c) D_v + (\sigma_j - \sigma_c) D_w. \tag{4}$$

For example, choosing colour 2 at both ends and colour 1 at the edge leaves $2D_v + 2D_w$. Equality with the target requires both coefficients to vanish, so all three choices must agree.

One triple fills a pair of columns. Give column $X_{t,c}$ a partner of weight $Y_{t,c} = H_t - X_{t,c}$, where H_t is a new symbol for element t. A row that visits both columns once receives exactly H_t.

A triple h has seven rows. Four visit all three elements of h at its chosen colour c. The other three are indexed by the elements of h. At a given element t, only its own incidence row F_{ht} joins the four shared rows. The five rows split as follows.

	$X_{t,c}$	$Y_{t,c}$
rows for the whole triple h	$P_h,\ G_h$	$R_h,\ K_h$
row for this incidence (h, t)	F_{ht}	F_{ht}

Each named row contributes one copy to its column. If each column accepts exactly one such filling, the three triples at t must choose different colours.

To realise these sums, give the pair P_h, G_h weights p_h, g_h with sum $L_h = p_h + g_h$. Give R_h, K_h the same total weight in a different way, using a fresh symbol r_h. Finally, give both columns at (t, c) target T_t. The incidence row F_{ht} must supply the missing weight $T_t - L_h$. The resulting row data are

Rows	Respective weights	Target of each row
$P_h,\ G_h$	$p_h,\ g_h$	$\sum_{t\in h} B_t$
$R_h,\ K_h$	$r_h,\ L_h - r_h$	$\sum_{t\in h}(H_t - B_t)$
F_{ht} for $t \in h$	$T_t - L_h$	H_t

Passing from symbols to integers. Two conditions will make every numerical filling visible in this description. First we place the row weights close to W and the column weights close to V, for positive scales W, V. We use the bounds

$$|p_h - W|,\ |g_h - W|,\ |r_h - W| \le 10^{-4}W, \qquad |T_t - 3W| \le 3 \cdot 10^{-4}W,$$
$$|B_t - V|,\ |H_t - 2V| \le 10^{-4}V, \qquad 0 < D_v < 10^{-5}V.$$

Substitution puts each row weight between $0.999W$ and $1.001W$, and each column weight between $0.999V$ and $1.001V$. Column targets lie within $0.003W$ of $3W$. Main row targets lie within $0.003V$ of $3V$, and incidence row targets within $0.001V$ of $2V$. Thus every column and main row must use exactly three copies, and every incidence row exactly two. This holds for each equation on its own, including in a table with no solution.

Second, the values assigned to distinct symbols on the same side must satisfy no unintended relation among the few terms in a single margin. Call the symbols H-independent if $\sum_\alpha \gamma_\alpha z_\alpha = 0$ with integer coefficients and $\sum_\alpha |\gamma_\alpha| \le H$ forces every $\gamma_\alpha = 0$. A row equation adds at most three *column* weights. For example, for $e = vw$ the weight $Y_{e,2} = H_e - B_e + 3D_v + 3D_w$ has total absolute coefficient 8, the largest on that axis. Subtracting a row target adds at most 6, so $3 \cdot 8 + 6 = 30$ suffices for the column symbols. A column equation adds three *row* weights, each of norm at most 3 (for example $T_t - p_h - g_h$), and subtracts the single symbol T_t. This gives $3 \cdot 3 + 1 = 10$ for the row symbols. Collect repeated symbols first.

Lemma 4.1. *Under the scale bounds above,* 30*-independence of the column symbols and* 10*-independence of the row symbols, the feasible tables are in bijection with the colourings of the triples that use every colour at each element. Their entries are all* 0 *or* 1*. In particular a table exists if and only if G' is properly* 3*-edge-colourable.*

Proof. First read the rows. In P_h or G_h, any Y-column would contribute a positive H-coefficient, absent from the target. The B-coefficients then force one X-column for each element of h. Equation (4) forces them to have one colour, since the endpoints are distinct. In R_h or K_h the three H-coefficients force one Y per element. Cancelling the H's leaves the same colour equation. Finally F_{ht} must use one $Y_{t,c}$ to obtain H_t and one $X_{t,i}$ to cancel $-B_t$. Their sum is H_t precisely when $i = c$.

Next read a column at t. Its target T_t requires exactly one incidence row F_{ht} for some h containing t. The remaining two rows must supply $p_h + g_h$. If either is $K_{h'}$, its $-r_{h'}$ forces $R_{h'}$ and the p, g-coefficients give $h' = h$. Otherwise the only pair is P_h, G_h. Hence the column uses three distinct rows and its occupied entries are all 1.

Now follow a single P_h. It chooses the same colour c at all three elements. Each of its columns requires G_h, F_{ht} alongside it. In turn F_{ht} enters $Y_{t,c}$ and forces R_h, K_h there. This is the displayed filling, and each $X_{t,c}$ admits exactly one triple. Conversely, given a colouring of the triples, put this filling into the two columns for each incidence at its chosen colour. Each column receives exactly one triple and each row meets its target. The two constructions are inverse, since the colour of each triple specifies every occupied cell in its rows. □

5 Encoding the table by two binary cycle lengths

Theorem 5.1. *For binary lists of cycle lengths and multiplicities, division is NP-complete under polynomial-time many-one reductions, even when $A = C_a + C_b$ consists of two coprime cycles with $a, b > 1$. In the resulting table each margin equation allows at most three copies.*

Proof. Membership in NP is Lemma B.1. For hardness take $G' = G$ in Lemma 4.1, where G is a nonempty instance of Holyer's NP-complete problem of three-edge-colourability of simple cubic graphs [16]. To turn its table into a division instance, we take b to be the lcm of the row weights and a the lcm of the column weights, as in Lemma 3.1. We choose the rows first. Once b is known, making each column weight 1 modulo b ensures that a and b are coprime.

The row weights determine b. Let $s = |V(G)| + |E(G)|$. Choose a base $Q = 1024$, and reserve a different low digit j for each of the $J_r = O(s)$ row symbols. For $0 \le j < J_r$, with $W_0 = Q^{J_r+3}$, assign the value

$$cW_0 + Q^j, \qquad c = 1, 1, 1, 3 \quad \text{for } p_h, g_h, r_h, T_t. \tag{5}$$

The high term puts the symbol at its prescribed scale and the low digit identifies it. In a zero relation of norm at most 30, the low terms have absolute sum at most $30Q^{J_r-1} < W_0$. Hence the high terms cancel. If any low term survived, division by the least occupied power of Q and reduction modulo Q would force its nonzero coefficient of absolute value at most 30 to be divisible by 1024.

Set $b = \operatorname{lcm}_i u_i$ over the row weights. Distinct rows have different formal weights, and their differences have norm at most 6, so their values are distinct. All lie in $[0.999W_0, 1.001W_0]$. There are at least two, and no one can be a multiple of a smaller one in this range. They are therefore all *proper* divisors of b.

The column weights must be coprime to b. We arrange that every column weight is 1 modulo b. To do so give B_t, H_t, D_v residues $1, 2, 0$, respectively. Both $X_{t,c} = B_t + \sigma_c \Delta_t$ and $Y_{t,c} = H_t - X_{t,c}$ then have the desired residue. The full values use distinct digit positions $0 \le j < J_c$, where $J_c = O(s)$. With $V_0 = Q^{J_c+3}$, they are

$$b(cV_0 + Q^j) + c, \qquad c = 1, 2, 0 \quad \text{for } B_t, H_t, D_v. \tag{6}$$

These values meet the scale bounds with $V = bV_0$. In any tested relation the sum of the final offsets has absolute value at most $60 < b$. Reduction modulo b forces this sum to vanish. Dividing the remaining equality by b gives the same digit argument as on the row side. The formal column weights are distinct, and the norm of their pairwise differences is at most 16, so the numerical weights are distinct. Again the scale bounds make them proper divisors of their lcm a. As each weight is 1 modulo b, we have $\gcd(a, b) = 1$.

Form the target B of Lemma 3.1 from these weights and their row and column targets. That lemma and Lemma 4.1 give the required equivalence. It remains to pay for writing a, b in binary. Each row weight has $O(s)$ bits, so bounding the lcm by the product gives $\log b = O(s^2)$. Each column weight then has $O(s^2)$ bits, giving $\log a = O(s^3)$. There are $O(s)$ target terms and the pair of cycle lists uses $O(s^4)$ bits altogether. The digit assignments, gcds and lcms are computable in polynomial time. □

6 Writing the two permutations explicitly

The binary reduction assigns one digit position to every name in the table. A cycle length with $O(s^3)$ bits fits in that encoding, but writing out its successors could take $2^{O(s^3)}$ steps. To obtain a quasipolynomial bound we must control the *values* of the two cycle lengths. Each is the least common multiple of a whole family of weights. We give the vertices binary addresses and route each edge through a sequence of addresses, changing one bit at a time. The unchanged bits will supply common factors. The two-bit example below shows how this keeps the lcm small.

Theorem 6.1. *Let G be a nonempty simple cubic graph and $s = |V(G)|+|E(G)|$. In deterministic time $\exp(O(\log^3(s+2)))$ one can write successor tables for $A = C_a + C_b$ and a permutation B, with $a, b > 1$ and $\gcd(a, b) = 1$, such that $AX \cong B$ is solvable exactly when G has a proper three-edge-colouring. Each individual margin equation in the construction still allows at most three copies.*

This bound already proves the quasipolynomial comparison in Theorem 1.1(ii). For part (iii), Appendix D keeps the same graph and changes the integers assigned to its addresses.

Putting the endpoints of an edge close together

Consider the weight $T_t - p_h - g_h$ of the auxiliary row for $t \in h$. The name T_t belongs to an element and p_h, g_h belong to a triple. Suppose we could put these names at the same binary address or at two addresses differing in one bit. If the multiplier for address x were a product of one factor per bit, flipping one bit would leave all the other factors common to the two terms. This is our reason for first modifying the graph.

Number the vertices and give each edge its own nonzero index. Store a vertex index and an edge index in two registers of $d = O(\log(s + 2))$ bits in total. Represent vertex v by $(v, 0)$. Orient an edge $e = vw$ by vertex order and route it through the addresses

$$(v, 0) \longrightarrow (v, e) \longrightarrow (w, e) \longrightarrow (w, 0).$$

Each arrow flips the differing bits of one register in increasing order. At each internal address on the route insert a private copy of K_4 with one edge deleted. Use its two degree-two vertices to join it to the previous and next parts. Copies on different routes stay separate even when they occupy the same address.

Such a copy transmits the colour of the original edge. Indeed, each colour occurs at four vertex incidences. Internal edges account for an even number, so the two external edges must have the same colour. Every choice of this common colour extends across the copy by colouring the three perfect matchings of K_4 and deleting the designated edge. This is the diamond used in [6, proof of Theorem 3.1]. Thus the new graph G' is simple, cubic and three-edge-colourable exactly when G is. It has $O(sd)$ vertices.

Each address has bounded occupancy. An address $(i, 0)$ contains at most the original vertex i. At (i, j) with $j \neq 0$, at most three incident edges of i contribute an outer segment, and only edge j can contribute a middle segment. Each such segment visits the address at most once, giving at most four copies or sixteen vertices. Give all four vertices of a diamond the address where it was inserted. An edge of G' now joins vertices at equal addresses or at addresses one bit apart. Number the new vertices and choose the lower-numbered endpoint of each edge as its owner. Put the names attached to v_ℓ and D_v at v, and those attached to an edge and its triples at its owner. Only a bounded number of names share an address. Every weight in Lemma 4.1 uses one address or two neighbouring addresses. The targets need only have short binary descriptions, since only the weights enter the lcms.

Multipliers for the addresses

A binary address x will also denote the set of bits equal to one. Its multiplier $U(x)$ should keep short relations among different addresses distinguishable and should keep all the names in their prescribed size ranges. We also prescribe a modulus M. Having $U(x) \equiv 1 \pmod{M}$ will let us choose the two axis lcms to be coprime. A symbol at x will have value $\mu U(x)$ with $|\mu| \leq C$. Collecting at most H signed occurrences therefore gives address coefficients whose absolute values sum to at most HC.

Lemma 6.2 (Address multipliers)**.** *Fix $H = 30$. For positive integers d, M, C one can choose positive integers $q, z_0, \dots, z_{d-1}$, with $q \le z_i < 2q$ and $\log q = O(d + \log(2M) + \log(2C))$, such that*

$$U(x) = q^{d-|x|} \prod_{i \in x} z_i, \qquad U(x) \equiv 1 \pmod{M}, \qquad 1 \le U(x)/q^d < 1 + 10^{-7}.$$

If integers b_x have at most H nonzero entries and $\sum_x |b_x| \le HC$, then $\sum_x b_x U(x) = 0$ forces every $b_x = 0$. The integers q, z_i are computable in time polynomial in d, $\log(2M)$ and $\log(2C)$.

Appendix A constructs these multipliers. We first use the product formula for $U(x)$ to see how much it saves.

Why the weights have a small common multiple. With two bits, for instance, the addresses $\{0\}$ and $\{0, 1\}$ have multipliers qz_0 and z_0z_1. A linear combination factors as $z_0(c_1q + c_2z_1)$. Only the term in parentheses depends on the bit that changed. In general every $U(x)$ divides $D_0 = q^d \prod_{i<d} z_i$. A weight at one address is $cU(x)$. When a weight uses x and $x \cup \{i\}$ with $i \notin x$, we can factor out the contribution from every unchanged bit and obtain

$$c_1U(x) + c_2U(x \cup \{i\}) = \frac{U(x)}{q}(c_1q + c_2z_i), \qquad \frac{U(x)}{q} \mid D_0.$$

Suppose for the moment that c, c_1, c_2 range over bounded sets of integers. For positive weights, all the residual factors c and $c_1q + c_2z_i$ that occur are positive. There are $O(1)$ possibilities for the first kind and $O(d)$ for the second, each with $O(\log q)$ bits. Multiplying D_0 by all distinct residual factors gives a common multiple. Thus the logarithm of the axis lcm is $O(d \log q)$, regardless of how many addresses the graph uses.

Choosing the two coprime cycle lengths

Proof of Theorem 6.1. At each address number the names on each axis by distinct slots $j = 0, 1, \dots$. We can reuse slot numbers at different addresses. If J_r, J_c are the maximum slot counts on the two axes, both are bounded independently of s. Put $W_0 = 1024^{J_r+3}$ and $V_0 = 1024^{J_c+3}$. Each name will receive its slot coefficient multiplied by the $U(x)$ of its address. We choose the coefficients and the moduli in the following order.

Row coefficients. Give a row name the coefficient $\rho = cW_0 + 1024^j$, where $c = 1$ for p, g, r and $c = 3$ for T. Temporarily set all address multipliers to one. Call the row weights calculated this way $\bar{u}_i$, and let $R = \operatorname{lcm}_i \bar{u}_i$. For instance an auxiliary row gives $\rho(T_t) - \rho(p_h) - \rho(g_h)$. Each $\bar{u}_i$ differs from W_0 by at most $3\,1024^{J_r-1}$, so it exceeds 60. The bounded slot count gives an absolute bound on the set of possible $\bar{u}_i$, hence on R. This preliminary R lets us finish the columns first. We will make their lcm a coprime to R, then arrange that the final row weights agree with the preliminary ones modulo a.

Column weights and the first lcm. Give a column name the coefficient $\nu = R(cV_0 + 1024^j) + c$, with $c = 1, 2, 0$ for B_t, H_t, D_v, respectively. Apply Lemma 6.2 with $C_c = \max \nu$ as C and $M = R$. At address x take the name's value to be $\nu U(x)$, including the offset $+c$ in the multiplier ν. The names B_t, H_t, D_v retain residues $1, 2, 0$ modulo R. Consequently both $X_{t,c}$ and $Y_{t,c}$ are one modulo R. Their lcm a is coprime to R.

Row weights and the second lcm. Now use Lemma 6.2 with $C_r = \max \rho$ and $M = a$, and take a row name's value to be $\rho U(x)$ with these new multipliers. Each row weight satisfies $u_i \equiv \bar{u}_i \pmod{a}$. Since $\bar{u}_i \mid R$ and $\gcd(a, R) = 1$, we have $\gcd(a, u_i) = 1$ for every i. Their lcm b is therefore coprime to a.

To apply the table construction we also need its independence and scale conditions. In a short relation, first collect all names at the same address. Lemma 6.2 separates those address coefficients. Within one address the low digits then separate the individual names, just as in the binary encoding. The estimate $U(x)/q^d < 1 + 10^{-7}$ keeps the values in the required

intervals. Appendix A checks these bounds with the full coefficients, including the column offsets. It follows that on either axis the weights are distinct and the largest is less than twice the smallest. With at least two weights, all are proper divisors of their lcm. Each edge of G' contributes a column of colour zero and a row from layer zero. Thus both $\tau(a)$ and $\tau(b)$ are at least $|E(G')|$, so the parameter in Theorem 2.2 grows with the graph. Lemmas 3.1 and 4.1 now apply.

Write q_c, q_r for the two values of q. Both C_c and C_r are bounded by absolute constants. Each weight uses a bounded number of slot coefficients, so the integers c, c_1, c_2 in the common-multiple calculation above have the promised bounded ranges. The column choice $M = R$ gives $\log q_c = O(d)$ and $\log a = O(d^2)$. The row choice $M = a$ gives $\log q_r = O(d + \log a) = O(d^2)$ and then $\log b = O(d^3)$. All the weights and targets have $O(d^3)$ bits as well. On every input graph the number of states to write is exactly

$$N = a + b + \sum_i R_i a u_i + \sum_j S_j b v_j.$$

For instance, row i prescribes R_i cycles of length au_i. The state count therefore applies even if no table is feasible. There are $O(sd)$ terms, each with $O(d^3)$ bits, so $N \le \exp(O(\log^3(s+2)))$. The routing and all the binary arithmetic take time polynomial in s. For each output cycle, list successive state indices and then the initial index. Writing the two successor tables takes $O(N \log(2N))$ bit operations, as required. □

Proof of the representation comparison and Theorem 1.1(ii). The comparison for $C_2 + C_b$ follows from Theorem 7.1 with $h = 1, q = 2$ and Theorem 2.2 with $K = 2$. Theorem 5.1 gives binary hardness with the additional three-copy bound on each margin. For explicit inputs $\mathcal{D}_2 \in \mathsf{NP}$, since one may guess a permutation on n states and verify the product by comparing its cycle lengths with those of B. Thus $\mathsf{NP} \subseteq \mathsf{QP}$ implies $\mathcal{D}_2 \in \mathsf{QP}$. Conversely, SAT reduces in polynomial time to three-edge-colourability of simple cubic graphs [16]. Theorem 6.1 reduces this problem to $\mathcal{D}_2$ in deterministic quasipolynomial time, including the time to write the successor tables. A quasipolynomial algorithm for $\mathcal{D}_2$ would give one for SAT, since composing two such bounds again gives quasipolynomial time. This proves (ii). Under $\mathsf{NP} \not\subseteq \mathsf{QP}$ we have $\mathsf{P} \neq \mathsf{NP}$, so (i) excludes NP-hardness and (ii) excludes even membership in QP. □

7 Counting quotients in binary cycle lists

Take $A = C_2 + C_b$ with $b > 1$ odd, and let u be a proper divisor of b. The first cycle of A gives the same product with $2C_u$ as with C_{2u}, namely $2C_{2u}$. The interval construction for explicit division uses the same identity with a prime-length cycle [20, Section 2, p. 4]. Here the second cycle tells which choice we made. It gives $2uC_b$ in the first case and uC_{2b} in the second. Thus a target can offer a separate choice for each of several weights u, then prescribe the sum of the weights selected.

Theorem 7.1. *Fix $h \ge 1$ and a prime $q \nmid h$. For binary cycle lists, counting quotient isomorphism classes is $\#\mathsf{P}$-complete under parsimonious polynomial-time reductions, even for divisors $C_{hq} + C_{hb}$ with $b > 1$ and $\gcd(q, b) = 1$. The corresponding decision problem is NP-complete. In particular, $h = 1$ and $q = 2$ suffice.*

We first identify every quotient of a target whose weights are given, then choose weights that encode exact covers.

Proof. Reading a subset from a quotient. Let $u_1, \dots, u_s$ be distinct positive proper divisors of b, where $q \nmid b$. Write $U = \sum_i u_i$, fix $0 \le S \le U$, and set

$$A = C_q + C_b, \qquad B = \sum_{i=1}^{s} qC_{qu_i} + qSC_b + (U - S)C_{qb}.$$

Assume each prime dividing h divides every u_i. Write S_h for multiplication of cycle lengths by h, preserving their multiplicities. Our output is $A' = S_h(A)$ and $B' = hS_h(B)$. The outer h multiplies the target multiplicities. For the $C_2 + C_b$ case, read the row calculation below with $h = 1$ and $q = 2$. For larger h, the issue is whether scaling introduces any additional quotient lengths.

Suppose $A'Y = B'$. Call the hq target cycles of length hqu_i row i. All come from C_{hq}, since $hb \mid hqu_i$ would imply $b \mid u_i$. A quotient length y contributing to row i therefore satisfies $\operatorname{lcm}(hq, y) = hqu_i$. For each prime $p \mid h$, the exponent of p in hqu_i exceeds its exponent in hq. Thus y supplies the full exponent, and $h \mid y$. Writing $y = ht$ leaves $\operatorname{lcm}(q, t) = qu_i$. Since q is prime and $q \nmid u_i$, only $t = u_i, qu_i$ work. Their contributions to the row are h and hq cycles per copy. Writing z_d for the multiplicity of C_d in Y, the row equation becomes $z_{hu_i} + qz_{hqu_i} = q$. Its two nonnegative integer solutions give the alternatives

$$qC_{hu_i} \quad \text{(select } i\text{)}, \qquad C_{hqu_i} \quad \text{(omit } i\text{)}.$$

The two choices use the same hqu_i states. A quotient cycle can supply only one row because its product with C_{hq} has a single length. Together the rows use hqU states. Since $|B| = q(q+b)U$, $|B'| = h^2|B|$, and $|A'| = h(q+b)$, the rows exhaust $|B'|/|A'|$. This excludes even a quotient cycle whose two products would both miss the rows. The length hqb is divisible by both divisor lengths, so Lemma 3.1 alone would not separate it. The exhaustion of the quotient states is what settles this shared length.

The second divisor cycle C_{hb} receives the choices from all rows. Each selected row contributes hqu_i copies of C_{hb} and each omitted row contributes hu_i copies of C_{hqb}. Thus the selected indices I give the prescribed target exactly when $\sum_{i\in I} u_i = S$. The bijection and its inverse are

$$Y_I = \sum_{i\in I} qC_{hu_i} + \sum_{i\notin I} C_{hqu_i}, \qquad I(Y) = \{i : z_{hu_i} = q\}.$$

Indeed, the u_i are distinct and none is divisible by q, so the $2s$ lengths hu_i, hqu_i are pairwise distinct. The formula specifies a cycle type, hence a single isomorphism class even when it uses q copies of one cycle. It works for every fixed h.

Making the selected weights form an exact cover. Our input consists of s distinct triples E_i drawn from $\{0, \dots, 3k-1\}$. The triples may overlap. Counting exact covers by triples is #P-complete under parsimonious reductions [17, Theorem 3.8 and Corollary 3.11]. For an empty ground set return $A_0 = B_0 = C_{hq} + C_{h(q+1)}$, with unique quotient C_1. If $s < k$, return A_0, C_1, with no quotient. We can now assume $1 \le k \le s$ before forming any powers. In his reduction from exact cover to knapsack, Karp uses base $s+1$ for s sets and one digit per element [19, p. 100]. Summing any subset of the weights counts incidences without carries, so the solutions of his 0–1 equation are precisely the exact covers. For the two cycles we also need weights coprime to q. The extra unit below makes each weight one modulo q and records the number of chosen triples. Let $Q = s + 2$ and $M = qQ$, and put

$$v_i = 1 + M \sum_{j\in E_i} Q^j, \qquad T = k + M \sum_{j=0}^{3k-1} Q^j.$$

In $\sum_{i\in I} v_i = T$, reduction modulo M forces $|I| = k$, since both integers lie between 0 and $s < M$. After subtracting k and dividing by M, the jth base-Q digit on the left counts the selected triples containing j. There are no carries because $s < Q$. Each target digit is one, so precisely the exact covers satisfy this equation.

Take $u_i = hv_i$, $S = hT$, and $b = (q+1)\prod_i u_i$. Distinct triples give distinct weights. Each u_i divides b properly and $q \nmid b$, since $q \nmid h$ and $v_i \equiv 1 \bmod q$. If $S > U$, return the same zero-solution instance as above. Otherwise the row construction gives exactly one quotient per exact cover. Each weight uses $O(1 + \log(hq) + k\log(s+2))$ bits, and the bit length of the

product defining b is at most the sum of the bit lengths of its factors. Hence all lengths and multiplicities are produced in polynomial bit time. The same construction reduces the decision version of exact cover to division. Lemma B.1 gives membership in NP.

Counting every quotient once. For any two divisor lengths a_1, a_2 of gcd h, a quotient length d produces two observed target lengths r, c. Comparing prime exponents gives

$$g = \gcd(r, c) = \gcd(\operatorname{lcm}(a_1, d), \operatorname{lcm}(a_2, d)) = \operatorname{lcm}(h, d), \qquad g/d = h/\gcd(h, d).$$

Thus for each pair of target lengths, allowing repetition, we need only test g/e for divisors e of both h and g. With m distinct target lengths this gives at most $m^2\tau(h)$ candidates. Sort the list and remove duplicates. If β is the largest target multiplicity, a quotient length can occur at most β times, as each copy contributes at least one cycle at a given output length. Guess exactly $\lceil \log_2(1+\beta) \rceil$ bits for each multiplicity, reject values above β, and check the whole product by the cycle rule. Since h is fixed, these steps take polynomial bit time. There is one accepting path for each quotient class, proving membership in #P. □

The divisibility condition on the weights prevents extra quotients under scaling. For example, $(C_3 + C_4)X = 3(C_3 + C_4)$ has just the quotient $3C_1$. With $h = 2$ and $u_1 = 1$, the scaled equation is $(C_6 + C_8)Y = 6(C_6 + C_8)$. Any partition of six states into cycles of lengths one and two works, giving four quotient classes.

The colouring construction of Section 4 also gives an exact count. A proper labelled edge-colouring in layer zero has two extensions on each connected component. In layer one the three colours at each vertex must undergo a fixed-point-free permutation, a shift by $+1$ or -1 modulo three. Neighbours must choose the same sign, and layer two then takes the remaining colour on each edge. Thus a graph G gives $2^{\mathrm{cc}(G)}\mathrm{Col}_3(G)$ quotient classes, where $\mathrm{Col}_3(G)$ counts its proper labelled three-edge-colourings. For K_4 the six colourings give twelve classes. Appendix C preserves these classes for other fixed gcds using $S_h(B)$ as target and leaving the quotient itself unchanged. Its proof uses the special multiplicities of that construction.

The restriction on the gcd matters even for writing the answer. If it can grow with binary input length L, the number of bits in the quotient count can reach $2^{\Theta(L/\log L)}$. Proposition C.2 gives a family attaining this bound and a matching upper bound for all binary cycle lists.

8 When is explicit division polynomial?

When one of the two lengths after division by their gcd has a bounded number of divisors, Theorem 2.2 already gives a polynomial algorithm for counting all quotient classes. Now allow both reduced divisor counts to grow and ask only for one quotient. Two size estimates determine exactly how fast SAT would have to be for this task to take polynomial time, as claimed in Theorem 1.1(iii).

Proof. From a SAT algorithm to a division decision. Suppose SAT on m-bit inputs takes time $m^{O(\log\log m)}$. Let $N = |A| + |B|$ be the number of states in a division input. Lemma 2.1 gives each block a SAT encoding whose bit length ℓ satisfies, for sufficiently large N and ℓ,

$$\log \ell = O(\log N/\log\log N), \qquad \log\log \ell = O(\log\log N).$$

Multiplying these estimates shows that the time per block is $\exp(O(\log N)) = N^{O(1)}$. There are at most N blocks, and division is possible exactly when every block is feasible.

From a division decision to a SAT algorithm. Suppose we can decide $\mathcal{D}_2$ in polynomial time. Convert an m-bit SAT instance in polynomial time to a simple cubic graph of size $s = m^{O(1)}$ whose three-edge-colourability gives the answer [16]. Theorem D.1 writes the two successor tables in $s^{O(\log\log s)}$ bit operations, so their total bit length is bounded by the same expression.

Running the division algorithm takes at most a fixed power of this time. The resulting SAT algorithm runs in $m^{O(\log\log m)}$ time.

Recovering the cycles of one quotient. First ask whether A divides B. On a positive answer, try quotient cycles C_t in order of length, up to $n = |B|/|A|$. To try one copy, form $P = AC_t$. If the current target contains all cycles of P with their multiplicities, subtract them and ask whether the remainder is divisible by the same A. An empty remainder succeeds immediately. On success, record C_t, keep the remainder and try the same length again. On failure, keep the current target and advance to the next length.

The test succeeds exactly when some quotient of the current target contains C_t, since $A(C_t + Y) = P + AY$. Once a length fails, it cannot become possible later. Restoring the cycles already removed would produce a quotient containing it at the earlier stage. A feasible nonempty target always has some quotient cycle, so the process ends with a complete quotient. In fact n decision calls suffice on a positive instance. If the output has r cycles and its last cycle has length t, at most $t-1$ lengths incur a failed oracle call. Only $r-1$ successful removals use the oracle, since the last leaves an empty target. Including the initial test gives at most $t + r - 1 \le n$ calls. Each query keeps the same divisor and has at most the original number of states. This argument applies to any explicit permutation divisor A. □

Whether SAT meets the bound in part (iii) is open. Two October 2026 preprints report deterministic algorithms for 3-SAT with subexponential running time in the number of variables [7, 25]. Even on 3-SAT, the stated bounds do not yield $m^{O(\log\log m)}$ time in the input bit length m. Parts (i) and (iii) both consider reductions from SAT to division. The construction used here has $m^{O(\log\log m)}$ time to write its tables. If a polynomial-time SAT algorithm could use a division oracle, part (i) would already give $\mathsf{P} = \mathsf{NP}$. In particular, if SAT misses the stated time bound, then $\mathsf{P} \neq \mathsf{NP}$, part (i) rules out NP-hardness, and part (iii) rules out membership in P. If SAT meets the bound, $\mathcal{D}_2$ belongs to P. This proves the claimed characterisation of when $\mathcal{D}_2$ is NP-intermediate.

For each fixed pair a, b, Theorem 2.2 already gives a polynomial-time algorithm. What would it take for one polynomial time bound to work as both lengths vary?

Corollary 8.1 (Division by cycles of consecutive lengths)**.** *Fix $h \ge 1$ and let $d \ge 2$ vary. Suppose the divisor is $C_{hd} + C_{h(d+1)}$ and the target is a nonempty permutation.*

(i) *With successor tables, a deterministic polynomial-time division test exists if and only if SAT on m-bit formulas can be decided in deterministic time $m^{O(\log\log m)}$. Such a division test suffices to construct one quotient in polynomial time.*

(ii) *With successor tables, the exact quotient count is computable in deterministic polynomial time if and only if the satisfying assignments of an m-bit formula can be counted in deterministic time $m^{O(\log\log m)}$.*

(iii) *With binary cycle lists, counting quotient isomorphism classes is $\#\mathsf{P}$-complete under parsimonious polynomial-time reductions, and deciding division is NP-complete.*

The two explicit equivalences also hold for arbitrary pairs of lengths and for the family in which neither length divides the other, with either a prescribed or an unrestricted greatest common divisor.

Proof. Changing the gcd on the instances of the graph construction. Theorem A handles any input at its own gcd. For the hard direction here the starting graph gives coprime lengths, while the output gcd h may be prescribed in advance. The separated table of Lemma 3.1 permits a direct construction with a smaller explicit size bound. Its row weights u_i divide b properly, its column weights v_j divide a properly, and its required multiplicities are R_i, S_j.

To prescribe the gcd, write $h = \alpha\beta$, placing in β the full powers of the primes of h that divide b. Then $U = \alpha a$ and $V = \beta b$ are coprime and exceed one. Take P to be the inverse of U

modulo V in $\{1,\dots,V-1\}$ and put $Q=(UP-1)/V$. We have $1\le Q<U$ and $UP=VQ+1$, so the new bases Pa, Qb are coprime. With R_i, S_j the original margins, output

$$A'=C_{hUP}+C_{hVQ},\qquad B'=\sum_i h\alpha R_i C_{h^2Pau_i}+\sum_j h\beta S_j C_{h^2Qbv_j}.$$

Here $d=VQ\ge 2$. We show that multiplying every old quotient length by h^2 gives a bijection on solutions.

Take a cycle of length y in any new quotient and let r,c be its two product lengths. Comparing prime exponents gives

$$\gcd(r,c)=\gcd\big(\operatorname{lcm}(hUP,y),\operatorname{lcm}(hVQ,y)\big)=\operatorname{lcm}(h,y).$$

Every cycle length in B' is a multiple of h^2, so h^2 divides both r and c. For each $p^e \parallel h$, $e\ge 1$, we therefore have $\max(e,\nu_p(y))\ge 2e$, which forces $p^{2e}\mid y$. Thus $y=h^2t$ and $y=\gcd(r,c)$. Now $r=h^2\operatorname{lcm}(Pa,t)$ and $c=h^2\operatorname{lcm}(Qb,t)$, because $\gcd(Pa,\beta)=\gcd(Qb,\alpha)=1$. The first length can hit only a row, since $Pa\nmid Qbv_j$, and the second only a column, since $Qb\nmid Pau_i$. Their gcd recovers $y=h^2u_iv_j$. One such cycle contributes $h\alpha v_j$ copies to its row and $h\beta u_i$ copies to its column. Cancelling the corresponding factor in each margin gives exactly the old table equations, also showing that every old table gives a solution.

The extended Euclidean algorithm finds P,Q in polynomial bit time. If the input tables have N states together, each row's contribution to the target size grows by $h^3\alpha P\le h^4N$, and each column's by $h^3\beta Q\le h^4N$. The divisor lengths grow by at most h^2N. Hence the output has at most h^4N^2 states, including on negative inputs. For fixed h both successor tables can be written in polynomial bit time. In binary lists the construction uses integers of polynomial bit length throughout.

Explicit decision and counting. For decision, apply the two implications proved above for Theorem 1.1(iii). Lemma 2.1 applies to every pair of divisor lengths. In the reverse direction, Theorem D.1 produces a weighted-table instance of Lemma 3.1. It is enough to change the divisor lengths on the instances produced by this reduction. Making them consecutive incurs polynomial overhead in the written tables and preserves every quotient class under the stated bijection. The construction of a witness keeps the divisor unchanged in every query.

For counting, encode each block of Lemma 2.1 by a Boolean circuit. Give each multiplicity a fixed number of bits, impose its upper bound, and check the equations. In the SAT encoding require the output to be true and impose an equivalence at every gate. Each feasible vector then has exactly one satisfying extension to the gate variables. The encoding is polynomial in the short block length, so a counting-SAT algorithm with the stated time bound takes $N^{O(1)}$ time per block. Multiply the counts of the independent blocks. These products have $O(N\log(2N))$ bits because there are at most N variables with at most $N+1$ choices each.

For the converse, Cai and Govorov prove #P-completeness for counting labelled three-edge-colourings of simple cubic graphs under polynomial-time Turing reductions [6, Theorem 1.1]. The empty graph has one colouring. For a nonempty queried graph G, suppose the routing in Section 6 inserts k diamonds. The two exterior edges of each diamond must have the same colour. Restore the missing edge of K_4 in that colour. Its opposite edge is then forced, and the remaining four edges form a cycle admitting exactly two alternating colourings. The routed graph therefore has $2^k\mathrm{Col}_3(G)$ labelled colourings and the same number of components as G. By Section 7 its table has exactly $2^{k+\mathrm{cc}(G)}\mathrm{Col}_3(G)$ quotient classes. The consecutive construction preserves that number. Exact division by the known power of two recovers $\mathrm{Col}_3(G)$, including when it is zero. On an m-bit counting-SAT input, the Turing reduction asks polynomially many questions about graphs of polynomial size. For each question, writing the successor tables by Theorem D.1, making the lengths consecutive and running a polynomial-time quotient counter together take $m^{O(\log\log m)}$ bit operations.

Binary counting. Theorem 7.1 supplies, for each prescribed h, a parsimonious reduction to binary division with divisor gcd h. Theorem A transforms each resulting instance in polynomial bit time, keeps that gcd and preserves every quotient class. The same fixed-gcd counting algorithm as in Theorem 7.1 proves membership in #P. Existence gives the corresponding NP-complete decision problem. □

A fixed number of distinct divisor lengths. For every fixed $r \geq 2$, the two explicit equivalences of Corollary 8.1 also hold for divisors with *at most* r distinct lengths and arbitrary multiplicities. Indeed the lower directions already use two cycles. For the upper directions $D \leq N^r$, so the extension of Lemma 2.1 in Section 2 applies, with one bounded variable per quotient length. Its gate encoding again preserves the count. The witness construction in this section keeps the divisor fixed.

What changes if the target has transient states?

Attach one leaf to each vertex of $C_2 + C_3$. This ten-state target is divisible by $C_2 + C_3$: the quotient is a fixed point with one leaf. Now move the two leaves of the 2-cycle onto the same vertex. A quotient would still have $10/5 = 2$ states. It must have a transient state, since the target does. The only such two-state system is a fixed point with one leaf, and its product puts one leaf at *every* cycle vertex. The position of the leaves therefore rules out a quotient.

Bridoux et al. already suggest using repeated tree patterns to decide division by a fixed permutation for targets with transient states [5, Section 6]. Porreca and Rolland construct one quotient in polynomial time when the shortest divisor length divides all the others [26, Theorem 61]. Here both divisor lengths may vary and be incomparable. We use the patterns to carry the count, the witness and the change of lengths over to these targets.

Corollary 8.2. *Let $A = C_a + C_b$ and let B be any nonempty finite dynamical system given by a successor table. The consecutive normal form of Theorem A still preserves all quotient classes, with $O(N^4)$ output states. The counting and witness algorithm of Theorem 2.2 retains its $(N+1)^{K+O(1)}$ bit bound. The explicit decision and counting equivalences in Corollary 8.1 hold for these targets as well.*

Proof. Delete the cycle edges of a connected component. At each cycle vertex a rooted tree remains, possibly just the root. Read the trees in successor order around the cycle. Their cyclic sequence is a repetition of a unique shortest word W, up to rotation. Choosing a different first vertex rotates the word. Once its image is chosen, the successor determines the images of the other cycle vertices. Thus W up to rotation and the cycle length t specify the component up to isomorphism. The admissible lengths are the multiples of $e = |W|$.

Multiplication by C_a gives $\gcd(a, t)$ components of length $\operatorname{lcm}(a, t)$, all with the same primitive word W. Indeed the first coordinate has a unique predecessor at every step, so following a tree backwards copies it unchanged. This is the repetition identity of Naquin and Gadouleau [24, Lemma 13 and Corollary 14], also stated by Porreca and Rolland [26, Lemma 13]. In particular, every word of a quotient occurs in the target. Compute tree types from the leaves upwards by sorting the types of the children. Comparing rotations and testing periods yields all the words in polynomial time.

For each observed W, keep only its target components and record their cycle lengths. Equation (1) now counts components of this word. Keep precisely the quotient variables with $e \mid t$. The equations for different words have disjoint variables, and each solution vector gives exactly one multiset of decorated components. Within a word, all state-balance calculations use *periodic* states. The short-query construction and the row dynamic program simply skip the excluded lengths. There are at most N words. Multiply their counts and assemble a witness by repeating each word around its chosen cycles. For the SAT-to-division implications, permutation targets

already suffice. In the opposite direction, a satisfying assignment for each short block can be recovered by fixing its bits successively.

For the normal form, use the same P, Q for all words, and apply Section 3.1 to each word's periodic lengths. Keep the decoration W in each output component. Its new cycle length is a multiple of its old one, so this is always possible. Could a new quotient cycle of length y carry W while its original length t cannot? In the block of label R, write $E = P_R Q_R$. The proof of Theorem A gives $t = y/E$. This length divides R and contains the full prime power of R at each prime dividing E. Thus $\gcd(E, R/t) = 1$ and $\gcd(y, R) = t \gcd(E, R/t) = t$. The period e divides R because it divides an observed cycle length in this block. It follows that $e \mid y$ if and only if $e \mid t$, even if e and E share a prime. The inverse therefore preserves admissible lengths.

Finally, let the trees in one copy of W have h states in total, including their roots. A component of cycle length r contains r/e copies of W, hence $(r/e)h$ states. The proof of Theorem A bounds the growth in periodic target states for this word by a factor of $PQ \max(P, Q)$. Multiplying both state counts by h/e gives the same bound for all its states, however large a single tree may be. Summing over the words gives $|B'| \le PQ \max(P, Q)|B|$. Together with the new divisor this gives the claimed $O(N^4)$ output size. □

Computational checks. The companion scripts and reference outputs test the block equations, diamond colourings and exact-cover construction. They compare quotient classes before and after the change to consecutive lengths, including cases where a row and a column share a length. For targets with trees, they test changes of the starting vertex and count the states after attaching a large tree to one cycle vertex. For Proposition C.3, they compare the two-by-two table with exhaustive search and check the queries used to recover prime factors.

A The arithmetic behind the explicit encoding

Proof of Lemma 6.2. Write $u(x) = \sum_{i \in x} 2^i$, the integer encoded by the address. To see what must be distinguished, consider two different addresses. If their signed coefficients have both sum zero and first moment $\sum_x b_x u(x) = 0$, both coefficients vanish. For up to H addresses we will read the first H moments from a single congruence.

Choose a base $P = M(1 + HC\,2^{dH})$ and a modulus $K = P^H$. Set $q = 1 + 10^8 dK$. For each i let δ_i be the least nonnegative residue of $(1+P)^{2^i} - 1$ modulo K, and put $z_i = q + \delta_i$. Multiplication of the factors for the bits gives

$$U(x) \equiv (1+P)^{u(x)} \pmod{K}.$$

In particular all $U(x)$ equal one modulo M. Since $0 \le \delta_i < K$, the ratio $U(x)/q^d$ is at most $(1 + 10^{-8}/d)^d < 1 + 10^{-7}$. The size of q gives the claimed logarithmic bound, and repeated squaring computes each residue.

In fact it suffices to have $\sum_x b_x U(x) \equiv 0 \pmod{K}$. Expanding the displayed congruence in powers of P yields

$$\sum_{j=0}^{H-1} P^j M_j \equiv 0 \pmod{P^H}, \qquad M_j = \sum_x b_x \binom{u(x)}{j}.$$

Each signed coefficient satisfies $|M_j| \le HC\,2^{dH} < P$. Reducing modulo P gives $M_0 = 0$. Dividing by P and repeating gives all $M_j = 0$. If $r \le H$ addresses have nonzero coefficients, the polynomials $\binom{u}{j}$ for $0 \le j < r$ span the polynomials of degree less than r over $\mathbb{Q}$. At any one of the r distinct points $u(x)$ we may therefore interpolate a polynomial that takes value one there and zero at the others. Its vanishing weighted sum gives $b_x = 0$ at that point. □

Checking the individual names in Theorem 6.1. In a relation of norm at most H, write γ_α for the signed coefficient of name α and μ_α for its full slot coefficient (ρ or ν). Collect the names at each address into a single coefficient b_x, so that

$$b_x = \sum_{\alpha \text{ at } x} \gamma_\alpha \mu_\alpha, \qquad \sum_x |b_x| \le C \sum_\alpha |\gamma_\alpha| \le CH.$$

At most H addresses contribute, so a relation $\sum_x b_x U(x) = 0$ gives $b_x = 0$ at each of them. Within one row address the scale W_0 separates the centre term from the slot sum of absolute value at most $30\,1024^{J_r-1} < W_0$. For a column address, first reduce modulo R. The sum of the signed offsets has absolute value at most $60 < R$, so it vanishes. Divide by R and separate the centre in the same way using V_0. On either axis the slot sum is now zero. Dividing by the smallest occupied power of 1024 and reducing modulo 1024 forces its coefficient to vanish, since its absolute value is at most 30. Hence there was no nonzero short relation.

Write q_c, q_r for the values of q on the two axes. The scales are $V = RV_0 q_c^d$ and $W = W_0 q_r^d$. Before multiplication by $U(x)$, each slot term is at most 1024^{-4} times RV_0 or W_0 on its axis. The column offset adds at most $2/(RV_0)$ to the relative error, and $1 \le U(x)/q^d < 1 + 10^{-7}$. This gives all the scale conditions of Lemma 4.1, including $0 < D_v < 10^{-5}V$. The difference of two distinct formal weights on either axis has norm at most 30. Independence therefore makes the weights numerically distinct, as required.

B Short witnesses for binary division

Constructing a quotient for one divisor length. With a single divisor length $A = \alpha C_u$, different target lengths can be treated separately. We give the single-length criterion of Dennunzio et al. [10, Theorems 3 and 4] in the notation of this paper. For βC_ℓ first require $u \mid \ell$. Write $r = \ell/u$ and $g = \gcd(u, r^{\lceil \log_2(u+1) \rceil})$. This extracts the full prime powers of u at the primes dividing r. If $\operatorname{lcm}(u,t) = \ell$, then $g \mid \gcd(u,t)$, and $t = rg$ attains equality and has the required lcm. Thus $\alpha g \mid \beta$ is necessary and sufficient. Taking $\beta/(\alpha g)$ copies of C_{rg} constructs that part of the quotient. A modular power and a gcd suffice, so this also gives a polynomial-time binary algorithm.

For two coprime divisor lengths, the list of candidate gcds in Section 3 already bounds the number of multiplicities we need to guess. For a general divisor an economical certificate comes from choosing the solution with the fewest distinct lengths.

Lemma B.1 (A short binary witness)**.** *For any binary cycle lists A, B, a positive instance has a quotient with at most $\sum_j \log_2(1+\beta_j)$ distinct cycle lengths, where β_j are the positive multiplicities in B. Binary division is in NP.*

Proof. Choose a quotient with as few distinct lengths as possible. For each used length t let $w(t)$ be the nonzero nonnegative vector of multiplicities in AC_t at the target lengths. Every subset sum of these vectors lies in the box $\prod_j \{0, \dots, \beta_j\}$. If two distinct subsets have the same sum, cancel their intersection. Both remaining subsets are nonempty. Let k be the least multiplicity in the first subset. Subtract k from each multiplicity in that subset and add k to each multiplicity in the other. The product is unchanged and a used length disappears. This contradicts minimality. Comparing the number of subsets with the box size proves the bound, the target-box bound of Eisenbrand and Shmonin [13, Lemma 3] applied to these product columns. The same exchange bounds the number of distinct lengths needed in a minimum-size realisation of prescribed prime marginals [22, Lemma 4.1, pp. 8–9]. Each used length divides a target length, and each multiplicity is at most $\max_j \beta_j$. The resulting binary list has polynomial bit length and can be verified by the gcd/lcm product rule. □

Proposition B.2 (One binary quotient from the reduced ratio)**.** *Let $A = C_a + C_b$ and B be given by binary cycle lists of total bit length L. Put $g = \gcd(a, b)$ and $k = \tau(a/g)\tau(b/g)$. A deterministic algorithm constructs a binary quotient when one exists, and detects infeasibility otherwise, in $f(k)L^{O(1)}$ bit operations for a computable function f.*

Proof. Suppose a quotient cycle produces lengths u and v under the two projections. We need only one representative length for this ordered pair. To find it, let $m_a(u)$ be the least t with $\operatorname{lcm}(a, t) = u$, defined when $a \mid u$. The single-length calculation above gives $m_a(u) = r \gcd(a, r^{\lceil \log_2(a+1) \rceil})$ with $r = u/a$. Every admissible t is a multiple of $m_a(u)$ because for each prime whose exponent in u exceeds its exponent in a, the full power in u must come from t. These are exactly the prime powers in $m_a(u)$. This least-length computation also appears in Porreca and Rolland [26, Theorems 20–23].

Try every ordered pair of observed lengths, including $u = v$, with $a \mid u$, $b \mid v$. Form $s = \operatorname{lcm}(m_a(u), m_b(v))$ and keep it if $\operatorname{lcm}(a, s) = u$ and $\operatorname{lcm}(b, s) = v$. If any t realises the pair, then $s \mid t$. Also $m_a(u) \mid s \mid t$, so $u \mid \operatorname{lcm}(a, s) \mid u$, and similarly for v. Now $\gcd(a, t) = at/u = (t/s)\gcd(a, s)$, with the analogous identity for b. Thus one C_t can always be replaced by t/s copies of C_s without changing either projection. Porreca and Rolland use a related replacement for pseudo-cancelable divisors [26, proof of Lemma 28]. Fixing the ordered output pair here gives the same multiplicity ratio t/s in both projections for arbitrary a, b. Different quotients may therefore have the same representative. For $A = C_4 + C_6$, both C_2 and $2C_1$ give the target $2A$.

Build the multiplicity equations only for the retained lengths, adding both contributions when $u = v$ and including every target equation. There are at most quadratically many candidate pairs in the binary list. The gcds and lcms take polynomial time in the binary input length. Each coefficient is at most $a + b$, even when both contributions go to the same equation. Write $D = \operatorname{lcm}(a, b)$ and group the equations as in Lemma 2.1, by $R = \operatorname{lcm}(D, s)$. For fixed R the row and column bounds in the proof of Theorem 2.2 give at most $\tau(b/g)$ possible first lengths and $\tau(a/g)$ possible second lengths. We kept only one representative per ordered pair, so the block has at most k variables. Grouping the observed pairs requires no factorisation of a or b. This is integer programming in fixed dimension [23]. Kannan's bound separates the dependence on the dimension from the polynomial in the binary input length [18, Theorem 5.4]. Since each block has dimension at most k and the entire list of blocks has polynomial encoding length, they take $f(k)L^{O(1)}$ bit operations in total. If only a feasibility oracle is used, recover each multiplicity by binary search between zero and the largest target multiplicity, fixing coordinates as they are recovered. Their union gives the required binary quotient. □

C Fixed and growing common divisors

Theorem C.1. *For every fixed $h \geq 1$, division of binary cycle lists is NP-complete for divisors $C_{ha} + C_{hb}$ with $a, b > 1$ and $\gcd(a, b) = 1$. In the reduction the quotient classes are in bijection with the tables of Lemma 4.1.*

Proof. The case $h = 1$ is Theorem 5.1. For $h > 1$ we will multiply each divisor and target length by h. Two properties of the original instance will preserve its quotients. Coprimality of each quotient length with h lets us carry a quotient forward. Multiplicity one will let us reverse the construction. A cycle whose length shares a factor with h splits into repeated cycles under the hth iterate.

Replace each row identifier $cW_0 + Q^j$ of (5) by $c(hW_0 + 1) + hQ^j$. The resulting identifiers have residues $1, 1, 1, 3$ modulo h for the four kinds of names p, g, r, T. All row weights, including $p + g - r$ and $T - p - g$, are therefore 1 modulo h. Their least common multiple b is coprime to h. In (6) use the multiplier hb in place of b, keeping the respective offsets $1, 2, 0$ on B, H, D. Every column weight is then 1 modulo hb, so its axis lcm a is coprime to both b and h. The

digit proof still works. On rows the low part of any tested relation has absolute value less than $hW_0 + 1$. On columns the sum of the offsets in a tested relation has absolute value at most $60 < hb$. Removing the centre or the offsets, respectively, leaves the same independent digits as before. The relative error windows continue to hold about the new scales $hW_0 + 1$ and hbV_0. The bit lengths remain polynomial for fixed h.

Let $A = C_a + C_b$ and B be the target constructed with these identifiers. By Lemma 3.1, the quotient lengths are among the distinct products $u_i v_j$, all coprime to h. By Lemma 4.1, each column of a feasible table uses three distinct rows. Thus its cells are 0 or 1, giving the second property as well.

Write $S_h(Z)$ for the result of multiplying each length in Z by h, and $I_h(Z)$ for the permutation obtained by taking the hth iterate. For example, $S_2(C_3) = C_6$ and $I_2(C_6) = 2C_3$. Output $S_h(A), S_h(B)$. If $AX = B$, each length d in X is coprime to h, so $C_{ha}C_d = S_h(C_aC_d)$ and likewise for b. Thus the same X solves the new equation. Conversely suppose $S_h(A)Y = S_h(B)$. Iteration commutes with products, and $I_h(S_h(Z)) = hZ$. Applying I_h and cancelling the common multiplicity h gives $AI_h(Y) = B$. Each C_t of Y contributes $\gcd(h,t)$ copies of $C_{t/\gcd(h,t)}$ to $I_h(Y)$. Every length in this quotient occurs just once, so $\gcd(h,t) = 1$ for every such t. Hence $Y \cong I_h(Y)$ and Y already solves the original equation. We have preserved the entire set of quotient classes. NP membership follows from Lemma B.1. □

Proposition C.2. *Among binary cycle-list inputs of length at most L, the maximum number of bits needed to write the quotient count is $2^{\Theta(L/\log L)}$. The lower bound is attained by divisors $A = C_{2g} + C_{3g}$. For the same divisor family Proposition B.2 constructs a binary quotient in polynomial bit time for every target that admits one, since each block has at most four representative variables. With successor tables, Theorem 2.2 also gives polynomial-time exact counting on the entire family.*

Proof. Every quotient length divides a target length, and every target length b satisfies $b \le 2^L$. To bound $\tau(b)$, write $b = \prod p^{e_p}$. Primes $p \le \sqrt{L}$ contribute at most $\sqrt{L}\log_2(L+1)$ to $\log_2 \tau(b) = \sum_{p|b} \log_2(e_p + 1)$. For larger primes, $\sum e_p \le 2L/\log_2 L$ and $\log_2(e_p + 1) \le e_p$. Thus even the union of the divisor sets of all target lengths has size $2^{O(L/\log L)}$. Each quotient multiplicity has at most $1 + 2^L$ possible values. Taking the logarithm of the product of these bounds gives the claimed upper bound on the number of answer bits.

The lower bound already occurs in a one-cycle example of Bridoux et al. [5, Appendix B, Lemmas 13–14], using the divisor-partition estimate of Erdős and Odlyzko [4]. For a primorial n, their count for $C_nX = nC_n$ needs at least $2^{\Omega(\log n/\log\log n)}$ bits. The two-cycle divisor $A = C_{2n} + C_{3n}$ preserves the entire solution set. Indeed, in $AX = nA$ the C_{2n} projection can only produce length $2n$, and the C_{3n} projection can only produce length $3n$. Every quotient length must therefore divide n, and $|X| = n$. Conversely, these two conditions give both $AX = nA$ and $C_nX = nC_n$ by the cycle product rule. They are also necessary for the latter equation. For the product of the first k primes, both inputs use $\Theta(\log n) = \Theta(k \log k)$ bits. Taking $k = \lfloor cL/\log L \rfloor$ for a sufficiently small fixed $c > 0$ gives the lower bound for every sufficiently large input budget L. In particular, when the gcd is unrestricted, the counting function has superpolynomial output length and lies outside #P. □

Proposition C.3 (Finding a second quotient)**.** *Let $A = C_{2g} + C_{3g}$, with $g \ge 1$ and A, B given by binary cycle lists. Deciding whether $AX = B$ has at least two isomorphism classes of quotients and factoring a positive integer given in binary are equivalent under deterministic polynomial-time Turing reductions. The same equivalence holds for deciding whether there is exactly one quotient class. One can factor g with $O(\log(2g))$ queries to either decision oracle. In the other direction, one factorisation of g suffices to construct two distinct quotients whenever they exist, with polynomial bit complexity apart from that call.*

Proof. Consider first $B = kA$, the disjoint union of k copies of A. For any nonempty permutation A, the quotients of $AX = kA$ are the partitions of k into divisors of the gcd of all cycle lengths of A. Indeed $|X| = k$. For any cycle length a of A, its ath iterate has $F_a(A) > 0$ fixed points. Since $F_a(A)F_a(X) = F_a(kA) = kF_a(A)$, we have $F_a(X) = k$, so every quotient length divides a. The cycle product rule proves the converse. The single-cycle version is the restricted-partition example of Bridoux et al. when $A = C_g$ and $k = g$ [5, Appendix B, Lemma 13].

For $A = C_{2g}+C_{3g}$ with $g > 1$, the all-ones partition is always available. The first k admitting a second partition is the least prime divisor of g. Every nonunit part is at least that prime, and a part of that size can be completed with ones. For example, when $g = 15$ the first change occurs at $k = 3$, with quotients $3C_1$ and C_3. For a remaining factor $h > 1$, use the divisor $A_h = C_{2h} + C_{3h}$. Query the targets kA_h at $k = 2, 4, 8, \ldots$, capping the last probe at h. The target at $k = 1$ has a unique quotient. The first probe admitting a second quotient brackets the least prime divisor p of h between the preceding value and the current one. The interval has length at most p, so binary search in it finds p in a total of $O(\log p)$ queries, including the probes. Either oracle detects the change since kC_1 is always a quotient. Divide h by p and repeat until $h = 1$. The primes found, with multiplicity, have product g. Summing their logarithms gives $O(\log(2g))$ queries in all, each of $O(\log(2g))$ bits.

The same first-change test with $A = C_g$ already recovers the prime factors from one-cycle division. We now go in the other direction. Given B, replace each cycle in a putative quotient by copies of the shortest cycle with the same ordered pair of image lengths, as in Proposition B.2. The resulting multiplicities form its *representative vector*. Two quotients may give different vectors, or they may give the same vector. We test each possibility.

Can the vector of representative multiplicities change? Split each block's row and column multiplicities by (2), rejecting a failed split. For a row length u in block R, we have $2g \mid u$ and $R = \operatorname{lcm}(u, 3g)$. Hence $R/u = 3g/\gcd(u, 3g)$ divides 3. Likewise R/v divides 2 for every column length v. There are therefore at most two rows and two columns, as in the proof of Theorem 2.2.

Measure each cell in *quotient states*. A quotient cycle C_t supplies $\gcd(2g, t)$ cycles of length $\operatorname{lcm}(2g, t)$ to its row, or $2gt$ image states for its t quotient states. Row i needs $u_i\rho_i$ image states for its ρ_i cycles of length u_i, so its sum in quotient states is $r_i = u_i\rho_i/(2g)$. In column j, the requirement of γ_j cycles of length v_j gives the sum $c_j = v_j\gamma_j/(3g)$. Use the quotas of this block, keeping the two shares at R separate. Pad a missing row or column with zeros. Reject nonintegral margins or unequal total sums.

For each attainable pair (u_i, v_j), let s_{ij} be its least representative, computed as in Proposition B.2. Its cell must contain a nonnegative multiple of s_{ij}: each copy of $C_{s_{ij}}$ supplies that many quotient states to both margins. An unattainable or padded cell must be zero. Conversely, dividing the occupied cells by their representatives recovers a feasible multiplicity vector. Once the upper left cell x is chosen, the table is forced:

$$\begin{pmatrix} x & r_1 - x \\ c_1 - x & r_2 - c_1 + x \end{pmatrix}, \qquad \max(0, c_1 - r_2) \le x \le \min(r_1, c_1).$$

Every attainable cell gives one congruence for x. For example, the upper right cell requires $r_1 - x$ to be divisible by s_{12} when that pair is attainable, so $x \equiv r_1 \pmod{s_{12}}$. The generalised Chinese remainder theorem either detects a conflict or combines these congruences into a single residue class. Intersect it with the displayed interval and with the equalities from the zero cells. The first admissible value and, if present, the next one can be found by integer division. The combined modulus divides R, since each s_{ij} divides both of its image lengths. All these integers have $O(L)$ bits for input bit length L. There are at most as many blocks as observed lengths. An empty block solution set rules out a quotient. Otherwise take the first value in each block. A second value in any block gives a second representative vector in polynomial bit time.

Can the same vector lift to two quotients? Suppose the representative vector z is unique.

For instance, with $A = C_{10} + C_{15}$ and $B = 25C_{30}$, both $5C_6$ and C_{30} give B. Each C_6 supplies five target cycles and C_{30} supplies all twenty-five, so both choices give $z_6 = 5$. When can shorter cycles be grouped in this way? For an ordered pair of output lengths u, v, write s for its least representative and $m = \gcd(u, v)$. The periods giving this pair are precisely the multiples of s dividing m: necessity was shown in Proposition B.2. For sufficiency, the lcm with the first divisor length lies between $\operatorname{lcm}(2g, s) = u$ and u in the divisibility order. The same argument applies to v. Set $H = m/s$. A cycle of length se, $e \mid H$, contributes exactly as much as e copies of C_s. Thus the lifts of z_s copies are exactly the partitions of z_s into divisors of H. Different ordered pairs have disjoint sets of periods, so a second lift exists precisely when, for at least one pair, $H > 1$ and the least prime divisor p of H satisfies $p \le z_s$. Replacing p copies of C_s by C_{ps} constructs it. Here $m = \gcd(\operatorname{lcm}(2g, s), \operatorname{lcm}(3g, s)) = \operatorname{lcm}(g, s)$, so $H = g/\gcd(g, s)$. One factorisation of g provides the primes to test for every pair. Each constructed list uses at most four representatives per block and one extra length $ps \mid m$. Its multiplicities have $O(L)$ bits, since each representative copy contributes at least one target cycle. This completes the bit bound for distinguishing zero, one and at least two classes and returning the witnesses. □

The same factoring equivalence holds for every fixed coprime pair of reduced lengths U, V, with $A = C_{Ug} + C_{Vg}$ and variable g. The targets kA give exactly the same divisor partitions. In the upper bound Proposition B.2 gives an integer program with at most $\tau(U)\tau(V)$ variables per block. Given one solution z, test each coordinate j in turn with the added bound $z'_j \le z_j - 1$ or $z'_j \ge z_j + 1$, allowing all other coordinates of that block to vary. These fixed-dimensional tests find any alternative representative vector. For every representative s the identity $\gcd(\operatorname{lcm}(Ug, s), \operatorname{lcm}(Vg, s)) = \operatorname{lcm}(g, s)$ still holds.

D A sharper bound for writing the successor tables

Theorem D.1. *For the same routed graph as in Theorem 6.1, one can choose the numerical weights so that both successor tables can be written in deterministic bit time* $\exp(O(\log(s + 2) \log\log(s + 3)))$*, with the same bound on the number of states written. All conclusions about the divisor, the colouring and the three-copy margins continue to hold.*

Recall that each graph address uses $d = O(\log(s + 2))$ bits. The two-bit calculation in Section 6 reduces the desired bound to a choice of small integers. If the factors q, z_i assigned to the bits are polynomial in d, the logarithm of each axis lcm is $O(d \log d)$. Two requirements constrain the choice. Short signed sums of address weights must distinguish the graph's local conditions, and the weights on the second axis must avoid every prime dividing the first lcm a. The next lemma meets the first requirement whenever each bit has enough candidates. The sieve later supplies candidates that satisfy the second requirement as well.

Lemma D.2. *Fix positive integers H, K and an integer $J > 3 \cdot 2^H$. For each of d bits take a disjoint pool of d^J integers greater than one, all pairwise coprime across the union of the pools. For all sufficiently large d and any integer $q > 0$, one can choose a z_i in each pool so that no relation $\sum_x b_x U(x) = 0$ has between one and H nonzero coefficients, all with $|b_x| \le K$. Here $U(x) = q^{d-|x|} \prod_{i \in x} z_i$. If the candidates and q are $d^{O(1)}$, deterministic search finds such a choice in* $\exp(O(d \log d))$ *bit operations.*

Proof. Choose independently and uniformly from the pools. In any relation take an inclusion-minimal vanishing subsum, keeping its original nonzero coefficients. It has ℓ distinct addresses with $2 \le \ell \le H$. For example, for addresses $110, 111, 001$ written in bit order $1, 2, 3$, a relation with coefficients u, v, w reads

$$z_1 z_2 (uq + v z_3) + w q^2 z_3 = 0.$$

Once z_3 is fixed this relation prescribes $z_1 z_2$. As no prime factor occurs in two pool entries, it even prescribes both choices separately. Bits 1 and 2 occur in exactly the same terms, the first two. In general we group bits by the set of terms in which they occur. Cancel the common factor from each bit on which all these addresses agree. There remain $k \geq 1$ bits. Each bit selects the nonempty proper subset of the ℓ terms in which it is one. Some subset occurs on a set F of at least $k/2^H$ bits. With all other choices fixed, the equation becomes

$$A_1 \prod_{i \in F} z_i + A_2 q^{|F|} = 0.$$

If $A_1 = 0$, the terms of that proper subset already sum to zero, contrary to minimality. Otherwise the equation prescribes the product uniquely. Each candidate has a prime factor shared by no other candidate, so this product prescribes every choice on F. The probability of this minimal relation is at most $d^{-Jk/2^H}$.

There are at most $(2K+1)^\ell \binom{d}{k} 2^{\ell k}$ possible lists of coefficients, surviving bit positions and bit patterns. Bits removed in the cancellation do not affect the event. Summing the probability bound over these lists and $2 \leq \ell \leq H$ gives at most

$$O_{H,K}\left(\sum_{k=1}^{d} (2^H d^{1-J/2^H})^k \right) = o(1).$$

For deterministic construction, try all d^{Jd} choices. For each, there are $O_{H,K}(2^{Hd})$ possible relations, and each can be tested using integers of $O(d \log d)$ bits. □

Proof of Theorem D.1. Use the routed graph, slots, coefficients ρ, ν and preliminary row lcm R of Section 6. Their slot counts, coefficient bounds and the possible values of R are bounded by absolute constants. Fix $H = 30$, choose a constant K at least H times the largest slot coefficient on either axis, and fix J as in the lemma. Already $H = 30$ requires $J > 3 \cdot 2^{30}$ in the search through d^{Jd} choices. The time bound is asymptotic. On each axis we will put $q < z_i \leq q(1 + \eta/d)$, with $\eta = 10^{-8}$. Then $1 \leq U(x)/q^d \leq e^\eta < 1 + 10^{-7}$. The lemma supplies the address independence used in Appendix A; the slot and scale calculations there apply with these new multipliers.

Column weights. Choose a fixed integer $E_0 > J + 3$. The prime number theorem in the progression 1 mod R supplies $\gg_R d^{E_0}/\log d$ primes in $[d^{E_0}, 2d^{E_0}]$ [3, Theorem 1.3]. Only finitely many R occur, so a common lower bound suffices. Partition the interval into $O_\eta(d)$ pieces of length at most $\eta d^{E_0}/d$. Some piece contains at least $1 + d^{J+1}$ of the primes for large d. Use its smallest prime as q, divide d^{J+1} of the others into pools and apply Lemma D.2. Since q and every z_i equal one modulo R, all column weights again equal one modulo R. Their lcm a is coprime to R.

Every column weight divides $q^d \prod_i z_i$ times the product of the distinct positive residual factors c and $c_1 q + c_2 z_i$ from Section 6. There are $O(d)$ such factors and each is $d^{O(1)}$. In particular $\log a = O(d \log d)$, and every prime dividing a is at most d^{C_1} for an absolute constant C_1.

Row weights. Fix $b_0 > \max(C_1, J + 3)$ and put $Z = d^{b_0}$. Choose a prime $q \in [d^E, 2d^E]$ for a sufficiently large fixed integer $E > b_0 + 1$ specified below, so $q > Z$. We seek d^{J+1} pairwise coprime integers in $I = (q, q(1 + \eta/d)]$ from which to form the row pools. Let $L = |I \cap \mathbb{Z}|$, so $L \asymp d^{E-1}$. We sieve out all small prime factors so that choosing one survivor will discard few others. At primes dividing a we must also protect the row weights. These are two separate steps. The sieve gives many admissible integers in the interval. A greedy selection among the survivors then ensures that no two chosen integers share a prime. For each prime $p \leq Z$ forbid the residue zero. When $p \mid a$, also forbid the zeros modulo p of each affine form $c_1 q + c_2 z$ that can occur in a row weight at two neighbouring addresses. Use all coefficient pairs that actually occur, in their order at the zero and one ends of the flipped bit. At $z = q$ this form has value

$(c_1 + c_2)q = \bar{u}_i q$, where $\bar{u}_i \mid R$ is that row's preliminary weight. Thus $q \bmod p$ avoids every forbidden residue, and none of the forms is identically zero modulo p. A one-address weight has residual factor $c = \bar{u}_i$, already coprime to a. The number $\rho(p)$ of forbidden residues therefore satisfies $\rho(p) \le \min(\kappa, p-1)$ for an absolute κ, independent of d and E.

Let E_p be the integers in I hitting a forbidden residue modulo p, put $g(p) = \rho(p)/p$, and write $V = \prod_{p \le Z}(1-g(p))$. Mertens' estimate for the prime reciprocal sum gives $V \gg_\kappa (\log Z)^{-\kappa}$ and, on any subinterval of primes from w to z, the inverse product is $\ll_\kappa (\log z/\log w)^\kappa$. Indeed for $p > 2\kappa$ each logarithm is at most $\kappa/p + O_\kappa(p^{-2})$, while the remaining primes contribute a bounded factor. Set $z = \lfloor Z \rfloor + \frac{1}{2}$ and $P(z) = \prod_{p<z} p$, so $p < z$ selects exactly the primes $p \le Z$. For $e \mid P(z)$ put $\rho(e) = \prod_{p \mid e} \rho(p)$ and extend g multiplicatively, so $g(e) = \rho(e)/e$. The Chinese remainder theorem gives $Lg(e) + r_e$ integers hitting every E_p with $p \mid e$, with $|r_e| \le \rho(e)$.

At sieve level z^u, the fundamental lemma gives the following lower bound for the number of survivors, for sufficiently large fixed $u \ge 9\kappa + 1$ [28, Corollary 19].

$$\#\{n \in I \cap \mathbb{Z} : n \notin E_p \text{ for all } p \le Z\} \;\ge\; \tfrac{1}{2}LV - O\left(\sum_{\substack{e \mid P(z) \\ e \le z^u}} |r_e|\right).$$

To use the divisibility formulation of Tao and Teräväinen [29, Lemma 5.1, p. 1338], associate to each integer of I the product of the primes at which it hits a forbidden class, counting repeated products with their multiplicities. Divisibility by e counts precisely the simultaneous hits above. The dimension bound depends only on κ, so u can be fixed before the interval exponent E is chosen. Since $\rho(e) \le \kappa^{\omega(e)} \le e^{\log_2 \kappa}$, the total remainder is $O_{\kappa,u}(Z^{u(1+\log_2 \kappa)})$. Now choose $E > 2 + b_0 u(1 + \log_2 \kappa)$. We obtain $\gg_\kappa L/(\log Z)^\kappa$ survivors.

Every survivor has only $O_{E,b_0}(1)$ prime divisors, all greater than Z. Choosing one survivor and removing all those sharing a prime with it costs $O_{E,b_0}(1+L/Z)$ candidates. Greedy selection therefore yields $\gg Z/(\log Z)^\kappa > d^{J+1}$ pairwise coprime integers for large d. Split this common supply into the d pools and use Lemma D.2. Every row weight is coprime to a, since its bit factors and residual factor avoid all $p \mid a$. The same common-multiple calculation gives $\log b = O(d \log d)$ for the row lcm b.

All primes required here can be found by trial division up to polynomial bounds in d, including the prime divisors of a. Bertrand's postulate supplies the row prime q. Scanning the intervals, testing forbidden classes and selecting the pools likewise take $d^{O(1)}$ bit operations apart from reading the routed graph. For small d, append unused address bits and repeat if a pool or a relation test leaves no admissible choice. The estimates above give a uniform threshold d_0 that works for every permitted choice of slots. Starting at d_{in} bits, the first successful trial uses at most $\max(d_{\text{in}}, d_0)$ bits. The search need not know d_0. Both exhaustive pool searches fit in $\exp(O(d \log(d+2)))$ time.

How many successors must we write? All the cycle lengths and multiplicities of the target are fixed by the routed graph. To count its states, take one triple h in Section 4. The weights of its first pair of main rows sum to $L_h = p_h + g_h$, as do the weights of its second pair. Including the incidence rows F_{ht}, the contribution of this triple to the weighted sum of row targets is

$$L_h \sum_{t \in h} B_t + L_h \sum_{t \in h} (H_t - B_t) + \sum_{t \in h} (T_t - L_h) H_t = \sum_{t \in h} T_t H_t.$$

Each element t occurs in three triples. On the column side the two weights $X_{t,c}$ and $Y_{t,c}$ add to H_t, both columns have target T_t, and there are three colours c. Thus $\sum_i u_i R_i = \sum_j v_j S_j = 3\sum_t T_t H_t$. Substituting these two sums into the state count of Section 6 gives

$$N = |A| + |B| = a + b + a\sum_i u_i R_i + b\sum_j v_j S_j = (a+b)\Big(1 + 3\sum_t T_t H_t\Big). \tag{7}$$

This calculation uses only the prescribed margins, so it applies even when the graph has no proper three-edge-colouring. There are $O(sd)$ elements t, and each of a, b, T_t, H_t has $O(d \log(d+2))$ bits. Hence $\log N = O(\log(s+2) + d \log(d+2))$. Writing a successor index for each state in both tables takes $O(N \log(2N))$ bit operations. Since $d = O(\log(s+2))$, this fits in the claimed $\exp(O(\log(s+2) \log\log(s+3)))$ time. Lemmas 4.1 and 3.1 give the colouring equivalence and the three-copy condition. □

The number of states required by this encoding. There is a matching lower bound on the sizes of these successor tables. On each axis the construction assigns a distinct weight to each edge of G', as observed in Section 6. Each row weight divides b, and each column weight divides a. Thus for $m = a$ and for $m = b$ we have $\tau(m) \geq |E(G')| \geq |E(G)| = 3s/5$. In particular $m \geq 3s/5$. Using the divisor estimate of Lemma 2.1 in the form $\log \tau(m) = O(\log m / \log\log m)$, we obtain $\log m = \Omega(\log s \log\log s)$ as s grows. The explicit divisor alone therefore has $\exp(\Omega(\log s \log\log s))$ states. Combined with the upper bound, the total size is $\exp(\Theta(\log s \log\log s))$. This lower bound applies to encodings that retain a distinct divisor weight per edge on an axis.

References

[1] E. Allender and M. Koucký. Amplifying lower bounds by means of self-reducibility. *Journal of the ACM* 57(3), article 14, 2010. Publisher. Author's copy, Theorem 5.1(3).

[2] R. Ascone, A. E. Porreca, and M. Rolland. Polynomial equations over bijective dynamical systems and their semiring extensions. Extended abstract, Workshop on Cellular Automata and Boolean Automata Networks, UCNC 2026. Extended abstract.

[3] M. A. Bennett, G. Martin, K. O'Bryant, and A. Rechnitzer. Explicit bounds for primes in arithmetic progressions. *Illinois Journal of Mathematics* 62, 427–532, 2018. Published article.

[4] D. Bowman (problem), P. Erdős and A. M. Odlyzko (solution). Problem 6640 and solution. *American Mathematical Monthly* 99 (1992), 276–277. Published solution.

[5] F. Bridoux, C. Crespelle, T. H. D. Phan, and A. Richard. Dividing sum of cycles in the semiring of functional digraphs. *Natural Computing* 25, article 11, 2026. Publisher. arXiv:2504.11943v2. The decomposition lemma also appears as Lemma 2 in their AUTOMATA 2024 conference paper, *Dividing permutations in the semiring of functional digraphs*, LNCS 14782, 95–107.

[6] J.-Y. Cai and A. Govorov. The complexity of counting edge colorings for simple graphs. *Theoretical Computer Science* 889, 14–24, 2021. Publisher.

[7] Claude (Anthropic). A $2^{o(n)}$ time algorithm for 3-SAT. Hexagon preprint, 27 October 2026. `hexagon:2610.00577v1`.

[8] C. Crespelle, T. H. D. Phan, and A. Richard. Tensor product of functional graphs: algorithms for factorization and polynomial root-finding restricted to the cycle part. Preprint, 2025. HAL version 1.

[9] W. H. Cunningham and J. Geelen. On integer programming and the branch-width of the constraint matrix. *IPCO 2007*, LNCS 4513, 158–166. Authors' copy.

[10] A. Dennunzio, E. Formenti, L. Margara, and S. Riva. On solving basic equations over the semiring of functional digraphs. *Discrete Mathematics & Theoretical Computer Science* 27(3), article 6, 2025. Publisher.

[11] F. Doré, K. Perrot, A. E. Porreca, S. Riva, and M. Rolland. Infinite trees for division and roots over finite discrete-time dynamical systems. *Natural Computing* 25, article 16, 2026. Publisher.

[12] A. W. M. Dress and C. Siebeneicher. The Burnside ring of the infinite cyclic group and its relations to the necklace algebra, λ-rings, and the universal ring of Witt vectors. *Advances in Mathematics* 78 (1989), 1–41. Publisher.

[13] F. Eisenbrand and G. Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters* 34, 564–568, 2006. Publisher.

[14] F. V. Fomin, F. Panolan, M. S. Ramanujan, and S. Saurabh. On the optimality of pseudo-polynomial algorithms for integer programming. *Mathematical Programming* 198, 561–593, 2023. Publisher.

[15] M. Gadouleau and M. Johnson. Semirings of formal sums and injective partial transformations. Preprint, 2026. arXiv:2603.26508v1.

[16] I. Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing* 10, 718–720, 1981. Publisher.

[17] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, and R. E. Stearns. The complexity of planar counting problems. *SIAM Journal on Computing* 27, 1142–1167, 1998. Publisher.

[18] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research* 12(3), 415–440, 1987. Publisher.

[19] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, 85–103, 1972. Publisher.

[20] N. Kuosa. Division of Explicit Finite Dynamical Systems Is NP-Complete. Hexagon preprint, 2026. `hexagon:2611.00886v1`.

[21] N. Kuosa. Finding Two Connected Factors of a Finite Dynamical System in Quasipolynomial Time. Hexagon preprint, 2026. `hexagon:2611.00849v1`.

[22] N. Kuosa. How Many Cycle Lengths Can a Unique Quotient Require? Hexagon preprint, 2026. `hexagon:2611.00945v1`.

[23] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research* 8 (1983), 538–548. Publisher.

[24] É. Naquin and M. Gadouleau. Factorisation in the semiring of finite dynamical systems. *Theoretical Computer Science* 998 (2024), 114509. Publisher.

[25] OpenAI. The exponential time hypothesis is false. Preprint, revised 25 October 2026. Manuscript and accompanying materials.

[26] A. E. Porreca and M. Rolland. Injective and pseudo-injective polynomial equations: from permutations to dynamical systems. 2026. arXiv:2604.04065v2.

[27] M. Rolland. *Des unrolls aux systèmes dynamiques totaux et partiels: algorithmes de résolution et énumérations des (in)équations polynomiales*. Doctoral thesis, Aix-Marseille Université, 2026, NNT 2026AIXM0179. Thesis.

[28] T. Tao. 254A, Notes 4: Some sieve theory. 21 January 2015. Corollary 19.

[29] T. Tao and J. Teräväinen. Quantitative bounds for Gowers uniformity of the Möbius and von Mangoldt functions. *Journal of the European Mathematical Society* 27, 1321–1384, 2025. Publisher. Lemma 5.1, p. 1338.