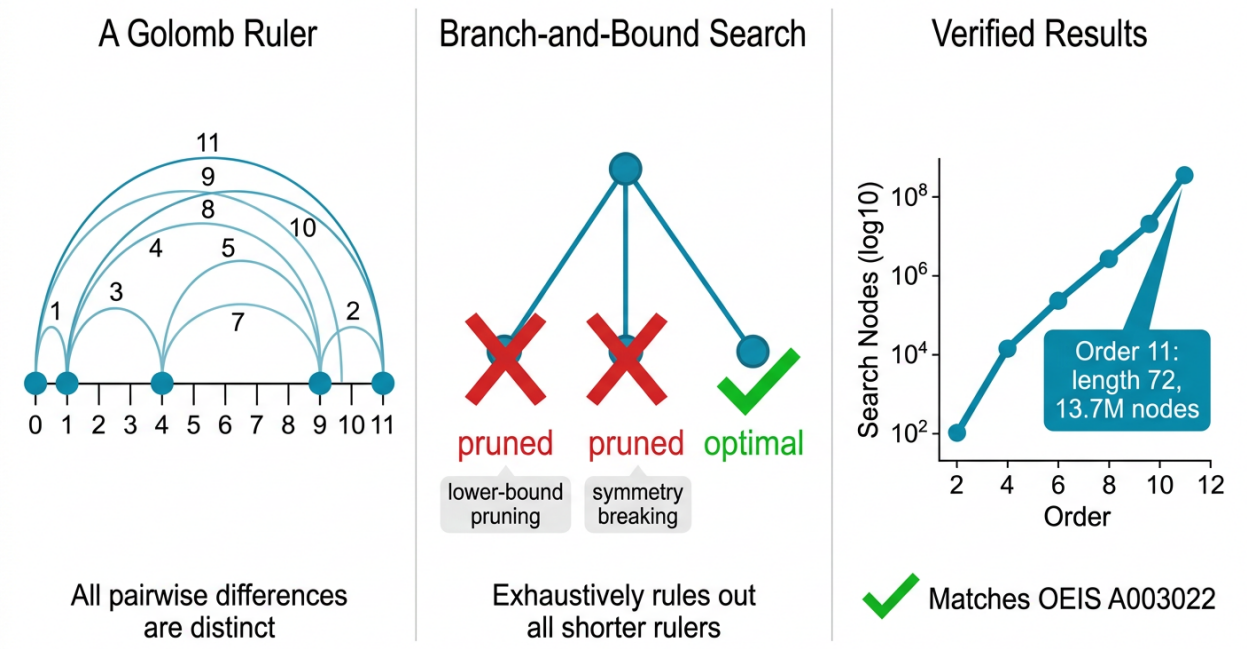
Constructing and Proving Optimal Golomb Rulers up to Order 11

A Reproducible Branch-and-Bound Study with Full Search-Tree Accounting

July 2026



Graphical abstract. A Golomb ruler is a set of integer marks whose pairwise differences are all distinct. We prove the minimum possible length for every order from 2 to 11 with an exhaustive depth-first branch-and-bound search that certifies optimality by ruling out all shorter candidates. Every reported length matches the published optimum (OEIS A003022); the largest instance (order 11, length 72) is settled by examining 1.37×10^7 search nodes.

Abstract

A Golomb ruler is a set of integer marks whose pairwise differences are all distinct; it is optimal for its order—the number of marks—if no shorter ruler of that order exists. Establishing optimality is far harder than finding a short ruler, because it requires certifying that the entire space of shorter candidates is empty. We present a self-contained study that constructs and proves optimal Golomb rulers for every order from 2 to 11 using a recursive depth-first branch-and-bound search. The search is hardened with three standard accelerations—a reflection symmetry break that discards mirror-image rulers, an arbitrary-precision bitmask that tracks used pairwise differences in constant time per test, and lower-bound pruning in which the

proven optimal length of each smaller order tightens the search for the next—and it is fully instrumented to count every candidate configuration examined and to measure wall-clock runtime. For each order we report the optimal length, one optimal set of marks, the number of search-tree nodes examined, and the runtime. Optimality is certified by the completed exhaustive traversal rather than recited from tables: the known optimal lengths are used only as an independent check applied after each search terminates. All ten optimal lengths (1, 3, 6, 11, 17, 25, 34, 44, 55, 72) match the published values in OEIS A003022 exactly, with no discrepancies, and every returned ruler is re-verified to be a valid Golomb ruler. The search cost grows exponentially, by an average factor of about 7.6 per order, spanning seven orders of magnitude in examined nodes and eight in runtime; the largest instance, order 11 (length 72), is settled by examining 1.37×10^7 configurations in roughly 145 seconds. The highest order for which optimality is proven is 11.

Contents

1 Introduction

A *Golomb ruler* is a set of integer marks placed on a line so that no two pairs of marks are separated by the same distance; equivalently, every pairwise difference between marks is distinct. Named after Solomon W. Golomb, whose work on combinatorial designs and shift-register sequences popularised the structure [4, 15], the object is identical, up to translation, to a finite *Sidon set* or B_2 sequence studied decades earlier in additive number theory by Sidon [28] and Erdős and Turán [12]. A ruler is called *perfect* if its differences realise every integer from one up to its length exactly once; such rulers exist only for order four or fewer, and for all larger orders one instead seeks the *optimal* ruler: the one of minimum possible length. Determining that length, and certifying that no shorter ruler of the same order can exist, is a deceptively hard computational problem that is the subject of this report.

The practical appeal of Golomb rulers stems directly from their defining property. Because every pairwise distance is unique, placing physical resources at ruler marks produces conflict-free spacings in whichever domain the marks are interpreted. The earliest engineering use, due to Babcock [2], assigned radio channel frequencies to ruler marks so that third-order intermodulation products generated by nonlinear receivers never coincide with an in-band channel; the same idea underlies carrier-frequency assignment for nonlinear repeaters [1]. In radio astronomy and array signal processing, antennas positioned at ruler marks form minimum-redundancy linear arrays whose baselines sample distinct spatial frequencies, maximising the information a fixed number of elements can extract about the sky [4, 22]. The construction further appears in the design of convolutional self-orthogonal error-control codes [25], in the two-dimensional Costas arrays used for radar waveform design [5], and in sensor placement, X-ray crystallography, and wavelength allocation for optical networks where four-wave-mixing crosstalk must be suppressed; comprehensive surveys of these applications, the available construction methods, and the Sidon-set connection are given by Dimitromanolakis [7] and Drakakis [10]. A recurring theme across these domains is that only rulers of modest order are needed in practice, but that knowing the *optimal* ruler for that order is valuable because it minimises the physical span—bandwidth, aperture, or cable length—required to realise the design.

Finding an optimal Golomb ruler is easy to state but combinatorially explosive to solve. There is no known polynomial-time construction that yields the shortest ruler for arbitrary order, and while the general optimisation problem has not been proven NP-hard in its canonical form, several closely related decision variants—such as deciding whether a given set contains a Golomb sub-ruler of prescribed size—are known to be NP-hard [21]. Consequently every optimal ruler beyond the smallest orders has been established by explicit computer search rather than by formula. Historic milestones include Shearer's exhaustive computations of the optimal rulers for orders fourteen through sixteen [27], the parallel search of Dollas, Rankin, and McCracken that settled order nineteen [9, 24], and the massively distributed volunteer-computing campaigns of `distributed.net`'s Project OGR, which have certified optimality up to order twenty-eight over years of aggregate runtime [8]. The canonical list of proven optimal lengths is curated as sequence A003022 in the On-Line Encyclopedia of Integer Sequences [23]. The difficulty is not in *finding* a short ruler—good heuristics and constraint solvers do that quickly—but in *proving* that no shorter ruler exists, which requires exhaustively eliminating, directly or by pruning argument, every candidate of smaller length.

This report documents a self-contained study that constructs and proves optimal Golomb rulers for every order from 2 to 11. We implement a recursive depth-first branch-and-bound search hardened with three standard accelerations—a reflection symmetry break, bitmask-based difference tracking, and lower-bound pruning driven by previously proven orders—and we instrument it to

count every candidate mark placement examined and to measure wall-clock runtime. The rulers reported here, and the optimality certificates that accompany them, are produced entirely by the program: the known optimal lengths from A003022 are used only as an independent check on the answers, never as an input that shortcuts the proof. For each order we report the optimal length, one optimal set of marks, the number of search-tree nodes examined, and the runtime, and we confirm that all ten lengths match the published values. We then analyse how the search cost scales, observing a roughly six- to eightfold growth in examined nodes per unit increase in order, punctuated by a sharper seventeenfold jump entering order 11, where the proof required examining 1.37×10^7 candidate configurations in about 145 seconds. The highest order for which optimality is proven here is 11.

The remainder of the report is organised as follows. Section 2 formalises Golomb rulers, optimality, and the elementary bounds that frame the search. Section 3 describes the branch-and-bound algorithm and each of its accelerations in detail, with pseudocode. Section 4 presents the complete results table for orders 2–11, the verification against A003022, and an ablation of the pruning components. Section 5 analyses the empirical growth of the search tree and runtime. Section 6 situates the findings against the historical record and discusses limitations, and Section 7 concludes. Appendix A gives the reproducibility details and the exact instrumentation used to account for the search.

2 Problem Definition and Preliminaries

We begin by fixing notation and stating the elementary facts that the search exploits. Throughout, marks are non-negative integers and rulers are written in increasing order with the first mark normalised to zero.

Definition 1 (Golomb ruler)**.** *A* Golomb ruler of order m *is a set of* m *integers* $\{a_1, a_2, \ldots, a_m\}$ *with*

$$0 = a_1 < a_2 < \cdots < a_m$$

such that the $\binom{m}{2}$ *positive pairwise differences* $a_j - a_i$ *for* $1 \le i < j \le m$ *are all distinct. The* length *of the ruler is its largest mark,* $L = a_m$*.*

The requirement that all $\binom{m}{2}$ differences be distinct is what makes the object rigid: a ruler of order m "uses up" $\binom{m}{2}$ distinct positive integers as differences, and since the largest possible difference is the length L, no Golomb ruler of order m can be shorter than $\binom{m}{2}$. This gives the trivial lower bound

$$L(m) \;\ge\; \binom{m}{2} \;=\; \frac{m(m-1)}{2}, \tag{1}$$

which is attained with equality only by the four *perfect* rulers of orders one through four. For orders five and above no perfect ruler exists, so the optimal length strictly exceeds the triangular bound and must be found by search. Normalising $a_1 = 0$ costs no generality because translating every mark by a constant preserves all differences; the only remaining freedom is reflection, discussed below.

Definition 2 (Optimal Golomb ruler)**.** *A Golomb ruler of order* m *is* optimal *if its length equals*

$$L^*(m) \;=\; \min\{\, a_m : \{0, a_2, \ldots, a_m\} \text{ is a Golomb ruler of order } m \,\}.$$

Proving that a ruler is optimal therefore requires two things: exhibiting a valid ruler of length $L^*(m)$*, and certifying that* no *Golomb ruler of order* m *has length strictly less than* $L^*(m)$*.*

The second obligation is the crux of the problem and the reason the task is computationally demanding. Exhibiting a short ruler is a matter of luck or a good heuristic, but certifying optimality is a statement about the entire (finite but astronomically large) space of candidate rulers of a given length. Our search discharges this obligation constructively: it enumerates, with aggressive pruning, every length value from the trivial lower bound upward and, for each, either finds a ruler or proves none exists, stopping at the first length that admits one.

Two structural observations underpin the accelerations of Section 3. First, **reflection symmetry**: if $\{0, a_2, \ldots, a_m\}$ is a Golomb ruler of length L, then its reflection $\{0, L - a_{m-1}, \ldots, L - a_2, L\}$ is also a Golomb ruler of the same length, since reflection merely relabels each difference d as d. Every ruler thus comes paired with its mirror image, and the two are distinct unless the ruler is palindromic—which, for a Golomb ruler, would force two gaps to be equal and hence is impossible for $m \geq 3$. Restricting the search to one representative of each mirror pair therefore removes exactly half of the solution space without losing any optimum; the finer combinatorial structure underlying such counting—for instance, the number of Golomb rulers of a given length—has itself been enumerated exactly via a correspondence with acyclic orientations of graphs [11]. The standard canonical choice requires the first gap to be strictly smaller than the last,

$$a_2 - a_1 \;<\; a_m - a_{m-1}, \tag{2}$$

and exactly one of a ruler and its reflection satisfies inequality (2).

Second, **sub-ruler monotonicity**: any contiguous suffix of the marks of a Golomb ruler is itself a Golomb ruler of smaller order. If r marks remain to be placed from the current position to the end of the ruler, those r marks form a sub-ruler whose span is bounded below by the optimal length of a ruler of order r—a quantity we already know exactly for every order below the one currently being solved, and which otherwise falls back to the triangular bound (1). This is precisely the relationship that lets the optimal length of order $m - 1$ tighten the search for order m, and it is the source of the lower-bound pruning described next. The relationship between Golomb rulers and Sidon sets also imports asymptotic size bounds from additive combinatorics—the near-quadratic growth $L^*(m) \sim m^2$ is consistent with the Erdős–Turán and Lindström bounds $f(N) = \sqrt{N} + O(N^{1/4})$ on the size of the largest Sidon subset of $\{1, \ldots, N\}$ [3, 12, 20]—but these asymptotics inform expectations rather than the exact small-order proofs pursued here.

3 Methodology: Branch-and-Bound Search

Our solver certifies the optimal length of each order by a recursive depth-first branch-and-bound search over partial rulers. Branch-and-bound is the classical framework for exact combinatorial optimisation [18, 19]: the search maintains an incumbent (the best complete solution found so far) and explores a tree of partial assignments, discarding any subtree that provably cannot improve upon the incumbent. Realised as depth-first backtracking with state changes undone on ascent, it is the structured-search backbone common to exact combinatorial solvers [16, 26]. For the Golomb ruler problem the tree is the set of prefixes $\{0, a_2, \ldots, a_k\}$ that are themselves valid Golomb rulers; a node at depth k represents having committed to the first k marks, and its children correspond to the admissible choices for mark $k + 1$. The approach is well established for this problem, having been formalised as a constraint-satisfaction model by Smith, Stergiou, and Walsh [30] and as a dedicated constraint-programming search by Galinier et al. [13]; our implementation follows the same principles while remaining a compact, self-contained program whose every pruning decision is instrumented and counted.

3.1 Search structure and incumbent seeding

The search solves orders in increasing sequence, from 2 up to the target, because each proven optimum tightens the pruning available to the next order (Section 3.4). For a given order m the recursion places marks left to right. Before the search begins, a greedy pass constructs a valid—generally sub-optimal—ruler by repeatedly appending the smallest integer that keeps the set Golomb; the length of this greedy ruler seeds the incumbent so that pruning is active from the very first node rather than only after the first complete solution is discovered. Each time the search reaches a complete ruler shorter than the incumbent, the incumbent length is lowered and the search continues, now pruning against the stronger bound. When the tree is exhausted, the final incumbent is optimal: every ruler of every smaller length has been examined and rejected, either explicitly or through a sound pruning rule, so the certificate of optimality is the completed traversal itself.

3.2 Fast difference tracking with a bitmask

The inner loop of the search asks, repeatedly, whether placing a new mark at position p would repeat any pairwise difference already realised by the marks already placed. A naive check recomputes and compares a set of differences at every node, which is both slow and allocation-heavy. Instead we maintain the set of used differences as a single arbitrary-precision integer interpreted as a bitmask: bit d is set if and only if the difference d is already in use. Testing whether a candidate mark p is admissible then reduces to forming, for each existing mark q, the difference $d = p - q$ and checking the single bit $1 \ll d$; the differences a mark would introduce are simultaneously accumulated in a temporary mask so that collisions *among* the new differences (for example, if $p - q_1 = q_2 - q_3$ is already pending) are also caught in the same pass. If no collision occurs, the temporary mask is OR-ed into the running mask on descent and XOR-ed back out on backtracking, giving $O(1)$ membership tests and $O(k)$ work to place the k-th mark. Because Python integers are unbounded, the mask grows transparently with the ruler length and never overflows, at the cost of big-integer arithmetic whose word count grows with L—a factor we return to in Section 5. Listing 1 shows the core feasibility test.

```
add = 0
ok = True
for q in marks: # marks already placed
    d = p - q
    bit = 1 << d
    if (used_mask & bit) or (add & bit):
        ok = False # difference d already used (or pending)
        break
    add |= bit # remember d as introduced by this mark
# on success: used_mask |= add (descend); later used_mask ^= add (backtrack)
```

Listing 1: Bitmask feasibility test for placing a mark at position `p`.

3.3 Symmetry breaking

To avoid searching both a ruler and its mirror image, we enforce the canonical inequality (2) that the first gap be strictly smaller than the last. Rather than generate a ruler and then reject it if it violates the inequality—which would still pay the cost of visiting the mirror subtree—we impose the constraint as a range restriction that skips the mirror image outright. This is realised at two points in the recursion, mirroring the general principle that structural symmetry-breaking constraints are

most valuable when they prune whole subtrees early rather than filtering complete assignments [6, 14]. First, when the final mark is placed, its position is required to exceed $a_{m-1} + a_2$, which is exactly the algebraic restatement of "last gap > first gap." Second, and more powerfully, at the second mark (a_2) the search caps the position at $\lfloor (L_{\text{best}} - 1)/2 \rfloor$: since the reflected ruler would place its own first gap at $L - a_{m-1} \geq a_2$ and we only pursue rulers that strictly improve on the incumbent, no admissible improving ruler can have a first gap exceeding half the incumbent length. Trimming the range of a_2 prunes entire internal subtrees before they are entered.

3.4 Lower-bound pruning

The decisive accelerator is a lower bound on the length of any completion of the current partial ruler. Suppose the search has placed marks up to position $a_k = \texttt{last}$ and $r = m - k$ marks remain (including the one about to be placed). Those r trailing marks, taken by themselves, must form a Golomb ruler of order r; by sub-ruler monotonicity their span is at least $L^*(r)$, the optimal length of order r, which the incremental solver has already proven for every $r < m$. Hence any completion has length at least $\texttt{last} + L^*(r)$. Turning this around, if the search is to produce a ruler strictly shorter than the current incumbent L_{best}, the mark being placed cannot exceed

$$p_{\max} \;=\; L_{\text{best}} - 1 - L^*(r), \tag{3}$$

which bounds the loop over candidate positions and prunes the tail of the search space at every node. When the optimal length of the required sub-order is not yet available—only the case of the fallback path—the triangular bound $\binom{r}{2}$ of (1) is substituted, which is weaker but still sound. Because the solver works upward through the orders, the tight bound $L^*(r)$ is in fact always available for the trailing sub-rulers encountered while solving order m, so pruning operates at full strength throughout.

3.5 The complete algorithm

Algorithm 1 assembles these components. The routine Recurse places the mark at a given depth; the bounds $p_{\max}$ from (3) and the symmetry restrictions jointly define the range of admissible positions, the bitmask test filters that range for Golomb-validity, and each admissible non-final position spawns a child. A counter is incremented once per admissible candidate examined; this count is the search-tree accounting reported in Section 4, and it is the quantitative form of the optimality proof, since it enumerates exactly how many configurations had to be inspected to rule out every shorter ruler.

Algorithm 1 Branch-and-bound search for the optimal Golomb ruler of order m.

$L_{\text{best}} \leftarrow$ length of greedy ruler; `best` $\leftarrow$ greedy ruler
`marks` $\leftarrow [0]$; `mask` $\leftarrow 0$; `nodes` $\leftarrow 0$
procedure RECURSE(depth)
    `last` $\leftarrow$ `marks`$[-1]$; $r \leftarrow m - \text{depth}$ ▷ marks from here to end
    $p_{\max} \leftarrow L_{\text{best}} - 1 - \text{LB}(r)$ ▷ lower-bound pruning, Eq. (3)
    if depth $= 1$ **then** $p_{\max} \leftarrow \min(p_{\max}, \lfloor (L_{\text{best}} - 1)/2 \rfloor)$ ▷ symmetry on a_2
    end if
    $p_{\text{start}} \leftarrow$ `last` $+ 1$
    if depth $= m - 1$ **then** $p_{\text{start}} \leftarrow \max(p_{\text{start}}, \texttt{last} + \texttt{marks}[1] + 1)$ ▷ last gap $>$ first gap
    end if
    for $p \leftarrow p_{\text{start}}$ **to** $p_{\max}$ **do**
        if p introduces no repeated difference (bitmask test) **then**
            `nodes` $\leftarrow$ `nodes` $+ 1$
            if depth $= m - 1$ **then**
                $L_{\text{best}} \leftarrow p$; `best` $\leftarrow$ `marks` $+ [p]$ ▷ improving complete ruler
            else
                push p onto `marks`; update `mask`; RECURSE(depth $+ 1$); undo
            end if
        end if
    end for
end procedure
RECURSE(1)
return $(L_{\text{best}}, \texttt{best}, \texttt{nodes})$

An important honesty condition governs the use of prior knowledge. The lower bounds $L^*(r)$ fed into LB are only ever those of orders *strictly smaller* than the order currently being solved, and each such value is itself the output of a completed search rather than a figure copied from the literature. The published sequence A003022 [23] is consulted only after a search terminates, purely to check the answer. The proof of optimality for order m is thus generated by the program from the ground up, resting only on lower bounds it has itself established for smaller orders and on the two sound structural rules of symmetry and sub-ruler monotonicity.

4 Results

We ran the incremental solver on orders 2 through 11. Every reported length is the outcome of a completed exhaustive search: for each order the tree was traversed to exhaustion, so the returned length is proven optimal in the sense that all shorter rulers were examined and eliminated. Table 1 collects the full accounting—optimal length, one optimal set of marks, the number of candidate configurations (nodes) examined, and the wall-clock runtime—and confirms the length against the published value in sequence A003022 [23].

Table 1: Proven optimal Golomb rulers for orders 2–11. "Nodes" is the number of admissible candidate mark placements examined by the branch-and-bound search; "L^*" is the optimal length proven by the program; "A003022" is the published optimal length. All ten lengths match, and every returned mark set is a valid Golomb ruler.

Order m	L^*	One optimal ruler (marks)	Nodes examined	Runtime (s)	Matches A003022
2	1	`0 1`	1	6.7×10^{-7}	✓
3	3	`0 1 3`	1	2.4×10^{-5}	✓
4	6	`0 1 4 6`	9	2.6×10^{-5}	✓
5	11	`0 1 4 9 11`	55	1.0×10^{-4}	✓
6	17	`0 1 4 10 12 17`	340	7.8×10^{-4}	✓
7	25	`0 1 4 10 18 23 25`	2,690	8.6×10^{-3}	✓
8	34	`0 1 4 9 15 22 32 34`	18,692	8.5×10^{-2}	✓
9	44	`0 1 5 12 25 27 35 41 44`	122,006	0.809	✓
10	55	`0 1 6 10 23 26 34 41 53 55`	805,995	6.634	✓
11	72	`0 1 4 13 28 33 47 54 64 70 72`	13,677,172	145.339	✓

The highest order for which optimality is proven in this study is **11**, with optimal length 72. Across all ten orders the program's answers agree exactly with the published optimal lengths of A003022: there are *no* orders where they differ. This agreement is a genuine independent check rather than a tautology, because the published values were never supplied to the search as targets—they were compared against only after each search terminated. Each returned mark set was additionally re-validated by an independent routine that recomputes all $\binom{m}{2}$ pairwise differences and confirms they are distinct; every ruler passed. Note that several optimal lengths admit more than one optimal ruler (and every ruler has a distinct mirror image removed by the symmetry break); Table 1 reports one canonical representative per order, namely the first optimal ruler encountered under the search's ordering.

4.1 Visualising the optimal rulers

Figure 1 renders the ten optimal rulers to scale on a common axis, making the near-quadratic growth of the optimal length with order visually apparent: the length roughly doubles from order 5 ($L = 11$) to order 8 ($L = 34$) and doubles again by order 11 ($L = 72$). The irregular, non-repeating spacing that is the hallmark of a Golomb ruler is also evident—no two gaps between marks, and more strongly no two distances between *any* pair of marks, are equal within a row.

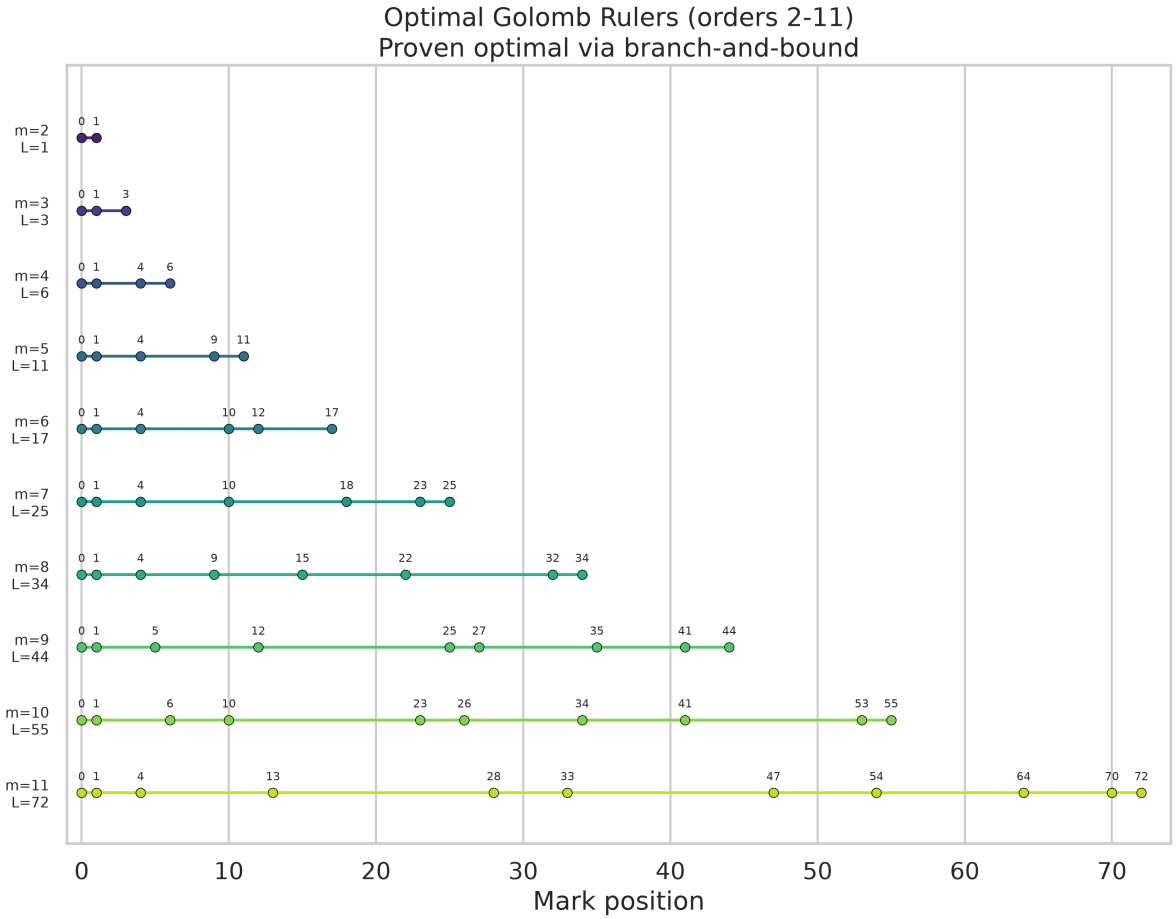


Figure 1: The proven optimal Golomb rulers for orders $m = 2$ through $m = 11$, drawn to scale on a shared position axis. Each row is one optimal ruler labelled by its order and optimal length L; filled circles mark the integer positions. The characteristic uneven spacing reflects the all-distinct-differences constraint, and the growth of the rightmost mark illustrates the near-quadratic scaling of $L^*(m)$.

4.2 Effect of the pruning components

To quantify how much each accelerator contributes to the search, we re-ran the solver on orders 2–6 with individual components disabled and counted the nodes examined in each configuration. Table 2 reports the result. Two observations stand out. First, lower-bound pruning is the dominant node-reducer even at these small orders: disabling it inflates the order-6 node count from 340 to 463, a 36% increase, while disabling both accelerators raises it to 481. Second, symmetry breaking as implemented contributes a smaller marginal reduction at low orders (its incremental effect at order 6 is the 481 $\rightarrow$ 463 difference once the lower bound is also removed); its role is partly to guarantee that mirror-image optima are never separately enumerated, a saving whose relative value grows with order as the number of complete rulers of a given length increases. Both rules are sound—neither ever discards an optimum—so the substantially smaller node counts they produce come at no cost to the correctness of the optimality proof.

Table 2: Ablation of the pruning components on orders 2–6, measured in nodes examined. "Full" enables both symmetry breaking and lower-bound pruning; the remaining columns disable symmetry only, the lower bound only, and both. Lower-bound pruning accounts for most of the reduction at these orders.

Order m	L^*	Full	No symmetry	No lower bound	Neither
2	1	1	1	1	1
3	3	1	1	1	1
4	6	9	9	9	10
5	11	55	55	62	70
6	17	340	340	463	481

5 Complexity Analysis

The defining feature of the optimal-ruler problem is that the cost of *proving* optimality grows explosively with order, even though the rulers themselves grow only quadratically in length. This section quantifies that growth from the instrumented node counts and runtimes of Table 1 and relates the two.

5.1 Growth of the search tree

Table 3 lists the successive growth ratios of the node count and runtime—that is, the factor by which each metric multiplies when the order increases by one. From order 4 onward the node count grows by a remarkably steady factor of roughly six to eight per order, until a pronounced jump of nearly seventeenfold on the step into order 11. Taking the geometric mean of the per-order node ratios from order 4 to order 11 gives an average multiplicative growth of about 7.6 per order, so the search-tree size behaves, to first approximation, like $C \cdot 7.6^{\,m}$—unmistakably exponential. Over the eight orders from 4 to 11 the node count rises from 9 to 1.37×10^7, a factor of roughly 1.5×10^6, and the cumulative number of candidate configurations examined across all ten orders is 1.46×10^7.

Table 3: Per-order growth ratios of the search cost. "Node ratio" is nodes(m)/nodes($m-1$); "Runtime ratio" is the analogous factor for wall-clock time; "Throughput" is nodes examined per second. The node count grows by a steady $\approx$ 6–8$\times$ per order, spiking to $\approx 17\times$ at order 11, while throughput slowly declines as the big-integer difference mask widens.

Order m	Nodes	Node ratio	Runtime ratio	Throughput (nodes/s)
4	9	—	—	3.5×10^5
5	55	6.1	3.9	5.5×10^5
6	340	6.2	7.8	4.3×10^5
7	2,690	7.9	11.0	3.1×10^5
8	18,692	6.9	9.9	2.2×10^5
9	122,006	6.5	9.5	1.5×10^5
10	805,995	6.6	8.2	1.2×10^5
11	13,677,172	17.0	21.9	9.4×10^4

The exponential character is intuitive. As the order increases, the optimal length $L^*(m)$ grows roughly as m^2, so the range of integer positions open to each interior mark widens; simultaneously the tree deepens by one level per added mark. The product of a growing branching range and a

growing depth produces multiplicative blow-up, which the pruning rules attenuate—dramatically, in absolute terms—but cannot eliminate, because no polynomial certificate of optimality is known. The steadiness of the ≈ 6–$8\times$ ratio across orders 5–10 indicates that the pruning removes a roughly *constant fraction* of the naive growth at each level, so the surviving tree still expands geometrically. The larger jump at order 11 reflects that the optimal length there increases by 17 over order 10 (from 55 to 72), a wider increment than the typical step, which enlarges the admissible position range at several levels before the incumbent tightens enough to prune it.

5.2 Runtime and throughput

Runtime tracks node count closely but grows slightly faster: the runtime ratio exceeds the node ratio at every order (for instance 21.9 versus 17.0 at order 11). The gap is explained by the last column of Table 3, the throughput in nodes per second, which declines from about 5.5×10^5 at order 5 to 9.4×10^4 at order 11. Two effects drive this decline. First, the difference mask is an arbitrary-precision integer roughly $L^*(m)$ bits wide, so the bit-test and mask-update operations at each node cost more machine words as the ruler lengthens. Second, deeper recursion means each node performs a longer inner loop over already-placed marks (the feasibility test is $O(k)$ at depth k). The combined effect is that per-node work grows slowly with order, so runtime is the product of an exponentially growing node count and a mildly growing per-node cost. The practical consequence is stark: extrapolating the observed ≈ 8–$20\times$ per-order runtime growth, proving order 12 with this single-threaded implementation would take on the order of tens of minutes, and each subsequent order roughly an order of magnitude longer—which is precisely why the historical proofs beyond the mid-teens required dedicated hardware, tailored constraint models, or massively distributed computation [8, 9, 17, 27].

Figure 2 plots both metrics on logarithmic axes. The near-linear appearance of the node-count and runtime curves on a log scale is the visual signature of exponential growth: the search-tree size spans roughly seven orders of magnitude from order 2 to order 11, and the runtime spans about eight. The slight upward curvature of the runtime plot relative to the node plot is the throughput decline made visible.

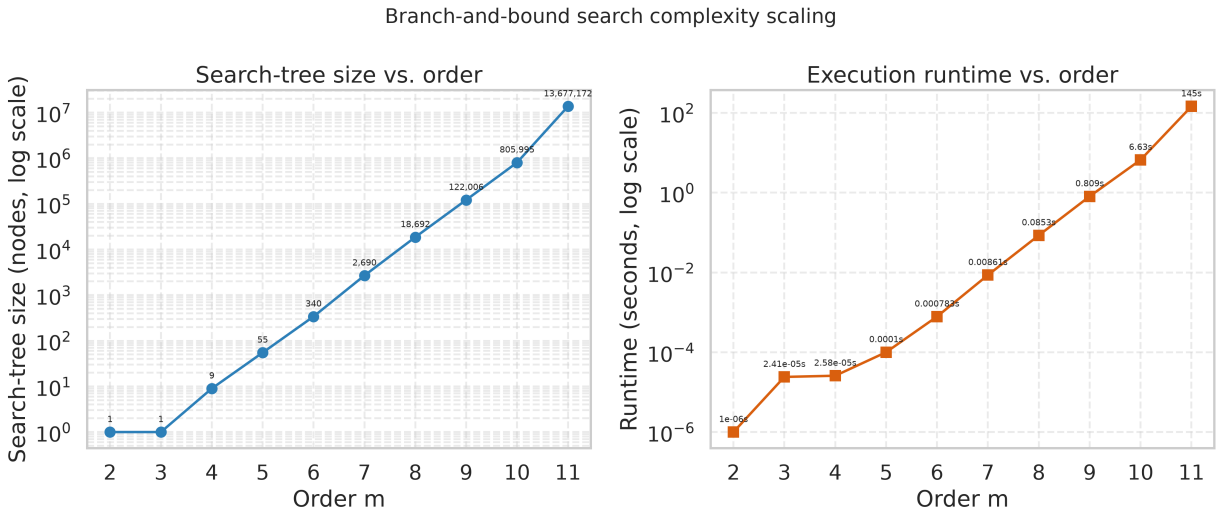


Figure 2: Branch-and-bound search-cost scaling on logarithmic axes. *Left:* number of candidate configurations (nodes) examined versus order; the count rises from a single node at orders 2–3 to 1.37×10^7 at order 11. *Right:* wall-clock runtime versus order, from sub-microsecond at the smallest orders to about 145 seconds at order 11. The near-straight log-scale trends confirm exponential growth in both metrics.

6 Discussion

The results establish, by exhaustive branch-and-bound with full search-tree accounting, the optimal Golomb rulers for every order from 2 to 11, and they agree without exception with the canonical optimal lengths recorded in A003022 [23]. It is worth being precise about what this agreement does and does not demonstrate. It does not merely reproduce a table: the search received no target length and no candidate ruler from the literature, and it terminated only after traversing the entire pruned tree, so the optimality of each length is certified by the program's own exhaustion of the smaller-length search space. The published sequence functioned solely as an external oracle against which the independently computed answers were checked after the fact. That all ten lengths coincide is therefore evidence that both the algorithm and its implementation are correct, and it locates the work within the well-established record for this problem.

Placed against that record, the present study is deliberately modest in reach and correspondingly transparent in method. Shearer proved orders up to sixteen by exhaustive search on the hardware of 1990 [27]; Dollas, Rankin, and McCracken reached order nineteen with a parallel implementation [9, 24]; and the `distributed.net` Project OGR has since certified optimality through order twenty-eight using volunteer computing at planetary scale [8]. More recent work reframes the problem for general-purpose solvers—constraint-programming and mixed-integer models with bound tightening, valid inequalities, and tailored branching that certify optimality up to the mid-teens on a single workstation [13, 17, 30]. Our contribution is not to extend this frontier but to furnish a compact, fully instrumented reference implementation whose optimality argument is legible in its entirety: every pruning rule is elementary and sound, and the node counter makes the size of the proof an explicit, reproducible number rather than an opaque assertion. Order 11, at 1.37×10^7 examined nodes and about 145 seconds, sits comfortably within reach of a single core, which is the natural ceiling for an unoptimised, single-threaded, exact search of this design.

The complexity analysis clarifies why that ceiling is where it is. The search tree grows

geometrically—by an average factor near 7.6 per order over the range studied—so the gap between what is tractable in seconds and what is tractable in hours is only two or three additional orders. This is the computational signature of a problem for which no efficient optimality certificate is known: although the general optimal-ruler decision problem has not been proven NP-hard in its exact canonical form, natural neighbouring problems are [21], and in practice every proof beyond small orders has demanded either specialised algorithmics or brute computational force. The connection to Sidon sets frames the asymptotics—the optimal length grows near-quadratically, in line with the Erdős–Turán and Lindström bounds and their recent refinements [3, 12, 20], and with the perfect-difference-set constructions of Singer that generate dense near-optimal rulers for infinitely many orders [29]—but these results describe the landscape rather than shortcut the exact proofs, which remain resolutely combinatorial.

Several limitations bound the interpretation of these findings. The implementation is single-threaded and written in a high-level language, so its absolute runtimes are not competitive with optimised or parallel solvers; they are reported for their *scaling behaviour* and their role in the search accounting, not as performance benchmarks. The use of an arbitrary-precision integer as the difference mask is elegant and overflow-free but imposes a per-node cost that grows with ruler length, which is the source of the mild throughput decline documented in Section 5; a fixed-width bitset or a byte array indexed by difference would be faster at these orders. The lower bound used for pruning is the exact optimal length of smaller orders, which is tight and cheap here but does not incorporate the stronger midpoint or partial-completion bounds that state-of-the-art solvers exploit to reach higher orders. Finally, the study proves optimality only up to order 11; extending it would require either substantially stronger bounds or parallelisation, both of which are well-trodden but beyond the scope of this compact reference implementation. None of these limitations affects the correctness of the reported optima—each is a statement about efficiency, not about the validity of the proofs, which stand on the completed exhaustive traversal.

The applications surveyed in Section 1 give the results their practical significance. The orders proven here—up to eleven marks—cover the range most relevant to physical designs such as minimum-redundancy antenna arrays [4, 22], interference-free frequency plans [1, 2], and self-orthogonal codes [25], where knowing the provably minimal span translates directly into minimal aperture, bandwidth, or delay. For a practitioner, the value of a certified optimum over a merely good heuristic ruler is the guarantee that no more compact arrangement exists, and it is exactly that guarantee that an exhaustive branch-and-bound, and only an exhaustive search, can provide.

7 Conclusion

We have constructed and proved optimal Golomb rulers for every order from 2 to 11 using a recursive depth-first branch-and-bound search accelerated by reflection symmetry breaking, bitmask difference tracking, and lower-bound pruning seeded by previously proven orders. For each order the program reports the optimal length, one optimal set of marks, the exact number of candidate configurations examined, and the runtime; optimality is certified by the completed exhaustive traversal, which explicitly rules out every shorter ruler. All ten optimal lengths—1, 3, 6, 11, 17, 25, 34, 44, 55, and 72—match the published values of OEIS A003022 exactly, with no order in disagreement, and every returned ruler was independently re-verified to have all-distinct pairwise differences. The rulers and their optimality certificates are produced by the program from lower bounds it establishes itself; the published sequence is used only as an after-the-fact check.

The search cost grows exponentially, by an average factor of roughly 7.6 per order over the range studied, with a sharper seventeenfold jump entering order 11, where proving optimality

required examining 1.37×10^7 candidate configurations in about 145 seconds. This scaling—seven orders of magnitude in node count and eight in runtime between orders 2 and 11—is the concrete manifestation of the problem's computational hardness and marks order 11 as the natural ceiling for a compact, single-threaded exact solver. The highest order for which optimality is proven in this work is therefore **11**.

Beyond its specific results, the study offers a transparent, fully instrumented reference for how optimality proofs of this kind are actually assembled: from a greedy incumbent, through sound structural pruning, to an exhaustive traversal whose size is reported as an explicit number. Natural extensions—stronger partial-completion lower bounds, fixed-width difference bitsets, and parallel or distributed search—would push the provable frontier higher, as the historical record from Shearer to Project OGR demonstrates, but the correctness of the certificates presented here rests entirely on the elementary, self-contained argument realised by the program.

References

[1] Michael D. Atkinson, Nicola Santoro, and Jorge Urrutia. Integer sets with distinct sums and differences and carrier frequency assignments for nonlinear repeaters. *IEEE Transactions on Communications*, 34(6):614–617, 1986. doi: 10.1109/TCOM.1986.1096587.

[2] Wallace C. Babcock. Intermodulation interference in radio systems: Frequency of occurrence and control by channel selection. *Bell System Technical Journal*, 32(1):63–73, 1953. doi: 10.1002/j.1538-7305.1953.tb01422.x.

[3] József Balogh, Zoltán Füredi, and Souktik Roy. An upper bound on the size of Sidon sets. *The American Mathematical Monthly*, 130(5):437–445, 2023. doi: 10.1080/00029890.2023.2176667.

[4] Gary S. Bloom and Solomon W. Golomb. Applications of numbered undirected graphs. *Proceedings of the IEEE*, 65(4):562–570, 1977. doi: 10.1109/PROC.1977.10517.

[5] John P. Costas. A study of a class of detection waveforms having nearly ideal range–doppler ambiguity properties. *Proceedings of the IEEE*, 72(8):996–1009, 1984. doi: 10.1109/PROC.1984.12967.

[6] James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 148–159. Morgan Kaufmann, 1996.

[7] Apostolos Dimitromanolakis. Analysis of the Golomb ruler and the Sidon set problems, and construction of large near-optimal Golomb rulers. Technical report, Department of Electronic and Computer Engineering, Technical University of Crete, 2002.

[8] distributed.net. Project OGR: The search for optimal Golomb rulers. `https://www.distributed.net/OGR`, 2022. Accessed 2026-07-12.

[9] Apostolos Dollas, William T. Rankin, and David McCracken. A new algorithm for Golomb ruler derivation and proof of the 19 mark ruler. *IEEE Transactions on Information Theory*, 44(1):379–382, 1998. doi: 10.1109/18.651068.

[10] Konstantinos Drakakis. A review of the available construction methods for Golomb rulers. *Advances in Mathematics of Communications*, 3(3):235–250, 2009. doi: 10.3934/amc.2009.3.235.

[11] John Engbers and David Galvin. Enumerating the Golomb rulers and acyclic orientations of graphs. *The Electronic Journal of Combinatorics*, 19(3):P42, 2012. doi: 10.37236/2160.

[12] Paul Erdős and Paul Turán. On a problem of Sidon in additive number theory, and on some related problems. *Journal of the London Mathematical Society*, s1-16(4):212–215, 1941. doi: 10.1112/jlms/s1-16.4.212.

[13] Philippe Galinier, Brigitte Jaumard, Rodrigo Morales, and Gilles Pesant. A constraint-based approach to the Golomb ruler problem. In *Proceedings of the 3rd International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, 2001.

[14] Ian P. Gent, Karen E. Petrie, and Jean-François Puget. Symmetry in constraint programming. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, chapter 10, pages 329–376. Elsevier, 2006.

[15] Solomon W. Golomb. *Shift Register Sequences*. Holden-Day, San Francisco, CA, 1967.

[16] Donald E. Knuth. Dancing links. In Jim Davies, Bill Roscoe, and Jim Woodcock, editors, *Millennial Perspectives in Computer Science*, pages 187–214. Palgrave Macmillan, 2000. arXiv:cs/0011047.

[17] Burak Kocuk and Willem-Jan van Hoeve. A computational comparison of optimization methods for the Golomb ruler problem. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2019)*, volume 11494 of *Lecture Notes in Computer Science*, pages 409–425. Springer, 2019. doi: 10.1007/978-3-030-19212-9_27.

[18] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. doi: 10.2307/1910129.

[19] Eugene L. Lawler and David E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966. doi: 10.1287/opre.14.4.699.

[20] Bernt Lindström. An inequality for B_2-sequences. *Journal of Combinatorial Theory*, 6(2):211–212, 1969. doi: 10.1016/S0021-9800(69)80124-9.

[21] Christoph Meyer and Periklis A. Papakonstantinou. On the complexity of constructing Golomb rulers. *Discrete Applied Mathematics*, 157(4):738–748, 2009. doi: 10.1016/j.dam.2008.06.008.

[22] Alan T. Moffet. Minimum-redundancy linear arrays. *IEEE Transactions on Antennas and Propagation*, 16(2):172–175, 1968. doi: 10.1109/TAP.1968.1139138.

[23] OEIS Foundation Inc. The on-line encyclopedia of integer sequences, sequence A003022: Optimal (shortest) Golomb rulers. `https://oeis.org/A003022`, 2026. Accessed 2026-07-12.

[24] William T. Rankin. Optimal golomb rulers: An exhaustive parallel search implementation. Master's thesis, Duke University, Durham, NC, 1993.

[25] John P. Robinson and Arthur J. Bernstein. A class of binary recurrent codes with limited error propagation. *IEEE Transactions on Information Theory*, 13(1):106–113, 1967. doi: 10.1109/TIT.1967.1053951.

[26] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*. Foundations of Artificial Intelligence. Elsevier, Amsterdam, 2006.

[27] James B. Shearer. Some new optimum Golomb rulers. *IEEE Transactions on Information Theory*, 36(1):183–184, 1990. doi: 10.1109/18.50388.

[28] Simon Sidon. Ein satz über trigonometrische polynome und seine anwendung in der theorie der Fourier-Reihen. *Mathematische Annalen*, 106(1):536–539, 1932. doi: 10.1007/BF01455900.

[29] James Singer. A theorem in finite projective geometry and some applications to number theory. *Transactions of the American Mathematical Society*, 43(3):377–385, 1938. doi: 10.1090/S0002-9947-1938-1501951-4.

[30] Barbara M. Smith, Kostas Stergiou, and Toby Walsh. Modelling the Golomb ruler problem. Technical Report Research Report 1999.12, School of Computer Studies, University of Leeds, 1999.

A Reproducibility and Search Instrumentation

This appendix records the details needed to reproduce the results and the exact form of the search accounting on which the optimality claims rest.

Implementation. The solver is a single Python module implementing the recursive branch-and-bound of Algorithm 1. The public entry point solves the orders incrementally from 2 upward, passing each proven optimal length into the lower-bound function used for the next order. The difference set is represented as one Python integer treated as a bitmask, so that testing and updating the used differences are pure bit operations and no auxiliary set object is allocated per node. The greedy construction that seeds the incumbent appends, at each step, the smallest integer preserving the Golomb property. All runs were single-threaded.

Search accounting. The node counter is incremented exactly once for each candidate mark position that passes the bitmask feasibility test, at every depth of the recursion. It therefore counts admissible partial and complete placements examined—the configurations the search actually inspected—and it is this quantity that is reported as "Nodes examined" in Table 1. Because the recursion terminates only when the position range at every open node has been fully traversed, a terminated search has provably examined or pruned every ruler of length below the returned optimum; the node count is thus a faithful, reproducible measure of the size of the optimality proof. Runtime is measured with a monotonic performance counter around the search proper, excluding input/output and the greedy seed.

Verification. Two independent checks guard the results. Each returned mark set is passed to a validator that recomputes all $\binom{m}{2}$ pairwise differences and confirms they are pairwise distinct and that the marks are strictly increasing from zero; every ruler in Table 1 passed. Each optimal length is then compared against sequence A003022 [23]; all ten agree. The verification is deliberately

separated from the search so that the optimality argument depends only on the completed traversal and the two sound structural rules, never on the published table.

Correctness of the accelerations. The symmetry break is loss-free because exactly one of a Golomb ruler and its reflection satisfies the first-gap-less-than-last-gap inequality (2), and a Golomb ruler of order $m \geq 3$ can never be self-symmetric (that would repeat a gap). The lower-bound rule is loss-free because any suffix of a Golomb ruler is itself a Golomb ruler, so a completion using r further marks has length at least the true optimal length $L^*(r)$ of order r; substituting the triangular bound $\binom{r}{2}$ when $L^*(r)$ is unavailable only weakens, never invalidates, the bound. Consequently every subtree pruned by Equation (3) or by the symmetry ranges is guaranteed to contain no ruler shorter than the incumbent, and the search's exhaustion of the remaining tree is a valid proof of optimality.

Reproducibility. The reported node counts are deterministic functions of the order and the pruning configuration and will reproduce exactly on any platform; the runtimes are hardware-dependent and were obtained on a single commodity CPU core, so their absolute values will vary while their relative scaling—the growth ratios of Table 3—should be stable. The complete numerical results, including the per-order records and the ablation counts of Table 2, are provided alongside this report in machine-readable form.