Title

There is no Leech tree on 18 vertices, and no Leech spider of order at least five, with a leaf-deletion bound at order 25

Author

Meng, L

Publication Date

2026-09-06

DOI

10.5281/zenodo.22573641

There is no Leech tree on 18 vertices

and no Leech spider of order at least five,
with a leaf-deletion bound at order 25

Lingsen Meng
Department of Earth, Planetary, and Space Sciences,
University of California, Los Angeles, CA 90095, USA
lsmeng@g.ucla.edu

Draft of 6 September 2026

Abstract

A *Leech tree* of order n is a tree on n vertices with positive integer edge weights whose $\binom{n}{2}$ pairwise path-weights are exactly $1, 2, \ldots, \binom{n}{2}$. Leech (1975) found five, of orders $2, 3, 4, 4, 6$; Taylor (1977) showed that the order must be a square or a square plus two; earlier searches excluded the admissible orders 9, 11 and 16, leaving 18 as the smallest open order.

We prove that there is no Leech tree on 18 vertices, by an exhaustive generation of forced forests over all 123,867 trees of that order. The search visits 59,779,854,336 nodes and returns no survivor; its per-level counts were reproduced by independent runs and by a clean-room implementation written from the algorithm description alone. The same conclusion was reached independently and concurrently by Ghodsi, through a different reduction and a different trusted base. We also prove that no spider of order $n \geq 5$ is a Leech tree, by a finite exact analysis of the largest distances together with independently implemented checks.

For the next admissible order $n = 25$ we prove a structural bound rather than a non-existence result: deleting a leaf ℓ forces a block of consecutive rooted depths whose pairs are too crowded for the block to be long, so that $\operatorname{diam}(T - \ell) \geq 281$ for every leaf, unconditionally. Four finite certified searches raise this to 285, reducing order 25 to fourteen values of one explicit finite normal form; we make no claim that order 25 is settled.

Variants of the same engines settle neighbouring questions: $M(11) = 60$ and $M(12) = 77$ for the minimal distinct-distance trees of Calhoun et al., each with exactly two minimal trees; there is no modular Leech tree of order 9 or 11, while order 5 does admit them, contrary to a statement in the literature; and there are exactly six leaf-Leech trees with six leaves.

Keywords. Leech tree, Leech spider, perfect distance tree, distinct distance tree, minimal distinct distance tree, modular Leech tree, leaf-Leech tree, Golomb ruler, Sidon set, exhaustive search, computer-assisted proof.
MSC 2020. 05C05, 05C78, 05C85, 11B83.

1 Introduction

Let T be a tree on n vertices with a weight $w(e) \in \mathbb{Z}_{>0}$ on every edge, and let $d(x, y)$ denote the weight of the (unique) x–y path. Following Leech [11] we call (T, w) a *Leech tree* (also *perfect distance tree*, *perfect distance-distinct tree*) if the multiset $\{d(x, y) : \{x, y\} \subseteq V(T)\}$ equals $\{1, 2, \ldots, N\}$ with $N = \binom{n}{2}$. Leech exhibited the five Leech trees of Table 1 and asked whether there are any others. None has been found since.

Table 1: The five known Leech trees [11]. Weights are listed as edge lists u–v:w on vertices $0, 1, \ldots$; each labelling is unique up to isomorphism.

n	N	tree	weighted edges	remark
2	1	K_2	0–1:1	
3	3	P_3	0–1:1, 1–2:2	
4	6	$K_{1,3}$ (star)	0–1:1, 0–2:2, 0–3:4	
4	6	P_4 (path)	0–1:1, 1–2:3, 2–3:2	weight 3 must be the middle edge
6	15	$B_{2,2}$ (double star)	0–1:1, 0–2:2, 0–3:5, 3–4:4, 3–5:8	centres $0, 3$ joined by weight 5

Necessary conditions on the order were given by Taylor [20]: counting odd distances shows that a Leech tree of order n exists only if $n = k^2$ or $n = k^2 + 2$ for some integer k (Lemma 2.2 below). The admissible orders are therefore $2, 3, 4, 6, 9, 11, 16, 18, 25, 27, \ldots$. Székely, Wang and Zhang [18] excluded $n = 9$ (also excluded by Shen Lin's search reported by Taylor [21]) and $n = 11$ by computer search, and proved asymptotic upper bounds on the length of a path and on the maximum degree of a Leech tree. Calhoun, Ferland, Lister and Polhill [3] introduced *distinct distance trees* (all path weights distinct, not necessarily forming an interval), gave a backtracking algorithm over weighted forests with a *forced next weight* (their Lemma 2.6 and Algorithm 2.7), and determined all perfect distance trees of order $n < 18$; in particular they excluded $n = 16$. Their summary states the bound as $n \leq 24$, but their abstract, their Table 1 (where the entry for $n = 18$ is only the trivial lower bound), and every later paper give $n < 18$; we treat "24" as a misprint. Further structural results—the non-existence of Leech trees of diameter 3 for $n \geq 7$, of certain "brooms", the "leaf-Leech" transformation, and Luo and Yu's finiteness result for diameter 4 and improved degree bound—are in [12, 15, 22]; a Leech labelling of general graphs was introduced in [8]. Varghese, Lakshmanan and Arumugam [23] define a tristar $T_{m,r,l}$ from a three-vertex path u–v–w by attaching m, r, l leaves at its three vertices, and prove that its diameter-4 cases ($m, l > 0$) are non-Leech. The subfamily $T_{1,r,1}$ is a spider with two legs of length two and r legs of length one; thus it is a previously known special case of our second theorem below. The order $n = 18$ is listed as the smallest open case, for instance in [15, 22] and in the FrontierMath open-problem list [6].

Our first main result settles the case $n = 18$.

Theorem 1.1. *There is no Leech tree on* 18 *vertices.*

Our second main result excludes an infinite topology class. A *spider* is a tree with at most one vertex of degree at least three; paths are included.

Theorem 1.2. *No spider of order $n \geq 5$ is a Leech tree.*

Theorem 1.2 is also computer-assisted, but its finite computation has only thousands of states. Its main point is a uniform "top window": once the two largest leg tips are fixed, every attempted Leech labelling fails while realising a bounded number of distances immediately below $N = \binom{n}{2}$. A star is the spider all of whose legs have length one, so Theorem 1.2 contains [12, Thm. 1.1], which states that there is no Leech star of order $n \geq 5$ and which was proved there by an elementary argument; the same paper rules out diameter 3 for $n \geq 7$ and shows that there are at most finitely many diameter-4 Leech trees. Spiders are unbounded in diameter, and the difficulty the top window addresses is exactly that a long leg cannot be handled one distance at a time.

The proof of Theorem 1.1 is a computer search. Its mathematical content is small—the forcing lemma of [3, 18] and an elementary description of the automorphism group of a forest with distinct edge weights (Lemma 3.2), which yields a search tree in which every relevant weighted forest appears exactly once up to isomorphism—and its cost is modest (about 6×10^{10}

search nodes, a few CPU-hours). We nevertheless think that a careful account is warranted, for three reasons. First, because $n = 18$ has been repeatedly cited as open, the result should be stated with a precise description of what was computed and how it was checked. Second, the per-level node counts of the search are a well-defined combinatorial quantity (the number of non-isomorphic "forced forests" with k edges that fit inside a Leech tree of order 18), so that the computation is falsifiable in a much stronger sense than a bare "no solution found": any re-implementation must reproduce eighteen integers, not one bit. Third, we want to be explicit about the status of the result as a computer-assisted theorem, including the parts that are and are not covered by independent verification (Section 9).

Our third result concerns the order that Theorem 1.1 makes the smallest open one, $n = 25$. It is not a non-existence result; it is a bound on what a Leech tree of that order would have to look like after one leaf is deleted. Write diam for the weighted diameter.

Theorem 1.3. *Let T be a Leech tree of order 25, so $N = \binom{25}{2} = 300$.*

(a) $\operatorname{diam}(T - \ell) \geq 281$ *for every leaf ℓ of T.*

(b) The unique pair $\{a, b\}$ of vertices at distance 300 consists of two leaves, and they can be named so that $\operatorname{diam}(T - b) = 299$ and $281 \leq \operatorname{diam}(T - a) \leq 298$. Every other leaf ℓ has $\operatorname{diam}(T - \ell) = 300$.

Part (a) is a counting theorem (Section 8.3): the deleted leaf forces the rooted depths of $T - \ell$ to contain a block of $m = 300 - \operatorname{diam}(T - \ell)$ consecutive values, and the $\binom{m}{2}$ distances inside that block fall into few classes on each of which they are confined to an interval of length $2m - 3$, so that m cannot be large. Part (b) is elementary (Theorem 8.5) and says that only one leaf deletion carries information. Four finite searches, described with their certificates in Section 8.4, exclude the four further values $\operatorname{diam}(T - \ell) = 281, 282, 283, 284$; with them the bound in (a) and (b) becomes 285, and the order-25 problem is reduced to fourteen values of one explicit finite normal form (Corollary 8.11). We prove nothing about the existence of a Leech tree of order 25.

The rest of the note is organised as follows. Section 2 collects the lemmas that are used, marking which are classical and which are (minor) additions. Section 3 describes the algorithm and proves that it is complete and free of isomorphic duplicates. Section 4 describes the verification strategy. Section 5 states the results for $n \leq 18$, including the per-level tables. Section 6 proves the uniform spider theorem. Section 7 reports what the same two engines give on three neighbouring problems—minimal distinct distance trees, modular Leech trees, and leaf-Leech trees—where the literature leaves gaps that a search of this kind can close. Section 8 proves Theorem 1.3, states the diametral-endpoint reduction, and reports the finite certificates for order 25 and the state of the first case beyond them. Section 9 discusses the status of the proof and what remains open.

2 Preliminaries

Throughout, n is the order, $N = \binom{n}{2}$, and "distance" means weighted path length. Weighted forests are considered up to isomorphism (a bijection of vertices preserving edges and weights). We say that a weighted forest is *distance-distinct* if the distances between pairs of vertices in the same component are pairwise distinct. All lemmas below are elementary; we include proofs of the two that carry the main argument (Lemmas 2.1 and 3.2) and sketch the rest.

Lemma 2.1 (forcing lemma; [18, "Find-Next-Weight"], [3, Lemma 2.6])**.** *Let (T, w) be a Leech tree with edge weights $w_1 < w_2 < \cdots < w_{n-1}$, and for $1 \leq k \leq n - 1$ let F_{k-1} be the forest formed by the edges of weights $w_1, \ldots, w_{k-1}$. Then w_k equals the least positive integer that is not a distance within a component of F_{k-1}.*

Proof. Let m be that least missing integer. If $w_k < m$ then w_k is already a distance in F_{k-1}, contradicting distinctness. If $w_k > m$ then m is not an edge weight, and any path realising m has at least two edges, hence every edge on it has weight $< m < w_k$ and lies in F_{k-1}; so m would be a distance in F_{k-1}, contradiction. Hence $w_k = m$. □

Consequently $w_1 = 1$, $w_2 = 2$, and $w_3 \in \{3, 4\}$ according as the edges of weight 1 and 2 are non-adjacent or adjacent; a Leech labelling of a given tree is determined by the *order* in which its edges receive their weights. Note that Lemma 2.1 fails for graphs with cycles [3].

Lemma 2.2 (Taylor's parity condition [20]; cf. [18, Thm. 1], [3, Thm. 1.3])**.** *If a Leech tree of order n exists, then $n = k^2$ or $n = k^2 + 2$ for an integer k; the vertex classes A, B at even and odd weighted distance from a fixed vertex satisfy $|A|\,|B| = \lceil N/2 \rceil$. For $n = 18$: $N = 153$, $\{|A|, |B|\} = \{11, 7\}$.*

Sketch. $d(x, y) \equiv d(x, z) + d(y, z) \pmod 2$ in a tree; a distance is odd iff its ends lie in different classes, so $|A|\,|B| = \lceil N/2 \rceil$, and $(|A| - |B|)^2 = n^2 - 4|A|\,|B| \in \{n, n - 2\}$. □

Lemma 2.3 (cut identity)**.** *For any weighted tree, $\sum_{x<y} d(x, y) = \sum_e c_e\, w(e)$, where $c_e = s_e(n - s_e)$ is the number of vertex pairs separated by e and s_e is the number of vertices on one side of e. For a Leech tree the left side is $N(N + 1)/2$.*

Proof. The path between x and y passes through e if and only if x and y lie on opposite sides of e, and there are $s_e(n - s_e)$ such pairs. Writing each $d(x, y)$ as the sum of the weights on its path and exchanging the order of summation gives $\sum_{x<y} d(x, y) = \sum_e w(e)\,|\{\, \{x, y\} : e \in \mathrm{path}(x, y)\,\}| = \sum_e c_e\, w(e)$. If T is Leech the multiset of distances is $\{1, \ldots, N\}$, whose sum is $N(N + 1)/2$. □

Lemma 2.4 (Golomb and containment bounds)**.** *In a distance-distinct weighted tree:*

(a) the vertices of any path with h edges, placed on a line at their cumulative distances, form a Golomb ruler with $h + 1$ marks, so $d(x, y) \geq G(h + 1)$ where $G(m)$ is the length of an optimal Golomb ruler with m marks;

(b) if the tree is Leech, every path weight is at most N, so the (hop) diameter D satisfies $G(D + 1) \leq N$;

(c) (containment) let s_x be the number of vertices u whose path to y passes through x, and s_y the number of v whose path to x passes through y, so that x counts in s_x and y in s_y, and exactly $s_x s_y$ pairs have paths containing the whole x–y path. Then in a Leech tree $d(x, y) \leq N + 1 - s_x s_y$; in particular $w(e) \leq N + 1 - c_e$ for every edge.

Proof. (a) Place the $h + 1$ vertices of the path on a line at their distances from one end. The difference between two marks is the tree distance between the corresponding vertices, and those are pairwise distinct, so the marks form a Golomb ruler with $h + 1$ marks whose length is $d(x, y)$. An optimal ruler is one of least length among those with a given number of marks, so $d(x, y) \geq G(h + 1)$.

(b) Apply (a) to a path realising the hop-diameter D. Its length is a distance of T, hence at most N, so $G(D + 1) \leq N$.

(c) For u counted in s_x and v counted in s_y the u–v path runs through x and then through y, so $d(u, v) = d(u, x) + d(x, y) + d(y, v) \geq d(x, y)$, with equality exactly for the single pair $(u, v) = (x, y)$. The remaining $s_x s_y - 1$ pairs therefore have distances strictly greater than $d(x, y)$; they are pairwise distinct and at most N, so $N - d(x, y) \geq s_x s_y - 1$. For an edge $e = xy$ the two counts are the two side sizes, whose product is c_e. □

The values

$$G(1), \ldots, G(17) = 0, 1, 3, 6, 11, 17, 25, 34, 44, 55, 72, 85, 106, 127, 151, 177, 199$$

(OEIS A003022 [14]) are used; $G(m)$ for $m \le 13$ was re-verified by exhaustive search for this project, and $G(14), \ldots, G(17)$ are taken from the literature (Shearer's tables [17]; the proof of the 19-mark ruler [5]; the distributed.net OGR project [4]). Since $G(16) = 177 > 153 = N$, a Leech tree of order 18 has hop-diameter at most 14. This strengthens the exact finite form of [18, Thm. 2] (which gives diameter ≤ 15 at $n = 18$). We note that [18] carries a published erratum [19], which we have not been able to obtain; every statement of [18] that we rely on is either re-proved here from scratch (Lemma 2.1) or independently recomputed (the exclusions at $n = 9$ and $n = 11$, with DRAT-checked certificates in Section 4.3), so the erratum cannot affect our results; but a reader weighing the comparison in the previous sentence, or the one at Lemma 2.5, should consult it.

Lemma 2.5 (star-Sidon degree bound)**.** *Let v be a vertex of degree d in a Leech tree of order n with incident weights $a_1 < \cdots < a_d$. Then the a_i and the $\binom{d}{2}$ sums $a_i + a_j$ $(i < j)$ are $d + \binom{d}{2}$ pairwise distinct integers in $[1, N]$; call a set of positive integers with this property (pairwise sums of* distinct *elements distinct and different from the elements)* star-Sidon*. Let $S(d)$ be the minimum of $a_{d-1} + a_d$ over star-Sidon sets of size d; then $S(d) \le a_{d-1} + a_d \le N + 1 - b_{(1)}b_{(2)}$, where $b_{(1)} \le b_{(2)}$ are the two smallest branch sizes at v. Since $a_{d-1} + a_d$ is the largest sum, $S(d)$ is the least T admitting a star-Sidon set of size d with all elements and sums at most T, so each entry of the table below is certified by a witness at $T = S(d)$ and by an exhaustive refutation at $T = S(d) - 1$. Exhaustive computation (`src/star_sidon_table.c`, reproduced for $d \le 9$ by the independent Python implementation `src/star_sidon_table.py`; output in `results/star_sidon_table.csv`) gives*

d	2	3	4	5	6	7	8	9	10	11	12
$S(d)$	3	6	11	19	31	43	63	80	110	138	169

and S is non-decreasing in d. Since $S(12) = 169 > 153$, the maximum degree of a Leech tree of order 18 *is at most* 11 *(a degree-*11 *vertex is possible only with $b_{(1)}b_{(2)} \le 16$).*

Remark (correction). An earlier version of this lemma asserted that the incident weights form a Golomb ruler (equivalently a Sidon set in the strong sense) and used $G(d) + G(d-1) + 2$ in place of $S(d)$. That is false: the Leech condition only makes $\{a_i\}$ a *weak* Sidon set (sums of two distinct elements distinct); three-term arithmetic progressions among the a_i are not excluded. The star $K_{1,7}$ with weights $\{1, 2, 4, 8, 14, 19, 24\}$ has all 28 distances distinct, yet $a_6 + a_7 = 43 < 44 = G(7) + G(6) + 2$; the correct minima are those in the table (the two sequences differ from $d = 6$ on). The counterexample was found during the internal audit of the search code (Section 4). The maximum-degree consequence at $n = 18$ ($\Delta \le 11$) is unaffected: it also follows from an exhaustive search over star-Sidon sets with all elements and sums ≤ 153, which finds a set of size 11 (e.g. $\{1, 2, 4, 7, 12, 23, 32, 41, 56, 70, 83\}$) and none of size 12. Lemma 2.5 is a finite, explicit version of the idea behind [18, Thm. 3], and we claim no priority for that idea. The sharpest asymptotic form is [12, Thm. 1.4]: for every fixed k, the k largest degrees of a Leech tree of order n satisfy $d_1^2 + \cdots + d_k^2 \le (\frac{1}{4} + o(1))n^2$, whence $\Delta \le (\frac{1}{2} + o(1))n$, improving the bound $\Delta \le (\frac{1}{\sqrt{2}} + o(1))n$ of [18] and its erratum. The two statements do different work. Theirs is asymptotic and says nothing at a single order; ours is explicit and finite, and it is the finite form that the proof of Theorem 1.1 uses, since at $n = 18$ it is the concrete inequality $S(12) = 169 > 153$ that removes degree 12.

Lemma 2.6 (weight bounds; [3, Thms. 2.8, 2.9])**.** *In a Leech tree of order n the largest edge weight satisfies $m_1 \le \lfloor (4n-1)^2/48 \rfloor$ and the second largest $m_2 \le \lfloor (N-1)/2 \rfloor$; more generally the weights of any set of edges lying on a common path sum to at most N. At $n = 18$: $m_1 \le 105$, $m_2 \le 76$.*

What is used where. It is worth being precise about which of these statements the proof of Theorem 1.1 depends on. The forest search of Section 3 uses only Lemma 2.1, the trivial facts that all distances are distinct and at most N and that a subforest of a Leech tree has at most n vertices, the "common path" case of Lemma 2.6 (any two edges lie on a common path, so $w(e) + w(f) \le N$; this rule was found never to prune anything at $n \le 18$), and Lemma 3.2 below. Lemmas 2.2–2.5 and the rest of Lemma 2.6 are *not* needed for Theorem 1.1; they are used in the independent per-topology engine (Section 4.2) as pruning rules and as topology filters, and are recorded here because they were verified in the course of the project and because Lemma 2.5 corrects a statement that circulated in our own notes. Lemmas 2.1, 2.2, 2.6 and Lemma 2.4(a),(b) are from the literature (2.4(b) as applied to the diameter is folklore; [3, Prop. 1.2] uses it for paths); Lemma 2.3, Lemma 2.4(c), Lemma 2.5 (in the corrected form) and Lemma 3.2 do not seem to appear in the Leech tree literature, though none of them is deep.

3 The algorithm

The search described in this section is a re-implementation of Algorithm 2.7 of Calhoun, Ferland, Lister and Polhill [3], which is itself built on the forcing lemma (their Lemma 2.6; the "Find-Next-Weight" rule of [18]). The algorithm is theirs: a depth-first search whose nodes are weighted forests, in which the next edge receives a forced weight and is inserted by joining two components, attaching a new vertex, or starting a disjoint edge. What is offered here is an implementation efficient enough to carry that search to $n = 18$ — roughly 7.6 times the work of $n = 17$, which is as far as [3] went — together with an explicit proof of the isomorph-rejection rule (Lemma 3.2, asserted without proof in [3]), a sharding scheme, and the verification chain of Section 4. We make no claim of algorithmic novelty.

3.1 Forced forests

Fix n and $N = \binom{n}{2}$. By Lemma 2.1, if (T, w) is a Leech tree and $t \ge 1$, then the subforest $F_t = \{e \in T : w(e) < t\}$ satisfies: (i) F_t is distance-distinct with all distances in $[1, N]$; (ii) F_t has at most n vertices (we regard F_t as a forest without isolated vertices; the vertex set is the set of endpoints of its edges); (iii) if t' is the least positive integer that is not a distance of F_t, then either $F_t = T$ (in which case $t' = N + 1$), or T has an edge e of weight exactly t' and $F_{t'+1} = F_t + e$. Call a weighted forest *forced* (for the target $[1, N]$ and order n) if it arises from the empty forest by repeatedly adding an edge whose weight is the least missing distance, in one of three ways: *join* (the new edge connects two existing components), *attach* (one endpoint is a new vertex), or *new* (both endpoints are new; a fresh single-edge component), never exceeding n vertices and never repeating a distance or exceeding N. This is exactly the search of [3, Alg. 2.7]. By (iii) every Leech tree of order n is a forced forest with $n - 1$ edges and n vertices; conversely a forced forest with n vertices and $n - 1$ edges is a tree in which every value $1, \dots, N$ is realised (the least missing value has been pushed to $N + 1$), hence a Leech tree.

The natural DFS enumerates forced forests level by level (level $k = k$ edges). Its cost is dominated by the last few levels, and without symmetry reduction the same abstract forest is generated many times, since the endpoints of single-edge components and the components themselves can be listed in many orders. The one idea that makes the search cheap is complete isomorph rejection, for which the following two lemmas suffice.

3.2 Isomorph rejection

Lemma 3.1 (canonical parent)**.** *In a forced forest the edge weights are pairwise distinct and the weights of successive edges are strictly increasing; hence removing the edge of maximum weight from a forced forest with $k+1$ edges yields a forced forest with k edges, and this parent is well defined up to isomorphism.*

Proof. Each new weight t is the least missing distance and becomes a distance once the edge is added, so later weights are larger. Deleting the last edge returns to the previous state of the DFS. □

Lemma 3.2 (automorphisms of a distinct-weight forest)**.** *Let P be a forest without isolated vertices whose edge weights are pairwise distinct. Then the group of weight-preserving automorphisms of P is generated by the transpositions of the two endpoints of the single-edge components of P; in particular $\mathrm{Aut}(P) \cong \mathbb{Z}_2^s$, where s is the number of single-edge components, and it acts on components independently.*

Proof. Let φ be a weight-preserving automorphism. Since the weights are distinct, φ fixes every edge setwise. If two distinct edges e, f share a vertex v, then $\varphi(v) \in e \cap f = \{v\}$, so v is fixed. In a component with at least two edges every edge has an endpoint of degree ≥ 2; that endpoint is fixed, and then the other endpoint of the (setwise fixed) edge is fixed too. So every component with ≥ 2 edges is fixed pointwise. A single-edge component admits exactly the identity and the endpoint swap, and swaps of different components commute and are independent. □

Proposition 3.3 (exactly-once generation)**.** *Consider the following restricted DFS. Vertices are labelled in order of appearance. At a node P (a labelled forced forest), the* eligible *vertices are all vertices of P except the larger-labelled endpoint of each single-edge component. Children are: all joins between eligible vertices $x < y$ in different components; all attachments of a new vertex to an eligible vertex x; and, if P has at most $n-2$ vertices, the single "new" child. (Each child is kept only if it is a forced forest, i.e. passes the distance and vertex tests.) Then every abstract forced forest with k edges is generated exactly once at level k, for every k.*

Proof. Induction on k. Assume the level-k nodes are pairwise non-isomorphic and cover every abstract forced forest with k edges. Let G be an abstract forced forest with $k+1$ edges, e its maximum-weight edge and $P = G - e$; by Lemma 3.1, P is a forced forest with k edges, so it appears exactly once as a labelled node P'. Two extensions $P' + (x, y, t)$ and $P' + (x', y', t)$ (with the same forced weight t) are isomorphic iff some $\varphi \in \mathrm{Aut}(P')$ maps $\{x, y\}$ to $\{x', y'\}$: an isomorphism of the children fixes the unique maximum-weight edge and restricts to an automorphism of the parents. By Lemma 3.2, $\mathrm{Aut}(P')$ is the product of the endpoint swaps of the single-edge components, so the orbit of an endpoint set is obtained by independently swapping the endpoint(s) that lie in single-edge components. Choosing the smaller label in each single-edge component therefore selects exactly one representative per orbit for joins (unordered pairs in distinct components), for attachments (orbit of x), and trivially for the "new" child. Children of non-isomorphic parents are non-isomorphic (their canonical parents would be isomorphic). □

Proposition 3.3 is what allows the search tree to be *counted*: its level-k node count equals the number of non-isomorphic forced forests with k edges for the given (n, N). This makes the computation checkable in the strong sense discussed in the Introduction. Because Aut is recomputed from the current component structure at each node, no automorphism state is cached; the appearance-order labelling is used only to pick "the smaller endpoint of a two-vertex component".

3.3 Implementation and sharding

The engine (`forest_search.cpp`, C++17, no dependencies) keeps, for each vertex x, the bitset $D_0[x]$ of distances from x within its component, and the bitset R of realised distances. Adding an edge (x, y, t) generates the new distances as shifted unions $D_0[x] + t + D_0[y]$; a child is rejected as soon as a shifted set meets R or exceeds N, and a per-vertex prefilter discards x when already the distances from x to the *new neighbour* alone collide (these are a subset of the new distances of any join or attachment at x). All state is undone on return (incremental, no copies). The only other prune is $t + w_{\max} \le N$ (Lemma 2.6, common path), which never fires at $n \le 18$.[1] There are no bounds, counting arguments or lookahead in this engine; its efficiency comes from Proposition 3.3 (which alone reduced the node count roughly tenfold at $n = 12$–13 relative to the same DFS without isomorph rejection) and from bit-parallel distance sets.

For parallelism the DFS is sharded at a fixed level L: the level-L nodes are enumerated in DFS order (deterministic), and shard i of K explores the subtrees of the level-L nodes with index $\equiv i \pmod K$; the levels $\le L$ are visited by every shard. Hence, if $h_i(d)$ is the number of level-d nodes visited by shard i, one must have $h_i(d) = h(d)$ for $d \le L$ and every i, and $\sum_i h_i(d) = h(d)$ for $d > L$; the total number of visited nodes is $\sum_i \text{nodes}_i = \text{unique} + (K - 1) \cdot \text{prefix}(\le L)$. These identities were checked in 36 (n, target, K, L) configurations, including K not dividing the number of level-L nodes and K larger than it. The verdict script for a sharded run asserts: exactly K records, indices $0, \ldots, K-1$ each once, identical (K, L), all `DONE`, and the histogram identities above.

4 Verification

The engine of Section 3 was written by an AI coding agent under human direction (see the statement in Section 10). We therefore treated it as untrusted and audited it in the following ways. Everything in this section refers to code that is available with the paper (Section 11).

4.1 Independent enumerator (oracle) and differential tests

A Python enumerator (`referee_forest_enum.py`), written independently of the C++ code from the definition in Section 3.1, generates *all* labelled join/attach/new children of every node without any symmetry rule, keeps a child iff its full distance multiset (recomputed from scratch by BFS) is duplicate-free and inside the target, and de-duplicates each level by a complete canonical form for distinct-weight forests without isolated vertices (the sorted multiset of per-vertex sorted incident-weight tuples). Its per-level counts equal the engine's for every level and every $n \le 12$ (at $n = 12$: $1, 1, 2, 8, 41, 229, 1384, 8877, 58933, 278539, 356479$ nodes on levels $0, \ldots, 10$, 23 minutes in Python), for planted targets with holes at $n = 6, \ldots, 9$, and for the prefix of the $n = 17$ and $n = 18$ trees through level 9 (at $n = 18$: 480,085 level-9 forests). A second Python mode that re-implements the engine's own generation rule *without* de-duplication produced zero canonical-form collisions in all these cases, i.e. the isomorph rejection is irredundant as implemented and not only as proved.

Two independent formulations of the same problem were used as further oracles: (a) a per-topology engine (Section 4.2) whose solution counts, summed over *all* unlabelled trees of a given order, must equal the forest engine's; this was checked on 45 instances ($n = 6, \ldots, 12$, standard and planted targets, including targets with two non-isomorphic solutions) with 0 mismatches; and (b) a plain edge-by-edge backtracking over target values with *no* forcing

[1] More precisely, the clean-room implementation of Section 4.5 found that this rule fires exactly once at $n = 12$ (removing one level-10 node) and never in the $n = 18$ tree; since it is a necessary condition it cannot remove a solution, and both engines apply it, so the per-level counts reported here are those *with* the rule.

lemma, summed over topologies as labellings divided by $|\operatorname{Aut}|$, for $n \le 9$. Full-depth planted instances at $n = 13, \ldots, 18$ (random weighted trees with distinct distances, custom target set) were run through the engine: in all 16 instances the planted tree was found among the solutions and every reported solution was re-checked; these are the only tests that exercise the $n = 18$, full-vertex-count code path with a witness. Finally the production binary was shown to be a faithful build of the archived source by reproducing its depth histograms at $n = 4, \ldots, 14$ and the exact node count (100,024,625) of one $n = 18$ production shard from a byte-for-byte rebuild.

4.2 Independent method: per-topology search

Before the forest engine we built a per-topology exact search (`leech_search.cpp`): for each of the 123,867 unlabelled trees on 18 vertices (OEIS A000055 [14]; enumerated by two independent generators, the WROM algorithm [25] as implemented in NetworkX [7] and nauty's `gentreeg` [13], which agree for $n = 5, 9, 11, 16, 18$: $3, 47, 235, 19320, 123867$) it branches on which unassigned edge receives the least missing value (Lemma 2.1) and prunes with distinctness, the containment and Golomb windows (Lemma 2.4), the cut identity (Lemma 2.3), Hall-type counting on value windows, Taylor's parity (Lemma 2.2), Lemma 2.6, and complete symmetry breaking over Aut of the topology (verified as *exactly one representative per orbit* by the identity $\text{count(sym)} \cdot |\operatorname{Aut}| = \text{count(nosym)}$ on planted instances with $|\operatorname{Aut}|$ up to 20,000). This engine was audited by a differential harness against a rule-free DFS and against a CP-SAT enumeration [16]: 574 instances (all topologies $n = 7, \ldots, 10$ with planted targets, the standard target $n = 7, \ldots, 9$, parity-skewed families, symmetry stress set) $\times$ 3 binaries $\times$ 32 flag subsets, and a second seed with 560 instances, with 0 discrepancies. It reproduces the literature id-by-id at $n = 9$ (47 topologies) and $n = 11$ (235), agreeing with CP-SAT and with the DRAT-certified SAT route (Section 4.3), and it has now decided *all* 19,320 topologies of order 16 on the cluster as UNSAT (19,308 with the first-generation engine at a 1800 s cap; the 12 topologies that hit the cap were re-run with the improved engine and finished in 673–1317 s each). This is an independent-method reproduction of the non-existence of a Leech tree of order 16 [3].

The per-topology engine is between three and four orders of magnitude slower in total than the forest engine (it re-derives the same small-weight forests once per topology: $n = 16$ took 321 CPU-s with the forest engine against roughly 500–700 CPU-h topology-by-topology), which is why it is used as a *cross-check* rather than as the primary proof.

At $n = 18$ this cross-check is *incomplete*, and we state its status precisely. A first pass over all 122,344 surviving topologies (those left after the proven filters of Section 5.2) has finished on the cluster with a 300 s time cap per topology: 73,021 topologies UNSAT, 49,323 undecided at the cap, and—this is the part that matters—0 topologies SAT. The 49,323 undecided topologies were re-run with the improved engine at a 3600 s cap; that run was stopped before it finished. As of 20 August 2026 it had raised the independent-method coverage to 112,082 of the 122,344 survivor topologies, i.e. 91.6%, with 1,169 re-run cases still undecided at the cap and no SAT result. The remaining 1,523 of the 123,867 topologies had been removed beforehand by the filters of Section 5.2, and the forest engine of Theorem 1.1 uses no topology filters at all. No topology anywhere in the project, at any order, has ever been reported SAT except for those carrying the three known Leech trees of orders 4 and 6. Until the re-run is complete, Theorem 1.1 rests on the forest engine, its audits (Section 4.1), the three bit-identical replications (Section 4.4) and the clean-room implementation (Section 4.5); the per-topology run is corroboration by a different method, not a second complete proof. We leave this cross-check at 91.6% deliberately, and we state the residue exactly, since it is easy to misread. Of the 122,344 survivors, 112,082 carry an independent-method verdict and 10,262 do not; of those, 1,169 were re-run and were still undecided at the 3600 s cap, and the remaining 9,093 were

never reached before the run was stopped. Finishing them means running the per-topology engine past its cap on precisely the instances it finds hardest, together with a first pass over the 9,093; the projected cost is 10^4–10^5 CPU-hours, and it buys no logical strength, since Theorem 1.1 does not rest on this method. A disagreement would announce itself as a SAT result, and no topology anywhere in this project has ever been reported SAT except those carrying the known trees of orders 4 and 6. Should any future run report SAT at $n = 18$, the theorem must be withheld pending reconciliation.

4.3 SAT encoding with checked proofs ($n \leq 11$)

For small orders we also produced machine-checkable certificates: an order-encoding CNF per topology (pair/value variables, ternary sum relations along a rooted tree, sibling-subtree symmetry breaking), solved with CaDiCaL [1, 2] (version 3.0.1; Kissat 4.0.4 was used as a second solver) producing DRAT proofs, checked with `drat-trim` [24]. All 47 topologies at $n = 9$ and all 235 at $n = 11$ are UNSAT with verified proofs (also the 5 non-Leech topologies at $n = 6$; the sixth yields the known tree); this reproduces [18] with independent certificates. The approach does not scale (30 random $n = 16$ topologies: 0/30 decided in 600 s each), so no DRAT-style certificate exists for $n = 18$; see Section 9.

4.4 Replication of the $n = 18$ run

The $n = 18$ forest search was run three times with the reference engine, with two shard layouts, on two architectures and with two compilers, and once more with the clean-room implementation of Section 4.5; the per-shard depth histograms were compared (Table 2).

Table 2: Replication of the $n = 18$ forest search. $\sum$nodes counts the shared prefix (levels $\leq L$) once per shard; "unique" $= \sum \text{nodes} - (K-1) \cdot 73{,}408$, where 73,408 is the number of nodes on levels $0, \ldots, 8$. Hoffman2 is the UCLA shared cluster (SLURM array jobs); "local" is an Apple M4 laptop.

run	machine / compiler	(K, L)	$\sum$nodes	unique nodes	levels 9–16	L17	trees
Hoffman2 job 83703	x86_64, gcc 11.5	(210, 8)	59,795,196,608	59,779,854,336	Table 4	0	0
Hoffman2 job 83752	x86_64, gcc 11.5	(175, 9)	59,876,162,118	59,779,854,336	identical	0	0
local, reference	arm64 M4, clang	(512, 8)	59,817,365,824	59,779,854,336	n/a[2]	—	0
local, clean-room	arm64 M4, clang	(100, 8)	59,787,121,728	59,779,854,336	identical	0	0

The four sums of visited nodes differ exactly by the multiplicity of the shared prefix ($(K-1) \cdot \text{prefix}(\leq L)$), and the unique node counts agree bit-for-bit. For the two cluster runs and the clean-room run the per-level sums agree with each other on every level, and agree with the Python oracle on levels ≤ 9. The node counts of the reference engine are deterministic, so its three-fold replication detects hardware, compiler and job-management faults (lost or truncated shards, mis-sharding), not algorithmic errors; the algorithmic checks are those of Sections 4.1–4.2 and 4.5.

4.5 Clean-room re-implementation

A second implementation of the forced-forest search (`cr_forest.c`, C, single file) was written from the algorithmic description of Section 3 and [3] only, by an agent that had no access to the production source; it uses its own state representation (state copied per DFS level,

[2]The local driver of the reference engine discarded the per-shard stderr histograms; only the node counts and status of each shard were kept. The identity of the unique node count with the two cluster runs is the check available for this run.

full within-component distance matrix, component identifiers by smallest label) and its own ordering of children, and shares with the reference engine only the mathematical rules of Proposition 3.3. It comes with its own brute-force Python oracle (all labelled children, BFS re-computation of distances, canonical-form de-duplication per level), with which it agrees at every level for $n = 4, \dots, 12$, and it passes the same sharding identities on 11 (n, K, L) cases. It reproduces the reference engine's per-level counts at every level for $n = 4, \dots, 16$ (at $n = 16$: 1,096,039,152 nodes, identical histogram), for levels $0, \dots, 12$ at $n = 17$, and—run over the full $n = 18$ tree with $K = 100$, $L = 8$ (34,207 CPU-s at low priority on the laptop)—all eighteen per-level counts of Table 4, the unique node count 59,779,854,336, and no tree on 18 vertices. The verdict script `cleanroom/verify_18.py` performs the shard-integrity checks of Section 3.3 and prints the comparison with the reference counts.

5 Results

5.1 Orders $n \leq 17$

The forest engine finds exactly the known Leech trees and no others: at $n = 4$ two (the star and the path), at $n = 6$ one (the double star), and none at $n = 5$ and $7 \leq n \leq 17$. (The condition of Lemma 2.2 is not imposed, so the non-admissible orders are searched as well and come out empty, as they must.) Node counts and CPU times (single core, Apple M4, under load) are given in Table 3.

Table 3: Forest search for $n \leq 17$ (target $[1, \binom{n}{2}]$). "Last levels" lists the node counts of the three deepest non-empty levels. For $n = 17$ the run was sharded ($K = 32$, $L = 8$); the unique node count subtracts the 31 repeated copies of the 73,408-node prefix.

n	nodes (unique)	CPU	Leech trees	last levels
11	123,309	0.05 s	0	L7 8,645; L8 43,855; L9 69,144
12	704,494	0.14 s	0	L8 58,933; L9 278,539; L10 356,479
13	4,178,290	0.9 s	0	L9 424,878; L10 1.86M; L11 1.82M
14	25,486,327	6.4 s	0	L10 3.25M; L11 13.0M; L12 8.66M
15	162,497,458	44 s	0	L11 26.2M; L12 93.5M; L13 38.4M
16	1,096,039,152	321 s	0	L12 220.0M; L13 683.5M; L14 154.7M
17	7,875,306,893	1.7 CPU-h	0	(histograms not retained)

The full level histogram at $n = 16$ (levels $0, \dots, 15$) is

1, 1, 2, 8, 41, 229, 1384, 8899, 62843, 480048, 3968025, 33269745, 220001476, 683511193, 154735257, 0.

The counts at $n = 16$ and $n = 17$ reproduce the non-existence results of [3]; those at $n = 9, 11$ reproduce [18], and are also confirmed by the per-topology engine, CP-SAT, and (for $n = 9, 11$) DRAT-checked SAT proofs. The per-level counts at low levels are the same for all n once n is large enough for the vertex bound not to bite $(1, 1, 2, 8, 41, 229, 1384, 8899, 62843, \dots)$, and grow by a factor that rises from 4 to about 7 across that range. The whole tree grows by a factor that is not constant either: from Table 3 and Table 4 the successive ratios for $n = 12, \dots, 18$ are 5.71, 5.93, 6.10, 6.38, 6.74, 7.19, 7.59, increasing throughout. Extrapolations below use the last of these and are correspondingly optimistic.

5.2 Order 18

Theorem 5.1 (Theorem 1.1, quantitative form)**.** *For $n = 18$, $N = 153$, the search tree of Section 3 has the per-level node counts of Table 4, a total of* 59,779,854,336 *nodes, and no*

node at level 17. *Consequently there is no Leech tree of order* 18.

Table 4: Number of non-isomorphic forced forests with k edges for $(n, N) = (18, 153)$. Verified identical in both cluster runs and in the clean-room implementation; levels ≤ 9 also by the Python oracle.

level k	0	1	2	3	4	5
forests	1	1	2	8	41	229
level k	6	7	8	9	10	11
forests	1,384	8,899	62,843	480,085	3,984,162	35,540,837
level k	12	13	14	15	16	17
forests	332,597,341	2,896,330,052	17,017,193,146	37,669,242,243	1,824,413,062	**0**

The tree dies at the last two levels: of the 1.8×10^9 distance-distinct forced forests with 16 edges (one edge short of a spanning tree on 18 vertices, or a spanning tree on 17 of them), none can be completed. The whole computation cost about 5–11 CPU-hours depending on machine and load (roughly 0.3 μs per node unloaded; 39,708 CPU-s for the local 512-shard run at low priority); on the cluster each of the 210 shard processes ran for about 1.5 minutes of wall time. For comparison, a per-topology search of the same order was projected at 10^4–10^5 CPU-hours.

We record for completeness that the published structural results prune very little at the topology level: of the 123,867 trees on 18 vertices, the diameter bound (Lemma 2.4(b)), the degree bound (Lemma 2.5), the exclusion of stars, double stars, paths and the broom of [22] remove 152, leaving 123,715. Two real relaxations then leave 122,344. Write A for the (pairs $\times$ edges) path-incidence matrix of the topology and $d = Aw$ for the distance vector. If the topology carries a Leech labelling then d is a permutation of $v = (1, \ldots, N)$, so (i) $d \prec v$ in the majorisation order, which we impose through the cutting planes "the sum of the s largest entries of d is at most $N + (N-1) + \cdots + (N-s+1)$"; and (ii) $\sum_P d_P = M_1 := N(N+1)/2$ while $\sum_P d_P^2 = M_2 := N(N+1)(2N+1)/6$, so the projection bound $\min\{\, w^{\mathsf{T}} A^{\mathsf{T}} A w : \mathbf{1}^{\mathsf{T}} A w = M_1 \,\} = M_1^2 / \|P_A \mathbf{1}\|^2$ must not exceed M_2. Both are solved numerically, at a tolerance of 10^{-6}, and every kill is re-verified by an independently formulated linear program with disagreements resolved in favour of keeping the topology; they are therefore not rigorous filters, and we quote 122,344 only to say how little the published structural results prune. The forest search does not use topologies at all, so nothing in Theorem 1.1 depends on either relaxation.

6 A uniform theorem for spiders

We now prove Theorem 1.2. This computation is independent of the forced-forest computation above: it uses the geometry of a spider and works downwards from the largest distances. Give the centre coordinate zero and represent each leg by the strictly increasing positive cumulative weights of its vertices. Distances are then centre marks, differences of marks on one leg, and sums of marks on different legs. For a path we choose any internal vertex as centre. Let $T > U$ be the two largest leg tips. Since all distances are distinct, the tips are distinct; since the largest distance joins the two largest tips,

$$T + U = N = \binom{n}{2}. \tag{1}$$

6.1 Complete top-down search

Start with the two pinned tips T, U and the realised distances T, U, N. Process $N-1, N-2, \ldots$ in descending order. When a target v is absent, enumerate every way to realise it: the centre and one new mark; an existing mark and one new mark, on the same or on a different leg; or two new marks, again on the same or on different legs. An option is retained exactly when every distance newly created by its one or two marks is positive, at most N, and distinct from every old and new distance. Existing legs and the necessary one- and two-fresh-leg cases are all included. The top leg is capped at T, the distinguished second leg at U, and every other leg strictly below U.

Lemma 6.1. *If a state is contained in a Leech-labelled spider, then at the next missing target at least one child retained by the top-down search is contained in that spider.*

Proof. Choose the pair realising the target in the completed spider. Zero, one or two of its endpoints are already pinned. The cases just listed place precisely the unpinned endpoints, including their actual old or fresh legs. All distances created by those endpoints occur in the completed Leech spider, hence lie in $[1, N]$ and are distinct, so that option is retained. No symmetry reduction identifies the distinguished T- and U-legs; only otherwise interchangeable fresh legs are canonically renamed. □

Thus a closed search tree is a proof of non-existence for its anchor (T, U). The exact program `spider_window_probe.py` implements this search; its paranoid mode recomputes the complete distance set from scratch at every state.

6.2 The regular region

Put $\delta = T - U$ and consider $N - k$. If $k \le W$, $U > W$ and $\delta > W$, then $N - k$ cannot be a centre mark or a same-leg difference: each is at most T, which would imply $U \le k$. Nor can it be the sum of two marks on lower legs: that sum is at most $2U$, which would imply $\delta \le k$. Every realiser therefore joins the top leg to a lower leg and has the form

$$(T - a) + (U - b) = N - k, \qquad a, b \ge 0, \quad a + b = k. \tag{2}$$

The finite program `spider_offset_prover.py` enumerates the resulting offset states. It enforces only necessary conditions: distinct high sums $a+b$, distinct same-leg offset differences, distinct sums between different lower legs, and distinct centre marks. In particular it omits collision classes rather than adding any, so its tree is a relaxation (a supertree) of the exact spider search. Lower legs other than the distinguished U-tip leg are canonically permuted. At $W = 10$ the relaxation closes: it has 111 states, 110 accepted children and 57 dead states, reaches missing offset 10, and has no frontier state at offset 11. By Lemma 6.1,

$$U > 10 \text{ and } \delta > 10 \quad \Longrightarrow \quad \text{no Leech spider extends the anchor } (T, U). \tag{3}$$

6.3 Affine boundary strips

It remains to handle $U \le 10$ or $\delta \le 10$. For each fixed $s \in \{1, \ldots, 10\}$ introduce an unbounded parameter P:

$$\begin{aligned} \text{small-}U\text{:} &\quad U = s, \quad P = \delta, \quad T = P + s, \quad N = P + 2s; \\ \text{small-}\delta\text{:} &\quad \delta = s, \quad P = U, \quad T = P + s, \quad N = 2P + s. \end{aligned}$$

The programs store every mark and distance as an affine form $qP + c$ and run the same centre/same-leg/cross-leg case split as the exact search through the first 25 offsets.

The quantifier in this step deserves care: a spider at one numerical value of P need not extend to a family of spiders. What is needed is that every *finite high-window branch* at every $P \geq 51$ has an affine encoding. This follows by induction on the descending targets. In the small-U strip, all lower-leg marks are bounded constants; a target $N - k = P + (2s - k)$ can only introduce a top mark $P + c$, a bounded constant top mark paired by a same-leg difference with a $P + c$ mark, or a bounded lower mark. These are exactly the affine types enumerated by the program. Notice in particular that bounded constant marks on the top leg are retained. In the small-δ strip, for $P \geq 51$ and $k \leq 25$ the target $2P + s - k$ is too large to be a centre mark or a same-leg difference. If $x + y = 2P + s - k$ is a cross-leg realisation, the cap $x, y \leq P + s$ gives

$$P - k \leq x, y \leq P + s;$$

hence both endpoints have the form $P + c$. Addition to or subtraction from an already affine endpoint preserves the asserted types. This completes the induction.

For each type the coefficient is therefore fixed and the cap inequalities give a finite interval of possible constants c. The affine search enumerates that interval exactly. Every sign, order, cap comparison and equality is then checked uniformly for all real $P \geq 51$; execution aborts if a comparison can change on that tail. The cutoff is sharp for this computation in the following sense: at $P = 50$ the reached comparison $25 = P - 25$ changes status, while all comparisons reached for $P \geq 51$ are stable.

The resulting finite computations are in Table 5. The production implementation maintains distances incrementally. A second implementation, `spider_affine_cleanroom.py`, imports no production code, uses immutable leg states and recomputes every distance after every addition. It agrees field-for-field in all 20 regimes. This is an independent implementation of the affine search, not an independent mathematical proof: the two programs deliberately share the affine abstraction and realiser case split. Exact numerical searches at $P = 51, 52, 137$ provide 60 additional regression checks; they are not used to justify the infinite quantifier.

Table 5: Finite certificate for the affine boundary strips. "Frontier" is the number of states surviving the stated window; zero is the conclusion used in the proof.

regimes	number	states	deepest missing offset	max marks	frontier
$U = 1, \ldots, 10$, $\delta \geq 51$	10	209	9	6	0
$\delta = 1, \ldots, 10$, $U \geq 51$	10	914	25	9	0

6.4 Finite base and conclusion

The exact numeric search closes every order $n = 5, \ldots, 15$, with no solution among 16,153 visited states. For $n \geq 16$, $N \geq 120$. If $U \leq 10$ and $\delta \leq 50$, then $N = 2U + \delta \leq 70$, a contradiction; otherwise the anchor lies in the small-U affine tail. If $\delta \leq 10$ and $U \leq 50$, then $N = 2U + \delta \leq 110$, again a contradiction; otherwise it lies in the small-δ affine tail. Every remaining anchor has $U > 10$ and $\delta > 10$ and is excluded by (3). These alternatives partition all $T > U \geq 1$, proving the following quantitative form of the second main theorem.

Theorem 6.2 (Theorem 1.2, certificate form)**.** *The exact searches for* $5 \leq n \leq 15$*, the regular relaxation with window* 10*, and the* 20 *affine boundary searches with window* 25 *all have zero frontier or solution count. Consequently no spider of order* $n \geq 5$ *is a Leech tree.*

The complete verifier is `verify_spider_uniform.py`. It recomputes all counts above rather than trusting the archived JSON transcript, compares both affine implementations and runs the 60 numerical checks. The archived JSON is therefore a regression target, not a standalone proof object. The reported run used CPython 3.14.7; the verifier uses explicit

exceptions for certificate invariants and refuses to run under `python -O` or `-OO`. No SAT witness is produced, so the two witness checkers used elsewhere in the project are not part of this theorem's trust boundary. An independent adversarial review reproduced the full verifier, the paranoid finite base, 2,000 boundary anchors and 2,500 regular anchors, and found no missing regime or unsound pruning rule; its report and the source revision reviewed are included in the archive.

7 Neighbouring problems settled by the same engines

The forced-forest search of Section 3 and the per-topology search of Section 4.2 are not specific to the target $\{1, \ldots, N\}$. With small modifications they decide three neighbouring problems on which the literature stops short of exact values, and on which the published bounds are, in each case, more than twenty years old or explicitly left open. All results in this section were obtained after the $n = 18$ computation, with copies of the engines kept separate from the production sources; every witness reported below was re-checked by an independent breadth-first-search checker that shares no code with any engine, and the ones printed here are small enough to be verified by hand.

7.1 Minimal distinct distance trees: $M(11) = 60$ and $M(12) = 77$

Calhoun, Ferland, Lister and Polhill [3] define: "If all of these distances are distinct, we call the tree a distinct distance tree. Define the function $M(n)$ to be the smallest integer so that there exists a distinct distance tree with n vertices and maximum distance $M(n)$" [3, p. 33]; a *minimal distinct distance tree* is one attaining $M(n)$, and a Leech tree is exactly a distinct distance tree with $M(T) = \binom{n}{2}$. They determine $M(n)$ for $n \leq 10$ $(0, 1, 3, 6, 11, 15, 22, 30, 39, 50)$ and give the bounds of their Table 1 for $11 \leq n \leq 18$, among them $59 \leq M(11) \leq 77$, $69 \leq M(12) \leq 94$ and $80 \leq M(13) \leq 119$, with the remark that "it will take too long for the algorithm to check higher numbers of gaps and find the exact value of $M(n)$" [3, p. 44].

The forest engine adapts to $M(n)$ by replacing the forcing lemma with a three-way decision at each value level t (all values $< t$ decided): either t is already a distance, or t is declared a *gap* (never a distance; at most $D - \binom{n}{2}$ gaps are available when the target maximum is D), or t is the weight of the next edge, added by join, attach or new as before. Every state (forest, t) is again reached by exactly one decision sequence, so the isomorph rejection of Proposition 3.3 carries over verbatim and the number of solutions at $D = M(n)$ is the number of minimal trees up to isomorphism. Feasibility is decided for increasing D; the run at $D = M(n) - 1$ is the exclusion. The engine reproduces all of Calhoun's values and, where they are stated, their counts of minimal trees ($n = 7$: 2; $n = 8$: 2; $n = 9$: 1; $n = 10$: 1, with the same weights $1, 2, 3, 4, 5, 11, 14, 18, 28$), at costs from 215 nodes ($n = 7$, $D = 21$) to 2.2×10^7 nodes ($n = 10$, $D = 49$).

Theorem 7.1. *$M(11) = 60$ and $M(12) = 77$. At each order there are exactly two minimal distinct distance trees. The two trees on 11 vertices, namely (edge lists u–v:w)*

0–1:1, 0–2:2, 3–4:4, 5–6:5, 5–7:6, 3–8:8, 3–5:9, 3–9:16, 2–7:26, 7–10:27,
0–1:1, 0–2:2, 3–4:4, 5–6:5, 5–7:6, 3–8:8, 3–5:9, 3–9:16, 7–10:26, 0–7:27,

whose 55 distances are $\{1, \ldots, 60\}$ minus $\{7, 10, 21, 36, 54\}$ and $\{7, 10, 21, 36, 56\}$ respectively. The two trees on 12 vertices are

0–1:1, 2–3:2, 4–5:3, 2–6:4, 4–7:5, 0–8:9, 4–9:12, 2–7:14, 7–10:20, 9–11:29, 8–9:30,
0–1:1, 2–3:2, 4–5:3, 2–6:4, 4–7:5, 0–8:9, 4–9:12, 3–7:14, 7–10:18, 9–11:29, 8–9:30.

Their 66 *distances have maximum* 77 *and omit from* $\{1,\dots,77\}$*, respectively,*

$$\{7,11,13,27,32,43,48,49,50,53,58\},$$
$$\{7,11,13,27,36,43,48,49,50,53,58\}.$$

Moreover $M(13)\le 110$*, witnessed by*

$$0\text{–}1{:}1,\ 0\text{–}2{:}2,\ 3\text{–}4{:}4,\ 5\text{–}6{:}5,\ 0\text{–}3{:}6,\ 5\text{–}7{:}13,\ 8\text{–}9{:}14,\ 4\text{–}8{:}15,$$

$$6\text{–}9{:}16,\ 8\text{–}10{:}17,\ 9\text{–}11{:}37,\ 9\text{–}12{:}69.$$

The computations behind Theorem 7.1 are as follows (all on the cluster unless stated; "UNSAT" means the exhaustive run at that D found no tree and all shards completed). At $n=11$: $D=59$ UNSAT (1.80×10^8 nodes locally, 1.83×10^8 in 7 shards on the cluster), $D=60$ two trees (5.86×10^8 nodes, 111 CPU-s). For $D=61,\dots,65$ the numbers of distinct distance trees with maximum distance exactly D are $0,6,3,12,32$. As an independent-method check, a per-topology backtracking program with no forcing lemma, no forest search and no isomorph rejection was run over all 235 topologies of order 11: at $D=59$ it finds 0 labellings in 12,620,879,225 nodes and at $D=60$ it finds 18 labellings in 15,106,490,676 nodes, with identical node counts and identical solution lists on two machines (arm64/clang and x86_64/gcc); the 18 labellings are exactly the two trees above counted with their automorphisms (6 and 12). The same check at $n=10$ gives 0 labellings at $D=49$ and 6 (one tree, $|\operatorname{Aut}|=6$) at $D=50$. At $n=12$ the exhaustive runs at $D=69,70,\dots,76$ are all UNSAT, at costs 3.7×10^8, 1.5×10^9, 5.2×10^9, 1.6×10^{10}, 4.2×10^{10}, 1.0×10^{11}, 2.4×10^{11} and 5.1×10^{11} nodes, so $M(12)\ge77$; and the witness-hunting mode found the tree

$$0\text{–}1{:}1,\ 0\text{–}2{:}2,\ 3\text{–}4{:}4,\ 5\text{–}6{:}5,\ 0\text{–}3{:}6,\ 5\text{–}7{:}13,\ 8\text{–}9{:}14,\ 4\text{–}8{:}15,\ 6\text{–}9{:}16,\ 8\text{–}10{:}17,\ 9\text{–}11{:}37$$

with all 66 distances distinct and maximum 78. The exhaustive run at $D=77$ subsequently completed in 70 shards: all shards reported `DONE`, the aggregate search contained 1,020,925,849,186 nodes, and it found exactly the two trees printed in Theorem 7.1. Their distance sets were recomputed independently by both breadth-first and rooted-depth/LCA checkers. Together with the exhaustive exclusions through $D=76$, this proves $M(12)=77$. Appending the leaf 9–12:69 to the 12-vertex witness of maximum distance 78 displayed above gives the 13-vertex tree printed in Theorem 7.1, whose 78 distances are distinct with maximum 110; we verified it by two independent distance computations, a traversal and a lowest-common-ancestor path sum. An earlier run of this project recorded the bound $M(13)\le105$, by appending a leaf of weight 62 to a 12-vertex witness of maximum distance 79; that witness was not archived and we have not been able to recover it, so we withdraw the bound 105 and state only what we can exhibit. Either way the bound improves the $M(13)\le119$ of [3]; whence $M(13)\le110$ (Calhoun: 119).

7.2 Modular Leech trees: none of order 9 or 11, but two of order 5

Leach and Walsh introduced the modular relaxation, and Leach [9] states it as: "Let T be a tree on n vertices and let $k=\binom{n}{2}+1$. We say that T is a modular Leech tree if there exists an edge weighting function $w: E(T)\to\mathbb{Z}_k$ such that each of the $\binom{n}{2}$ paths within T has a distinct weight from $1,2,\dots,\binom{n}{2}$ with the sums taken modulo $\binom{n}{2}+1$" [9, p. 1]; equivalently w induces a bijection between the paths of T and $\mathbb{Z}_k\setminus\{0\}$. Every Leech tree is a modular Leech tree, so orders $2,3,4,6$ occur. Leach [9, Thms. 1–2] shows that if $n\equiv2,3 \pmod 4$ then a modular Leech tree of order n forces $n=m^2+2$ (so none exist for $n=7,10,14,15,\dots$), that multiplying a labelling by a unit of $\mathbb{Z}_k$ gives a labelling, and, by computer search, that

there is exactly one modular Leech tree of order 8 (over $\mathbb{Z}_{29}$, unique up to group and graph isomorphism). Nothing is stated there about $n \geq 9$.

Our engine here is a per-topology depth-first search over the frozen topology lists, placing edges in breadth-first order from a maximum-degree root and maintaining the set of realised residues in a 128-bit word; solutions are counted as orbits under the unit action of $\mathbb{Z}_k$. Two implementations with different symmetry breaking (restriction of the first label versus a lexicographic-minimum test over the unit stabiliser, the latter with forward checking) were written and give identical orbit counts for $n = 4, \ldots, 9$; brute force over all $(k-1)^{n-1}$ labellings independently confirms $n = 4, 5, 6$; and at $n = 8$ the search reproduces Leach exactly, finding the single topology of his Figure 3 with one labelling up to the leaf swap, our labelling being 3 times his in $\mathbb{Z}_{29}$.

Table 6: Modular Leech trees of order n over $\mathbb{Z}_k$, $k = \binom{n}{2} + 1$. "Orbits" counts $\mathbb{Z}_k$-Leech labellings up to multiplication by a unit (graph automorphisms are not divided out). New results in bold.

n	k	topologies	trees / orbits	search nodes	remark
4	7	2	2 / 4	16–18	Leech's two trees
5	11	3	2 / 4	58–127	= [10, Lem. 4.2]; see below
6	16	6	2 / 34	1.7×10^3–2.2×10^4	the Leech tree and one other
7	22	11	0	3.1×10^4–4.6×10^5	agrees with [9, Thm. 2]
8	29	23	1 / 2	2.4×10^5–7.6×10^5	= [9]
9	37	47	**0**	6.5×10^6–1.8×10^7	new
10	46	106	0	3.6×10^8–1.4×10^{10}	consistent with [9, Thm. 1]
11	56	235	**0**	1.3×10^{10}	new; single implementation

Table 6 summarises the outcome. The two new non-existence results are $n = 9$ and $n = 11$; both orders are permitted by Leach's theorems ($9 = 3^2$, $11 = 3^2 + 2$), so they were open. The order-9 result is produced by both implementations; the order-11 result comes from the faster implementation only (the slower one did not finish), and we label it accordingly as a single-implementation computation.

The order-5 entry is a reproduction of a published result, not a discovery, and what is new here is only the correction of a later summary. Leach writes: "By Leach and Walsh [6] and Theorem 2 we know that none exist for 5 or 7" [9, p. 2], his [6] being [10]. His Theorems 1–2 exclude $n = 7$; they say nothing about $n = 5$ (as $5 \equiv 1 \pmod 4$), so the order-5 statement rests entirely on [10]. That paper says the opposite. Its Lemma 4.2 reads, in full, "All trees on five vertices except for $K_{1,4}$ are $\mathbb{Z}_{11}$-Leech trees", and its proof prints a labelling for each: for the path, the edge labels $1, 9, 7, 8$ in sequence, which a computer search there shows to be unique up to group and graph automorphisms; for the tree of degree sequence $(3, 2, 1, 1, 1)$, the label 8 on the central edge, 1 and 2 on the pendant edges at the degree-3 vertex and 7 on the pendant edge at the degree-2 vertex. The sentence quoted from [9] therefore misreports [10], and it is that misreport, not the existence of the trees, that we correct.

Our search reproduces exactly those two labellings. Over $\mathbb{Z}_{11}$ it returns the path and the spider

$$0\text{–}1{:}5,\ 1\text{–}2{:}1,\ 2\text{–}3{:}2,\ 3\text{–}4{:}7 \qquad \text{and} \qquad 0\text{–}1{:}1,\ 1\text{–}2{:}5,\ 0\text{–}3{:}3,\ 0\text{–}4{:}7,$$

with 4 unit-orbits in total; the first is 5 times the labelling of [10, Lem. 4.2] and the second is 7 times it, so the two agree as unit-orbits and the entry in Table 6 is a reproduction. The ten path sums of the first are $5, 1, 2, 7, 6, 3, 9, 8, 10$ and $15 \equiv 4$, and those of the second are $1, 5, 6, 3, 7, 4, 8, 9, 13 \equiv 2$ and 10: in both cases exactly $\mathbb{Z}_{11} \setminus \{0\}$, as the reader can check in a minute. Both labellings use distinct nonzero labels, so they also satisfy the more restrictive readings of the definition. We recomputed them by exhaustive search over all 10^4 labellings of

all three topologies as well as by both search engines. The discrepancy is thus located exactly: it lies in the summary sentence of [9], not in [10] and not in any difference of definition. Everything else we compute agrees with [9].

7.3 Leaf-Leech trees: six leaves, and Question 3.1 of Ozen–Wang–Yalman

Ozen, Wang and Yalman [15] define: "A (not weighted) tree T with n leaves is a leaf-Leech tree if the distances between pairs of leaves are exactly $3, 4, \ldots, \binom{n}{2}+2$" [15, Def. 1] (distances 1 and 2 between leaves are impossible except in trivial cases, whence the shift). They prove the Taylor analogue (the number of leaves is k^2 or k^2+2), that the *expansion* T^e of a Leech tree—subdivide each edge into a path of its weight and hang a pendant vertex at every original vertex—is leaf-Leech, and, conversely, that a leaf-Leech tree with no *irreducible* vertex (a vertex "with degree at least three and no leaf neighbors" [15, p. 4]) is the expansion of a Leech tree. They then ask [15, Question 3.1]: "Does there exist a leaf-Leech tree that contains an irreducible vertex?", remarking that they have not been able to find a leaf-Leech tree that is not the expansion of a Leech tree; before the present note the largest number of leaves for which any leaf-Leech tree was known was 6, from Leech's order-6 tree.

Contracting the degree-2 vertices turns a leaf-Leech tree with L leaves into a series-reduced tree (all internal degrees ≥ 3) with L leaves and positive integer edge weights whose $S = \binom{L}{2}$ weighted leaf-to-leaf distances are exactly $\{3, \ldots, S+2\}$; conversely every such weighted tree expands to a leaf-Leech tree. The series-reduced topologies ($L+1 \leq |V| \leq 2L-2$) were enumerated with nauty's `gentreeg` and cross-checked against NetworkX, and each was solved with CP-SAT [16] in enumerate-all-solutions mode (weights in $[1, S+2]$, an `AllDifferent` over the S leaf-pair distance variables constrained to $[3, S+2]$, which forces the bijection); solutions were de-duplicated modulo isomorphism of the expanded unweighted tree and re-checked by the independent checker.

Proposition 7.2. *There are exactly six leaf-Leech trees with six leaves. Exactly one of them, the expansion of Leech's order-*6 *tree, has no irreducible vertex; each of the other five contains at least one (the witness displayed below has two). In particular the answer to Question 3.1 of [15] is yes.*

For $L = 3$ and $L = 4$ the enumeration gives 1 and 2 leaf-Leech trees, namely the expansions of the Leech trees of those orders, and for $L = 5$ none (as Proposition 1 of [15] already excludes). That exactly one of the six trees at $L = 6$ is irreducible-vertex-free is a consistency check on the enumeration, since by their Theorem 2 such a tree must be the expansion of a Leech tree of order 6, and there is exactly one of those. An explicit witness for Proposition 7.2, in contracted form, is

$$1\text{–}0{:}1,\ 1\text{–}2{:}6,\ 1\text{–}5{:}4,\ 0\text{–}6{:}1,\ 0\text{–}9{:}1,\ 2\text{–}3{:}6,\ 2\text{–}4{:}2,\ 6\text{–}7{:}3,\ 6\text{–}8{:}1,$$

whose expansion has 26 vertices and six leaves $3, 4, 5, 7, 8, 9$ at the fifteen pairwise distances $3, 4, \ldots, 17$; the vertices 1 and 2 have degree 3 and, in the expanded tree, no leaf neighbour, so both are irreducible. This tree is therefore not the expansion of any Leech tree.

The next admissible numbers of leaves are $L = 9$ and $L = 11$ (73 and 488 series-reduced topologies), where CP-SAT is no longer adequate—exactly as in the Leech problem itself, it decides the small cases and then stalls (the 10-vertex topology at $L = 9$ is infeasible in 41 s, the 11-vertex ones were undecided after 3600 s). A forcing-style engine over leaf distances, analogous to Section 3, would be the natural next step; we leave it open.

7.4 Status of the results of this section

These are computational results of the same kind as Theorem 1.1, with weaker but explicitly stated verification. The existence statements (the two minimal trees at $n = 11$, the two

minimal trees at $n = 12$, the modular trees of order 5, the six leaf-Leech trees) are witnesses, printed above or in the archive, and are checkable by hand or by any independent checker; they do not depend on the search being exhaustive. The non-existence statements ($M(11) \geq 60$, $M(12) \geq 77$, no modular Leech tree of order 9 or 11, only six leaf-Leech trees at $L = 6$) do depend on exhaustiveness: $M(11) \geq 60$ and the count of minimal trees at $n = 11$ are confirmed by a second, structurally different program on two architectures; the modular order-9 result by two implementations with different symmetry breaking; the modular order-11 result by one implementation only; the leaf-Leech enumeration by a single CP-SAT model over an independently cross-checked topology list. We label them accordingly and regard the modular order-11 result as the least independently supported of the group.

8 Removing a diametral endpoint: top-block crowding at order 25

With $n = 18$ settled, the smallest open order is $n = 25$, where $N = \binom{25}{2} = 300$. The extrapolation in Section 9 puts a direct forced-forest search at that order four to five orders of magnitude out of reach, so we look for structure instead. This section studies one invariant of a Leech tree T of order n, namely the weighted diameter of $T - \ell$ as ℓ ranges over the leaves of T. Two facts drive everything: only the two endpoints of the (unique) diametral pair are worth deleting, and for those the deleted-leaf spectrum forces a long block of consecutive rooted depths, which in turn forces so many repeated distances that the block cannot be long. The result is an unconditional lower bound on $\operatorname{diam}(T - \ell)$ at order 25 (Theorem 8.9), a strengthening of it by four finite computer searches (Section 8.4), and a reduction of the order-25 problem to fourteen explicit finite cases (Corollary 8.11). We prove nothing about the existence of a Leech tree of order 25.

Throughout this section $\operatorname{Spec}(K)$ denotes the set of pairwise distances of a weighted tree K and $\operatorname{diam}(K) = \max \operatorname{Spec}(K)$; both refer to weighted, not hop, distance.

8.1 The leaf spectrum identity and the top block

The following two lemmas are elementary. They were recorded in the notes of this project for the special case in which the deleted leaf hangs on the globally heaviest edge of T; that hypothesis is not needed, and we state and use them for an arbitrary leaf.

Lemma 8.1 (leaf spectrum identity)**.** *Let T be a Leech tree of order $n \geq 3$ and $N = \binom{n}{2}$. Let ℓ be a leaf of T, let x be its neighbour and $s = w(\ell x)$. Put $K = T - \ell$, $k = n - 1$, $F = \operatorname{Spec}(K)$ and $D = \{d(x, v) : v \in V(K)\}$. Then D consists of k distinct non-negative integers, $0 \in D$, $|F| = \binom{k}{2}$, and*

$$F \sqcup (s + D) = \{1, 2, \ldots, N\}. \tag{4}$$

Proof. Every pair of vertices of T either lies in K, contributing an element of F, or has ℓ as one endpoint, contributing $d(\ell, v) = s + d(x, v)$ for $v \in V(K)$. These $\binom{k}{2} + k = \binom{n}{2} = N$ values are the N distances of T, hence pairwise distinct and equal to $\{1, \ldots, N\}$ as a set. In particular the k values $s + d(x, v)$ are distinct, so the k depths $d(x, v)$ are distinct; $d(x, x) = 0$ gives $0 \in D$. □

Lemma 8.2 (top block)**.** *In the situation of Lemma 8.1 put*

$$\delta = \operatorname{diam}(K), \qquad r = \delta - \binom{k}{2}, \qquad m = N - \delta, \qquad A = N - s, \qquad B = A - m + 1 = \delta + 1 - s.$$

Then $r \geq 0$, $m = k - r$, $\max D \leq A$ with equality when $m \geq 1$, and

$$D = L \sqcup \{B, B+1, \ldots, A\}, \qquad |L| = r. \tag{5}$$

Moreover $r = 0$ holds if and only if K is itself a Leech tree of order k, and if $r \geq 1$ then $B \geq 1$, $0 \in L$ and $L \subseteq [0, B-1]$.

Proof. F consists of $\binom{k}{2}$ distinct integers in $[1, \delta]$ with $\delta \in F$, so $\delta \geq \binom{k}{2}$ and exactly $r = \delta - \binom{k}{2} \geq 0$ integers of $[1, \delta]$ lie outside F. By (4) those are cross values, so $r = |\{d \in D : s+d \leq \delta\}|$ and the remaining $k - r$ cross values fill $\{\delta+1, \ldots, N\}$ exactly; there are $N - \delta = m$ of these, whence $m = k - r$ and

$$s + D \supseteq \{\delta+1, \ldots, N\}, \qquad \text{i.e.} \qquad D \supseteq \{\delta+1-s, \ldots, N-s\} = \{B, \ldots, A\}.$$

Every cross value is at most N, so $\max D \leq N - s = A$; and if $m \geq 1$ then $N \notin F$, so N is a cross value and $\max D = A$. Setting $L = D \setminus [B, A]$ gives (5) with $|L| = k - m = r$ (for $m = 0$ the block is empty and $L = D$).

If $r = 0$ then $F = [1, \delta]$ and $|F| = \binom{k}{2} = \delta$, so the $\binom{k}{2}$ distances of K are exactly $1, \ldots, \binom{k}{2}$ and K is a Leech tree of order k; the converse is immediate. Now let $r \geq 1$. The k elements of D are distinct non-negative integers, so $A \geq \max D \geq k - 1$ and therefore $s = N - A \leq N - k + 1 = \binom{k}{2} + 1 = \delta - r + 1 \leq \delta$, which is $B = \delta + 1 - s \geq 1$. As $0 \in D$ and $0 < B$, we get $0 \in L$, and $L \subseteq [0, B-1]$ by construction. □

Remark. At $n = 25$ the alternative $r = 0$ of Lemma 8.2 cannot occur, since it would make K a Leech tree of order 24 and 24 is neither a square nor a square plus two (Lemma 2.2). So $r \geq 1$, $0 \in L$ and $B \geq 1$ throughout the order-25 discussion.

Call the m vertices of K whose depth lies in $[B, A]$ *high*, writing h_y for the one at depth $B+y$ ($0 \leq y \leq m-1$), and the r vertices whose depth lies in L *low*. Assume $r \geq 1$, equivalently $s \leq \delta$, equivalently x itself is low. Because all edge weights are positive, the depth of a vertex is strictly larger than the depth of its parent, so every ancestor of a low vertex is low: the low vertices form a subtree of K containing the root x, and no high vertex is an ancestor of a low one. Note also that the r distinct low depths must fit into $[0, B-1]$, which forces $B \geq r$, that is

$$s \leq \delta + 1 - r. \tag{6}$$

Example 8.3. Leech's order-6 tree of Table 1 has $N = 15$ and $d(2, 5) = 15$. Deleting the leaf $\ell = 5$ gives $s = 8$, $x = 3$, $k = 5$, $D = \{0, 4, 5, 6, 7\}$ and $F = \{1, 2, 3, 4, 5, 6, 7, 9, 10, 11\}$, so that $\delta = 11$, $r = 1$, $m = 4$, $A = 7$, $B = 4$ and $L = \{0\}$: a top block of four consecutive depths $4, 5, 6, 7$ above the single low vertex x.

The parity count of Lemma 2.2, applied to (4) instead of to T itself, gives a second constraint.

Lemma 8.4 (parity of the depth set)**.** *In the situation of Lemma 8.1, let E and O be the numbers of even and of odd elements of D. Then*

$$E\,O + O = \lceil N/2 \rceil \quad \text{if } s \text{ is even}, \qquad E\,O + E = \lceil N/2 \rceil \quad \text{if } s \text{ is odd}.$$

Proof. In a tree $d(u, v) = d(x, u) + d(x, v) - 2\,d(x, \mathrm{lca}(u, v))$, so $d(u, v)$ has the parity of $d(x, u) + d(x, v)$ and F contains exactly $E\,O$ odd values. The set $s + D$ contains O odd values if s is even and E of them if s is odd, and $[1, N]$ contains $\lceil N/2 \rceil$ odd values. Now use (4). □

At $n = 25$ we have $E + O = 24$ and $\lceil N/2 \rceil = 150$, so s even forces $O(25 - O) = 150$ and s odd forces $E(25 - E) = 150$; hence

$$(E, O) \in \{(14, 10), (9, 15)\} \text{ if } s \text{ is even}, \qquad (E, O) \in \{(15, 9), (10, 14)\} \text{ if } s \text{ is odd}. \tag{7}$$

Combining (7) with (5) pins the parities of s and of the low depths. Since $A = 300 - s$ and $B = A - m + 1$, the parities of the m consecutive high depths are determined by that of s, and D adds to them the depth 0 and the $r - 1$ remaining low depths; enumerating the 2^r parity patterns of $(s, l_1, \ldots, l_{r-1})$ and keeping those satisfying (7) leaves, for instance, no pattern at all at $\delta = 277$ and $\delta = 278$; exactly one at $\delta = 279$ (s, l_1, l_2 all even); exactly one at $\delta = 280$ (s and all three low depths even); and five at $\delta = 281$ (s even with exactly one low depth odd, in four ways, or s odd with all four low depths odd). These counts are recomputed by the arithmetic verifier of Section 8.4. The crowding count of Section 8.3 excludes every $\delta \le 280$ on its own, so we do not use Lemma 8.4 to exclude anything; its role is to restrict the parities of s and of the low depths in the cases that remain, and it is the kind of constraint a search at larger δ will need, since the class conditions used in Section 8.4 are parity-blind.

8.2 Which leaf to delete

Since the N distances of a Leech tree are distinct, the value N is attained by exactly one pair $\{a, b\}$, and a, b are leaves: if a had a neighbour c off the a–b path then $d(c, b) = d(a, b) + w(ac) > N$. For every leaf $\ell \notin \{a, b\}$ the pair $\{a, b\}$ survives in $T - \ell$, so $\operatorname{diam}(T - \ell) = N$ and Lemma 8.2 is vacuous ($m = 0$). The next theorem says that the two interesting leaves behave differently from each other, and pins one of them exactly.

Theorem 8.5 (diametral endpoint reduction)**.** *Let T be a Leech tree of order $n \ge 3$, $N = \binom{n}{2}$, and let $\{a, b\}$ be the unique pair with $d(a, b) = N$. Then the value $N - 1$ is attained by a pair containing a or b. Consequently the two endpoints can be named so that $d(a, w) = N - 1$ for some vertex w, and then*

$$\operatorname{diam}(T - b) = N - 1, \qquad \operatorname{diam}(T - a) \le N - 2.$$

With that naming, moreover, every pair of vertices avoiding both a and b has distance at most $N - 3$; in particular, if $\operatorname{diam}(T - a) = N - 2$ then the pair attaining it contains b.

Proof. Let P be the a–b path. For a vertex $u \notin \{a, b\}$ let $q(u)$ be the vertex of P nearest to u and put $p(u) = d(a, q(u))$ and $h(u) = d(u, q(u)) \ge 0$, so that $d(a, u) = p(u) + h(u)$ and $d(b, u) = \big(N - p(u)\big) + h(u)$. Define the deficits

$$e(u) = N - d(a, u) = N - p(u) - h(u), \qquad f(u) = N - d(b, u) = p(u) - h(u).$$

Both are at least 1: $d(a, u) < N$ because $\{a, b\}$ is the only pair at distance N and $u \ne b$, and likewise $d(b, u) < N$ because $u \ne a$.

Let $u \ne v$ be two vertices outside $\{a, b\}$, labelled so that $p(u) \le p(v)$. If $q(u) \ne q(v)$, the u–v path leaves u, meets P at $q(u)$, runs along P to $q(v)$ and leaves P to v, so

$$d(u, v) = h(u) + \big(p(v) - p(u)\big) + h(v) = N - f(u) - e(v).$$

The same holds if $q(u) = q(v)$ and the two paths from u and v to P leave that common vertex by different edges. In the remaining case $q(u) = q(v)$ and the two paths share an initial segment; if their last common vertex is at distance $h_c \ge 0$ from P then $d(u, v) = h(u) + h(v) - 2h_c = N - f(u) - e(v) - 2h_c$. In every case

$$d(u, v) = N - f(u) - e(v) - 2h_c \quad \text{with } h_c \ge 0,$$

so $d(u, v) \le N - 2$. Hence no pair avoiding a and b realises $N - 1$, and the unique pair realising $N - 1$ contains a or b; it is not $\{a, b\}$, so after renaming it is $\{a, w\}$ with $w \notin \{a, b\}$.

The tree $T - b$ contains both a and w, so $N - 1 \in \operatorname{Spec}(T - b)$, while $N \notin \operatorname{Spec}(T - b)$; thus $\operatorname{diam}(T - b) = N - 1$. The tree $T - a$ contains neither of the unique pairs realising N and $N - 1$, so $\operatorname{diam}(T - a) \le N - 2$.

For the last statement, note that with this naming the unique pair realising $N-1$ is $\{a,w\}$, which does not contain b; hence $d(b,u) \le N-2$, that is $f(u) \ge 2$, for every $u \notin \{a,b\}$. The displayed identity then gives $d(u,v) \le N - f(u) - e(v) \le N-3$ for every pair avoiding a and b. A pair of $T-a$ realising $N-2$ therefore cannot avoid b. □

The last statement of Theorem 8.5 holds for every value of the deleted diameter, not only for $N-2$, and the same identity locates the other end of the diametral pair of $T-a$ and bounds the pendant weight at a.

Corollary 8.6 (the second end of $T-a$)**.** *Keep the naming of Theorem 8.5, let $\delta = \operatorname{diam}(T-a)$, $m = N-\delta$, and let W be the set of the m vertices u with $d(a,u) > \delta$, so that $u \mapsto d(a,u)$ is a bijection from W onto $\{\delta+1,\dots,N\}$.*

(a) The unique pair of $T-a$ at distance δ contains b. Equivalently $\delta = N - \min\{f(u) : u \notin \{a,b\}\}$: there is a unique vertex y with $f(y) = m$, it is the only vertex with $d(b,y) = \delta$, and $d(a,y) = m + 2h(y)$.

(b) The neighbour a_1 of a satisfies $w(aa_1) = f(a_1) \ge m$, with equality if and only if $y = a_1$.

(c) Let $m \ge 2$ and let x_0 be the last vertex common to the paths from a to the members of W. Every vertex $x \notin W$ on the a–x_0 path, at distance p from a, has $\{d(x,u) : u \in W\} = \{\delta-p+1,\dots,\delta-p+m\}$, a run of m consecutive distances; hence two such vertices are at distance at least m apart, every edge of the a–x_0 path has weight at least m, and $d(a,x_0) \le (2N-1-\binom{m}{2})/2$, improving to $(2N-1-\binom{m+1}{2})/2$ when $x_0 \notin W$.

Proof. (a) Let $x,y \notin \{a,b\}$ be labelled so that $p(x) \le p(y)$. The pair $\{b,x\}$ avoids a, so $d(b,x) \le \delta$ and $f(x) \ge N-\delta = m$; also $e(y) \ge 1$. The identity in the proof of Theorem 8.5 gives $d(x,y) \le N - f(x) - e(y) \le \delta - 1$. So no pair avoiding both a and b realises δ; the pair realising it avoids a by definition, hence contains b. Writing it as $\{b,y\}$, $f(y) = N-\delta = m$, and no f is smaller because $d(b,u) \le \delta$ for all $u \ne a$; f is injective since the $d(b,u)$ are distinct. Finally $d(a,y) = N + 2h(y) - d(b,y)$.

(b) The leaf a has a unique neighbour a_1, which lies on the a–b path at position $p(a_1) = w(aa_1)$ and height 0, so $f(a_1) = w(aa_1)$, and $m = \min f \le f(a_1)$, with equality exactly when a_1 is the vertex y.

(c) A vertex x on the a–x_0 path lies on the path from a to every $u \in W$, so $d(x,u) = d(a,u) - p$, and the values $d(a,u)$ run through $\delta+1,\dots,N$. If $x \ne x'$ are two such vertices outside W, the pairs $\{x,u\}$ and $\{x',u'\}$ with $u,u' \in W$ are all distinct, so the two runs are disjoint sets of distances, which forces $|p-p'| \ge m$.

We claim that no vertex of W other than possibly x_0 lies on the path. Suppose $v \in W$ lies on it. Then v lies on the path from a to every $u \in W$, so $d(a,v) < d(a,u)$ for $u \ne v$, which forces $d(a,v) = \delta+1$ and $\{d(v,u) : u \in W \setminus \{v\}\} = \{1,\dots,m-1\}$. If $m = 2$ then $W = \{v,b\}$ and the path from a to b passes through v, so $x_0 = v$. If $m \ge 3$, let u_1 be the member with $d(v,u_1) = 1$; if $v \ne x_0$ then x_0 lies on the v–u_1 path beyond v, so $x_0 = u_1$ and $d(v,x_0) = 1$, and every other member u' has $d(x_0,u') = d(v,u') - 1 \in \{1,\dots,m-2\}$ with $d(x_0,u') \ne 1 = d(v,x_0)$: that is $m-2$ distinct values in the $m-3$ slots $\{2,\dots,m-2\}$, impossible. Hence $v = x_0$.

The edge bound now follows for every edge of the path whose ends are both outside W. If $x_0 \in W$, the last edge $x'x_0$ needs one more word: then $\{d(x_0,u) : u \in W \setminus \{x_0\}\} = \{1,\dots,m-1\}$ as above, so a weight $w(x'x_0) < m$ would equal some $d(x_0,u')$ and the distinct pairs $\{x',x_0\}$ and $\{x_0,u'\}$ would be at the same distance. For the depth bound, the $\binom{m}{2}$ distances inside W (and the further m distances from x_0 to W when $x_0 \notin W$) are distinct positive integers, each at most $N+(N-1)-2d(a,x_0)$, since two members of W have $d(u,v) = d(a,u)+d(a,v)-2d(a,z)$ with z their last common ancestor at depth at least $d(a,x_0)$. □

In the normal form of Section 8.4 part (b) reads $s \geq N - \delta$, and part (c) says that the low vertices between the root and the last common ancestor of the high block are spaced at least $N - \delta$ apart; neither was imposed in the searches reported below, which enumerate the full range of s.

Remark 8.7 (moment identities). Taylor's parity argument (Lemma 2.2) is the value at $z = -1$ of the polynomial identity $\sum_{u<v} z^{d(u,v)} = z + z^2 + \cdots + z^N$, which a Leech tree satisfies exactly. Its first derivatives are linear in the edge weights once residues are fixed. At $z = 1$ one gets the cut form of the Wiener index, $\sum_e w_e \, |A_e| \, (n - |A_e|) = \binom{N+1}{2}$, where A_e is one side of the edge e. At $z = -1$, with $\sigma_v = (-1)^{d(o,v)}$ for a root o, $S = \sum_v \sigma_v$ and $q_e = \sum_{v \in A_e} \sigma_v$, one gets

$$\sum_{u<v} (-1)^{d(u,v)} \, d(u,v) = \sum_e w_e \, q_e \, (S - q_e) = \begin{cases} N/2, & N \text{ even}, \\ -(N+1)/2, & N \text{ odd}, \end{cases}$$

because $(-1)^{d(u,v)} = \sigma_u \sigma_v$ and a path crossing e contributes w_e once. Once the parities of the depths are fixed this is a second integer linear equation in the depths, valid for every δ. We have not found it in the literature on Leech trees; it was suggested, in this form, by an OpenAI model consulted during the preparation of this paper (2026-09-04), and we verified it on the five known Leech trees and on random weighted trees. Wired into the depth phase of the class search together with the Wiener equation, however, it changed the node counts of the $\delta = 284$ records by less than one part in a hundred: with seven or more free depths the two-equation system is almost always solvable, and the exact test at complete assignments is subsumed by the distance check. It bites only when few depths are free.

Theorem 8.5 says that all the information in Lemma 8.2 is concentrated in one leaf. Deleting any leaf other than a or b leaves the diameter at N; deleting b leaves it at exactly $N - 1$, which is the case $m = 1$ of (5) and carries no information; deleting a is the only case in which the top block can be long, and it is exactly there that the next theorem bites.

8.3 The crowding inequality

Theorem 8.8 (top-block crowding)**.** *Let T be a Leech tree of order n, let ℓ be a leaf, and adopt the notation of Lemmas 8.1 and 8.2, with $r \geq 1$. Then*

$$\binom{m}{2} \;\leq\; (r+1) \, \max(0,\, 2m - 3). \tag{8}$$

The bound is vacuous for $m \leq 1$ (both sides are 0) and is used only for $m \geq 2$, where it reads $\binom{m}{2} \leq (r+1)(2m-3)$ and is equivalent, with $k = n - 1$ and $r = k - m$, to

$$5m^2 - (4k + 11)\, m + 6k + 6 \;\leq\; 0. \tag{9}$$

Proof. If $m \leq 1$ there is no pair of distinct high vertices and both sides of (8) vanish, so assume $m \geq 2$. Let $h_y, h_{y'}$ be distinct high vertices, at depths $B + y$ and $B + y'$ with $0 \leq y \neq y' \leq m - 1$, and let z be their nearest common ancestor in K rooted at x. Then $d(h_y, h_{y'}) = (B + y) + (B + y') - 2\, d(x, z)$, and z is either high or low. If z is high, say at depth $B + q$, then

$$d(h_y, h_{y'}) = y + y' - 2q; \tag{10}$$

if z is low, at depth $l \in L$, then

$$d(h_y, h_{y'}) = 2B - 2l + y + y'. \tag{11}$$

Since $y \neq y'$ and $0 \leq y, y' \leq m - 1$, the sum $y + y'$ takes values in $[1, 2m - 3]$, an interval of $2m - 3$ integers.

Split the $\binom{m}{2}$ high-high pairs into $r+1$ classes: one class for the pairs whose nearest common ancestor is high, and one class for each low vertex u, consisting of the pairs whose nearest common ancestor is u. Every high-high pair lies in exactly one class. Inside the class of a fixed low vertex u of depth l, the distance (11) is a strictly increasing function of $y+y'$, hence takes at most $2m-3$ values. Inside the high class, the distance (10) is positive and satisfies $y+y'-2q \le y+y' \le 2m-3$, hence lies in the interval $[1, 2m-3]$ and again takes at most $2m-3$ values. (We do *not* claim that q is determined by $y+y'$; only that the distance itself is confined to that interval.) All $\binom{m}{2}$ distances are pairwise distinct because T is a Leech tree, so each class contains at most $2m-3$ pairs, which is (8). Substituting $r = k-m$ and clearing denominators turns it into (9). □

Inequality (9) bounds m by the larger root of the quadratic,

$$m \;\le\; \frac{4k+11+\sqrt{(4k+11)^2-120(k+1)}}{10} \;<\; \frac{4k+11}{5},$$

so $r = k-m > \frac{1}{5}(k-11)$. In words: whenever $T-\ell$ is not itself a Leech tree, a leaf deletion leaves more than $\frac{1}{5}(n-12)$ of the k rooted depths strictly below the top block, that is,

$$\operatorname{diam}(T-\ell) \;>\; \binom{n-1}{2} + \tfrac{1}{5}(n-12).$$

(The two displayed bounds also hold trivially when $m \le 1$, where Theorem 8.8 says nothing.) At $n=25$ the quadratic is $5m^2-107m+150 \le 0$, whose roots are $(107 \pm \sqrt{8449})/10 = 1.508\ldots$ and $19.892\ldots$, so $m \le 19$. The individual values are in Table 7.

Table 7: The crowding count (8) at $n=25$, $N=300$, $k=24$. Here $\delta = \operatorname{diam}(T-\ell)$, $r = \delta - 276$ is the number of low depths and $m = 300-\delta = 24-r$ the length of the top block. The value $\delta = 276$ is excluded separately (it would make $T-\ell$ a Leech tree of order 24).

δ	277	278	279	280	281	282	283	284
r	1	2	3	4	5	6	7	8
m	23	22	21	20	19	18	17	16
pairs $\binom{m}{2}$	253	231	210	190	171	153	136	120
capacity $(r+1)(2m-3)$	86	123	156	185	210	231	248	261
verdict	**no**	**no**	**no**	**no**	—	—	—	—

Theorem 8.9. *Let T be a Leech tree of order 25. Then $\operatorname{diam}(T-\ell) \ge 281$ for every leaf ℓ of T. Moreover, if $\{a,b\}$ is the unique pair at distance $N=300$, then a and b are leaves and they can be named so that $\operatorname{diam}(T-b) = 299$ and $281 \le \operatorname{diam}(T-a) \le 298$, while $\operatorname{diam}(T-\ell) = 300$ for every other leaf ℓ.*

Proof. Let ℓ be a leaf and $\delta = \operatorname{diam}(T-\ell)$. Lemma 8.2 gives $r \ge 0$, that is $\delta \ge \binom{24}{2} = 276$, with equality only if $T-\ell$ is a Leech tree of order 24, which Lemma 2.2 forbids. So $r \ge 1$, and for $277 \le \delta \le 280$ we have $20 \le m \le 23$, so Theorem 8.8 applies and is violated by Table 7. Hence $\delta \ge 281$. The remaining statements are Theorem 8.5 together with the observation preceding it. □

Theorem 8.9 is unconditional: it uses no computation beyond the arithmetic of Table 7. The next subsection removes four more values of δ by finite search.

8.4 Finite certificates for $\delta = 281, \ldots, 284$

Fix $n = 25$ and a value of δ, hence $r = \delta - 276$ and $m = 24 - r$. By Lemmas 8.1 and 8.2, a leaf ℓ of a Leech tree of order 25 with $\operatorname{diam}(T - \ell) = \delta$ produces the following *normal form*: a rooted tree K on 24 vertices with root x, whose depth multiset is $D = L \sqcup [B, A]$ with $|L| = r$, $0 \in L$, $A = 300 - s$ and $B = \delta + 1 - s$, and whose $\binom{24}{2} = 276$ pairwise distances are distinct and fill $[1, \delta] \setminus (s + L)$. Conversely any such K, with a pendant edge of weight s attached at x, is a Leech tree of order 25. The parent of a high vertex h_y is a vertex of smaller depth, hence a low vertex or a high vertex h_p with $p < y$; a normal form is therefore determined by the rooted low tree on r vertices, the low depths L, the weight s, and a parent function

$$\mathrm{par}\colon \{h_0, \ldots, h_{m-1}\} \longrightarrow \{\text{low vertices}\} \cup \{h_p\}, \qquad \mathrm{par}(h_y) = h_p \Rightarrow p < y.$$

The programs described below enumerate exactly this normal form. They live in the directory `theory-lab/s1_crowding/` of the archive and share no code with the engines of Sections 3 and 6. The SHA-256 digest of every run record quoted below is in Table 10.

Arithmetic verifier. `verify_s1_top_block_crowding.py` (pure Python, no project imports) re-derives the parity table of Lemma 8.4 for $\delta = 277, \ldots, 281$, checks the distance formulas (10) and (11) against breadth-first search on random trees carrying the required depth structure (3000 trees for each of $r = 3$ chain, $r = 3$ star, $r = 4, 5, 6$ random low shapes), checks on those trees and on hill-climbed ones that the number of distinct high-high distances never exceeds $(r + 1)(2m - 3)$, and recomputes Table 7. It prints `VERIFIED_S1_TOP_BLOCK_CROWDING_ARITHMETIC` and runs in about a minute.

Direct search. `s1_direct_search.cpp` places the vertices in depth order; each high vertex chooses its parent among the low vertices and the shallower high vertices, and every new pair distance is tested incrementally against $[1, \delta] \setminus (s + L)$ and against the distances already used. Positive controls: the order-6 Leech tree of Example 8.3 is recovered ($r = 1$, one survivor), and planted random distance-injective trees of orders $9, 10, 12$ with $r = 2, 3, 4$ are recovered in every trial. As a redundant mechanisation of the $\delta = 279$ case of Theorem 8.9 the whole $\delta = 279$ normal form was swept for $s \in [13, 160]$ (all 148 values), both rooted low trees on three vertices, and every $0 < d_1 < d_2 < B$ of either parity: 296 instance summaries, 5,725,972 instances, 571,781,366 nodes, zero survivors, zero aborts, four minutes of wall time on four cores (rows 1 and 2 of Table 10). The range of s is the one available when the deleted leaf hangs on the heaviest edge of T ($300 \le 24s$ gives $s \ge 13$, and $\delta \le 2A$ gives $s \le 160$); the restriction is immaterial because Theorem 8.9 covers all s. Per instance the search dies after at most ten placed high vertices, which is the crowding theorem visible in the search tree.

Depth-free class search. `abstract_class_search.cpp` drops s and L entirely. For a given m and a given rooted low tree it enumerates every parent function par and rejects a partial structure only when it violates one of the class conditions of the proof of Theorem 8.8: the within-component distances $y + y' - 2q$ of (10) must be pairwise distinct, and for each low vertex u the sums $y + y'$ over the pairs with nearest common ancestor u must be pairwise distinct. Neither condition involves L or s, and both are necessary for any tree of the normal form, so if the search finds no complete structure then no such tree exists for any s and any low depths. Only the nearest-common-ancestor structure of the low tree enters, so the low trees are taken up to rooted isomorphism (AHU canonical form) from $r = 5$ on; at $r = 3, 4$ all $(r - 1)!$ labelled parent vectors were enumerated, which is a superset. The semantics of the enumeration were cross-checked against `brute_abstract_check.py`, written by a separate agent from the written specification alone and without access to the C++ source, which rebuilds each tree explicitly and finds every nearest common ancestor by walking ancestor

lists: on eight cases ($m = 4, \dots, 7$, $r = 1, \dots, 4$, up to 604,800 candidate structures) the counts agree exactly and the emitted structure sets are identical after sorting. The counts are 9/24, 7/120, 99/720, 117/5040, 1207/20160 (chain) and 504/20160 (star), 1593/181440, 3476/604800.

Depth phase. When the depth-free search does produce complete structures, each of them is handed to a constraint search over the remaining unknowns s and $L = \{0 = l_0 < l_1 < \cdots < l_{r-1}\}$, which requires all 276 pair values to be distinct and to lie in $[1, \delta] \setminus (s + L)$; a survivor would be a genuine tree of the normal form, hence a Leech tree of order 25. One labelled representative per rooted-isomorphism class of low trees suffices here, because the constraint search assumes only that each low vertex is strictly deeper than its parent and that all depths are distinct, never that $l_i < l_j$ for $i < j$; automorphisms of the low tree can duplicate survivors but cannot lose any. The depth phase recovers the order-6 Leech tree as a positive control ($r = 1$, $s = 8$), and was checked differentially against `s1_direct_search` on a relaxed target (order 9, $m = 5$, $r = 3$, all values in $[1, 60]$ allowed and no holes, $s \in \{10, 12, 14, 16\}$, every increasing L): the survivor counts agree exactly for the chain low tree (1385, 501, 190, 34) and are exactly twice the direct-search counts for the star (5182, 2232, 714, 126 against 2591, 1116, 357, 63), the factor two being the automorphism of the star that the depth-order-free search counts twice.

The outcome is Table 8.

Table 8: Depth-free class search at order 25. "Low trees" is the number of rooted low trees searched (all $(r-1)!$ labelled parent vectors at $r = 3, 4$; rooted-isomorphism classes from $r = 5$ on), "with structures" the number of them admitting a complete structure. The rows $\delta = 279, 280$ are a second mechanisation of Theorem 8.9; the rows $\delta = 281, \dots, 284$ are the certificates of Theorem 8.10, the "with structures" count there being followed by an exhaustive depth phase; the rows $\delta \geq 285$ are the depth-free phase only and prove nothing on their own.

δ	r	m	low trees	with structures	remark
279	3	21	2	0	< 0.1 s
280	4	20	6	0	< 0.1 s
281	5	19	9	0	0.3 s; 1,697,398 nodes on the chain
282	6	18	20	0	9 s; 37,868,410 nodes on the chain
283	7	17	48	1	chain only, 14,730 structures; depth phase empty
284	8	16	115	7	1.64×10^9 structures; depth phase empty
285	9	15	286	95	depth phase running
286	10	14	719	317	depth phase not started
287	11	13	1842	1523	depth phase not started

Theorem 8.10 (certificate form)**.** *The depth-free class search is empty for* $\delta = 281$ *(*9 *rooted low trees) and for* $\delta = 282$ *(*20 *rooted low trees). For* $\delta = 283$ *it is empty for* 47 *of the* 48 *rooted low trees, the exception being the chain, which admits exactly* 14,730 *complete structures; the depth phase on those* 14,730 *structures has no survivor. For* $\delta = 284$ *it is empty for* 108 *of the* 115 *rooted low trees; the remaining seven admit* 1,641,497,082 *complete structures in total, and the depth phase on all of them has no survivor. Consequently, for a Leech tree* T *of order* 25 *and any leaf* ℓ*,*

$$\operatorname{diam}(T - \ell) \neq 281, 282, 283, 284.$$

Combined with Theorem 8.9, which rules out every $\delta \leq 280$ *without computation, this gives* $\operatorname{diam}(T - \ell) \geq 285$ *for every leaf, and in the notation of that theorem* $285 \leq \operatorname{diam}(T - a) \leq 298$*.*

The computations behind Theorem 8.10 are as follows. The $\delta = 281$ and $\delta = 282$ searches ran in a single local process in 0.3 and 9 seconds and were repeated on a workstation from the same source. For $\delta = 283$ the depth-free phase ran in four shards over the 48 rooted low trees with the first-structure switch enabled and no shard aborting: 47 shapes exhausted with zero structures, and the chain, whose first structure appears after 11,571 nodes (row 3 of Table 10). The depth phase on the chain was run three times:

- on the workstation as a single process: 14,730 leaves, 932,540,226 nodes, 16,302,184 constraint-search nodes, zero survivors (row 4);
- on the Hoffman2 cluster as 64 prefix-split array tasks, all reporting `COMPLETE`: 14,730 leaves, 934,428,210 nodes (the difference from the single-process count is exactly the prefix that every split task re-traverses), the same 16,302,184 constraint-search nodes, zero survivors (row 5). The collector `verify_depth_splits.py` checks that there is exactly one final file per split with no unfinished temporary file, that every file ends with a `COMPLETE` summary containing one exhausted record per shape at the expected m and split index, and that the total survivor count is zero; it reports `VERIFIED_DEPTH_SPLITS_EXHAUSTED_EMPTY` and writes a JSON certificate with the SHA-256 of every split file, itself recorded as row 6;
- locally with a memoised variant of the constraint search that first solves for (s, L) at the level of the map sending each high vertex to the low vertex carrying its component (108 CPU-s): again 14,730 leaves and zero survivors, with only 2,045 distinct such maps among the 14,730 structures, every one of them already infeasible.

For $\delta = 284$ the depth-free phase over the 115 rooted low trees left seven of them with complete structures (row 9 of Table 10); the other 108 are exhausted with none. The depth phase on those seven ran on the cluster as 64 prefix-split array tasks of the memoised implementation, split at level 5, one labelled representative per rooted-isomorphism class, taking between 2 h 02 and 3 h 52 per task. All 64 tasks completed and the collector `verify_depth_splits.py` reports `VERIFIED_DEPTH_SPLITS_EXHAUSTED_EMPTY` with an empty error list; the merged split outputs and the JSON certificate are rows 10 and 11 of Table 10. The per-shape totals are in Table 9: 1,641,497,082 complete structures, 3.93×10^{10} abstract search nodes, 49,212,675 distinct fibre assignments actually solved, and no survivor anywhere. The chain carries 98.6% of the structures on its own.

Table 9: The $\delta = 284$ depth phase ($m = 16$, $r = 8$), per rooted low tree. A shape is given by its parent vector $(p_1, \ldots, p_7)$, the parent of low vertex i being $p_i < i$; the last row is the chain. "Structures" counts the complete depth-free structures handed to the depth phase, "ρ" the distinct fibre assignments for which the constraint search was actually run (the rest are memoisation hits), and "survivors" the trees of the normal form found. Summed over the 64 prefix-split tasks.

low tree	structures	abstract nodes	distinct ρ	CSP nodes	survivors
$0,0,1,3,4,5,6$	3,018,576	1,905,844,936	219,758	6,060,630,718	0
$0,1,1,2,4,5,6$	3,094,884	1,925,488,408	226,254	777,654,870	0
$0,1,2,2,3,5,6$	3,180,114	1,953,706,332	233,464	742,211,809	0
$0,1,2,3,3,4,6$	3,281,182	2,007,391,240	242,116	847,594,294	0
$0,1,2,3,4,4,5$	3,428,058	2,229,724,050	256,082	899,376,730	0
$0,1,2,3,4,5,5$	7,329,832	4,285,409,714	567,932	2,006,443,404	0
$0,1,2,3,4,5,6$	1,618,164,436	25,005,269,686	47,467,069	244,492,129,704	0
total	1,641,497,082	39,312,834,366	49,212,675	255,826,041,529	**0**

Why a sharper class bound will not help. The exact per-class maxima were computed, over all high forests, by the auxiliary program `hh_capacity.cpp`: the largest possible number of within-component pairs with distinct distances, and the largest possible number of cross-component pairs with distinct sums. Both equal or nearly equal the crude bound $2m-3$ for $m \le 14$, the value $2m-3$ being attained by the component pattern $\{0\}, \{m-1\}, \{1, \dots, m-2\}$. So (8) cannot be improved by sharpening the capacity of a single class; from $\delta = 281$ on one needs the joint search of Theorem 8.10, in which the constraints coming from different classes interact.

Status of the certificates. Theorem 8.10 is a finite computation, not a hand proof, and we label it accordingly: *computer-verified* rather than *proved*. Its mathematical content, that the enumeration is exhaustive, is the paragraph above Table 8; its empirical content is that the programs execute that enumeration. The supporting evidence is the independent brute-force cross-check of the class semantics, the positive controls, the differential test of the depth phase, and, for $\delta = 283$, three runs on two machines with two sharding layouts agreeing on the leaf and constraint-node counts and not only on the zero.

The evidence is thinner at $\delta = 284$, and we state exactly how. For the six non-chain shapes the per-split structure counts of the memoised implementation coincide with those of an independent per-leaf implementation, which was run on the same splits and then cancelled before it reached the chain; so the six non-chain shapes are covered by two implementations, but the chain, which carries 98.6% of the structures, rests on a single run of the memoised program. The two implementations share the depth-free enumeration and differ in the constraint phase, the memoised one solving the constraint search once per distinct fibre assignment instead of once per structure; the differential test described above exercises exactly that difference. A second full pass over the chain, ideally with the unmemoised implementation, is the natural next check.

Provenance is now recorded as follows. The final revision of each source is frozen in the subdirectory `frozen/` of `theory-lab/s1_crowding/`, under a name that carries that file's own SHA-256 prefix: for example `abstract_class_search_d83323870704.cpp` and `s1_direct_search_44994ae7484c.cpp`. The runs reported above were made with earlier revisions of those sources, whose hashes are recorded run by run in Table 10 and in the archive; the revisions differ in the constraint phase and in sharding and output plumbing, and the enumeration logic that the exhaustiveness argument depends on is unchanged between them.

All four searches have since been repeated from the frozen source itself. Running `abstract_class_search_d83323870704.cpp` on a single workstation reproduces $\delta = 281$ (nine rooted classes, no structure), $\delta = 282$ (20 classes, no structure) and $\delta = 283$ (48 classes, only the chain with structures, and on the chain 14,730 complete structures, 932,540,226 abstract nodes, 2,045 distinct fibre assignments and no survivor), that is, exactly the numbers reported above, including both the node count of the original single-process run and the count of distinct fibre assignments of the memoised one; the results file is row 8 of Table 10.

The $\delta = 284$ search has now been repeated from the same frozen source as a 64-task cluster array, and we state the comparison at the finest granularity the outputs allow. The re-run emits one `ABSTRACT` record per (split, shape) pair, 448 records in all. Every one of the 448 keys is present in both runs, and every record agrees exactly in its leaf count, its node count and its survivor count: 1,641,497,082 leaves and 39,312,834,366 abstract nodes in total, every record `EXHAUSTED`, no survivor anywhere. Per shape the leaf counts are 3,018,576, 3,094,884, 3,180,114, 3,281,182, 3,428,058, 7,329,832 and 1,618,164,436, the last being the chain, which carries 98.6% of the enumeration. The re-run's own split certificate is row 15 of Table 10, and it carries the same acceptance token `VERIFIED_DEPTH_SPLITS_EXHAUSTED_EMPTY` with an empty error list.

Theorem 8.10 is therefore, for every one of $\delta = 281, 282, 283, 284$, a finite computation

reproduced from a single archived source whose hash is printed in Table 10. Theorem 8.9 needs none of this.

8.5 The reduction, and where the computation stands

Corollary 8.11 (reduction of order 25)**.** *There is no Leech tree of order* 25 *if and only if the normal form of Section 8.4 admits no tree for any* δ *with* $285 \le \delta \le 298$*, any* s*, and any low structure.*

Proof. If T is a Leech tree of order 25, let a be the diametral endpoint given by Theorem 8.5, so $\operatorname{diam}(T-a) \le 298$, and $\operatorname{diam}(T-a) \ge 285$ by Theorem 8.10; deleting a produces a tree of the normal form with $\delta = \operatorname{diam}(T-a) \in [285, 298]$. Conversely, let K be a tree of the normal form for some such δ and restore the pendant edge of weight s at x. Its distances are those of K, namely $[1, \delta] \setminus (s + L)$, together with $s + D = (s + L) \sqcup \{\delta + 1, \dots, 300\}$; the union is $[1, 300]$ and it is disjoint, so the result is a Leech tree of order 25. □

The order-25 problem is therefore reduced to the fourteen values $\delta = 285, \dots, 298$ of one explicit finite normal form. We stress that the reduction does not use the "heaviest edge" hypothesis under which the identity (4) was first recorded in this project: any leaf satisfies it, and Theorem 8.5 selects the leaf that matters without reference to edge weights. Without the certificates of Theorem 8.10 the range is $\delta \in [281, 298]$, which is still finite; with them it is $[285, 298]$.

The first of the fourteen cases was begun and then set aside, and we record where it stands. At $\delta = 285$ the depth-free phase leaves 95 of the 286 rooted low trees with complete structures (row 12 of Table 10), and the depth phase on those 95 shapes was run on the cluster as a prefix-split array of 256 tasks. Every task processes the 95 shapes in the same order, so a task stopped by the wall clock still contributes complete evidence for the shapes it finished; the acceptance criterion for the whole value of δ is, as at $\delta = 284$, `VERIFIED_DEPTH_SPLITS_EXHAUSTED_EMPTY` with an empty error list from `verify_depth_splits.py` over all 256 splits. That criterion was not reached. The array was stopped after 3,196 of the 24,320 (split, shape) pairs had completed, every one of them `EXHAUSTED` with zero survivors; those partial outputs are archived together with a per-file hash manifest, so the computation can be resumed rather than restarted. Completing $\delta = 285$ was costed at about 10^4 CPU-hours in several wall-clock-limited waves. We make *no* claim for $\delta = 285$ here, and none is made anywhere in this paper. The certified range of Theorem 8.10 therefore ends at $\delta = 284$, and the open range left by Corollary 8.11 for order 25 stands at $\delta \in [285, 298]$. If a survivor ever does appear, at $\delta = 285$ or above, it is a candidate Leech tree of order 25 and must be checked by the independent witness checkers before anything is claimed from it.

The method also has a visible horizon. The depth-free phase has been run further, for information only: at $\delta = 286$, 317 of 719 rooted low trees admit complete structures, and at $\delta = 287$, 1523 of 1842 do (rows 13 and 14). The class-injectivity conditions of Theorem 8.8 therefore stop being restrictive around $r = 10$, as the widening gap between the last two rows of Table 7 already suggests, and the remaining cases $\delta \ge 286$ will need constraints that use the low depths from the start rather than as a second phase.

What this section does and does not prove. Theorem 8.9 is unconditional. Theorem 8.10 is a finite computer verification with the audit trail described above. Neither excludes a Leech tree of order 25, and neither says anything about deletions other than of a single leaf: the whole argument rests on the identity (4), which is special to a pendant edge, and on the top block (5), which is special to the diametral endpoint. In particular nothing here bounds the structure of T away from the leaf a, and nothing here is a statement about branches, about the heaviest edge, or about any of the other reductions attempted in the course of this project.

Table 10: SHA-256 digests of the sources and run records quoted in Sections 8.4 and 8.5. "Merged output" is the concatenation, in shard order, of the per-shard output files. Row 7 is the source revision recorded with the $\delta = 281, 282, 283$ runs and row 8 the re-run of those three values from the frozen source; the frozen final sources carry their own hashes in their file names. See the discussion of provenance in Section 8.4.

#	object and digest
1	$\delta = 279$ direct sweep, merged summaries
	`d130f57d25aefc9b5097c6eb57e27731c06496e8ae51f7621ea911dc8318e3ed`
2	`s1_direct_search.cpp` as run
	`433e9a8884a04a5bb068c063df1d6c246d3e74f919ecf911512836906262c3d7`
3	$\delta = 283$ depth-free phase, merged output
	`5e9b40afaa05bd9551d27f473314abb05f0e6faa10b38255a07c4f27ff350b09`
4	$\delta = 283$ depth phase, single-process run, output
	`fc360372265552ae0dc5db057cdda80d506d0009b2927541676e42f847407cec`
5	$\delta = 283$ depth phase, 64-task cluster run, merged output
	`b1bce6b547ac84acf94aaa63a901f51472ea06569e82f32d8dac54e62de18c97`
6	$\delta = 283$ split certificate `certificate.json`
	`32603c95154113d2354fd6f53a7d0026ecb86b5cc5ff0c4e54e33eb7316fb3fe`
7	`abstract_class_search.cpp`, revision recorded with the runs
	`3a7a58beed9e02a9c97aae9db84e5425989871f7dc351d40921b0a0f17ebcae3`
8	$\delta = 281, 282, 283$ re-run from the frozen source, results file
	`4d9ec4985f45d24fce96b54bec3c580535dd657b8341118d2f9af9195fd88511`
9	$\delta = 284$ depth-free phase, merged output (108 empty classes)
	`18082a3a9062f712b36d58c229c1a1cb875c2a1ca07e0f04cd6146a1de46787b`
10	$\delta = 284$ depth phase, 64-task cluster run, merged split outputs
	`bccd6ab737e1a5aff30944bf82d3bd4506b9edc5c49721f6166e8a335ddd7b55`
11	$\delta = 284$ split certificate `certificate.json`
	`b5769434011afa56dcc3d124f0bacfb831052b7fe3e39a699d837589248ca6be`
12	$\delta = 285$ depth-free phase, merged output
	`ac29908f71d1d20f8688278daeeb7fc7149b4fee5d8f5a7ec06da1a0df6019f1`
13	$\delta = 286$ depth-free phase, merged output
	`2f91a166a5ca9adfbbac0445782358c0ec7b9149e49e3d30aacef40ffac21723`
14	$\delta = 287$ depth-free phase, merged output
	`048883d27554d51113576fe2a7dfbf27c946f4f4c7461f4e7e886c28c4a83aa3`
15	$\delta = 284$ re-run from the frozen source, split certificate
	`8f8ee5475b31ed852d55b59c41862d893c1478dc6568b6ca3f0ba7f4dadb0175`
16	`abstract_class_search_d83323870704.cpp`, the frozen source
	`d83323870704cd5060abff3d0a5d879b43dc22a6131a25f11a27f4327a112a9d`

9 Discussion

Status of the result. Theorem 1.1 is a computer-assisted theorem in the usual sense: the mathematics (Lemmas 2.1, 3.1, 3.2 and Proposition 3.3) is short and has been checked by hand and by exhaustive small cases; the remainder is a finite computation whose correctness depends on the faithful execution of a program. What distinguishes this computation from a bare "no solution found" is that its intermediate output—the eighteen per-level counts of Table 4—is a mathematically defined sequence (the number of non-isomorphic forced forests of each size), so that any independent implementation can be checked against it level by level. The shallow levels have been confirmed by an oracle written from the definitions; all eighteen levels have been reproduced by an independent implementation written from the algorithm description; and three replications of the reference engine on different hardware and compilers agree bit-for-bit. The one verification we would like to have and do not yet have in full is the independent *method*: the per-topology engine, which shares no code and no search order with the forest engine, has decided 112,082 of the 122,344 surviving 18-vertex topologies, that is 91.6%, all UNSAT and none SAT, and was stopped on the remainder (Section 4.2). It has,

however, already reproduced the complete order-16 result (19,320 topologies, all UNSAT) by this independent route, which is the strongest available calibration of the two engines against each other and against [3].

Theorem 1.2 has a different trust boundary. Lemma 6.1, the regular-window reduction and the affine-lifting induction are the mathematical argument; the finite closures are recomputed by the repository verifier. The affine clean-room program checks the implementation of that finite search but shares its abstraction, so we do not count it as a second proof. The independent adversarial review described at the end of Section 6 found one non-mathematical defect—the original verifier used Python `assert` statements, which disappear under `-O`—and no mathematical defect. All certificate checks now use explicit exceptions and optimized execution is rejected.

Theorem 1.3 has the widest gap of the three between the proved and the computed part, and Section 8 keeps them apart on purpose. Parts (a) and (b) are hand proofs of a few lines each, verified independently by a Python script that also checks the two distance formulas against breadth-first search on random trees. The improvement of the lower bound from 281 to 285, that is the exclusion of the four values through 284, is a finite computer verification: its exhaustiveness argument is mathematical and short, its class semantics were reproduced by a brute-force enumerator written from the specification by an agent with no access to the production source, and the $\delta = 283$ search was run three times on two machines with agreeing intermediate counts. All four values, $\delta = 284$ included, have since been reproduced from the frozen hash-tagged source: the $\delta = 284$ re-run agreed with the original in every one of its 448 (split, shape) records, leaf count, node count and survivor count alike (Section 8.4, rows 15–16 of Table 10). The residual weakness is therefore not a missing replication but a shared abstraction: both depth-phase runs sit on the same depth-free enumeration, and the independent per-leaf implementation that would have tested that abstraction was cancelled before it reached the chain shape. We label the two levels separately in the statements themselves.

Certificates. We do not have a proof certificate for $n = 18$ that could be checked by a formally verified checker, as we do for $n \leq 11$ (DRAT). Two routes are open. (i) Cube-and-conquer: use the forcing-lemma tree to a fixed depth as cubes and certify each cube with a CDCL solver producing DRAT proofs; the completeness of the cube set follows from Lemma 2.1 by a one-paragraph argument, so the search engine's pruning need not be trusted, only its branching order. Our experiments at $n = 16$ suggest a cost of the order of one CPU-hour per topology and proofs of 10^2 MB per cube, i.e. 10^4–10^5 CPU-hours for all of $n = 18$—feasible on a cluster but not done. (ii) Proof logging inside the forest engine (VeriPB/pseudo-Boolean), which would certify the isomorph rejection as well; not attempted. We regard the present level of verification (independent oracle for the shallow levels, differential tests on planted instances, three bit-identical replications, and an independent implementation reproducing all per-level counts) as adequate for the claim, and the per-level counts as the reproducible artefact that a sceptical reader should target.

What remains. The next admissible orders are $n = 25$ ($N = 300$) and $n = 27$ ($N = 351$). Extrapolating the measured growth factors (Table 3 and the completed $n = 19$, $n = 20$ runs below: 7.93 and 8.19 per unit of n, still rising) gives 1.4×10^{17}–2.2×10^{17} nodes at $n = 25$, which is beyond the present method by four to five orders of magnitude; new pruning ideas (for instance exploiting the top-value structure and the parity classes inside the forest search, or a meet-in-the-middle over the smallest and largest weights) or a genuinely different argument would be needed. Whether Leech's five trees are the only Leech trees remains open. Theorem 1.1 moves the smallest open order from 18 to 25, while Theorem 1.2 excludes an infinite family at every order. Section 8 suggests one way in which order 25 might still be reachable without a 10^{16}-node search: Corollary 8.11 replaces it by fourteen instances of a

normal form with only 24 vertices, of which the first was begun and set aside at about 10^4 CPU-hours and the rest grow rapidly in cost. What makes those instances hard is that the class conditions of Theorem 8.8 stop being restrictive once the top block is short, so a search that keeps the low depths out of the loop cannot finish the job; the natural next step is a search in which the low depths and the block structure are decided together, or a bound on the low depths from the parity constraint (7). A separate pilot of a rooted-depth exact search for the remaining normal-form cases (`theory-lab/depth_search/`, validated against the known results for $n \le 11$) puts the cost of $\delta \ge 286$ under its present pruning at well over 10^{12} CPU-hours, while the increasing-weight forest engine of this paper, run to completion at $n = 19$ and $n = 20$ on the cluster to measure the trend (respectively 4.74×10^{11} and 3.88×10^{12} unique nodes, no survivors, 61 and 510 CPU-hours at 0.46–0.47 μs per node; growth factors 7.93 and 8.19 per vertex, continuing the rise of Table 3), extrapolates to 1.4×10^{17}–2.2×10^{17} nodes and 2×10^7–3×10^7 CPU-hours at order 25 according to whether the growth factor is held at 8.19 or continues to rise by 0.27 per vertex; we therefore do not expect the fourteen cases $\delta \in [285, 298]$ to fall to direct search, and regard new structural input as the thing that is missing. One candidate is the additive structure of the top of the spectrum: by the proof of Theorem 8.5 the deficits $e(u) = N - d(a, u)$ and $f(u) = N - d(b, u)$ of the two diametral endpoints satisfy $d(u, v) = N - f(u) - e(v) - 2h_c$ with $h_c \ge 0$, so the distances near N are governed by the sumset of the two deficit sets, in which distinctness of the distances forces distinct pairs to give distinct values. Independently of that, the natural next structural target is a tree with two branching vertices (a "bi-spider"): it is the first class not covered by the one-centre coordinate system, and would test whether the bounded top-window method survives one additional branch point. In the neighbouring problems of Section 7 the frontier is much closer: $M(13)$ is out of reach of the present engine, modular Leech trees of order 12 and above are untouched, and the leaf-Leech question at 9 leaves needs an engine of the kind built here for Leech trees rather than a generic solver.

10 AI-assisted research statement

Anthropic's Claude, OpenAI's ChatGPT and Google's Gemini were used as coding and analysis assistants throughout this project. They implemented and executed the processing, reasoning, and statistical code, prepared documentation, surveyed literature, and assisted in preparing manuscript text. The author reviewed and takes responsibility for all mathematical claims, code and text in this paper.

11 Data and code availability

All of the material below is deposited at https://github.com/LSMENG/leech-trees, tagged `v1.1-deposit-2026-09-06` and released under that tag; the tag is the citable identifier, and the commit it resolves to is printed on the release page. It names a single curated snapshot rather than the project's working history, because that history contains intermediate files above the hosting limit. The same snapshot is archived on Zenodo as doi:10.5281/zenodo.22334399; that record carries, as a separate file, the per-shard records of the $n = 19$ and $n = 20$ runs quoted in Section 9 (`forest19_20_shards.tar.gz`, MD5 `377e316106b89aa7086919bf9c4ad659`). The earlier tag `v1.0-deposit-2026-09-04` is left in place unchanged; it predates the $n = 19$ and $n = 20$ runs and the final revision of this paper. The archive deliberately omits one thing: the working artefacts of the top-window programme that Section 8 supersedes, about 2.3 GB of intermediate JSON that no statement in this paper cites. The archive contains: the two search engines (`forest_search.cpp`, `leech_search.cpp`) and the clean-room `cr_forest.c`; the independent Python enumerators and differential harnesses; the SAT

encoder, its driver, and the per-instance records of the DRAT-checked runs for $n \leq 11$, each carrying the instance parameters, the solver, the status and the size of the proof that was checked, together with the pinned Python environment and the one command that regenerates the proof files themselves (they run to gigabytes and are not deposited); the frozen list of the 123,867 trees of order 18 with the two-generator cross-check; the per-shard output records and depth histograms of the $n = 18$ runs and of the $n \leq 17$ runs, together with those of the completed $n = 19$ and $n = 20$ runs of Section 9 and the exact source and binary hashes of the engine that produced them; the star-Sidon table generators `star_sidon_table.c` and `star_sidon_table.py` of Lemma 2.5 with their output; the verdict scripts; and the referee reports on which Section 4 is based. The spider directory contains the exact, relaxed and affine searches, the no-spider verifier, its archived regression transcript, and the independent adversarial report. It also contains, in a separate directory, the modified engines and the raw outputs behind Section 7 (minimal distinct distance trees, modular Leech trees, leaf-Leech trees), including the witnesses printed above and the independent checker used on them. The order-25 material of Section 8 is in the directory `theory-lab/s1_crowding/`, which shares no code with any other engine in the project and contains the arithmetic verifier `verify_s1_top_block_crowding.py`; the direct normal-form search `s1_direct_search.cpp`; the depth-free class search and depth phase `abstract_class_search.cpp`; the per-class capacity computation `hh_capacity.cpp`; the independent brute-force cross-check `brute_abstract_check.py`; the split collector and certificate writer `verify_depth_splits.py`; and the run drivers `run_r3_full_sweep.sh`, `hoffman2_s1_abstract_array_slurm.sh` and `hoffman2_s1_depth_v7_array_slurm.sh`, together with the merged outputs and JSON certificates whose SHA-256 digests are quoted in Section 8.4. Its subdirectory `frozen/` holds the final revision of each source under a name carrying that file's own SHA-256 prefix. The variant of the depth phase carrying the two moment equations of Remark 8.7, and the scripts that check those identities against the five known Leech trees, are deposited beside them. The five known Leech trees are checked by two independent checkers, and every witness produced by any engine in the project was passed through both.

Acknowledgements

Computations were carried out on the author's laptop and on the Hoffman2 Shared Cluster of the UCLA Office of Advanced Research Computing.

References

[1] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[2] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, and F. Pollitt. CaDiCaL 2.0. In *Computer Aided Verification (CAV 2024)*, volume 14681 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2024. doi: 10.1007/978-3-031-65627-9_7.

[3] B. Calhoun, K. Ferland, L. Lister, and J. Polhill. Minimal distinct distance trees. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 61:33–57, 2007.

[4] distributed.net. OGR (optimal Golomb rulers) project. https://www.distributed.net/OGR, 2022. Distributed verification of optimal Golomb rulers with 24–28 marks (OGR-24 to OGR-28, 2004–2022). Accessed 2026-08-18.

[5] A. Dollas, W. T. Rankin, and D. 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.

[6] Epoch AI. FrontierMath open problems: Leech trees. https://epoch.ai/frontiermath/open-problems/leech-trees, 2025. Accessed 2026-08-18.

[7] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, pages 11–15, 2008.

[8] A. Lakshmanan S. and M. M. Eldho. On Leech labelings of graphs and some related concepts. *Discrete Mathematics*, 347(4):113837, 2024. doi: 10.1016/j.disc.2023.113837.

[9] D. Leach. Modular Leech trees of order at most 8. *International Journal of Combinatorics*, 2014:Article ID 218086, 2 pages, 2014. doi: 10.1155/2014/218086.

[10] D. Leach and M. Walsh. Generalized Leech trees. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 78:15–22, 2011. URL https://combinatorialpress.com/article/jcmcc/Volume%20078/vol-078-paper%202.pdf.

[11] J. Leech. Another tree labelling problem. *The American Mathematical Monthly*, 82(9):923–925, 1975. doi: 10.1080/00029890.1975.11993981.

[12] T. Luo and L. Yu. A graph labeling problem. *Involve, a Journal of Mathematics*, 17(2):327–335, 2024. doi: 10.2140/involve.2024.17.327.

[13] B. D. McKay and A. Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014. doi: 10.1016/j.jsc.2013.09.003.

[14] OEIS Foundation Inc. The on-line encyclopedia of integer sequences. https://oeis.org, 2026. Sequences A000055 (number of trees with n unlabeled nodes) and A003022 (length of optimal Golomb rulers). Accessed 2026-08-18.

[15] M. Ozen, H. Wang, and D. Yalman. Note on Leech-type questions of trees. *Integers*, 16:Paper No. A21, 2016.

[16] L. Perron and F. Didier. CP-SAT, OR-Tools v9. Google, https://developers.google.com/optimization/cp/cp_solver, 2024. Accessed 2026-08-18.

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

[18] L. A. Székely, H. Wang, and Y. Zhang. Some non-existence results on Leech trees. *Bulletin of the Institute of Combinatorics and its Applications*, 44:37–45, 2005.

[19] L. A. Székely, H. Wang, and Y. Zhang. Erratum to "some non-existence results on Leech trees". *Bulletin of the Institute of Combinatorics and its Applications*, 52:6, 2008.

[20] H. Taylor. Odd path sums in an edge-labeled tree. *Mathematics Magazine*, 50(5):258–259, 1977. doi: 10.1080/0025570X.1977.11976658.

[21] H. Taylor. A distinct distance set of 9 nodes in a tree of diameter 36. *Discrete Mathematics*, 93(2–3):167–168, 1991. doi: 10.1016/0012-365X(91)90252-W.

[22] S. Varghese, A. Lakshmanan S., and S. Arumugam. Two classes of non-Leech trees. *Electronic Journal of Graph Theory and Applications*, 8(1):205–210, 2020. doi: 10.5614/ejgta.2020.8.1.15.

[23] S. Varghese, A. Lakshmanan Savithri, and S. Arumugam. Leech labeling problem on tristar. *AIP Conference Proceedings*, 2649(1):020007, 2023. doi: 10.1063/5.0114834.

[24] N. Wetzler, M. J. H. Heule, and W. A. Hunt. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing – SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014. doi: 10.1007/978-3-319-09284-3_31.

[25] R. A. Wright, B. Richmond, A. Odlyzko, and B. D. McKay. Constant time generation of free trees. *SIAM Journal on Computing*, 15(2):540–548, 1986. doi: 10.1137/0215039.