SQUEEZEBENCH: SATURATION-RESILIENT EVALUATION OF MATHEMATICAL CAPABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

AI systems are making rapid progress in mathematics, but this progress is hard to measure. Benchmarks of solved problems saturate, and in benchmarks of open problems each problem can be resolved for the first time only once. We present SQUEEZEBENCH, a benchmark that is resilient to saturation: a success raises the bar instead of removing the problem. Each problem fixes an integer function $f(n)$ whose values are known only for small n, and the benchmark keeps a lower and an upper record on it. A record is a computable Lean function with a machine-checked proof that it bounds f. A submission proposes a new function and proves that it bounds f, that it is no worse than the current record at any n, and that it is strictly better at some n. Once accepted, it becomes the record that the next submission has to beat. The system thus chooses the bound as well as its proof, and it may improve the record at a single n or improve its growth rate. At every n, the gap between the two records shows how far the certified bounds are from determining $f(n)$. Every proof is checked by two independent Lean kernels, so no referee needs to check the mathematics. The checks also reject a function that computes f by exhaustive search at small n, and a maintainer only reads the submitter's report and looks for such a search at larger n. We release ten problems from extremal combinatorics with twenty proved initial records, together with the admission tools. As a baseline, agents based on a frontier language model, with at most one hour per record, had 82 submissions accepted and improved every record, without going beyond the published literature. Agents based on two newer models then started from these records, had 60 more submissions accepted and improved 19 of the 20 records again. For eight of the ten problems, at least one record is still asymptotically behind the best published bound.

1 INTRODUCTION

AI systems have made rapid progress in mathematics. On miniF2F, a benchmark of formalized olympiad problems, the best prover solved 29.2% of the test problems when the benchmark appeared in 2021 (Zheng et al., 2022), and 99.6% four years later (Chen et al., 2025). In 2025, a system reached a gold-medal score at the International Mathematical Olympiad, certified by the competition's coordinators (Luong & Lockhart, 2025), and AI systems have since resolved a few open problems posed by Erdős (Feng et al., 2026).

This progress is hard to measure. A benchmark of solved problems has a maximum score, and as the best systems approach it, their scores can no longer be told apart (Akhtar et al., 2026). Four systems have reported solving all 672 problems of PutnamBench (Tsoukalas et al., 2024; PutnamBench maintainers, 2026), and the maintainers of FrontierMath consider its hardest tier saturated (Stewart, 2026). The answers of such benchmarks are often public, so they may also end up in the training data of the systems they evaluate. Benchmarks of open problems, such as FrontierMath Erdős (Adamczewski & Bloom, 2026), avoid known answers, but each problem yields at most one first resolution. After that, its solution is public, and a later system that solves it may be reproducing a published solution, so the set of informative problems shrinks as the systems improve. Their scores count resolved problems, and intermediate progress, such as a better bound, earns little or no credit. In both kinds of benchmark the statement to be proved is given, and the system is not asked to find it. Record archives such as EinsteinArena (Bianchi et al., 2026) and Optimization Constants in Math-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ematics (Davis et al., 2026) keep improvable records, but a record there is a number for a fixed problem, checked by running a witness or by citing a paper, and no kernel checks a bound's proof.

We propose SQUEEZEBENCH, a benchmark designed to avoid these difficulties. Each problem fixes a function $f\colon \mathbb{N} \to \mathbb{N}$ whose values are known only for small n, such as the size of the largest cap set in $\mathbb{F}_3^n$ or the diagonal Ramsey number $R(n, n)$. For each problem the benchmark keeps two records, a lower bound and an upper bound on f. A record is a computable function written in Lean 4 (de Moura & Ullrich, 2021) together with a proof that it bounds f. A submission proposes a new function and proves three obligations: it bounds f, it is no worse than the current record at any n, and it is strictly better at some n. An accepted submission becomes the new record, and the next submission has to beat it, so a success leaves the problem in the benchmark with a stronger record. We call this property *resilience*. It does not guarantee that systems remain distinguishable, since the benchmark stagnates when no system improves any record. At each n the two records enclose the unknown value $f(n)$, and the distance between them shows how far the certified bounds are from determining $f(n)$ (Figure 2). We include only problems with no known closed formula for f.

Each accepted submission is stored with its author and date. At any time each record has a holder, and the record history of a problem shows who held which bound, when, and by how much each submission moved it. This history is the leaderboard. We do not reduce it to a single score, since progress on twenty records, each a function of n, has many dimensions. To compare systems in a controlled way, we start each of them from the same snapshot of records with the same budget and compare the records they reach, value by value, not the number of accepted submissions. Replaying a bound that the current record already matches earns nothing, and a bound beyond the published literature cannot have been copied from it. Our initial records are elementary bounds weaker than the published ones, so the first steps can draw on the literature, and we report this catch-up phase as such.

The system proposes the bound itself, and only the three obligations are fixed. An improvement at a single n is accepted, and so is an improvement of the asymptotic growth. This freedom has a catch. The function f is itself a correct bound, and a record that computes f by exhaustive search could never be improved. The checks therefore evaluate each new record at every $n \leq 100$ within a fixed budget, which rejects such a search below $n = 100$, and a maintainer reads the submitter's report and looks for a search placed above it. Otherwise admission needs no mathematical referee. The proofs are verified by two independent Lean kernels against statements fixed before the submission, and the other checks also run mechanically. Once per problem, a maintainer also checks that the formal definition matches the intended question.

As a baseline, we ran agents based on a frontier language model, Opus 5 in Claude Code, one per record, with at most one hour each and 18.9 hours in total, starting from the initial records. They had 82 submissions accepted and improved every record at least once, for example the upper bound for the Ruzsa–Szemerédi problem from $\binom{n}{3}$ to $\lfloor n(n+3)/18 \rfloor$. Nearly all accepted submissions use general arguments and explicit constructions, not exhaustive search. None goes beyond the published literature, and for every problem at least one record stays asymptotically behind the literature.

We then ran agents based on two newer models from the baseline's records, Fable 5.1 on 19 records and Opus 5.5 on the lower record for Golomb rulers, in one run of up to about two hours per record, and the agents could consult a second language model. This continuation is not a controlled comparison of models but a test of resilience. The records left by one system were the starting point of the next, and the newer agents had 60 submissions accepted and improved 19 of the 20 records again. The number of records that match the leading term of the best published bound rose from four to ten, for example through the Ellenberg–Gijswijt bound for cap sets and Shearer's bound for $R(3, k)$, both formalized by the agent itself. All agents could search the web. One of them found a public Lean formalization of a published construction and adapted it, which the benchmark accepts as part of the catch-up phase when the license allows it and the source is credited. Neither run exhausted the problems. For eight of the ten problems, at least one record is still asymptotically behind the best published bound (Table 2).

Our contributions are as follows.

- A benchmark design that measures mathematical progress without a fixed target: lower and upper records that are functions of the parameter, replaced only by submissions that prove the three obligations (Section 2).
- Ten problems from extremal combinatorics, formalized in Lean 4 on top of Mathlib, each with proved initial lower and upper records (Section 4).
- An admission pipeline that checks a submission mechanically against statements fixed in advance, with two independent kernels, the three standard axioms and a computation budget, together with tools to prepare, verify and accept submissions (Section 2).
- An evaluation of agents based on three frontier language models on all twenty records, a baseline from the initial records and a continuation from the baseline's records, with the harness that runs the baseline (Section 5).

2 FRAMEWORK

As part of our benchmark, we formalize ten problems in the field of extremal combinatorics. This branch of mathematics studies how large or how small a finite object can be, subject to certain conditions. For instance, as we will further explain in Section 4.2, SQUEEZEBENCH includes the *Golomb ruler* problem (Bloom & Golomb, 1977), which asks the following question: for every natural number n, what is the smallest possible length of a set of n integers whose pairwise differences are all distinct, where the length is the difference between the largest and the smallest integer? In other words, these are rulers with n marks that measure no distance twice. For small values of n, one could even solve this problem by hand: for $n = 3$, the ruler $\{0, 1, 3\}$ has length 3, but no ruler of length 2 exists, since $\{0, 1, 2\}$ measures the distance 1 twice. Thus, the smallest possible length is 3. For $n = 4$, one can find rulers of length 6, such as $\{0, 1, 4, 6\}$, but not of length 5, since the six pairwise differences would be distinct positive integers no larger than 5. Therefore, the shortest ruler has length 6. However, as the value of n increases, finding such shortest rulers becomes intractable, since there are $\binom{L-1}{n-2}$ candidate rulers of length L. Although there is no known solution for the Golomb ruler problem for all possible values of n, researchers have proved upper and lower bounds on these numbers (Singer, 1938; Erdős & Turán, 1941; Lindström, 1969). This problem is typical of the problems in SQUEEZEBENCH. For each of them, the exact answer is known only for a small number of values of the parameter, or for special families of them, and no closed formula is known for the general case. In contrast, we consider bounds that are explicit: each has to be a computable function that can be evaluated at every value of the parameter. They can be closed formulas, tables or programs that build explicit constructions. The task is to narrow the gap between these known functions and the unknown answer. Note that our requirement on bounds is stronger than what the literature typically provides, where the best bounds are often asymptotic, with unspecified constants or valid only for sufficiently large values.

In short, SQUEEZEBENCH keeps the best lower and upper bounds on each f proved so far and accepts a submission only if it provably improves one of them. The rest of this section covers how a problem is formalized, what counts as a bound, how records are kept, what a submission consists of and how it is verified.

Problems. We have formalized each problem in Lean 4, building on its mathematical library, Mathlib (The mathlib Community, 2020). The answer to each problem is a function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the optimal value for the parameter n. In Lean, f is defined by its combinatorial meaning, typically as a maximum or minimum over a finite set, but no closed form for f is known. This is what makes the benchmark resilient to saturation: if f had a closed form, the lower and upper bounds could meet at every n, and the problem would be solved. This could eventually happen—and we would welcome a submission that settles an open problem—but we select problems for which it is not expected soon. Each problem also fixes a smallest parameter value $n_{\min}$, in practice 1, 2 or 3, below which f is set to 0. Some definitions degenerate at such small values, and pinning them lets a bound be stated for every n without side conditions.

Bounds and records. A *lower bound* on f is a function $l : \mathbb{N} \to \mathbb{N}$ with $l(n) \leq f(n)$ for all n, and an *upper bound* is a function u with $f(n) \leq u(n)$ for all n. For each problem, SQUEEZEBENCH keeps a lower and an upper *record*, the best bound proved so far in each direction, as a sequence of

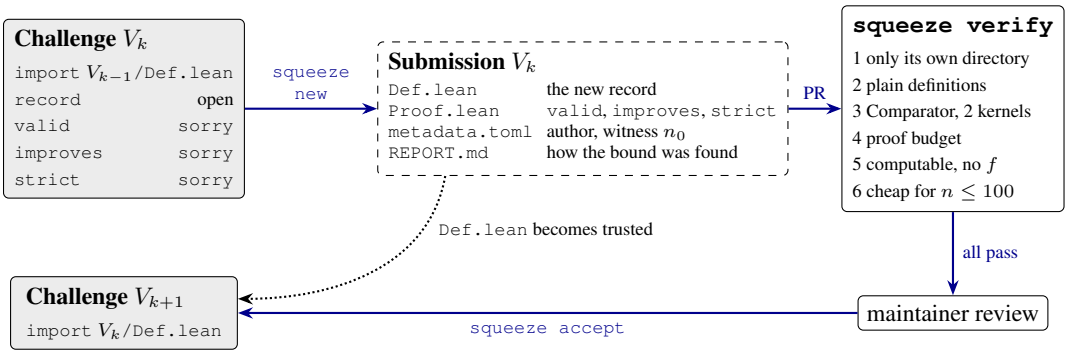


Figure 1: One version step: trusted files written by `squeeze` (solid), untrusted submission (dashed).

versions that starts from an elementary seed (Figure 2). The *record gap* $u(n)-l(n)$ measures how far the records are from determining $f(n)$; for Golomb rulers, $536 \leq f(28) \leq 585$. A submission that replaces a lower record l consists of a new Lean function l' and proofs of three theorems: `valid` ($l' \leq f$), `improves` ($l \leq l'$), both for all n, and `strict` ($l(n_0) < l'(n_0)$ for some $n_0 \geq n_{\min}$). For upper records, the inequalities are reversed. The `valid` theorem is usually the one that requires most of the proof effort in a submission. The other two are more easily automatable: we provide *idioms* that allow defining a new bound from the current record by improving it at a single point or taking the pointwise maximum with another valid bound, so the `improves` requirement is trivial in these cases. For proving `strict`, the bounds are evaluated and compared at a witness point n_0.

Challenges and submissions. A submission must be checked without trusting the person who wrote it. For this, we use Comparator (Lean FRO, 2026a), a tool from the Lean FRO for judging Lean proofs, which is also used by other projects such as `lean-eval` (Lean FRO, 2026b) and Palomar (Palomar, 2026). Comparator takes a trusted *challenge*, which states theorems with their proofs left as `sorry` and may leave some definitions open, and an untrusted *solution*, which fills them in. It checks that the solution proves exactly the statements of the challenge and uses no axioms beyond a given list. We allow only the three standard axioms `propext`, `Quot.sound` and `Classical.choice`, so `sorry` and `native_decide` are rejected. It then checks the proofs with the Lean kernel and, in our setup, also with nanoda (Bailey, 2026), an independent implementation of the kernel. Our design follows this interface. For every new version of a record, SQUEEZEBENCH provides a *Challenge*: a trusted Lean file in which the new `record` definition is left open, and `valid`, `improves` and `strict` are stated in terms of f and the current record. The solution is split into two Lean files: `Def.lean` defines the new function, and `Proof.lean` imports it and proves the three theorems. The split matters because the next Challenge imports `Def.lean` to refer to this record, which makes that file trusted; keeping the proofs out of it keeps the trusted part small. Two more files complete the submission: `metadata.toml` gives the author name and the witness n_0, and `REPORT.md` explains how the bound and its proofs were obtained and the setup used, including the models or agents. SQUEEZEBENCH will be hosted in a public GitHub repository, and a submission is a pull request that adds these four files.

Verification and acceptance. Every step is run by `squeeze`, a command-line tool that we implemented in Lean (Figure 1). A submitter runs `squeeze new`, which creates the four files of a submission with the statements copied from the open Challenge, and opens a pull request that adds them. `squeeze verify` checks the submission in six *gates*, both locally and on the pull request.

- *Gates 1–2.* They run before any submitted code is built. The pull request may change only files inside the new record's directory, and `Def.lean` must be at most 64 KiB and contain only plain definitions, since later Challenges import it and anything else could alter them.
- *Gate 3.* Comparator, as described above.
- *Gates 4–6.* They check what Comparator does not. (4) The proofs in `Proof.lean` stay within a computation budget, per declaration and in total, so that they argue instead of making the kernel search by brute force. (5) The new record is computable, and its definition

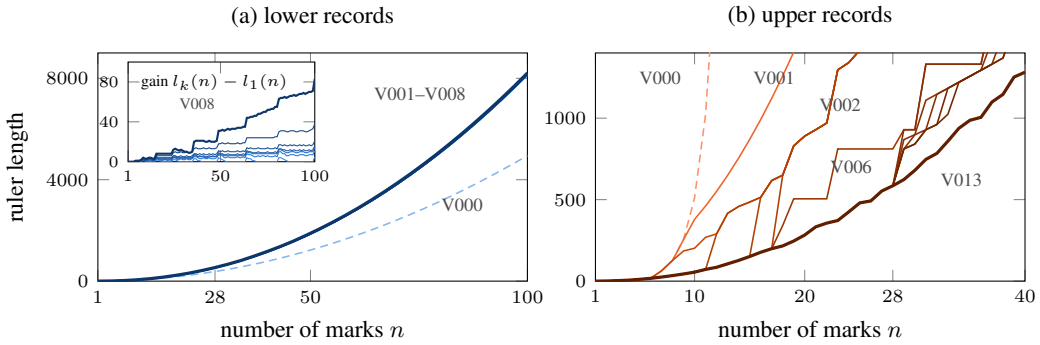


Figure 2: Every accepted record for the Golomb ruler problem. Colors run from light for the oldest version to dark for the newest, the standing record is drawn thick, and each version is at least as good as its predecessor at every n, which its submission proved. *(a)* The lower records for $n \leq 100$: the seed $\binom{n}{2}$ (dashed) and versions V001–V008, which differ only in lower-order terms and overlap at this scale. Writing l_k for version V00k of the lower record, the inset shows the gain $l_k(n) - l_1(n)$ of each later version over V001, for $k = 2, \ldots, 8$ ($k = 8$ drawn thick). *(b)* The upper records for $n \leq 40$, from the seed (dashed) to the standing record V013.

does not refer to f. (6) The kernel evaluates the new record at every n from $n_{\min}$ to 100, and at n_0 if it is larger, each within a fixed budget.

Gate 6 lets the next submitter evaluate the record to beat it. It also rejects a record that computes f by exhaustive search, such as a brute-force computation of f under another name, which passes every other gate; such a record is a correct bound that no later submission could improve. No known algorithm computes $f(n)$ within the budget for n near 100, while every record accepted so far passes. Budgets are counted in heartbeats, Lean's measure of work, so for fixed Lean and Mathlib versions they do not depend on the machine. Once all gates pass:

- *Review.* A maintainer reads the submission, including its report, and merges it. The maintainer also rejects a search that starts above $n = 100$.
- *Next Challenge.* `squeeze accept` writes the next Challenge, stated against the new record. If two submissions answer the same Challenge, the first one merged wins, and `squeeze verify` rejects the other until it is restated against the new record.
- *Audit and history.* Since Challenges are never edited, `squeeze audit` can re-check every past version against the Challenge that admitted it. Each problem's page in the repository lists every accepted version with its date, author and witness.

3 Related Work

SqueezeBench lies between mathematical reasoning benchmarks, archives of best known results and formal theorem proving, and Table 1 compares it with benchmarks and archives.

Benchmarks of solved problems, such as miniF2F (Zheng et al., 2022) and ProofNet (Azerbayev et al., 2023), hold out known answers, which can leak into training data, and they saturate. Benchmarks of open problems avoid known answers. FrontierMath Erdős (Adamczewski & Bloom, 2026) asks for Lean proofs or disproofs of 68 Erdős problems, and Formal Conjectures (Firsching et al., 2026) collects open conjectures from many areas, such as number theory and spectral theory, and HorizonMath (Wang et al., 2026) scores programs that compute values for open problems against fixed baselines.

Archives of best known results track moving targets, such as the surveys of small Ramsey numbers (Radziszowski, 2026) and of cages (Exoo & Jajcay, 2013). Their updates rest on human checking, and each covers its own problems. Newer collections gather many problems in one place, such as Leaps in Bounds (LoB) (Munro, 2026), FrontierShift (AItonomy Foundation, 2026) and

Table 1: Mathematical benchmarks and record archives. *NA* marks a system that maintains no standing record. Only where a submission replaces a *total function* can a replacement be tighter at one n and looser at another, and only SQUEEZEBENCH both replaces functions and proves that the replacement never regresses. **Notes:** [a] Formal Conjectures' function-valued bound goals are unproved (`:= by sorry`) and asymptotic, so improving a single n earns no credit. [b] HorizonMath compares against a fixed value in `data/baselines.json`; beating it does not replace it. [c] SorryDB and ArXivLean refresh the task stream: an answer key exists, but is kept fresh. [d] Formal Conjectures' anchors `c1a_lower_bound` and `c1a_upper_bound` both carry `sorry` while tagged as solved. [e] FrontierShift shows a gap on some pages, but only its upper side is a maintained record. **Sources.** PutnamBench (Tsoukalas et al., 2024), FormalMATH (Yu et al., 2025), Lean Workbook (Ying et al., 2024), CombiBench (Liu et al., 2025), SorryDB (Letson et al., 2026), ArXivLean (Gehrunger et al., 2026), FM: Open Problems (Epoch AI, 2026), Covering (Acerbi, 2026), Grassl (Grassl, 2026), EinsteinArena (Bianchi et al., 2026). The remaining systems are cited above.

	Known	Moves	Non-reg.	Target	2-sided	Cited	Check	Mode	Sub.
Benchmarks: A system is scored against mathematical problems									
miniF2F, ProofNet, PutnamBench, FormalMATH, Lean Workbook	yes	no	NA	prop.	NA	NA	proof	auto	proof
CombiBench	yes	no	NA	prop.	NA	NA	proof	auto	answer + proof
SorryDB, ArXivLean	yes[c]	no	NA	prop.	NA	NA	proof	auto	proof
FrontierMath Erdős	no	no	NA	prop.	NA	NA	proof	auto	proof
Formal Conjectures	no	no	no	function[a]	yes	yes[d]	proof	auto	answer term[a]
HorizonMath	no	no[b]	NA	scalar	no	yes	exec.	auto	Python program
FM: Open Problems	no	no	NA	scalar	no	yes	exec.	auto	real value
Records: An archive of the community's best known results									
Ramsey, Cages	no	yes	yes	table	yes	yes	none	human	citation
LoB, Covering, Grassl	no	yes	yes	table	yes	yes	exec.	human	construction
Opt. Constants	no	yes	yes	scalar	yes	yes	none	human	a claim + link
EinsteinArena	no	yes	yes	scalar	no	yes	exec.	auto	a witness
FrontierShift	no	yes	yes	scalar[e]	no	yes	exec.	human	a witness
SqueezeBench	no	yes	yes	**function**	**yes**	**no**	proof	auto	**function + proofs**

Optimization Constants in Mathematics (Davis et al., 2026), which Terence Tao started[1]. They are updated semi-automatically and accept a claim backed by a citation, without a formal proof. FunSearch (Romera-Paredes et al., 2024), AlphaEvolve (Novikov et al., 2025) and PatternBoost (Charton et al., 2024) find constructions by search and check each with a program; FunSearch's cap sets of growth 2.2202 exceed our lower record. These systems are potential submitters, not competitors, since a construction enters the benchmark once proved in Lean. SQUEEZEBENCH is not a search method: it adds a Lean proof valid at every n and keeps a lower and an upper record.

Progress on such problems rarely comes as a single proof. In combinatorics in particular, the bounds on an extremal function are tightened step by step by many researchers, and even an improvement at a single value can matter. A benchmark that asks for a proof of a fixed statement or for a better scalar cannot credit such steps on a function whose values are unknown, and a single proof of a conjecture does not credit the earlier work it builds on.

The columns of Table 1 record whether the answers are *Known*, whether a better submission replaces the incumbent (*Moves*), whether the incumbent can never get worse (*Non-reg.*), the *Target* (a proposition, a scalar or a function of n), whether a lower and an upper bound are both kept (*2-sided*), whether trust rests on a citation (*Cited*), what performs the *Check* (a kernel, or a program run on a witness), whether the check is automatic or human (*Mode*), and the submission (*Sub.*).

[1] https://terrytao.wordpress.com/2026/01/22/a-crowdsourced-repository-for-optimization-constants/

4 PROBLEMS

We include ten problems in SQUEEZEBENCH, but the list is not closed. We will keep adding problems that meet the same requirements as the current ones (Section 2), and we welcome proposals of new problems from the community. Appendix A describes how a new problem could be added.

4.1 THE TEN PROBLEMS

The ten problems come from three areas of extremal combinatorics (Table 2): additive combinatorics (cap sets, Roth numbers, the Ruzsa–Szemerédi problem and Golomb rulers), extremal graph theory (graphs without a 4-cycle, graphs without cycles of length 3 or 4, and subgraphs of the hypercube without a 4-cycle), and Ramsey theory (the diagonal Ramsey numbers $R(n, n)$, the Ramsey numbers $R(3, k)$ and the Schur numbers). Each problem has a single integer parameter, and its exact values are known only up to a small size, for example $n \leq 4$ for $R(n, n)$, $k \leq 5$ for the Schur numbers, $n \leq 6$ for cap sets and for the hypercube, and $n \leq 28$ for Golomb rulers.

Table 2: The ten problems and their standing records after the runs of Section 5. Seed records are omitted; each is the simplest provable bound, and several are trivial. Golomb is a minimization problem, and its upper record and lower record are the construction and impossibility argument, respectively. For all other problems, constructions give the lower record. * matches the leading term of the best published bound, or its base if the bound is exponential, in that direction. † matches the best elementary bound; the published frontier uses the triangle removal lemma. Unmarked records are asymptotically behind the literature.

Problem	f counts	Domain	Standing lower record	Standing upper record
Additive combinatorics				
Cap set	largest subset of $\mathbb{F}_3^n$ with no three collinear points	$n \geq 1$	growth $112^{1/6} \approx 2.196$ (extended products of caps)	$O(2.756^n)$ (Ellenberg–Gijswijt) *
Roth	largest subset of $\{0, \ldots, n-1\}$ with no 3-term progression	$n \geq 1$	$\Theta(n^{0.638})$; exact for $n \leq 209$	$4n/11 + O(1)$ (Moser)
Ruzsa–Szemerédi	most triangles in an n-vertex graph whose every edge lies in exactly one triangle	$n \geq 3$	$\Theta(n^{1.716})$ (from cap sets)	$n^2/18 + O(n)$, smaller for $n \geq 33$ (counting) †
Golomb	shortest ruler with n marks and all differences distinct	$n \geq 1$	$n^2 - 2n^{3/2} + O(n)$ *	$\approx n^2$ (Singer) *; best known for $n \leq 100$
Extremal graph theory				
C_4-free	most edges in an n-vertex graph with no 4-cycle	$n \geq 1$	$\approx \frac{1}{2} n^{3/2}$ (sub-polarity graphs) *	Reiman's bound $- O(\sqrt{n})$ *
Girth five	most edges in an n-vertex graph with no 3- or 4-cycle	$n \geq 1$	$\Theta(n^{3/2})$ for all n (sub-planes of projective planes)	Moore bound $+ O(1)$ *
Hypercube C_4	most edges in a C_4-free subgraph of the n-cube	$n \geq 1$	$(n + \Theta(\sqrt{n}))\, 2^{n-2}$ (Brass–Harborth–Nienborg) *	≈ 0.623 of the edges (Chung)
Ramsey theory				
Ramsey	least N such that every 2-coloring of K_N has a monochromatic K_n	$n \geq 2$	$(1 + o(1)) \frac{\sqrt{2}}{e}\, n\, 2^{n/2}$ (Spencer) *; best known at $n = 5, 6$	Erdős–Szekeres with Walker's step
$R(3, k)$	least N such that every N-vertex graph has a triangle or k independent vertices	$k \geq 2$	$\approx (k/2)^{3/2}$ (ordered point–line incidences)	$(1 + o(1))\, k^2 / \log k$ (Shearer) *
Schur	largest N such that $\{1, \ldots, N\}$ has a k-coloring with no monochromatic $x + y = z$	$k \geq 1$	growth $380^{1/5} \approx 3.28$ (template constructions) *	$\lfloor e\, k! \rfloor - k$ for odd k

```
def IsSidon (S : Finset ℕ) : Prop :=
  ∀ a ∈ S, ∀ b ∈ S, ∀ c ∈ S, ∀ d ∈ S,
    a + b = c + d → (a = c ∧ b = d) ∨ (a = d ∧ b = c)
def RulerLengths (m : ℕ) : Set ℕ :=
  {L | ∃ S : Finset ℕ, S ⊆ range (L + 1) ∧ S.card = m ∧
        IsSidon S ∧ L ∈ S}
noncomputable def f (m : ℕ) : ℕ :=
  if m < nMin then 0 else sInf (RulerLengths m)
```

Figure 3: The Lean definition of f for Golomb rulers (parameter m, $n_{\min} = 1$).

4.2 Running example: Golomb rulers

The values of $f(n)$ for the Golomb ruler problem of Section 2 are known for $n \leq 28$; the last five were settled by exhaustive searches by distributed.net (OEIS Foundation Inc., 2026). The lower seed is $\binom{n}{2}$, since its $\binom{n}{2}$ distances are distinct and at most the length. The upper seed is the shorter of the binary ruler $\{0, 1, 3, \ldots, 2^{n-1} - 1\}$ and a ruler of length about $2n^3$, patched at $n = 5, 6$.

Figure 3 shows how f is formalized. Like every problem, it builds on Mathlib (The mathlib Community, 2020), and four problems reuse a Mathlib definition of f directly, such as `rothNumberNat` for Roth numbers. The ruler condition is stated through sums: in a Sidon set, $a + b = c + d$ determines the pair, which for a finite set of integers is equivalent to all positive differences being distinct. Since f is an infimum and the infimum of the empty set is 0 in Lean, the file proves that the binary ruler above is a ruler; without it, no lower bound on f could be proved. The definition is `noncomputable`: nothing in the pipeline evaluates f, and a record may not refer to f or `RulerLengths`, since it would compute f under another name.

5 Evaluation

On the ten problems included as part of SqueezeBench, we performed an agent-based evaluation that provides a baseline and a continuation from its records. It has two phases:

1. We obtained preliminary bounds using Claude Code agents with the model Opus 5, starting from elementary seeds under a budget of one hour (or until two idle sessions in a row) and 16 GiB of memory. They ran in parallel on an AMD Ryzen 9 5900X machine with 24 threads at 3.7 GHz and 125 GiB of memory under Ubuntu 22.04.5 LTS.
2. From the bounds of the first phase, we launched Fable 5.1 agents at maximum effort on 19 of the 20 records, and an Opus 5.5 agent on the lower record for Golomb rulers, with the possibility of consulting GPT-6 Astra through its API. Each agent had at most 32 GiB of memory, and each run took up to about two hours. We sent a stop notice after 80 minutes or at 55% of the context, and a notice to continue when an agent stopped earlier; the first six runs ended before this rule existed. The prompt changed slightly over the runs, and a rule on outside sources was added during them. The agents ran on two AMD EPYC 9274F CPUs (96 threads) with 755 GiB of memory under Ubuntu 24.04.

In both phases, one agent was used per track (10 problems $\times$ 2 directions). They were all prompted to first read the documentation about the problem and the current records, possibly search the literature and choose a bound. Then they had to develop a Lean proof of the `valid`, `improves` and `strict` theorems, run `squeeze verify` and, if it passed, accept the bound as a new record and repeat the loop. The agents accepted records in their own copies of the repository. We re-verified every record of the second phase independently before merging it, and later re-ran `squeeze audit` on every record of both phases. The prompt also asks for improvements at infinitely many n and discourages many tiny submissions. The supplementary material has the full prompt.

Every one of the twenty records improved on its seed. In the first phase, the Opus 5 agents had 82 submissions accepted in 18.9 hours, and sixteen of the twenty tracks were stopped by the time limit rather than by two idle sessions. In the second phase, the Fable 5.1 and Opus 5.5 agents had 60 more submissions accepted and improved 19 of the 20 records again. Table 2 lists the standing records: ten match the leading term or the growth rate of the best published bound, one matches the

best elementary bound, and nine are asymptotically behind. Nearly all accepted proofs are general arguments, such as explicit algebraic constructions, products of smaller constructions, and counting arguments; `decide` only checks explicit witnesses. Gate 6 was added during the second phase, and it rejected one record that had passed the earlier checks in an agent's copy of the repository but was not yet merged. The first formalization of the Ellenberg–Gijswijt bound computed its values by a recursion over lists and exceeded the budget at every n from 59 to 100, so the next submitter could not have evaluated it. The agent resubmitted the bound with a computation that needs fewer than 100 heartbeats per n. One record may go beyond the literature at concrete values. For the Ruzsa–Szemerédi problem, a Fable 5.1 agent localized a counting argument of Fox et al. (2017), suggested by the second model it consulted, and combined it with the degree bounds of earlier versions. The resulting upper record gives $f(50) \leq 140$ and $f(100) \leq 529$, where the best bounds we could derive from the literature are 144 and 566. We found no published bound that gives these values; asymptotically, the record is still far from the $o(n^2)$ bound of the triangle removal lemma.

The running example shows the difference between the two phases (Figure 2). Opus 5 reached the Erdős–Turán ruler, of length about $2n^2$, with the optimal rulers for $n \leq 17$, and refined counting arguments over the distances of a ruler to $n^2 - 2n^{3/2} + O(n)$, the form of Lindström's bound (Lindström, 1969). Fable 5.1 brought in constructions from the literature, down to length about n^2 (Singer, 1938; Bose & Chowla, 1962; Ruzsa, 1993), and tables of explicit rulers, the optimal ones for $n \leq 28$ and the best known ones up to $n = 100$ (Rokicki & Dogon, 2016); the kernel checks each ruler. The upper record went from 1570 to the optimum 585 at $n = 28$, and from 20,002 to 8831 at $n = 100$. Three submissions by Opus 5.5 refined the lower-order terms of the lower record, from 529 to 536 at $n = 28$ and from 8143 to 8210 at $n = 100$. In total, 13 upper and 8 lower versions were accepted. The records now determine $f(n)$ for $n \leq 5$, and the gap is 49 at $n = 28$ and 621 at $n = 100$, where the seeds left 42,687 and 1,984,851. Both records match the leading term n^2 of the best published bounds and go no further; the published lower bound has a better second-order term (Balogh et al., 2023).

Apart from possibly the Ruzsa–Szemerédi record, no record goes beyond the published literature. Both records match the leading term of the best published bounds only for C_4-free graphs and Golomb rulers; for the other eight problems at least one record stays behind (Table 2). For example, the upper record for the diagonal Ramsey numbers gives $R(5,5) \leq 58$, while the published bound is 46 (Radziszowski, 2026). Some of the best bounds rest on results that Mathlib lacks, or whose explicit constants are far too weak at these n, such as the triangle removal lemma. Others were obtained by computer, and there the two directions differ. A construction found by a search comes with a witness, which is cheap to check once it is published: the agents imported the best known Golomb rulers up to 100 marks and a Schur coloring that shows $S(7) \geq 1696$. An impossibility result found by an exhaustive search has no short certificate and exceeds the computation budget, as for $f(28) = 585$, so it can enter a record only through an argument that is not known yet. Published but unformalized bounds leave room for further submissions. Beyond them, the records reach values of f that nobody knows, and formalizing the literature alone cannot saturate the benchmark.

6 Conclusions

SqueezeBench measures mathematical progress by records rather than by solved statements. A record is a computable function with a proof that it bounds f, and a submission replaces it only by proving that it is valid, no worse at any n and better at some n, checked by two kernels against statements fixed in advance. In our evaluation the records left by one system became the starting point of the next. After 82 accepted submissions of the baseline, agents based on newer models had 60 more accepted and improved 19 of the 20 records again, and eight of the ten problems still have a record behind the literature. The record history is the leaderboard.

The benchmark has limits. Its ten problems come from one area of mathematics and have one parameter each. Our runs were not a controlled comparison of models from one snapshot with equal budgets, which is how we intend the benchmark to be used. A maintainer still reads each report for a search above $n = 100$. Bounds known only from exhaustive computer searches cannot enter a record within the budget. Future work will add problems with several parameters and admit proofs that rely on a verified checker for SAT certificates, marked as such, as a route for these bounds. SqueezeBench will be public, and we invite submissions from people and AI systems alike.

USE OF AI ASSISTANTS

We used generative AI tools to implement the benchmark tools and the Lean formalizations, to give feedback on the design of the admission checks, and to compare the records with the literature. The records of Section 5 were produced by the AI agents under evaluation. The other tasks with required disclosure do not apply to this work. We also used AI tools to draft and edit text and figures, search the literature and format references. We reviewed all AI-assisted work. Code and proofs are checked by tests and the Lean kernel, and every cited source was checked against its original. We take responsibility for the final content of this paper, including text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Giovanni Acerbi. Covering Repository. https://www.coveringrepository.com/, 2026. Incorporates the data of D. Gordon's La Jolla Covering Repository up to March 2026; updates dating to November 1996; distributes validation software. Accessed 2026-09-21.

Tom Adamczewski and Thomas F. Bloom. FrontierMath Erdős, 2026. URL https://arxiv.org/abs/2609.25050. 68 Erdős problems open as of August 2026; resolution required in Lean.

AItonomy Foundation. FrontierShift: Open problems. https://aitonomy.org/projects/open-problems, 2026. 167 problems; incumbent-beating submission rule; per-problem lower/upper bounds and gap. Accessed 2026-09-21.

Mubashara Akhtar, Anka Reuel, Prajna Soni, Sanchit Ahuja, Pawan Sasanka Ammanamanchi, Ruchit Rawal, Vilém Zouhar, Srishti Yadav, Chenxi Whitehouse, Dayeon Ki, Jennifer Mickel, Leshem Choshen, Marek Suppa, Jan Batzner, Jenny Chim, Jeba Sania, Yanan Long, Hossein A. Rahmani, Christina Q Knight, Yiyang Nan, Jyoutir Raj, Yu Fan, Shubham Singh, Subramanyam Sahoo, Eliya Habba, Usman Gohar, Siddhesh Milind Pawar, Robert Scholz, Arjun Subramonian, Jingwei Ni, Mykel Kochenderfer, Sanmi Koyejo, Mrinmaya Sachan, Stella Biderman, Zeerak Talat, Avijit Ghosh, and Irene Solaiman. When AI benchmarks plateau: A systematic study of benchmark saturation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=YC1Otscjbs.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *CoRR*, abs/2302.12433, 2023. doi: 10.48550/ARXIV.2302.12433. URL https://doi.org/10.48550/arXiv.2302.12433.

Chris Bailey. nanoda_lib: a type checker for Lean 4. https://github.com/ammkrn/nanoda_lib, 2026. Accessed 2026-09-25.

József Balogh, Zoltán Füredi, and Souktik Roy. An upper bound on the size of Sidon sets. *Am. Math. Mon.*, 130(5):437–445, 2023. doi: 10.1080/00029890.2023.2176667. URL https://doi.org/10.1080/00029890.2023.2176667.

Federico Bianchi, Yongchan Kwon, Aneesh Pappu, and James Zou. Harnessing the collective intelligence of AI agents in the wild for new discoveries. *CoRR*, abs/2606.10402, 2026. doi: 10.48550/ARXIV.2606.10402. URL https://doi.org/10.48550/arXiv.2606.10402.

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

R. C. Bose and S. Chowla. Theorems in the additive theory of numbers. *Commentarii Mathematici Helvetici*, 37(1):141–147, 1962. doi: 10.1007/BF02566968.

François Charton, Jordan S. Ellenberg, Adam Zsolt Wagner, and Geordie Williamson. PatternBoost: Constructions in mathematics with a little help from AI. *CoRR*, abs/2411.00566, 2024. doi: 10.48550/ARXIV.2411.00566. URL https://doi.org/10.48550/arXiv.2411.00566.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu, Yuchen Wu, Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan, Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, and Thomas Hanwen Zhu. Seed-Prover: Deep and broad reasoning for automated theorem proving. *CoRR*, abs/2507.23726, 2025. doi: 10.48550/ARXIV.2507.23726. URL https://doi.org/10.48550/arXiv.2507.23726.

Damek Davis, Paata Ivanisvili, and Terence Tao. Optimization Constants in Mathematics. https://teorth.github.io/optimizationproblems/, 2026. Announced 22 January 2026. Accessed 2026-09-21.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe (eds.), *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pp. 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37. URL https://doi.org/10.1007/978-3-030-79876-5_37.

Epoch AI. FrontierMath: Open problems. https://epoch.ai/frontiermath/open-problems, 2026. Pilot January 2026; expanded to 50 problems 31 July 2026. Accessed 2026-09-21.

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

Geoffrey Exoo and Robert Jajcay. Dynamic cage survey. *The Electronic Journal of Combinatorics*, 2013. doi: 10.37236/37. Dynamic Survey DS16; first published 2008, revision 26 July 2013.

Tony Feng, Trieu H. Trinh, Garrett Bingham, Jiwon Kang, Shengtong Zhang, Sang-hyun Kim, Kevin Barreto, Carl Schildkraut, Junehyuk Jung, Jaehyeon Seo, Carlo Pagano, Yuri Chervonyi, Dawsen Hwang, Kaiying Hou, Sergei Gukov, Cheng-Chiang Tsai, Hyunwoo Choi, Youngbeom Jin, Wei-Yuan Li, Hao-An Wu, Ruey-An Shiu, Yu-Sheng Shih, Quoc V. Le, and Thang Luong. Semi-autonomous mathematics discovery with Gemini: A case study on the Erdős problems. *CoRR*, abs/2601.22401, 2026. doi: 10.48550/ARXIV.2601.22401. URL https://doi.org/10.48550/arXiv.2601.22401.

Moritz Firsching, Paul Lezeau, Salvatore Mercuri, Miklós Z. Horváth, Yaël Dillies, Calle Sönne, Eric Wieser, Fred Zhang, Thomas Hubert, Blaise Agüera y Arcas, and Pushmeet Kohli. Formal Conjectures: An open and evolving benchmark for verified discovery in mathematics. *CoRR*, abs/2605.13171, 2026. doi: 10.48550/ARXIV.2605.13171. URL https://doi.org/10.48550/arXiv.2605.13171.

Jacob Fox, Hao Huang, and Benny Sudakov. On graphs decomposable into induced matchings of linear sizes. *Bulletin of the London Mathematical Society*, 49(1):45–57, 2017. doi: 10.1112/blms.12005.

Tim Gehrunger, Jasper Dekoninck, and Martin Vechev. ArXivLean: How well can LLMs formally prove research math? https://matharena.ai/arxivlean/, 2026. First public release 21 April 2026. Accessed 2026-09-21.

Markus Grassl. Bounds on the minimum distance of linear codes and quantum codes. http://www.codetables.de, 2026. Accessed 2026-09-21.

Lean FRO. Comparator. https://github.com/leanprover/comparator, 2026a. Accessed 2026-09-25.

Lean FRO. lean-eval. https://github.com/leanprover/lean-eval, 2026b. Accessed 2026-09-25.

Austin Letson, Leopoldo Sarra, Auguste Poiroux, Oliver Dressler, Paul Lezeau, Dhyan Aranha, Frederick Pu, Aaron Hill, Miguel Corredera Hidalgo, Julian Berman, George Tsoukalas, and Lenny Taelman. SorryDB: Can AI provers complete real-world Lean theorems? *CoRR*,

abs/2603.02668, 2026. doi: 10.48550/ARXIV.2603.02668. URL https://doi.org/10.48550/arXiv.2603.02668.

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

Junqi Liu, Xiaohan Lin, Jonas Bayer, Yael Dillies, Weijie Jiang, Xiaodan Liang, Roman Soletskyi, Haiming Wang, Yunzhou Xie, Beibei Xiong, Zhengfeng Yang, Jujian Zhang, Lihong Zhi, Jia Li, and Zhengying Liu. CombiBench: Benchmarking LLM capability for combinatorial mathematics. *CoRR*, abs/2505.03171, 2025. doi: 10.48550/ARXIV.2505.03171. URL https://doi.org/10.48550/arXiv.2505.03171.

Thang Luong and Edward Lockhart. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad. Google DeepMind blog, 2025. URL https://deepmind.google/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/. Accessed 2026-09-24.

Daniel Munro. Leaps in Bounds. https://leapsinbounds.org/, 2026. Per-parameter histories of lower and upper bounds. Accessed 2026-09-21.

Alexander Novikov, Ngân Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *CoRR*, abs/2506.13131, 2025. doi: 10.48550/ARXIV.2506.13131. URL https://doi.org/10.48550/arXiv.2506.13131.

OEIS Foundation Inc. Sequence A003022: Length of shortest (or optimal) Golomb ruler with n marks. The On-Line Encyclopedia of Integer Sequences, https://oeis.org/A003022, 2026. Accessed 2026-09-25.

Palomar. Palomar: A registry of Lean-verified mathematics. https://palomar-registry.org/about, 2026. Accessed 2026-09-25.

PutnamBench maintainers. PutnamBench leaderboard, 2026. URL https://trishullab.github.io/PutnamBench/leaderboard.html. Accessed 2026-09-24.

Stanisław P. Radziszowski. Small Ramsey numbers. *The Electronic Journal of Combinatorics*, 2026. doi: 10.37236/21. Dynamic Survey DS1; first published 1994, revision #18, 24 April 2026.

Tomas Rokicki and Gil Dogon. Possibly optimal Golomb rulers calculated for 160 to 40,000 marks. https://cube20.org/golomb/, 2016. Accessed 2026-09-25.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nat.*, 625(7995):468–475, 2024. doi: 10.1038/S41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Imre Z. Ruzsa. Solving a linear equation in a set of integers I. *Acta Arithmetica*, 65(3):259–282, 1993. doi: 10.4064/aa-65-3-259-282.

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

Elliot Stewart. The Epoch Brief, September 12, 2026. Epoch AI newsletter, 2026. URL https://epochai.substack.com/p/the-epoch-brief-september-12-2026. Accessed 2026-09-24.

The mathlib Community. The Lean mathematical library. In Jasmin Blanchette and Catalin Hritcu (eds.), *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, pp. 367–381. ACM, 2020. doi: 10.1145/3372885.3373824. URL https://doi.org/10.1145/3372885.3373824.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the Putnam Mathematical Competition. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/1582eaf9e0cf349e1e5a6ee453100aa1-Abstract-Datasets_and_Benchmarks_Track.html.

Erik Y. Wang, Sumeet Ramesh Motwani, James V. Roggeveen, Eliot Hodges, Dulhan Jayalath, Charles London, Kalyan Ramakrishnan, Flaviu Cipcigan, Philip Torr, and Alessandro Abate. HorizonMath: Measuring AI progress toward mathematical discovery with automatic verification. *CoRR*, abs/2603.15617, 2026. doi: 10.48550/ARXIV.2603.15617. URL https://doi.org/10.48550/arXiv.2603.15617.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean Workbook: A large-scale Lean problem set formalized from natural language math problems. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/bf236666a2cc5f3ae05d2e08485efc4c-Abstract-Datasets_and_Benchmarks_Track.html.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. FormalMATH: Benchmarking formal mathematical reasoning of large language models. *CoRR*, abs/2505.02735, 2025. doi: 10.48550/ARXIV.2505.02735. URL https://doi.org/10.48550/arXiv.2505.02735.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for formal Olympiad-level mathematics. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

A Adding a problem

To add a new problem to SqueezeBench, the following steps must be followed.

1. *Define the problem.* `Problems/<P>/Def.lean` defines $n_{\min}$ and f by its combinatorial meaning, pinned to 0 below $n_{\min}$, reusing a definition from Mathlib where one exists. It also provides, for each direction, a lemma that reduces a bound to its combinatorial content, such as the existence of a witness of a given size. Where f can be evaluated, its smallest values are checked by `decide` on every build. If f is a least element, the file also proves that the set is nonempty; without this, no lower record can be proved.
2. *Seed both records.* Each direction gets a version V000 in `Records/<Dir>/V000/`, with the same four files as a submission: the record in `Def.lean`, a proof of `valid` alone in `Proof.lean`, `metadata.toml` and `REPORT.md`. A seed should be simple and quick to formalize, and it may not be taken from the known values.
3. *Configure the checks.* A trusted file `Challenges/<P>/problem.toml` sets $n_{\min}$, the directions, the budget of gate 6 and the oracles, the constants that a record may not refer to: f and the definitions it is built from.

4. *Write the first Challenges.* `squeeze accept`, run once per direction, writes the statement of the seed and the first open Challenge, which asks to beat the seed.
5. *Check the seeds.* `squeeze audit` checks that the seeds pass their own Challenges, with the three standard axioms only.
6. *Describe the problem.* A `README.md` gives the statement, the known values with their sources, and the literature; `squeeze accept` later adds the record history to it.
7. *Review.* A maintainer checks that the definition of f denotes the intended object, including any reused Mathlib definition, and admits the problem. From then on it appears in `squeeze` like every other problem.

The repository also contains a tutorial problem that is not part of the benchmark, the largest subset of $\{1, \dots, m\}$ that contains no element together with its double. Its f is computable, so a newcomer can check values of f by `decide` and practice the submission workflow on it.