Performance Analysis of Repeated LLM Attempts at a Research-Level Mathematics Problem

Bartosz Naskręcki
(co-generated with Claude Code)

Abstract

We analyse eleven independent attempts by a large language model (LLM) to solve a research-level problem in arithmetic algebraic geometry. The problem requires computing an exact large integer through a chain of seven conceptual steps (S1–S7), each demanding distinct mathematical insight. All eleven attempts were run independently under identical conditions. Only **1 of 11** attempts produced the correct answer. Yet the collective knowledge across all attempts covers the majority of the solution space—a striking illustration of the *last-mile problem* in AI mathematical reasoning. This report presents anonymised performance metrics, token-budget allocation patterns, solution-step discovery rates, and behavioural observations, without disclosing the specific mathematical techniques involved.

Acknowledgements. This analysis was generated and operationalised by Bartosz Naskręcki, based on reports from Epoch AI regarding the solutions of a FrontierMath Tier 4 problem submitted by the author. The report is fully anonymised to remove all critical details of the solution and was approved by Epoch AI for public view.

Contents

1 Experimental Setup

1.1 Problem characterisation

The target problem is drawn from a research-level mathematical competition. It asks for the exact value of a summation over a family of algebraic objects. The answer is a large integer. The official solution proceeds through **seven key steps**, labelled **S1**–**S7**, described only by their conceptual role:

S1. Structural decomposition: Identify a relevant lower-dimensional structure.

S2. Invariant computation: Determine a key numerical invariant of the objects.

S3. Deep geometric structure: Recognise that the objects belong to a special class.

S4. Associated objects: Derive explicit auxiliary objects.

S5. Counting formula: Express the count in terms of data of the auxiliary objects.

S6. Constraint classification: Determine which subsets satisfy the problem's constraints.

S7. Global cancellation: Use symmetry to simplify the final calculation.

1.2 Experimental conditions

- **Model**: A frontier LLM with code-execution capabilities.
- **Iterations**: 11 independent runs under identical conditions, labelled Iter 1–10 and Final.
- **Token budget**: Each run used ∼100K+ reasoning tokens.
- **Isolation**: No inter-iteration memory; each of the 11 runs started from scratch.
- **Transcript lengths**: 1 909–3 235 lines per iteration; 26 128 lines total across Iter 1–10.

2 Overall Performance

2.1 Solution quality scores

Each attempt is scored on a 0–7 scale: 1 point per solution step fully discovered, 0.5 points for partial discovery.

Figure 1: Solution quality scores. The dashed line marks the mean of Iter 1–10 (2.2/7). Only the Final attempt achieves a perfect score.

Table 1: Score distribution (Iter 1–10).

Statistic	**Value**
Mean	2.20 / 7
Median	2.50 / 7
Standard deviation	1.42
Minimum	0.5 (Iters 2, 8, 10)
Maximum	4.5 (Iter 9)

2.2 Submission types and correctness

Submissions fell into three categories: *code* (executable algorithm), *heuristic* (numerically interpolated guess), and *zero* (conjectured trivial answer).

Table 2: Submission types: count, mean score, and correctness.

Type	Count	Mean score	Correct	Attempts
Code	6	3.4 / 7	1	I4, I5, I6, I7, I9, Final
Heuristic	3	0.7 / 7	0	I1, I8, I10
Zero	2	0.5 / 7	0	I2, I3

Key Finding

Of 11 attempts, only the Final Solution produced the correct answer, and it did so via executable code—not analytic derivation. The three heuristic guesses all overestimated the answer by 6.4–6.8%, suggesting a shared systematic bias in the estimation heuristic. The two zero-submissions arose from a specific conceptual error: confusing the vanishing of one component of the answer with the emptiness of the feasible parameter set.

3 Solution Step Discovery

3.1 Aggregate discovery rates

Figure 2 shows how many of the 11 attempts fully or partially discovered each of the seven solution steps.

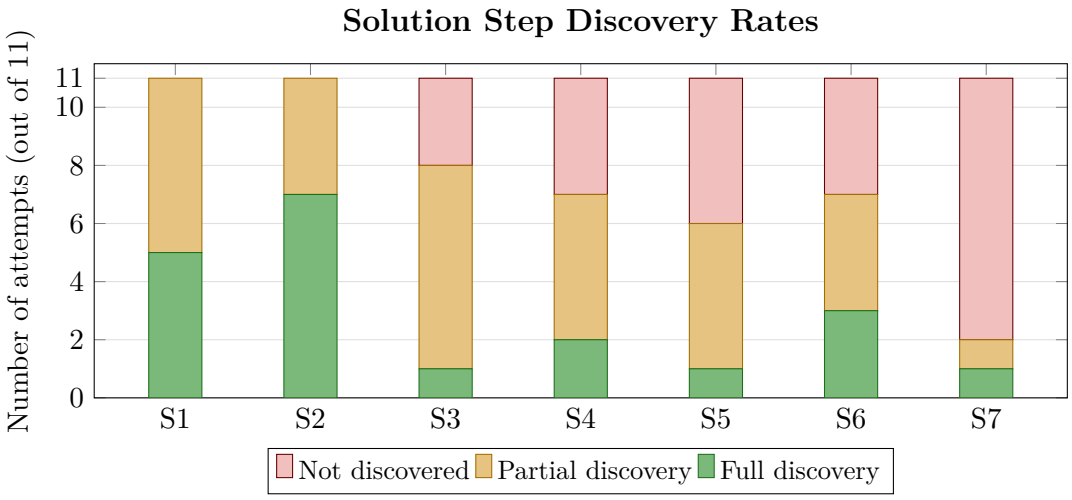

Figure 2: Stacked discovery rates for each solution step. Steps S1–S2 are universally attempted; S7 is discovered fully by only 1 of 11 attempts.

3.2 Per-attempt heatmap

Table 3 provides a detailed per-step, per-attempt breakdown.

Table 3: Solution step discovery heatmap. ✓ = full, ∼ = partial, × = not discovered.

	Iter 1	*Iter 2*	*Iter 3*	*Iter 4*	*Iter 5*	*Iter 6*	*Iter 7*	*Iter 8*	*Iter 9*	*Iter 10*	*Final*
S1	∼	∼	∼	✓	✓	∼	✓	∼	✓	∼	✓
S2	∼	∼	✓	✓	✓	✓	✓	∼	✓	∼	✓
S3	∼	×	∼	∼	∼	∼	∼	×	∼	×	✓
S4	×	×	∼	∼	∼	∼	∼	×	✓	×	✓
S5	×	×	×	∼	∼	∼	∼	×	∼	×	✓
S6	×	×	∼	∼	∼	∼	✓	×	✓	×	✓
S7	×	×	∼	×	×	×	×	×	×	×	✓
Score	1	0.5	2.5	3	3	2.5	4	0.5	4.5	0.5	**7**

Phase-transition at S3

Steps S1 and S2 are at least partially discovered by all 11 attempts (100% engagement). Starting from S3, a sharp drop-off occurs: 3 attempts miss S3 entirely, and 9 of 11 miss S7. The transition from "surface-level analysis" (S1–S2) to "deep structural insight" (S3–S7) represents a qualitative barrier the model rarely crosses fully.

4 Token Budget Allocation

Each iteration's token expenditure was classified into five phases: *symbolic manipulation* (algebraic transformations), *empirical testing* (small-case experiments), *structural analysis* (geometric reasoning), *algorithm development* (code writing), and *planning & hedging* (strategy deliberation, uncertainty expression).

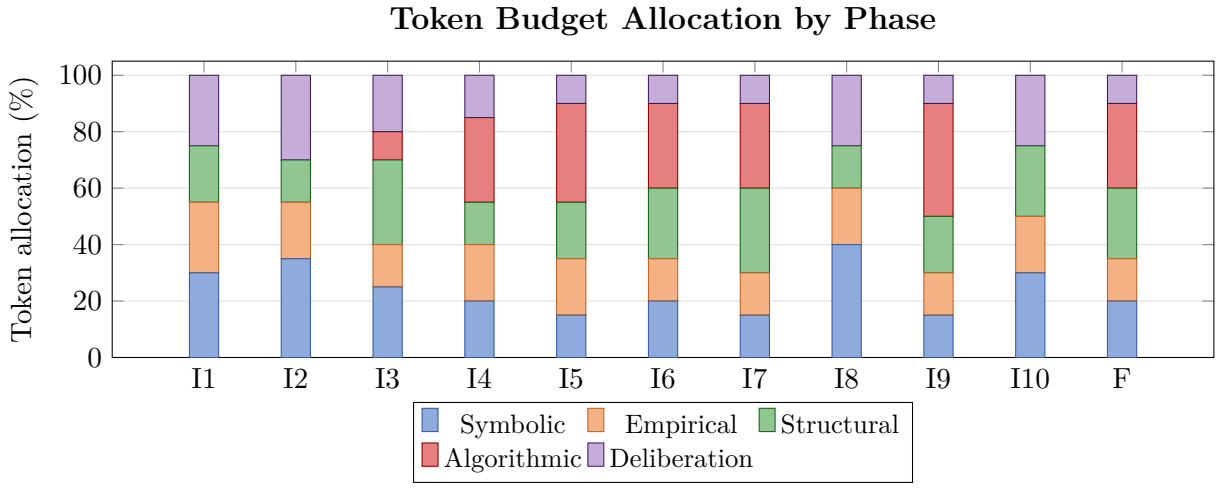

Figure 3: Approximate token budget allocation across all attempts. Code-submitting iterations (I4–I7, I9, Final) allocate 30–40% to algorithm development, while non-code iterations spend 25–30% on deliberation and hedging.

Code investment correlates with score

Iterations that allocated ≥30% of tokens to algorithm development (Iters 4, 5, 6, 7, 9, Final) achieved a mean score of **3.4/7**. Those that allocated 0% (Iters 1, 2, 8, 10) achieved a mean of **0.6/7**—a factor of **5.7**× difference. The act of writing code, even incorrect code, appears to force more rigorous structural reasoning.

5 Strategic Diversity

After a shared initial phase (algebraic decomposition and small-case experiments, consuming ∼20−30% of tokens), iterations diverged into five broad strategies:

M1. Analytic decomposition: Express the count using tools from analytic number theory.

M2. Structural identification: Recognise that the object decomposes into a product of simpler components.

M3. Parameter-space reasoning: Use dimension arguments over the parameter space to constrain invariants.

M4. Numerical extrapolation: Interpolate from small-case data to guess the answer.

M5. Classification-based filtering: Classify auxiliary objects by discrete invariants to determine the feasible set.

Table 4: Strategy adoption by attempt (• = used).

	Iter 1	*Iter 2*	*Iter 3*	*Iter 4*	*Iter 5*	*Iter 6*	*Iter 7*	*Iter 8*	*Iter 9*	*Iter 10*	*Final*
M1: Analytic	•			•	•				•		•
M2: Structural			•	•	•	•	•		•		•
M3: Parameter		•	•			•		•		•	
M4: Numerical	•							•		•	
M5: Classification				•	•		•		•		•
Count	2	1	2	3	3	2	2	2	3	2	3

The three highest-scoring independent iterations (Iter 7: score 4, Iter 9: score 4.5, Iter 4: score 3) all employed at least three strategies simultaneously, including both M2 (structural identification) and M5 (classification-based filtering)—the combination closest to the official solution.

6 Behavioural Patterns

6.1 The last-mile problem

Critical Failure Mode

The most striking pattern is the *last-mile failure*: every independent iteration discovers substantial fragments of the solution but cannot assemble them into a complete, correct computation. Across all 11 attempts:

- 11/11 partially identified S1 (structural decomposition).
- 11/11 at least partially identified S2 (key invariant).
- 8/11 conjectured the deep geometric structure (S3).
- 7/11 attempted classification-based filtering (S6).
- **1/11** fully discovered the global cancellation (S7).

The gap between correct intuition and correct implementation is consistently the bottleneck.

6.2 Universal initial trajectory

All 11 attempts independently converged on the same three initial sub-steps within the first 20−30% of their token budgets:

1. Recognise an algebraic structure and compute a key quantity.
2. Run brute-force experiments for small parameter values.
3. Determine the relationship between two natural ways of counting solutions.

This "common preamble" was remarkably consistent across runs, suggesting these steps lie well within the model's reliable capability envelope.

6.3 Systematic bias in heuristic guesses

Table 5: Relative error of the three heuristic (non-zero) guesses.

Attempt	Relative error	Direction
Iteration 1	+6.8%	Overestimate
Iteration 8	+6.5%	Overestimate
Iteration 10	+6.4%	Overestimate

All three heuristic guesses overestimate the correct answer by 6.4–6.8%, a remarkably tight cluster suggesting a shared systematic bias in the estimation strategy—likely an incorrect approximation of the feasible set size.

6.4 Pervasive uncertainty

Uncertainty regardless of correctness

Every attempt, *including the one that produced the correct answer*, expressed significant epistemic uncertainty. The Final Solution described its own correct output as "our best guess" and noted "I'm not fully confident." This pattern has four identified causes:

1. **Verification gap**: The model cannot execute its code for the target parameter value within the session.
2. **Error accumulation**: Each reasoning step introduces potential errors; the model tracks this consciously.
3. **Token pressure**: As the budget depletes, reasoning becomes compressed and less rigorous.
4. **Anomalous data**: Small-case experiments occasionally produce edge-case results that undermine framework confidence.

6.5 Depth vs. breadth trade-off

Iterations that explored many strategies superficially (Iters 2, 8, 10; mean strategies used: 1.3) achieved a mean score of **0.5/7**. Iterations that went deep into 2–3 strategies (Iters 4, 5, 7, 9; mean strategies: 2.8) achieved **3.6/7**. Depth of engagement with the problem structure, rather than breadth of exploration, is the stronger predictor of performance.

7 Score vs. Algorithm Investment

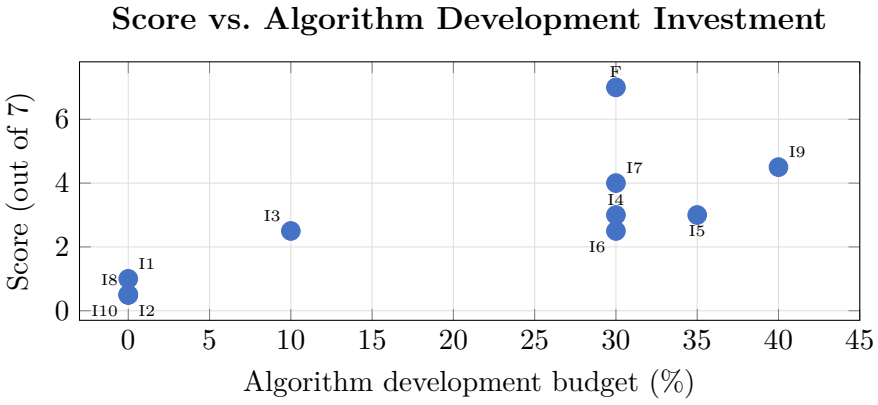

Figure 4: Scatter plot of solution quality score against the fraction of token budget devoted to algorithm development. Two distinct clusters are visible: non-coding attempts at (0%, 0.5–1) and coding attempts at (30–40%, 2.5–7).

8 Aggregate Statistics

Table 6: Summary statistics across all 11 attempts.

Metric	Value
Total attempts	11
Total transcript lines (Iter 1–10)	26 128
Mean transcript length	2 613 lines
Shortest transcript	1 909 lines (Iter 8)
Longest transcript	3 235 lines
Correct answers	1 / 11 (9.1%)
Code submissions	6 / 11 (54.5%)
Heuristic submissions	3 / 11 (27.3%)
Zero submissions	2 / 11 (18.2%)
Mean score (Iter 1–10)	2.20 / 7 (31.4%)
Mean score (all 11)	2.68 / 7 (38.3%)
Median score (Iter 1–10)	2.50 / 7
Score std. deviation (Iter 1–10)	1.42
Steps with 100% engagement	2 (S1, S2)
Steps with <20% full discovery	4 (S3, S4, S5, S7)
Hardest step (lowest discovery)	S7 (1/11 full, 1/11 partial)
Mean code-submitter score	3.4 / 7
Mean non-code-submitter score	0.6 / 7
Closest wrong answer (relative error)	6.4% (Iter 10)

9 Conclusions

1. **Partial coverage is high; full synthesis is rare.** Across 11 attempts, the model collectively engaged with all 7 solution steps. Steps S1–S2 were universally attempted; S3–S6 were partially discovered by a majority. Yet only 1 attempt assembled all pieces correctly, with only 1 of 11 runs succeeding.

2. **The bottleneck is S7 (global cancellation).** This step requires recognising a structural symmetry that simplifies the entire computation. It represents a qualitatively different type of mathematical insight—a "cancellation at scale" argument—that the model fails to discover independently.

3. **Code submission is a strong positive signal.** Code-submitting iterations scored 5.7× higher on average than non-code iterations. Writing executable code forces concrete, testable reasoning and prevents the drift into unverifiable heuristics.

4. **Depth beats breadth.** Iterations pursuing 2–3 strategies deeply outperformed those sampling many strategies superficially. The model's generative breadth—its ability to propose diverse approaches—is a strength, but without the disciplined depth to follow through, it leads to diffuse, low-scoring attempts.

5. **Self-calibration is poor.** The model expresses uncertainty uniformly, regardless of whether its answer is correct. This limits the value of confidence signals for downstream evaluation.

6. **The last-mile problem is the central challenge.** The model can generate the right mathematical framework—the right conjectures, the right structural decompositions, the right algorithmic skeletons—but cannot reliably close the gap between framework and verified computation. Bridging this gap likely requires either (a) integrated verification tools (code execution with feedback) or (b) access to prior successful attempts.