POST-TRAINING LANGUAGE MODELS FOR GOLD-MEDAL PERFORMANCE IN CODING COMPETITIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

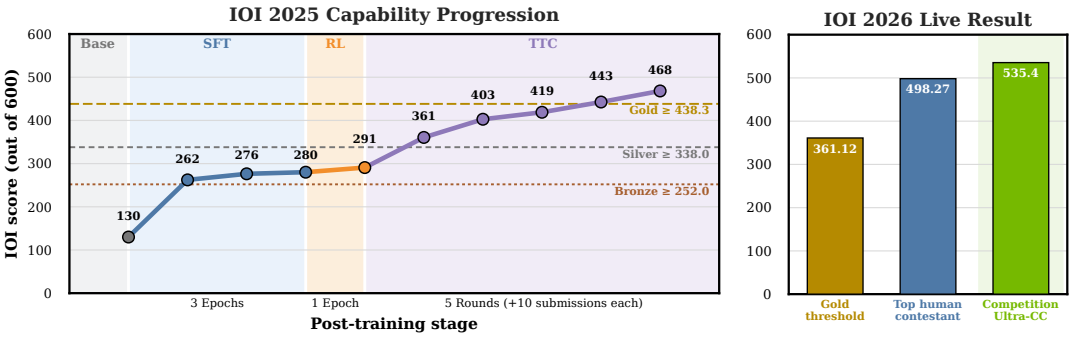
Competitive programming has become a key test of language model reasoning, with IOI and ICPC among its hardest settings. We present an end-to-end specialization pipeline combining problem curation, synthetic reasoning traces, supervised fine-tuning (SFT), reinforcement learning (RL), and test-time compute. Using 22,000 problems, we train Nano-CC (30B-A3B) with SFT and RL and Ultra-CC (550B-A55B) with SFT alone. Across Nano-CC, SFT drives most single-sample gains and RL adds modest improvements, while feedback-driven GenCorrect iteratively generates and refines diverse solutions and provides the largest system-level gain. On IOI 2025, Nano-CC improves from 130 points to 291 after post-training and to 468 with GenCorrect, exceeding the gold threshold of 438.3 while Ultra-CC reaches 502. Guided by these results, we develop a competition-specific Ultra-CC system and evaluate it prospectively during IOI 2026. Under the same time, internet-access, and submission constraints as human contestants, it scores 535.4 out of 600, exceeding both the gold threshold of 361.12 and the top human score of 498.27. To our knowledge, this is the first AI system to outscore the highest-scoring human contestant on an IOI problem set.

1 INTRODUCTION

Competitive programming has emerged as a challenging domain for evaluating the reasoning capabilities of large language models (LLMs). Unlike conventional coding benchmarks, competitive programming requires models to synthesize novel algorithms, reason over complex constraints, and produce implementations that pass hidden tests under time and memory limits. Performance on competitive programming has become a strong indicator of a model's reasoning and coding ability (Li et al., 2022; Jain et al., 2024; Zheng et al., 2025). Recent years have seen progress on these tasks, with proprietary and open-weight models achieving gold-medal performance in competitions such as the International Olympiad in Informatics (IOI) and the International Collegiate Programming Contest (ICPC) (OpenAI, 2025a; DeepSeek-AI, 2025; Samadi et al., 2026; Yang et al., 2026). Despite this progress, it remains unclear which stages of post-training and inference account for gold-medal performance, and how their contributions change when starting from a larger, stronger model.

To study this, we build a competitive-programming pipeline combining large-scale problem curation, synthetic data generation, supervised fine-tuning (SFT), reinforcement learning (RL), and iterative test-time refinement, and apply it to both a compact model with a weak starting point and a larger, stronger one. We curate 22,000 problems and use DeepSeek-V4-Flash (DeepSeek-AI, 2026) to generate 1.2 million reasoning traces for our compact model and 477,642 traces for our larger model, excluding and deduplicating all evaluation problems from training. Starting from Nemotron-3-Nano-30B-A3B (NVIDIA, 2025), we apply SFT and RL to produce **Nano-CC**, a 30B-parameter mixture-of-experts model with 3B active parameters. Starting from an intermediate Nemotron-3-Ultra-550B-A55B (NVIDIA, 2026) checkpoint, we apply SFT without code-specific RL to produce **Ultra-CC**, a 550B-parameter model with 55B active parameters. Both models use GenCorrect, our iterative inference strategy that refines diverse solutions using feedback from previous submissions. Figure 1a summarizes the progression of Nano-CC on IOI 2025. Its Score@1 increases from 130 to 280 after SFT and 291 after RL, while five GenCorrect rounds raise its score to 468, exceeding the gold threshold of 438.3 under the official 50-submission limit (International Olympiad in Informatics, 2025). Our general Ultra-CC improves from 271 to 304 with SFT alone and reaches 502 after five rounds of GenCorrect.

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.



(a) General Nano-CC on IOI 2025: SFT and RL improve Score@1, while GenCorrect achieves gold-medal performance with 50 submissions.

(b) Competition Ultra-CC surpasses highest human score.

Figure 1: Performance of our pipeline and models on IOI 2025 and IOI 2026.

After completing these general-pipeline experiments, we apply their insights to develop a competition-specific model and inference pipeline for IOI 2026. We use IOI 2025 as a development benchmark to select the training and inference adaptations used by this system. We then evaluate the resulting system live on the IOI 2026 problem set under competition time and submission, before the problems are publicly released.[1] Under matched competition time, internet-access, and submission constraints, Competition Ultra-CC scores 535.4/600, exceeding the gold threshold of 361.12 and the top official contestant score of 498.27, as shown in Figure 1b (International Olympiad in Informatics, 2026). We plan to release the competition model checkpoint and provide runnable inference and evaluation code. To summarize, our main contributions are:

- We develop an end-to-end specialization pipeline for competitive programming, including large-scale problem curation, synthetic reasoning data, long-context supervised fine-tuning, and reinforcement learning with executable rewards.
- We introduce GenCorrect, a closed-loop test-time compute strategy that iteratively generates diverse solutions, incorporates evaluator feedback, and refines subsequent generations under a constrained submission budget.
- We provide an empirical analysis of how post-training and test-time compute contribute to competitive-programming performance in both a compact model with a weak starting point and a larger, stronger model.
- We apply these findings to develop a competition-specific system evaluated live during IOI 2026. Under the same time limits, submission platform, and internet restrictions as human contestants, it scored 535.4/600, surpassing the highest-scoring human contestant. To our knowledge, this is the first time an AI system has done so on any IOI problem set.

2 Competition Settings

2.1 International Olympiad in Informatics.

IOI is an individual programming competition conducted over two contest days (IOI 2026 Organizing Committee, 2026). At IOI 2025, contestants were presented with six problems, each worth up to 100 points and divided into subtasks covering different input constraints. A submission received credit for each subtask whose test cases it passed, allowing partially correct or less efficient algorithms to earn partial scores. Contestants could submit at most 50 solutions per problem, and the six problem scores were summed to produce a maximum score of 600 (International Olympiad in Informatics, 2025; OpenAI, 2025a). Medal thresholds were determined from the final score distribution, with approximately the top 1/12 of contestants receiving gold, the next 1/6 receiving silver, and the next 1/4 receiving bronze (IOI 2026 Organizing Committee, 2026).

[1] Our system was not an official IOI contestant and the run was not supervised by IOI. Therefore, its score was not included in the official rankings and the evaluation is reported as an unofficial, unsupervised benchmark.

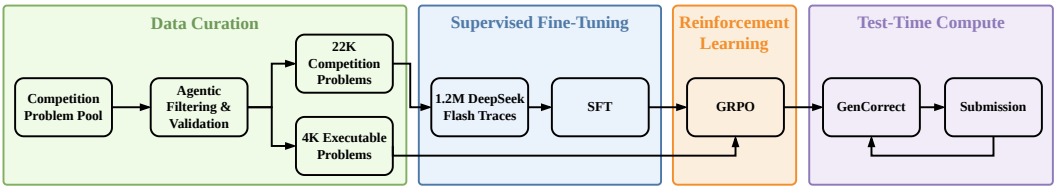


Figure 2: Our competitive-programming pipeline. Nano-CC undergoes supervised fine-tuning and reinforcement learning, while Ultra-CC uses supervised fine-tuning only. Both models use GenCorrect at inference time.

2.2 International Collegiate Programming Contest.

At the ICPC 2025 World Finals, teams of three contestants competed for five hours using a single shared computer. The contest contained 12 algorithmic problems evaluated using binary scoring: a problem was solved only when a submission passed all hidden test cases. Teams were ranked primarily by the number of problems solved, with ties broken by total penalty time. For each solved problem, penalty time consisted of the elapsed contest time before acceptance plus 20 minutes for each preceding incorrect submission (International Collegiate Programming Contest, 2025c). The top four teams received gold medals, teams placing 5th–8th received silver medals, and teams placing 9th–12th received bronze medals (International Collegiate Programming Contest, 2025b).

3 Method

Our pipeline, summarized in Figure 2, combines supervised fine-tuning (SFT), reinforcement learning (RL), and iterative test-time compute. Modifications used for our live IOI 2026 benchmark run are described in Section 5.

3.1 Data Curation

We curate 22,000 problems from 16 regional and international competition families spanning the last two decades, together with problems from online programming platforms. An automated pipeline packages each problem into an executable evaluation environment containing its statement, constraints, test cases, auxiliary files, and reference solutions. We retain only environments that produce consistent verdicts across reference and generated solutions. We exclude all IOI 2025, ICPC 2025, and LiveCodeBench Pro problems from the SFT and RL data and deduplicate the training corpus against these evaluations. IOI 2026 is a strictly prospective evaluation because our system was run before the problems were publicly released. Appendix A provides additional filtering details.

3.2 Supervised Fine-Tuning

We use DeepSeek-V4-Flash to generate 1.2 million reasoning traces for Nemotron-3-Nano-30B-A3B and 477,642 traces for Nemotron-3-Ultra-550B-A55B (DeepSeek-AI, 2026; NVIDIA, 2025; 2026). We allocate more generations to difficult problems and include self-improvement traces in which the teacher refines a previously generated solution, as visualized in Figure 9. These traces expose the models to the iterative refinement behavior used by GenCorrect. We fine-tune Nano for three epochs and Ultra for one epoch, using a global batch size of 64 and sequence packing up to 262K tokens. Nano starts from the public base model, which scores 21.7% on IOI 2025, while Ultra is initialized from its RLVR-teacher checkpoint (NVIDIA, 2026), which already scores 45.1%. This lets us test whether our recipe improves both a smaller, weaker model and a larger, more capable one. Complete optimization, parallelism, and compute settings are provided in Appendix B.

3.3 Reinforcement Learning

We apply RL only to Nemotron-3-Nano-30B-A3B. After filtering for reliable and sufficiently fast executable environments, the RL corpus contains 3,219 problems, split into 2,847 training and 372 validation problems. We split at the parent-problem level to prevent subtasks from the same problem

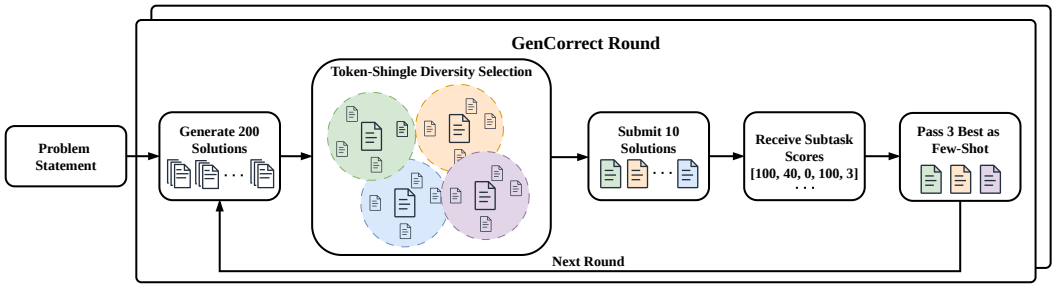


Figure 3: GenCorrect iteratively generates candidate solutions, selects a diverse subset for evaluation, and uses evaluator feedback to guide the next round.

appearing in both sets. We train using NeMo RL (NVIDIA, 2025) with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). Each step samples 16 rollouts for each of 64 prompts at temperature 1.0, yielding 1,024 rollouts. Generated C++ solutions are compiled and executed, receiving a terminal reward of 1 for full credit and 0 otherwise. We optimize a token-level clipped policy-gradient objective with no reference-policy KL penalty (Yu et al., 2025) and select the final checkpoint using held-out validation performance. Further details are provided in Appendix B.

3.4 Test-Time Compute

We introduce GenCorrect, an iterative test-time compute strategy applied for up to five rounds (Figure 3). GenCorrect combines large-scale sampling and behavior-based clustering for competitive programming (Li et al., 2022; Leblond et al., 2023), iterative self-critique and test-based refinement (Ahmad et al., 2025a; Ridnik et al., 2024), and execution-grounded test-time selection (Samadi et al., 2026; Li et al., 2025). Each round consists of:

- **Generation.** We generate up to 200 candidate solutions in parallel and compile them locally. The first round uses only the problem statement; subsequent rounds additionally use solutions and evaluator feedback from earlier rounds.
- **Diversity selection.** After filtering invalid outputs, we initialize a center set C using a score-blind local heuristic $Q(c)$ and iteratively select the candidate farthest from the existing centers:
$$c_{\text{next}} \in \arg\max_{c \notin C} \min_{z \in C} \left[1 - \text{sim}(c, z)\right]. \tag{1}$$
Here, $\text{sim}(c, z)$ denotes the candidate similarity score. We select up to $K = 10$ centers, assign every candidate to its most similar center, and choose the candidate with the highest $Q(c)$ as the representative of each cluster.
- **Execution.** We submit the 10 representatives to the evaluator. For IOI, feedback consists of subtask scores. All filtering and selection for the current round occur before these scores are observed.
- **Refinement.** We accumulate the best score observed for each subtask:
$$\begin{aligned} A_r(t) &= \max\left(A_{r-1}(t), \max_{c \in S_r} s_t(c)\right), \\ A_0(t) &= 0, \end{aligned} \tag{2}$$
where S_r denotes the solutions submitted in round r. The next round is conditioned on the accumulated per-subtask score vector (A_r) and three complementary references selected to preserve solved subtasks, target remaining gaps, and maintain diversity.

For IOI, we perform five rounds of 10 submissions, matching the official limit of 50 submissions per problem. Previous works divide problems into subtasks while our approach provides all subtasks to the model and lets it decide which to work on, leading to substantially fewer necessary generations per problem (OpenAI, 2025a; Samadi et al., 2026; Yang et al., 2026). For ICPC, where feedback is binary, we continue until the problem is solved or performance plateaus. Complete filtering, ranking, tie-breaking, and carry-forward rules along with the necessary prompts are provided in Appendix E.

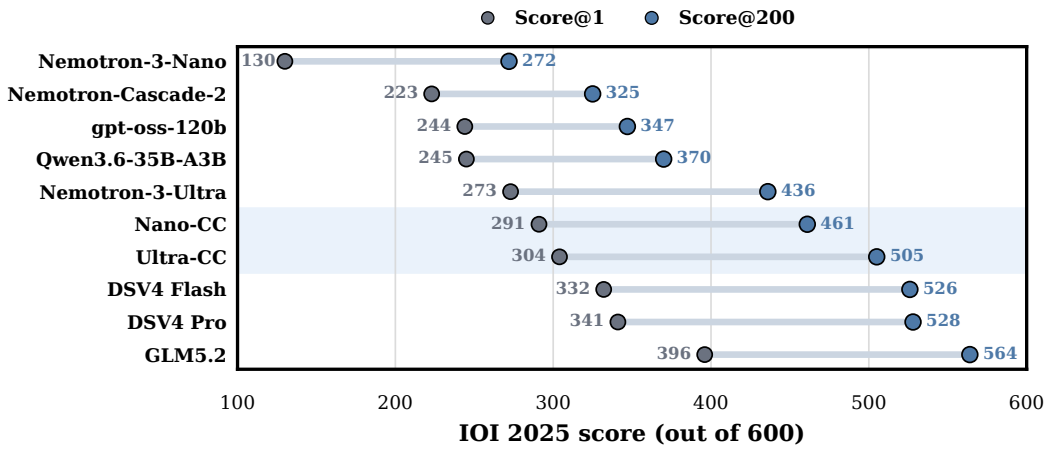


Figure 4: IOI 2025 Score@1 (grey) and Score@200 (green) across evaluated models with the grey line representing Score@k in between. Our submitted general Nano-CC and Ultra-CC are highlighted.

4 Experiments

4.1 Evaluation Setup

We evaluate on IOI 2025, ICPC 2025, and LiveCodeBench Pro (LCB Pro) (International Olympiad in Informatics, 2025; International Collegiate Programming Contest, 2025b; Zheng et al., 2025), ensuring that all evaluation problems are excluded and deduplicated from our SFT and RL data. For IOI, Score@k is computed by grouping generated solutions into independent runs. Within each run, we retain the highest score achieved on each subtask, sum across all subtasks and problems, and then average the resulting totals across runs. We report Score@1 for single-sample performance and Score@200 for parallel sampling. For ICPC and LCB Pro, Pass@1 is the fraction of problems solved by one sampled solution. IOI results are reported either as raw scores out of 600 or as normalized percentages, as indicated; ICPC Pass@1 is the percentage of the 12 problems solved. Final IOI and ICPC results are averaged over 1,000 runs, Nano-CC checkpoints over 50 runs, Score@200 over five runs, and LCB Pro over eight runs. Checkpoints are selected exclusively using the held-out validation set. We compare against gpt-oss-120b (OpenAI, 2025b), Qwen3.6-35B-A3B (Qwen Team, 2026), Nemotron-Cascade 2 (Yang et al., 2026), the base Nemotron-3 Nano and Ultra models (NVIDIA, 2025; 2026), DeepSeek-V4-Flash and DeepSeek-V4-Pro (DeepSeek-AI, 2026) (Max thinking), and GLM-5.2 (GLM-5-Team, 2026). We sample with temperature 1.0 and top-p 0.95 using a maximum context length of 262,144 tokens.

4.2 Main Results

Figure 4 and Table 1 summarize the main results while Figure 1 shows our Nano-CC progression with each pipeline stage. On IOI 2025, Nano-CC improves over its base model from 130 (21.7%) to 291 (48.5%) at Score@1 and from 272 to 461 at Score@200. Despite having only 3B active parameters, Nano-CC exceeds the base Nemotron-3-Ultra model at both sampling budgets. At Score@1, it outperforms all evaluated baselines except DeepSeek-V4-Flash, DeepSeek-V4-Pro, and GLM-5.2. The gains transfer beyond IOI: Nano-CC reaches 51.0% Pass@1 on ICPC 2025 and 71.6% on LCB Pro, compared with 16.9% and 17.6% for the base model. On LCB Pro, Nano-CC also exceeds gpt-oss-120b, Qwen3.6-35B-A3B, Nemotron-Cascade-2-30B-A3B, and DeepSeek-V4-Flash. Ultra-CC, trained with SFT but without the CC RL stage, improves over the Ultra RLVR base model from 45.1% to 50.7% on IOI, 50.1% to 57.4% on ICPC, and 70.2% to 74.5% on LCB Pro. It exceeds Nano-CC by 2.2, 6.4, and 2.9 percentage points, respectively, and achieves the strongest results among our models on all three benchmarks. At its size, Nano-CC is the strongest evaluated model with a comparable active parameter count, whereas Ultra-CC provides the best performance among our models even though it uses substantially less SFT training and no RL.

Table 1: Main single-sample results on IOI 2025, ICPC 2025, and LiveCodeBench Pro. IOI Score@1 is reported as a normalized percentage, while ICPC and LiveCodeBench Pro report Pass@1.

Model	IOI 2025 Score@1	ICPC 2025 Pass@1	LCB Pro Pass@1
Nemotron-3-Nano-30B-A3B	21.7%	16.9%	17.6%
Nemotron-Cascade-2-30B-A3B	37.2%	42.0%	45.6%
gpt-oss-120b	40.7%	45.8%	66.4%
Qwen3.6-35B-A3B	40.8%	32.0%	58.4%
Nemotron-3-Ultra RLVR-teacher	45.1%	50.1%	70.2%
Nemotron-3-Ultra-550B-A55B	45.5%	54.0%	72.6%
DSV4 Flash	55.3%	65.8%	69.5%
DSV4 Pro	56.8%	69.6%	78.2%
GLM 5.2	66.0%	65.7%	83.8%
Nano-CC	48.5%	51.0%	71.6%
Ultra-CC	50.7%	57.4%	74.5%

4.3 Effect of Supervised Fine-Tuning

Figure 5 shows that SFT produces most of Nano-CC's improvement. Over three epochs, IOI 2025 Score@1 increases from 21.7% to 46.7%, ICPC 2025 Pass@1 from 16.9% to 47.3%, and LCB Pro Pass@1 from 17.6% to 70.7%. Most gains occur during the first epoch, with performance beginning to saturate by the third. Ultra-CC receives a single SFT epoch over 477,642 examples, improving IOI from 45.1% to 50.7%, ICPC from 50.1% to 57.4%, and LCB Pro from 70.2% to 74.5%. These gains are smaller than Nano's SFT improvements of 25.0, 30.4, and 53.1 percentage points, respectively, reflecting the substantially stronger Ultra initialization. Nevertheless, the SFT-only Ultra-CC model outperforms the final Nano-CC model on all three benchmarks after roughly 478k SFT samples, whereas Nano-CC receives three SFT epochs 1.2M samples each, followed by RL. Thus, when model size and inference cost are not primary constraints, adapting a stronger base model with limited SFT can outperform extensive post-training of a smaller model.

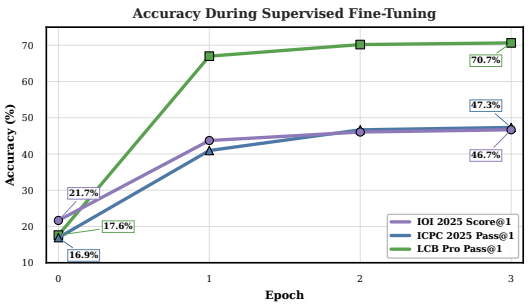


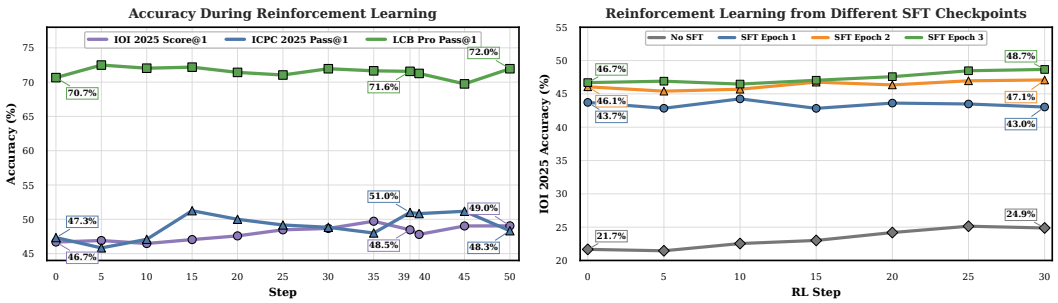
Figure 5: Normalized IOI 2025 Score@1, ICPC 2025 Pass@1, and LCB Pro Pass@1 before and during supervised fine-tuning of Nano-CC.

4.4 Effect of Reinforcement Learning

We apply RL only to Nano-CC, in part because of the substantial computational cost of running RL at Ultra's scale. As shown in Figure 6a, starting from the third-epoch SFT checkpoint, RL improves IOI 2025 Score@1 from 46.7% to 48.5%, ICPC 2025 Pass@1 from 47.3% to 51.0%, and LCB Pro Pass@1 from 70.7% to 71.6%. Although smaller than the SFT gains, the improvements occur across all three benchmarks and we select step 39 exclusively using the held-out validation set. Run-level uncertainty estimates for these comparisons are provided in Appendix B.3.

We hypothesize that the modest gains reflect both the strong SFT initialization and the challenging optimization setting. With binary-reward GRPO, a problem provides a relative learning signal only when its rollout group contains both successful and unsuccessful solutions (Shao et al., 2024), so RL primarily targets capabilities near the model's frontier. Furthermore, rollouts of up to 255K tokens receive only a terminal execution reward, creating a long-horizon credit-assignment problem. These long trajectories and sparse relative rewards may limit the size and stability of the RL improvement.

Figure 6b compares RL initialized from the base Nano checkpoint and from checkpoints after each SFT epoch. Without SFT, RL improves IOI 2025 Score@1 from 21.7% to 24.9% after 30 steps, demonstrating that executable-reward RL can produce measurable gains directly from the base model. However, RL does not recover the substantially larger gains provided by SFT due to the strength



(a) Performance during GRPO training. Step 39 is selected using held-out validation performance.

(b) IOI 2025 Score@1 during RL when initialized from different SFT checkpoints.

Figure 6: Normalized performance during reinforcement learning of Nano-CC on multiple benchmarks and when initializing RL from different SFT checkpoints.

of the teacher models used: after 30 steps, models initialized from SFT epochs one, two, and three achieve 43.0%, 47.1%, and 48.7%, respectively. Thus, under our training budget, RL can improve performance without prior SFT but does not substitute for the capabilities acquired through supervised fine-tuning, with the strongest final performance obtained from the third-epoch SFT checkpoint.

4.5 Effect of Test-Time Compute

Score@200 measures gains from parallel sampling, while GenCorrect additionally uses evaluator feedback to refine solutions across successive rounds and concentrate improvements from 200 generations into 10 submissions per round. As shown in Figure 7, Nano-CC's mean IOI 2025 score increases from 360.6 after the first round to 468.2 after five rounds, a gain of 107.6 points. Ultra-CC improves from 343.9 to 502.0, a substantially larger gain of 158.1 points. Although Ultra-CC begins below Nano-CC in the first round, it finishes 33.8 points ahead after five rounds. Relative to the official IOI 2025 thresholds, Ultra-CC exceeds the gold threshold after three rounds, while Nano-CC does so after four (International Olympiad in Informatics, 2025).

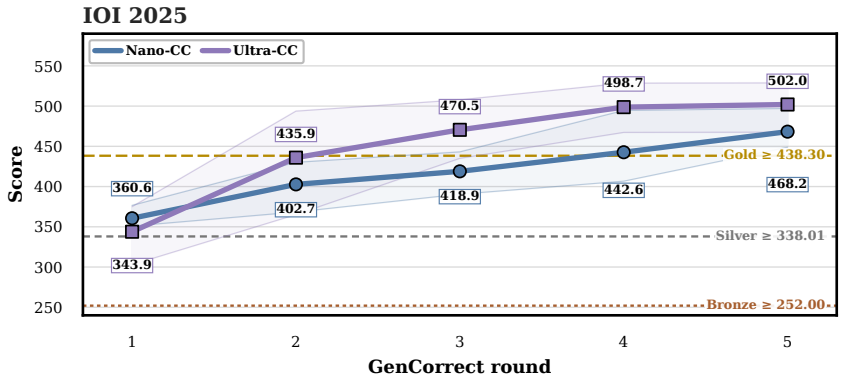


Figure 7: Mean IOI 2025 score across GenCorrect rounds for Nano-CC and Ultra-CC. Dashed lines indicate medal thresholds; shaded regions show the min–max range over five runs per round.

The larger GenCorrect improvement is consistent with a broader difference in how the two models benefit from test-time compute. At Score@1, Ultra-CC exceeds Nano-CC by only 2.2 percentage points, corresponding to approximately 13 raw IOI points. At Score@200, however, Ultra-CC reaches 505 compared with 461 for Nano-CC, widening the gap to 44 points. Thus, Ultra-CC's advantage grows substantially under parallel sampling and then is magnified by the iterative correction from GenCorrect. Together with its larger improvement across GenCorrect rounds, this suggests that Ultra-CC benefits both from stronger pool of candidate solutions from parallel sampling and from more effective use of evaluator feedback.

On ICPC 2025, Figure 8 shows that the mean number of problems solved by Nano-CC increases from 8.6 to 9.4, reaching nine solved problems after two rounds and beginning to plateau after the third. Nine solved problems match the problem count of the fourth-place gold-medal team (International Collegiate Programming Contest, 2025b), though this alone does not establish gold-equivalent placement due to competition time penalties. Ultra-CC starts at 9.0 problems solved and reaches 9.6 after two rounds, maintaining this performance through round five. It therefore reaches nine solved problems one round earlier than Nano-CC and remains ahead throughout the correction process. Both models plateau more quickly than on IOI, suggesting that ICPC's binary feedback provides less information for continued refinement than IOI's subtask-level scores.

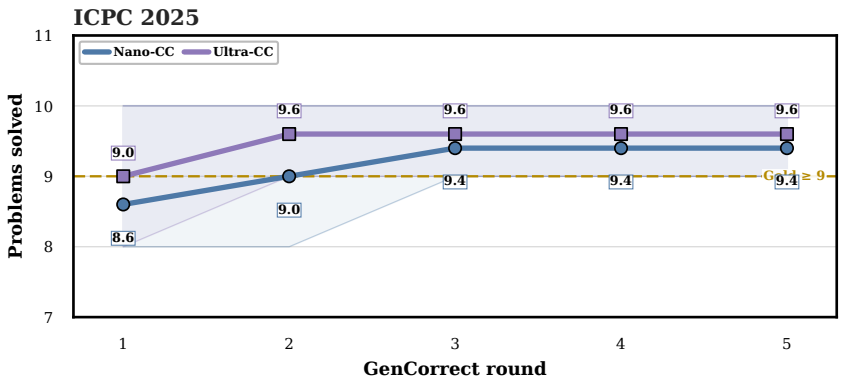


Figure 8: Mean ICPC 2025 score across GenCorrect rounds for Nano-CC and Ultra-CC. Dashed lines indicate medal thresholds; shaded regions show the min–max range over five runs per round

5 IOI 2026

5.1 Competition Setting

We evaluate our system prospectively on the IOI 2026 problem set during the competition, before problems were publicly available. The competition comprised two five-hour sessions over two days, with three problems released simultaneously to us and human contestants through the same portal at each session's start. Submissions used the same graders, tests, and execution time and memory limits. We followed the same time, internet-access, and submission constraints: no internet access, permitted local execution, and at most 50 submissions per problem, limited to one per minute except during the final 15 minutes of each session (IOI 2026 Organizing Committee, 2026). Our system had no competition-imposed compute limit; we reserved 760 NVIDIA GB300 GPUs for both sessions

5.2 Competition-Specific Adaptations

Having completed the general-pipeline experiments, we use IOI 2025 as a development benchmark and make three adaptations to maximize the score of a single live run within the official limits. First, we fine-tune Ultra using GLM-5.2 traces, which provide stronger IOI 2025 performance and shorter outputs than DeepSeek-V4-Flash traces. Second, after four standard GenCorrect rounds, we expand the final-round pool from 200 to 1,000 solutions and rank them using generated tests before selecting 10 submissions. Third, we quantize Ultra to NVFP4, increasing throughput from 199.1 to 736.8 tokens/s/GPU at a 6.6-point reduction in IOI 2025 Score@1. Appendix C provides the full selection procedure, teacher comparison, calibration recipe, ablations and justifications for these modifications.

5.3 Final Results

Table 2 summarizes our IOI 2026 results, Figure 1b visualizes the live result, and Appendix D provides a per-problem score breakdown. Ultra-CC scores 535.4 during the competition window, exceeding the gold-medal threshold by 174.3 points and the top human contestant by 37.1 points (International Olympiad in Informatics, 2026). This score comes from a single prospective run using the competition-specific adaptations described above and conducted under the same time limits, submission platform,

and internet restrictions as the human contestants. To the best of our knowledge, this is the first time an AI system has outscored the highest-scoring human contestant on an IOI problem set.

Table 2: Performance on IOI 2026. The live competition result was obtained under the same time and submission constraints as human contestants. For the General GenCorrect Pipeline, we report the mean score followed by the observed minimum–maximum range across 5 independent runs.

System	Score	Medal Level
Top Human Contestant	498.27	Gold
Gold Medal Threshold	361.12	Gold
Ultra-CC Live Competition Run	**535.40**	**Gold**
Ultra-CC General GenCorrect Pipeline	521.72 (495.0–545.8)	Gold

To contextualize it against our general approach, we additionally conduct independent post-competition runs using the standard five-round GenCorrect pipeline. This pipeline achieves a mean score of 521.72, with an observed range of 495.0–545.8. The live result is 13.68 points above this mean while remaining within the observed range. The general GenCorrect pipeline also achieves a mean score that exceeds both the gold-medal threshold and the highest human score.

6 Related Work

Previous work has established competitive programming as a challenging evaluation setting for language models. AlphaCode combined domain-specific training with large-scale sampling and clustering, and AlphaCode 2 added stronger models and learned reranking (Li et al., 2022; Leblond et al., 2023); LiveCodeBench and LiveCodeBench Pro provide temporally separated, olympiad-level evaluations (Jain et al., 2024; Zheng et al., 2025). Proprietary systems have since reached medal-level performance at IOI and ICPC (OpenAI, 2025a; International Collegiate Programming Contest, 2025a; Lin & Cheng, 2025), while open-weight systems including GenCluster, DeepSeek-V3.2-Speciale, and Nemotron-Cascade 2 report comparable results (Samadi et al., 2026; DeepSeek-AI, 2025; Yang et al., 2026). Our approach builds on synthetic reasoning data and self-critique (Ahmad et al., 2025b;a), execution-based refinement and test-time scaling (Snell et al., 2024; Ridnik et al., 2024; Li et al., 2025), and reinforcement learning with verifiable rewards (Guo et al., 2025; Yu et al., 2025). Because self-correction without external feedback can be unreliable (Huang et al., 2024), GenCorrect uses execution-grounded feedback. We integrate these directions into an end-to-end study isolating the effects of synthetic data, SFT, RL, model scale, and iterative test-time refinement.

7 Conclusion

We presented an end-to-end competitive-programming pipeline combining synthetic data, SFT, RL, and GenCorrect. Our experiments show that SFT provides the largest single-sample gains, RL adds smaller additional improvements, and GenCorrect substantially improves performance through feedback-driven refinement. On IOI 2025, post-training and GenCorrect raise Nano-CC from 130 to 468 points, exceeding the gold threshold with 3B active parameters, while Ultra-CC reaches 502. Guided by these results, we construct a competition-specific Ultra-CC system that scores 535.4/600 prospectively on IOI 2026 under the human time, submission, and internet constraints. This exceeds both the gold threshold and, for the first time the highest human score on an IOI problem set.

Limitations

Our approach requires substantial compute, making this a system-level comparison under matched competition constraints rather than an equal-resource human comparison. Compute limits prevented Ultra-CC RL and exhaustive ablations, and results may not generalize beyond competitive programming. We plan to release the model checkpoint and runnable code. Third-party restrictions prevent releasing the training data, so we document its construction.

AI USE STATEMENT

We used AI tools to help edit and revise the text of the paper and to assist with drafting sections of the manuscript. We also used AI tools to discover related work; each suggested reference was validated by the authors before inclusion. Additionally, we used AI tools to generate the synthetic data described in Section 3.2. The authors take responsibility for the final content of the paper, including its claims, citations, and data.

REFERENCES

Wasi Uddin Ahmad, Somshubra Majumdar, Aleksander Ficek, Sean Narenthiran, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Vahid Noroozi, and Boris Ginsburg. OpenCodeReasoning-II: A Simple Test Time Scaling Approach via Self-Critique, 2025a. URL https://arxiv.org/abs/2507.09075.

Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning: Advancing data distillation for competitive coding, 2025b. URL https://arxiv.org/abs/2504.01943.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024. URL https://arxiv.org/abs/2402.14740.

DeepSeek-AI. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models, 2025. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

GLM-5-Team. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large Language Models Cannot Self-Correct Reasoning Yet, 2024. URL https://arxiv.org/abs/2310.01798.

International Collegiate Programming Contest. OpenAI at the 2025 ICPC World Finals. ICPC World Finals, 2025a. URL https://worldfinals.icpc.global/2025/openai.html.

International Collegiate Programming Contest. 49th ICPC world finals baku: Final scoreboard, 2025b. URL https://wf.icpc.global/scoreboard/2025/finals/index.html. Accessed: 2026-08-28.

International Collegiate Programming Contest. ICPC world finals rules, 2025c. URL https://icpc.global/worldfinals/rules. Accessed: 2026-08-28.

International Olympiad in Informatics. IOI 2025: Results, 2025. URL https://stats.ioinformatics.org/results/2025. Accessed: 2026-08-28.

International Olympiad in Informatics. IOI 2026: Results, 2026. URL https://stats.ioinformatics.org/results/2026. Accessed: 2026-08-28.

IOI 2026 Organizing Committee. IOI 2026 Contest Rules, 2026. URL https://www.ioi2026.uz/contest-rules. Accessed: 2026-08-28.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Rémi Leblond et al. AlphaCode 2 technical report. Technical report, Google DeepMind, 2023. URL https://storage.googleapis.com/deepmind-media/AlphaCode2/AlphaCode2_Tech_Report.pdf.

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E. Gonzalez, and Ion Stoica. S*: Test time scaling for code generation, 2025. URL https://arxiv.org/abs/2502.14382.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, December 2022. ISSN 1095-9203. doi: 10.1126/science.abq1158. URL http://dx.doi.org/10.1126/science.abq1158.

Hanzhao Lin and Heng-Tze Cheng. Gemini Achieves Gold-Medal Level at the International Collegiate Programming Contest World Finals. Google DeepMind, 2025. URL https://deepmind.google/blog/gemini-achieves-gold-medal-level-at-the-international-collegiate-programming-contest-world-finals/.

NVIDIA. NeMo RL: A Scalable and Efficient Post-Training Library. https://github.com/NVIDIA-NeMo/RL, 2025. GitHub repository.

NVIDIA. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2025. URL https://arxiv.org/abs/2512.20848.

NVIDIA. Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2026. URL https://arxiv.org/abs/2606.15007.

NVIDIA Corporation. NVIDIA Model Optimizer. https://github.com/NVIDIA/Model-Optimizer, 2024–2026. GitHub repository.

OpenAI. Competitive programming with large reasoning models, 2025a. URL https://arxiv.org/abs/2502.06807.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025b. URL https://arxiv.org/abs/2508.10925.

Qwen Team. Qwen3.6-35B-A3B, 2026. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B. Model card.

Tal Ridnik, Dedy Kredo, and Itamar Friedman. Code generation with alphacodium: From prompt engineering to flow engineering, 2024. URL https://arxiv.org/abs/2401.08500.

Mehrzad Samadi, Aleksander Ficek, Sean Narenthiran, Siddhartha Jain, Wasi Uddin Ahmad, Somshubra Majumdar, Vahid Noroozi, and Boris Ginsburg. Scaling test-time compute to achieve ioi gold medal with open-weight models, 2026. URL https://arxiv.org/abs/2510.14232.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Zhuolin Yang, Zihan Liu, Yang Chen, Wenliang Dai, Boxin Wang, Sheng-Chieh Lin, Chankyu Lee, Yangyi Chen, Dongfu Jiang, Jiafan He, Renjie Pi, Grace Lam, Nayeon Lee, Alexander Bukharin, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-cascade 2: Post-training llms with cascade rl and multi-domain on-policy distillation, 2026. URL https://arxiv.org/abs/2603.19220.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zihan Zheng, Zerui Cheng, Zeyu Shen, Shang Zhou, Kaiyuan Liu, Hansen He, Dongruixuan Li, Stanley Wei, Hangyi Hao, Jianzhu Yao, Peiyao Sheng, Zixuan Wang, Wenhao Chai, Aleksandra Korolova, Peter Henderson, Sanjeev Arora, Pramod Viswanath, Jingbo Shang, and Saining Xie. Livecodebench pro: How do olympiad medalists judge llms in competitive programming?, 2025. URL https://arxiv.org/abs/2506.11928.

A Data Curation Details

Problem collection. We collect 22,000 problems from 16 regional and international competition families spanning the last two decades, together with problems from online programming platforms. For each competition-year pair, an automated agentic pipeline retrieves the problem statement, time and memory limits, available official test cases, auxiliary files, and reference solutions. These components are packaged into a standardized executable evaluation environment.

Environment validation. We validate each environment by requiring known correct solutions to receive the expected score and known incorrect solutions to fail. As an additional validation step, we use gpt-oss-120b to generate candidate solutions, which are compiled and executed against the packaged test suite. This identifies malformed statements, missing auxiliary files, compiler incompatibilities, and inconsistent test harnesses that may not be exposed by the available reference solutions.

We remove problems whose environments produce inconsistent verdicts, whose reference solutions fail, or whose provided solutions do not compile. For the RL corpus, we additionally remove problems whose evaluation consistently requires more than 300 seconds when executed sequentially.

Evaluation contamination. We exclude all IOI 2025, ICPC 2025, and LiveCodeBench Pro problems from both the SFT and RL corpora and deduplicate the remaining training problems against these evaluations. IOI 2026 requires no retrospective contamination filtering because our system was executed during the live competition before the problems were publicly available.

B Training Details

B.1 Supervised Fine-Tuning

We use DeepSeek-V4-Flash to generate 1.2 million reasoning traces for Nemotron-3-Nano-30B-A3B and 477,642 traces for NVIDIA-Nemotron-3-Ultra-550B-A55B. We categorize problems as easy, medium, or hard and allocate additional generations to more difficult problems.

The SFT mixture also contains self-improvement traces in which the teacher receives a problem and a previously generated solution and is instructed to produce an improved solution. These examples expose the student models to the iterative refinement behavior used by GenCorrect at inference time.

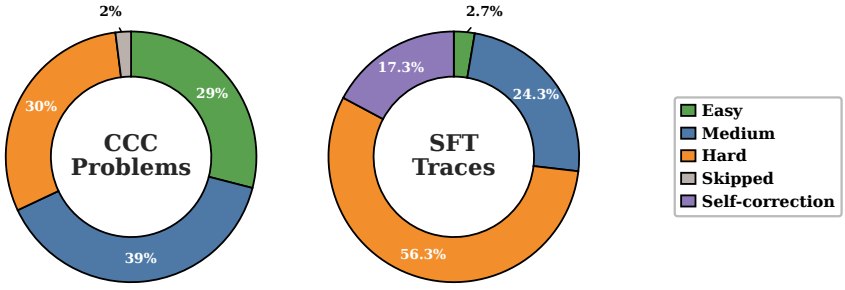


Figure 9: Composition of the curated competitive coding competition (CCC) corpus (left) and SFT traces (right).

Table 3 reports the complete SFT configurations. Nano-CC achieves an average training throughput of 397.2 TFLOPs/s/GPU. Ultra-CC is initialized from its RLVR-teacher checkpoint, which was previously used to distill strong reasoning capabilities into the general-purpose model.

B.2 Reinforcement Learning

Training and validation data. We apply RL only to Nemotron-3-Nano-30B-A3B. From the curated corpus, we initially identify 4,000 problems with reliable executable test cases. We remove problems whose evaluation consistently requires more than 300 seconds when executed sequentially, leaving 3,219 problems.

Table 3: Supervised fine-tuning configurations for Nano-CC and Ultra-CC.

Setting	Nano-CC	Ultra-CC
Initialization	Nemotron-3-Nano-30B-A3B	RLVR-teacher checkpoint
Teacher model	DeepSeek-V4-Flash	DeepSeek-V4-Flash
Training examples	1,200,000	477,642
Epochs	3	1
Optimizer	AdamW	AdamW
Global batch size	64	64
Learning rate	5×10^{-5}	1.5×10^{-5} peak
Learning-rate schedule	Constant	Cosine
Warmup ratio	0	0.1
Maximum packed sequence length	262K	262K
Tensor parallelism	4	8
Context parallelism	4	–
Hardware	64 NVIDIA GB300 GPUs	128 NVIDIA GB300 GPUs

We divide these into 2,847 training and 372 validation problems. The split is performed at the parent-problem level so that subtasks derived from the same original problem cannot appear in both sets. To promote coverage across competitions, we target approximately one validation problem from each competition-year pair.

Rollout and reward configuration. We use Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with a maximum sequence length of 262,144 tokens, including up to 255,144 generated tokens. At each optimization step, we sample 16 rollouts at temperature 1.0 for each of 64 prompts, producing a global batch of 1,024 rollouts. Each rollout generates a C++17 solution that is compiled and executed against the target problem or subtask. We assign a terminal reward of 1 when the solution receives full credit and 0 otherwise; partial scores do not produce intermediate rewards. Rewards are normalized within each rollout group, and relative advantages are computed using a leave-one-out baseline (Ahmadian et al., 2024).

Optimization. We optimize a token-level clipped policy-gradient objective using Adam with a constant learning rate of 3×10^{-6}, zero weight decay, and no reference-policy KL penalty (Yu et al., 2025). Generations truncated at the maximum context length are excluded from the policy loss. The full RL configuration is summarized in Table 4.

Table 4: Reinforcement-learning configuration for Nano-CC.

Setting	Value
Training problems	2,847
Validation problems	372
Algorithm	GRPO
Prompts per step	64
Rollouts per prompt	16
Rollouts per step	1,024
Sampling temperature	1.0
Maximum sequence length	262,144
Maximum generation length	255,144
Reward	Binary full credit
Optimizer	Adam
Learning rate	3×10^{-6}
Learning-rate schedule	Constant
Weight decay	0
Reference-policy KL	None

Checkpoint selection. We evaluate intermediate checkpoints on the held-out validation set. The step 39 checkpoint achieves the highest validation performance and is used for subsequent evaluations.

B.3 Run-Level Uncertainty

Table 5 reports mean performance and standard errors across independent inference runs. These estimates capture sampling variability on the fixed benchmarks, not uncertainty over the broader population of programming problems.

Table 5: Mean $\pm$ standard error and RL gains in percentage points. The p-values use two-sided Welch tests across independent runs.

Benchmark	Runs	SFT epoch 3	RL step 39	Gain	p
IOI 2025	50	46.690 ± 0.677	48.465 ± 0.738	1.775	0.080
ICPC 2025	50	47.333 ± 1.257	51.000 ± 1.283	3.667	0.044
LCB Pro	8	70.655 ± 0.437	71.570 ± 0.646	0.915	0.263

RL increases mean performance on all three benchmarks, although only ICPC reaches the uncorrected 0.05 threshold and none reaches the Bonferroni-adjusted threshold across three comparisons. We therefore interpret RL as providing modest mean gains rather than uniform significant improvements.

B.4 Compute Accounting

Table 6 reports measured GPU-hours for the listed SFT and RL training trajectories and GenCorrect inference runs. GPU-hours are computed as the allocated GPU count multiplied by the counted elapsed time, without hardware normalization. The accounted compute totals 80,397.8 GPU-hours. This is a partial accounting: it excludes teacher trace generation, startup, unsuccessful attempts, standalone evaluations outside the listed jobs, conversions, exploratory experiments and CPU compute. For IOI 2026 live inference, we reserved 760 NVIDIA GB300 GPUs throughout the two five-hour competition sessions, with requests load-balanced across GPU servers.

Nano, General Ultra, and Competition Ultra SFT use 64, 128, and 256 NVIDIA GB300 GPUs, respectively; all RL trajectories use 64 GB300 GPUs. RL timing includes rollouts, optimization, checkpointing, and validation within completed steps. Training compute includes the full checkpoint-selection trajectories. For example, Competition Ultra SFT accounts for 19,236.7 GPU-hours over 2,256 steps, of which 9,186.7 GPU-hours precede the selected checkpoint at iteration 1,068.

Table 6: Accounted compute and coverage. Training coverage gives the number of counted steps. Inference coverage gives the number of preserved run-rounds relative to five runs of five rounds per benchmark–model configuration.

Experiment	Coverage	GPU-hours
SFT		
Nano-CC	18,734 steps	13,656.8
General Ultra-CC	2,500 steps	21,772.2
Competition Ultra-CC	2,256 steps	19,236.7
SFT subtotal		54,665.7
RL by Nano-CC initialization		
No SFT	30 steps	751.7
SFT epoch 1	30 steps	1,494.5
SFT epoch 2	30 steps	1,673.8
SFT epoch 3	50 steps	3,042.7
RL subtotal		6,962.7
GenCorrect inference		
IOI 2025 Nano-CC	25 rounds	913.0
IOI 2025 Ultra-CC	25 rounds	8,291.8
ICPC 2025 Nano-CC	25 rounds	1,023.8
ICPC 2025 Ultra-CC	25 rounds	8,540.8
GenCorrect subtotal		18,769.4
Total accounted		80,397.8

C IOI 2026 COMPETITION-SPECIFIC ADAPTATIONS

Having completed the general-pipeline experiments, we use IOI 2025 as a development benchmark and make several competition-specific adaptations to maximize the score of a single live run within the official time and submission limits.

Ultra SFT with GLM-5.2 data. Our previous results show that adapting the stronger Ultra model with limited SFT can outperform extensive post-training of the smaller Nano model, particularly in later GenCorrect rounds. We therefore focus on fine-tuning Ultra for the live IOI 2026 run. We use SFT rather than RL because SFT provides most of the post-training gains in our experiments, while RL at Ultra scale exceeds our available compute budget. We evaluate GLM-5.2 and DeepSeek-V4-Flash (GLM-5-Team, 2026; DeepSeek-AI, 2026) on IOI 2025 as candidate SFT teachers. GLM-5.2 achieves a higher score with shorter outputs than DeepSeek-V4-Flash, as shown in Table 7. This advantage transfers after SFT, with the GLM-5.2-trained Ultra-CC variant achieving both higher Score@1 and shorter average outputs than the DeepSeek-V4-Flash-trained variant. We therefore select GLM-5.2 training data for the live run.

Table 7: IOI 2025 score and average number of generation tokens for the teacher models and corresponding Ultra-CC variants. Lower output length permits more candidates to be generated within a fixed inference window.

System	IOI 2025 Score@1	Mean Generation Length
GLM-5.2	66.0%	85,927
DeepSeek-V4-Flash	55.3%	120,456
Ultra-CC (GLM-5.2)	59.4%	84,244
Ultra-CC (DeepSeek-V4-Flash)	50.7%	89,626

Expanded final-round selection. For the first four GenCorrect rounds, we generate 200 solutions and submit 10 selected candidates per problem as described in Section 3.4. In the final round, we increase the generation budget to 1,000 solutions while retaining the final 10-submission limit. Inspired by GenCluster (Samadi et al., 2026), we use an execution-based selection procedure to rank this larger candidate pool:

1. We prompt the model to produce 50 problem-specific test-input generators and validators.
2. We execute and filter the generators using the validators until we obtain 100 valid test inputs.
3. We execute every compiled candidate solution on the generated inputs.
4. We prompt the model to produce a scoring script based on the problem's subtask criteria.
5. We use this script to rank the candidates and submit the 10 highest-ranked solutions.

Because we have only a single live attempt, our objective is to maximize the score of that run using all available inference compute. Our IOI 2025 and ICPC 2025 results show that Ultra-CC begins to saturate over successive rounds of the general GenCorrect pipeline, suggesting limited benefit from using the standard 200-generation procedure again in the fifth round. We therefore use our available compute to expand the final-round candidate pool to 1,000 solutions. The execution-based selection procedure, adapted from GenCluster, enables us to select the most promising 10 submissions from this substantially larger pool and thereby make better use of the additional generation budget.

NVFP4 quantization. To increase inference throughput, we apply post-training quantization to Nemotron-3-Ultra (NVIDIA, 2026) using the NVIDIA Model Optimizer NVFP4 recipe (NVIDIA Corporation, 2024–2026). We calibrate the model on 1,000 sequences of 32,768 tokens sampled from our SFT mixture and use the resulting activation statistics to determine the quantization scales. The calibrated model is then exported to NVFP4 for live inference. Table 8 reports the resulting trade-off between IOI 2025 performance and generation throughput. Across the evaluated NVFP4 configurations, IOI 2025 Score@1 remains within a narrow range of 52.7%–53.5%. In the matched BF16 KV-cache and prefix-enabled comparison, setting MTP to 5 nearly doubles throughput, from

345.9 to 698.5 tokens/s/GPU, while reducing Score@1 by only 0.6 percentage points, from 53.5% to 52.9%. For the live run, we select NVFP4 with an FP8 KV cache, prefix caching disabled, and MTP 5, which achieves 52.8% Score@1 at 736.8 tokens/s/GPU. Relative to the BF16 baseline, this configuration sacrifices 6.6 percentage points of Score@1 for a 3.7× increase in throughput, enabling the large candidate batches required by GenCorrect within the competition window.

Table 8: Effect of NVFP4 quantization and runtime configuration on IOI 2025 Score@1 performance and inference throughput. Throughput is normalized by the number of GPUs (tokens/s/GPU).

Precision	KV Cache	Prefix	MTP	IOI 2025 Score@1	Throughput
BF16	—	—	—	59.4%	199.1
NVFP4	FP8	Off	5	52.8%	736.8
NVFP4	BF16	Off	5	52.7%	741.9
NVFP4	BF16	On	5	52.9%	698.5
NVFP4	BF16	On	Off	53.5%	345.9

D Per-Problem IOI 2026 Results

Figure 10 decomposes the live Competition Ultra-CC score by problem and compares it with the mean scores of all official human contestants and of gold medalists. Competition Ultra-CC exceeds the gold-medalist mean on five of the six problems and matches it on Tiling Game. Its largest margins over the gold-medalist mean occur on Monuments and Partition. The six problem scores sum to the live result of 535.4/600 reported in the main text.

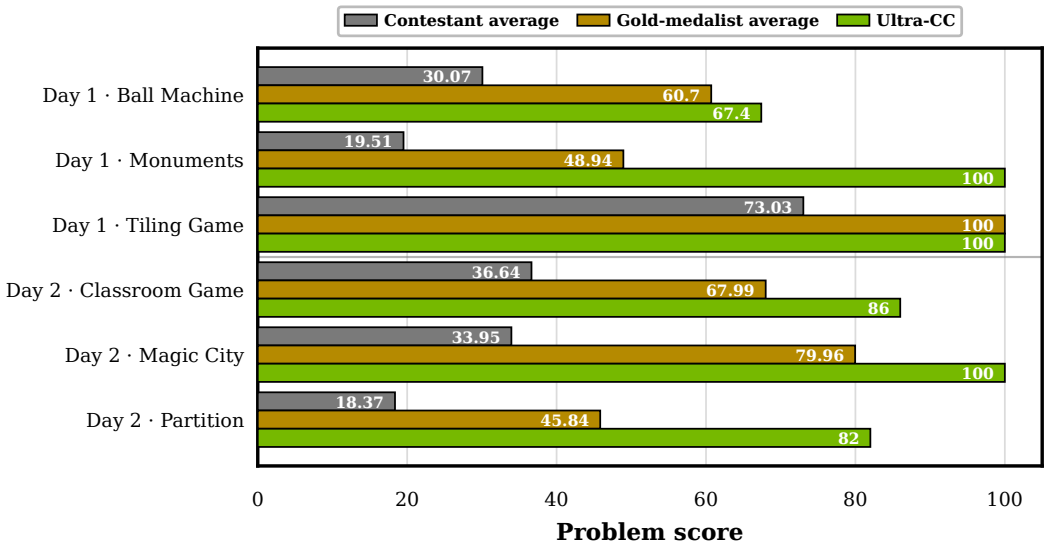


Figure 10: Per-problem IOI 2026 scores for Competition Ultra-CC, compared with the mean scores of all official human contestants and of gold medalists. Competition Ultra-CC matches or exceeds the gold-medalist mean on every problem and scores 535.4/600 overall. Human averages are computed from the official final scoreboard.

E Additional GenCorrect Details

E.1 Algorithm Details

Candidate filtering. We extract and normalize the C++ program from each model response before computing program features or similarities. Before clustering, we retain candidates that compile,

contain nonempty normalized code with at least 40 tokens, and have a nonnegative structural-feature score. If no candidates satisfy all requirements, we progressively relax them until the candidate pool is nonempty.

Similarity and clustering. The similarity function $\mathrm{sim}(c, z)$ in Equation 1 compares token shingles extracted from the normalized programs. We initialize the center set with the highest-ranked candidate under the score-blind heuristic $Q(c)$. Subsequent centers are selected using Equation 1, with ties broken by $Q(c)$. After selecting $K = 10$ centers, each remaining candidate is assigned to its most similar center. Equal-similarity ties are assigned to the smaller cluster, with any remaining ties resolved by center-selection order.

Score-blind ranking. The heuristic used for center initialization, tie-breaking, and representative selection is

$$Q(c) = \big(\,\mathrm{Compiles}(c), \mathrm{FeatureScore}(c), \lfloor \mathrm{CodeLength}(c)/800 \rfloor, \\ \mathrm{ExactFrequency}(c), \mathrm{CodeLength}(c), -\mathrm{GeneratedTokens}(c), -\mathrm{RunId}(c)\big), \tag{3}$$

with entries compared lexicographically and larger values preferred. Here, $\mathrm{ExactFrequency}(c)$ is the number of candidates with the same normalized code, $\mathrm{GeneratedTokens}(c)$ is the length of the original model output, and $\mathrm{RunId}(c)$ is its deterministic generation index. The feature score rewards indicators of complete C++ programs, including headers, a main function, input/output operations, and a return statement, while penalizing unfinished placeholders. Neither $Q(c)$ nor the clustering procedure uses competition scores.

Representative selection. We select the highest-ranked candidate under $Q(c)$ from each cluster. If this produces fewer than 10 representatives, the remaining submission slots are filled with the highest-ranked unselected candidates. Thus, cluster membership promotes diversity, while $Q(c)$ selects a locally well-formed representative from each cluster. All filtering, clustering, and representative selection for a round are completed before its evaluator feedback is observed.

Feedback and carry-forward. The accumulated subtask scores $A_r(t)$ in Equation 2 form an internal feedback state used to guide subsequent generations; they do not alter the evaluator's official scoring procedure.

After each round, the highest-total-scoring submissions from that round becomes the reference solution. Ties are resolved using the score-blind heuristic. We then select three complementary references from the current submissions and solutions retained from previous rounds:

1. The strongest overall candidate across the available solutions.
2. A candidate addressing the largest remaining subtask gap.
3. A candidate with strong aggregate coverage of the remaining gaps.

Subtask gaps are determined from the accumulated feedback state A_r. Similarity and $Q(c)$ are used as tie-breakers to avoid redundant references and prefer locally well-formed programs. The next round is conditioned on the problem statement, the accumulated subtask feedback and three reference solutions.

E.2 Ablation Studies

We compare GenCorrect with two non-adaptive selection baselines on IOI 2025, using pools of 1,000 first-round generations per problem. **Random 50** randomly selects 50 candidates. **Diversity 50** randomly partitions each pool into five batches of 200 and applies the GenCorrect selection procedure to choose 10 candidates per batch. Neither baseline conditions generation on evaluator feedback. Both use 50 submissions per problem, matching GenCorrect's submission limit and maximum generation budget.

Table 9: Mean IOI 2025 scores out of 600. Selection baselines average 10,000 resampling trials over fixed candidate pools; full GenCorrect averages five independent inference runs. Resampling captures selection variability conditional on these pools, rather than independent generation variability.

Method	Nano-CC	Ultra-CC
Random 50	393.39	419.63
Diversity 50	441.78	419.20
Full GenCorrect	**468.20**	**502.00**

Full GenCorrect exceeds the non-adaptive diversity baseline by 26.4 points for Nano-CC and 82.8 points for Ultra-CC, consistent with benefits from feedback-driven generation beyond additional sampling. The selection pipeline's benefit varies across models: Diversity 50 improves Nano-CC by 48.4 points over random selection, while Ultra-CC's aggregate score is nearly unchanged. For Ultra-CC, a 20.33-point loss on *Souvenirs* offsets a 19.90-point aggregate gain across the remaining five problems.

E.3 Test-Time Compute Prompts

First Gencorrect Round

```
system: |-
  Your response must use the following format:

  Explanation: <your explanation for the final answer>
  Confidence: <your confidence score between 0% and 100%>
  Answer:
  ```cpp
  <your complete C++20 implementation>
  ```

  The Answer must contain exactly one complete C++20 code block,
  and that
  code block must be the final content in your response. Do not
  write
  anything after its closing fence.

user: |-
   You are an expert competitive programmer. You will be given a
   problem
   statement, test case constraints, and example test inputs and
   outputs.
   Please reason step by step about the solution, then provide a
   complete
   implementation in C++20.

   You should correctly implement the routine(s) described in
   the
   Implementation Details, without reading or writing anything
   directly
   from stdin or to stdout, as input and output are passed
   through the
   implemented routines.

   Assume your code will be run on the OFFICIAL grader, and do
   not add a
   main function, a sample grader, or any other functionality
   unless it
```

```
  has been explicitly requested.

  If multiple subtasks are listed, first choose one subtask to
  target,
  state that choice briefly in your reasoning, and focus on an
  approach
  that is sufficient for that chosen subtask rather than trying
  to solve
  every listed subtask.

  Put your final solution within a single code block:

  ```cpp
  <your code here>
  ```

  {problem}
```

Next Gencorrect Round

```
system: |-
  Your response must use the following format:

  Explanation: <your explanation for the final answer>
  Confidence: <your confidence score between 0% and 100%>
  Answer:
  ```cpp
  <your complete C++20 implementation>
  ````

  The Answer must contain exactly one complete C++20 code block,
  and that
  code block must be the final content in your response. Do not
  write
  anything after its closing fence.

user: |-
  You are an expert competitive programmer.

  You will be given:

  * the problem statement
  * multiple candidate solutions from previous evaluated
  generations
  * each candidate solution's subtask scores
  * the best achieved subtask scores so far across rounds
  * the maximum subtask scores

  The candidate solutions are provided only as inspiration.
  They may solve different subtasks, may contain bugs, and may
  be
  incomplete. There is no main candidate solution; treat all
  candidate
  solutions as peer references. You may discard them completely
  and
  write a new solution from scratch.

  Choose one target subtask using the best achieved scores so
  far:
```

```
* a subtask is eligible only if
  achieved_subtask_scores[subtask] <
  max_subtask_scores[subtask]
* choose exactly one eligible target subtask and focus on
solving
  that target better
* prefer the eligible target with the largest remaining score
gap,
  unless another eligible target is clearly easier to improve
* do not spend effort on subtasks that are already fully
achieved

Use the candidate solutions to identify useful ideas, edge
cases, or
implementation patterns. Do not simply concatenate candidate
solutions.
Do not spend time reconciling differences between candidate
solution
scores and best achieved scores so far. Do not choose a target
based on
any single candidate's scores alone.

If all candidate solutions mainly solve already-achieved
subtasks,
ignore them and design a fresh approach for the chosen
unsolved target.

In your reasoning:

1. identify which subtasks are still globally unsolved or
partially
   solved using achieved_subtask_scores and max_subtask_scores
2. choose exactly one target subtask
3. briefly identify any useful ideas from the candidate
solutions,
   if any
4. explain an approach sufficient for the chosen target
5. then provide a complete C++20 implementation

You should correctly implement the routine(s) described in
Implementation Details, without reading or writing anything
directly
from stdin or to stdout, as input and output are passed
through the
implemented routines.

Assume your code will be run on the OFFICIAL grader, and do
not add a
main function, a sample grader, or any other functionality
unless it
has been explicitly requested.

Put your final solution within a single code block:

```cpp
<your code here>
```
```

```
## Problem
{problem}

## Candidate Solutions From Previous Evaluated Generations
{candidate_solutions}

## Best Achieved Subtask Scores So Far
{achieved_subtask_scores}

## Maximum Subtask Scores
{max_subtask_scores}
```