SCALING TEST-TIME COMPUTE TO ACHIEVE IOI GOLD MEDAL WITH OPEN-WEIGHT MODELS

Mehrzad Samadi, Aleksander Ficek, Sean Narenthiran, Siddhartha Jain
Wasi Uddin Ahmad, Somshubra Majumdar, Vahid Noroozi, Boris Ginsburg
NVIDIA
Santa Clara, CA 95051, USA
{msamadi,aficek,snarenthiran,siddjain
wasiuddina,smajumdar,vnoroozi,bginsburg}@nvidia.com

ABSTRACT

Competitive programming has become a rigorous benchmark for evaluating the reasoning and problem-solving capabilities of large language models (LLMs). The International Olympiad in Informatics (IOI) stands out as one of the most prestigious annual competitions in competitive programming and has become a key benchmark for comparing human and AI-level programming ability. While several proprietary models have been claimed to achieve gold medal-level performance at the IOI, often with undisclosed methods, achieving comparable results with open-weight models remains a significant challenge. In this paper, we present GENCLUSTER, a scalable and reproducible test-time compute framework that attains IOI gold-level performance using open-weight models. It combines large-scale generation, behavioral clustering, ranking, and a round-robin submission strategy to efficiently explore diverse solution spaces under limited validation budgets. Our experiments show that the performance of our proposed approach scales consistently with available compute, narrowing the gap between open and closed systems. Notably, we will show that GENCLUSTER can achieve a gold medal at IOI 2025 for the first time with an open-weight model `gpt-oss-120b`, setting a new benchmark for transparent and reproducible evaluation of reasoning in LLMs.

1 INTRODUCTION

As large language models (LLMs) have advanced in solving coding problems, traditional benchmarks such as HumanEval (Chen et al., 2021a) and MBPP (Austin et al., 2021) have reached saturation. This has motivated a shift toward more challenging benchmarks like LiveCodeBench (Jain et al., 2025) and Codeforces (Penedo et al., 2025), which feature significantly more complex competitive programming problems. Furthermore, considering the challenging aspects of understanding and solving the competitive programming questions, they have been shown to be effective for both training and evaluating the reasoning capabilities of LLMs (DeepSeek-AI et al., 2025). Recent progress on these benchmarks has been driven by both improved training strategies and test-time compute methods. For example, AlphaCode2 (Leblond et al., 2023) achieved performance exceeding that of 85% of Codeforces participants by generating over one million candidate programs per problem and then applying a pipeline of filtering, clustering, and ranking to select 10 solutions for final submissions.

Analogous to the International Mathematical Olympiad (IMO)[1], the International Olympiad in Informatics (IOI)[2] and International Collegiate Programming Contest (ICPC)[3] are widely regarded as the pinnacle of algorithmic programming competitions. These competitions serve as an important milestone in assessing AI competency on competitive programming and general reasoning capabilities as a whole. OpenAI reported achieving a gold medal at IOI 2024 with their *o1-ioi* and *o3*

[1] https://www.imo-official.org/
[2] https://ioinformatics.org/
[3] https://icpc.global/

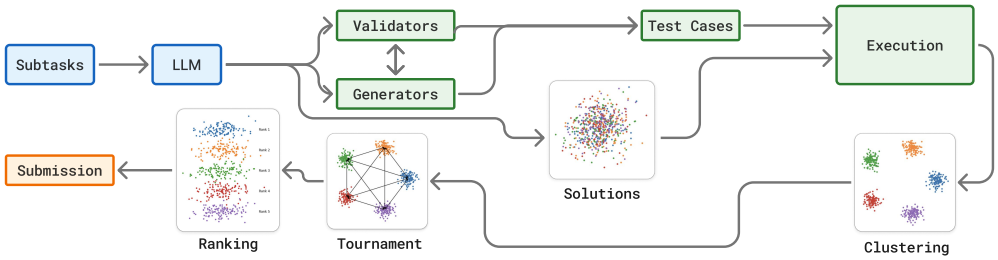

Figure 1: The overall pipeline of GENCLUSTER for a single subtask, a process to be repeated for every subtask in the IOI benchmark.

models (OpenAI et al., 2025), using specialized test-time compute approaches. *o1-ioi* is a dedicated version of their *o1* model fine-tuned for competitive programming and leveraging external tools. Additionally, OpenAI claimed a gold medal and a 6th-place human-equivalent ranking at IOI 2025 with their latest models (Decoder, 2025), though the details of these systems have not yet been fully disclosed. Recently, both OpenAI and Google DeepMind (Lin & Cheng, 2025) have reported achieving gold-level performance at ICPC 2025 using proprietary methods.

Despite their impressive results, the techniques and models used to achieve gold-level performance at such competitions are rarely disclosed. In contrast, open-weight models such as `DeepSeek-R1-0528` (DeepSeek-AI et al., 2025) and `Qwen3-235B-A22B` (Team, 2025) have achieved increasingly competitive results on LiveCodeBench and Codeforces, but still lag behind proprietary models. In this paper, we propose GENCLUSTER, a scalable test-time compute approach that substantially improves the performance of open-weight LLMs on IOI problems. Our approach first generates a large pool of candidate solutions for each problem, which are then refined through a pipeline of filtering, behavioral clustering, and ranking with tournament. Finally, a round-robin submission strategy selects the best candidates while adhering to the same submission constraints imposed on human contestants at IOI (Figure 1). In our experiments, we demonstrate that GENCLUSTER enables open-weight model `gpt-oss-120b` (OpenAI, 2025) to achieve gold-level performance at IOI 2025.

In our experiments, we evaluated and analyzed some of the best open-weight models publicly available including `DeepSeek-R1-0528` and `Qwen3-235B-A22B`, and we show that `gpt-oss-120b` is significantly superior on IOI problems, providing better scalability in terms of the number of generations. Furthermore, we demonstrate that GENCLUSTER improves scores with increased compute and larger generation budgets which is an indicator of scalability of our proposed test-time compute approach.

In summary, our contributions are as follows:

1. We propose GENCLUSTER, a scalable test-time compute approach to improve the competitive programming performance of LLMs by generating many solutions in parallel and then using behavioral clustering and tournaments to select the best candidates.
2. We demonstrate, for the first time, that **gold-level performance** at IOI 2025 can be achieved using only **open-weight models** combined with a transparent and reproducible **test-time compute** strategy.
3. We show that scaling GENCLUSTER consistently improves scores, providing a promising path to surpass gold-level performance.

2 BACKGROUND

2.1 RELATED WORK

This paper investigates scalable test-time compute (TTC) for large language models (LLMs) applied to coding problems, particularly those with a limited verification budget. Test-time compute refers to allocating additional computational resources during inference to improve output quality. This can be achieved through extended chain-of-thought reasoning (DeepSeek-AI et al., 2025; Li et al., 2025;

Wei et al., 2022; Zhang et al., 2025b), generative verifiers for scoring and selection (Fu et al., 2025; Mahan et al., 2024; Zhang et al., 2025a), or best-of-N strategies (Chen et al., 2025; Toshniwal et al., 2025; Brown et al., 2024; Li et al., 2022; Leblond et al., 2023; OpenAI et al., 2025). While there exist some limited works that have leveraged large scale test-time compute pipelines to achieve state-of-the-art results on coding problems (Li et al., 2022; Leblond et al., 2023; OpenAI et al., 2025), most works have explored smaller compute scales (Chen et al., 2023; Le et al., 2022; Zhang et al., 2023) or less challenging benchmarks (Chen et al., 2021b; Inala et al., 2022; Zhao et al., 2025; To et al., 2024). These studies have not yet explored their methods at the scale required for competitive programming, where problems are highly complex and only a small fraction of the generated candidates are correct. In the following, we highlight works that specifically explore scalable approaches to competitive programming and coding competitions.

AlphaCode (Li et al., 2022) followed by its successor AlphaCode 2 (Leblond et al., 2023) were the first to demonstrate that large-scale test-time compute can achieve competitive performance on Codeforces contests. AlphaCode achieved results comparable to the top 54% of participants by generating around 100K solutions per problem and selecting the top 10 via clustering and ranking. AlphaCode 2 improved this further, surpassing 85% of the participants by scaling up to 1M solutions and refining the selection mechanism. Both used proprietary models and their evaluations were performed on the Codeforces benchmark.

OpenAI then showed gold-medal performance at the International Olympiad in Informatics (IOI) by scaling test-time compute (OpenAI et al., 2025). They fine-tuned a variant of o1 (o1-ioi) for IOI and achieved gold by generating up to 10K solutions per problem and filtering with clustering and heuristic-based selection. They also demonstrated that their o3 model can achieve similar with only 1K solutions per problem and a simpler selection procedure. While they reported achieving gold again at IOI 2025 and a 6th overall ranking, details about the models and selection procedures were not released publicly. Recently, both OpenAI (Decoder, 2025) and Google DeepMind (Lin & Cheng, 2025) claimed gold-medal at ICPC 2025 with limited disclosed details. All these works rely on closed models while they provide little to no detail of the test-time compute approach used. Our work is the first to show that achieving the gold medal at competitions like IOI is feasible with open-weight models and we provide the approach to achieve this.

2.2 IOI Competition and Benchmark

At the IOI 2025 competition, contestants are presented with six problems, each divided into 2-12 subtasks. Each subtask evaluates specific aspects of the solution under strict time and memory constraints. Among the six problems, five require executable code submissions, while one accepts either executable code or precomputed output. Scoring is determined at the subtask level, with the contestant's final score for a subtask being the maximum achieved across all submissions. Subtask scores are then aggregated to yield a problem score, with each problem worth up to 100 points. Contestants may refine and resubmit their solutions, but each problem allows a maximum of 50 submissions, making submission strategy an essential factor in performance.

Our evaluation dataset is organized at the subtask level rather than the problem level. Each of the six problems is decomposed into its constituent subtasks following the official scoring guidelines. For every subtask, we created a standalone instance by removing all information related to other subtasks, isolating it as an independent evaluation unit. The resulting benchmark, IOI-2025, follows the structure of the publicly available IOI-2024 benchmark (Hugging Face, 2025). For scoring, we execute the solutions using the grader programs provided by the IOI organization with the same memory and time limits.

3 Proposed Approach

In this section, we propose GenCluster, a scalable test-time compute approach that improves the performance of LLMs on the IOI competition. GenCluster consists of four stages: (1) parallel generation, (2) behavioral clustering, (3) ranking with tournament, and (4) submission. For each subtask, we first generate a large set of candidate programs in parallel. Given the constraint of 50 submissions per problem, we must efficiently select the most promising candidates from the large pool of solutions. We cluster solutions based on behavioral similarity under several inputs, rank

clusters using an LLM-based tournament, and finally employ a round-robin submission policy to maximize scores under submission constraints. The overall pipeline can be seen in Figure 1.

3.1 Parallel Candidates Generation

We first generate K candidate solutions for each subtask using the prompt in Figure 8. Since generations are independent, they can be executed in parallel, making our approach scalable. Generations that result in no parsed code or code that does not compile are filtered.

3.2 Behavioral Clustering

In this stage, we group candidate solutions based on similar behavior to reduce the number of candidates and make ranking more effective. Inspired by OpenAI et al. (2025), for each subtask, we first instructed the LLM to produce multiple distinct programs that generate randomized test inputs consistent with the subtask's specifications (prompt shown in Figure 9). To ensure the correctness of test inputs, we also asked the model to produce multiple independent validators (prompt shown in Figure 10). Each validator program checks whether a given input meets the specific constraints of a subtask. Once we have created all test generators and validators, we iterate through the generators. For each one, we produce a test input and evaluate it using all validators. If at least 75% of the validators approve the input, we keep it as a valid test case. This process is repeated over all the generators until we collect the required number of accepted test cases.

All candidate solutions for a subtask are executed on all the test cases, and then clustered based on their outputs. Solutions that produce exactly the same outputs are grouped into the same cluster. For efficiency, we compute hash values of the outputs to accelerate clustering. If solutions in a cluster produce an empty output for any of the test inputs due to any reason such as runtime errors, that cluster is removed completely.

3.3 Ranking with Tournament

Clustering the candidates results in a large number of groups of similarly behaving solutions. We therefore require a ranking mechanism to prioritize the most promising clusters for submission. There has been prior work on selecting the best solution among multiple generated candidates; however, most existing approaches are not well suited to our setting or have not been evaluated under comparable conditions (Liu et al., 2025; Toshniwal et al., 2025; Brown et al., 2024). Specifically, (1) they do not scale to the large number of solutions required in our experiments, (2) they are primarily designed and evaluated on tasks such as mathematics, where majority voting can trivially identify the correct answer, or (3) they focus on selection rather than ranking of candidate solutions. In the following section, we present our proposed approach, and report the results of some of the alternative selection strategies in Section 4.3.

We propose to hold a tournament between clusters inspired by a partial round-robin tournament. Each cluster plays G_n times with randomly selected clusters. We select the solution with the longest thinking length as the representative solution from each cluster. The representatives of two clusters are then given to the LLM which is prompted to select the better solution in a pairwise comparison (prompt shown in Figure 11). Each play produces a single winner. After the tournament, clusters are then ranked based on the number of wins of their representative candidate. To mitigate recency bias (Shi et al., 2024) in the model's judgments, we randomize the presentation order of the two solutions; each solution appears first in approximately half of its matches.

3.4 Submission Strategy

We submit up to 50 solutions per problem (the maximum permitted by IOI) using a round-robin strategy. We begin with the final subtask of each problem, which is typically the most difficult. For each subtask, we iterate through its clusters in ranked order, selecting one solution from each cluster and submitting it for scoring. Within each cluster, individual solutions are ranked by their reasoning length, which we use as a proxy for correctness likelihood. We continue submitting one solution at a time from the top-ranked clusters, cycling across clusters in a round-robin fashion and rotating

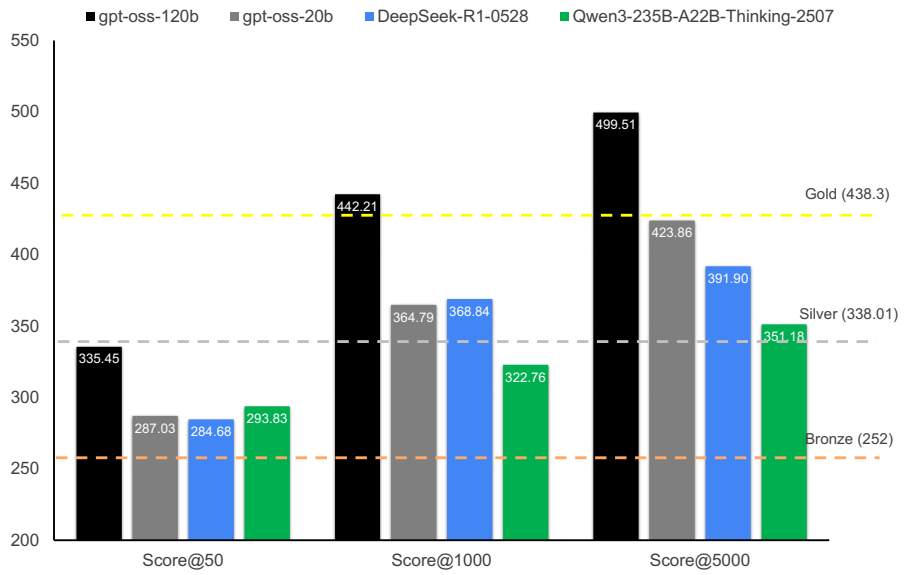

Figure 2: Final scores of different models on IOI 2025 when generating K solutions per subtask, assuming unlimited submissions. Medal thresholds are indicated on the chart, and the maximum achievable score is 600.

through solutions within each cluster. Once a subtask is solved (i.e., its maximum score is achieved), we skip its remaining clusters and proceed to the next subtask following OpenAI et al. (2025).

4 Experiments

In our experiments, all solutions, test generators, and validators are generated in C++ since it is the most common language for IOI competition. For each subtask, we produced 100 test generator functions and 100 validators to obtain 100 validated test cases per subtask. We study the impact of varying the number of test cases per subtask in Section 4.4. In all experiments, we used 10 competitions per cluster, $G_n = 10$, and studied the impact of this parameter in Section 4.5. We used maximum generation lengths of 120K, 120K, 64K, and 80K tokens for models `gpt-oss-120b`, `gpt-oss-20b`, `DeepSeek-R1-0528`, and `Qwen3-235B-A22B`, respectively. The first two values correspond to each model's architectural limit, while 80K was recommended by Qwen developers for complex tasks such as mathematical and programming competition problems (Qwen Team, 2025). The effect of the generation length is studied in Section 4.7. To calculate the scores for each solution, we used the graders officially provided by IOI with the exact same time and memory constraints (International Olympiad in Informatics, 2025b).

4.1 Performance Comparison between Models

To identify the most competitive open-weight models for the IOI benchmark, we evaluated four top-performing available candidate models: `gpt-oss-120b`, `gpt-oss-20b`, `DeepSeek-R1-0528`, and `Qwen3-235B-A22B-Thinking`. For each subtask, we generated up to 5000 candidate solutions to estimate the maximum potential performance of each model without submission constraints. The resulting scores on the IOI 2025 benchmark, reported using the `Score@K` metric, are shown in Figure 2 for $K \in \{50, 1000, 5000\}$. The results are averaged over 20 runs by randomly sampling from the pool of 5000 generated solutions, except for $K = 5000$, which corresponds to a single run.

As illustrated, `gpt-oss-120b` achieves the highest score among all models by a significant margin and is the only one with the potential to reach gold medal performance with up to 5000 generations. Moreover, the `gpt-oss` models exhibit stronger gains as the number of generations increases, suggesting that they scale more effectively with test-time compute. In contrast, while

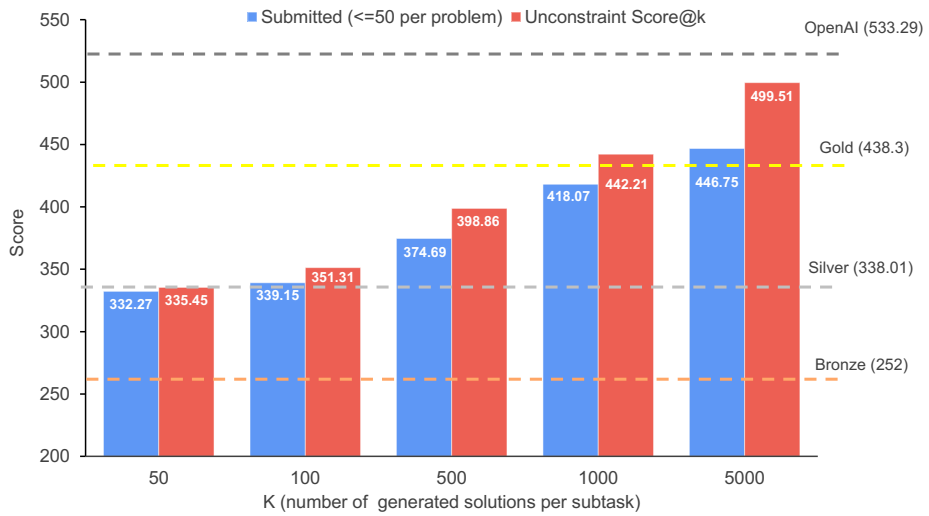

Figure 3: Performance of `gpt-oss-120b` on IOI 2025 with and without the 50-submission limit, varying generation counts. Results are averaged over five runs (except K=5000, single run). Medal thresholds are indicated on the chart, and the score reported by OpenAI is shown for comparison.

`Qwen3-235B-A22B-Thinking` outperforms `gpt-oss-20b` and `DeepSeek-R1-0528` at smaller generation budgets, its performance scales less favorably.

4.2 Evaluating GenCluster under Submission Constraints

We next study the scalability and effectiveness of GenCluster with `gpt-oss-120b`, the best open-weight model for IOI in our experiments. As shown in Section 4.1, it is the only model that can potentially win a gold medal with up to 5000 generations. In this section, we use GenCluster to follow the submission restrictions of IOI akin to human contestants. Figure 3 shows the scores when using different generation sizes, $K \in \{50, 100, 500, 1000, 5000\}$. We report two outcomes: (1) the **Submitted Score**, which respects the 50-submission limit by using GenCluster, and (2) the **Unconstrained Score@K**, which assumes no submission cap and reports the best score among the K solutions. This comparison quantifies the gap between the practical effectiveness of GenCluster and its theoretical upper bound.

As demonstrated, we achieve a gold medal with the GenCluster approach and 5000 generations per subtask while following the constraints of IOI competition. This is the first work to achieve a gold medal with open-weight models at an IOI competition. Although this score remains below the level claimed by OpenAI for IOI 2025, our transparent methodology provides a reproducible baseline for narrowing the gap between the open and closed models on competitive programming tasks. It is also worth noting that OpenAI has not disclosed the compute budget or methodology used to obtain their reported results.

Our results reveal a clear scaling trend: larger candidate pools improve both constrained and unconstrained scores. Even under the 50-submission limit, the submitted score steadily improves from 332.27 for $K = 50$ to 446.75 for $K = 5000$. These results demonstrate the benefits of scaling test-time compute in conjunction with GenCluster. At smaller K values, the gap between the two is modest, but as K increases, the gap gets larger which shows the challenge of selecting the best solutions when we have a large number of candidates. The remaining gap between constrained and unconstrained performance highlights the need for more effective ranking and selection strategies to fully realize the available potential.

4.3 COMPARISON WITH OTHER TEST-TIME COMPUTE STRATEGIES

We compare GENCLUSTER with several alternative test-time compute selection strategies using 5000 generated solutions per subtask from `gpt-oss-120b`, evaluated under standard IOI submission constraints. We also include several GENCLUSTER variants as ablations. Results are reported as final scores and summarized in Table 1. The following approaches are considered for comparison:

- **Random**: Selects solutions randomly from all available generated candidates.
- **Longest**: Selects solutions by longest reasoning trace determined by number of tokens.
- **Cluster-Size**: Ranks clusters by size, assuming larger clusters imply higher correctness likelihood.
- **Cluster-Majority**: Ranks clusters based on aggregated majority vote over outputs from generated solutions. Each unique output across the generated test inputs is treated as a vote. For a given test case, we count how many generated solutions produced each particular output. Within each cluster, we then sum the counts of all outputs produced by the generated solutions in that cluster. This prioritizes clusters which collectively produce outputs that are found from the majority of all solutions.
- **GENCLUSTER (Random-Rep)**: A variant of GENCLUSTER where each cluster's representative solution is chosen randomly.
- **GENCLUSTER (ScoreBased)**: A variant of GENCLUSTER where clusters are ranked based on their average score during competitions instead of the number of wins. During comparison, we prompt the model to provide scores for each solution as a number between 0 to 10 (Figure 11).
- **GENCLUSTER**: Our proposed approach which uses the solution with the longest thinking trace as the representative of each cluster and clusters are ranked based on the number of their wins in the tournament.

The results clearly show that GENCLUSTER is the most effective method among those evaluated, outperforming all alternatives by a substantial margin. Simple approaches such as selecting solutions with the longest reasoning trace or random selection perform considerably worse. Heuristic-based methods, such as ranking clusters by size (referred to as Cluster-size), are suboptimal and yield scores comparable to random selection, a phenomenon magnified by the difficulty of problems in IOI compared to other benchmarks with easier questions. On such challenging problems where models rarely generate correct solutions, approaches based on majority vote may not be very effective (Cobbe et al., 2021; Yang et al., 2024; Wu et al., 2025).

To validate the selection strategy used in GENCLUSTER for choosing cluster representatives, we compare it against a simple baseline that selects a random solution from each cluster (referred to as **GENCLUSTER (Random-Rep)**). The results show that choosing the solution with the longest reasoning trace performs better than random selection, indicating a positive correlation between reasoning length and correctness within a cluster. Furthermore, ranking clusters by the number of wins provides a slight improvement over using average scores (referred to as **GENCLUSTER (Score-Based)**).

Table 1: Performance comparison of different test-time compute strategies with GENCLUSTER. The results show the final scores achieved by each method after submitting up to 50 solutions per problem. **Random** and **Longest** are non-clustering baselines, while all other methods use clustering. "Longest" refers to the solution with the longest reasoning trace.

Method	Cluster Representative	Cluster Ranking	Solution Selection	Score
Random	–	–	Random	300.10
Longest	–	–	Longest	277.36
Cluster-Size	–	Size	Longest	299.87
Cluster-Majority	–	Majority Voting	Longest	314.22
GENCLUSTER (Random-Rep)	Random	Number of Wins	Longest	406.49
GENCLUSTER (Score-Based)	Longest	Average Score	Longest	441.11
GENCLUSTER	Longest	Number of Wins	Longest	446.75

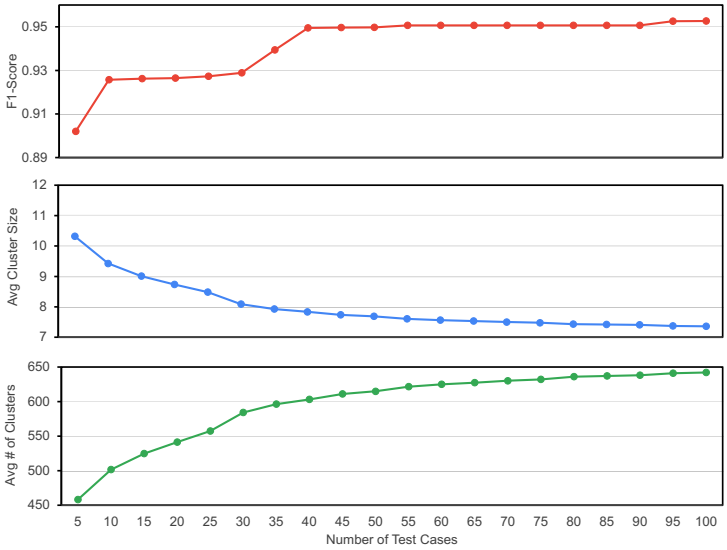

Figure 4: Shown are the cluster purity (F1-score), average cluster size, and average number of clusters for different numbers of test cases. Results are reported for $K = 5000$ using `gpt-oss-120b`.

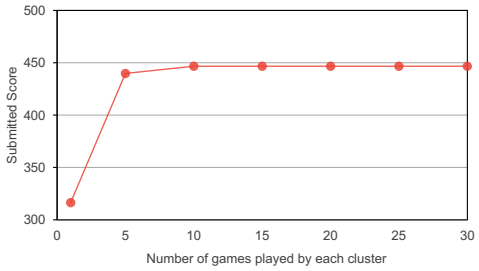

Figure 5: Effect of the number of games per cluster on final score.

Figure 6: Quality of cluster ranking measured by top-K inclusion.

4.4 Impact of the Test Set Size on Clustering

To understand how the number of test cases affects our clustering mechanism, we analyzed the impact of test set size on three key metrics: cluster purity, average cluster size, and the average number of clusters. The results are shown in Figure 4.

We evaluate cluster purity using the F1-score, which measures how effectively our clustering separates high-quality solutions from others. A "good" solution is defined as one that achieves the maximum achieved grade for its subtask, while other solutions are considered as "bad". The ideal clustering should result in clusters where all the solutions are either all good or all bad. Each solution is assigned a binary label indicating whether it achieves the maximum score. For each cluster, the majority of solutions would determine the prediction of the cluster. The F1-score is then computed by comparing the cluster-level predictions to the ground-truth labels of all solutions, representing the harmonic mean of precision and recall. A higher F1-score indicates purer clusters.

As the number of test cases increases, cluster purity improves, reaching a high F1-score that demonstrates the effectiveness of our clustering approach. However, as more tests enhance cluster purity, they also increase the average number of clusters while reducing the average cluster size. This introduces a key challenge: although additional test cases improve our ability to distinguish correct from incorrect solutions, they also produce a larger number of distinct clusters, making it more difficult to identify the most promising ones under submission constraints.

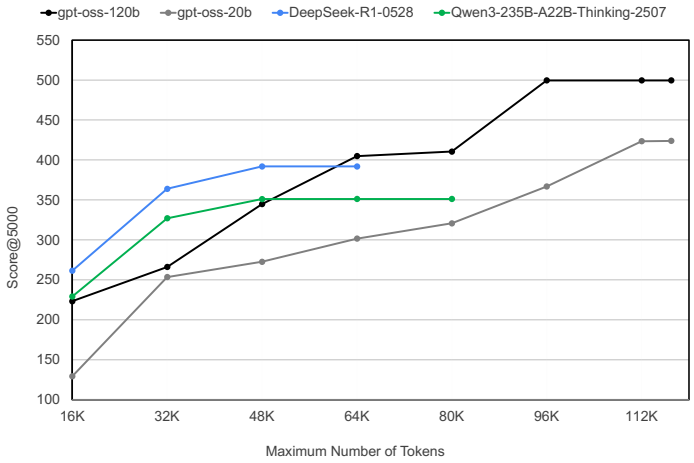

Figure 7: Score@K for different maximum number of tokens in generation with different models.

4.5 Impact of the Number of Games in Tournament

In this experiment, we examine the effect of the parameter G_n (the number of tournament rounds per cluster representative) on the final score under submission constraints. Results for `gpt-oss-120b` with 5000 generations are shown in Figure 5. As shown, increasing the number of games improves the final score but largely saturates after 10 rounds. The gap between one and ten or more games indicates that a single competition is insufficient, and multiple rounds are necessary to obtain reliable judgments and rankings for clusters. Moreover, the observed saturation may reflect inherent limitations of the LLM-as-a-judge paradigm for identifying the best solutions.

4.6 Evaluating the Quality of the Ranking

To assess the effectiveness of our cluster ranking method, we evaluated how often the best solution, defined as the one achieving the highest score for a given problem, appeared within the top-K ranked clusters. This metric provides a clear measure of ranking quality. As shown in Figure 6, in 35 out of 39 subtasks, the best solution is included among the top 50 clusters, demonstrating strong performance while still leaving room for further improvement.

4.7 Impact of the Maximum Number of Tokens

Previous work has shown a correlation between reasoning length and accuracy on challenging problems when reasoning models are used (OpenAI, 2024). In this section, we analyze the relationship between performance and generation length by comparing the `Score@5000` of different models under varying maximum token limits. Results are presented in Figure 7, and the rationale behind the chosen generation limits for each model is discussed in Section 4. The scores of `gpt-oss-120b`, `gpt-oss-20b`, and `DeepSeek-R1-0528` continue to improve up to their respective token limits, whereas `Qwen3-235B-A22B` saturates after approximately 48K tokens, below the 80K length recommended for this model. Notably, the family of gpt-oss models generate longer reasoning traces for difficult problems and achieve stronger performance as a result. While `DeepSeek-R1-0528` and `Qwen3-235B-A22B` perform better at shorter reasoning lengths, the gpt-oss models surpass them when provided with greater compute budgets.

5 Conclusion

In this work, we present a test-time compute approach that achieves gold-medal performance at the IOI using open-weight models for the first time. Our method generates a large pool of candidate solutions in parallel, followed by a pipeline of behavioral clustering, ranking, and round-robin submission to identify the most promising solutions within the IOI submission constraints. Our experiments show that `gpt-oss-120b`, when equipped with our test-time compute approach, consistently improves its score on IOI as the number of generated candidates increases.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint*, 2024. arXiv:2407.21787 [cs.LG].

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021b.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. Rm-r1: Reward modeling as reasoning, 2025. URL https://arxiv.org/abs/2505.02387.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

The Decoder. Openai's ai system wins a gold medal-level score at the international olympiad in informatics 2025. https://the-decoder.com/openais-ai-system-wins-a-gold-medal-level-score-at-the-international-olympiad-in-informatics-2025/, 2025. Accessed: 2025-10-15.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu,

Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025. URL https://arxiv.org/abs/2508.15260.

Hugging Face. IOI Dataset (open-r1/ioi). https://huggingface.co/datasets/open-r1/ioi, 2025. Accessed: 2025-10-14.

Jeevana Priya Inala, Chenglong Wang, Mei Yang, Andres Codas, Mark Encarnación, Shuvendu Lahiri, Madanlal Musuvathi, and Jianfeng Gao. Fault-aware neural code rankers. *Advances in Neural Information Processing Systems*, 35:13419–13432, 2022.

International Olympiad in Informatics. Results of IOI 2025. https://stats.ioinformatics.org/results/2025, 2025a. Accessed: 2025-10-14.

International Olympiad in Informatics. Ioi 2025 tasks. https://ioi2025.obi.org.bo/tasks.html, 2025b. Accessed: 2025-10-14.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

Rémi Leblond, Felix Gimeno, Florent Altché, Alaa Saade, Anton Ruddock, Corentin Tallec, George Powell, Jean-Bastien Grill, Maciej Mikuła, Matthias Lochbrunner, et al. AlphaCode 2 technical report. Technical report, Google DeepMind, December 2023. URL https://storage.googleapis.com/deepmind-media/AlphaCode2/AlphaCode2_Tech_Report.pdf. Accessed: 2025-01-14.

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 Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*, 2022.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models, 2025. URL https://arxiv.org/abs/2502.17419.

Hanzhao (Maggie) Lin and Heng-Tze Cheng. Gemini achieves gold-level performance at the international collegiate programming contest world finals. https://deepmind.google/discover/blog/gemini-achieves-gold-level-performance-at-the-international-collegiate-programming-contest-world-finals/, September 2025. DeepMind Blog. Accessed: 2025-10-14.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Pairjudge rm: Perform best-of-n sampling with knockout tournament. *arXiv preprint arXiv:2501.13007*, 2025.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models, 2024. URL https://arxiv.org/abs/2410.12832.

OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: YYYY-MM-DD.

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

OpenAI, :, Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, Jerry Tworek, Lorenz Kuhn, Lukasz Kaiser, Mark Chen, Max Schwarzer, Mostafa Rohaninejad, Nat McAleese, o3 contributors, Oleg Mürk, Rhythm Garg, Rui Shu, Szymon Sidor, Vineet Kosaraju, and Wenda Zhou. Competitive programming with large reasoning models, 2025. URL https://arxiv.org/abs/2502.06807.

Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. CodeForces Dataset (open-r1/codeforces). https://huggingface.co/datasets/open-r1/codeforces, 2025. Hugging Face repository. Accessed: 2025-10-14.

Qwen Team. Qwen3-235B-A22B-Thinking-2507-FP8. https://huggingface.co/Qwen/Qwen3-235B-A22B-Thinking-2507-FP8, 2025. Quantized FP8 model. Accessed: 2025-10-14.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in llm-as-a-judge. *arXiv preprint arXiv:2406.07791*, 2024. URL https://arxiv.org/abs/2406.07791.

Qwen Team. Qwen3, April 2025. URL https://qwenlm.github.io/blog/qwen3/.

Hung To, Minh Nguyen, and Nghi Bui. Functional overlap reranking for neural code generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3686–3704, 2024.

Shubham Toshniwal, Ivan Sorokin, Aleksander Ficek, Ivan Moshkov, and Igor Gitman. Genselect: A generative approach to best-of-n, 2025. URL https://arxiv.org/abs/2507.17797.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS) 35*, pp. 24824–24837. Curran Associates, Inc., 2022.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models, 2025. URL https://arxiv.org/abs/2408.00724.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Kexun Zhang, Danqing Wang, Jingtao Xia, William Yang Wang, and Lei Li. Algo: Synthesizing algorithmic programs with llm-generated oracle verifiers, 2023. URL https://arxiv.org/abs/2305.14591.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2025a. URL https://arxiv.org/abs/2408.15240.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well?, 2025b. URL https://arxiv.org/abs/2503.24235.

Zhuorui Zhao, Ruidi Qiu, Chao Lin, Grace Li Zhang, Bing Li, and Ulf Schlichtmann. Vrank: Enhancing verilog code generation from large language models via self-consistency. In *2025 26th International Symposium on Quality Electronic Design (ISQED)*, pp. 1–7. IEEE, 2025.

A PROMPTS

Solution Generation Prompt

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++17. 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
```

{question}

Figure 8: The prompt used for generating the solutions

Test Data Generator Prompt

You are an expert competitive programmer. You will be given a problem statement, its constraints, and example test cases. Your task is to write a **test case generator** in C++17 that produces valid inputs for the problem, following the constraints and reflecting the variety suggested by the examples.

First, **reason step by step** about how to design the generator.

Then, provide the **final complete implementation** inside a single code block:
```cpp
// your code here
```

Question:

{problem}

Answer: (use the provided format with backticks)

Figure 9: The prompt used for generating the test data generators

Test Data Validator Prompt

You are an expert competitive programmer. You will be given a problem statement, its input format, constraints, and examples. Your task is to write an **input validator** in C++17 that reads from stdin and checks that the input fully matches the spec and all constraints.

First, reason step by step about what must be validated.

Then, provide the **final complete implementation** inside a single code block:
```cpp
// your code here
```

The program should print "passed" if the input is valid, otherwise "failed" and a short error to stderr. It must also ensure no extra tokens remain.

Question:

{problem}

Answer: (use the provided format with backticks)

Figure 10: The prompt used for generating the test data validators

Solution Selection Prompt

You are an expert competitive programmer. You will be given a problem statement, its constraints, and two solutions. Your task is to evaluate each solution's correctness based on the problem statement and its constraints and select the best solution.
First, **reason step by step** about each solution.

Question:
{problem}

Solution A
{code_A}

Solution B
{code_B}

Finish your reasoning with exactly three lines, nothing else:
Score A: <0-10>
Score B: <0-10>
Judgment: [A] or Judgment: [B]

Figure 11: The prompt used for comparing the solutions in the tournament

B BEST SCORES ON IOI-2025

Table 2: Scores of the top gold and silver medalists at IOI 2025, compared with the results achieved by GENCLUSTER and OpenAI (International Olympiad in Informatics, 2025a). Per-problem scores and overall rankings are shown.

Rank	Name	S	T	W	F	M	O	Total	Medal
1	Hengxi Liu	100.00	100.00	100.00	100.00	91.23	100.00	591.23	Gold
6	**OpenAI**	100.00	75.29	93.00	100.00	65.00	100.00	533.29	Gold
26	**GENCLUSTER**	100.00	87.66	72.00	82.00	22.09	83.00	446.75	Gold
29	Maxim Tsoy	100.00	44.83	100.00	82.00	72.89	37.00	436.72	Silver