SubtaskEval: Benchmarking LLMs on Competitive Programming Subtasks

Samik Goyal
DAV Public School
Patiala, Punjab 147001, India
samikgoyal@gmail.com

Abstract

Existing code generation benchmarks such as HumanEval, MBPP, and LiveCodeBench evaluate only full solutions, overlooking meaningful partial progress on competitive programming tasks. We introduce **SubtaskEval**, a benchmark of 287 olympiad problems (2017–2025) that preserves official subtask structures, metadata, and online-judge links. Evaluating six recent LLMs, including a tool-augmented variant, we find that even the best model achieves only 18.47% accuracy (pass@1) though tool use improves subtask performance. Models exhibit bottom-heavy score distributions, in contrast to the more balanced distributions of human contestants. Subtask-based evaluation thus provides a finer-grained view of model problem-solving and highlight directions for advancing LLMs in code generation.

1 Introduction

Large language models (LLMs) have achieved near-saturation on standard code generation benchmarks such as HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and LiveCodeBench (Jain et al., 2024), with state-of-the-art models able to solve 80% of problems in entirety in a single run (pass@1). In contrast, competitive programming (CP) problems from informatics olympiads are far more challenging: in our benchmark the best pass@1 was only 18.47%. To probe deeper capabilities, competitive programming (CP) tasks are attractive: they are subdivided into *subtasks*, constrained variants of the main problem that award partial credit (e.g., solving $N \leq 1000$ cases in a problem with $N \leq 10^5$ may award 40% of the max total score). Subtasks also carry different weights (e.g., a simpler case may yield 20% of points, while solving a subtask with a bigger scope may yield 80%).

Example. Consider a shortest-path problem where solving the full version requires $O(N)$ algorithms, but a subtask restricted to $N \leq 500$ is solvable with the Floyd–Warshall algorithm. A model that fails on the full problem but succeeds on this subtask still shows meaningful competence. Subtask-level evaluation thus reveals differences that case-by-case test scoring or pass@1 metrics obscure.

Recent benchmarks extend to competition-level problems, including CodeContests+ (Wang et al., 2025b), USACO Benchmark (Shi et al., 2024), OIBench (Zhu et al., 2025), and OJBench (Wang et al., 2025a). However, none preserve official subtasks. Alternatives like test-case fractions (Hendrycks et al., 2021) do not capture the structured partial credit used in contests.

Our Contributions. (1) We introduce SUBTASKEVAL, a benchmark of 287 olympiad problems (2017–2025) with full subtask structure, metadata, and online-judge links. (2) We evaluate six modern LLMs, including tool-augmented variants, using new metrics such as the Normalized Subtask Score, and examine differences between human and LLM performance distributions, the benefits of tool use, and temporal performance trends.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Deep Learning for Code in the Agentic Era.

2 Benchmark Curation

SUBTASKEVAL comprises 287 tasks curated from international informatics olympiads between 2017 and 2025. Each task includes the full problem statement, input/output specification, sample tests, and a description of subtasks. All problems are designed to be solved in C++. An illustrative problem is provided in Appendix D. In addition to problem content, we provide metadata annotations capturing each task's *Name*, *Year*, *Source*, *Number of subtasks*, and *Input type* (standard I/O or interactive).

For reproducibility, each task is linked to the online judge `https://qoj.ac/`, which supports direct submission and evaluation. We also release automation scripts for submission and result retrieval, and provide the dataset in the supplementary material for now (see Appendix A).

The benchmark draws tasks from five olympiads: the Japanese Olympiad in Informatics (JOI) (JOI Committee, 2017–2025), the European Girls' Olympiad in Informatics (EGOI) (EGOI Committee, 2020–2025), the Central European Olympiad in Informatics (CEOI) (CEOI Committee, 2017–2025), the Asia-Pacific Informatics Olympiad (APIO) (APIO Committee, 2017–2025), and the Baltic Olympiad in Informatics (BOI) (BOI Committee, 2017–2025). Attribution details for each source are given in Appendix B.

3 Evaluation Methodology

3.1 Models

We evaluated the zero-shot performance of 6 LLMs: *gpt-5-mini*, *gpt-4.1*, *o4-mini*, *gemini-2.5-flash*, *gemini-2.5-pro*, and *deepseek-v3.1-non-thinking* on the benchmark to provide a baseline. We also evaluated *gpt-5-mini* with the *Code Execution Tool*. The prompts used can be found in Appendix E.

3.2 Metrics

Each model is ran once per problem and the code is evaluated across all subtasks of the problem. The verdict is defined as the sum of the weights of successfully solved subtasks, with a maximum of 100 per problem. In addition to numeric scores, two special verdicts are possible: *Compile Error*, assigned when the generated code fails to compile, and *No Output*, assigned when a model refuses or fails to produce code for a given task. Valid generations refer to problems where the model produced code which compiled.

We report both standard metrics from prior code generation benchmarks and new metrics tailored to the subtask setting.

Let the total number of problems be N, and let problem i contain S_i subtasks. Denote by $v_{i,s}$ the score for subtask s of problem i. If the verdict for problem i is a *Compile Error* or *No Output*, we set $v_{i,s} = 0 \ \forall\, s \in \{1, 2, \ldots, S_i\}$.

Full-problem pass@1 (%) Introduced in Chen et al. (2021), this metric reports the percentage of problems solved in their entirety in a single run:

$$pass@1 = 100 \cdot \frac{1}{N} \sum_{i=1}^{N} \mathbb{1} \left\{ \sum_{j=1}^{S_i} v_{i,j} = 100 \right\}.$$

Average score over valid generations (AVG) This is the average score per problem, computed only over problems that did not result in *Compile Error* or *No Output*.

Normalised Subtask Score (NSS) This is the average percentage of subtasks solved per problem:

$$NSS = 100 \cdot \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{S_i} \cdot \sum_{j=1}^{S_i} \mathbb{1}\{v_{i,j} > 0\} \right).$$

Table 1: Comparison of model performance across error rates, accuracy, and scoring metrics. Columns 2–5 report percentage of cases with no output, compile errors, zero score, and pass@1. Columns 6–8 report average score over valid generations (see 3.2), normalized subtask score (NSS) over valid generations (see 3.2), and NSS over all problems. Best values are in **bold**, second-best are underlined.

Model	No Output %	Compile Error %	0%	pass@1	AVG	NSS(val)	NSS
deepseek-v3.1-nonthink	32.40	7.32	32.40	6.27	15.54	19.01	12.85
gemini-2.5-flash	57.84	3.83	21.95	6.97	24.47	24.87	10.49
gemini-2.5-pro	70.03	**1.05**	**12.20**	7.32	**33.49**	**37.24**	11.16
gpt-4.1	**0.00**	26.48	52.96	1.05	5.35	6.66	6.66
gpt-5-mini	**0.00**	16.03	38.33	14.63	25.40	26.37	26.37
gpt-5-mini+tool	0.35	16.38	32.40	**18.47**	31.36	30.79	**30.68**
o4-mini	3.14	14.98	39.37	15.68	25.33	26.14	25.32

4 Results and Discussion

We present results evaluating multiple LLMs, comparing them to human performance, assessing tool use, and analyzing temporal trends. Both Gemini models (*gemini-2.5-flash* and *gemini-2.5-pro*) exhibited a high *No Output* rate: despite retries, they often used 20–30 thinking tokens but returned an empty string. In contrast, the *gpt* models failed mainly with compile errors, particularly on interactive tasks, where they produced full programs with a `main` function instead of the required function.

4.1 Performance of LLMs relative to human benchmark

We plotted violin distributions for all models and for the top 40 contestants from EGOI[1] (Figure 1). Full dataset distributions are in Appendix C, though human scores are unavailable. Model distributions are bottom-heavy, with mass at low scores and some density near the maximum, while humans have a balanced distribution centered around 20 points. This likely reflects strategy: humans secure partial credit, whereas LLMs produce all-or-nothing solutions.

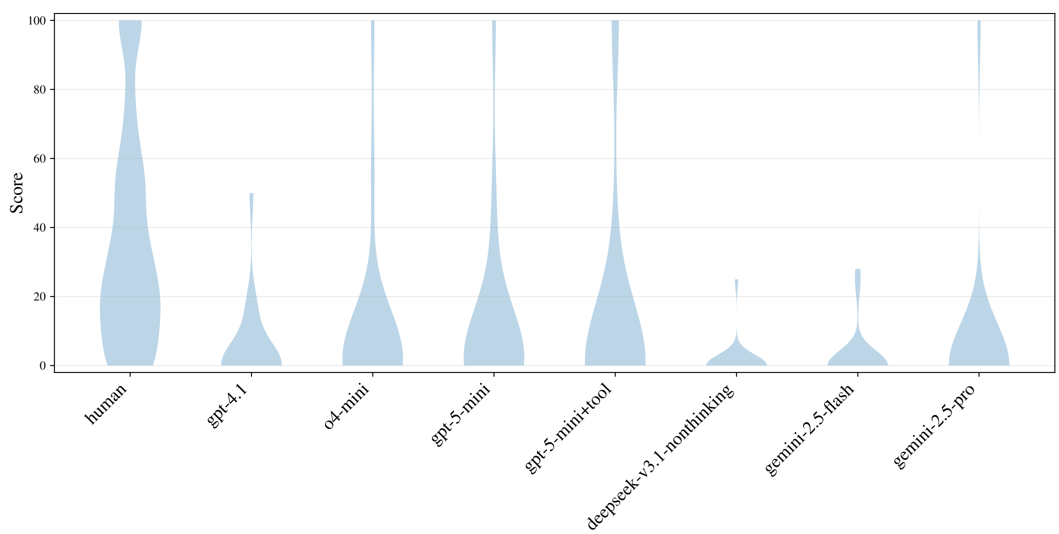

Figure 1: Score distribution of models with human reference (EGOI). Width indicates score density.

4.2 Benefits of Tool Use

The *Code Execution Tool* enables the model to run Python code for verifying its solutions against self-generated test cases. This additional feedback loop helps the model identify and correct errors

[1]The human group corresponds to the top 40 contestants from EGOI.

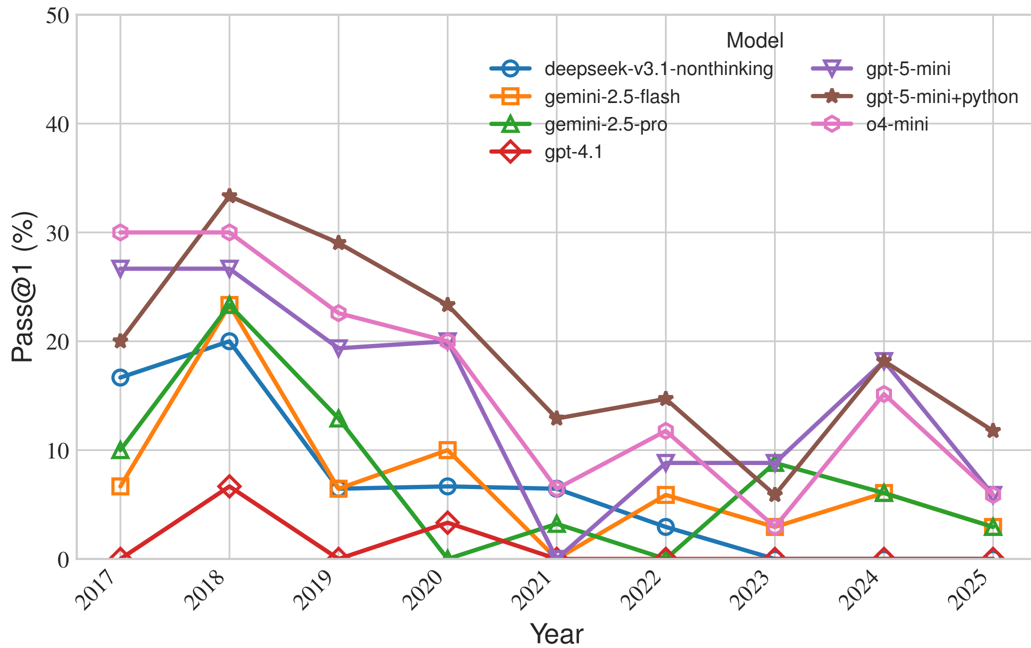

Figure 2: Pass@1 by year of problem release.

before generating the final output. This could explain why we observe substantial gains in both NSS and average score, even though the frequency of compile errors remains largely unchanged.

4.3 Temporal Trends

Because many of the benchmark problems were released prior to the knowledge cut-off dates of the evaluated models, we also studied the effect of problem release year on model accuracy. Figure 2 shows $pass@1$ across problems for each model grouped by year of release. Performance is higher on problems from 2017–2020 but levels off from 2021–2025. Since all model cut-offs extend beyond 2021, the decline is unlikely due to memorisation and instead reflects increased problem difficulty. Moreover, modern CP-oriented LLMs rely on reinforcement learning from outcome signals rather than direct data memorisation, reinforcing difficulty as the main factor.

5 Conclusions and Future Work

We present SubtaskEval, a benchmark that evaluates LLMs on competitive programming problems with subtasks. Our experiments show that while current models achieve partial progress, they still lag behind human performance, with distinct error patterns and bottom-heavy score distributions. In future, we plan to expand coverage to more models and agentic settings, evaluate multiple runs to better capture stochastic variability, perform deeper per-model analysis, and explore ways to mitigate potential data contamination.

Limitations

This work has three main limitations: (i) we could not rewrite problems to mitigate temporal effects, as preserving the correctness and difficulty of Olympiad tasks makes such rewriting infeasible; (ii) our evaluation covers only six models and one tool-augmented variant, evaluated with a single run per problem, due to budget limitations, leaving broader families and agentic setups unexplored; and (iii) because all problems were publicly released after their contests, data leakage into model training corpora is possible, and contamination or fine-tuning remain future concerns.

Acknowledgments and Disclosure of Funding

We thank Dr. Srujana Merugu for insightful discussions and for reviewing the final version of this work.

The author declares no funding and no competing interests.

References

APIO Committee. Asia-pacific informatics olympiad (apio), 2017–2025. URL `https://apio2025.org/`. International olympiad held across Asia-Pacific countries.

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`.

BOI Committee. Baltic olympiad in informatics (boi), 2017–2025. URL `https://boi2025.ut.ee/`. Regional olympiad for Baltic countries.

CEOI Committee. Central european olympiad in informatics (ceoi), 2017–2025. URL `https://ceoi.inf.elte.hu/`. Regional olympiad for Central European countries.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL `https://arxiv.org/abs/2107.03374`.

EGOI Committee. European girls' olympiad in informatics (egoi), 2020–2025. URL `https://egoi.org/`. International olympiad for female and non-binary participants.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps, 2021. URL `https://arxiv.org/abs/2105.09938`.

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`.

JOI Committee. Japanese olympiad in informatics (joi), 2017–2025. URL `https://www.ioi-jp.org/`. National Olympiad of Japan used for IOI selection.

Quan Shi, Michael Tang, Karthik Narasimhan, and Shunyu Yao. Can language models solve olympiad programming?, 2024. URL `https://arxiv.org/abs/2404.10952`.

Zhexu Wang, Yiping Liu, Yejie Wang, Wenyang He, Bofei Gao, Muxi Diao, Yanxu Chen, Kelin Fu, Flood Sung, Zhilin Yang, Tianyu Liu, and Weiran Xu. Ojbench: A competition level code benchmark for large language models, 2025a. URL `https://arxiv.org/abs/2506.16395`.

Zihan Wang, Siyao Liu, Yang Sun, Hongyan Li, and Kai Shen. Codecontests+: High-quality test case generation for competitive programming, 2025b. URL `https://arxiv.org/abs/2506.05817`.

Yaoming Zhu, Junxin Wang, Yiyang Li, Lin Qiu, ZongYu Wang, Jun Xu, Xuezhi Cao, Yuhuai Wei, Mingshi Wang, Xunliang Cai, and Rong Ma. Oibench: Benchmarking strong reasoning models with olympiad in informatics, 2025. URL `https://arxiv.org/abs/2506.10481`.

A Code release

We release the complete dataset as a supplementary material to this paper with the CC-BY-SA licence. We also release the code used to submit the problems to the *qoj.ac* website under the MIT licence. Both are available at https://doi.org/10.5281/zenodo.17370525.

B Dataset Sources and Attributions

We gratefully acknowledge the contest organizers who created and released the problems used in the dataset. All tasks were sourced from publicly available contest archives. Below we attribute each olympiad to its official organizers:

JOI (Japanese Olympiad in Informatics). Organized by the Japanese Olympiad in Informatics Committee. Problems and solutions are released on the official JOI archive (`https://www.ioi-jp.org/`).

EGOI (European Girls' Olympiad in Informatics). Organized by the EGOI International Committee with hosting rotating among participating countries. Problems are released on the official EGOI site (`https://egoi.org/`).

CEOI (Central European Olympiad in Informatics). Organized annually by a rotating Central European host under the coordination of the CEOI committee. Problems are available on the official CEOI archive (`https://ceoi.inf.elte.hu/`).

APIO (Asia-Pacific Informatics Olympiad). Organized by the APIO Scientific Committee with hosting rotating across Asia–Pacific countries. Problems are published on the official APIO sites of each year. The website for the last edition is at `https://apio2025.uz/`.

BOI (Baltic Olympiad in Informatics). Organized by the Baltic Olympiad in Informatics Committee, with hosting rotating among Baltic countries. Problems are released on the official BOI sites of each year. The website for the last edition is at `https://boi2025.mat.umk.pl/`.

C Score distribution of models on the full dataset

Figure 3 has the score distributions of the models over the complete dataset.

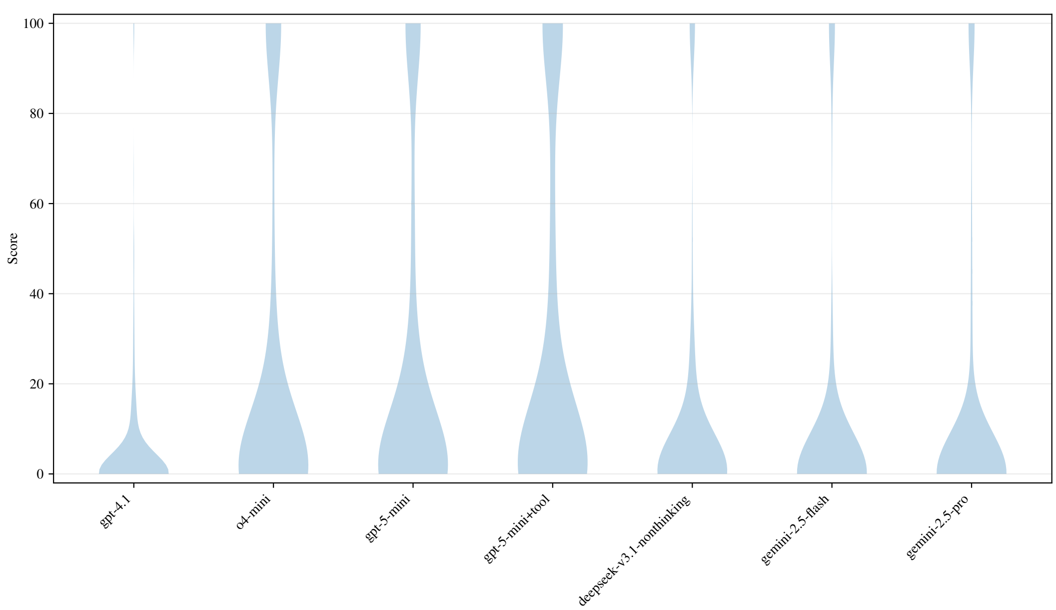

Figure 3: Score distributions of the models on the full dataset. Width indicates density of scores.

D Illustrative problem

The pdf version of one of the problems in the data set can be found at https://www2.ioi-jp.org/joi/2024/2025-ho/2025-ho-t5-en.pdf.

E Zero-shot prompts

The prompt we used for the non-tool models was:

````
Please reply with a C++14 solution to the below problem. Make sure to wrap your code in ```cpp and ``` Markdown delimiters, and include exactly one block of code with the entire solution.

Reason through the problem and think step by step.

No outside libraries are allowed.

Test out your solution on the provided samples, and ensure it works correctly.

If you are unable to solve the full problem, try to solve the subtasks and merge them into one solution with strict case checking.

Your goal is to maximize your score which is the sum of points for all subtasks.

[BEGIN PROBLEM]

{problem_block}

[END PROBLEM]
````

The prompt that we used for the tool variant was:

````
Solve the problem below and produce a **single** final answer: one C++17 program.

Think step-by-step:
1) Restate the problem in your own words.
2) Design the approach in plain English; consider constraints and edge cases.
3) Draft pseudocode.
4) Use the python tool to **verify** the approach by stress testing (no logs in the final output):
   - In Python, generate random test cases within the problem's constraints.
   - Implement a small-input brute-force oracle.
   - Implement a correct efficient solver (Python is fine for the test harness).
   - Compare brute vs efficient over many random cases (use a fixed RNG seed).
   - If any mismatch occurs, investigate and fix before proceeding.
   - Optionally check invariants (monotonicity, bounds, graph properties, etc.).

Final deliverable (the ONLY thing you print):
- Output exactly one fenced code block containing valid C++17 that solves the **full** problem if possible.
- If only subtasks are solvable, write the best mixed strategy and **merge** them into one file using clear condition checks.
- Start the code block with ```cpp and end it with ```.

Your goal is to maximize your score which is the sum of points for all subtasks.
````

```
Output policy:
- Do **not** print any analysis, pseudocode, stress-test logs, or
↪  explanations outside the final code block.
- The final code block must be the **only** content in your reply.

[BEGIN PROBLEM]
{problem_block}
[END PROBLEM]
```