ByteDance | Seed

# AetherCode: Evaluating LLMs' Ability to Win In Premier Programming Competitions

**ByteDance, M-A-P**

Full author list in Contributions

## Abstract

Competitive programming has emerged as a critical benchmark for evaluating the reasoning and coding capabilities of Large Language Models (LLMs). Despite impressive progress on existing benchmarks, we argue that current evaluations overstate model proficiency, masking a substantial gap between LLMs and elite human programmers. This gap arises from two key limitations: insufficient difficulty and scope of benchmark problems, and evaluation bias from low-quality test cases. To address these shortcomings, we present **AetherCode**, a new benchmark that draws problems from premier programming competitions such as IOI and ICPC, offering broader coverage and higher difficulty. AetherCode further incorporates comprehensive, expert-validated test suites built through a hybrid of automated generation and human curation, ensuring rigorous and reliable assessment. By combining challenging problem design with robust evaluation, AetherCode provides a more faithful measure of LLM capabilities and sets a new standard for future research in code reasoning.

**Date:** August 22, 2025
**Correspondence:** Zihan Wang at zh.wang@bytedance.com, Jiaze Chen at chenjiaze@bytedance.com
**Project Page:** https://huggingface.co/datasets/m-a-p/AetherCode

## 1 Introduction

Competitive programming is widely regarded as a crucial benchmark for evaluating the reasoning and coding capabilities of Large Language Models (LLMs) [14]. Solving complex competitive programming problems demands not only sophisticated reasoning abilities but also knowledge from diverse domains, including mathematics, data structures, and algorithms. Recent years have witnessed rapid advancements in the reasoning capabilities of LLMs, a key indicator of which is their success on a majority of existing code reasoning benchmarks. State-of-the-art models now achieve over 90% *Pass@1* accuracy on MBPP [1] and HumanEval [2], and over 80% on LiveCodeBench [6]. These encouraging developments might lead one to ask: has competitive programming been mastered by LLMs?

In this paper, we argue that a significant gap still exists between the performance of LLMs and top-tier human competitors in programming contests. We propose that the perception of LLM dominance stems primarily from the limitations in the breadth and rigor of current code reasoning benchmarks, which are no longer sufficient to fully assess the capabilities of today's increasingly powerful models. Specifically, we identify two main shortcomings in existing benchmarks:

- **Insufficient Difficulty and Scope.** Early benchmarks such as HumanEval [2] and MBPP [1] consist of basic coding tasks, for instance, sorting or reversing a list, which present minimal reasoning challenges

for state-of-the-art LLMs. More recent "competition-level" benchmarks often source problems from a limited set of websites. For example, LiveCodeBench [6] collects problems mainly from LeetCode and AtCoder, while CodeELO [15] and LiveCodeBench Pro [21] focus solely on CodeForces. Each of these websites has inherent limitations. LeetCode problems are generally easier and often require only the implementation of a single function rather than a complete program. CodeForces contests, which typically feature 5-7 problems within a 2-3 hour timeframe, constrain the design space for problem setters, for example, leading to a scarcity of problems that require complex, large-scale implementations.

- **Evaluation Bias from Low-Quality Test Cases.** The correctness of a piece of code is verified using a comprehensive set of test cases (input-output pairs). An incomplete test suite may fail to detect incorrect submissions, particularly those with subtle flaws, such as the mishandling of corner cases or solutions that exceed time limits under specific, extreme conditions. Consequently, designing high-quality test cases is a huge challenge that requires a deep understanding of potential failure points, a skill typically honed through extensive competitive programming experience. Most past benchmarks lack sufficiently rigorous test cases. HumanEval [2] and MBPP [1], for instance, rely on a small number of handwritten test cases. Others, including EvalPlus [10], CodeContests [8], and LiveCodeBench [6], employ naive test case generation pipelines, such as random mutation, which fall far short of the quality of expert-designed test suites. Furthermore, recent research [19] has revealed issues with test case correctness itself; for example, many test cases in the CodeContests dataset do not adhere to the problem's constraints, causing even correct solutions to fail. It is worth noting that some recent benchmarks, such as CodeELO [15] and LiveCodeBench Pro [21], have attempted to leverage the official CodeForces judging service to indirectly access its high-quality, expert-crafted test cases. However, this approach presents two significant issues. First, it raises compliance risks, as CodeForces explicitly prohibits the use of crawlers on its judging interface. Second, this method is constrained by submission frequency limits, which impedes agile and flexible experimentation. Therefore, we contend that an open-source benchmark with high-quality, self-contained test cases remains critically important for the LLM community.

To address these challenges, we introduce AetherCode, a new benchmark with the following key contributions:

**Problem Curation from Top-Tier Competitions.** AetherCode is the first benchmark to systematically collect problems from premier programming competitions worldwide, including the Olympiad in Informatics (OI) and the International Collegiate Programming Contest (ICPC). Our process involved a comprehensive collection, meticulous cleaning, and format conversion of problems from PDF to a Markdown+LaTeX structure. Each problem statement was manually proofread for correctness, and a team of competitive programming experts annotated each problem with classification tags.

**High-Quality Test Case Generation.** We developed a hybrid methodology, combining automated generation with expert annotation, to create high-quality test cases for every problem. We evaluated the correctness and comprehensiveness of our test cases by validating them against a large corpus of collected solutions, enforcing a standard of zero false positives and zero false negatives.

This paper is organized as follows: Section 2 details the benchmark curation process. Section 3 presents our evaluation results. Section 4 concludes the paper and discusses directions for future research.

## 2 Benchmark Curation

This Section details the curation process of the AetherCode Benchmark. Sections 2.1 and 2.2 describe the specifics of problem collection and categorizing, respectively. Section 2.3 explains how we construct high-quality test cases for each problem, and Section 2.4 presents the statistical data of AetherCode.

### 2.1 Problem Collection

We source our problems from premier programming competitions worldwide rather than from online programming websites. Based on their target audience, these competitions can be broadly categorized into two main series: the Olympiad in Informatics (OI) series, which is aimed at pre-college school students, and the International Collegiate Programming Contest (ICPC) series, which is designed for college students.

**Table 1** Comparison between AetherCode and other code reasoning benchmarks

| Dataset | Difficulty | # Problems | Updates | Test Cases Construction | Source |
|---|---|---|---|---|---|
| HumanEval [2] | ★ | 164 | ✘ | Handcrafted | Original |
| MBPP [1] | ★ | 974 | ✘ | Handcrafted | Original |
| APPS [5] | ★★★ | 5,000 | ✘ | Crawled | CodeForces, AtCoder *etc.* |
| USACO [17] | ★★★ | 307 | ✘ | Publicly accessible | USACO |
| CodeContests [8] | ★★★ | 165 | ✘ | Mutation | CodeForces, AtCoder *etc.* |
| LiveCodeBench [6] | ★★ | 1055 | ✔ | Semi-automatic | LeetCode, AtCoder |
| CodeELO [15] | ★★★ | 387 | ✔ | ✘ | CodeForces |
| LiveCodeBench Pro [21] | ★★★ | 584 | ✔ | ✘ | CodeForces |
| AetherCode | ★★★★ | 456 | ✔ | G-V Agent[19] & Experts | Premier Contests |

**OI Series.** The Olympiad in Informatics is a series of competitions aimed at popularizing computer science knowledge among middle-school students and cultivating outstanding talents in computer science. The OI competitions usually require participants to solve algorithm-related problems by programming. Take the International Olympiad in Informatics (IOI), the top-level event of OI, as an example. Each contestant competes individually, and each country can send up to 4 players. During the two-day competition, players need to independently solve 3 problems within 5 hours each day, mainly using C++. Furthermore, various countries and regions host their own national or regional OI competitions, such as the National Olympiad in Informatics (NOI) in China and the USA Computing Olympiad (USACO) in the United States. Top-performing contestants in these competitions earn the opportunity to advance to the IOI.

**ICPC Series.** The ICPC is the oldest, largest, and most prestigious university-level programming contest in the world. Each team consists of up to 3 students and uses one computer to solve 10 - 13 problems in 5 hours, using programming languages such as C, C++, Java, or Python. The team that correctly solves the most problems with the least total time wins.

The world is divided into several regions for the ICPC. In Europe, there are Central Europe (CERC), North Europe (NWERC), South-East Europe (SEERC), and South-West Europe (SWERC) regions. Other regions include Asia-Pacific, Asia East Continent, North America, Latin America, Africa, and Arab region, etc. The ICPC is a multi-tiered event. First, there are **regional contests** held worldwide from September to November each year. The top-performing teams in the regional contests advance to the **regional finals or championships**. Then, the best teams from these finals or championships qualify for the ICPC **World Finals**, which is usually held from April to June each year. This is the highest-level stage of the ICPC, where the best teams from around the world compete for the championship.

In addition to the official ICPC events, we also incorporated problems from other large-scale and renowned collegiate programming contests, such as the China Collegiate Programming Contest (CCPC).

For each problem, we collected the following components:

- **Problem Statement.** The statement typically comprises a title, a detailed problem description, input/output specifications, sample inputs and outputs with explanations, data range constraints, and time/memory limits. The majority of the problem statements was originally in PDF format. To enhance comprehension for LLMs, we converted these PDFs into a Markdown format with LaTeX for mathematical notations. Each converted file was then manually proofread to ensure its accuracy.
- **Solutions.** We curated a collection of over 30,000 human-written solutions for these problems, encompassing both correct and incorrect submissions. For each problem, we ensured a minimum of 5 correct and 20 incorrect solutions. The primary purpose of collecting these solutions is to evaluate the quality of the subsequently generated test cases, a process detailed in Section 2.3.
- **Test Cases.** A minority of the competitions, e.g., USACO, publicly released their official test cases, which we collected and standardized. For problems where official test cases were not available, we

constructed our high-quality test cases. The methodology for this construction is described in Section 2.3.

- **Metadata.** We also gathered auxiliary information, such as the date of the competition (for decontamination purposes) and human contestant performance data (to facilitate difficulty assessment), among other available data points.

## 2.2 Problem Categorization

Beyond curating problems, an equally critical step in constructing AetherCode was the systematic categorization of each problem to ensure comprehensive coverage and facilitate fine-grained evaluation. To this end, we adopted a multi-dimensional categorization framework designed with the input of competitive programming experts:

1. **Difficulty Segmentation.** Problems were divided into four levels of difficulty: *Easy*, *Medium*, *Hard*, and *Extreme*. This classification was guided by expert judgment as well as official contest results. Notably, problems that no human contestant was able to solve during a competition were classified as *Extreme*, representing challenges that push the boundaries of algorithmic reasoning.
2. **Temporal and Contextual Dimensions.** Each problem was annotated with metadata to enable both decontamination and longitudinal analysis of model performance:
   - Year of the contest, allowing chronological tracking of trends in problem design and model capabilities.
   - Competition type, primarily distinguishing between Olympiad in Informatics (OI) and International Collegiate Programming Contest (ICPC) series.
   - Competition scope, categorizing contests as regional-level, national-level, continental-level, or world finals.
3. **Problem Format Constraints.** Some problems require additional considerations beyond a standard input–output interface:
   - Problems dependent on visual or image-based input were excluded from the benchmark.
   - Problems requiring special judges or custom checkers were explicitly labeled to ensure proper handling during evaluation.
4. **Algorithmic and Domain Categories.** To capture the breadth of algorithmic knowledge tested in programming contests, we implemented a hierarchical taxonomy as shown in Table 5:
   - Primary categories correspond to major domains such as *Dynamic Programming*, *Graph Theory*, *Computational Geometry*, *Data Structures*, and *Mathematics*.
   - Secondary categories provide finer granularity, such as *tree dynamic programming*, *flow algorithms*, *convex hull geometry*, or *modular arithmetic*. Problems can belong to multiple categories to reflect their cross-disciplinary nature.

This structured categorization enables targeted evaluation of model strengths and weaknesses while also ensuring that AetherCode serves as a scalable resource for future research. In particular, it allows progress to be tracked across difficulty levels, problem types, and algorithmic domains, providing a more comprehensive understanding of model capabilities.

## 2.3 Test Case Construction

Recent studies [10, 19] have highlighted concerns regarding the quality of test cases in several existing code datasets. For instance, benchmarks such as MBPP [1] and HumanEval [2] include only a limited number of handwritten test cases per problem. Others, like CodeContests [8] and EvalPlus [10], rely on naive methods such as mutation to generate test cases. Consequently, such test cases are insufficient for comprehensively

evaluating the correctness and efficiency of a program. Therefore, we contend that the quality of test case construction is a critical factor determining the overall quality of a benchmark.

Notably, some recent benchmarks [15, 21] directly utilize the CodeForces's judging service for evaluation. This approach allows them to indirectly access high-quality test cases created by professional problem setters, thereby circumventing the challenge of test case construction. However, this method presents potential compliance risks, as CodeForces explicitly prohibits the use of crawlers on its judging interface. Furthermore, this approach is constrained by submission frequency limits, which impedes agile and flexible evaluation. Therefore, we argue that a benchmark equipped with its own high-quality test cases remains critically important for the LLM community.

To ensure AetherCode possesses sufficiently high-quality test cases, we approached the task from two perspectives. First, we established more stringent evaluation criteria for test case quality, which is presented in Section 2.3.1. Second, we employed a hybrid approach, combining automated generation with expert annotation, to construct the test cases, which are presented in Sections 2.3.2 and 2.3.3.

### 2.3.1 Test Case Quality Assessment

Previous research on test case quality has predominantly focused on quantity, operating under the assumption that a greater number of test cases correlates with higher quality [7, 8]. However, recent studies [19] indicate that quantity is not a direct proxy for quality. This discrepancy arises from two primary issues. First, test cases in some older datasets, despite their volume, suffer from significant correctness issues, often violating the problem's explicit constraints. Second, conventional test case generation methods that merely amass large volumes of random data fail to provide adequate coverage of various special and corner cases.

Consequently, we depart from evaluating test cases by their quantity and instead propose a direct assessment of their ability to discriminate between correct and incorrect solutions. In our framework, we conceptualize the entire test suite for a problem as a binary classifier, that is, a classifier that distinguishes between correct and incorrect solutions. We then evaluate the performance of this classifier using a large, curated collection of both correct and incorrect submissions. We adopt the True Positive Rate (TPR) and True Negative Rate (TNR) as our primary evaluation metrics.

$$\text{TPR} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{\text{Number of Passed Correct Solutions}}{\text{Number of Correct Solutions}} \tag{1}$$

$$\text{TNR} = \frac{\text{True Negative}}{\text{True Negative} + \text{False Positive}} = \frac{\text{Number of Rejected Incorrect Solutions}}{\text{Number of Incorrect Solutions}} \tag{2}$$

The TPR measures the **correctness** of the test cases; a high TPR indicates that correct solutions are not erroneously failed, which is expected when the test cases themselves are valid. Conversely, the TNR measures the **comprehensiveness** or **coverage** of the test cases, quantifying their ability to detect (or "hack") incorrect solutions.

By employing a hybrid approach that combines automated generation with expert curation, we have achieved a 100% TPR and 100% TNR on our collected solution set. This signifies that all collected correct solutions pass our test cases, while all collected incorrect solutions are successfully rejected. To the best of our knowledge, AetherCode is the first benchmark that sets such a high standard for test cases.

### 2.3.2 Automatic Construction of Test Cases

We employed the Generator-Validator Agent System [19] to automatically construct the test cases, a methodology whose effectiveness has been well-established in prior research [19]. Building upon this foundation, we incorporated an additional step of manual verification for the Validator. This step ensures the Validator's correctness, thereby guaranteeing that all generated test cases adhere to every constraint specified in the problem description.

Recognizing that the initial Automatic Construction phase could not achieve a 100% TNR on its own, we introduced an additional expert annotation stage to further strengthen the test cases.

#### 2.3.3 Expert Annotation of Test Cases

To this end, we recruited 67 competitive programming experts. The majority of them hold Codeforces ratings above 2000, with one expert exceeding 2600 and achieving the title of International Grandmaster. These experts were tasked with constructing targeted test cases specifically designed to fail the various incorrect solutions we had collected. These manually crafted test cases were then merged with the automatically generated ones to form the final test suite.

Furthermore, we recognized that for certain problems with a limited number of collected incorrect solutions (fewer than 50), achieving a 100% TNR might not sufficiently guarantee the robustness of the test cases. To address this, we subjected the test cases for all problems to a manual quality audit by a specialized review team. Each member of this elite team holds at least three ICPC gold medals and has a minimum of two years of experience in competitive programming problem-setting. Their deep understanding of potential pitfalls and common errors in each problem allows them to leverage their extensive experience to further ensure the quality and comprehensiveness of the test cases.

Additionally, for problems that accept multiple valid outputs, customized judging scripts (a.k.a. checker, or special judge) were provided and thoroughly reviewed by these experts to ensure correct evaluation.

### 2.4 Statistics

The number of problems in different difficulties and years of AetherCode v1 is presented in Table 2. The number of problems in different categories of AetherCode v1 is presented in Table 3.

**Table 2** The number of problems in different difficulties and years of AetherCode v1 (2401-2505).

| Difficulty | | | | Year | |
|---|---|---|---|---|---|
| Easy | Medium | Hard | Extreme | 2024 | 2025 |
| 159 | 145 | 132 | 20 | 400 | 56 |

**Table 3** The number of problems in different categories of AetherCode v1 (2401-2505).

| Category | Basic | Search | DP | Str. | Math | DS | Graph | Geo. | Tech. | Tree |
|---|---|---|---|---|---|---|---|---|---|---|
| Count | 225 | 50 | 110 | 26 | 96 | 120 | 64 | 36 | 147 | 24 |

## 3 Evaluation

Our evaluation includes 8 reasoning models and 5 non-reasoning models. The reasoning models comprise `o4-mini-high` [13], `Gemini-2.5-Pro/Flash` [3], `Seed-1.6-Thinking` [16], `DeepSeek-R1` [4], and `Qwen3` [20], among others. The non-reasoning models consist of `GPT-4.1` [11], `GPT-4o` [12], `Kimi-K2` [18], `DeepSeek-V3` [9], and `Qwen3-Coder`. All models are configured with a maximum output length of 32,768 tokens. Each model is evaluated four times in each problem, and the average results are reported.

### 3.1 Main Result

Table 4 presents a comprehensive performance evaluation of several prominent models on AetherCode. For full results, please refer to the online leaderboard. The analysis yields the following key conclusions:

**Table 4** Performance comparison between reasoning models and non-reasoning models on AetherCode v1 (%, 2401-2505).

| Model | Difficulty | | | | Year | | Pass@ | | |
|---|---|---|---|---|---|---|---|---|---|
| | Easy | Medium | Hard | Extreme | 2024 | 2025 | 1 | 2 | 4 |
| *Reasoning Models* | | | | | | | | | |
| o4-mini-high | 65.3 | 32.1 | 8.0 | 3.8 | 35.8 | 32.6 | **35.5** | 43.0 | 46.6 |
| Gemini-2.5-Pro | 60.1 | 28.6 | 8.5 | 2.5 | 33.7 | 25.0 | **32.7** | 39.8 | 46.0 |
| Seed-1.6-thinking-0715 | 53.9 | 20.2 | 4.7 | 0 | 28.3 | 14.7 | **26.6** | 33.0 | 38.5 |
| DeepSeek-R1-0528 | 46.2 | 16.0 | 3.8 | 0 | 23.4 | 14.3 | **22.3** | 27.4 | 32.4 |
| Gemini-2.5-Flash | 42.1 | 15.2 | 2.7 | 0 | 22.0 | 8.0 | **20.3** | 24.5 | 28.5 |
| Qwen3-235B-A22B | 37.6 | 12.4 | 1.9 | 0 | 19.1 | 7.1 | **17.6** | 21.7 | 25.2 |
| Qwen3-32B | 34.8 | 10.9 | 2.7 | 0 | 17.7 | 6.7 | **16.3** | 20.4 | 23.9 |
| Qwen3-8B | 23.7 | 4.8 | 0.8 | 0 | 11.1 | 2.7 | **10.0** | 13.0 | 15.5 |
| *Non-Reasoning Models* | | | | | | | | | |
| GPT-4.1 | 23.9 | 5.7 | 1.1 | 0 | 11.3 | 4.5 | **10.5** | 13.2 | 15.3 |
| Kimi-K2 | 23.1 | 4.7 | 1.0 | 0 | 10.6 | 4.0 | **9.8** | 12.2 | 14.5 |
| DeepSeek-V3-0324 | 20.8 | 4.0 | 0 | 0 | 8.9 | 5.4 | **8.5** | 10.5 | 12.3 |
| Qwen3-Coder-480B-A35B | 19.7 | 2.2 | 0.6 | 0 | 8.6 | 1.8 | **7.7** | 9.9 | 11.8 |
| GPT-4o | 11.6 | 1.0 | 0.2 | 0 | 4.9 | 1.3 | **4.4** | 5.6 | 7.0 |

**Significant Performance Gap between Models** The performance of `o4-mini-high` and `Gemini-2.5-Pro` is exceptional, establishing a significant performance gap that places them in a tier of their own above other models. Furthermore, they are the only two models capable of successfully solving problems at the "Extremely Difficult" level. Across all difficulty tiers, the performance of these two models substantially surpasses that of their competitors.

**Reasoning Models Comprehensively Outperform Non-Reasoning Models** As anticipated, reasoning models demonstrate markedly superior performance compared to non-reasoning models. For instance, models from the Qwen3 series, such as `Qwen3-32B`, outperform several non-reasoning models despite having fewer parameters. More notably, even with four sampling attempts (*Pass@4*), the performance of non-reasoning models still falls short of that achieved by reasoning models. This phenomenon indicates that for complex tasks like coding competitions, the solution space exploration capabilities of non-reasoning models are constrained, making it difficult to find correct solutions through limited sampling. This bottleneck is particularly pronounced in weaker models.

**Top-Tier Models Exhibit Great Exploration Potential** A comparison of *Pass@1* and *Pass@4* scores reveals that increasing the number of samples yields a more substantial performance improvement for top-tier models. For example, `o4-mini-high`'s score improved by 11.1% (from 35.5% to 46.6%), whereas the weaker `Qwen3-32B` only saw a gain of 7.6% (from 16.3% to 23.9%). Particularly noteworthy is `Gemini-2.5-Pro`, which achieved a remarkable performance increase of 13.3% (from 32.5% to 46.0%). This demonstrates its vast exploration potential in solving complex programming problems, enabling it to generate more diverse and high-quality solutions through multiple attempts.

## 3.2 Performance Across Algorithms

The performance comparison in Table 6 reveals a significant differentiation in model capabilities across various problem categories. All models, regardless of being reasoning or non-reasoning types, uniformly excel at pattern-based tasks such as “Basic Algorithms” and “Strings”. However, their limitations become equally apparent when handling highly abstract problems. Most models struggle to tackle “Computational Geometry” and “Tree Structures”, with the performance of `o4-mini-high` in computational geometry being a notable exception. Furthermore, the shortcomings of non-reasoning models are particularly pronounced, as their capability bottlenecks extend into domains that also demand deep logic and abstract thinking, such as “Dynamic Programming” and “Mathematics”.

## 4 Conclusion

In this paper, we introduced AetherCode, a challenging, rigorously evaluated benchmark purpose-built to assess LLMs' coding and reasoning capabilities. AetherCode distinguishes itself by sourcing all its problems from premier global programming competitions, including OI series and ICPC series, which ensures a high degree of challenge and relevance. Furthermore, it features a comprehensive and meticulously validated suite of test cases, created through a hybrid model of automated generation and expert curation. By validating against a dataset of over 30,000 human submissions, our test suite achieves 100% TPR and 100% TNR on our collected solution set, guaranteeing exceptional accuracy and reliability in evaluation.

Our comprehensive evaluation of several leading-edge models on AetherCode yielded critical insights. We observed a significant performance disparity among models, with top performers like `o4-mini-high` and `Gemini-2.5-Pro` establishing a distinct upper tier. Reasoning models demonstrated a clear and consistent advantage over their non-reasoning counterparts across all difficulty levels, highlighting the crucial role of logical deduction in solving complex algorithmic problems. Overall, even the most advanced models today can only solve a small fraction of problems in AetherCode. This indicates that current LLMs still have considerable room for improvement in reasoning and coding, and there remains a significant gap compared to top human experts.

**Table 5** Category division and detailed tag distribution of AetherCode.

| Category | Tags |
|---|---|
| **Algorithm Basics** | Enumeration, Simulation, Recursion, Greedy, Sorting, Divide and Conquer, Binary Search, Doubling, Recurrence |
| **Search** | DFS, BFS, Bidirectional Search, Heuristic Search, A*, Iterative Deepening Search, IDA*, Dancing Links |
| **Dynamic Programming** | Basic DP, Memorization Search, Knapsack DP, Range DP, DP on DAGs, Tree DP, Bitmask DP, Digit DP, Plug DP, Counting DP, Dynamic DP, Probability DP, DP Optimization |
| **Strings** | String Matching, String Hashing, Trie, Palindrome Automation, Prefix Function, Z-function, Automation, AC Automation, Suffix Array, Suffix Automation, Suffix Balanced Tree, Generalized Suffix Automation, Suffix Tree, Manacher's Algorithm, KMP Algorithm, Sequence Automation, Minimal Representation, Lyndon Factorization, Main-Lorentz Algorithm |
| **Mathematics** | Number Theory, Linear Algebra, Linear Programming, Abstract Algebra, Probability Theory, Game Theory, Young Matrix, Inclusion-Exclusion Principle, Combinatorics, Polynomials |
| **Data Structures** | Stack, Queue, Linked List, Hash Table, Disjoint Set Union, Heap, Block Structure, Monotonic Queue, ST Table, Binary Indexed Tree, Segment Tree, Balanced Tree, Binary Tree & Balanced Tree, Block Decomposition, Persistent Data Structures, Tree-in-Tree, K-D Tree, Cartesian Tree, Huffman Tree, STL-based Data Structure |
| **Graph Theory** | Matrix-Tree Theorem, Directed Acyclic Graph, Topological Sort, Minimum Spanning Tree, Minimum Diameter Spanning Tree, Minimum Tree Spanning, Connectivity, Shortest Path, 2-SAT, Difference Constraints, Hamiltonian Graph, Modular Shortest Path, Graph Coloring, Eulerian Graph, Dominating Tree, Bipartite Graph, Prüfer Sequence, Planar Graph, Chordal Graph, Network Flow, Graph Matching, Random Walk on Graphs, LGV Lemma, Strongly Connected Components |
| **Computational Geometry** | Euclidean Distance, Manhattan Distance, Chebyshev Distance, Pick's Theorem, Triangulation, Convex Hull, Sweep Line, Rotating Calipers, Half-Plane Intersection, Closest Pair of Points, Random Increment Method, Reflection Transformation, Misc. CG |
| **Common Techniques** | Discretization, Two Pointer Technique, Prefix Sum & Difference, Fractional Programming, Randomization, Hanging Line Method, Binary Thinking, Pattern Recognition, Gray Code, Expression Evaluation, Construction, Properties of Bitwise Operations, Conjecture of Conclusions, Interactive Problems, Meet in Middle, Ad-hoc, Uncertainty Algorithms, Square Root Decomposition |
| **Problems on Trees** | LCA, DSU on Tree, Divide and Conquer on Points, Block Decomposition on Tree, Heavy-Light Decomposition, Chain Decomposition, Tree Diameter and Centroid, LCT |

**Table 6** Performance comparison (Pass@1) between reasoning models and non-reasoning models across different categories of algorithmic problems. The abbreviations Basic, Search, DP, Str., Math, DS, Graph, Geo., Tech., Tree represent Algorithm Basics, Search, Dynamic Programming, Strings, Mathematics, Data Structures, Graph Theory, Computational Geometry, Common Techniques, and Problems on Trees, respectively.

| Model | Basic | Search | DP | Str. | Math | DS | Graph | Geo. | Tech. | Tree |
|---|---|---|---|---|---|---|---|---|---|---|
| *Reasoning Models* | | | | | | | | | | |
| o4-mini-high | 38.1 | 28.5 | 27.7 | 35.6 | 31.8 | 25.8 | 28.5 | 27.1 | 26.9 | 7.3 |
| Gemini-2.5-Pro | 36.1 | 24.5 | 24.6 | 29.8 | 31.5 | 25.4 | 26.2 | 18.1 | 23.0 | 7.3 |
| Seed-1.6-thinking | 32.2 | 17.0 | 17.3 | 26.0 | 24.2 | 17.9 | 18.8 | 12.5 | 19.2 | 1.0 |
| DeepSeek-R1-0528 | 26.3 | 16.0 | 14.6 | 23.1 | 19.3 | 16.3 | 15.6 | 10.4 | 13.8 | 7.3 |
| Gemini-2.5-Flash | 24.1 | 16.5 | 11.8 | 19.2 | 16.7 | 16.3 | 17.2 | 13.2 | 11.4 | 4.2 |
| Qwen3-235B-A22B | 22.2 | 13.0 | 8.4 | 20.2 | 13.5 | 11.0 | 12.5 | 11.1 | 9.4 | 4.2 |
| Qwen3-32b | 19.7 | 11.5 | 10.9 | 18.3 | 14.1 | 11.0 | 9.4 | 6.9 | 11.2 | 0 |
| Qwen3-8B | 13.3 | 9.0 | 3.9 | 15.4 | 7.6 | 7.9 | 6.3 | 1.4 | 4.9 | 1.0 |
| *Non-Reasoning Models* | | | | | | | | | | |
| GPT-4.1 | 13.9 | 9.5 | 3.4 | 19.2 | 4.2 | 8.3 | 5.5 | 6.3 | 6.0 | 0 |
| Kimi-K2 | 13.7 | 7.5 | 3.6 | 15.4 | 7.0 | 8.1 | 6.6 | 0.7 | 3.6 | 0 |
| DeepSeek-V3-0324 | 12.1 | 7.0 | 1.8 | 14.4 | 3.9 | 6.3 | 4.3 | 0 | 3.6 | 0 |
| Qwen3-Coder-480B-A35B | 11.1 | 5.5 | 1.8 | 14.4 | 4.2 | 5.2 | 4.3 | 1.4 | 2.9 | 1.0 |
| GPT-4o | 7.2 | 4.5 | 0.7 | 11.5 | 1.6 | 2.9 | 0.4 | 0 | 1.5 | 0 |

**Table 7** Curated Contest Source of AetherCode v1 (2401-2505).

| Competition Name | Category | Date |
|---|---|---|
| Croatian Open Competition in Informatics 2023/2024 Contest #3 | Croatian OI | 2024/1/13 |
| USACO 2024 January Contest (Platinum) | USACO Platinum | 2024/1/26 |
| The 2023-2024 ICPC Southwestern Europe Regional Contest | ICPC Regional Contests | 2024/1/28 |
| Croatian Open Competition in Informatics 2023/2024 Contest #4 | Croatian OI | 2024/2/10 |
| USACO 2024 February Contest (Platinum) | USACO Platinum | 2024/2/16 |
| USACO 2024 US Open Contest (Platinum) | USACO Platinum | 2024/3/15 |
| Singapore National Olympiad in Informatics 2024 Final Contest | NOI (SG) | 2024/3/16 |
| Croatian Open Competition in Informatics 2023/2024 Contest #5 | Croatian OI | 2024/3/16 |
| The 2024 ICPC Latin America Championship | ICPC Regional Championships/Finals | 2024/3/17 |
| The 2024 ICPC Europe Championship | ICPC Regional Championships/Finals | 2024/3/24 |
| The 2024 British Informatics Olympiad Final | British OI | 2024/4/6 |
| Baltic Olympiad in Informatics 2024 Day 1 | Baltic OI | 2024/5/4 |
| Baltic Olympiad in Informatics 2024 Day 2 | Baltic OI | 2024/5/5 |
| Asia-Pacific Informatics Olympiad 2024 (APIO 2024) | APIO | 2024/5/18 |
| The 2024 ICPC North America Championship | ICPC Regional Championships/Finals | 2024/5/27 |
| Central European Olympiad in Informatics 2024 Day 1 (CEOI 2024 Day 1) | Central European OI | 2024/6/25 |
| Central European Olympiad in Informatics 2024 Day 2 (CEOI 2024 Day 2) | Central European OI | 2024/6/27 |
| China National Olympiad in Informatics 2024 Day 1 | NOI | 2024/7/18 |
| China National Olympiad in Informatics 2024 Day 2 | NOI | 2024/7/20 |
| European Girls' Olympiad in Informatics 2024 Day 1 | European Girl's OI | 2024/7/23 |
| European Girls' Olympiad in Informatics 2024 Day 2 | European Girl's OI | 2024/7/25 |
| International Olympiad in Informatics 2024 Day 1 | IOI | 2024/9/3 |
| International Olympiad in Informatics 2024 Day 2 | IOI | 2024/9/5 |
| The 2024 ICPC World Finals Astana | ICPC World Finals | 2024/9/19 |
| The 2024 ICPC Kunming Invitational Contest | ICPC Regional Contests | 2024/9/28 |
| The 2024 Nordic Collegiate Programming Contest | NCPC | 2024/10/5 |
| Croatian Open Competition in Informatics 2024/2025 Contest #1 | Croatian OI | 2024/10/5 |
| CCPC 2024 Harbin Site | CCPC | 2024/10/26 |
| The 2024 ICPC Asia Chengdu Regional Contest | ICPC Regional Contests | 2024/10/27 |
| The 2024 ICPC Asia Nanjing Regional Contest | ICPC Regional Contests | 2024/11/3 |
| Croatian Open Competition in Informatics 2024/2025 Contest #2 | Croatian OI | 2024/11/9 |
| 2024-2025 ICPC Latin American Regional Programming Contest | ICPC Regional Championships/Finals | 2024/11/9 |
| 2024 Rocky Mountain Regional Contest | ICPC Regional Contests | 2024/11/9 |
| 2024 North Central NA Regional Contest | ICPC Regional Contests | 2024/11/9 |
| 2024 Mid-Central USA Programming Contest | ICPC Regional Contests | 2024/11/9 |
| CCPC 2024 Chongqing Site | CCPC | 2024/11/10 |
| The 2024 ICPC Greater NY Regional Contest | ICPC Regional Contests | 2024/11/10 |
| The 2024 ICPC Asia Hangzhou Regional Contest | ICPC Regional Contests | 2024/11/10 |

| Competition Name | Category | Date |
|---|---|---|
| CCPC 2024 Jinan Site | CCPC | 2024/11/16 |
| The 2024 ICPC Pacific Northwest Regional Contest (Div. 1) | ICPC Regional Contests | 2024/11/16 |
| The 2024 ICPC Pacific Northwest Regional Contest (Div. 2) | ICPC Regional Contests | 2024/11/16 |
| ICPC NA South Division 2024 - Division 2 | ICPC Regional Contests | 2024/11/16 |
| ICPC NA South Division 2024 - Division 1 | ICPC Regional Contests | 2024/11/16 |
| The 2024 ICPC Southern California Regional Contest | ICPC Regional Contests | 2024/11/16 |
| The 2024 ICPC Southeastern Europe Regional Contest (SEERC 2024) | ICPC Regional Contests | 2024/11/17 |
| The 2024 ICPC Asia Shanghai Regional Contest | ICPC Regional Contests | 2024/11/17 |
| The 2024 ICPC Asia Seoul Regional Contest | ICPC Regional Contests | 2024/11/23 |
| The 2024 ICPC Northwestern Europe Regional Contest (NWERC 2024) | ICPC Regional Contests | 2024/11/24 |
| The 2024 ICPC Asia Shenyang Regional Contest | ICPC Regional Contests | 2024/11/24 |
| Romanian Master of Informatics 2024 Day 1 | Romanian OI | 2024/11/28 |
| Romanian Master of Informatics 2024 Day 2 | Romanian OI | 2024/11/29 |
| The 2024 ICPC Asia Kunming Regional Contest | ICPC Regional Contests | 2024/12/1 |
| Croatian Open Competition in Informatics 2024/2025 Contest #3 | Croatian OI | 2024/12/12 |
| USACO 2024 December Contest (Platinum) | USACO Platinum | 2024/12/13 |
| The 2024 ICPC Northern Eurasia Finals | ICPC Regional Championships/Finals | 2024/12/15 |
| The 2024 ICPC Central Europe Regional Contest | ICPC Regional Contests | 2024/12/15 |
| CCPC 2024 Zhengzhou Site | CCPC | 2024/12/21 |
| The 2024 ICPC Asia Yokohama Regional Contest | ICPC Regional Contests | 2024/12/22 |
| The 2024 ICPC Asia Hong Kong Regional Contest | ICPC Regional Contests | 2024/12/22 |
| The 2024 ICPC Asia East Continent Final Contest | ICPC Regional Championships/Finals | 2024/12/28 |
| USACO 2025 January Contest (Platinum) | USACO Platinum | 2025/1/24 |
| Croatian Open Competition in Informatics 2024/2025 Contest #4 | Croatian OI | 2025/1/25 |
| The 24th Japanese Olympiad in Informatics Final Round (JOI 2024/2025) | Japanese OI | 2025/2/2 |
| Croatian Open Competition in Informatics 2024/2025 Contest #5 | Croatian OI | 2025/2/15 |
| USACO 2025 February Contest (Platinum) | USACO Platinum | 2025/2/21 |
| The 2025 ICPC Europe Championship | ICPC Regional Championships/Finals | 2025/3/2 |
| 2025 ICPC Asia West Finals | ICPC Regional Championships/Finals | 2025/3/7 |
| The 2025 ICPC Latin America Championship | ICPC Regional Championships/Finals | 2025/3/16 |
| USACO 2025 US Open Contest (Platinum) | USACO Platinum | 2025/3/21 |
| Singapore National Olympiad in Informatics 2025 Final Contest | NOI (SG) | 2025/3/22 |
| The 2025 British Informatics Olympiad Final | British OI | 2025/4/12 |
| Baltic Olympiad in Informatics 2025 Day 1 | Baltic OI | 2025/4/26 |
| Baltic Olympiad in Informatics 2025 Day 2 | Baltic OI | 2025/4/27 |
| The 2025 ICPC China Zhejiang Province Programming Contest (22nd) | ICPC Regional Contests | 2025/5/10 |
| CCPC Final 2024 | CCPC Final | 2025/5/11 |
| Asia-Pacific Informatics Olympiad 2025 (APIO 2025) | APIO | 2025/5/17 |
| The 2025 ICPC Asia Wuhan Invitational Contest | ICPC Regional Contests | 2025/5/17 |
| The 2025 ICPC North America Championship | ICPC Regional Championships/Finals | 2025/5/26 |

## Contributions

### Research & Development

Zihan Wang[1,2], Jiaze Chen[1], Zhicheng Liu[1]

### Management

Markus Mak[1], Yidi Du[1]

### Operations

Geonsik Moon[1], Luoqi Xu[1], Aaron Tua[1]

### Expert Partner

Kunshuo Peng[1], Jiayi Lu[1], Mingfei Xia[1]

### Data Operations

Boqian Zou[1], Chenyang Ran[1], Guang Tian[1], Shoutai Zhu[1], Yeheng Duan[1], Zhenghui Kang[1]

### Data Platform

**Front-End Developer**: Zhenxing Lin[1], Shangshu Li[1]

**Back-End Developer**: Qiang Luo[1], Qingshen Long[1]

**Product Manager**: Zhiyong Chen[1], Yihan Xiao[1]

### Writing

Yurong Wu[1], Daoguang Zan[1]

### Supervision

Yuyi Fu[1], Mingxuan Wang[1], Ming Ding[1]

### Affiliations

[1]ByteDance

[2]M-A-P

## Acknowledgments

We thank Siyao Liu, Jinxin Chi, Haojie Pan, Jingjing Xu, Ge Zhang, Wenhao Huang, Yonghui Wu, as well as other colleagues at ByteDance, and more importantly, the anonymized competitive programming expert team, for their support for the AetherCode project.

## Disclaimer

Your access to and use of this dataset are at your own risk. We do not guarantee the accuracy of this dataset. The dataset is provided "as is" and we make no warranty or representation to you with respect to it and we expressly disclaim, and hereby expressly waive, all warranties, express, implied, statutory or otherwise. This includes, without limitation, warranties of quality, performance, merchantability or fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable.

In no event will we be liable to you on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this public license or use of the licensed material.

The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

## References


[1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

[2] 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.

[3] Google DeepMind. Gemini. https://deepmind.google/models/gemini/.

[4] 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.

[5] 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. In Joaquin Vanschoren and Sai-Kit Yeung, editors, Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c24cd76e1ce41366a4bbe8a49b02a028-Abstract-round2.html.

[6] 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, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

[7] Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in Algorithmic COde generation dataset, December 2023. URL http://arxiv.org/abs/2312.14852. arXiv:2312.14852 version: 3.

[8] 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, 2022. doi: 10.1126/science.abq1158. URL https://www.science.org/doi/abs/10.1126/science.abq1158.

[9] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437, 2024.

[10] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

[11] OpenAI. Gpt-4.1. https://openai.com/index/gpt-4-1/, . Accessed: 2025-04-14.

[12] OpenAI. Gpt-4o. https://openai.com/index/hello-gpt-4o/, . Accessed: 2024-05-13.

[13] OpenAI. o4-mini-high. https://openai.com/index/introducing-o3-and-o4-mini/, . Accessed: 2025-04-16.

[14] 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.

[15] Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, et al. CodeElo: Benchmarking competition-level code generation of llms with human-comparable Elo ratings. arXiv preprint arXiv:2501.01257, 2025.

[16] ByteDance Seed. Doubao-1.6-thinking. https://seed.bytedance.com/en/seed1_6.

[17] Quan Shi, Michael Tang, Karthik Narasimhan, and Shunyu Yao. Can language models solve olympiad programming? arXiv preprint arXiv:2404.10952, 2024.

[18] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. arXiv preprint arXiv:2507.20534, 2025.

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

[20] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

[21] 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.