OpenAI

From Competitive Programming to AI Reasoning Models

Alexander Wei
Researcher, OpenAI

Oleg Murk
Researcher, OpenAI

Sheryl Hsu
Researcher, OpenAI

Competitive Programming at OpenAI

Many at OpenAI have backgrounds in competitive programming.

Jakub Pachocki

Chief Scientist, OpenAI

IOI'09 contestant

Mark Chen

Chief Research Officer, OpenAI

IOI'24, IOI'22 team leader

Competitive Programming at OpenAI

Many at OpenAI have backgrounds in competitive programming.

A few of us are here on-site in Sucre:

Alexander Wei
IOI’15 contestant

Zheng Shao
IOI’99 contestant

Oleg Murk
IOI’95, IOI’96, IOI’97 contestant

Yaroslav Tverdokhlib
IOI’09 contestant

Competitive Programming at OpenAI

Many at OpenAI have backgrounds in competitive programming.

A few of us are here on-site in Sucre:

Alexander Wei
IOI’15 contestant

Zheng Shao
IOI’99 contestant

Oleg Murk
IOI’95, IOI’96, IOI’97 contestant

Yaroslav Tverdokhlib
IOI’09 contestant

Several of us work on **reasoning** —
making models that can **think for longer**

Competitive Programmers and AI Research?

1. **Constant learning.** AI as a field progresses quickly; you are always learning to think in new ways.

Competitive Programmers and AI Research?

1. **Constant learning.** AI as a field progresses quickly; you are always learning to think in new ways.
2. **Hard problem-solving.** That’s research!

Competitive Programmers and AI Research?

1. **Constant learning.** AI as a field progresses quickly; you are always learning to think in new ways.
2. **Hard problem-solving.** That’s research!
3. **Systematic thinking.** Even in deep learning, the right concepts and theoretical foundations go a long way.

Competitive Programmers and AI Research?

1. **Constant learning.** AI as a field progresses quickly; you are always learning to think in new ways.
2. **Hard problem-solving.** That's research!
3. **Systematic thinking.** Even in deep learning, the right concepts and theoretical foundations go a long way.

Every once in a while, I get to solve an algorithm design problem too!

Measuring AI Progress with Competitive Programming

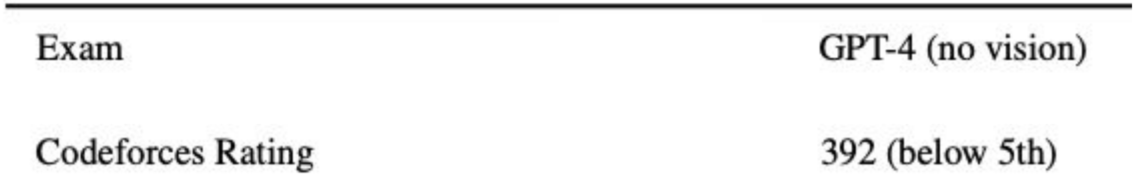
Competitive programming problems are:

1. Difficult — problems require significant **reasoning effort** to solve
2. Objective — **easily verifiable** with hidden test cases

Measuring AI Progress with Competitive Programming

Competitive programming problems are:

1. Difficult — problems require significant **reasoning effort** to solve
2. Objective — **easily verifiable** with hidden test cases

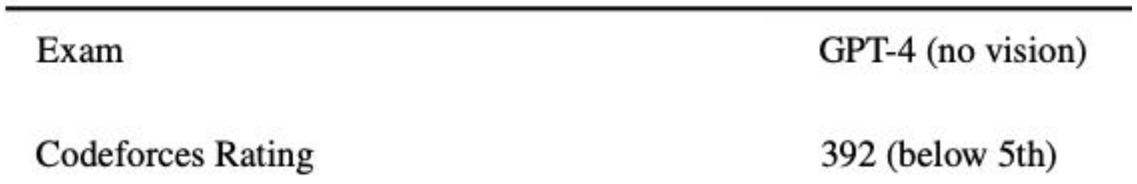


Exam	GPT-4 (no vision)
Codeforces Rating	392 (below 5th)

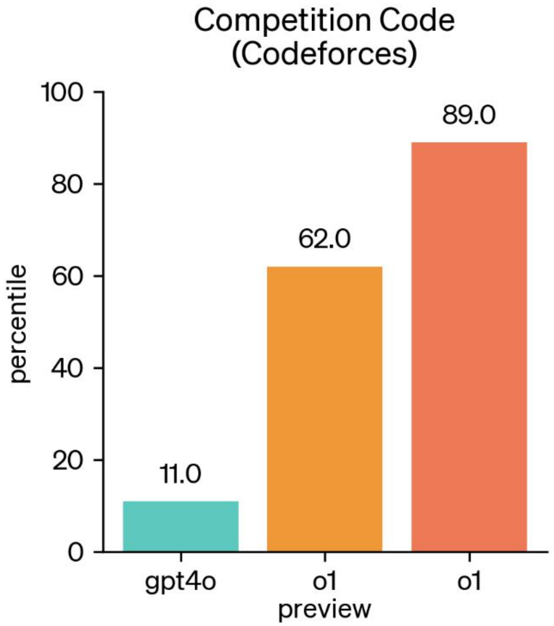
Measuring AI Progress with Competitive Programming

Competitive programming problems are:

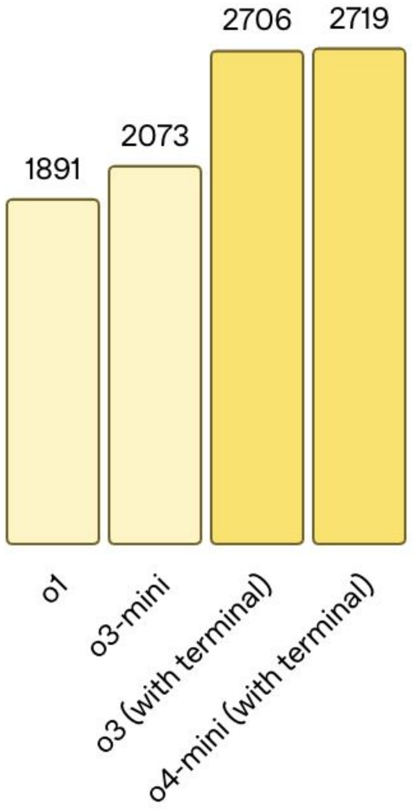
1. Difficult — problems require significant **reasoning effort** to solve
2. Objective — **easily verifiable** with hidden test cases



Exam	GPT-4 (no vision)
Codeforces Rating	392 (below 5th)

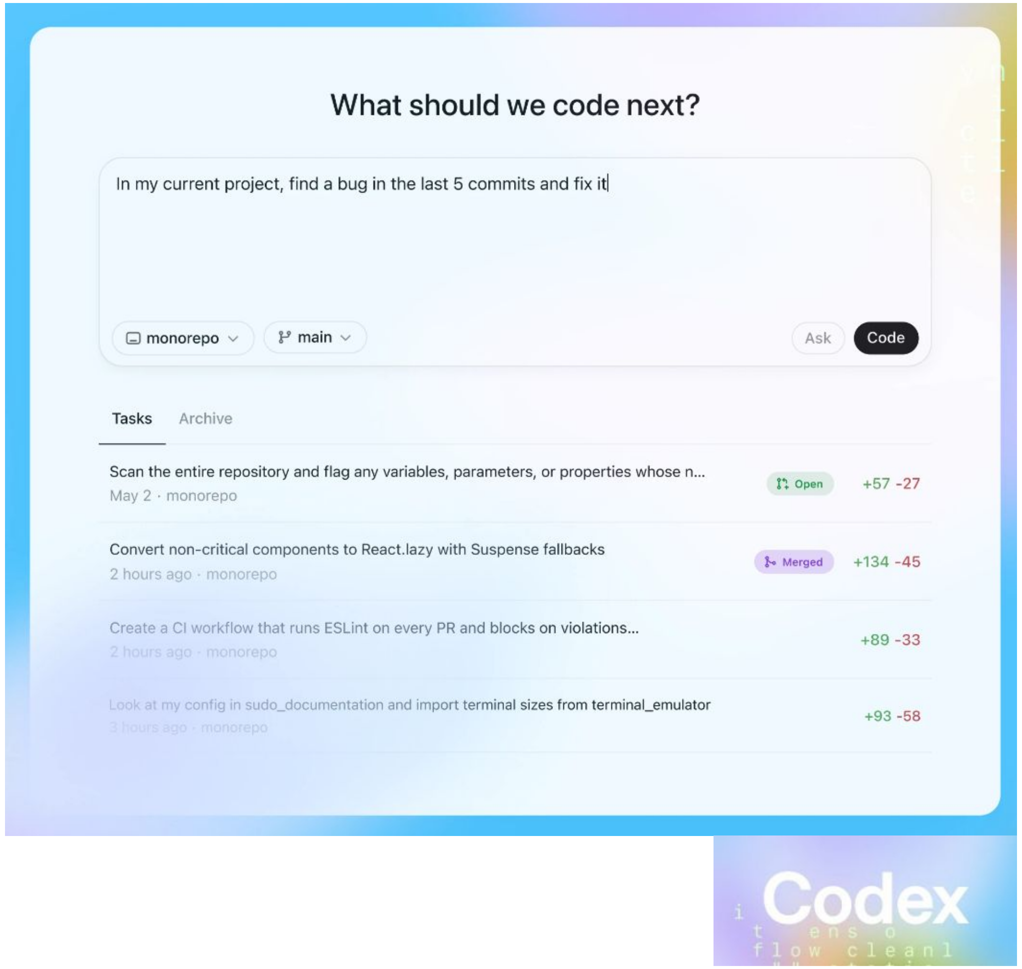


Codeforces Competition Code



AI + Real-world Coding

Progress in reasoning **transfers** to real-world programming.



What should we code next?

In my current project, find a bug in the last 5 commits and fix it

Scan the entire repository and flag any variables, parameters, or properties whose n... Open +57 -27
May 2 · monorepo

Convert non-critical components to React.lazy with Suspense fallbacks Merged +134 -45
2 hours ago · monorepo

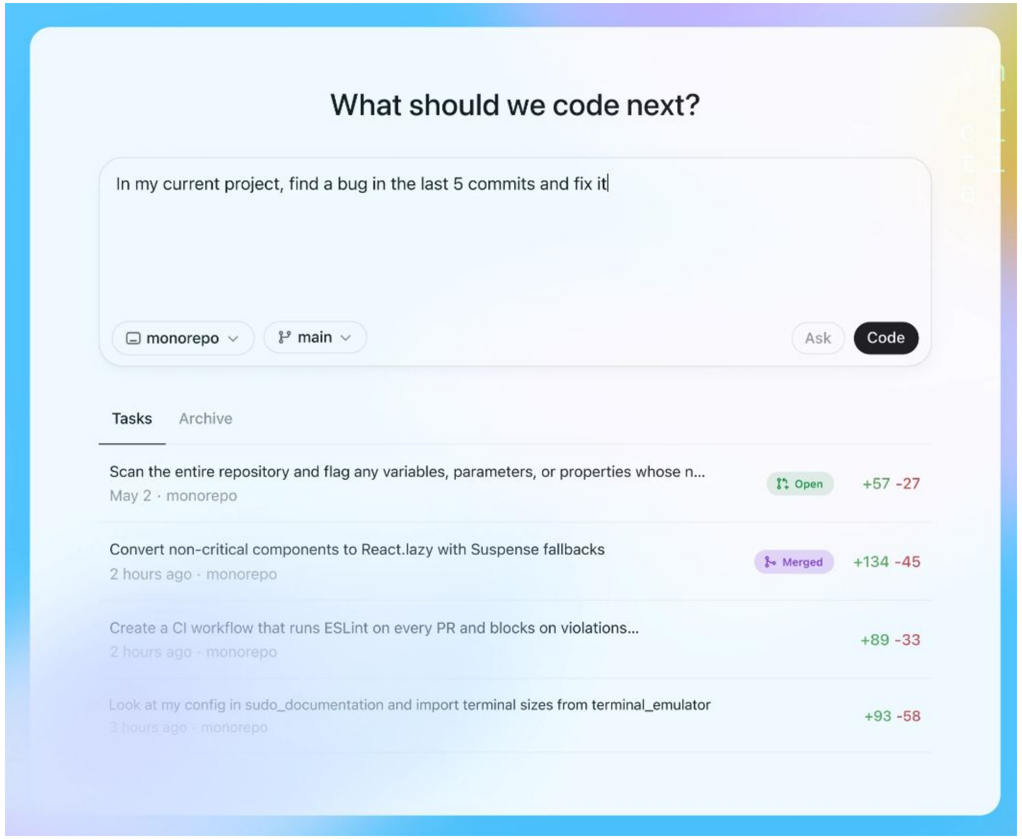
Create a CI workflow that runs ESLint on every PR and blocks on violations... +89 -33
2 hours ago · monorepo

Look at my config in sudo_documentation and import terminal sizes from terminal_emulator +93 -58
3 hours ago · monorepo

AI + Real-world Coding

Progress in reasoning **transfers** to real-world programming.

But real-world problems are also much more complex …



What should we code next?

In my current project, find a bug in the last 5 commits and fix it

Scan the entire repository and flag any variables, parameters, or properties whose n...
May 2 · monorepo
Open
+57 -27

Convert non-critical components to React.lazy with Suspense fallbacks
2 hours ago · monorepo
Merged
+134 -45

Create a CI workflow that runs ESLint on every PR and blocks on violations...
2 hours ago · monorepo
+89 -33

Look at my config in sudo_documentation and import terminal sizes from terminal_emulator
3 hours ago · monorepo
+93 -58

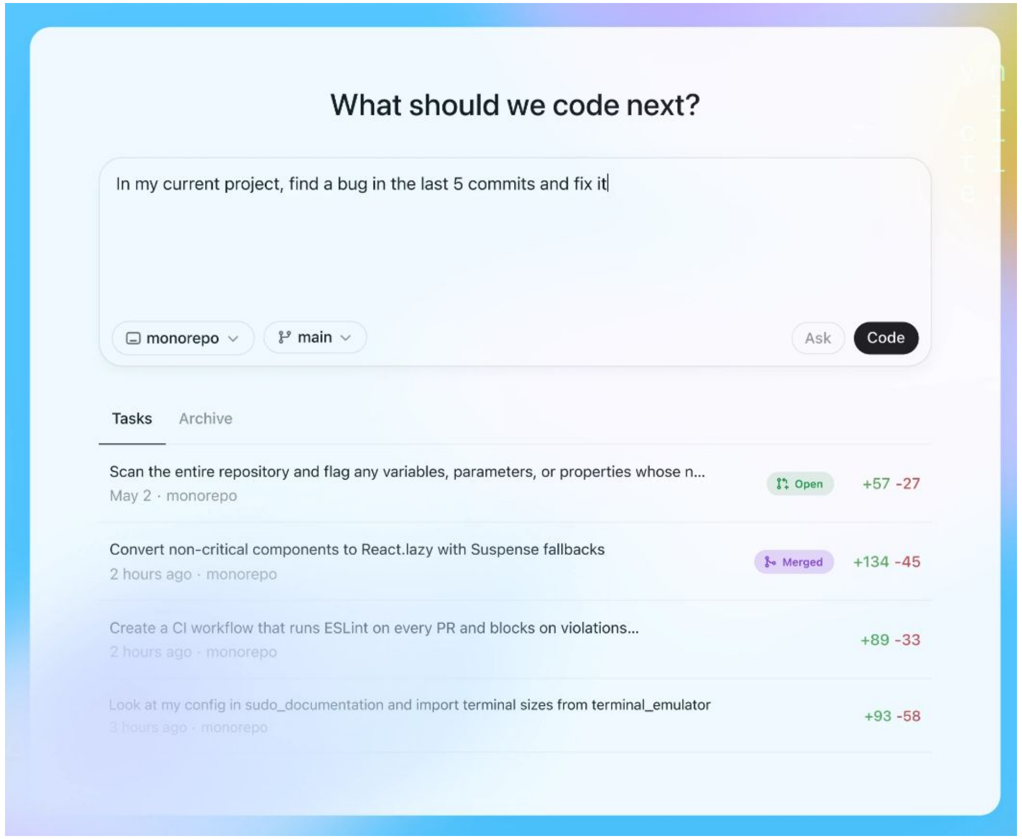
Codeforces Div. 2 A
0.125 hours
(GPT-4, April 2023)

IOI problem
2 hours
(o3, April 2025)

AI + Real-world Coding

Progress in reasoning **transfers** to real-world programming.

But real-world problems are also much more complex …



What should we code next?

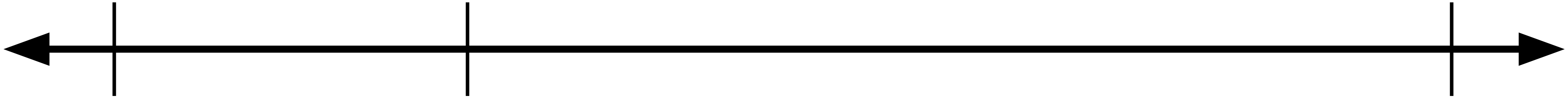
In my current project, find a bug in the last 5 commits and fix it

Scan the entire repository and flag any variables, parameters, or properties whose n...
May 2 · monorepo
Open +57 -27

Convert non-critical components to React.lazy with Suspense fallbacks
2 hours ago · monorepo
Merged +134 -45

Create a CI workflow that runs ESLint on every PR and blocks on violations...
2 hours ago · monorepo
+89 -33

Look at my config in sudo_documentation and import terminal sizes from terminal_emulator
3 hours ago · monorepo
+93 -58



Codeforces Div. 2 A
0.125 hours
(GPT-4, April 2023)

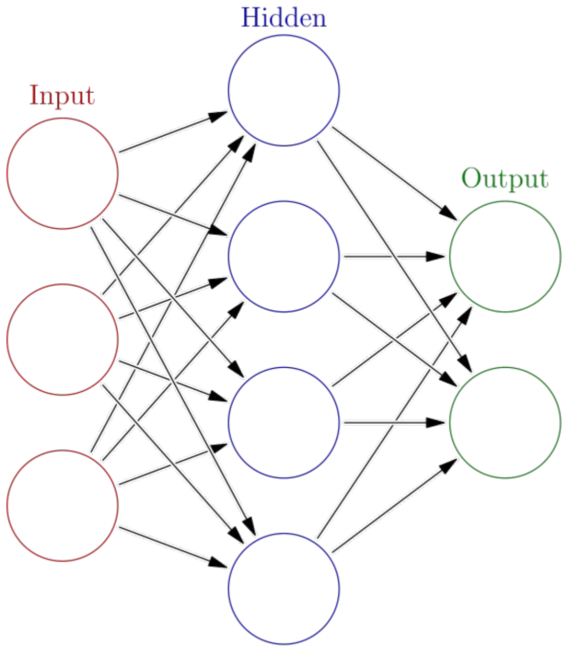
IOI problem
2 hours
(o3, April 2025)

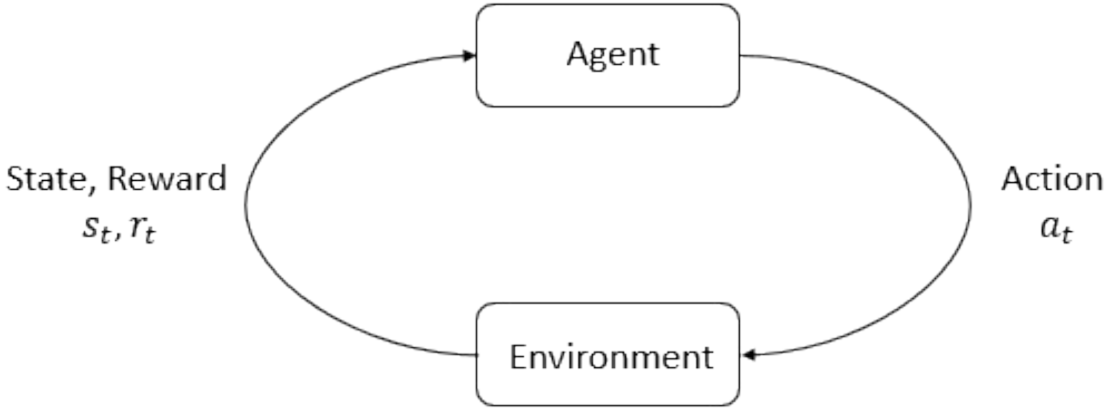
Research breakthrough
8,000 hours
(???)

OpenAI Large Language Models (LLMs)

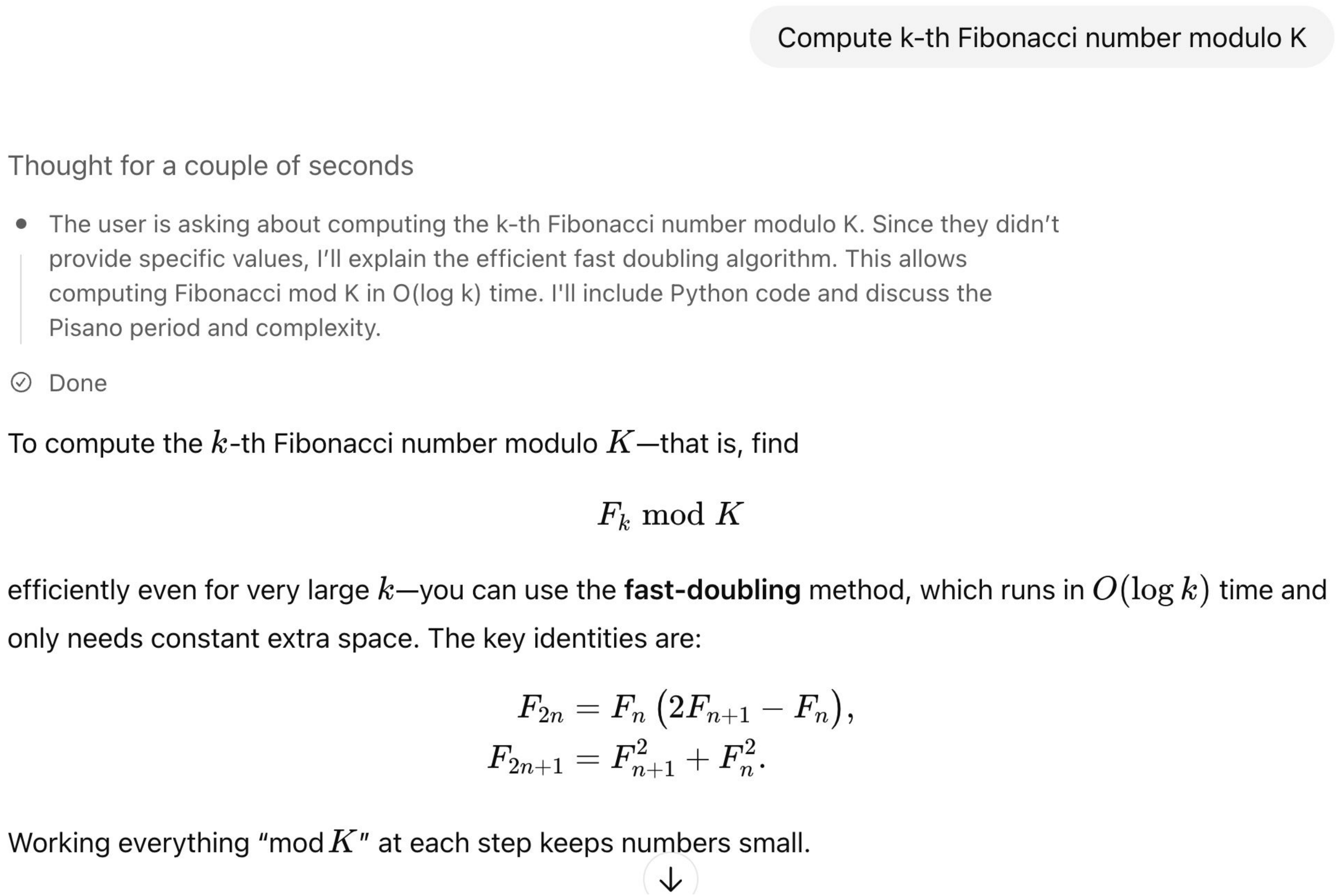
- Base models: GPT4
  - Neural networks
  - Billions of parameters
  - Trained on GPUs
  - Transformer architecture
  - Next word / token prediction loss
  - Optimized using gradient descent
  - Pretrained on a subset of Internet

- Reasoning models: o1, o3
  - Reinforcement Learning (RL)
  - Tasks with Verifiers
  - Chain of Thought (COT)
  - Tool use, eg python





LLM Chain of Thought Example



Compute k-th Fibonacci number modulo K

Thought for a couple of seconds

- The user is asking about computing the k-th Fibonacci number modulo K. Since they didn’t provide specific values, I’ll explain the efficient fast doubling algorithm. This allows computing Fibonacci mod K in O(log k) time. I'll include Python code and discuss the Pisano period and complexity.

✓ Done

To compute the k-th Fibonacci number modulo K—that is, find

$$F_k \bmod K$$

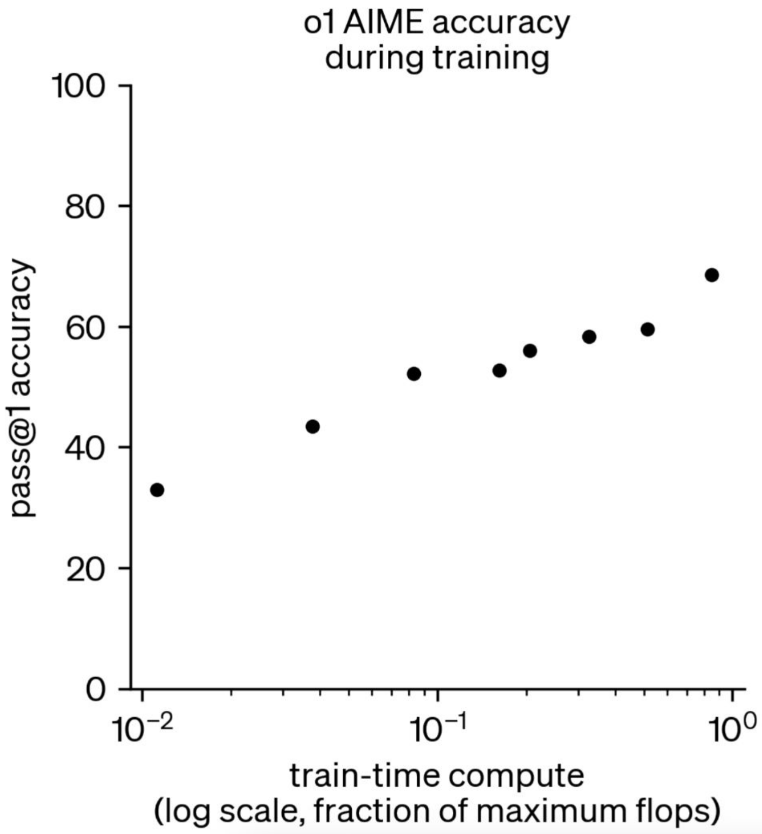
efficiently even for very large k—you can use the **fast-doubling** method, which runs in $O(\log k)$ time and only needs constant extra space. The key identities are:

$$\begin{aligned} F_{2n} &= F_n\left(2F_{n+1} - F_n\right), \\ F_{2n+1} &= F_{n+1}^2 + F_n^2. \end{aligned}$$

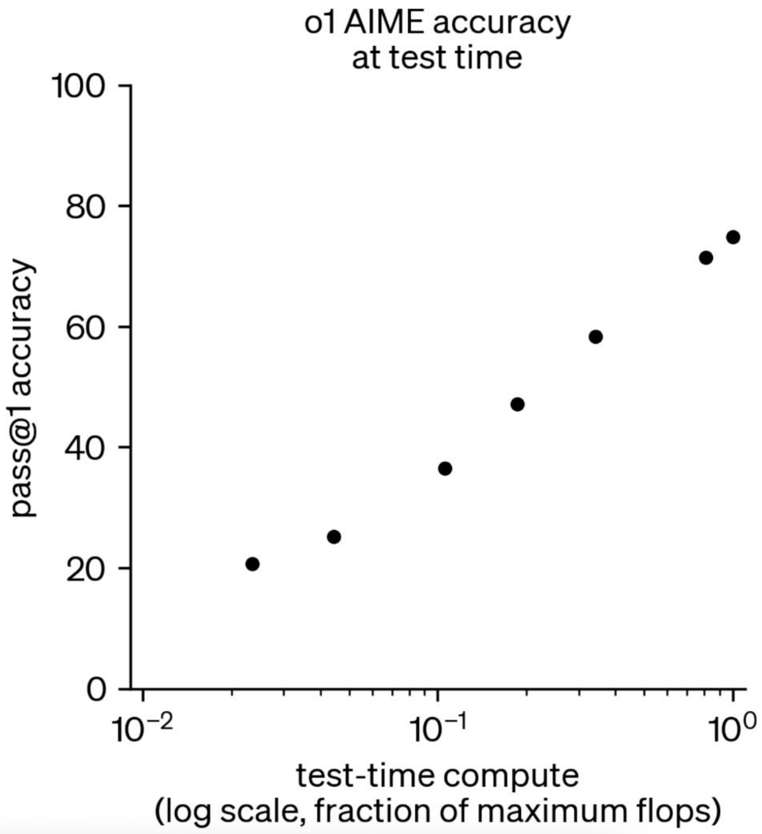
Working everything “mod K” at each step keeps numbers small.

LLM Performance Scaling

- Pre-training performance scales with:
  - Amount of data
  - Model size
  - Train-time compute
- RL performance scales with:
  - Amount of tasks
  - Train-time compute
  - Test-time compute
- Test-time strategies:
  - Multiple samples
  - Consensus
  - Ranking function
  - …



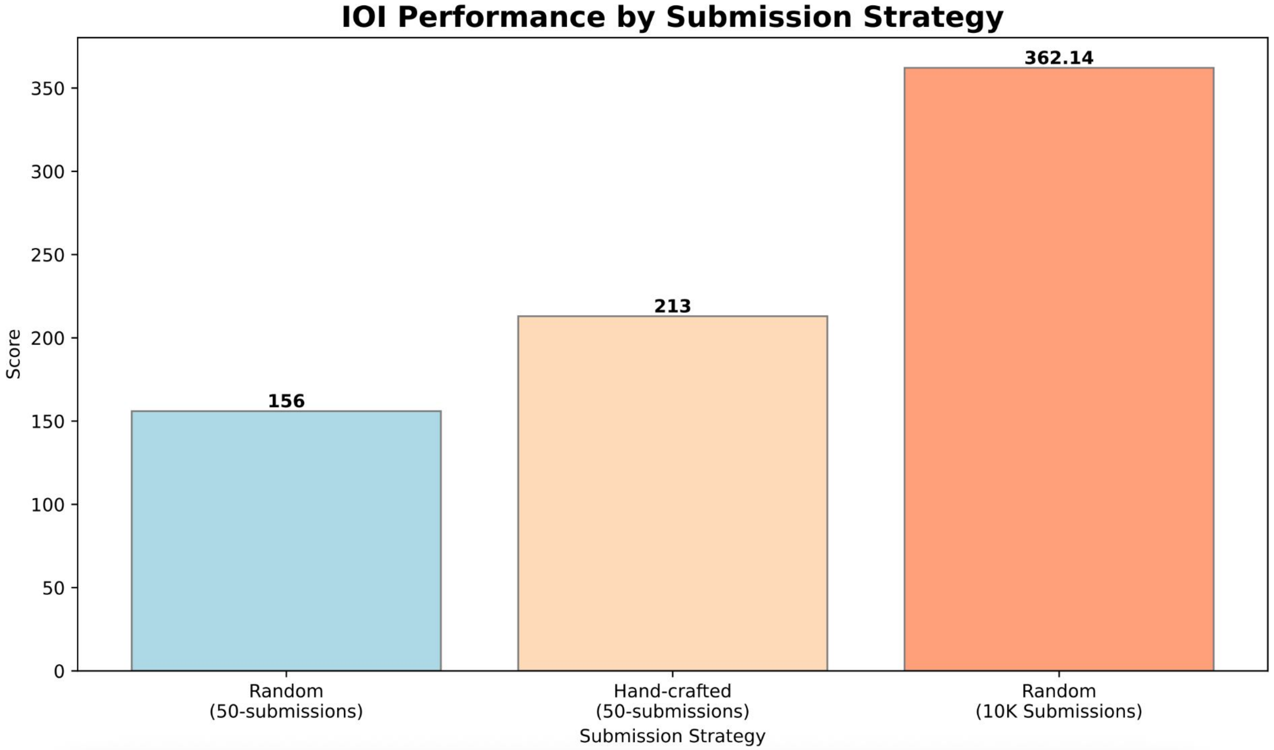
o1 AIME accuracy during training



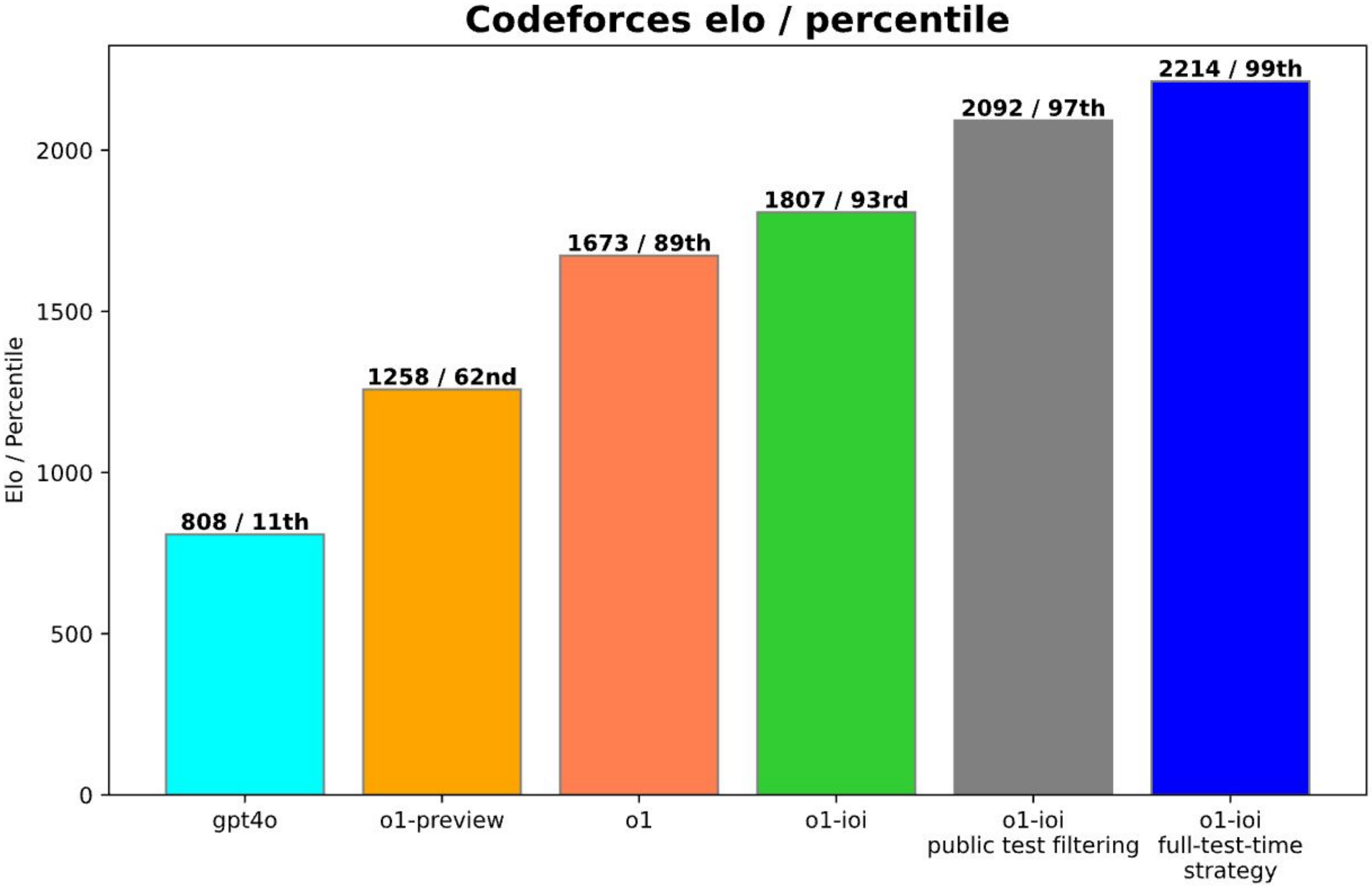
o1 AIME accuracy at test time

o1-ioi (September 2024)

- Train custom model for IOI starting from o1
  - Programming tasks + Test suites
- Sample 10K solutions
  - Split problems into separately gradable parts
- Filter by public tests
- Generate test cases
  - Sample generators from o1
- Cluster solutions
  - That produce the same test outputs
- Rank clusters and submit top 50
  - Learned ranking function

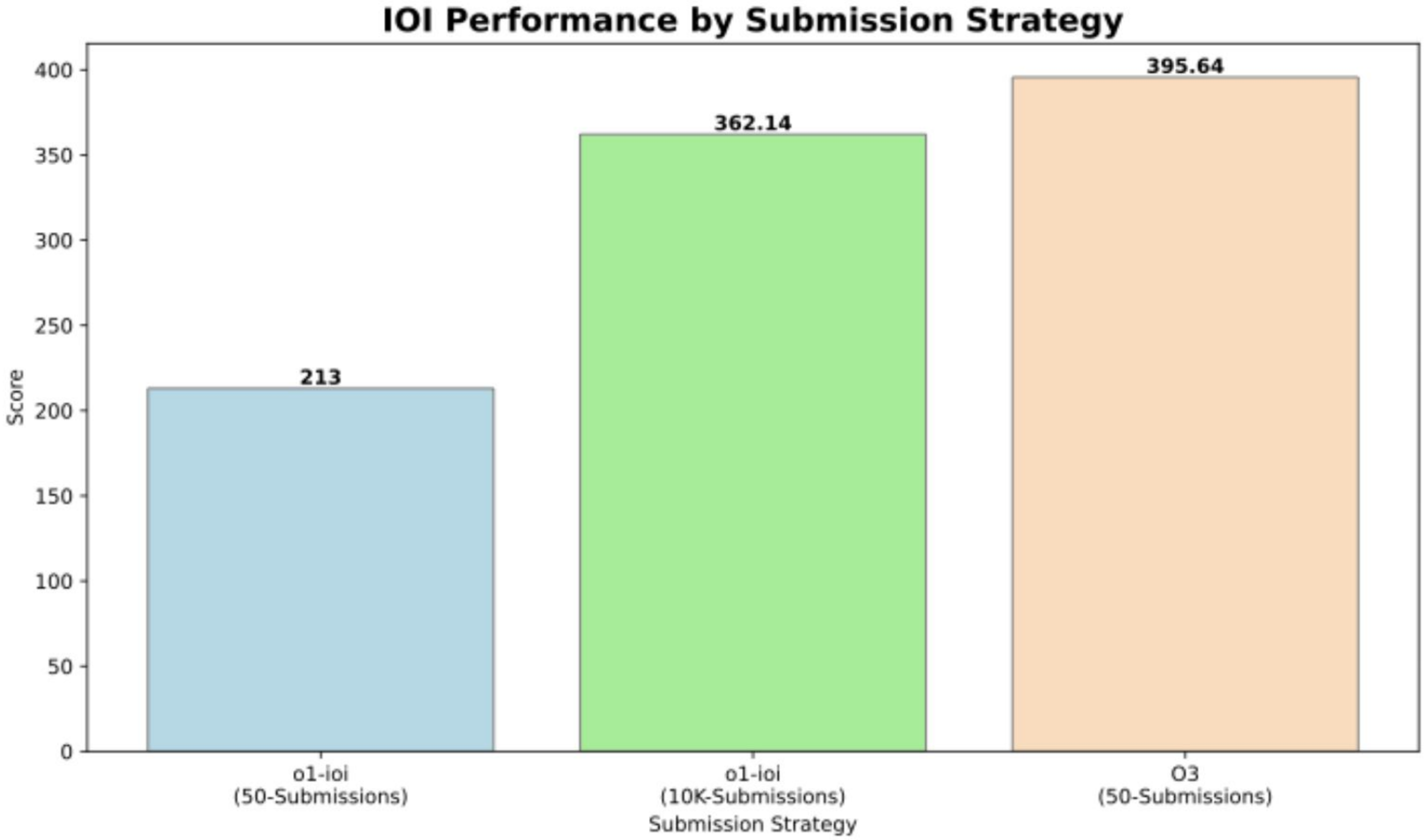


o1-ioi (September 2024)

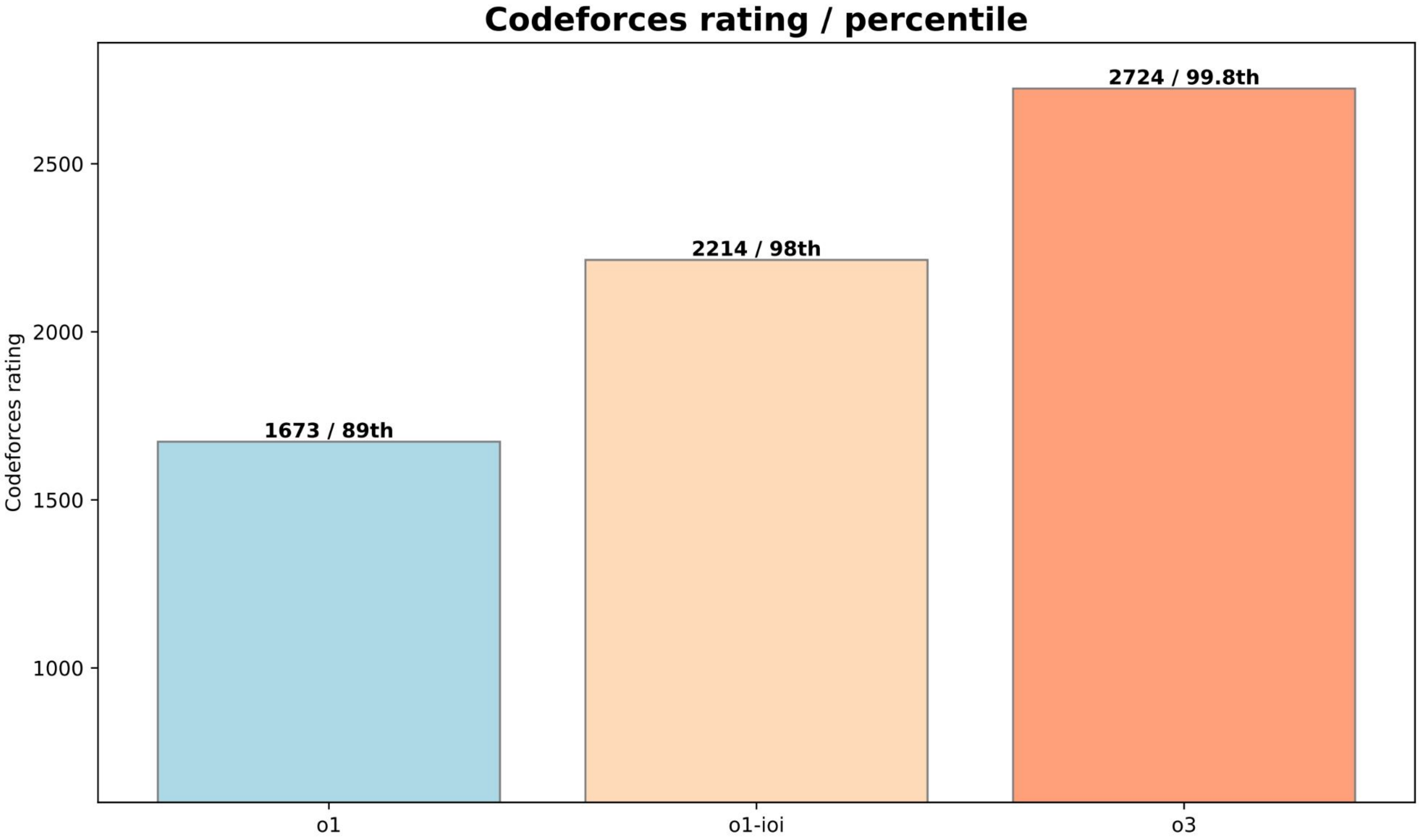


o3-preview (December 2024)

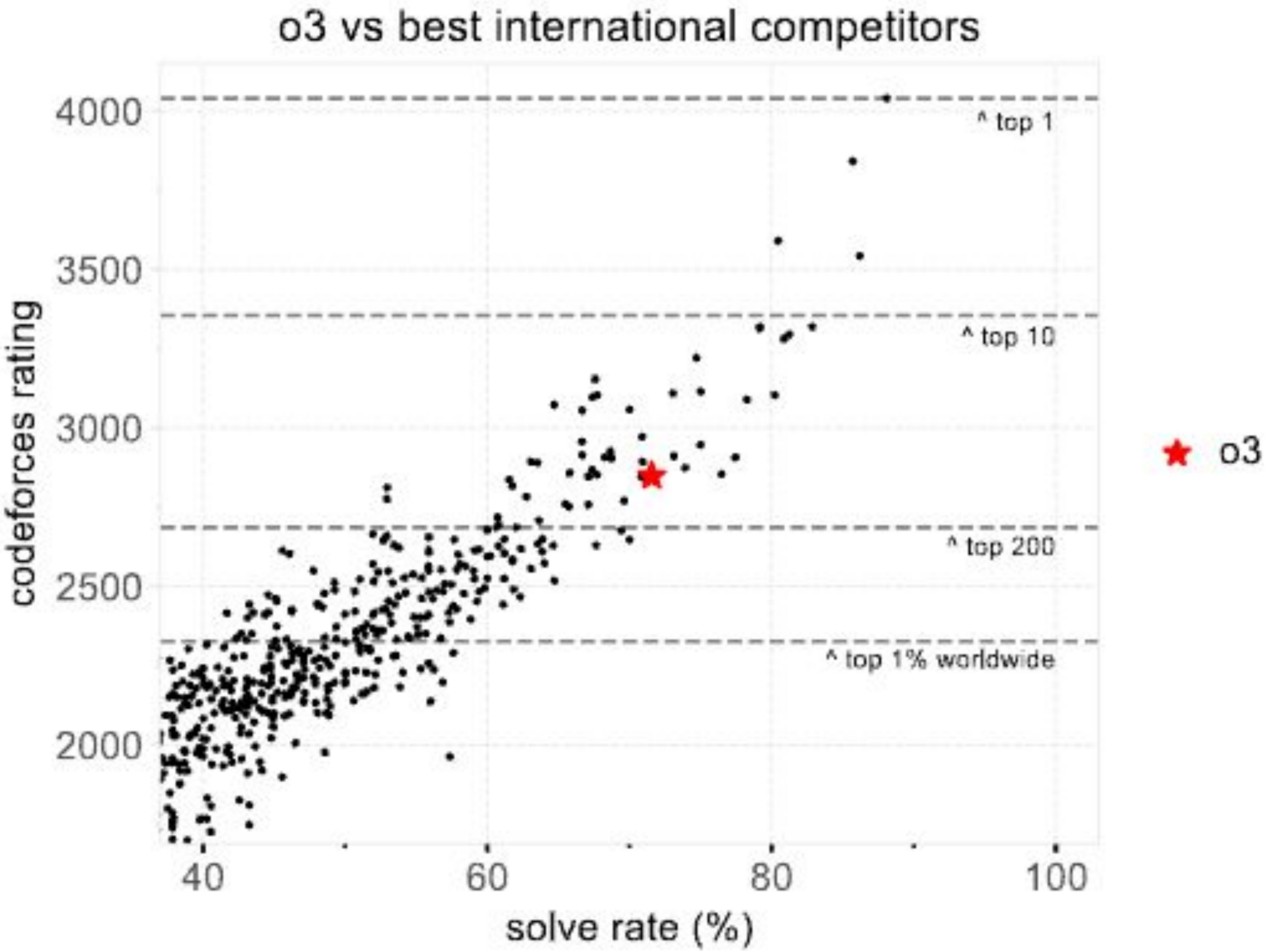
- Use generic o3-preview model for IOI
- Sample 1024 solutions
- Pick top 50 solutions by longest test-time compute



o3-preview (December 2024)

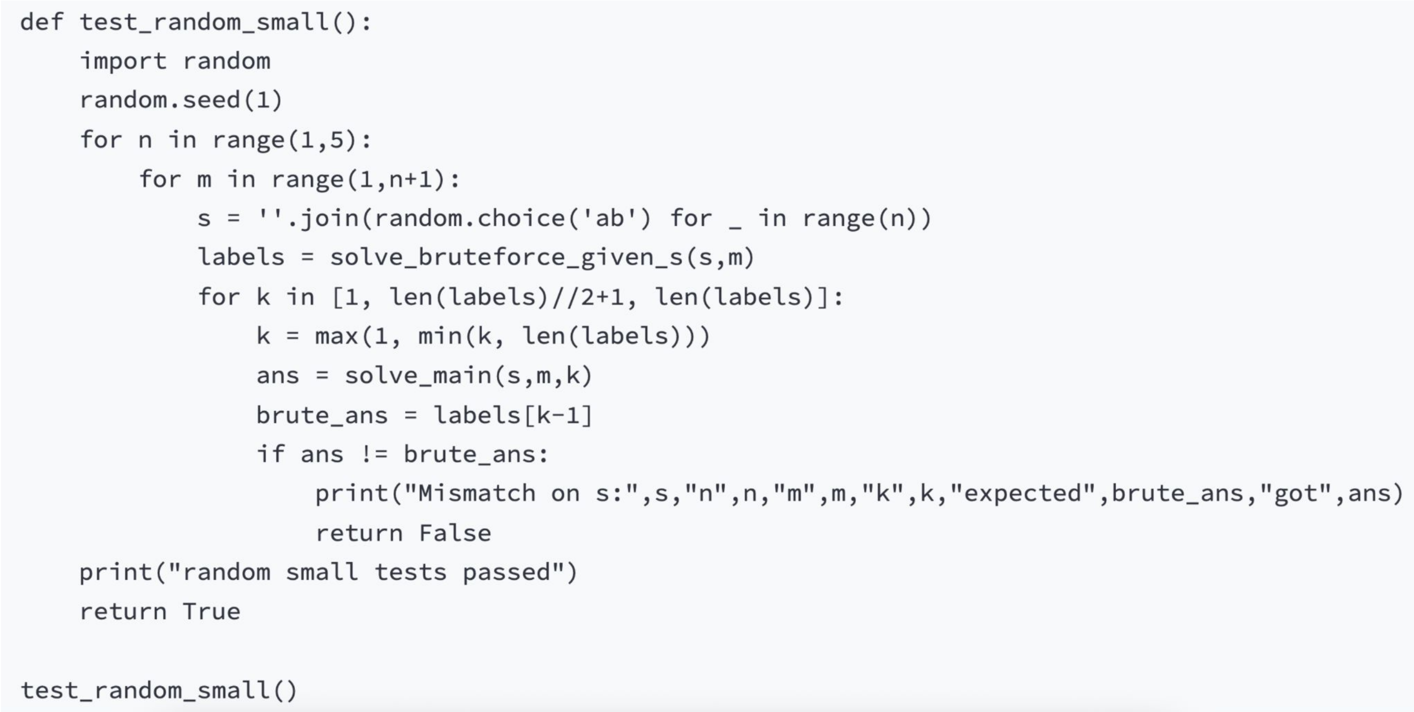


o3-preview (December 2024)



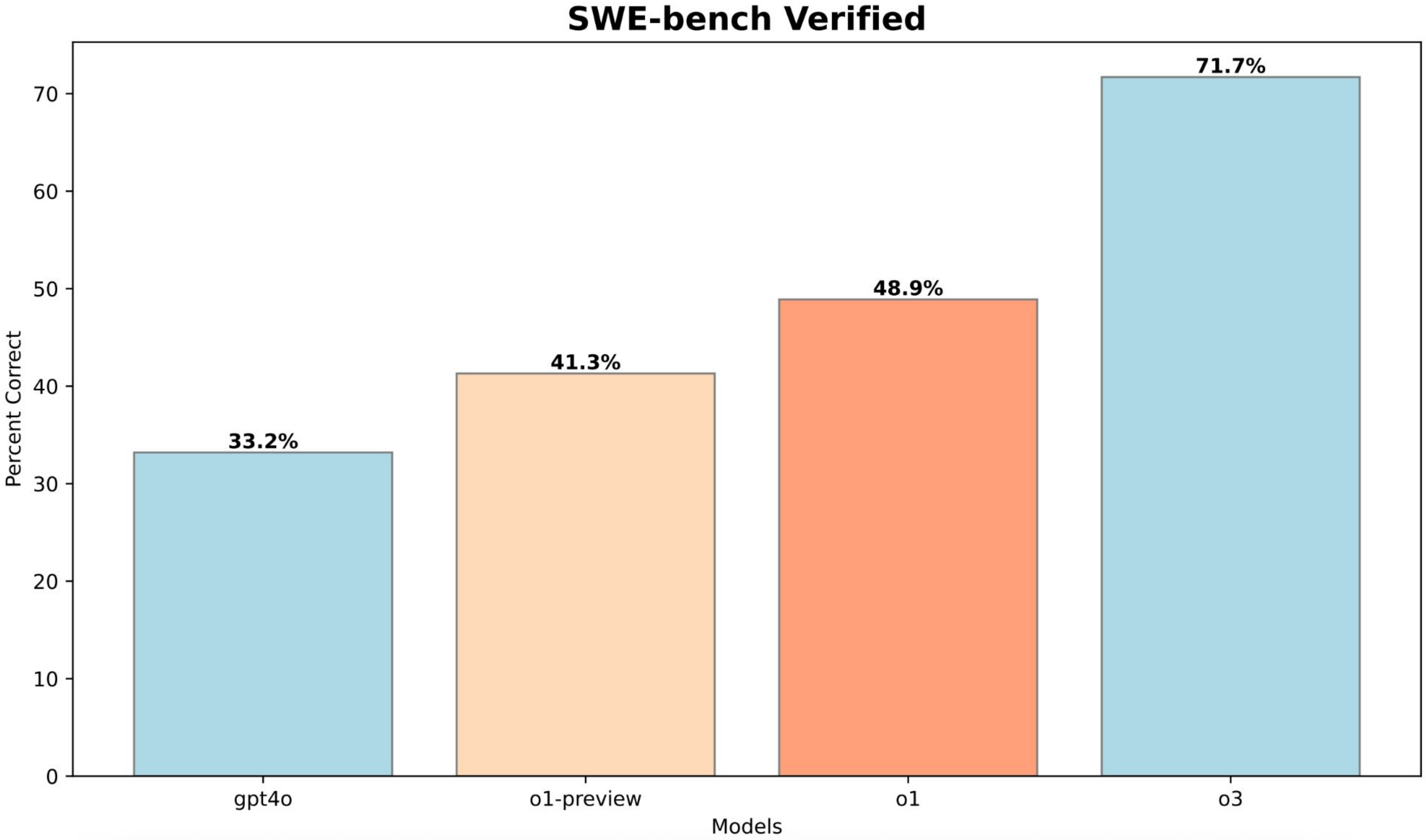
o3-preview (December 2024)

- Emergent behaviors
    - Error correction
    - Trying multiple strategies
    - Breaking down problems
    - Writing tests & comparing to slower solution



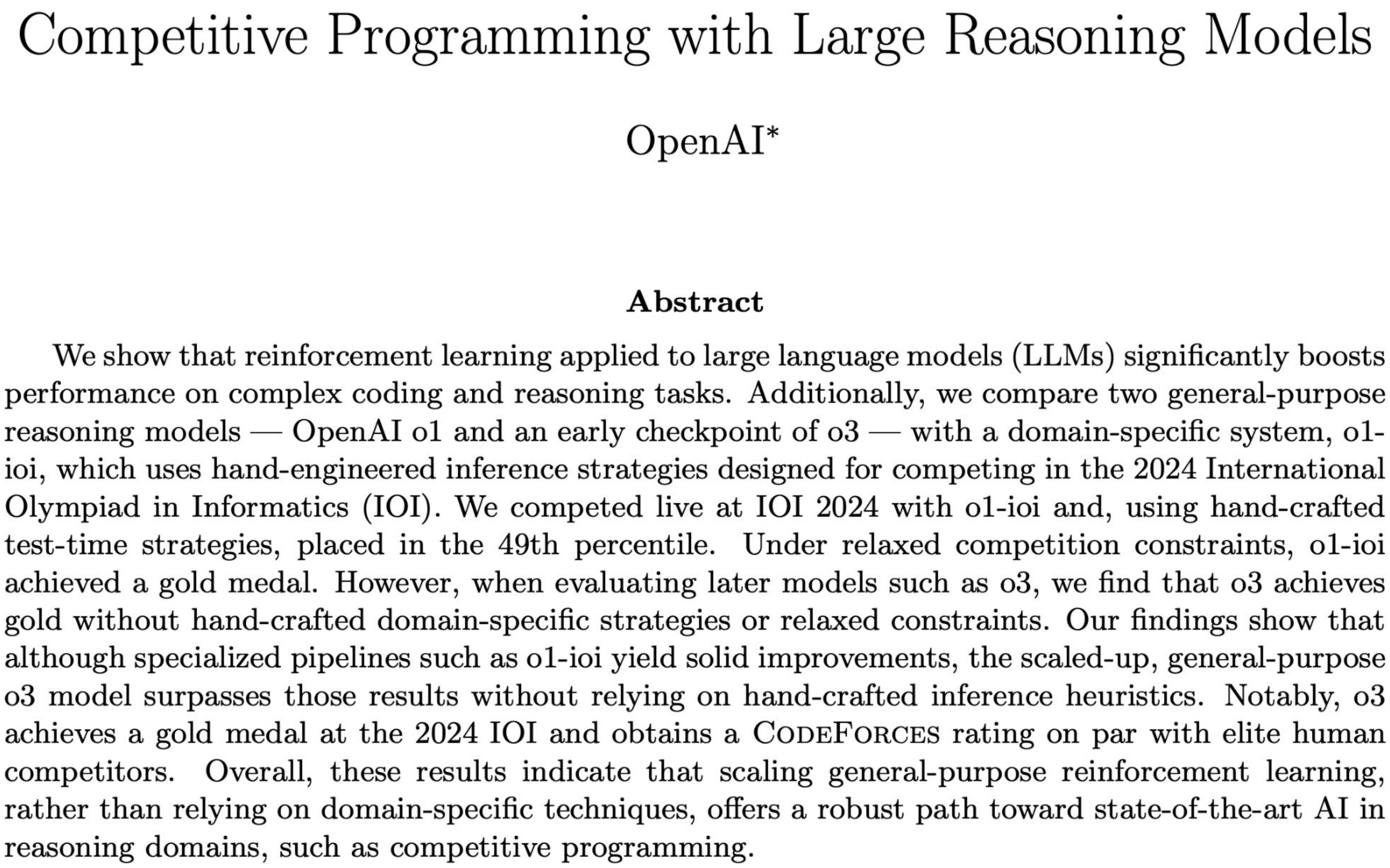
```
def test_random_small():
    import random
    random.seed(1)
    for n in range(1,5):
        for m in range(1,n+1):
            s = ''.join(random.choice('ab') for _ in range(n))
            labels = solve_bruteforce_given_s(s,m)
            for k in [1, len(labels)//2+1, len(labels)]:
                k = max(1, min(k, len(labels)))
                ans = solve_main(s,m,k)
                brute_ans = labels[k-1]
                if ans != brute_ans:
                    print("Mismatch on s:",s,"n",n,"m",m,"k",k,"expected",brute_ans,"got",ans)
                    return False
    print("random small tests passed")
    return True

test_random_small()
```

SWE-bench: real-world software issues from GitHub



LLM Reasoning Challenges

- Compute-hungry
- Training data is limited
- Reward / verifier hacking
- Models generalize … but not as good as humans
- Hence new types of IOI problems can be more challenging



Competitive Programming with Large Reasoning Models

OpenAI*

Abstract

We show that reinforcement learning applied to large language models (LLMs) significantly boosts performance on complex coding and reasoning tasks. Additionally, we compare two general-purpose reasoning models — OpenAI o1 and an early checkpoint of o3 — with a domain-specific system, o1-ioi, which uses hand-engineered inference strategies designed for competing in the 2024 International Olympiad in Informatics (IOI). We competed live at IOI 2024 with o1-ioi and, using hand-crafted test-time strategies, placed in the 49th percentile. Under relaxed competition constraints, o1-ioi achieved a gold medal. However, when evaluating later models such as o3, we find that o3 achieves gold without hand-crafted domain-specific strategies or relaxed constraints. Our findings show that although specialized pipelines such as o1-ioi yield solid improvements, the scaled-up, general-purpose o3 model surpasses those results without relying on hand-crafted inference heuristics. Notably, o3 achieves a gold medal at the 2024 IOI and obtains a CODEFORCES rating on par with elite human competitors. Overall, these results indicate that scaling general-purpose reinforcement learning, rather than relying on domain-specific techniques, offers a robust path toward state-of-the-art AI in reasoning domains, such as competitive programming.

https://arxiv.org/abs/2502.06807