Gemini 2.5 Pro Capable of Winning Gold at IMO 2025*

Yichen Huang (黄溢辰)† Lin F. Yang (杨林)‡

Presented by:

John Tan Chong Min

What is the IMO?

- The International Mathematical Olympiad (IMO) is an esteemed annual competition that convenes the world's most talented pre-university mathematicians

- IMO challenges participants with **exceptionally difficult problems in fields like algebra, geometry, number theory, and combinatorics**

- Contestants are given two **4.5-hour sessions over two days** to solve three problems per session, each graded out of seven points

Why use IMO?

- Other math benchmarks like GSM8K and MATH focus on grade-school and high-school level problems, respectively, where LLMs have achieved high performance through pattern recognition and retrieval from training data

- IMO problems surpass these in complexity, **requiring multi-step reasoning, abstraction, and innovation** akin to human expert-level cognition

OpenAI first to claim IMO Gold (non-IMO-verified)

Official Google Blog (IMO-verified)

> “
> "We can confirm that Google DeepMind has reached the much-desired milestone, earning 35 out of a possible 42 points — a gold medal score. Their solutions were astonishing in many respects. IMO graders found them to be clear, precise and most of them easy to follow."
>
> IMO PRESIDENT PROF. DR. GREGOR DOLINAR

https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/

Pre-LLM Dominant Approach for Math: Lean: Using domain-language to prove math theorems

```
1   import data.nat.basic
2   example (m n k : ℕ) (h₀ : n ≤ m) : n + k ≤ m + k := begin
3     induction k,
4     {
5       exact h₀
6     },
7     {
8       rw nat.succ_le_succ_iff,
9       exact k_ih
10    }
11  end
```

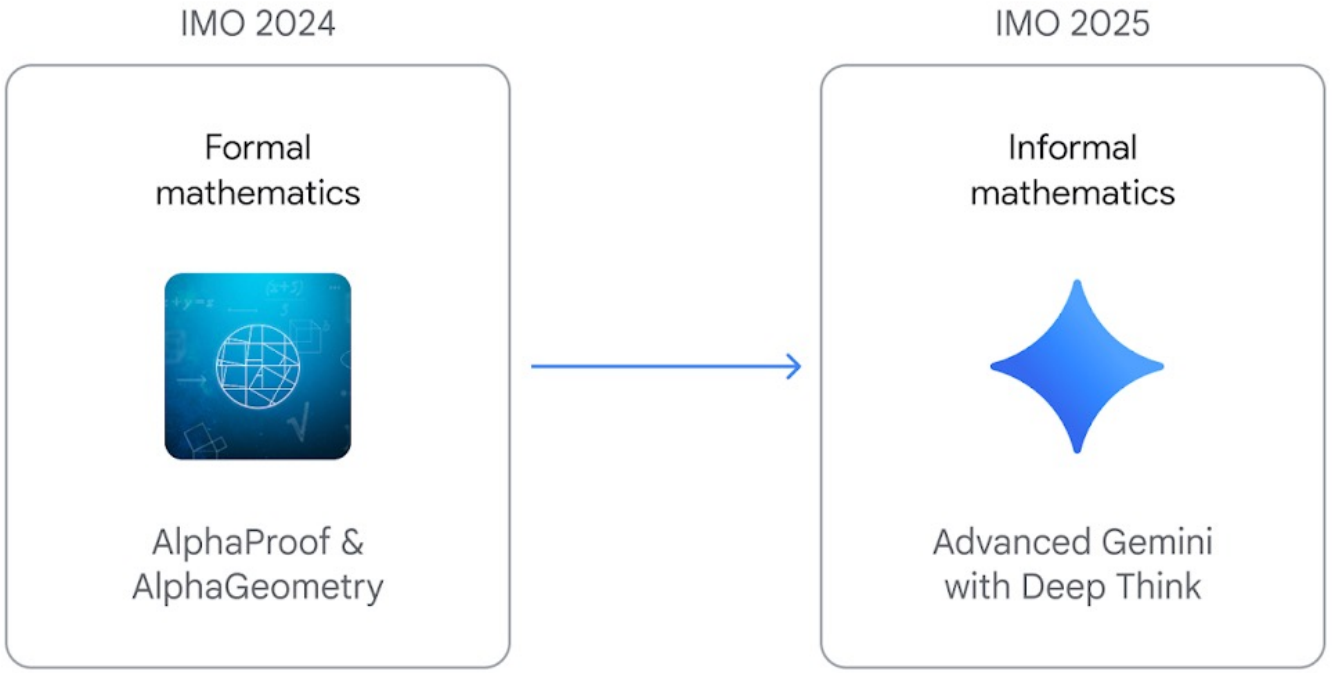
First subgoal : $n + 0 \leq m + 0$

Second subgoal : $n + k \leq m + k \Rightarrow n + k + 1 \leq m + k + 1$

Figure 1: **A simple proof of the statement $n \leq m \Rightarrow n + k \leq m + k$ in Lean.** The *induction* tactic reduces the initial statement to two subgoals, that can be solved independently.

HyperTree Proof Search for Neural Theorem Proving. Guillame et al. 2022. (Meta AI Research)

Domain-Specific-Language to Natural Language

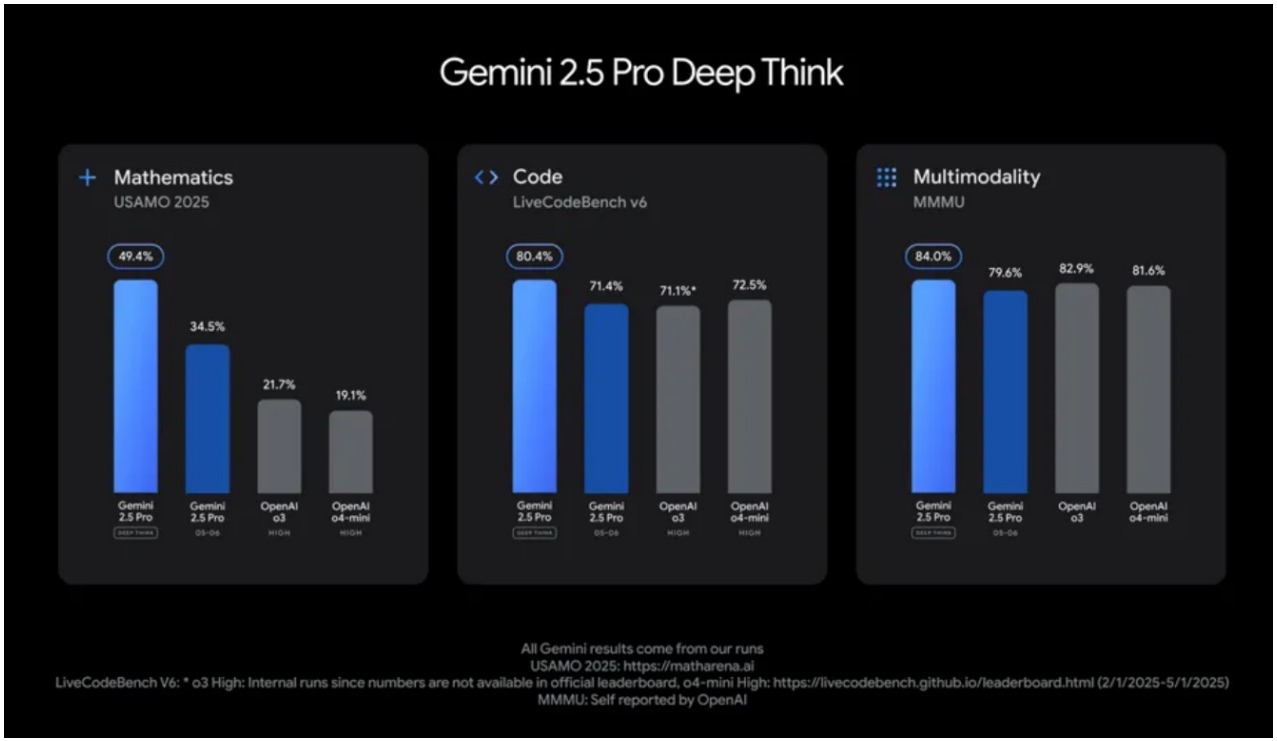


Silver Performance
2-3 days compute
Domain-Specific Language (e.g. Lean)

Gold Performance
Within 4.5 hours competition limit
Natural Language

https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/

Deep Think: Considers multiple hypotheses before responding



- DeepThink could be a form of parallel thinking (my speculation):
  - Generate *N* trajectories of potential output from input prompt
  - Evaluate all of them with an evaluator (potentially LLM)
  - Combine most promising answers together by using it as context in another LLM call

https://blog.google/technology/google-deepmind/google-gemini-updates-io-2025/#deep-think

Post-LLM Dominant Approach for Math: Using LLM + parallel workflows + better reasoning datasets + hints

We achieved this year's result using an advanced version of Gemini Deep Think – an enhanced reasoning mode for complex problems that incorporates some of our latest research techniques, including parallel thinking. This setup enables the model to simultaneously explore and combine multiple possible solutions before giving a final answer, rather than pursuing a single, linear chain of thought.

To make the most of the reasoning capabilities of Deep Think, we additionally trained this version of Gemini on novel reinforcement learning techniques that can leverage more multi-step reasoning, problem-solving and theorem-proving data. We also provided Gemini with access to a curated corpus of high-quality solutions to mathematics problems, and added some general hints and tips on how to approach IMO problems to its instructions.

https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/

AlphaEvolve: Human-in-the-loop self-improvement

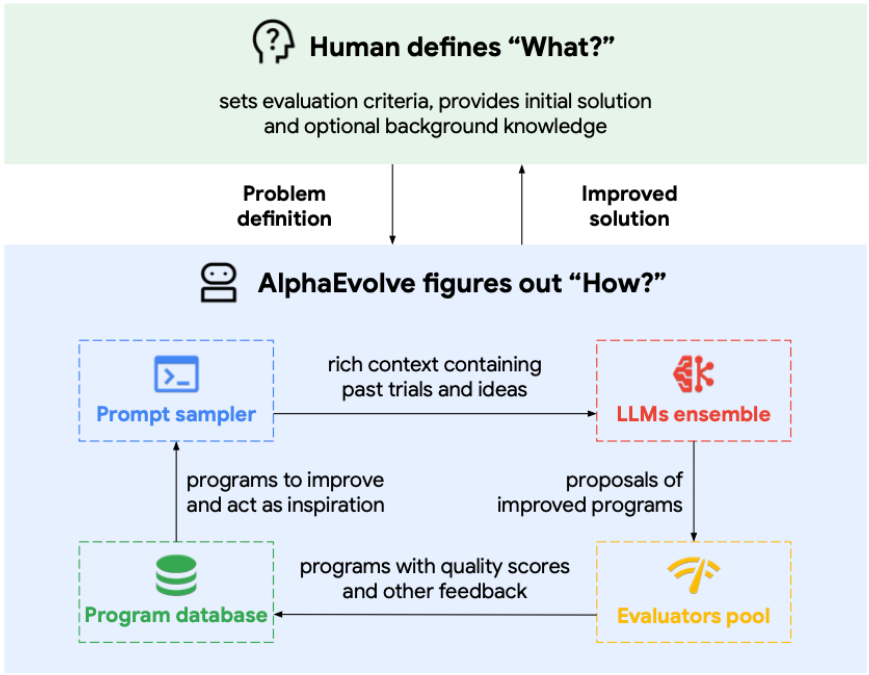


Figure 1 | *AlphaEvolve* high-level overview.

- Possible to use evolutionary-based method to generate + evaluate solutions
- **BUT:** This still requires **human-in-the-loop knowledge** to make it work well, since math problems have no clear verifier (unless one uses a DSL like Lean) if one does not have the ground truth

AlphaEvolve: A coding agent for scientific and algorithmic discovery. Google DeepMind. 2025.

How did it work?

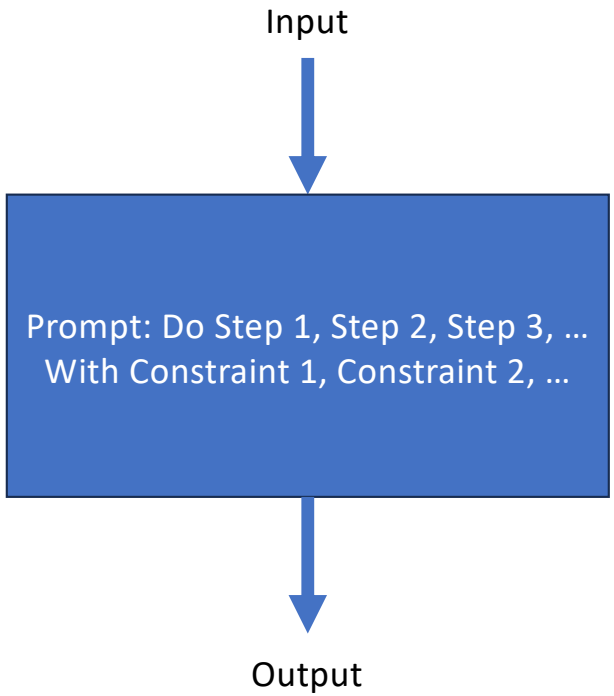
Potentially LLM + Multiple Sampling and Filtering + Verifier Generating Errors + Iterative Correction of Errors

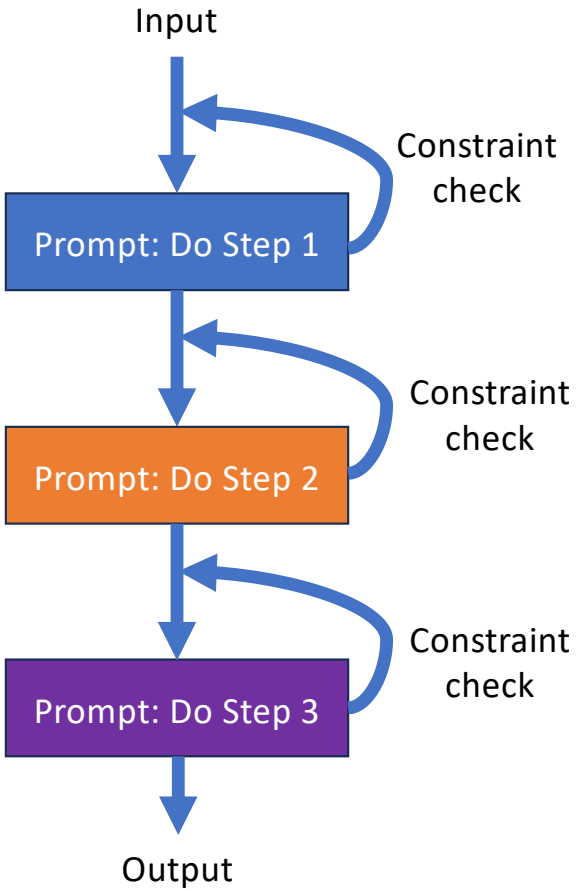
Insights from: Gemini 2.5 Pro Capable of Winning Gold at IMO 2025

(Yichen and Lin, 2025)

Overall Insight

From my previous video:
Opt for simpler, modular workflows





Overall Workflow: Using LLMs at each part of pipeline to iteratively generate solutions, improve, generate errors and correct errors

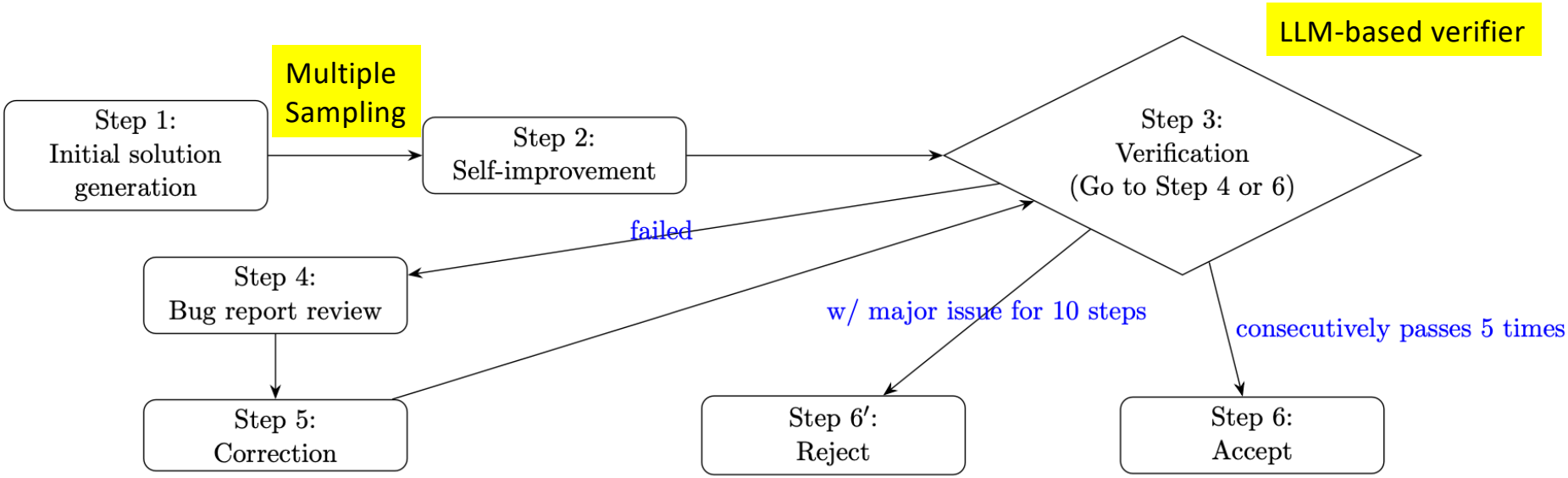


Figure 1: Flow diagram of our pipeline. See the main text for detailed explanations of each step.

Step 1: Initial solution generation

- Initially, we run the model (Gemini 2.5 Pro) some number of times and obtain some initial solution samples to the problem
- The sampling step is analogous to exploration - hope that at least one or more samples have some overlap with the correct approach
- **Some hints may be given to help guide solution**: E.g. Q1 use induction, Q2 use analytic geometry
- Prompt **emphasizes rigor** rather than focus on finding the final answer
  - My insight: LLMs often cannot gauge accurately if the answer is correct

Step 2: Self-improvement (More like continued thinking)

- The maximum number of thinking tokens of Gemini 2.5 Pro is 32768, which is not enough for solving a typical IMO problem

- **Allow model another thinking budget of 32768 tokens** to continue the output generated in Step 1

Step 3: Verification (by LLM)

- Use the verifier to generate a bug report for each solution outputted in Step 2

- The bug report contains a **list of issues** classified as critical errors or justification gaps

- **For each issue, an explanation is required.** The bug report will serve as useful information for the model to improve the solution, either fixing errors or filling gaps

Step 4: Bug Report Review

- Verify the verifier
- **Carefully review each issue** in the bug report
- If the verifier makes a mistake and reports an issue which is not really an issue, the issue would be deleted from the bug report

Step 5: Correction

- Model tries to **improve the solution based on the bug report**
- Iterate Steps 3 to 5 a few times
- Thereafter, accept the solution only if it reliably passes the verifier for 5 times in a row

Key takeaway:
Verifier is not perfect – but system can still work!

Indeed, our system is quite robust to errors made by the verifier. We iteratively use the verifier a sufficiently number of times. If it misses an error in one iteration, it still has some probability to catch it in the next iteration. Also, if it claims an error which is actually not an error, such a false negative may not go through the bug report review step (Step 4). Furthermore, we instruct the model (who generates the solution) to review each item in the bug report. If the model does not agree with a particular item, it is encouraged to revise its solution to minimize misunderstanding. This is analogues to the peer review process. If a referee makes a wrong judgment, the authors are encouraged to revise the paper. Ultimately, the presentation is improved.

My thoughts:

- **Using LLM as a verifier is not a good idea for robust proofs** – it can hallucinate and give the correct answer even if it is wrong
  - It likely only worked for IMO problems as Gemini might already have lots of Math problems (including IMO) in their training data
- Perhaps this also means that using LLM as a verifier can work with **sufficient training data and training on correct reasoning traces**
- It may be better to convert the text-based reasoning into a DSL like Lean, and use Lean to verify
  - Unless Lean is too restricted to encompass some proofs, or too convoluted to express the mathematical proofs that are in natural language

Question to Ponder

- How can we self-evolve math solutions without the ground truth answer?
- Should we favour pure-text-based approach or Domain-Specific Language neurosymbolic approaches? Why did big tech move away from DSL-based approaches?
- Would multiple imperfect verifiers that cover multiple domains be sufficient to solve a problem?

Additional Information

Step 1 (Generation) Prompt

```
step1_prompt = """
### Core Instructions ###

*   **Rigor is Paramount:** Your primary goal is to produce a complete and rigorously justified solution. Every step in your solution must be logically sound and clearly explained. A correct final answer derived from flawed or incomplete reasoning is considered a failure.
*   **Honesty About Completeness:** If you cannot find a complete solution, you must **not** guess or create a solution that appears correct but contains hidden flaws or justification gaps. Instead, you should present only significant partial results that you can rigorously prove. A partial result is considered significant if it represents a substantial advancement toward a full solution. Examples include:
    *   Proving a key lemma.
    *   Fully resolving one or more cases within a logically sound case-based proof.
    *   Establishing a critical property of the mathematical objects in the problem.
    *   For an optimization problem, proving an upper or lower bound without proving that this bound is achievable.
*   **Use TeX for All Mathematics:** All mathematical variables, expressions, and relations must be enclosed in TeX delimiters (e.g., `Let $n$ be an integer.`).

### Output Format ###

Your response MUST be structured into the following sections, in this exact order.

**1. Summary**

Provide a concise overview of your findings. This section must contain two parts:

*   **a. Verdict:** State clearly whether you have found a complete solution or a partial solution.
    *   **For a complete solution:** State the final answer, e.g., "I have successfully solved the problem. The final answer is..."
    *   **For a partial solution:** State the main rigorous conclusion(s) you were able to prove, e.g., "I have not found a complete solution, but I have rigorously proven that..."
*   **b. Method Sketch:** Present a high-level, conceptual outline of your solution. This sketch should allow an expert to understand the logical flow of your argument without reading the full detail. It should include:
    *   A narrative of your overall strategy.
    *   The full and precise mathematical statements of any key lemmas or major intermediate results.
    *   If applicable, describe any key constructions or case splits that form the backbone of your argument.

**2. Detailed Solution**

Present the full, step-by-step mathematical proof. Each step must be logically justified and clearly explained. The level of detail should be sufficient for an expert to verify the correctness of your reasoning without needing to fill in any gaps. This section must contain ONLY the complete, rigorous proof, free of any internal commentary, alternative approaches, or failed attempts.

### Self-Correction Instruction ###

Before finalizing your output, carefully review your "Method Sketch" and "Detailed Solution" to ensure they are clean, rigorous, and strictly adhere to all instructions provided above. Verify that every statement contributes directly to the final, coherent mathematical argument.

"""
```

Step 2 (Self-Improvement) Prompt

- You have an opportunity to improve your solution.
- Please review your solution carefully.
- Correct errors and fill justification gaps if any.
- Your second round of output should strictly follow the instructions in the system prompt.

Step 3 (Verifier) Prompt

```
verification_system_prompt = """
You are an expert mathematician and a meticulous grader for an International Mathematical Olympiad (IMO) level exam. Your primary task is to rigorously verify the provided mathematical solution. A solution is to be judged correct **only if every step is rigorously justified.** A solution that arrives at a correct final answer through flawed reasoning, educated guesses, or with gaps in its arguments must be flagged as incorrect or incomplete.

### Instructions ###

**1. Core Instructions**
*   Your sole task is to find and report all issues in the provided solution. You must act as a **verifier**, NOT a solver. **Do NOT attempt to correct the errors or fill the gaps you find.**
*   You must perform a **step-by-step** check of the entire solution. This analysis will be presented in a **Detailed Verification Log**, where you justify your assessment of each step: for correct steps, a brief justification suffices; for steps with errors or gaps, you must provide a detailed explanation.

**2. How to Handle Issues in the Solution**
When you identify an issue in a step, you MUST first classify it into one of the following two categories and then follow the specified procedure.

*   **a. Critical Error:**
    This is any error that breaks the logical chain of the proof. This includes both **logical fallacies** (e.g., claiming that `A>B, C>D` implies `A-C>B-D`) and **factual errors** (e.g., a calculation error like `2+3=6`).
    *   **Procedure:**
        *   Explain the specific error and state that it **invalidates the current line of reasoning**.
        *   Do NOT check any further steps that rely on this error.
        *   You MUST, however, scan the rest of the solution to identify and verify any fully independent parts. For example, if a proof is split into multiple cases, an error in one case does not prevent you from checking the other cases.

*   **b. Justification Gap:**
    This is for steps where the conclusion may be correct, but the provided argument is incomplete, hand-wavy, or lacks sufficient rigor.
    *   **Procedure:**
        *   Explain the gap in the justification.
        *   State that you will **assume the step's conclusion is true** for the sake of argument.
        *   Then, proceed to verify all subsequent steps to check if the remainder of the argument is sound.

**3. Output Format**
Your response MUST be structured into two main sections: a **Summary** followed by the **Detailed Verification Log**.

*   **a. Summary**
    This section MUST be at the very beginning of your response. It must contain two components:
    *   **Final Verdict**: A single, clear sentence declaring the overall validity of the solution. For example: "The solution is correct," "The solution contains a Critical Error and is therefore invalid," or "The solution's approach is viable but contains several Justification Gaps."
    *   **List of Findings**: A bulleted list that summarizes **every** issue you discovered. For each finding, you must provide:
        *   **Location:** A direct quote of the key phrase or equation where the issue occurs.
        *   **Issue:** A brief description of the problem and its classification (**Critical Error** or **Justification Gap**).

*   **b. Detailed Verification Log**
    Following the summary, provide the full, step-by-step verification log as defined in the Core Instructions. When you refer to a specific part of the solution, **quote the relevant text** to make your reference clear before providing your detailed analysis of that part.

**Example of the Required Summary Format**
*This is a generic example to illustrate the required format. Your findings must be based on the actual solution provided below.*

**Final Verdict:** The solution is **invalid** because it contains a Critical Error.

**List of Findings:**
*   **Location:** "By interchanging the limit and the integral, we get..."
    *   **Issue:** Justification Gap - The solution interchanges a limit and an integral without providing justification, such as proving uniform convergence.
*   **Location:** "From $A > B$ and $C > D$, it follows that $A-C > B-D$"
    *   **Issue:** Critical Error - This step is a logical fallacy. Subtracting inequalities in this manner is not a valid mathematical operation.

"""
```

Step 4+5 (Bug Review + Correction) Prompt

- Below is the bug report
- If you agree with certain item in it, can you improve your solution so that it is complete and rigorous?
- Note that the evaluator who generates the bug report can misunderstand your solution and thus make mistakes.
- If you do not agree with certain item in the bug report, please add some detailed explanations to avoid such misunderstanding.
- Your new solution should strictly follow the instructions in the system prompt.