本文由 AINLP 公众号整理翻译，更多 LLM 资源请扫码关注!

DeepSeek-Prover-V2: Advancing Formal Mathematical Reasoning via Reinforcement Learning for Subgoal Decomposition

Z.Z. Ren*, Zhihong Shao*, Junxiao Song*, Huajian Xin†, Haocheng Wang†, Wanjia Zhao†, Liyue Zhang, Zhe Fu
Qihao Zhu, Dejian Yang, Z.F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao
Daya Guo, Chong Ruan

DeepSeek-AI

https://github.com/deepseek-ai/DeepSeek-Prover-V2

Abstract

We introduce DeepSeek-Prover-V2, an open-source large language model designed for formal theorem proving in Lean 4, with initialization data collected through a recursive theorem proving pipeline powered by DeepSeek-V3. The cold-start training procedure begins by prompting DeepSeek-V3 to decompose complex problems into a series of subgoals. The proofs of resolved subgoals are synthesized into a chain-of-thought process, combined with DeepSeek-V3's step-by-step reasoning, to create an initial cold start for reinforcement learning. This process enables us to integrate both informal and formal mathematical reasoning into a unified model. The resulting model, DeepSeek-Prover-V2-671B, achieves state-of-the-art performance in neural theorem proving, reaching 88.9% pass ratio on the MiniF2F-test and solving 49 out of 658 problems from PutnamBench. In addition to standard benchmarks, we introduce ProverBench, a collection of 325 formalized problems, to enrich our evaluation, including 15 selected problems from the recent AIME competitions (years 24-25). Further evaluation on these 15 AIME problems shows that the model successfully solves 6 of them. In comparison, DeepSeek-V3 solves 8 of these problems using majority voting, highlighting that the gap between formal and informal mathematical reasoning in large language models is substantially narrowing.

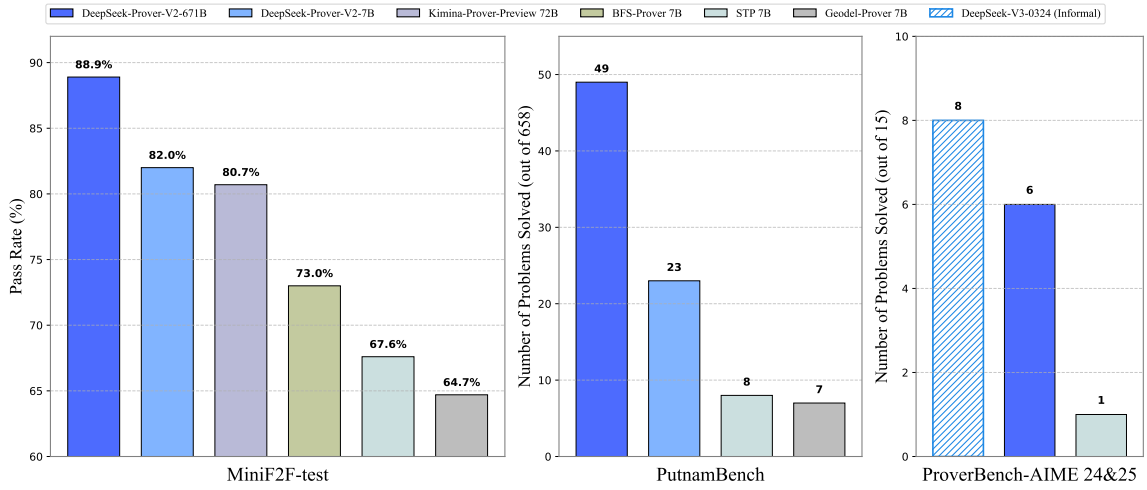


Figure 1 | Benchmark performance of DeepSeek-Prover-V2. On the AIME benchmark, DeepSeek-V3 is evaluated using the standard find-answer task for natural-language reasoning, while prover models generate Lean code to construct formal proofs for a given correct answer.

*Core contributors †Work done during internship at DeepSeek-AI.

DeepSeek-Prover-V2：通过强化学习推进子目标分解的形式数学推理

Z.Z. Ren*，邵志宏*，宋俊孝*，辛华健†，王浩成†，赵万佳†，张丽月，傅哲，朱启浩，杨德建，吴志飞，苟志斌，马世荣，唐宏轩，刘玉轩，高文俊，郭大雅，阮冲

DeepSeek-AI

https://github.com/deepseek-ai/DeepSeek-Prover-V2

摘要

我们介绍了DeepSeek-Prover-V2，一款开源的大型语言模型，旨在用于Lean 4中的形式定理证明，其初始化数据通过由DeepSeek-V3驱动的递归定理证明流程收集。冷启动训练过程首先让DeepSeek-V3将复杂问题分解为一系列子目标。已解决子目标的证明被合成为一个思维链过程，结合DeepSeek-V3的逐步推理，创建了强化学习的初始冷启动。这一过程使我们能够将非正式和正式的数学推理整合到一个统一的模型中。最终的模型DeepSeek-Prover-V2-671B在神经定理证明方面达到了最先进的性能，在MiniF2F测试中达到88.9%的通过率，并解决了PutnamBench中的658个问题中的49个。除了标准基准测试外，我们还引入了ProverBench，这是一个包含325个形式化问题的集合，以丰富我们的评估，包括从最近的AIME竞赛（24-25年）中精选的15个问题。对这15个AIME问题的进一步评估显示，该模型成功解决了其中的6个。相比之下，DeepSeek-V3通过多数投票方法解决了其中的8个问题，突显出大型语言模型中的正式与非正式数学推理之间的差距正显著缩小。

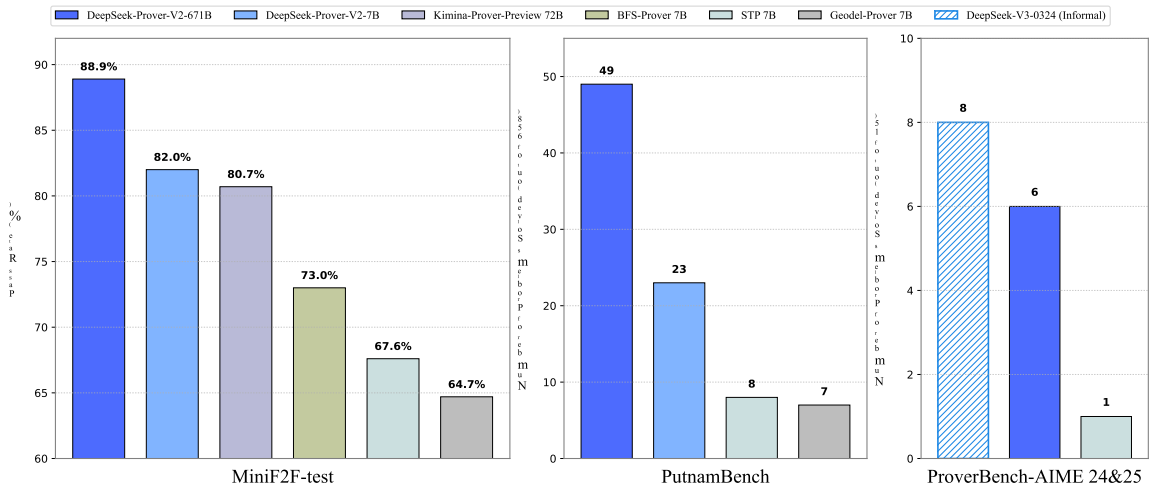


图 1 | DeepSeek-Prover-V2 的基准性能。在 AIME 基准测试中，DeepSeek-V3 使用标准的查找答案任务进行自然语言推理，而证明器模型生成 Lean 代码以构建给定正确答案的形式证明。

*Core contributors †Work done during internship at DeepSeek-AI.

1. Introduction

The emergence of reasoning capabilities in large language models (LLMs) has revolutionized numerous areas of artificial intelligence, particularly in the domain of mathematical problem solving (DeepSeek-AI, 2025). These advancements are largely enabled by the paradigm of inference-time scaling, most notably through natural language chain-of-thought reasoning (Jaech et al., 2024). Rather than relying solely on a single forward pass to arrive at an answer, LLMs can reflect on intermediate reasoning steps, improving both accuracy and interpretability. Despite the success of natural language reasoning in solving competition-level mathematical problems, its application to formal theorem proving remains fundamentally challenging. LLMs perform natural language reasoning in an inherently informal manner, relying on heuristics, approximations, and data-driven guessing patterns that often lack the rigorous structure required by formal verification systems. In contrast, proof assistants such as Lean (Moura and Ullrich, 2021), Isabelle (Paulson, 1994), and Coq (Barras et al., 1999) operate on strict logical foundations, where every proof step must be explicitly constructed and formally verified. These systems permit no ambiguity, implicit assumptions, or omission of details. Bridging the gap between informal, high-level reasoning and the syntactic rigor of formal verification systems remains a longstanding research challenge in neural theorem proving (Yang et al., 2024).

To harness the strengths of informal mathematical reasoning in support of formal theorem proving, a classical approach is to hierarchically decompose formal proofs based on the guidance of natural-language proof sketches. Jiang et al. (2023) proposed a framework, called *Draft, Sketch, and Prove* (DSP), that leverages a large language model to generate proof sketches in natural language, which are subsequently translated into formal proof steps. This informal-to-formal theorem proving paradigm closely mirrors the concept of subgoals in hierarchical reinforcement learning (Barto and Mahadevan, 2003; Nachum et al., 2018; Eppe et al., 2022), where complex tasks are broken down into a hierarchy of simpler subtasks that can be solved independently to progressively achieve the overarching objective. In formal theorem proving, a subgoal is typically an intermediate proposition or lemma that contributes to the proof of a larger theorem (Zhao et al., 2023, 2024). This hierarchical decomposition aligns with human problem-solving strategies and supports modularity, reusability, and more efficient proof search (Wang et al., 2024b; Zheng et al., 2024). Recent studies have extended this paradigm by employing multi-tiered hierarchies for structured proof generation (Wang et al., 2024a), and by leveraging reinforcement learning techniques to optimize the decomposition of complex theorems into manageable subgoals (Dong et al., 2024).

In this paper, we develop a reasoning model for subgoal decomposition, leveraging a suite of synthetic cold-start data and large-scale reinforcement learning to enhance its performance. To construct the cold-start dataset, we develop a simple yet effective pipeline for recursive theorem proving, utilizing DeepSeek-V3 (DeepSeek-AI, 2024) as a unified tool for both subgoal decomposition and formalization. We prompt DeepSeek-V3 to decompose theorems into high-level proof sketches while simultaneously formalizing these proof steps in Lean 4, resulting in a sequence of subgoals. Since the subgoal decomposition is powered by a large general-purpose model, we use a smaller 7B model to handle the proof search for each subgoal, thereby reducing the associated computational burden. Additionally, we introduce a curriculum learning framework that leverages the decomposed subgoals to generate conjectural theorems, progressively increasing the difficulty of training tasks to better guide the model's learning process. Once the decomposed steps of a challenging problem are resolved, we pair the complete step-by-step formal proof with the corresponding chain-of-thought from DeepSeek-V3 to create cold-start reasoning data. Based on the cold start, a subsequent reinforcement learning stage is applied to further strengthen

1. 引言

大型语言模型（LLMs）中推理能力的出现彻底改变了人工智能的众多领域，特别是在数学问题解决领域（DeepSeek-AI，2025）。这些进展主要得益于推理时刻扩展的范式，尤其是通过自然语言链式思考推理（Jaech 等，2024）。与仅依赖单次前向传播得出答案不同，LLMs可以反思中间推理步骤，从而提高准确性和可解释性。尽管自然语言推理在解决竞赛级数学问题方面取得了成功，但其在正式定理证明中的应用仍然面临根本性挑战。LLMs以一种本质上非正式的方式进行自然语言推理，依赖启发式、近似和数据驱动的猜测模式，这些往往缺乏正式验证系统所需的严格结构。相比之下，像 Lean（Moura 和 Ullrich，2021）、Isabelle（Paulson，1994）和 Coq（Barras 等，1999）这样的证明助手建立在严格的逻辑基础上，每一步证明都必须明确构建并经过形式验证。这些系统不允许模糊、不明确的假设或细节的遗漏。在非正式的高层次推理与形式验证系统的语法严谨性之间架起桥梁，仍然是神经定理证明领域长期以来的研究难题（Yang 等，2024）。

为了利用非正式数学推理的优势支持形式定理证明，一种经典的方法是基于自然语言证明草图的引导，将形式证明进行层次化分解。Jiang 等人（2023）提出了一个框架，称为 *Draft, Sketch, and Prove* (DSP)，该框架利用大型语言模型生成自然语言的证明草图，然后将其转换为形式证明步骤。这种由非正式到正式的定理证明范式与层次强化学习中的子目标概念（Barto 和 Mahadevan, 2003；Nachum 等人, 2018；Eppe 等人, 2022）密切相关，在层次强化学习中，复杂任务被拆分成一系列更简单的子任务，可以独立解决，从而逐步实现总体目标。在形式定理证明中，子目标通常是对更大定理的中间命题或引理（Zhao 等人, 2023, 2024）。这种层次化分解符合人类的问题解决策略，支持模块化、重用性以及更高效的证明搜索（Wang 等人, 2024b；Zheng 等人, 2024）。近期研究通过采用多层次的层级结构进行结构化证明生成（Wang 等人, 2024a），以及利用强化学习技术优化将复杂定理分解为可管理子目标的过程（Dong 等人, 2024），扩展了这一范式。

在本文中，我们开发了一种用于子目标分解的推理模型，利用一套合成的冷启动数据和大规模强化学习来提升其性能。为了构建冷启动数据集，我们开发了一个简单而有效的递归定理证明流程，使用DeepSeek-V3（DeepSeek-AI，2024）作为子目标分解和形式化的统一工具。我们提示DeepSeek-V3将定理分解为高层次的证明草图，同时在Lean 4中形式化这些证明步骤，生成一系列子目标。由于子目标分解由一个大型通用模型驱动，我们使用一个较小的7B模型来处理每个子目标的证明搜索，从而减轻相关的计算负担。此外，我们引入了一种课程学习框架，利用分解的子目标生成猜想定理，逐步增加训练任务的难度，以更好地引导模型的学习过程。一旦解决了具有挑战性问题的分解步骤，我们将完整的逐步形式化证明与DeepSeek-V3的对应思路链配对，生成冷启动推理数据。在冷启动的基础上，随后应用强化学习阶段，进一步增强模型的能力。

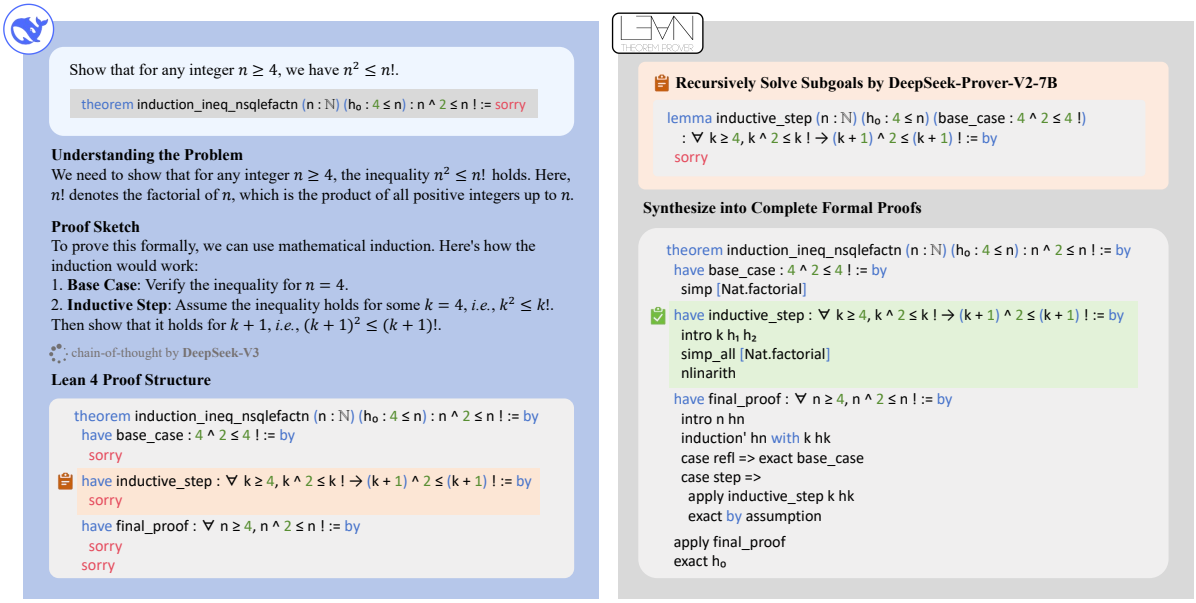


Figure 2 | Overview of the cold-start data collection process employed by DeepSeek-Prover-V2. We first prompt DeepSeek-V3 to generate a natural-language proof sketch while simultaneously formalizing it into a Lean statement with `sorry` placeholders for omitted proof details. A 7B prover model then recursively solves the decomposed subgoals. By combining these subgoal proofs, we construct a complete formal proof for the original complex problem. This composed proof is appended to DeepSeek-V3's original chain-of-thought, creating high-quality cold-start training data for formal mathematical reasoning.

the connection between informal mathematical reasoning and formal proof construction. Our experiments show that reinforcement learning starting from the cold start of informal reasoning in task decomposition significantly enhances the model's capabilities in formal theorem proving. The resulting DeepSeek-Prover-V2-671B model establishes a new state-of-the-art in neural theorem proving across multiple benchmarks. On MiniF2F-test, it achieves 82.4% accuracy with Pass@32, improving to 88.9% with Pass@8192. The model shows strong generalization capabilities to college-level theorem proving, solving 37.1% of ProofNet-test problems with Pass@1024 and tackling 49 out of 658 challenging PutnamBench problems. Additionally, we contribute ProverBench, a benchmark dataset containing 325 formalized problems to advance neural theorem proving research, including 15 from the prestigious AIME competitions (years 24-25). DeepSeek-Prover-V2-671B successfully solves 6 of these 15 challenging AIME problems, further demonstrating its sophisticated mathematical reasoning capabilities.

2. Method

2.1. Recursive Proof Search via Subgoal Decomposition

Decomposing the proof of a complex theorem into a sequence of smaller lemmas that serve as stepping stones is an effective strategy commonly employed by human mathematicians. Several previous studies have explored this hierarchical strategy in the context of neural theorem proving, aiming to enhance proof search efficiency by leveraging the informal reasoning capabilities of modern LLMs (Jiang et al., 2023; Zhao et al., 2023; Wang et al., 2024a; Dong et al., 2024). In this paper, we develop a simple yet effective pipeline that utilizes DeepSeek-V3 (DeepSeek-AI, 2024) as a unified tool for subgoal decomposition in formal theorem proving.

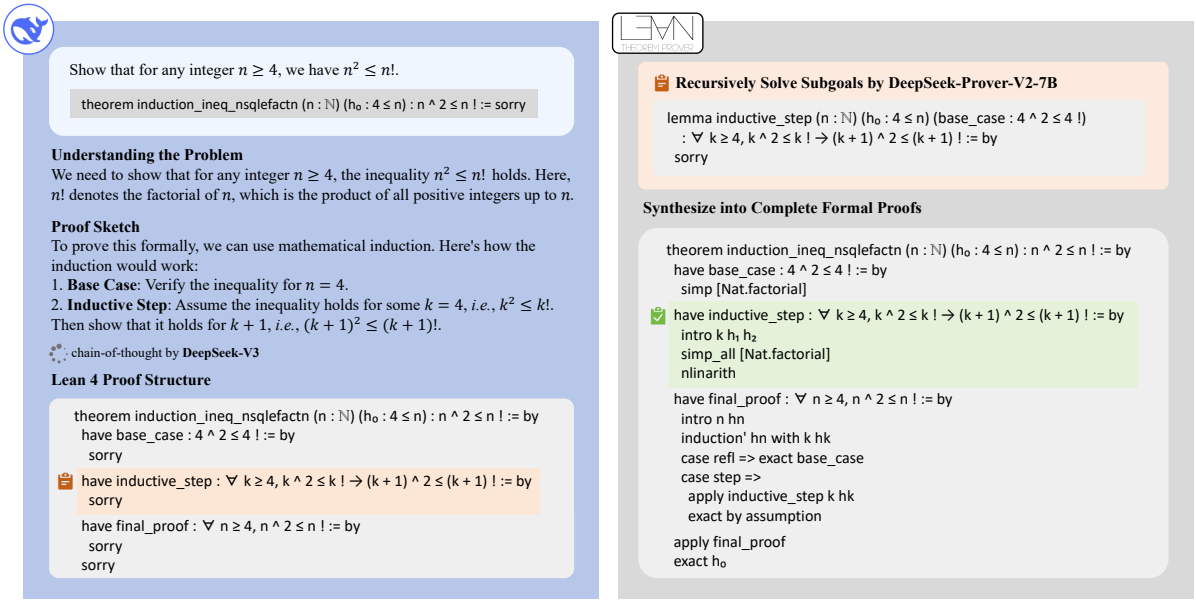


图2 | DeepSeek-Prover-V2采用的冷启动数据收集流程概述。我们首先提示DeepSeek-V3生成一个自然语言的证明草图，同时将其形式化为带有`sorry`占位符的Lean陈述，用于省略的证明细节。然后，一个7B的证明模型递归地解决分解后的子目标。通过结合这些子目标的证明，我们构建了原始复杂问题的完整形式化证明。这个组合的证明被附加到DeepSeek-V3的原始思路链中，生成高质量的正式数学推理冷启动训练数据。

正式数学推理与形式证明构建之间的联系。我们的实验表明，从任务分解中非正式推理的冷启动开始的强化学习显著提升了模型在形式定理证明中的能力。由此产生的DeepSeek-Prover-V2-671B模型在多个基准测试中建立了神经定理证明的新纪录。在MiniF2F测试中，它以Pass@32达到82.4%的准确率，提升至Pass@8192时为88.9%。该模型展现出强大的泛化能力，能够进行大学水平的定理证明，在ProofNet-test中以Pass@1024解决了37.1%的问题，并攻克了658个挑战性PutnamBench问题中的49个。此外，我们还贡献了ProverBench，这是一个包含325个形式化问题的基准数据集，旨在推动神经定理证明的研究，其中包括来自著名AIME竞赛（24-25年）的15个题目。DeepSeek-Prover-V2-671B成功解决了这15个具有挑战性的AIME问题中的6个，进一步展示了其复杂的数学推理能力。

2. 方法

2.1. 通过子目标分解的递归证明搜索

将复杂定理的证明分解为一系列较小的引理，作为铺垫步骤，是人类数学家常用的一种有效策略。此前的多项研究在神经定理证明的背景下探索了这种层级策略，旨在通过利用现代大型语言模型（LLMs）的非正式推理能力来提高证明搜索的效率（Jiang 等，2023；Zhao 等，2023；Wang 等，2024a；Dong 等，2024）。在本文中，我们开发了一个简单而有效的流程，利用 DeepSeek-V3（DeepSeek-AI，2024）作为统一工具，用于形式化定理证明中的子目标分解。

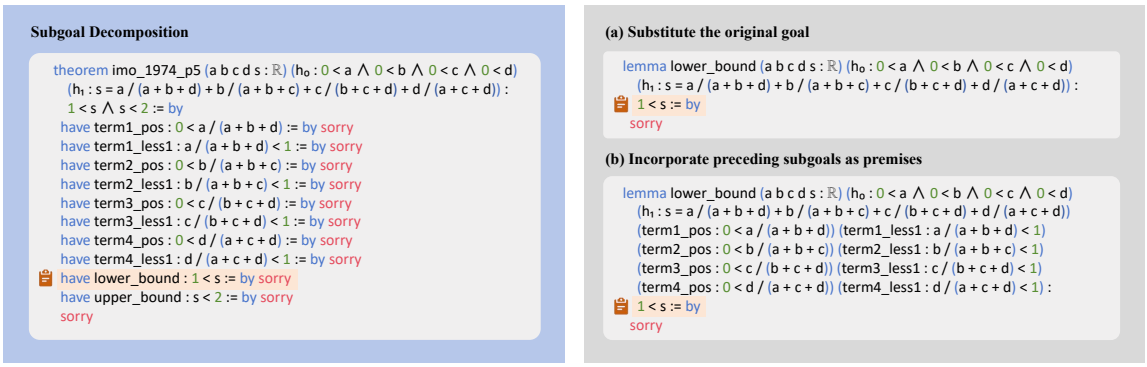


Figure 3 | An illustrative example of how we translate decomposed subgoals into a series of lemma statements. We first (a) replace the original goal state and then (b) incorporate preceding subgoals as premises. Statement type (b) is used for recursive solving of complex problems, while both types (a) and (b) are incorporated into the curriculum learning process.

Sketching Formal Proofs from Natural Language Reasoning. Recent advances in large language models have led to significant breakthroughs in informal reasoning capabilities. To bridge the gap between formal and informal reasoning, we leverage cutting-edge general-purpose LLMs, recognized for their strong mathematical reasoning and instruction-following abilities, to construct the foundational framework of our theorem-proving system. Our findings indicate that off-the-shelf models, such as DeepSeek-V3 (DeepSeek-AI, 2024), are capable of decomposing proof steps and expressing them in formal languages. To prove a given formal theorem statement, we prompt DeepSeek-V3 to first analyze the mathematical problem in natural language, then decompose the proof into smaller steps, translating each step into a corresponding Lean formal statement. Since general-purpose models are known to struggle with producing complete Lean proofs, we instruct DeepSeek-V3 to generate only a high-level proof sketch with the details omitted. The resulting chain of thought culminates in a Lean theorem composed of a sequence of `have` statements, each concluded with a `sorry` placeholder indicating a subgoal to be solved. This approach mirrors the human style of proof construction, in which a complex theorem is incrementally reduced to a sequence of more manageable lemmas.

Recursive Resolution of Subgoals. Leveraging the subgoals generated by DeepSeek-V3, we adopt a recursive solving strategy to systematically resolve each intermediate proof step. We extract subgoal expressions from `have` statements to substitute them for the original goals in the given problems (see Figure 3(a)), and then incorporate the preceding subgoals as premises (see Figure 3(b)). This construction enables subsequent subgoals to be resolved using the intermediate results of earlier steps, thereby promoting a more localized dependency structure and facilitating the development of simpler lemmas. To reduce the computational overhead of extensive proof search, we employ a smaller 7B prover model specifically optimized for processing the decomposed lemmas. Upon the successful resolution of all decomposed steps, a complete proof of the original theorem can be automatically derived.

Curriculum Learning for Subgoal-based Theorem Proving. The training of prover models requires large formal-language problem sets, typically derived by formalizing existing natural-language mathematical corpora (Xin et al., 2024a; Ying et al., 2024; Lin et al., 2025). Although formalization of human-authored texts provides high-quality and diverse formal content, the resulting training signals for prover models are often sparse, as a large proportion of computa-

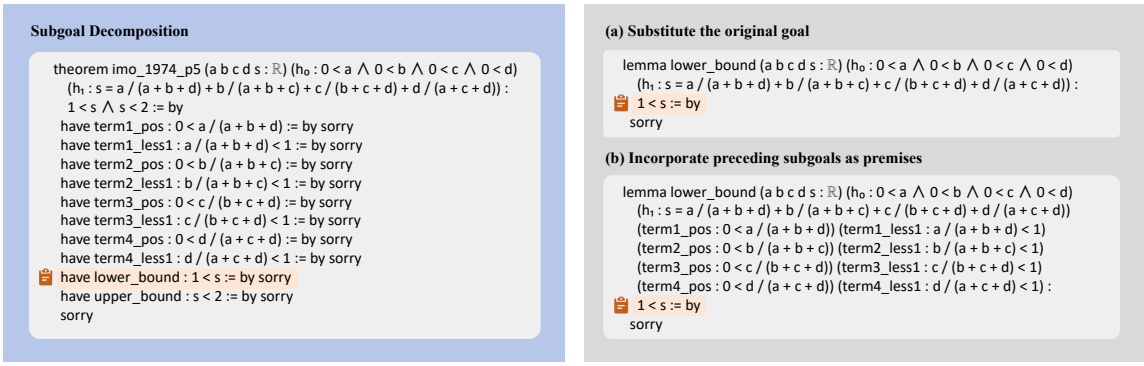


图3 | 一个关于我们如何将分解的子目标转化为一系列引理陈述的示例。我们首先（a）替换原始目标状态，然后（b）将前面的子目标作为前提纳入。陈述类型（b）用于递归解决复杂问题，而类型（a）和（b）都被纳入课程学习过程。

从自然语言推理中勾勒正式证明。近年来，大型语言模型的突破性进展极大地提升了非正式推理能力。为了弥合正式推理与非正式推理之间的差距，我们利用最先进的通用大型语言模型（LLMs），它们以强大的数学推理和指令执行能力而闻名，构建我们的定理证明系统的基础框架。我们的研究发现，现成的模型，如 DeepSeek-V3（DeepSeek-AI，2024），能够将证明步骤分解并用形式语言表达。为了证明给定的正式定理陈述，我们首先让 DeepSeek-V3 分析自然语言中的数学问题，然后将证明分解为更小的步骤，并将每个步骤翻译成对应的 Lean 正式陈述。由于众所周知，通用模型在生成完整的 Lean 证明方面存在困难，我们指示 DeepSeek-V3 只生成高层次的证明草图，省略细节。最终的思路链条以 Lean 定理的形式呈现，由一系列 `have` 陈述组成，每个陈述以一个 `sorry` 占位符结束，表示待解决的子目标。这种方法模仿人类的证明构建风格，将复杂的定理逐步简化为一系列更易处理的引理。

递归解决子目标。利用 DeepSeek-V3 生成的子目标，我们采用递归求解策略，系统地解决每个中间证明步骤。我们从 `have` 语句中提取子目标表达式，将其替换为给定问题中的原始目标（见图 3(a)），然后将前面的子目标作为前提加入（见图 3(b)）。这种构造方式使得后续的子目标可以利用前面步骤的中间结果进行求解，从而促进更局部的依赖结构，并有助于简化引理的推导。为了减少大量证明搜索的计算开销，我们使用了专门为处理分解引理优化的较小的 7B 证明器模型。在成功解决所有分解步骤后，可以自动推导出原始定理的完整证明。

基于子目标的定理证明的课程学习。证明器模型的训练需要大量的形式语言问题集，通常通过形式化现有的自然语言数学语料库（Xin 等，2024a；Ying 等，2024；Lin 等，2025）获得。虽然对人工撰写文本的形式化提供了高质量且多样的形式内容，但由此产生的证明器模型的训练信号通常较稀疏，因为大量的计算比例- {v*}

tional attempts do not yield successful proofs and therefore offer no positive reward signals. To generate denser training signals, Dong and Ma (2025) proposed a self-play approach that enriches training problem sets by generating tractable conjectures closely related to the original theorem statements, thereby enabling more efficient use of training compute. In this paper, we implement a straightforward approach that leverages subgoals to expand the scope of formal statements used for model training. We generate two types of subgoal theorems, one incorporating preceding subgoals as premises and one without, corresponding to Figure 3(b) and Figure 3(a), respectively. Both types are integrated into the expert iteration stage (Polu and Sutskever, 2020), establishing a curriculum that progressively guides the prover model toward systematically addressing a curated subset of challenging problems. This procedure builds on the same underlying principle as AlphaProof's test-time reinforcement learning (DeepMind, 2024), wherein variations of a target problem are generated to enhance the model's capability in solving challenging IMO-level problems.

2.2. Unifying Informal Reasoning and Proof Formalization

The algorithmic framework discussed above operates in two stages, leveraging two complementary models: DeepSeek-V3 for lemma decomposition and a 7B prover model to complete the corresponding formal proof details. This pipeline provides a natural and scalable mechanism for synthesizing formal reasoning data by combining high-level reasoning from language models with precise formal verification. In this manner, we unify the capabilities of informal mathematical reasoning and proof formalization within a single model.

Cold Start by Synthetic Data. We curate a subset of challenging problems that remain unsolved by the 7B prover model in an end-to-end manner, but for which all decomposed subgoals have been successfully resolved. By composing the proofs of all subgoals, we construct a complete formal proof for the original problem. This proof is then appended to DeepSeek-V3's chain-of-thought, which outlines the corresponding lemma decomposition, thereby producing a cohesive synthesis of informal reasoning and subsequent formalization. It enables the collection of hundreds of high-quality synthetic cold-start data, which serve as the foundation for training DeepSeek-Prover-V2. This cold-start dataset generation strategy differs from that of Kimina-Prover (Wang et al., 2025), a concurrent work on formal reasoning models. Specifically, we synthesize data by formalizing natural-language proofs directly into structured formal proof sketches. In contrast, Kimina-Prover adopts a reverse workflow: it begins by collecting complete formal proofs alongside their informal counterparts, then uses general-purpose reasoning models to retrosynthesize intermediate natural-language reasoning steps into coherent thinking blocks.

Reasoning-oriented Reinforcement Learning. After fine-tuning the prover model on the synthetic cold-start data, we perform a reinforcement learning stage to further enhance its ability to bridge informal reasoning with formal proof construction. Following the standard training objective for reasoning models (DeepSeek-AI, 2025), we use binary correct-or-incorrect feedback as the primary form of reward supervision. During the training process, we observe that the structure of the generated proofs frequently diverges from the lemma decomposition provided by the chain-of-thought guidance. To address this issue, we incorporate a consistency reward in the early steps of training, which penalizes the structural misalignment, explicitly enforcing the inclusion of all decomposed `have`-structured lemmas in the final proof. In practice, enforcing this alignment enhances proof accuracy, especially on complex theorems that demand multi-step reasoning.

尝试性的方法未能产生成功的证明，因此没有提供积极的奖励信号。为了生成更密集的训练信号，Dong 和 Ma (2025) 提出了一种自我对弈的方法，通过生成与原始定理陈述密切相关的可行猜想，丰富训练问题集，从而实现更高效的训练计算。在本文中，我们实现了一种直接的方法，利用子目标来扩展用于模型训练的形式化陈述的范围。我们生成了两种类型的子目标定理，一种将先前的子目标作为前提，另一种则没有，分别对应图3(b)和图3(a)。这两种类型都被整合到专家迭代阶段（Polu 和 Sutskever, 2020），建立了一个逐步引导证明模型系统性解决精选难题子集的课程。这一过程基于与 AlphaProof 的测试时强化学习（DeepMind, 2024）相同的基本原理，即生成目标问题的变体，以增强模型解决具有挑战性的 IMO 级问题的能力。

2.2. 统一非正式推理与形式化证明

上述讨论的算法框架分为两个阶段，利用两个互补的模型：DeepSeek-V3 用于词元分解，以及一个 7B 证明模型用于完成相应的正式证明细节。该流程为合成正式推理数据提供了一种自然且可扩展的机制，通过结合语言模型的高层次推理与精确的正式验证。以这种方式，我们在单一模型中统一了非正式数学推理和证明形式化的能力。

通过合成数据实现冷启动。我们策划了一组具有挑战性的问题子集，这些问题在端到端的方式下仍未被7B证明模型解决，但所有分解的子目标都已成功解决。通过组合所有子目标的证明，我们构建了原始问题的完整形式证明。然后，将此证明附加到DeepSeek-V3的思维链中，概述了相应的引理分解，从而实现了非正式推理与后续形式化的连贯整合。这一方法使得收集数百个高质量的合成冷启动数据成为可能，这些数据作为训练DeepSeek-Prover-V2的基础。这种冷启动数据集生成策略不同于Kimina-Prover（Wang等，2025年）的一项同时进行的形式推理模型工作。具体而言，我们通过将自然语言证明直接形式化为结构化的形式证明草图来合成数据。相比之下，Kimina-Prover采用逆向工作流程：它首先收集完整的形式证明及其非正式对应，然后利用通用推理模型将中间的自然语言推理步骤逆向合成为连贯的思考块。

以推理为导向的强化学习。在对证明器模型进行合成冷启动数据的微调后，我们进行强化学习阶段，以进一步增强其将非正式推理与正式证明构建相结合的能力。遵循推理模型的标准训练目标（DeepSeek-AI，2025），我们使用二元的正确或错误反馈作为主要的奖励监督形式。在训练过程中，我们观察到生成的证明结构经常偏离链式思考指导提供的引理分解。为了解决这一问题，我们在训练的早期步骤中引入了一致性奖励，该奖励惩罚结构不一致，明确强制在最终证明中包含所有分解的`have`结构引理。实际上，强制执行这种对齐方式可以提高证明的准确性，尤其是在需要多步推理的复杂定理上。

2.3. Training Details of DeepSeek-Prover-V2

Two-Stage Training. DeepSeek-Prover-V2 is developed through a two-stage training pipeline that establishes two complementary proof generation modes:

1. **High-efficiency non-Chain-of-Thought (non-CoT) mode:** This mode is optimized for the rapid generation of formal Lean proof codes, focusing on producing concise proofs without explicit intermediate reasoning steps.
2. **High-precision Chain-of-Thought (CoT) mode:** This mode systematically articulates intermediate reasoning steps, emphasizing transparency and logical progression, before constructing the final formal proofs.

Consistent with DeepSeek-Prover-V1.5 (Xin et al., 2024b), these two generation modes are governed by two distinct guiding prompts (see Appendix A for examples). In the first stage, we employ expert iteration within a curriculum learning framework to train a non-CoT prover model, meanwhile, synthesizing proofs for hard problems through subgoal-based recursive proving. The non-CoT generation mode is chosen to accelerate iterative training and data collection processes, as it offers significantly faster inference and validation cycles. Building on this foundation, the second stage leverages cold-start chain-of-thought (CoT) data synthesized by integrating DeepSeek-V3's sophisticated mathematical reasoning patterns with our synthetic formal proofs. The CoT mode is enhanced through a further reinforcement learning stage, following the standard training pipeline commonly used for reasoning models.

Expert Iteration. The training procedure for the non-CoT mode of DeepSeek-Prover-V2 follows the paradigm of *expert iteration* (Polu and Sutskever, 2020), a widely adopted framework for developing formal theorem provers. In each training iteration, the current best prover policy is used to generate proof attempts for those challenging problems that remain unsolved in prior iterations. Those successful attempts, verified by Lean proof assistant, are incorporated into the SFT dataset to train an improved model. This iterative loop ensures that the model not only learns from the initial demonstration datasets but also distills its own successful reasoning traces, progressively refining its ability to solve harder problems. The overall training procedure remains largely aligned with that of DeepSeek-Prover-V1 (Xin et al., 2024a) and DeepSeek-Prover-V1.5 (Xin et al., 2024b), with only two modifications to the distribution of training problems. First, we incorporate additional problems derived from autoformalization and various open-source datasets (Ying et al., 2024; Dong and Ma, 2025; Lin et al., 2025), broadening the coverage of the training problem domains. Second, we augment the dataset with problems generated through subgoal decomposition, aiming at solving more challenging instances from the `valid` split of the MiniF2F benchmark (Zheng et al., 2022).

Supervised Fine-tuning. We perform supervised fine-tuning on DeepSeek-V3-Base-671B (DeepSeek-AI, 2024) using a constant learning rate of 5e-6 within a context window of 16,384 tokens. Our training corpus consists of two complementary sources: (1) non-CoT data collected through expert iteration, which produces Lean codes without intermediate reasoning steps; and (2) the cold-start CoT data described in Section 2.2, which distills DeepSeek-V3's advanced mathematical reasoning processes into structured proving pathways. The non-CoT components emphasize formal verification skills in the Lean theorem prover ecosystem, while the CoT examples explicitly model the cognitive process of transforming mathematical intuition into formal proof structures.

2.3. DeepSeek-Prover-V2的训练细节

两阶段训练。DeepSeek-Prover-V2 通过一个两阶段的训练流程开发而成，建立了两种互补的证明生成模式：

1. 高效非链式思维（非-CoT）模式：该模式针对快速生成正式的 Lean 证明代码进行了优化，侧重于生成简洁的证明，而不包含明确的中间推理步骤。 2. 高精度链式思维（CoT）模式：该模式系统地表达中间推理步骤，强调透明性和逻辑递进，然后构建最终的正式证明。

与 DeepSeek-Prover-V1.5 (Xin 等, 2024b) 一致，这两种生成模式由两个不同的引导提示控制（见附录 A 示例）。在第一阶段，我们在课程学习框架内采用专家迭代训练非 CoT 证明模型，同时通过子目标递归证明合成难题的证明。选择非 CoT 生成模式是为了加快迭代训练和数据收集过程，因为它提供了显著更快的推理和验证周期。在此基础上，第二阶段利用通过将 DeepSeek-V3 的复杂数学推理模式与我们的合成形式证明相结合，合成的冷启动链式思维（CoT）数据。通过后续的强化学习阶段，进一步增强 CoT 模式，遵循通常用于推理模型的标准训练流程。

专家迭代。DeepSeek-Prover-V2的非CoT模式训练过程遵循*expert iteration* (Polu和Sutskever，2020)的范式，这是开发形式定理证明器的广泛采用的框架。在每次训练迭代中，当前最优的证明策略用于生成那些在前几次迭代中未能解决的具有挑战性的问题的证明尝试。由Lean证明助手验证成功的尝试被纳入SFT数据集，以训练改进的模型。这一迭代循环确保模型不仅从初始演示数据集中学习，还能提取其自身成功的推理轨迹，逐步提升解决更难问题的能力。整体训练流程基本与DeepSeek-Prover-V1（Xin等，2024a）和DeepSeek-Prover-V1.5（Xin等，2024b）保持一致，只有两个方面对训练问题的分布进行了调整。首先，我们引入了由自动形式化和各种开源数据集（Ying等，2024；Dong和Ma，2025；Lin等，2025）派生的额外问题，拓宽了训练问题领域的覆盖范围。其次，我们通过子目标分解生成的问题增强了数据集，旨在解决来自MiniF2F基准测试（Zheng等，2022）`valid`划分中更具挑战性的实例。

有监督微调。我们在 DeepSeek-V3-Base-671B（DeepSeek-AI，2024）上进行有监督微调，使用恒定学习率为 5e-6，上下文窗口为 16,384 个标记。我们的训练语料库由两个互补的来源组成：(1) 通过专家迭代收集的非 CoT 数据，生成没有中间推理步骤的 Lean 代码；(2) 第 2.2 节中描述的冷启动 CoT 数据，它将 DeepSeek-V3 的先进数学推理过程提炼成结构化的证明路径。非 CoT 组件强调在 Lean 定理证明器生态系统中的形式验证技能，而 CoT 示例则明确模拟将数学直觉转化为正式证明结构的认知过程。

Reinforcement Learning. We employ Group Relative Policy Optimization (GRPO; Shao et al., 2024) as our reinforcement learning algorithm, which has demonstrated superior effectiveness and efficiency in reasoning tasks (DeepSeek-AI, 2025). Unlike PPO (Schulman et al., 2017), GRPO eliminates the need for a separate critic model by sampling a group of candidate proofs for each theorem prompt and optimizing the policy based on their relative rewards. Training utilizes binary rewards, where each generated Lean proof receives a reward of 1 if verified as correct and 0 otherwise. To ensure effective learning, we curate training prompts to include only problems that are sufficiently challenging yet solvable by the supervised fine-tuned model. During each iteration, we sample 256 distinct problems, generating 32 candidate proofs per theorem with a maximum sequence length of 32,768 tokens.

Distillation. We extend the maximum context length of DeepSeek-Prover-V1.5-Base-7B (Xin et al., 2024b) from 4,096 to 32,768 tokens and fine-tune this extended-context model using rollout data collected during the reinforcement learning phase of DeepSeek-Prover-V2-671B. Alongside the CoT reasoning mode, we incorporate non-CoT proof data collected during expert iteration to enable a cost-efficient proving option that produces concise formal outputs with a small-size model. In addition, we perform the same reinforcement learning stage used in the training of the 671B model to boost the performance of DeepSeek-Prover-V2-7B.

3. Experimental Results

In this section, we present a systematic evaluation of DeepSeek-Prover-V2 across diverse benchmark datasets of formal theorem proving, covering both high school competition problems and undergraduate-level mathematics. All experimental results of DeepSeek-Prover-V2 are conducted with Lean 4.9.0, using the same testing environment as DeepSeek-Prover-V1.5 (Xin et al., 2024b). Without further specification, baseline evaluation results are sourced from their respective original papers.

3.1. Results on MiniF2F Benchmark

MiniF2F (Zheng et al., 2022) consists of 488 formalized problem statements sourced from a diverse range of mathematical materials, including the AIME, AMC, and IMO competitions, along with selected problems from the MATH dataset (Hendrycks et al., 2021). The benchmark includes Olympiad-level problems covering core areas of elementary mathematics, including algebra, number theory, and induction. These problems are divided into two equally sized subsets, denoted by miniF2F-valid and miniF2F-test, each containing 244 problems with an identical distribution across subject areas. We reserve the miniF2F-test set exclusively for evaluating model performance, while the miniF2F-valid problems are incorporated into curriculum learning with subgoal decomposition. We adopt the revised version of miniF2F released by Wang et al. (2025), and further introduce two additional revisions to miniF2F-valid and one revision to miniF2F-test (see Appendix C).

Comparison with SoTA Models. Table 1 summarizes a comparison of state-of-the-art formal theorem-proving modeling evaluated on the miniF2F-test dataset. The experimental results demonstrate that DeepSeek-Prover-V2-671B establishes a new state-of-the-art performance on the miniF2F-test benchmark, achieving an unprecedented 82.4% accuracy with only 32 samples when leveraging the CoT generation strategy. Notably, the more parameter-efficient

强化学习。我们采用群体相对策略优化（GRPO；Shao 等人，2024）作为我们的强化学习算法，该算法在推理任务中表现出优越的效果和效率（DeepSeek-AI，2025）。与 PPO（Schulman 等人，2017）不同，GRPO 通过为每个定理提示采样一组候选证明，避免了单独的评论者模型，并基于它们的相对奖励优化策略。训练采用二元奖励，每个生成的 Lean 证明如果被验证为正确则获得奖励 1，否则为 0。为了确保有效学习，我们策划训练提示，仅包含那些难度适中但由监督微调模型可解的问题。在每次迭代中，我们采样 256 个不同的问题，为每个定理生成 32 个候选证明，最大序列长度为 32,768 个标记。

蒸馏。我们将DeepSeek-Prover-V1.5-Base-7B（Xin等，2024b）的最大上下文长度从4,096扩展到32,768个标记，并使用在DeepSeek-Prover-V2-671B的强化学习阶段收集的rollout数据对这个扩展上下文模型进行微调。除了CoT推理模式外，我们还结合在专家迭代过程中收集的非CoT证明数据，以实现一种成本高效的证明选项，该选项使用小型模型生成简洁的正式输出。此外，我们还执行了与训练671B模型相同的强化学习阶段，以提升DeepSeek-Prover-V2-7B的性能。

3. 实验结果

在本节中，我们对DeepSeek-Prover-V2在各种形式化定理证明基准数据集上的表现进行了系统评估，涵盖了高中竞赛题目和本科水平的数学题目。所有DeepSeek-Prover-V2的实验结果均在Lean 4.9.0环境下进行，使用与DeepSeek-Prover-V1.5（Xin等人，2024b）相同的测试环境。未另行说明的基线评估结果均来自它们各自的原始论文。

3.1. MiniF2F 基准测试结果

MiniF2F（Zheng 等人，2022）由来自多样数学资料的488个正式问题陈述组成，包括AIME、AMC和IMO竞赛，以及从MATH数据集（Hendrycks 等人，2021）中精选的问题。该基准测试涵盖了奥林匹克级别的问题，涉及初等数学的核心领域，包括代数、数论和归纳法。这些问题被划分为两个大小相等的子集，分别用miniF2F-valid和miniF2F-test表示，每个子集包含244个问题，并在学科领域中具有相同的分布。我们将miniF2F-test集专门用于评估模型性能，而miniF2F-valid问题则被纳入课程学习，结合子目标分解。我们采用了Wang 等人（2025）发布的miniF2F的修订版本，并在此基础上对miniF2F-valid进行了另外两次修订，对miniF2F-test进行了一次修订（见附录C）。

与最先进模型的比较。表1总结了在miniF2F测试数据集上评估的最先进形式定理证明建模的比较。实验结果表明，DeepSeek-Prover-V2-671B在miniF2F测试基准上创造了新的最先进性能，在采用CoT生成策略时，仅用32个样本就达到了前所未有的82.4%的准确率。值得注意的是，更加参数高效的

Method	Model size	Sample budget	miniF2F-test
Tree Search Methods			
Hypertree Proof Search (Lample et al., 2022)	600M	64×5000	41.0%
InternLM2.5-StepProver + BFS + CG (Wu et al., 2024)	7B	$256 \times 32 \times 600$	65.9%
HunyuanProver v16 + BFS + DC (Li et al., 2024)	7B	$600 \times 8 \times 400$	68.4%
BFS-Prover (Xin et al., 2025)	7B	$2048 \times 2 \times 600$	70.83% ± 0.89%
Whole-proof Generation Methods			
Leanabell-Prover-GD-RL (Zhang et al., 2025)	7B	128	61.1%
Goedel-Prover-SFT (Lin et al., 2025)	7B	25600	64.7%
STP (Dong and Ma, 2025)	7B	25600	67.6%
Kimina-Prover-Preview-Distill (Wang et al., 2025)	7B	1	52.5%
		32	63.1%
		1024	70.8%
Kimina-Prover-Preview (Wang et al., 2025)	72B	1	52.94%
		32	68.85%
		1024	77.87%
		8192	80.74%
DeepSeek-Prover-V2 (non-CoT)	7B	1	55.5% ± 1.4%
		32	68.0% ± 0.5%
		1024	73.2% ± 0.5%
		8192	75.0%
	671B	1	59.5% ± 1.4%
		32	73.8% ± 0.4%
		1024	76.7% ± 0.2%
		8192	78.3%
DeepSeek-Prover-V2 (CoT)	7B	1	58.6% ± 1.1%
		32	75.6% ± 0.5%
		1024	79.9% ± 0.3%
		8192	82.0%
	671B	1	61.9% ± 1.6%
		32	82.4% ± 0.6%
		1024	86.6% ± 0.3%
		8192	**88.9**%

Table 1 | Comparison with state-of-the-art models on the miniF2F-test dataset. The notation $\mu \pm \sigma$ denotes the average accuracy μ and the standard deviation σ. The tags CoT and non-CoT refer to two generation modes of a unified model, each guided by a different prompt.

Problem Category		miniF2F-valid curriculum (+Pass@8192)	miniF2F-test Pass@8192
Olympiad	IMO	10/20 = 50.0%	10/20 = 50.0%
	AIME	10(+2)/15 = 80.0%	14/15 = 93.3%
	AMC	39/45 = 86.7%	35/45 = 77.8%
MATH	Algebra	69/70 = 98.6%	70/70 = 100.0%
	Number Theory	58/60 = 96.7%	58/60 = 96.7%
Custom	Algebra	18/18 = 100.0%	15/18 = 83.3%
	Number Theory	8/8 = 100.0%	7/8 = 87.5%
	Induction	8/8 = 100.0%	8/8 = 100.0%
Overall Pass Rate		**220**(+**2**)/**244** = **91.0**%	**217**/**244** = **88.9**%

Table 2 | Problems solved by DeepSeek-Prover-V2-671B on the miniF2F benchmark. Results on miniF2F-valid are collected throughout the curriculum learning process, and DeepSeek-Prover-V2-671B is further invoked with Pass@8192 on the remaining problems.

Method	Model size	Sample budget	miniF2F-test
Tree Search Methods			
Hypertree Proof Search (Lample et al., 2022)	600M	64×5000	41.0%
InternLM2.5-StepProver + BFS + CG (Wu et al., 2024)	7B	$256 \times 32 \times 600$	65.9%
HunyuanProver v16 + BFS + DC (Li et al., 2024)	7B	$600 \times 8 \times 400$	68.4%
BFS-Prover (Xin et al., 2025)	7B	$2048 \times 2 \times 600$	70.83% ± 0.89%
Whole-proof Generation Methods			
Leanabell-Prover-GD-RL (Zhang et al., 2025)	7B	128	61.1%
Goedel-Prover-SFT (Lin et al., 2025)	7B	25600	64.7%
STP (Dong and Ma, 2025)	7B	25600	67.6%
Kimina-Prover-Preview-Distill (Wang et al., 2025)	7B	1	52.5%
		32	63.1%
		1024	70.8%
Kimina-Prover-Preview (Wang et al., 2025)	72B	1	52.94%
		32	68.85%
		1024	77.87%
		8192	80.74%
DeepSeek-Prover-V2 (non-CoT)	7B	1	55.5% ± 1.4%
		32	68.0% ± 0.5%
		1024	73.2% ± 0.5%
		8192	75.0%
	671B	1	59.5% ± 1.4%
		32	73.8% ± 0.4%
		1024	76.7% ± 0.2%
		8192	78.3%
DeepSeek-Prover-V2 (CoT)	7B	1	58.6% ± 1.1%
		32	75.6% ± 0.5%
		1024	79.9% ± 0.3%
		8192	82.0%
	671B	1	61.9% ± 1.6%
		32	82.4% ± 0.6%
		1024	86.6% ± 0.3%
		8192	**88.9**%

表1 | 在miniF2F-test数据集上与最先进模型的比较。符号 $\mu \pm \sigma$ 表示平均准确率，μ 表示标准差，σ。标签CoT和非CoT指的是统一模型的两种生成模式，每种由不同的提示引导。

Problem Category		miniF2F-valid curriculum (+Pass@8192)	miniF2F-test Pass@8192
Olympiad	IMO	10/20 = 50.0%	10/20 = 50.0%
	AIME	10(+2)/15 = 80.0%	14/15 = 93.3%
	AMC	39/45 = 86.7%	35/45 = 77.8%
MATH	Algebra	69/70 = 98.6%	70/70 = 100.0%
	Number Theory	58/60 = 96.7%	58/60 = 96.7%
Custom	Algebra	18/18 = 100.0%	15/18 = 83.3%
	Number Theory	8/8 = 100.0%	7/8 = 87.5%
	Induction	8/8 = 100.0%	8/8 = 100.0%
Overall Pass Rate		**220**(+**2**)/**244** = **91.0**%	**217**/**244** = **88.9**%

表2 | DeepSeek-Prover-V2-671B 在 miniF2F 基准测试中解决的问题。在 miniF2F-valid 上的结果是在整个课程学习过程中收集的，且在剩余问题上，DeepSeek-Prover-V2-671B 进一步以 Pass@8192 进行调用。

DeepSeek-Prover-V2-7B also exhibits competitive performance, surpassing all existing open-source theorem provers in the literature. The comparative analysis further reveals a compelling scaling pattern: as the sample budget increases from 1 to 8192, the performance gap between the 7B and 671B variants widens considerably, with the larger model demonstrating superior sample efficiency and a steeper improvement trajectory.

Proving Challenging Problems through Subgoal-guided Curricula. Table 2 presents a detailed breakdown of the problems solved by DeepSeek-Prover-V2 on the miniF2F benchmark, where it achieves strong overall performance with a 91.0% pass rate on the validation set and 88.9% on the test set. Remarkably, our subgoal-guided curriculum learning framework, which integrates the general-purpose model DeepSeek-V3 with a lightweight specialized 7B prover, achieves a 90.2% success rate on miniF2F-valid, nearly matching the performance of DeepSeek-Prover-V2-671B. These findings highlight the potential of state-of-the-art general-purpose LLMs to extend beyond natural language understanding and effectively support complex formal reasoning tasks. Through strategic subgoal decomposition, the model is able to break down challenging problems into a sequence of tractable steps, serving as an effective bridge between informal reasoning and formal proof construction.

CoT vs. non-CoT. The experimental results in Table 1 demonstrate a substantial performance advantage of the CoT reasoning mode over the non-CoT mode in formal mathematical reasoning. This reinforces the effectiveness of CoT prompting, which encourages decomposition of complex problems into intermediate steps, and further confirms that inference-time scaling holds in the domain of formal theorem proving. Complementing these findings, Table 3 provides statistics on the number of tokens generated by DeepSeek-Prover-V2 under different reasoning modes. As expected, the CoT mode produces significantly longer outputs, reflecting its sophisticated reasoning process. Interestingly, within the non-CoT setting, the 671B model generates longer outputs on average compared to the 7B model. A closer examination reveals that, although explicit reasoning is not prompted in the non-CoT mode, the larger model often inserts brief natural language comments within the proof code that resemble implicit reasoning steps (see Appendix A). This suggests that high-capacity models may internalize and externalize intermediate reasoning implicitly, even in the absence of explicit CoT prompting

#output tokens	non-CoT	CoT
7B	442.6	4488.5
671B	761.8	6751.9

Table 3 | Average number of tokens generated by DeepSeek-Prover-V2 on miniF2F-test.

3.2. Results on Undergraduate-level Benchmarks

ProofNet (Azerbayev et al., 2023) consists of 371 problems in Lean 3, drawn from a range of popular undergraduate pure mathematics textbooks, covering topics such as real and complex analysis, linear algebra, abstract algebra, and topology. We use the Lean 4 translation of ProofNet made available by Xin et al. (2024b), which is further divided into two splits: ProofNet-valid and ProofNet-test, containing 185 and 186 problems, respectively. The test split of ProofNet is reserved exclusively for model evaluation, as variants of the ProofNet-valid problems are included in the public synthetic dataset provided by Dong and Ma (2025), which is used in our supervised fine-tuning. The results, shown in Table 4, indicate a substantial improvement in the pass rate of DeepSeek-Prover-V2 when using CoT reasoning compared to the non-CoT setting. Notably, despite the training data being predominantly drawn from high-school

DeepSeek-Prover-V2-7B 也展现出具有竞争力的性能，超越了文献中所有现有的开源定理证明器。对比分析进一步揭示了一种令人信服的扩展模式：随着样本预算从 1 增加到 8192，7B 和 671B 变体之间的性能差距显著扩大，较大的模型表现出更优的样本效率和更陡峭的提升轨迹。

通过子目标引导课程证明具有挑战性的问题。表2详细列出了DeepSeek-Prover-V2在miniF2F基准测试中解决的问题细节，其中它在整体表现上表现出色，在验证集上的通过率为91.0%，在测试集上为88.9%。值得注意的是，我们的子目标引导课程学习框架，将通用模型DeepSeek-V3与轻量级的专用7B证明器结合起来，在miniF2F-valid上实现了90.2%的成功率，几乎与DeepSeek-Prover-V2-671B的性能相匹配。这些发现突显了最先进的通用大型语言模型在超越自然语言理解、有效支持复杂形式推理任务方面的潜力。通过战略性地分解子目标，模型能够将具有挑战性的问题拆解为一系列可处理的步骤，成为非正式推理与正式证明构建之间的有效桥梁。

CoT 与非-CoT。表1中的实验结果显示，在正式数学推理中，CoT推理模式相较于非-CoT模式具有显著的性能优势。这进一步证明了CoT提示的有效性，它鼓励将复杂问题分解为中间步骤，并进一步确认了推理-{{v*}}。

#output tokens	non-CoT	CoT
7B	442.6	4488.5
671B	761.8	6751.9

表3 | DeepSeek-Prover-V2 在 miniF2F 测试中生成的平均令牌数。

时间缩放在形式定理证明的领域中成立。补充这些发现的是，表3提供了在不同推理模式下DeepSeek-Prover-V2生成的标记数的统计数据。如预期，CoT模式产生的输出明显更长，反映了其复杂的推理过程。有趣的是，在非CoT设置中，671B模型生成的输出平均比7B模型更长。仔细观察发现，尽管在非CoT模式中没有明确提示进行推理，但较大的模型经常在证明代码中插入简短的自然语言评论，这些评论类似于隐式推理步骤（见附录A）。这表明高容量模型可能在没有明确CoT提示的情况下，隐式地内化和外化中间推理。

3.2. 本科水平基准测试的结果

ProofNet（Azerbayev 等人，2023）由371个Lean 3中的问题组成，来源于一系列流行的本科纯数学教材，涵盖实分析、复分析、线性代数、抽象代数和拓扑等主题。我们使用由Xin 等人（2024b）提供的Lean 4版本的ProofNet，该版本进一步分为两个部分：ProofNet-valid和ProofNet-test，分别包含185个和186个问题。ProofNet的测试集专门用于模型评估，因为ProofNet-valid问题的变体被包含在Dong和Ma（2025）提供的公共合成数据集中，该数据集用于我们的有监督微调。表4显示的结果表明，使用CoT推理的DeepSeek-Prover-V2的通过率相比非CoT设置有显著提升。值得注意的是，尽管训练数据主要来自高中

Method	Model size	Sample budget	ProofNet-test	PutnamBench
Goedel-Prover-SFT (Lin et al., 2025)	7B	32	15.6%	6/644
		512	-	7/644
STP (Dong and Ma, 2025)	7B	128	19.5% ± 0.7%	7/644
		3200	23.9% ± 0.6%	8/644
		25600	26.9%	-
DeepSeek-Prover-V2 (non-CoT)	7B	32	21.6% ± 0.2%	11/658
		128	23.1% ± 0.6%	15/658
		1024	24.7%	**23/658**
	671B	32	23.8% ± 0.2%	9/658
		128	27.2% ± 0.5%	11/658
		1024	31.2%	16/658
DeepSeek-Prover-V2 (CoT)	7B	32	23.0% ± 0.4%	9/658
		128	25.4% ± 0.7%	10/658
		1024	29.6%	11/658
	671B	32	30.5% ± 0.7%	22/658
		128	33.6% ± 0.3%	33/658
		1024	**37.1%**	**49/658**

Table 4 | The experimental results on ProofNet-test and PutnamBench. The scores for Goedel-Prover-SFT and STP on PutnamBench are sourced from their original papers, which conducted evaluations on an earlier version of PutnamBench comprising 644 problems.

level mathematics, the model exhibits strong generalization to more advanced, college-level mathematical problems, underscoring its robust formal reasoning capabilities.

PutnamBench (Tsoukalas et al., 2024) is a continuously updated benchmark featuring competition mathematics problems from the *William Lowell Putnam Mathematical Competition*, spanning the years 1962 to 2023. The Putnam Competition is a highly prestigious annual mathematics competition for undergraduate students across the United States and Canada, encompassing a variety of college-level domains such as analysis, linear algebra, abstract algebra, combinatorics, probability, and set theory. We evaluate our model on the latest release of PutnamBench, which contains 658 problems formalized in Lean 4. We exclude problems that are incompatible with Lean 4.9.0 and evaluate the model on the remaining set of 649 problems. As shown in Table 4, DeepSeek-Prover-V2-671B demonstrates enhanced reasoning capabilities in the PutnamBench, solving 49 problems and significantly outperforming its non-CoT counterpart. These results further highlight the effectiveness of the CoT reasoning approach in handling challenging, college-level mathematical problems.

Skill Discovery by Reinforcement Learning. An unexpected finding in our evaluation is the exceptional performance of DeepSeek-Prover-V2-7B with non-CoT generation mode on the PutnamBench dataset. Remarkably, this smaller 7B model successfully solves 13 problems that remain unsolved by its larger counterpart, DeepSeek-Prover-V2-671B, raising our total number of solved problems on PutnamBench from 49 to 62 out of 658. Upon closer examination of the model's outputs, we identified a distinctive pattern in its reasoning approach: the 7B model frequently employs `Cardinal.toNat` and `Cardinal.natCast_inj` to handle problems involving finite cardinalities (see examples in Appendix B), which are noticeably absent in the outputs generated by the 671B version. This technique appears to enable the model to effectively solve a subset of problems that require nuanced manipulation of cardinal values.

Method	Model size	Sample budget	ProofNet-test	PutnamBench
Goedel-Prover-SFT (Lin et al., 2025)	7B	32	15.6%	6/644
		512	-	7/644
STP (Dong and Ma, 2025)	7B	128	19.5% ± 0.7%	7/644
		3200	23.9% ± 0.6%	8/644
		25600	26.9%	-
DeepSeek-Prover-V2 (non-CoT)	7B	32	21.6% ± 0.2%	11/658
		128	23.1% ± 0.6%	15/658
		1024	24.7%	**23/658**
	671B	32	23.8% ± 0.2%	9/658
		128	27.2% ± 0.5%	11/658
		1024	31.2%	16/658
DeepSeek-Prover-V2 (CoT)	7B	32	23.0% ± 0.4%	9/658
		128	25.4% ± 0.7%	10/658
		1024	29.6%	11/658
	671B	32	30.5% ± 0.7%	22/658
		128	33.6% ± 0.3%	33/658
		1024	**37.1**%	**49/658**

表4 | 在ProofNet-test和PutnamBench上的实验结果。Goedel- Prover-SFT和STP在PutnamBench上的得分来自它们的原始论文，这些论文在较早版本的PutnamBench（包含644个问题）上进行了评估。

在高等数学方面，该模型展现出对更复杂的大学水平数学问题的强大泛化能力，突显其稳健的形式推理能力。

PutnamBench（Tsoukalas 等，2024）是一个持续更新的基准测试，包含来自 *William Lowell Putnam Mathematical Competition* 的竞赛数学题目，涵盖 1962 年至 2023 年的年份。普特南竞赛是美国和加拿大本科生每年举行的极具声望的数学竞赛，涉及分析、线性代数、抽象代数、组合学、概率和集合论等多种大学水平的领域。我们在最新版本的 PutnamBench 上评估我们的模型，该版本包含用 Lean 4 正式化的 658 道题目。我们排除与 Lean 4.9.0 不兼容的题目，并在剩余的 649 道题目上进行评估。如表 4 所示，DeepSeek-Prover-V2-671B 在 Putnam Bench 上展现出增强的推理能力，解决了 49 道题目，显著优于其非 CoT 版本。这些结果进一步突显了 CoT 推理方法在处理具有挑战性的大学数学题目中的有效性。

通过强化学习进行技能发现。在我们的评估中，一个意想不到的发现是，DeepSeek-Prover-V2-7B在非CoT生成模式下在PutnamBench数据集上的表现异常出色。值得注意的是，这个较小的7B模型成功解决了13个其更大版本DeepSeek-Prover-V2-671B未能解决的问题，使我们在PutnamBench上的总解决问题数从49增加到62（总共658个问题）。通过对模型输出的仔细观察，我们发现其推理方法中存在一个显著的模式：7B模型经常使用`Cardinal.toNat`和`Cardinal.natCast_inj`来处理涉及有限基数的问题（见附录B中的示例），而在671B版本生成的输出中明显缺少这种模式。这一技术似乎使模型能够有效解决一部分需要对基数值进行细致操作的问题。

Method	Model size	Sample budget	ProverBench All	ProverBench AIME 24&25
STP (Dong and Ma, 2025)	7B	32	27.5% ± 0.7%	0/15
		128	31.4% ± 1.1%	1/15
		512	36.3%	1/15
DeepSeek-Prover-V2 (non-CoT)	7B	32	47.7% ± 0.6%	1/15
		128	48.8% ± 0.2%	1/15
		512	49.5%	1/15
	671B	32	49.5% ± 0.5%	1/15
		128	51.5% ± 0.3%	2/15
		512	52.3%	2/15
DeepSeek-Prover-V2 (CoT)	7B	32	49.0% ± 0.3%	1/15
		128	50.8% ± 0.5%	1/15
		512	51.7%	1/15
	671B	32	52.9% ± 0.9%	4/15
		128	56.5% ± 0.5%	5/15
		512	**59.1%**	**6/15**

Table 6 | The experimental results on ProverBench. The `All` category represents the complete evaluation set consisting of 325 problems, while AIME 24&25 denotes a subset of 15 problems formalized from recent AIME competitions. The results for STP (Dong and Ma, 2025) are evaluated using the open-source model weights.

3.3. Results on Combinatorial Problems

CombiBench (Liu et al., 2025) is a comprehensive benchmark comprising 100 combinatorial competition problems formalized in Lean 4, each paired with its corresponding natural-language statement. We evaluate DeepSeek-Prover-V2 in the with-solution setting of this benchmark, where the correct answer is embedded in the Lean statement, allowing the evaluation to focus solely on proof generation. After filtering out problems incompatible with Lean 4.9.0 and those containing multiple `sorry` placeholders, we evaluate on 77 problems from the benchmark and successfully solve 12 of them. These results indicate that, while the prover model is primarily trained in number theory and algebra, it demonstrates promising generalization to combinatorial problems, despite their persistent difficulty.

CombiBench		Pass@16
Kimina-Prover-Preview (Wang et al., 2025)		7/100
DeepSeek-Prover-V2-7B	non-CoT	8/100
	CoT	10/100
DeepSeek-Prover-V2-671B	non-CoT	9/100
	CoT	**12/100**

Table 5 | Evaluation results on CombiBench under the with-solution setting.

3.4. ProverBench: Formalization of AIME and Textbook Problems

To enhance existing benchmarks and advance research in formal theorem proving, we introduce a benchmark dataset comprising 325 problems. Of these, 15 are formalized from number theory and algebra questions featured in the recent AIME competitions (AIME 24 and 25), offering authentic high-school competition-level challenges. The remaining 310 problems are drawn from curated textbook examples and educational tutorials, contributing a diverse and pedagogically grounded collection of formalized mathematical problems. This benchmark is designed to enable more comprehensive evaluation across both high-school competition problems and undergraduate-level mathematics.

Method	Model size	Sample budget	ProverBench	
			All	AIME 24&25
STP (Dong and Ma, 2025)	7B	32	27.5% ± 0.7%	0/15
		128	31.4% ± 1.1%	1/15
		512	36.3%	1/15
DeepSeek-Prover-V2 (non-CoT)	7B	32	47.7% ± 0.6%	1/15
		128	48.8% ± 0.2%	1/15
		512	49.5%	1/15
	671B	32	49.5% ± 0.5%	1/15
		128	51.5% ± 0.3%	2/15
		512	52.3%	2/15
DeepSeek-Prover-V2 (CoT)	7B	32	49.0% ± 0.3%	1/15
		128	50.8% ± 0.5%	1/15
		512	51.7%	1/15
	671B	32	52.9% ± 0.9%	4/15
		128	56.5% ± 0.5%	5/15
		512	**59.1**%	**6/15**

表6 | ProverBench的实验结果。`All`类别代表由325个问题组成的完整评估集，而AIME 24&25表示从近期AIME比赛中形式化的15个问题的子集。STP（Dong和Ma，2025）的结果使用开源模型权重进行评估。

3.3. 组合问题的结果

CombiBench（Liu 等，2025）是一个全面的基准测试，包括100个用Lean 4形式化的组合竞赛问题，每个问题都配有相应的自然语言陈述。我们在该基准的带解设置中评估DeepSeek-Prover-V2，其中正确答案嵌入在Lean陈述中，从而使评估仅专注于证明生成。在筛选掉与Lean 4.9.0不兼容的问题后，

CombiBench		Pass@16
Kimina-Prover-Preview (Wang et al., 2025)		7/100
DeepSeek-Prover-V2-7B	non-CoT	8/100
	CoT	10/100
DeepSeek-Prover-V2-671B	non-CoT	9/100
	CoT	**12/100**

表5 | 在带解题方案设置下对CombiBench的评估结果。

对于包含多个 `sorry` 占位符的情况，我们在基准测试的77个问题中进行了评估，成功解决了其中的12个。这些结果表明，尽管证明器模型主要在数论和代数方面进行训练，但它在组合问题上的泛化能力仍然令人鼓舞，尽管这些问题一直具有挑战性。

3.4. ProverBench：AIME 和教科书问题的形式化

为了增强现有的基准测试并推动形式定理证明领域的研究，我们引入了一个包含 325 个问题的基准数据集。其中，15 个问题是从近期 AIME 竞赛（AIME 24 和 25）中的数论和代数题目正式化而来，提供了具有真实高中竞赛水平的挑战。其余的 310 个问题来自精心挑选的教科书示例和教育教程，构成了一个多样化且具有教育基础的形式化数学问题集。该基准旨在实现对高中竞赛题目和本科水平数学的更全面评估。

AIME Formalization. The *American Invitational Mathematics Examination* (AIME) is an annual mathematics competition designed to challenge and recognize talented high school students who demonstrate exceptional proficiency in mathematics. The problems from AIME 24&25 have become a standard benchmark for evaluating the reasoning capabilities of large language models. In order to bridge the evaluation of model performance across formal and informal mathematical reasoning, we curate and formalize a subset of problems from AIME 24&25. To ensure cleaner formalizations, we filter out geometry, combinatorics, and counting problems whose representations in Lean are potentially cumbersome. This results in 15 selected problems, covering competition-level topics in elementary number theory and algebra. We evaluate DeepSeek-V3-0324 on the selected set of problems using the standard find-answer task for natural-language mathematical reasoning. With majority voting over 16 sampled responses, the model successfully solves 8 out of 15 problems. In comparison, DeepSeek-Prover-V2-671B, operating under the formal proof generation setting with given correct answers, is able to construct valid formal proofs for 6 of 15 problems. This comparison highlights that the performance gap between informal mathematical reasoning and formal theorem proving is substantially narrowing, indicating growing alignment between linguistic understanding and formal logical rigor in advanced language models.

Contest	Problems
AIME 24I	**P2** , **P7** , **P13**
AIME 24II	**P4** , P7, P13 , P14
AIME 25I	**P1** , P8 , P9, P11
AIME 25II	**P2** , P4 , P13, P15

Table 7 | Selection of AIME 24&25 problems for formalization. Problems with underlined bolded indices have been solved by DeepSeek-Prover-V2. Problems solved by DeepSeek-V3-0324 using Maj@16 are highlighted with a gray background.

Textbook Formalization. In addition to AIME 24&25, we augment our benchmark with problems carefully selected from textbooks used in high school competitions and undergraduate-level courses to strengthen coverage in specific mathematical domains. This curation process ensures comprehensive representation across difficulty levels and topic areas. As a result, we formalize 310 problems that encompass a broad spectrum, ranging from elementary mathematics at the competition level to advanced topics typically encountered in undergraduate studies. This comprehensive benchmark covers number theory, elementary algebra, linear algebra, abstract algebra, calculus, real analysis, complex analysis, functional analysis, and probability. The deliberate inclusion of this diverse array of mathematical fields allows for a thorough assessment of model capabilities across varying levels of abstraction and reasoning styles. Number theory and algebra problems test a model's facility with discrete structures and equations, while analysis-oriented problems evaluate understanding of limits, continuity, and calculus. The abstract algebra and functional analysis components challenge models to reason about abstract structures and spaces, requiring sophisticated formal reasoning capabilities. The evaluation results are presented in Table 6. As shown, DeepSeek-Prover-V2-

Area	Count
AIME 24&25	15
Number Theory	40
Elementary Algebra	30
Linear Algebra	50
Abstract Algebra	40
Calculus	90
Real Analysis	30
Complex Analysis	10
Functional Analysis	10
Probability	10
Total	325

Table 8 | Distribution of mathematical areas represented in ProverBench.

AIME 正式化。*American Invitational Mathematics Examination* (AIME) 是一项年度数学竞赛，旨在挑战并表彰展现出卓越数学能力的优秀高中生。AIME 24&25 的题目已成为评估大型语言模型推理能力的标准基准。为了弥合正式与非正式数学推理模型性能的评估差距，我们策划并形式化了 AIME 24&25 中的部分题目。为了确保更清晰的正式化——

Contest	Problems
AIME 24I	**P2** , **P7** , **P13**
AIME 24II	**P4** , P7, P13 , P14
AIME 25I	**P1** , P8 , P9, P11
AIME 25II	**P2** , P4 , P13, P15

表7 | AIME 24&25题目正式化的选择。带有下划线加粗索引的问题已由DeepSeek-Prover-V2解决。使用Maj@16由DeepSeek-V3-0324解决的问题用灰色背景突出显示。

在这些工作中，我们筛选出几何、组合和计数问题，这些问题在 Lean 中的表示可能较为繁琐。最终选出了 15 个问题，涵盖了初等数论和代数中的竞赛级主题。我们使用标准的自然语言数学推理找答案任务，在选定的问题集上评估了 DeepSeek-V3-0324。通过对 16 个采样响应进行多数投票，模型成功解决了其中的 8 个问题。相比之下，在给定正确答案的正式证明生成设置下运行的 DeepSeek-Prover-V2-671B 能够为 15 个问题中的 6 个构建有效的正式证明。这一对比突显了非正式数学推理与正式定理证明之间的性能差距正显著缩小，表明在先进的语言模型中，语言理解与正式逻辑严谨性之间的趋同日益增强。

教科书形式化。除了AIME 24&25之外，我们还通过精心挑选的高中竞赛和本科课程中使用的教科书中的题目，增强了我们的基准测试，以加强在特定数学领域的覆盖。这一策划过程确保了在难度级别和主题领域的全面代表性。因此，我们形式化了310个题目，涵盖了从竞赛水平的基础数学到本科阶段常见的高级主题的广泛范围。这个全面的基准测试涵盖数论、初等代数、线性代数、抽象代数、微积分、实分析、复分析、泛函分析和概率。故意包含这一多样的数学领域，有助于全面评估模型在不同抽象层次上的能力。

Area	Count
AIME 24&25	15
Number Theory	40
Elementary Algebra	30
Linear Algebra	50
Abstract Algebra	40
Calculus	90
Real Analysis	30
Complex Analysis	10
Functional Analysis	10
Probability	10
Total	325

表8 | ProverBench中表示的数学领域分布。

以及推理风格。数论和代数问题测试模型对离散结构和方程的熟练程度，而分析导向的问题则评估对极限、连续性和微积分的理解。抽象代数和泛函分析部分挑战模型对抽象结构和空间的推理能力，要求具备复杂的形式推理能力。评估结果显示在表6中。如图所示，DeepSeek-Prover-V2-

671B with CoT reasoning consistently outperforms all baselines, reinforcing the trends observed in other benchmark evaluations.

4. Conclusion

In this work, we propose a comprehensive pipeline for synthesizing cold-start reasoning data to advance formal theorem proving. Our data construction process is grounded in a recursive theorem-proving framework, wherein DeepSeek-V3 serves as a unified model for both subgoal decomposition and lemma formalization within the Lean 4 proof assistant. Our approach combines high-level proof sketches with formal steps, creating a sequence of manageable subgoals that can be efficiently solved using a smaller 7B model, significantly reducing computational requirements. The curriculum learning framework we developed uses these decomposed subgoals to generate increasingly difficult training tasks, creating a more effective learning progression. By pairing complete formal proofs with DeepSeek-V3's chain-of-thought reasoning, we established valuable cold-start reasoning data that bridges informal mathematical thinking with formal proof structures. The subsequent reinforcement learning stage substantially enhanced this connection, leading to significant improvements in formal theorem proving capabilities. The resulting model, DeepSeek-Prover-V2-671B, consistently outperforms all baselines across a range of benchmarks, spanning both high-school competition problems and undergraduate-level mathematics. Our future work will focus on scaling this paradigm to an AlphaProof-like system with the ultimate aim of tackling IMO-level mathematical problems that represent the frontier of automated theorem proving challenges.

References

Z. Azerbayev, B. Piotrowski, H. Schoelkopf, E. W. Ayers, D. Radev, and J. Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. arXiv preprint arXiv:2302.12433, 2023.

B. Barras, S. Boutin, C. Cornes, J. Courant, Y. Coscoy, D. Delahaye, D. de Rauglaudre, J.-C. Filliâtre, E. Giménez, H. Herbelin, et al. The Coq proof assistant reference manual. INRIA, version, 6(11):17–21, 1999.

A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. Discrete event dynamic systems, 13:341–379, 2003.

DeepMind. AI achieves silver-medal standard solving international mathematical olympiad problems. https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/, 2024.

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

K. Dong and T. Ma. STP: Self-play llm theorem provers with iterative conjecturing and proving. arXiv preprint arXiv:2502.00212, 2025.

K. Dong, A. Mahankali, and T. Ma. Formal theorem proving by rewarding llms to decompose proofs hierarchically. arXiv preprint arXiv:2411.01829, 2024.

671B配合CoT推理始终优于所有基线，进一步巩固了在其他基准测试中观察到的趋势。

4. 结论

在本工作中，我们提出了一个用于合成冷启动推理数据的全面流程，以推动形式定理证明的发展。我们的数据构建过程基于递归定理证明框架，其中DeepSeek-V3作为一个统一模型，用于Lean 4证明助手中的子目标分解和引理形式化。我们的方法结合了高层次的证明草图与形式化步骤，创建了一系列可管理的子目标，这些子目标可以通过较小的7B模型高效地解决，从而显著降低计算需求。我们开发的课程学习框架利用这些分解的子目标生成越来越困难的训练任务，形成更有效的学习进程。通过将完整的形式化证明与DeepSeek-V3的链式推理相结合，我们建立了有价值的冷启动推理数据，桥接了非正式的数学思维与正式的证明结构。随后的强化学习阶段大大增强了这种联系，显著提升了形式定理证明的能力。最终模型DeepSeek-Prover-V2-671B在多个基准测试中持续优于所有基线，涵盖高中竞赛题目和本科水平的数学问题。我们的未来工作将专注于将这一范式扩展到类似AlphaProof的系统，最终目标是解决IMO级别的数学问题，这些问题代表了自动定理证明挑战的前沿。

参考文献

Z. Azerbayev, B. Piotrowski, H. Schoelkopf, E. W. Ayers, D. Radev 和 J. Avigad. ProofNet：自动形式化和形式证明本科水平的数学。arXiv 预印本 arXiv:2302.12433，2023年。B. Barras, S. Boutin, C. Cornes, J. Courant, Y. Coscoy, D. Delahaye, D. de Rauglaudre, J.-C. Filliâtre, E. Giménez, H. Herbelin 等。Coq 证明助手参考手册。INRIA，版本，6(11):17–21，1999年。A. G. Barto 和 S. Mahadevan。层次强化学习的最新进展。离散事件动态系统，13：341–379，2003年。DeepMind。人工智能在解决国际数学奥林匹克问题方面达到了银牌标准。https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/，2024年。DeepSeek-AI。Deepseek-v3 技术报告，2024年。网址——https://arxiv.org/abs/2412.194 37。DeepSeek-AI。Deepseek-r1：通过强化学习激励LLMs的推理能力，2025年。网址 https://arxiv.org/abs/2501.12948。K. Dong 和 T. Ma。STP：具有迭代猜测和证明的自我对弈LLM定理证明器。arXiv 预印本 arXiv:2502.00212，2025年。K. Dong、A. Mahankali 和 T. Ma。通过奖励LLMs分层分解证明的正式定理证明。arXiv 预印本 arXiv:2411.01829，2024年。

M. Eppe, C. Gumbsch, M. Kerzel, P. D. Nguyen, M. V. Butz, and S. Wermter. Intelligent problem-solving as integrated hierarchical reinforcement learning. Nature Machine Intelligence, 4(1): 11–20, 2022.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021.

A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, et al. Openai o1 system card. arXiv preprint arXiv:2412.16720, 2024.

A. Q. Jiang, S. Welleck, J. P. Zhou, T. Lacroix, J. Liu, W. Li, M. Jamnik, G. Lample, and Y. Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In The Eleventh International Conference on Learning Representations, 2023.

G. Lample, M.-A. Lachaux, T. Lavril, X. Martinet, A. Hayat, G. Ebner, A. Rodriguez, and T. Lacroix. Hypertree proof search for neural theorem proving. In Proceedings of the 36th International Conference on Neural Information Processing Systems, pages 26337–26349, 2022.

Y. Li, D. Du, L. Song, C. Li, W. Wang, T. Yang, and H. Mi. Hunyuanprover: A scalable data synthesis framework and guided tree search for automated theorem proving. arXiv preprint arXiv:2412.20735, 2024.

Y. Lin, S. Tang, B. Lyu, J. Wu, H. Lin, K. Yang, J. Li, M. Xia, D. Chen, S. Arora, et al. Goedel-Prover: A frontier model for open-source automated theorem proving. arXiv preprint arXiv:2502.07640, 2025.

J. Liu, X. Lin, J. Bayer, Y. Dillies, W. Jiang, X. Liang, R. Soletskyi, H. Wang, Y. Xie, B. Xiong, et al. CombiBench: Benchmarking llm capability for combinatorial mathematics. https://moonshotai.github.io/CombiBench/, 2025.

L. d. Moura and S. Ullrich. The Lean 4 theorem prover and programming language. In Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28, pages 625–635. Springer, 2021.

O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. Advances in neural information processing systems, 31, 2018.

L. C. Paulson. Isabelle a Generic Theorem Prover. Springer Verlag, 1994.

S. Polu and I. Sutskever. Generative language modeling for automated theorem proving. arXiv preprint arXiv:2009.03393, 2020.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, and D. Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

G. Tsoukalas, J. Lee, J. Jennings, J. Xin, M. Ding, M. Jennings, A. Thakur, and S. Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the putnam mathematical competition. In The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2024.

M. Eppe, C. Gumbsch, M. Kerzel, P. D. Nguyen, M. V. Butz 和 S. Wermter。作为集成层次强化学习的智能问题解决。《自然机器智能》，第4卷第1期：11–20，2022年。D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song 和 J. Steinhardt。使用 MATH 数据集衡量数学问题解决能力。在第35届神经信息处理系统会议的数据集与基准测试环节（第2轮），2021年。

A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, 等. Openai o1 系统卡片. arXiv 预印本 arXiv:2412.16720, 2024.

A. Q. Jiang、S. Welleck、J. P. Zhou、T. Lacroix、J. Liu、W. Li、M. Jamnik、G. Lample 和 Y. Wu。草稿、草图与证明：用非正式证明引导形式定理证明器。收录于第十一届国际学习表征会议，2023年。

G. Lample, M.-A. Lachaux, T. Lavril, X. Martinet, A. Hayat, G. Ebner, A. Rodriguez 和 T. Lacroix. 用于神经定理证明的超树证明搜索。发表于第36届国际神经信息处理系统会议论文集，第26337–26349页，2022年。

Y. Li, D. Du, L. Song, C. Li, W. Wang, T. Yang, 和 H. Mi. Hunyuanprover：一种可扩展的数据合成框架和引导树搜索用于自动定理证明。arXiv 预印本 arXiv:2412.20735, 2024。

Y. Lin, S. Tang, B. Lyu, J. Wu, H. Lin, K. Yang, J. Li, M. Xia, D. Chen, S. Arora, 等. Goedel- Prover：一种用于开源自动定理证明的前沿模型。arXiv 预印本 arXiv:2502.07640, 2025。

J. Liu, X. Lin, J. Bayer, Y. Dillies, W. Jiang, X. Liang, R. Soletskyi, H. Wang, Y. Xie, B. Xiong 等. CombiBench：用于组合数学的大模型能力基准测试。`https://moonshotai.github.io/CombiBench/`，2025年。

L. d. Moura 和 S. Ullrich. Lean 4 定理证明器与编程语言。在《自动推理–CADE 28：第28届国际自动推理会议》，虚拟会议，2021年7月12–15日，论文集第28卷，第625–635页。Springer，2021。

O. Nachum, S. S. Gu, H. Lee, 和 S. Levine. 数据高效的层次强化学习。神经信息处理系统进展，31，2018年。L. C. Paulson. Isabelle 通用定理证明器。Springer Verlag，1994年。S. Polu 和 I. Sutskever. 用于自动定理证明的生成式语言建模。arXiv预印本 arXiv:2009.03393，2020年。

J. Schulman, F. Wolski, P. Dhariwal, A. Radford 和 O. Klimov。近端策略优化算法。arXiv 预印本 arXiv:1707.06347，2017。

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, 和 D. Guo. DeepSeekMath：推动开放语言模型中数学推理的极限。arXiv 预印本 arXiv:2402.03300, 2024。

G. Tsoukalas, J. Lee, J. Jennings, J. Xin, M. Ding, M. Jennings, A. Thakur 和 S. Chaudhuri。PutnamBench：在普特南数学竞赛中评估神经定理证明器。收录于 2024 年第三十八届神经信息处理系统会议（NeurIPS）数据集与基准测试专场。

H. Wang, H. Xin, Z. Liu, W. Li, Y. Huang, J. Lu, Y. Zhicheng, J. Tang, J. Yin, Z. Li, et al. Proving theorems recursively. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024a.

H. Wang, H. Xin, C. Zheng, Z. Liu, Q. Cao, Y. Huang, J. Xiong, H. Shi, E. Xie, J. Yin, et al. Lego-prover: Neural theorem proving with growing libraries. In The Twelfth International Conference on Learning Representations, 2024b.

H. Wang, M. Unsal, X. Lin, M. Baksys, J. Liu, M. D. Santos, F. Sung, M. Vinyes, Z. Ying, Z. Zhu, et al. Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning. arXiv preprint arXiv:2504.11354, 2025.

Z. Wu, S. Huang, Z. Zhou, H. Ying, J. Wang, D. Lin, and K. Chen. Internlm2. 5-stepprover: Advancing automated theorem proving via expert iteration on large-scale lean problems. arXiv preprint arXiv:2410.15700, 2024.

H. Xin, D. Guo, Z. Shao, Z. Ren, Q. Zhu, B. Liu, C. Ruan, W. Li, and X. Liang. DeepSeek-Prover: Advancing theorem proving in llms through large-scale synthetic data. arXiv preprint arXiv:2405.14333, 2024a.

H. Xin, Z. Ren, J. Song, Z. Shao, W. Zhao, H. Wang, B. Liu, L. Zhang, X. Lu, Q. Du, et al. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. arXiv preprint arXiv:2408.08152, 2024b.

R. Xin, C. Xi, J. Yang, F. Chen, H. Wu, X. Xiao, Y. Sun, S. Zheng, and K. Shen. BFS-Prover: Scalable best-first tree search for llm-based automatic theorem proving. arXiv preprint arXiv:2502.03438, 2025.

K. Yang, G. Poesia, J. He, W. Li, K. Lauter, S. Chaudhuri, and D. Song. Formal mathematical reasoning: A new frontier in AI. arXiv preprint arXiv:2412.16075, 2024.

H. Ying, Z. Wu, Y. Geng, J. Wang, D. Lin, and K. Chen. Lean workbook: A large-scale lean problem set formalized from natural language math problems. In The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2024.

J. Zhang, Q. Wang, X. Ji, Y. Liu, Y. Yue, F. Zhang, D. Zhang, G. Zhou, and K. Gai. Leanabell-prover: Posttraining scaling in formal reasoning. arXiv preprint arXiv:2504.06122, 2025.

X. Zhao, W. Li, and L. Kong. Decomposing the enigma: Subgoal-based demonstration learning for formal theorem proving. arXiv preprint arXiv:2305.16366, 2023.

X. Zhao, L. Zheng, H. Bo, C. Hu, U. Thakker, and L. Kong. Subgoalxl: Subgoal-based expert learning for theorem proving. arXiv preprint arXiv:2408.11172, 2024.

C. Zheng, H. Wang, E. Xie, Z. Liu, J. Sun, H. Xin, J. Shen, Z. Li, and Y. Li. Lyra: Orchestrating dual correction in automated theorem proving. Transactions on Machine Learning Research, 2024.

K. Zheng, J. M. Han, and S. Polu. miniF2F: a cross-system benchmark for formal olympiad-level mathematics. In International Conference on Learning Representations, 2022.

H. Wang, H. Xin, Z. Liu, W. Li, Y. Huang, J. Lu, Y. Zhicheng, J. Tang, J. Yin, Z. Li 等. 递归证明定理。收录于《第38届神经信息处理系统年会》，2024a。H. Wang, H. Xin, C. Zheng, Z. Liu, Q. Cao, Y. Huang, J. Xiong, H. Shi, E. Xie, J. Yin 等. Lego-prover：具有增长库的神经定理证明。收录于《第12届学习表示国际会议》，2024b。H. Wang, M. Unsal, X. Lin, M. Baksys, J. Liu, M. D. Santos, F. Sung, M. Vinyes, Z. Ying, Z. Zhu 等. Kimina-Prover预览：面向强化学习的大型形式推理模型。arXiv预印本 arXiv:2504.11354，2025。

Z. Wu, S. Huang, Z. Zhou, H. Ying, J. Wang, D. Lin, 和 K. Chen. Internlm2. 5步验证器：通过在大规模精益问题上的专家迭代推进自动定理证明。arXiv预印本 arXiv:2410.15700, 2024。

H. Xin, D. Guo, Z. Shao, Z. Ren, Q. Zhu, B. Liu, C. Ruan, W. Li, 和 X. Liang. DeepSeek-Prover：通过大规模合成数据推进LLMs中的定理证明。arXiv预印本 arXiv:2405.14333, 2024a。

H. Xin, Z. Ren, J. Song, Z. Shao, W. Zhao, H. Wang, B. Liu, L. Zhang, X. Lu, Q. Du, 等. DeepSeek-Prover-V1.5：利用证明助手反馈进行强化学习和蒙特卡洛树搜索。arXiv 预印本 arXiv:2408.08152, 2024b.

R. Xin, C. Xi, J. Yang, F. Chen, H. Wu, X. Xiao, Y. Sun, S. Zheng, 和 K. Shen. BFS-Prover：基于大规模语言模型的自动定理证明的可扩展优先树搜索。arXiv预印本 arXiv:2502.03438, 2025。

K. Yang, G. Poesia, J. He, W. Li, K. Lauter, S. Chaudhuri, 和 D. Song. 正式数学推理：人工智能的新前沿。arXiv 预印本 arXiv:2412.16075, 2024。

H. Ying, Z. Wu, Y. Geng, J. Wang, D. Lin, 和 K. Chen. 精益工作簿：从自然语言数学问题形式化的大规模精益问题集。发表于 2024 年第三十八届神经信息处理系统会议 数据集与基准赛道。

J. Zhang, Q. Wang, X. Ji, Y. Liu, Y. Yue, F. Zhang, D. Zhang, G. Zhou, 和 K. Gai. Leanabell-prover：正式推理中的后训练缩放。arXiv 预印本 arXiv:2504.06122, 2025。

X. Zhao, W. Li, 和 L. Kong. 分解谜题：基于子目标的演示学习用于形式定理证明。arXiv预印本 arXiv:2305.16366, 2023。

X. Zhao, L. Zheng, H. Bo, C. Hu, U. Thakker, 和 L. Kong. Subgoalxl：基于子目标的专家学习用于定理证明。arXiv预印本 arXiv:2408.11172, 2024。

C. Zheng, H. Wang, E. Xie, Z. Liu, J. Sun, H. Xin, J. Shen, Z. Li, 和 Y. Li. Lyra：在自动定理证明中协调双重校正。机器学习研究汇刊，2024年。K. Zheng, J. M. Han, 和 S. Polu. miniF2F：一个面向正式奥林匹克水平数学的跨系统基准。在国际学习表示会议，2022年。

A. Examples of Non-CoT and CoT Prompting for Proof Generation

A.1. Non-CoT Generation of DeepSeek-Prover-V2

Input:

````
Complete the following Lean 4 code:

```lean4
import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat

/-- Prove that cos π/7 - cos 2π/7 + cos 3π/7 = 1/2. -/
theorem imo_1963_p5 : Real.cos (Real.pi / 7) - Real.cos (2 * Real.pi / 7) +
    Real.cos (3 * Real.pi / 7) = 1 / 2 := by
  sorry
```
````

Output:

````
```lean4
import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat

/-- Prove that cos π/7 - cos 2π/7 + cos 3π/7 = 1/2. -/
theorem imo_1963_p5 : Real.cos (Real.pi / 7) - Real.cos (2 * Real.pi / 7) + Real.cos (3 * Real.pi / 7) = 1
    / 2 := by
  have h₁ : Real.cos (Real.pi / 7) > 0 := Real.cos_pos_of_mem_Ioo ⟨by linarith [Real.pi_gt_three], by
    linarith [Real.pi_gt_three]⟩
  have h₂ : Real.cos (2 * Real.pi / 7) = Real.cos (2 * Real.pi / 7) := rfl
  have h₃ : Real.cos (3 * Real.pi / 7) = Real.cos (3 * Real.pi / 7) := rfl
  have h₄ : Real.cos (Real.pi / 7) - Real.cos (2 * Real.pi / 7) + Real.cos (3 * Real.pi / 7) = 1 / 2 := by
    -- Use the identity for the sum of cosines with specific angles to simplify the expression
    have h₅ : Real.cos (2 * Real.pi / 7) = Real.cos (2 * (Real.pi / 7)) := by ring
    have h₆ : Real.cos (3 * Real.pi / 7) = Real.cos (3 * (Real.pi / 7)) := by ring
    rw [h₅, h₆]
    have h₇ : Real.cos (2 * (Real.pi / 7)) = 2 * Real.cos (Real.pi / 7) ^ 2 - 1 := by
      have h₇₁ : Real.cos (2 * (Real.pi / 7)) = 2 * Real.cos (Real.pi / 7) ^ 2 - 1 := by
        rw [Real.cos_two_mul]
        <;> ring_nf
        <;> nlinarith [Real.cos_sq_add_sin_sq (Real.pi / 7)]
      rw [h₇₁]
    have h₈ : Real.cos (3 * (Real.pi / 7)) = 4 * Real.cos (Real.pi / 7) ^ 3 - 3 * Real.cos (Real.pi / 7) := by
      have h₈₁ : Real.cos (3 * (Real.pi / 7)) = 4 * Real.cos (Real.pi / 7) ^ 3 - 3 * Real.cos (Real.pi / 7) := by
        rw [show Real.cos (3 * (Real.pi / 7)) = Real.cos (3 * (Real.pi / 7)) by rfl]
        have h₈₂ : Real.cos (3 * (Real.pi / 7)) = 4 * Real.cos (Real.pi / 7) ^ 3 - 3 * Real.cos (Real.pi / 7) := by
          rw [show Real.cos (3 * (Real.pi / 7)) = Real.cos (3 * (Real.pi / 7)) by rfl]
          -- Use the triple angle identity for cosine
          have h₈₃ : Real.cos (3 * (Real.pi / 7)) = 4 * Real.cos (Real.pi / 7) ^ 3 - 3 * Real.cos (Real.pi / 7) := by
            rw [Real.cos_three_mul]
            <;> ring_nf
````
````

A. 非CoT和CoT提示用于证明生成的示例

A.1. DeepSeek-Prover-V2的非CoT生成

*Input*Translated Text:

```
Complete the following Lean 4 code:

'''lean4
import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat

/-- Prove that cos π/7 − cos 2π/7 + cos 3π/7 = 1/2 .-/
theorem imo_1963_p5 : Real.cos (Real.pi / 7) - Real.cos (2 * Real.pi / 7) +
    Real.cos (3 * Real.pi / 7) = 1 / 2 := by
  sorry
'''
```

*Output*Translated Text:

```
 ' 'set_option maxHeartbeats 0 open BigOperators Real Nat Topology Rat /-- 证明余弦 π7 − 余弦 2π7
+ 余弦 3π7 _ 12 .−/ theorem imo_1963_p5 : Real.cos (Real.pi / 7) − Real.cos (2 * Real.pi / 7) + Real.cos (3 *
Real.pi / 7) = 1 / 2 := by1 : Real.cos (Real.pi / 7) > 0 := Real.cos_pos_of_mem_Ioo ⟨by linarith [Real.
pi_gt_three], by linarith [Real.pi_gt_three]⟩2 : Real.cos (2 * Real.pi / 7) = Real.cos (2 * Real.pi / 7) :
= rfl have h3 : Real.cos (3 * Real.pi / 7) = Real.cos (3 * Real.pi / 7) := rfl have h4 : Real.cos (Real.pi / 7
) − Real.cos (2 * Real.pi / 7) + Real.cos (3 * Real.pi / 7) = 1 / 2 := by
−− Use the identity for the sum of cosines with specific angles to simplify the expression have h5 : Real.cos (2 * Real.pi
/ 7) = Real.cos (2 * (Real.pi / 7)) := by ring have h6 : Real.cos (3 * Real.pi / 7) = Real.cos (3 * (Real.pi / 7))
:= by ring rw [h5, h6] have h7 : Real.cos (2 * (Real.pi / 7)) = 2 * Real.cos (Real.pi / 7) ^2 − 1 := by have h71 :
Real.cos (2 * (Real.pi / 7)) = 2 * Real.cos (Real.pi / 7) ^2 − 1 := by rw [Real.cos_two_mul] <;> ring_nf;
> nlinarith [Real.cos_sq_add_sin_sq (Real.pi / 7)] rw [h71] have h8 : Real.cos (3 * (Real.pi / 7)) = 4 *
Real.cos (Real.pi / 7) ^3 − 3 * Real.cos (Real.pi / 7) := by have h81 : Real.cos (3 * (Real.pi / 7)) = 4 * Real.
cos (Real.pi / 7) ^3 − 3 * Real.cos (Real.pi / 7) := by rw [show Real.cos (3 * (Real.pi / 7)) = Real.cos (3 * (
Real.pi / 7)) by rfl] have h82 : Real.cos (3 * (Real.pi / 7)) = 4 * Real.cos (Real.pi / 7) ^3 − 3 * Real.
cos (Real.pi / 7) := by rw [show Real.cos (3 * (Real.pi / 7)) = Real.cos (3 * (Real.pi / 7)) by rfl]
−− Use the triple angle identity for cosine83 : Real.cos (3 * (Real.pi / 7)) = 4 * Real.cos (Real.pi / 7) ^3 − 3 * Real.
cos (Real.pi / 7) := by rw [Real.cos_three_mul] <;> ring_nf
```

```
    <;> nlinarith [Real.cos_sq_add_sin_sq (Real.pi / 7), Real.sin_le_one (Real.pi / 7),
   Real.cos_le_one (Real.pi / 7)]
    rw [h₈₃]
   rw [h₈₂]
  rw [h₈₁]
rw [h₇, h₈]
-- Simplify the expression using the derived identities
have h₉ : Real.cos (Real.pi / 7) > 0 := h₁
have h₁₀ : Real.cos (Real.pi / 7) ≠ 0 := by linarith
-- Use the quadratic equation to solve for the sum of cosines
have h₁₁ : Real.cos (Real.pi / 7) = Real.cos (Real.pi / 7) := rfl
have h₁₂ : 8 * Real.cos (Real.pi / 7) ^ 3 - 4 * Real.cos (Real.pi / 7) ^ 2 - 4 * Real.cos (Real.pi / 7) + 1 = 0
   := by
  -- Use the identity to form a polynomial equation and solve it
  have h₁₃ : Real.cos (Real.pi / 7) > 0 := h₁
  have h₁₄ : Real.cos (Real.pi / 7) ≠ 0 := by linarith
  have h₁₅ : Real.cos (Real.pi / 7) = Real.cos (Real.pi / 7) := rfl
  -- Use the polynomial equation to find the sum of cosines
  have h₁₆ : Real.cos (Real.pi / 7) ≥ 0 := by linarith
  -- Use the identity to form a polynomial equation and solve it
  have h₁₇ : Real.sin (Real.pi / 7) > 0 := by
   apply Real.sin_pos_of_pos_of_lt_pi
   <;> linarith [Real.pi_gt_three]
  have h₁₈ : Real.sin (3 * (Real.pi / 7)) > 0 := by
   apply Real.sin_pos_of_pos_of_lt_pi
   <;>
    (try ring_nf at * <;> nlinarith [Real.pi_gt_three]) <;>
    (try ring_nf) <;>
    (try nlinarith [Real.pi_gt_three])
  have h₁₉ : Real.sin (3 * (Real.pi / 7)) = 3 * Real.sin (Real.pi / 7) - 4 * Real.sin (Real.pi / 7) ^ 3 := by
   have h₁₉₁ : Real.sin (3 * (Real.pi / 7)) = 3 * Real.sin (Real.pi / 7) - 4 * Real.sin (Real.pi / 7) ^ 3 := by
    rw [Real.sin_three_mul]
    <;> ring_nf
    <;> nlinarith [Real.cos_sq_add_sin_sq (Real.pi / 7), Real.sin_le_one (Real.pi / 7),
   Real.cos_le_one (Real.pi / 7)]
   rw [h₁₉₁]
  have h₂₀ : Real.sin (2 * (Real.pi / 7)) = 2 * Real.sin (Real.pi / 7) * Real.cos (Real.pi / 7) := by
   have h₂₀₁ : Real.sin (2 * (Real.pi / 7)) = 2 * Real.sin (Real.pi / 7) * Real.cos (Real.pi / 7) := by
    rw [Real.sin_two_mul]
    <;> ring_nf
    <;> nlinarith [Real.cos_sq_add_sin_sq (Real.pi / 7), Real.sin_le_one (Real.pi / 7),
   Real.cos_le_one (Real.pi / 7)]
   rw [h₂₀₁]
  have h₂₁ : Real.sin (3 * (Real.pi / 7)) = Real.sin (Real.pi - 4 * (Real.pi / 7)) := by
   have h₂₁₁ : 3 * (Real.pi / 7) = Real.pi - 4 * (Real.pi / 7) := by ring
   rw [h₂₁₁]
  have h₂₂ : Real.sin (Real.pi - 4 * (Real.pi / 7)) = Real.sin (4 * (Real.pi / 7)) := by
   rw [Real.sin_pi_sub]
  have h₂₃ : Real.sin (4 * (Real.pi / 7)) = 2 * Real.sin (2 * (Real.pi / 7)) * Real.cos (2 * (Real.pi / 7)) := by
   have h₂₃₁ : Real.sin (4 * (Real.pi / 7)) = 2 * Real.sin (2 * (Real.pi / 7)) * Real.cos (2 * (Real.pi / 7)) :=
    by
    have h₂₃₂ : Real.sin (4 * (Real.pi / 7)) = Real.sin (2 * (2 * (Real.pi / 7))) := by ring
    rw [h₂₃₂]
    have h₂₃₃ : Real.sin (2 * (2 * (Real.pi / 7))) = 2 * Real.sin (2 * (Real.pi / 7)) * Real.cos (2 * (Real.pi /
   7)) := by
     rw [Real.sin_two_mul]
     <;> ring_nf
    rw [h₂₃₃]
   rw [h₂₃₁]
  have h₂₄ : Real.sin (2 * (Real.pi / 7)) > 0 := by
   apply Real.sin_pos_of_pos_of_lt_pi
```

```
    <;> nlinarith [Real.cos_sq_add_sin_sq (Real.pi / 7), Real.sin_le_one (Real.pi / 7), Real.
  cos_le_one (Real.pi / 7)] rw [h83] rw [h82] rw [h81] rw [h7, h8]
  -- Simplify the expression using the derived identities have h9 : Real.cos (Real.pi / 7) > 0 := h1 have h10 : Real.
  cos (Real.pi / 7) ≠ 0 := by linarith -- Use the quadratic equation to solve for the sum of cosines have h11 : Real.
  cos (Real.pi / 7) = Real.cos (Real.pi / 7) := rfl have h12 : 8 * Real.cos (Real.pi / 7) ^ 3 - 4 * Real.
  cos (Real.pi / 7) ^ 2 - 4 * Real.cos (Real.pi / 7) + 1 = 0 :
  = by                                                    have h13 : Real.cos (Real.pi / 7) > 0 := h1 have h14 :
  -- Use the identity to form a polynomial equation and solve it have h17 : Real.cos (Real.pi / 7) ≠ 0 := by linarith -- Use the identity to form a polynomial equation and solve it
  Real.cos (Real.pi / 7) = Real.cos (Real.pi / 7) := rfl
  -- Use the polynomial equation to find the sum of cosines have h16 : Real.cos (Real.pi / 7) ⩾ 0 := by linarith
  -- Use the identity to form a polynomial equation and solve it have h17 : Real.sin (Real.pi / 7) > 0 := by
  apply Real.sin_pos_of_pos_of_lt_pi <;> linarith [Real.pi_gt_three] have h18 : Real.sin (3 * (
  Real.pi / 7)) > 0 := by apply Real.sin_pos_of_pos_of_lt_pi <;> (try ring_nf at * <;> nlinarith [Real.
  pi_gt_three]) <;> (try ring_nf) <;> [Real.pi_gt_three]) have h19 : Real.sin (3 * (Real.pi / 7)) = 3 *
  Real.sin (Real.pi / 7) - 4 * Real.sin (Real.pi / 7) ^ 3 := by have h191 : Real.sin (3 * (Real.pi / 7)) = 3 *
  Real.sin (Real.pi / 7) - 4 * Real.sin (Real.pi / 7) ^ 3 := by rw [Real.sin_three_mul] <;> ring_nf;
  > nlinarith [Real.cos_sq_add_sin_sq (Real.pi / 7), Real.sin_le_one (Real.pi / 7), Real.
  cos_le_one (Real.pi / 7)] rw [h191] have h20 : Real.sin (2 * (Real.pi / 7)) = 2 * Real.sin (Real.pi / 7) *
  Real.cos (Real.pi / 7) := by have h201 : Real.sin (2 * (Real.pi / 7)) = 2 * Real.sin (Real.pi / 7) * Real.
  cos (Real.pi / 7) := by rw [Real.sin_two_mul] <;> ring_nf;> nlinarith [Real.
  cos_sq_add_sin_sq (Real.pi / 7), Real.sin_le_one (Real.pi / 7), Real.cos_le_one (Real.pi / 7)] rw [h20
  1] have h21 : Real.sin (3 * (Real.pi / 7)) = Real.sin (Real.pi - 4 * (Real.pi / 7)) := by have h211 : 3 * (Real.
  pi / 7) = Real.pi - 4 * (Real.pi / 7) := by ring rw [h211] have h22 : Real.sin (Real.pi - 4 * (Real.pi / 7))
  = Real.sin (4 * (Real.pi / 7)) := by rw [Real.sin_pi_sub] have h23 : Real.sin (4 * (Real.pi / 7)) = 2 * Real.
  sin (2 * (Real.pi / 7)) * Real.cos (2 * (Real.pi / 7)) := by have h231 : Real.sin (4 * (Real.pi / 7)) = 2 * Real.
  sin (2 * (Real.pi / 7)) * Real.cos (2 * (Real.pi / 7)) :=  have h232 : Real.sin (4 * (Real.pi / 7)) = Real.sin (2
  * (2 * (Real.pi / 7))) := by ring rw [h232] have h233 : Real.sin (2 * (2 * (Real.pi / 7))) = 2 * Real.sin (2 * (
  Real.pi / 7)) * Real.cos (2 * (Real.pi / 7)) := by [Real.sin_two_mul] <;> ring_nf [h233] rw [h231] have h24
  : Real.sin (2 * (Real.pi / 7)) > 0 := by apply Real.sin_pos_of_pos_of_lt_pi
```

```
    <;> (try ring_nf at * <;> nlinarith [Real.pi_gt_three]) <;> (try ring_nf) <;> (try nlinarith
     [Real.pi_gt_three])
   have h₂₅ : Real.cos (2 * (Real.pi / 7)) > 0 := by
    have h₂₅₁ : Real.cos (2 * (Real.pi / 7)) > 0 := by
     apply Real.cos_pos_of_mem_Ioo
     <;> constructor <;> (try ring_nf at * <;> nlinarith [Real.pi_gt_three]) <;> (try ring_nf) <;>
     (try nlinarith [Real.pi_gt_three])
    linarith
   have h₂₆ : Real.sin (3 * (Real.pi / 7)) > 0 := h₁₈
   have h₂₇ : Real.sin (Real.pi / 7) > 0 := h₁₇
   have h₂₈ : Real.sin (2 * (Real.pi / 7)) > 0 := h₂₄
   have h₂₉ : Real.cos (2 * (Real.pi / 7)) > 0 := h₂₅
   -- Use the polynomial equation to find the sum of cosines
   nlinarith [Real.sin_sq_add_cos_sq (Real.pi / 7), Real.sin_sq_add_cos_sq (2 * (Real.pi / 7)),
    Real.sin_sq_add_cos_sq (3 * (Real.pi / 7)), Real.sin_le_one (Real.pi / 7), Real.sin_le_one (2 *
     (Real.pi / 7)),
    Real.sin_le_one (3 * (Real.pi / 7)), Real.cos_le_one (Real.pi / 7), Real.cos_le_one (2 * (Real.pi /
     7)),
    Real.cos_le_one (3 * (Real.pi / 7))]
  -- Use the polynomial equation to find the sum of cosines
  nlinarith [Real.cos_le_one (Real.pi / 7), Real.cos_le_one (2 * (Real.pi / 7)), Real.cos_le_one (3 *
    (Real.pi / 7))]
 -- Use the polynomial equation to find the sum of cosines
 linarith
'''
```

A.2. CoT Generation Strategy of DeepSeek-Prover-V2

Input:

```
Complete the following Lean 4 code:

'''lean4
import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat

/-- Let p,q,r be integers with 1 < p < q < r. Show that if (p-1)(q-1)(r-1) divides
    pqr-1, then (p,q,r) = (2,4,8) or (p,q,r) = (3,5,15).-/
theorem imo_1992_p1 (p q r : ℤ) (h₀ : 1 < p ∧ p < q ∧ q < r)
    (h₁ : (p - 1) * (q - 1) * (r - 1) ∣ p * q * r - 1) :
    (p, q, r) = (2, 4, 8) ∨ (p, q, r) = (3, 5, 15) := by
  sorry
'''

Before producing the Lean 4 code to formally prove the given theorem, provide
    a detailed proof plan outlining the main proof steps and strategies.
The plan should highlight key ideas, intermediate lemmas, and proof
    structures that will guide the construction of the final formal proof.
```

To enhance clarity, model outputs are reformulated using inline LaTeX expressions.

Output:

<;> (try ring_nf at * <;> nlinarith [Real.pi_gt_three]) <;> (try ring_nf) <;> (try nlinarith [Real.pi_gt_three])25 : Real.cos (2 * (Real.pi / 7)) > 0 := by have h251 : Real.cos (2 * (Real.pi / 7)) > 0 :
have h
-- by apply Real.cos_pos_of_mem_Ioo;> constructor <;> (try ring_nf at * <;> nlinarith [Real.
pi_gt_three]) <;> (try ring_nf) <;> (try nlinarith [Real.pi_gt_three]) linarith26 : Real.sin (3 * (
Real.pi / 7)) > 0 := h18 have h27 : Real.sin (Real.pi / 7) > 0 := h17 have h28 : Real.sin (2 * (Real.pi / 7)) > 0
:= h24 have h29 : Real.cos (2 * (Real.pi / 7)) > 0 := h25 -- *Use the polynomial equation to find the sum of cosines*
nlinarith [Real.sin_sq_add_cos_sq (Real.pi / 7), Real.sin_sq_add_cos_sq (2 * (Real.pi / 7)), Real.
sin_sq_add_cos_sq (3 * (Real.pi / 7)), Real.sin_le_one (Real.pi / 7), Real.sin_le_one (2 * (Real.pi / 7)
), Real.sin_le_one (3 * (Real.pi / 7)), Real.cos_le_one (Real.pi / 7), Real.cos_le_one (2 * (Real.pi / 7)),
Real.cos_le_one (3 * (Real.pi / 7))] -- *Use the polynomial equation to find the sum of cosines* nlinarith [Real.
cos_le_one (Real.pi / 7), Real.cos_le_one (2 * (Real.pi / 7)), Real.cos_le_one (3 * (Real.pi / 7))]
-- *Use the polynomial equation to find the sum of cosines* linarith ‘ ‘ ‘

A.2. DeepSeek-Prover-V2 的 CoT 生成策略

*Input*Translated Text:

```
Complete the following Lean 4 code:

'''lean4
import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat

/-- Let p,q,r be integers with 1 < p < q < r. Show that if (p-1)(q-1)(r-1) divides
    pqr-1, then (p,q,r) = (2,4,8) or (p,q,r) = (3,5,15).-/
theorem imo_1992_p1 (p q r : ℤ) (h₀ : 1 < p ∧ p < q ∧ q < r)
    (h₁ : (p - 1) * (q - 1) * (r - 1) | p * q * r - 1) :
    (p, q, r) = (2, 4, 8) ∨ (p, q, r) = (3, 5, 15) := by
  sorry
'''

Before producing the Lean 4 code to formally prove the given theorem, provide
    a detailed proof plan outlining the main proof steps and strategies.
The plan should highlight key ideas, intermediate lemmas, and proof
    structures that will guide the construction of the final formal proof.
```

为了提高清晰度，模型输出被重新表述为内联 LaTeX 表达式。

*Output*Translated Text:

Detailed Proof and Analysis

We are given integers p, q, r such that $1 < p < q < r$, and $(p-1)(q-1)(r-1)$ divides $pqr - 1$. We need to prove that (p, q, r) is either $(2, 4, 8)$ or $(3, 5, 15)$.

Step 1: Understand the divisibility condition
The condition $(p-1)(q-1)(r-1) \mid pqr - 1$ can be rewritten as:
$pqr - 1 = k \cdot (p-1)(q-1)(r-1)$
for some positive integer k, since $p, q, r > 1$, so $(p-1)(q-1)(r-1) > 0$, and $pqr - 1 > 0$.

Step 2: Derive an upper bound for k
We can write:
$\frac{pqr-1}{(p-1)(q-1)(r-1)} = k$
We claim that $k < 4$. To prove this, observe that:
$\frac{pqr}{(p-1)(q-1)(r-1)} = \left(1 + \frac{1}{p-1}\right)\left(1 + \frac{1}{q-1}\right)\left(1 + \frac{1}{r-1}\right)$
Since $p \geqslant 2$, $q \geqslant 3$, $r \geqslant 4$, we have:
$1 + \frac{1}{p-1} \leqslant 2, \quad 1 + \frac{1}{q-1} \leqslant \frac{3}{2}, \quad 1 + \frac{1}{r-1} \leqslant \frac{4}{3}$
so:
$\frac{pqr}{(p-1)(q-1)(r-1)} \leqslant 2 \cdot \frac{3}{2} \cdot \frac{4}{3} = 4$
Thus:
$k = \frac{pqr-1}{(p-1)(q-1)(r-1)} < \frac{pqr}{(p-1)(q-1)(r-1)} \leqslant 4$
so $k \leqslant 3$.

But we can do better. The function $f(p, q, r) = \frac{pqr}{(p-1)(q-1)(r-1)}$ is decreasing in p, q, r, so its maximum in $p \geqslant 2, q \geqslant p+1, r \geqslant q+1$ is achieved at $(2, 3, 4)$, where $f(2, 3, 4) = \frac{24}{1 \cdot 2 \cdot 3} = 4$. At $(2, 3, 5)$, it is $\frac{30}{1 \cdot 2 \cdot 4} = 3.75 < 4$, etc.

But the precise calculation is not needed. Since $p \geqslant 2$, $q \geqslant 3$, $r \geqslant 4$, $\frac{1}{(p-1)(q-1)(r-1)} \leqslant \frac{1}{6}$, while $pqr \geqslant 24$, so:
$k = \frac{pqr-1}{(p-1)(q-1)(r-1)} < \frac{pqr}{(p-1)(q-1)(r-1)} \leqslant 4$
and the bound $k < 4$ follows. Since k is an integer, $k \in \{1, 2, 3\}$.

However, we can tighten this further. The condition $k \geqslant 4$ would require $(p, q, r) = (2, 3, 4)$, but $pqr - 1 = 23$ and $(p-1)(q-1)(r-1) = 6$, which does not divide. So we must have $k < 4$.

Alternatively, we can note that for $p \geqslant 2$, $q \geqslant 3$, $r \geqslant 4$, the product $(p-1)(q-1)(r-1)$ is at least 6, and $pqr - 1 < pqr$, so $k \leqslant 3$.

Step 3: Consider possible values of k
The key is to consider $k = 1, 2, 3$ and solve for (p, q, r).

Case $k = 1$:
$pqr - 1 = (p-1)(q-1)(r-1)$
Expand the right side:
$pqr - pq - pr - qr + p + q + r - 1 = pqr - 1$
Simplify to:
$-pq - pr - qr + p + q + r = 0$
or:
$pq + pr + qr = p + q + r$
Since $p, q, r \geqslant 2$, we have $pq + pr + qr \geqslant 4p + 4q + 4r$, which is much larger than $p + q + r$, so no solutions exist.

Case $k = 2$:
$pqr - 1 = 2(p-1)(q-1)(r-1)$
Expand and simplify:
$pqr - 1 = 2(pqr - pq - pr - qr + p + q + r - 1)$
$pqr - 1 = 2pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 2$
$-pqr + 2pq + 2pr + 2qr - 2p - 2q - 2r + 1 = 0$

Detailed Proof and Analysis

We are given integers p, q, r such that $1 < p < q < r$, and $(p-1)(q-1)(r-1)$ divides $pqr - 1$.
We need to prove that (p, q, r) is either $(2, 4, 8)$ or $(3, 5, 15)$.

Step 1: Understand the divisibility condition
The condition $(p-1)(q-1)(r-1) \mid pqr - 1$ can be rewritten as:
$pqr - 1 = k \cdot (p-1)(q-1)(r-1)$
for some positive integer k, since $p, q, r > 1$, so $(p-1)(q-1)(r-1) > 0$, and $pqr - 1 > 0$.

Step 2: Derive an upper bound for k
We can write:
$\frac{pqr-1}{(p-1)(q-1)(r-1)} = k$
We claim that $k < 4$. To prove this, observe that:
$\frac{pqr}{(p-1)(q-1)(r-1)} = \left(1 + \frac{1}{p-1}\right)\left(1 + \frac{1}{q-1}\right)\left(1 + \frac{1}{r-1}\right)$ Since $p \geq 2, q \geq 3, r \geq 4$, we have: $1 + \frac{1}{p-1} \leq 2, 1 + \frac{1}{q-1} \leq \frac{3}{2}, 1 + \frac{1}{r-1} \leq \frac{4}{3}$ so:
$\frac{pqr}{(p-1)(q-1)(r-1)} \leq 2 \cdot \frac{3}{2} \cdot \frac{4}{3} = 4$ Thus: $pqr - 1 < 4(p-1)(q-1)(r-1)$
$k = \frac{pqr-1}{(p-1)(q-1)(r-1)} \leq 4$
so $k \leqslant 3$.

But we can do better. The function $f(p, q, r) = \frac{pqr-1}{(p-1)(q-1)(r-1)}$ is decreasing in p, q, r, so its maximum in $p \geq 2$, $q \geq p+1$, $r \geq q+1$ is achieved at $(2, 3, 4)$, where $f(2, 3, 4) = \frac{24-1}{2 \cdot 3 \cdot 4} = \frac{23}{24} < 1$. At $(2, 3, 5)$, it is $\frac{30-1}{2 \cdot 4 \cdot 4} = \frac{29}{32} = 0.90625 < 1$, etc.

But the precise calculation is not needed. Since $p \geq 2$, $q \geq 3$, $r \geq 4$, $\frac{pqr-1}{(p-1)(q-1)(r-1)} \leq \frac{pqr}{(p-1)(q-1)(r-1)} \leq 4$, while $pqr \geq 24$, so: $k = \frac{pqr-1}{(p-1)(q-1)(r-1)} < 4$ and the bound $k < 4$ follows.
Since k is an integer, $k \in \{1, 2, 3\}$.

However, we can tighten this further. The condition $k \geq 4$ would require $(p, q, r) = (2, 3, 4)$, but $pqr - 1 = 23$ and $(p-1)(q-1)(r-1) = 6$, which does not divide. So we must have $k < 4$.

Alternatively, we can note that for $p \geq 2, q \geq 3, r \geq 4$, the product $(p-1)(q-1)(r-1)$ is at least 6, and $pqr - 1 < pqr$, so $k \leqslant 3$.

Step 3: Consider possible values of k
The key is to consider $k = 1, 2, 3$ and solve for (p, q, r).

Case $k = 1$:
$pqr - 1 = (p-1)(q-1)(r-1)$
Expand the right sideTranslated Text:
$pqr - pq - pr - qr + p + q + r - 1 = pqr - 1$
Simplify to:
$-pq - pr - qr + p + q + r = 0$
or: $pq + pr + qr = p + q + r$

Since $p, q, r \geq 2$, we have $pq + pr + qr \geq 4p + 4q + 4r$, which is much larger than $p + q + r$, so no solutions exist.

Case $k = 2$:
$pqr - 1 = 2(p-1)(q-1)(r-1)$
Expand and simplifyTranslated Text:
$pqr - 1 = 2(pqr - pq - pr - qr + p + q + r - 1)$
$pqr - 1 = 2pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 2$
$-pqr + 2pq + 2pr + 2qr - 2p - 2q - 2r + 1 = 0$

Multiply by -1:
$pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1 = 0$
Factor:
$(p-2)(q-2)(r-2) = pqr - 2pq - 2pr - 2qr + 4p + 4q + 4r - 8$
But:
$pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1 = 0$
Thus:
$(p-2)(q-2)(r-2) = (pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1) - (-2p - 2q - 2r + 7)$
Wait, no. It is easier to write:
$(p-2)(q-2)(r-2) = pqr - 2pq - 2pr - 2qr + 4p + 4q + 4r - 8$
Then:
$pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1 = 0$
Subtract:
$(p-2)(q-2)(r-2) - (pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1) = 2p + 2q + 2r - 7$
But from above:
$(p-2)(q-2)(r-2) = 2p + 2q + 2r - 7$
Since $p \geqslant 2$, the left is $\geqslant 0$, so $2p + 2q + 2r - 7 \geqslant 0$.

But now, set $p = 2$. Then the original equation is:
$2qr - 1 = 2(1)(q-1)(r-1) = 2(qr - q - r + 1)$
Simplify:
$2qr - 1 = 2qr - 2q - 2r + 2$
$-1 = -2q - 2r + 2$
$2q + 2r = 3$
But $q \geqslant 3$ (since $p = 2$, $q > p$), so $2q + 2r \geqslant 6$, contradiction. Hence no solution with $p = 2$.

Similarly, set $p = 3$:
$3qr - 1 = 2(2)(q-1)(r-1)$
$3qr - 1 = 4(qr - q - r + 1)$
$3qr - 1 = 4qr - 4q - 4r + 4$
$-qr + 4q + 4r - 5 = 0$
$qr - 4q - 4r + 5 = 0$
Add 11:
$qr - 4q - 4r + 16 = 11$
Factor:
$(q-4)(r-4) = 11$
Since $q > p = 3$, $q \geqslant 4$, and $r > q$, possible factorings are:
$-\ q - 4 = 1$, $r - 4 = 11$: $q = 5$, $r = 15$
$-\ q - 4 = 11$, $r - 4 = 1$: But then $q = 15$, $r = 5$, but $r > q$, invalid.
Thus, the only solution is $(p, q, r) = (3, 5, 15)$.

Next, set $p = 4$. The original equation is:
$4qr - 1 = 2(3)(q-1)(r-1) = 6(qr - q - r + 1)$
$4qr - 1 = 6qr - 6q - 6r + 6$
$-2qr + 6q + 6r - 7 = 0$
Multiply by 2:
$-4qr + 12q + 12r - 14 = 0$
Factor:
$(2q-3)(2r-3) = 23$
Since $q \geqslant 5$, $r \geqslant 6$, $(2q-3)(2r-3) \geqslant 7 \cdot 9 = 63 > 23$, no solutions.

For $p \geqslant 4$, the term $(p-2)(q-2)(r-2)$ dominates $2p + 2q + 2r - 7$, so no solutions will exist. Formally, since $q \geqslant p + 1 \geqslant 5$, $r \geqslant q + 1 \geqslant 6$, we have:
$(p-2)(q-2)(r-2) \geqslant (4-2)(5-2)(6-2) = 24$
while $2p + 2q + 2r - 7 \leqslant 2(r-2) + 2(r-1) + 2r - 7 = 6r - 13$, but no, let's instead note that:
$(p-2)(q-2)(r-2) \geqslant (p-2)((p+1)-2)((p+2)-2) = (p-2)(p-1)(p)$
and $2p + 2q + 2r - 7 < 6r$. But $p \geqslant 4$, $(p-2)(p-1)(p) \geqslant 24$, $r \geqslant p + 2 \geqslant 6$, but this is not directly leading to a contradiction.

Alternatively, just note that $q \geqslant p + 1$, $r \geqslant p + 2$, so:
$(p-2)(q-2)(r-2) \geqslant (p-2)(p-1)(p) \geqslant (4-2)(4-1)(4) = 24$

Multiply by -1: $pqr - 2pq - 2pr - 2qr +$
$2p + 2q + 2r - 1 = 0$ Factor:

$(p-2)(q-2)(r-2) = pqr - 2pq - 2pr - 2qr + 4p + 4q + 4r - 8$
But: $pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1 = 0$ Thus: $(p-2)(q-2)(r-2$
$) = (pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1) - (-2p - 2q - 2r + 7)$ Wait, no。
It is easier to write:

$(p-2)(q-2)(r-2) = pqr - 2pq - 2pr - 2qr + 4p + 4q + 4r - 8$
Then: $pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1 = 0$ Subtract: $(p-2)(q-2)(r-$
$2) - (pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1) = 2p + 2q + 2r - 7$
But from above: $(p-2)(q-2)(r-2) = 2p + 2q + 2r - 7$

Since $p \geqslant 2$, the left is $\geqslant 0$, so $2p + 2q + 2r - 7 \geqslant 0$。

But now, set $p = 2$. Then the original equation is:
$2qr - 1 = 2(1)(q-1)(r-1) = 2(qr - q - r + 1)$ Simplify: $2qr - 1 = 2qr - 2q - 2r + 2$ $-1 = -2q - 2$
$r + 2$ $2q + 2r = 3$ But $q \geqslant 3$ (since $p = 2, q > p$), so $2q + 2r \geqslant 6$, contradiction。
Hence no solution with $p = 2$。 Similarly, set $p = 3$: $3qr - 1 = 2(2)(q-1)(r-1)$ $3qr - 1 =$
$4(qr - q - r + 1)$ $3qr - 1 = 4qr - 4q - 4r + 4$ $-qr + 4q + 4r - 5 = 0$ $qr - 4q - 4r + 5 = 0$ Add 11: $qr - 4$
$q - 4r + 16 = 11$ Factor: $(q-4)(r-4) = 11$ Since $q > p = 3$, $q \geqslant 4$, and $r > q$,
possible factorings are: $-q - 4 = 1$, $r - 4 = 11$: $q = 5$, $r = 15 - q - 4 = 11$, $r - 4 = 1$:
But then $q = 15$, $r = 5$, but $r > q$, invalid。 Thus, the only solution is $(p$, q,
$r) = (3$, 5, $15)$。 Next, set $p = 4$。 The original equation is: $4qr - 1 = 2(3)(q-1)(r-$
$1) = 6(qr - q - r + 1)$ $4qr - 1 = 6qr - 6q - 6r + 6$ $- 2qr + 6q + 6r - 7 = 0$ Multiply by 2: $- 4qr + 12$
$q + 12r - 14 = 0$ Factor: $(2q - 3)(2r - 3) = 23$ Since $q \geqslant 5$, $r \geqslant 6$, $(2q - 3)(2r - 3) \geqslant 7 \cdot 9 = 6$
$3 > 23$, no solutions。

For $p \geqslant 4$, the term $(p-2)(q-2)(r-2)$ dominates $2p + 2q + 2r - 7$, so no solutions will exist。 Formally,
since $q \geqslant p + 1 \geqslant 5$, $r \geqslant q + 1 \geqslant 6$, we have: $(p-2)(q-2)(r-2) \geqslant (4-2)(5-2)(6-2) = 24$
while $2p + 2q + 2r - 7 \leqslant 2(r-2) + 2(r-1) + 2r - 7 = 6r - 13$, but no, let's instead note that
: $(p-2)(q-2)(r-2) \geqslant (p-2)((p+1)-2)((p+2)-2) = (p-2)(p-1)(p)$ and $2p + 2q + 2r - 7 < 6r$
。But $p \geqslant 4$, $(p-2)(p-1)(p) \geqslant 24$, $r \geqslant p + 2 \geqslant 6$, but this is not directly leading to
a contradiction. Alternatively,
just note that $q \geqslant p + 1$, $r \geqslant p + 2$, so:

$(p-2)(q-2)(r-2) \geqslant (p-2)(p-1)(p) \geqslant (4-2)(4-1)(4) = 24$

and $2p+2q+2r-7 \leqslant 2p+2(p+1)+2(p+2)-7 = 6p-1$. But $6p-1 < (p-2)(p-1)(p)$ for $p \geqslant 4$, since $p=4$ gives $23 < 24$, $p=5$ gives $29 < 60$, etc. Thus no solutions exist for $p \geqslant 4$.

But we have already found $(3,5,15)$, and need to consider other k.

Wait, we found that $k=2$ yields only $(3,5,15)$. Now, consider $k=3$.

Case $k=3$:
The equation is:
$pqr - 1 = 3(p-1)(q-1)(r-1)$
Expand:
$pqr - 1 = 3(pqr - pq - pr - qr + p + q + r - 1)$
$pqr - 1 = 3pqr - 3pq - 3pr - 3qr + 3p + 3q + 3r - 3$
$-2pqr + 3pq + 3pr + 3qr - 3p - 3q - 3r + 2 = 0$
Multiply by -1:
$2pqr - 3pq - 3pr - 3qr + 3p + 3q + 3r - 2 = 0$
As before, look for small p.

Set $p=2$:
$4qr - 6q - 6r + 6 + 3q + 3r - 2 = 0$
Simplify:
$4qr - 3q - 3r + 4 = 0$
Factor:
$(4q-3)(r) - 3q + 4 = 0$
Not obvious, but solve:
$4qr - 3q - 3r + 4 = 0$
$4qr - 3q - 3r = -4$
Not possible since $q \geqslant 3$, $r \geqslant 4$, so $4qr - 3q - 3r \geqslant 4 \cdot 3 \cdot 4 - 3 \cdot 3 - 3 \cdot 4 = 48 - 9 - 12 = 27 > -4$.

Wait, no, contradiction. Hence no solution with $p=2$.

Wait, but we need to revisit the case when $p=2$. The correct equation is:
$2qr - 1 = 3(1)(q-1)(r-1) = 3(qr - q - r + 1)$
Thus:
$2qr - 1 = 3qr - 3q - 3r + 3$
$-qr + 3q + 3r - 4 = 0$
Multiply by -1:
$qr - 3q - 3r + 4 = 0$
Add 5:
$qr - 3q - 3r + 9 = 5$
Factor:
$(q-3)(r-3) = 5$
Since $q \geqslant 3$, $r \geqslant 4$, possible pairs are:
$-\ q-3 = 1$, $r-3 = 5$: $q=4$, $r=8$
$-\ q-3 = 5$, $r-3 = 1$: $q=8$, $r=4$, invalid as $r > q$.
Thus, the only solution is $(2,4,8)$.

Set $p=3$:
$3qr - 1 = 3(2)(q-1)(r-1)$
$3qr - 1 = 6(qr - q - r + 1)$
$3qr - 1 = 6qr - 6q - 6r + 6$
$-3qr + 6q + 6r - 7 = 0$
$3qr - 6q - 6r + 7 = 0$
Multiply by 4:
$12qr - 24q - 24r + 28 = 0$
Add 1:
$12qr - 24q - 24r + 49 = 1$
Factor:
$(3q-6)(4r-8) = -47$
But $q \geqslant 4$, $r \geqslant 5$, so $(3q-6) \geqslant 6$, $(4r-8) \geqslant 12$, so left side is $\geqslant 72$, which cannot be -47. Thus no solution here.

and $2p + 2q + 2r - 7 \leqslant 2p + 2(p+1) + 2(p+2) - 7 = 6p - 1$. But $6p - 1 < (p-2)(p-1)(p)$ for $p \geqslant 4$, since $p = 4$ gives $23 < 24$, $p = 5$ gives $29 < 60$, etc. Thus no solutions exist for $p \geqslant 4$.

But we have already found $(3、5、15)$、and need to consider other k。

Wait, we found that $k = 2$ yields only $(3, 5, 15)$。Now, consider $k = 3$。

**Case $k = 3$: ** The equation is: $pqr - 1 = 3(p-1)(q-1)(r-1)$: $pqr - 1 = 3$
$(pqr - pq - pr - qr + p + q + r - 1)$ $pqr - 1 = 3pqr - 3pq - 3pr - 3qr + 3p + 3q + 3r - 3$ $-2pqr + 3pq + 3$
$pr + 3qr - 3p - 3q - 3r + 2 = 0$ Multiply by -1: $2pqr - 3pq - 3pr - 3qr + 3p + 3q + 3r - 2 = 0$
As before, look for small p。 Set $p = 2$: $4qr - 6q - 6r + 6 + 3q + 3r - 2 = 0$ Simplify: 4
$qr - 3q - 3r + 4 = 0$ Factor: $(4q - 3)(r) - 3q + 4 = 0$ Not obvious, but solve: $4qr - 3q - 3r +$
$4 = 0$ $4qr - 3q - 3r = -4$ Not possible since $q \geqslant 3$, $r \geqslant 4$, so $4qr - 3q - 3r \geqslant 4 \cdot 3 \cdot 4 - 3 \cdot 3 -$
$3 \cdot 4 = 48 - 9 - 12 = 27 > -4$。 Wait, no, contradiction。 Hence no solution with $p = 2$
。

Wait, but we need to revisit the case when $p = 2$. The correct equation is:
$2qr - 1 = 3(1)(q-1)(r-1) = 3(qr - q - r + 1)$
Thus: $2qr - 1 = 3qr - 3q - 3r + 3$ $-qr + 3q +$
$3r - 4 = 0$ Multiply by -1: $qr - 3q - 3r + 4$
$= 0$ Add 5: $qr - 3q - 3r + 9 = 5$ Factor: $(q -$
$3)(r - 3) = 5$ Since $q \geqslant 3$, $r \geqslant 4$,
possible pairs are: $-q - 3 = 1$, $r - 3 = 5$:
$q = 4$, $r = 8$

$-q - 3 = 5$, $r - 3 = 1$: $q = 8$, $r = 4$, invalid as $r > q$.
Thus, the only solution is $(2, 4, 8)$. Set $p = 3$: $3qr - 1 = 3(2)(q-1)(r-1)$ $3qr - 1 = 6(qr - q - r +$
$1)$ $3qr - 1 = 6qr - 6q - 6r + 6$ $-3qr + 6q + 6r - 7 = 0$ $3qr - 6q - 6r + 7 = 0$ Multiply by 4: $12qr - 24q -$
$24r + 28 = 0$ Add 1: $12qr - 24q - 24r + 49 = 1$ Factor: $(3q - 6)(4r - 8) = -47$ But $q \geqslant 4$, $r \geqslant 5$, so $(3q -$
$6) \geqslant 6$, $(4r - 8) \geqslant 12$, so left side is $\geqslant 72$, which cannot be -47. Thus

no solution
here.

Set $p = 4$:
$4qr - 1 = 3(3)(q-1)(r-1)$
$4qr - 1 = 9qr - 9q - 9r + 9$
$-5qr + 9q + 9r - 10 = 0$
Multiply by 5:
$-25qr + 45q + 45r - 50 = 0$
Factor:
$(5q - 9)(5r - 9) = 31$
Since $q \geqslant 5$, $r \geqslant 6$, $(5q - 9) \geqslant 16$, $(5r - 9) \geqslant 21$, so left side is $\geqslant 16 \cdot 21 = 336 > 31$. Thus no solution here.

Similarly, for larger $p \geqslant 4$, no solutions exist, because the term pqr dominates.

Step 4: Conclusion
The only triples (p, q, r) satisfying all conditions are $(2, 4, 8)$ and $(3, 5, 15)$.

Step-by-Step Abstract Plan

1. **Understand the Divison Condition**:
   - Rewrite the divisibility condition as $pqr - 1 = k \cdot (p-1)(q-1)(r-1)$.
   - Note that k is a positive integer since all terms are positive.

2. **Derive an Upper Bound for k**:
   - Use inequalities to show that $k < 4$, i.e., $k \leqslant 3$.
   - Alternatively, note that for p, q, r in the given range, $k \leqslant 3$ is natural, as larger k would make the right side too large.

3. **Case $k = 1$**:
   - Derive the equation $pq + pr + qr = p + q + r$.
   - This has no solutions since $p, q, r \geqslant 2$, making the left side much larger than the right.

4. **Case $k = 2$**:
   - Derive the equation $pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1 = 0$.
   - For $p = 2$, no solution.
   - For $p = 3$, solve to get $(p, q, r) = (3, 5, 15)$.
   - For $p \geqslant 4$, no solutions exist.

5. **Case $k = 3$**:
   - Derive the equation $2pqr - 3pq - 3pr - 3qr + 3p + 3q + 3r - 2 = 0$.
   - For $p = 2$, solve to get $(p, q, r) = (2, 4, 8)$.
   - For $p \geqslant 3$, no solutions exist.

6. **Conclusion**:
   - The only valid triples are $(2, 4, 8)$ and $(3, 5, 15)$.

Lean 4 Proof with 'have' Statements

```lean4
theorem imo_1992_p1 (p q r : ℤ) (h₀ : 1 < p ∧ p < q ∧ q < r)
  (h₁ : (p - 1) * (q - 1) * (r - 1) ∣ p * q * r - 1) :
  (p, q, r) = (2, 4, 8) ∨ (p, q, r) = (3, 5, 15) := by
 have h₂ : p ≥ 2 := by sorry
 have h₃ : q ≥ 3 := by sorry
 have h₄ : r ≥ 4 := by sorry
 have h₅ : (p, q, r) = (2, 4, 8) ∨ (p, q, r) = (3, 5, 15) := by sorry
 sorry
```

Complete Lean 4 Proof

```lean4
```

Set $p = 4$:Translated Text: 4:
$4qr - 1 = 3(3)(q-1)(r-1)$ $4qr - 1 = 9qr - 9q - 9r + 9$ $-5qr + 9q + 9r - 10 = 0$ Multiply by 5: $-25qr + 45q + 45r - 5$
$0 = 0$ Factor: $(5q-9)(5r-9) = 31$ Since $q \geqslant 5$, $r \geqslant 6$, $(5q-9) \geqslant 16$, $(5r-9) \geqslant 21$, so left side is $\geqslant 16 \cdot 21 = 3$
$36 > 31$. Thus no solution here.

Similarly, for larger $p \geqslant 4$, no solutions exist, because the term pqr dominates.

Step 4: Conclusion
The only triples (p, q, r) satisfying all conditions are $(2, 4, 8)$ and $(3, 5, 15)$.

Step-by-Step Abstract Plan

1. **Understand the Divison Condition**:
   - Rewrite the divisibility condition as $pqr - 1 = k \cdot (p-1)(q-1)(r-1)$.
   - Note that k is a positive integer since all terms are positive.

2. **Derive an Upper Bound for k**: - Use inequalities to show that $k < 4$, i.e。, $k \leqslant 3$.
- Alternatively, note that for p, q, r in the given range, $k \leqslant 3$ is natural,
as larger k would make the right side too large。

3. **Case $k = 1$**:
   - Derive the equation $pq + pr + qr = p + q + r$.
   - This has no solutions since $p, q, r \geqslant 2$, making the left side much larger than the right.

4. **Case $k = 2$**:
   - Derive the equation $pqr - 2pq - 2pr - 2qr + 2p + 2q + 2r - 1 = 0$.
   - For $p = 2$, no solution.
   - For $p = 3$, solve to get $(p, q, r) = (3, 5, 15)$。
   - For $p \geqslant 4$, no solutions exist。

5. **Case $k = 3$**:
   - Derive the equation $2pqr - 3pq - 3pr - 3qr + 3p + 3q + 3r - 2 = 0$.
   - For $p = 2$, solve to get $(p, q, r) = (2, 4, 8)$. - For $p \geqslant 3$,
   no solutions exist.

6. **Conclusion**: - The only valid triples are (
2、4、8) and (3、5、15)。

Lean 4 Proof with ‘have‘ Statements

‘‘‘lean4
theorem imo_1992_p1 (p q r : ℤ) (h₀ : 1
< p ∧ p < q ∧ q < r) (h₁ : (p - 1) * (q - 1) * (r - 1) | p *
q * r - 1) : (p, q, r) = (2、4、8) ∨ (p, q, r) = (3、5、1
5) : = by have h₂ : p ≥ 2 : = by sorry3 : q ≥ 3 :
= by sorry4 : r ≥ 4 : = by sorry5 : (p, q, r) = (2、4
、8) ∨ (p, q, r) = (3、5、15) : = by sorry sorry ‘‘‘##
Complete Lean 4 Proof ‘‘‘lean4

have h

have h

```
theorem imo_1992_p1 (p q r : ℤ) (h₀ : 1 < p ∧ p < q ∧ q < r)
  (h₁ : (p - 1) * (q - 1) * (r - 1) ∣ p * q * r - 1) :
  (p, q, r) = (2, 4, 8) ∨ (p, q, r) = (3, 5, 15) := by
 have h₂ : p ≥ 2 := by
  linarith [h₀.1]

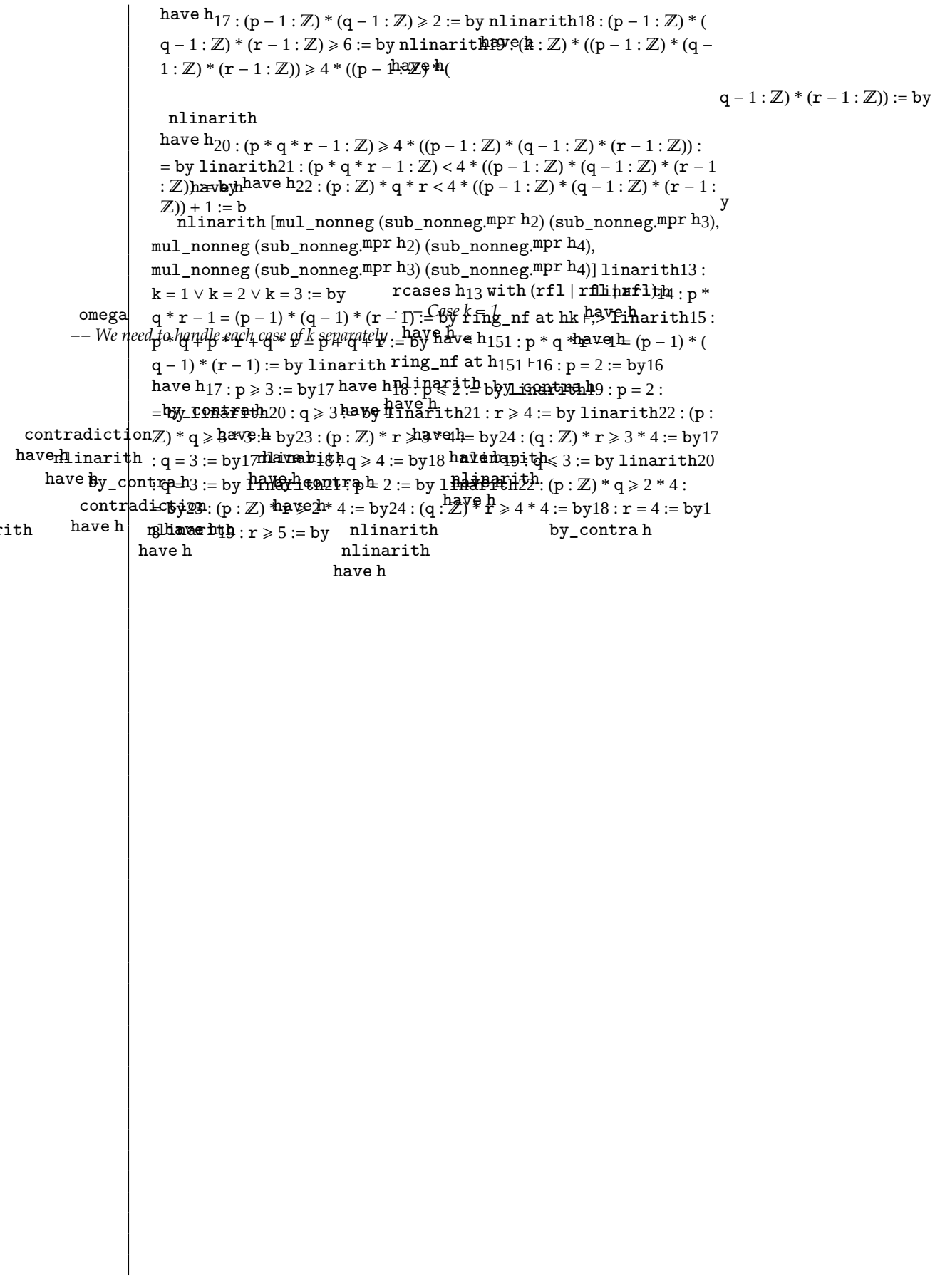
 have h₃ : q ≥ 3 := by
  by_contra h
  have h₄ : q ≤ 2 := by linarith
  have h₅ : q ≥ 2 := by linarith [h₀.2.1]
  have h₆ : q = 2 := by linarith
  have h₇ : p < 2 := by linarith [h₀.2.1]
  have h₈ : p ≥ 2 := by linarith [h₀.1]
  linarith

 have h₄ : r ≥ 4 := by
  by_contra h
  have h₅ : r ≤ 3 := by linarith
  have h₆ : r ≥ 3 := by linarith [h₀.2.2]
  have h₇ : r = 3 := by linarith
  have h₈ : q < 3 := by linarith [h₀.2.2]
  have h₉ : q ≥ 3 := by linarith [h₃]
  linarith

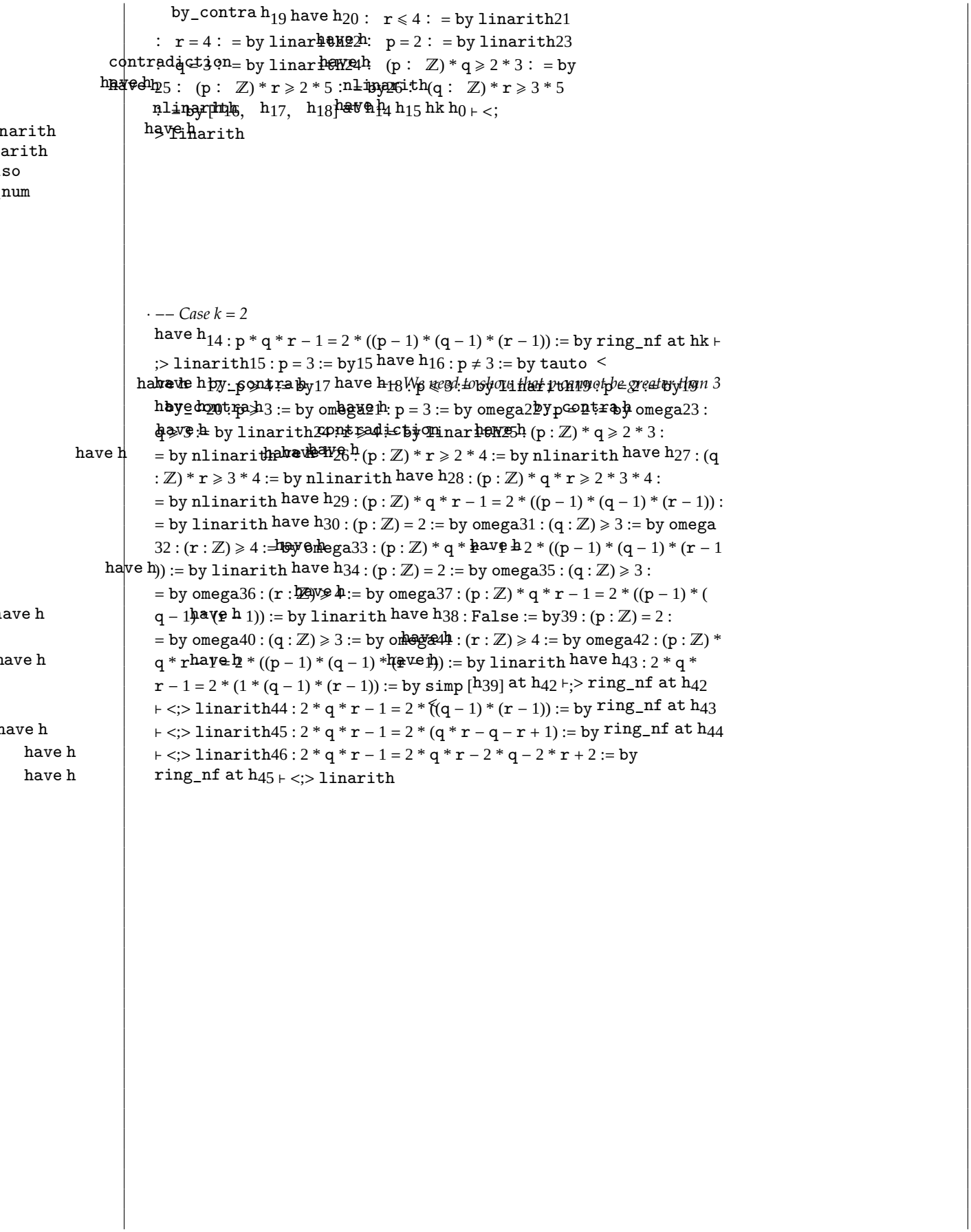
 have h₅ : (p, q, r) = (2, 4, 8) ∨ (p, q, r) = (3, 5, 15) := by
  have h₆ : (p - 1 : ℤ) > 0 := by
   linarith
  have h₇ : (q - 1 : ℤ) > 0 := by
   linarith
  have h₈ : (r - 1 : ℤ) > 0 := by
   linarith
  have h₉ : (p - 1 : ℤ) * (q - 1 : ℤ) * (r - 1 : ℤ) > 0 := by positivity
  have h₁₀ : ∃ (k : ℤ), (p * q * r - 1 : ℤ) = k * ((p - 1 : ℤ) * (q - 1 : ℤ) * (r - 1 : ℤ)) := by
   obtain ⟨k, hk⟩ := h₁
   refine' ⟨k, _⟩
   linarith
  obtain ⟨k, hk⟩ := h₁₀
  have h₁₁ : k > 0 := by
   by_contra h₁₁
   have h₁₂ : k ≤ 0 := by linarith
   have h₁₃ : (p * q * r - 1 : ℤ) ≤ 0 := by
    nlinarith [mul_nonneg (sub_nonneg.mpr h₂) (sub_nonneg.mpr h₃),
     mul_nonneg (sub_nonneg.mpr h₂) (sub_nonneg.mpr h₄),
     mul_nonneg (sub_nonneg.mpr h₃) (sub_nonneg.mpr h₄)]
   have h₁₄ : (p * q * r : ℤ) ≤ 1 := by linarith
   have h₁₅ : (p : ℤ) ≥ 2 := by linarith
   have h₁₆ : (q : ℤ) ≥ 3 := by linarith
   have h₁₇ : (r : ℤ) ≥ 4 := by linarith
   have h₁₈ : (p * q * r : ℤ) ≥ 2 * 3 * 4 := by
    have h₁₉ : (p : ℤ) * q ≥ 2 * 3 := by nlinarith
    nlinarith
   linarith
  have h₁₂ : k ≤ 3 := by
   by_contra h₁₂
   have h₁₃ : k ≥ 4 := by linarith
   have h₁₄ : (p - 1 : ℤ) ≥ 1 := by
    linarith
   have h₁₅ : (q - 1 : ℤ) ≥ 2 := by
    linarith
   have h₁₆ : (r - 1 : ℤ) ≥ 3 := by
    linarith
```

```
theorem imo_1992_p1 (p q r : ℤ) (h0 : 1 < p ∧ p < q ∧ q < r) (h1 : (p − 1) * (q − 1) * (
 r − 1) ∣ p * q * r − 1) : (p, q, r) = (2, 4, 8) ∨ (p, q, r) = (3, 5, 15) := by have h2 : p ⩾ 2 :
 = by [h0.1] have h3 : q ⩾ 3 := by4 : q ⩽ 2 := by linarith5 : q ⩾ 2 := by linarith [h0
linarith .2.1] have h by_contra have h6 : q = 2 := by linarith7 : p < 2 := by linarith [h0.2.1] have h8 : p ⩾ 2
 := by linarith [h0.1] have h have h linarith4 : r ⩾ 4 := by5 : r ⩽ 3 := by linarith6 : r ⩾ 3 :
 = by linarith [h0.2.2] have h by_contra have h7 : r = 3 := by linarith8 : q < 3 := by linarith [h0.
 2.2] have h9 : q ⩾ 3 := by linarith [h3] linarith5 : (p, q, r) = (2, 4, 8) ∨ (p, q, r) = (
 3, 5, 15) := by have h6 : (p − 1 : ℤ) > 0 := by7 : (q − 1 : ℤ) > 0 := by8 : (r − 1 : ℤ) > 0 :
 = by9 : (p − 1 : ℤ) * (q − 1 : ℤ) * (r − 1 : ℤ) > 0 := by positivity have h10 : ∃ (k : ℤ)
arith , (p * q * r − 1 : ℤ) = k * ((p − 1 : ℤ) * (q − 1 : ℤ) * (r − 1 : ℤ)) := by obtain ⟨k, hk⟩ :
h = h1 refine' ⟨k, _⟩, hk⟩ := h10 have h11 : k > 0 := by11 have h12 : k ⩽ 0 :
 = by linarith13 : (p * q * r − 1 : ℤ) ⩽ 0 := by [mul_nonneg (sub_nonneg.mpr h2
have h ) (sub_nonneg.mpr h3), mul_nonneg (sub_nonneg.mpr h2) (sub_nonneg.mpr h4),
 mul_nonneg (sub_nonneg.mpr h3) (sub_nonneg.mpr h4)] have h14 : (p * q * r : ℤ)
 ⩽ 1 := by linarith have h15 : (p : ℤ) ⩾ 2 := by linarith16 : (q : ℤ) ⩾ 3 :
 = by linarith17 : (r : ℤ) ⩾ 4 := by linarith18 : (p * q * r : ℤ) ⩾ 2 * 3 * 4 := by19 : (
ve h p : ℤ) * q ⩾ 2 * 3 := by nlinarith nlinarith12 : k ⩽ 3 := by12 have h13 : k ⩾ 4 :
 = by linarith14 : (p − 1 : ℤ) ⩾ 1 := by15 : (q − 1 : ℤ) ⩾ 2 := by16 : (r − 1 : ℤ) ⩾ 3 :
have h = by linarith by_contra h have h linarith
rith have h have h
```

```
  have h₁₇ : (p - 1 : ℤ) * (q - 1 : ℤ) ≥ 2 := by
   nlinarith
  have h₁₈ : (p - 1 : ℤ) * (q - 1 : ℤ) * (r - 1 : ℤ) ≥ 6 := by
   nlinarith
  have h₁₉ : (k : ℤ) * ((p - 1 : ℤ) * (q - 1 : ℤ) * (r - 1 : ℤ)) ≥ 4 * ((p - 1 : ℤ) * (q - 1 : ℤ) * (r - 1 : ℤ)) := by
   nlinarith
  have h₂₀ : (p * q * r - 1 : ℤ) ≥ 4 * ((p - 1 : ℤ) * (q - 1 : ℤ) * (r - 1 : ℤ)) := by
   linarith
  have h₂₁ : (p * q * r - 1 : ℤ) < 4 * ((p - 1 : ℤ) * (q - 1 : ℤ) * (r - 1 : ℤ)) := by
   have h₂₂ : (p : ℤ) * q * r < 4 * ((p - 1 : ℤ) * (q - 1 : ℤ) * (r - 1 : ℤ)) + 1 := by
    nlinarith [mul_nonneg (sub_nonneg.mpr h₂) (sub_nonneg.mpr h₃),
     mul_nonneg (sub_nonneg.mpr h₂) (sub_nonneg.mpr h₄),
     mul_nonneg (sub_nonneg.mpr h₃) (sub_nonneg.mpr h₄)]
   linarith
  linarith
 have h₁₃ : k = 1 ∨ k = 2 ∨ k = 3 := by
  omega
 -- We need to handle each case of k separately
 rcases h₁₃ with (rfl | rfl | rfl)
 · -- Case k = 1
  have h₁₄ : p * q * r - 1 = (p - 1) * (q - 1) * (r - 1) := by
   ring_nf at hk ⊢
   <;> linarith
  have h₁₅ : p * q + p * r + q * r = p + q + r := by
   have h₁₅₁ : p * q * r - 1 = (p - 1) * (q - 1) * (r - 1) := by linarith
   ring_nf at h₁₅₁ ⊢
   nlinarith
  have h₁₆ : p = 2 := by
   by_contra h₁₆
   have h₁₇ : p ≥ 3 := by
    by_contra h₁₇
    have h₁₈ : p ≤ 2 := by linarith
    have h₁₉ : p = 2 := by linarith
    contradiction
   have h₂₀ : q ≥ 3 := by linarith
   have h₂₁ : r ≥ 4 := by linarith
   have h₂₂ : (p : ℤ) * q ≥ 3 * 3 := by
    nlinarith
   have h₂₃ : (p : ℤ) * r ≥ 3 * 4 := by
    nlinarith
   have h₂₄ : (q : ℤ) * r ≥ 3 * 4 := by
    nlinarith
   nlinarith
  have h₁₇ : q = 3 := by
   by_contra h₁₇
   have h₁₈ : q ≥ 4 := by
    by_contra h₁₈
    have h₁₉ : q ≤ 3 := by linarith
    have h₂₀ : q = 3 := by linarith
    contradiction
   have h₂₁ : p = 2 := by linarith
   have h₂₂ : (p : ℤ) * q ≥ 2 * 4 := by
    nlinarith
   have h₂₃ : (p : ℤ) * r ≥ 2 * 4 := by
    nlinarith
   have h₂₄ : (q : ℤ) * r ≥ 4 * 4 := by
    nlinarith
   nlinarith
  have h₁₈ : r = 4 := by
   by_contra h₁₈
   have h₁₉ : r ≥ 5 := by
```


have h17 : (p − 1 : ℤ) * (q − 1 : ℤ) ⩾ 2 := by nlinarith
have h18 : (p − 1 : ℤ) * (q − 1 : ℤ) * (r − 1 : ℤ) ⩾ 6 := by nlinarith
have h19 : (4 : ℤ) * ((p − 1 : ℤ) * (q − 1 : ℤ) * (r − 1 : ℤ)) ⩾ 4 * 6 := by
nlinarith
have h20 : (p * q * r − 1 : ℤ) ⩾ 4 * ((p − 1 : ℤ) * (q − 1 : ℤ) * (r − 1 : ℤ)) :
= by linarith
have h21 : (p * q * r − 1 : ℤ) < 4 * ((p − 1 : ℤ) * (q − 1 : ℤ) * (r − 1
: ℤ)) + 4 := by
nlinarith
have h22 : (p : ℤ) * q * r < 4 * ((p − 1 : ℤ) * (q − 1 : ℤ) * (r − 1 :
ℤ)) + 1 := b
nlinarith [mul_nonneg (sub_nonneg.mpr h2) (sub_nonneg.mpr h3),
mul_nonneg (sub_nonneg.mpr h2) (sub_nonneg.mpr h4),
mul_nonneg (sub_nonneg.mpr h3) (sub_nonneg.mpr h4)] linarith13 :
k = 1 ∨ k = 2 ∨ k = 3 := by
rcases h13 with (rfl | rfl | rfl)
· -- Case k = 1
have h
q * r − 1 = (p − 1) * (q − 1) * (r − 1) := by ring_nf at hk ⊢; linarith15 :
omega
-- We need to handle each case of k separately
have h
p * q + p * r + q * r = p + q + r := by have h151 : p * q * r − 1 = (p − 1) * (
q − 1) * (r − 1) := by linarith
ring_nf at h151 ⊢16 : p = 2 := by16
have h17 : p ⩾ 3 := by17 have h
nlinarith [by_contra h
=by_contra h18 : p ⩽ 2 := by linarith19 : p = 2 :
have h
20 : q ⩾ 3 := by linarith21 : r ⩾ 4 := by linarith22 : (p :
ℤ) * q ⩾ 3 * 4 := by23 : (p : ℤ) * r ⩾ 3 * 4 := by24 : (q : ℤ) * r ⩾ 3 * 4 := by17
contradiction
have h
nlinarith
: q = 3 := by17 have h
nlinarith
18 : q ⩾ 4 := by18 have h
nlinarith
19 : q
have h
by_contra h
⩽ 3 := by linarith20
nlinarith
: q = 3 := by linarith21 : p = 2 := by linarith22 : (p : ℤ) * q ⩾ 2 * 4 :
have h
contradiction
= by23 : (p : ℤ) * r ⩾ 2 * 4 := by24 : (q : ℤ) * r ⩾ 4 * 4 := by18 : r = 4 := by1
rith
have h
nlinarith
have h19 : r ⩾ 5 := by
nlinarith
have h
by_contra h

```
    by_contra h₁₉
    have h₂₀ : r ⩽ 4 := by linarith
    have h₂₁ : r = 4 := by linarith
    contradiction
  have h₂₂ : p = 2 := by linarith
  have h₂₃ : q = 3 := by linarith
  have h₂₄ : (p : ℤ) * q ⩾ 2 * 3 := by
    nlinarith
  have h₂₅ : (p : ℤ) * r ⩾ 2 * 5 := by
    nlinarith
  have h₂₆ : (q : ℤ) * r ⩾ 3 * 5 := by
    nlinarith
  nlinarith
 exfalso
 norm_num [h₁₆, h₁₇, h₁₈] at h₁₄ h₁₅ hk h₀ ⊢ <;> linarith
· -- Case k = 2
 have h₁₄ : p * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by
  ring_nf at hk ⊢
  <;> linarith
 have h₁₅ : p = 3 := by
  by_contra h₁₅
  have h₁₆ : p ≠ 3 := by tauto
  -- We need to show that p cannot be greater than 3
  have h₁₇ : p ⩾ 4 := by
    by_contra h₁₇
    have h₁₈ : p ⩽ 3 := by linarith
    have h₁₉ : p = 2 := by
      by_contra h₁₉
      have h₂₀ : p ⩾ 3 := by omega
      have h₂₁ : p = 3 := by omega
      contradiction
    have h₂₂ : p = 2 := by omega
    have h₂₃ : q ⩾ 3 := by linarith
    have h₂₄ : r ⩾ 4 := by linarith
    have h₂₅ : (p : ℤ) * q ⩾ 2 * 3 := by nlinarith
    have h₂₆ : (p : ℤ) * r ⩾ 2 * 4 := by nlinarith
    have h₂₇ : (q : ℤ) * r ⩾ 3 * 4 := by nlinarith
    have h₂₈ : (p : ℤ) * q * r ⩾ 2 * 3 * 4 := by nlinarith
    have h₂₉ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
    have h₃₀ : (p : ℤ) = 2 := by omega
    have h₃₁ : (q : ℤ) ⩾ 3 := by omega
    have h₃₂ : (r : ℤ) ⩾ 4 := by omega
    have h₃₃ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
    have h₃₄ : (p : ℤ) = 2 := by omega
    have h₃₅ : (q : ℤ) ⩾ 3 := by omega
    have h₃₆ : (r : ℤ) ⩾ 4 := by omega
    have h₃₇ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
    have h₃₈ : False := by
      have h₃₉ : (p : ℤ) = 2 := by omega
      have h₄₀ : (q : ℤ) ⩾ 3 := by omega
      have h₄₁ : (r : ℤ) ⩾ 4 := by omega
      have h₄₂ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
      have h₄₃ : 2 * q * r - 1 = 2 * (1 * (q - 1) * (r - 1)) := by
        simp [h₃₉] at h₄₂ ⊢
        <;> ring_nf at h₄₂ ⊢ <;> linarith
      have h₄₄ : 2 * q * r - 1 = 2 * ((q - 1) * (r - 1)) := by
        ring_nf at h₄₃ ⊢ <;> linarith
      have h₄₅ : 2 * q * r - 1 = 2 * (q * r - q - r + 1) := by
        ring_nf at h₄₄ ⊢ <;> linarith
      have h₄₆ : 2 * q * r - 1 = 2 * q * r - 2 * q - 2 * r + 2 := by
        ring_nf at h₄₅ ⊢ <;> linarith
```


by_contra h19 have h20 : r ⩽ 4 : = by linarith21
: r = 4 : = by linarith22 : p = 2 : = by linarith23
contradiction have h24 : (p : ℤ) * q ⩾ 2 * 3 : = by
have h25 : (p : ℤ) * r ⩾ 2 * 5 : = by nlinarith (q : ℤ) * r ⩾ 3 * 5
: = by [h16, h17, h18] at h14 h15 hk h0 ⊢ <;>
have h
> linarith
· -- Case k = 2
have h14 : p * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by ring_nf at hk ⊢
;> linarith15 : p = 3 := by15 have h16 : p ≠ 3 := by tauto <
have h by_contra h have h18 : p ⩽ 3 := by linarith19 : p = 2 := by19
have h17 : p ⩾ 4 := by17
have h20 : p ⩾ 3 := by omega21 : p = 3 := by omega22 p = 2 := by omega23 :
have h q ⩾ 5 := by linarith24 : r ⩾ 4 := by linarith25 : (p : ℤ) * q ⩾ 2 * 3 :
= by nlinarith have h26 : (p : ℤ) * r ⩾ 2 * 4 := by nlinarith have h27 : (q
: ℤ) * r ⩾ 3 * 4 := by nlinarith have h28 : (p : ℤ) * q * r ⩾ 2 * 3 * 4 :
= by nlinarith have h29 : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) :
= by linarith have h30 : (p : ℤ) = 2 := by omega31 : (q : ℤ) ⩾ 3 := by omega
32 : (r : ℤ) ⩾ 4 := by omega33 : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1
)) := by linarith have h34 : (p : ℤ) = 2 := by omega35 : (q : ℤ) ⩾ 3 :
= by omega36 : (r : ℤ) ⩾ 4 := by omega37 : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (
q - 1) * (r - 1)) := by linarith have h38 : False := by39 : (p : ℤ) = 2 :
= by omega40 : (q : ℤ) ⩾ 3 := by omega41 : (r : ℤ) ⩾ 4 := by omega42 : (p : ℤ) *
q * r - 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith have h43 : 2 * q *
r - 1 = 2 * (1 * (q - 1) * (r - 1)) := by simp [h39] at h42 ⊢;> ring_nf at h42
⊢ <;> linarith44 : 2 * q * r - 1 = 2 * ((q - 1) * (r - 1)) := by ring_nf at h43
⊢ <;> linarith45 : 2 * q * r - 1 = 2 * (q * r - q - r + 1) := by ring_nf at h44
⊢ <;> linarith46 : 2 * q * r - 1 = 2 * q * r - 2 * q - 2 * r + 2 := by
ring_nf at h45 ⊢ <;> linarith
We need to show that p cannot be greater than 3

```
    have h₄₇ : -1 = -2 * q - 2 * r + 2 := by
     linarith
    have h₄₈ : 2 * q + 2 * r = 3 := by linarith
    have h₄₉ : (q : ℤ) ⩾ 3 := by omega
    have h₅₀ : (r : ℤ) ⩾ 4 := by omega
    have h₅₁ : 2 * q + 2 * r ⩾ 14 := by
     nlinarith
    linarith
   exact h₃₈
  have h₁₉ : q ⩾ p + 1 := by omega
  have h₂₀ : r ⩾ q + 1 := by omega
  have h₂₁ : (p : ℤ) ⩾ 4 := by omega
  have h₂₂ : (q : ℤ) ⩾ 5 := by omega
  have h₂₃ : (r : ℤ) ⩾ 6 := by omega
  have h₂₄ : (p : ℤ) * q ⩾ 4 * 5 := by nlinarith
  have h₂₅ : (p : ℤ) * r ⩾ 4 * 6 := by nlinarith
  have h₂₆ : (q : ℤ) * r ⩾ 5 * 6 := by nlinarith
  have h₂₇ : (p : ℤ) * q * r ⩾ 4 * 5 * 6 := by nlinarith
  have h₂₈ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
  have h₂₉ : (p : ℤ) ⩾ 4 := by omega
  have h₃₀ : (q : ℤ) ⩾ 5 := by omega
  have h₃₁ : (r : ℤ) ⩾ 6 := by omega
  have h₃₂ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
  have h₃₃ : False := by
   nlinarith [sq_nonneg ((p : ℤ) - 2), sq_nonneg ((q : ℤ) - 2), sq_nonneg ((r : ℤ) - 2),
    mul_nonneg (sub_nonneg.mpr h₂) (sub_nonneg.mpr h₃),
    mul_nonneg (sub_nonneg.mpr h₂) (sub_nonneg.mpr h₄),
    mul_nonneg (sub_nonneg.mpr h₃) (sub_nonneg.mpr h₄)]
  exact h₃₃
 have h₁₆ : q = 5 := by
  have h₁₇ : p = 3 := by linarith
  have h₁₈ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
  have h₁₉ : (p : ℤ) = 3 := by norm_num [h₁₇]
  have h₂₀ : (q : ℤ) ⩾ 4 := by
   by_contra h₂₀
   have h₂₁ : q ⩽ 3 := by linarith
   have h₂₂ : q = 3 := by linarith
   have h₂₃ : (p : ℤ) = 3 := by norm_num [h₁₇]
   have h₂₄ : (q : ℤ) = 3 := by norm_num [h₂₂]
   have h₂₅ : (r : ℤ) ⩾ 4 := by linarith
   norm_num [h₁₇, h₂₂, h₂₃, h₂₄] at h₁₈
   <;>
    (try omega) <;>
    (try nlinarith) <;>
    (try
     {
      nlinarith [mul_pos (sub_pos.mpr h₀.2.1) (sub_pos.mpr h₀.2.2)]
     })
  have h₂₁ : (r : ℤ) ⩾ q + 1 := by linarith
  have h₂₂ : (q : ℤ) ⩾ 4 := by linarith
  have h₂₃ : (p : ℤ) = 3 := by norm_num [h₁₇]
  have h₂₄ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
  have h₂₅ : 3 * q * r - 1 = 2 * (2 * (q - 1) * (r - 1)) := by
   norm_num [h₁₇, h₁₉] at h₂₄ ⊢
   <;> ring_nf at h₂₄ ⊢ <;> linarith
  have h₂₆ : 3 * q * r - 1 = 4 * ((q - 1) * (r - 1)) := by
   ring_nf at h₂₅ ⊢
   <;> nlinarith
  have h₂₇ : 3 * q * r - 1 = 4 * (q * r - q - r + 1) := by
   ring_nf at h₂₆ ⊢
   <;> nlinarith
```

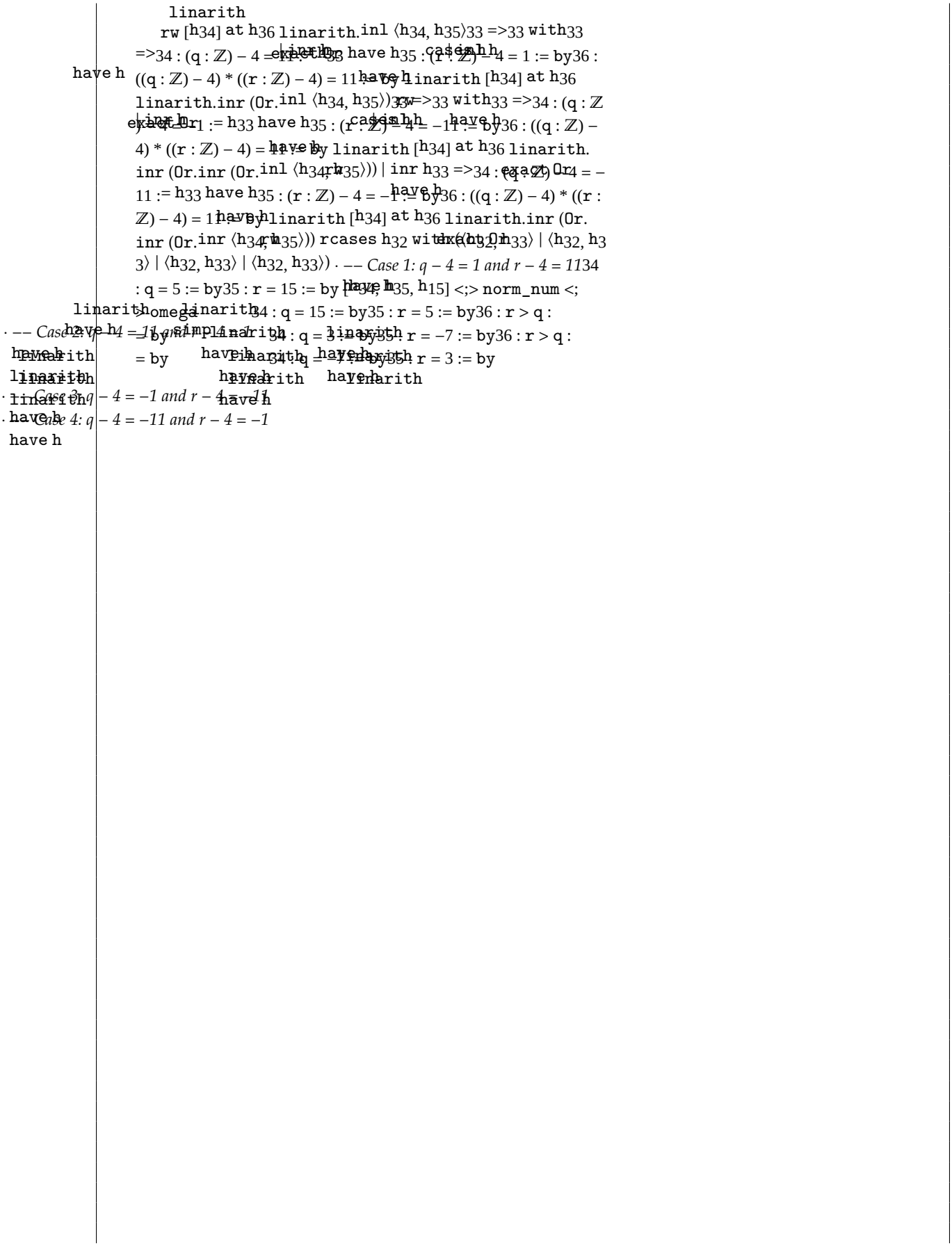
have h47 : −1 = −2 * q − 2 * r + 2 := by48 : 2 * q + 2 * r = 3 := by linarith have h49 :
(q : ℤ) ≥ 3 := by omega50 : (r : ℤ) ≥ 4 := by omega51 : 2 * q + 2 * r ≥ 14 := by38 have h19
: q ≥ p + 1 := by omega20 : r ≥ q + 1 := by omega21 : (p : ℤ) ≥ 4 := by omega22 : (q : ℤ) ≥
5 := by omega23 : (r : ℤ) ≥ 6 := by omega24 : (p : ℤ) * q ≥ 4 * 5 := by nlinarith have h2
5 : (p : ℤ) * r ≥ 4 * 6 := by nlinarith have h26 : (q : ℤ) * r ≥ 5 * 6 := by nlinarith
have h27 : (p : ℤ) * q * r ≥ 4 * 5 * 6 := by nlinarith have h28 : (p : ℤ) * q * r − 1 = 2 * ((
p − 1) * (q − 1) * (r − 1)) := by linarith have h29 : (p : ℤ) ≥ 4 := by omega30 : (q : ℤ) ≥ 5
:= by omega31 : (r : ℤ) ≥ 6 := by omega32 : (p : ℤ) * q * r − 1 = 2 * ((p − 1) * (q − 1) * (r −
1)) := by linarith have h33 : False := by [sq_nonneg ((p : ℤ) − 2), sq_nonneg ((q :
ℤ) − 2), sq_nonneg ((r : ℤ) − 2), mul_nonneg (sub_nonneg.mpr h2) (sub_nonneg.mpr h
3), mul_nonneg (sub_nonneg.mpr h2) (sub_nonneg.mpr h4), mul_nonneg (sub_nonneg
.mpr h3) (sub_nonneg.mpr h4)] exact h33 have h16 : q = 5 := by17 : p = 3 :
= by linarith18 : (p : ℤ) * q * r − 1 = 2 * ((p − 1) * (q − 1) * (r − 1)) := by linarith
have h19 : (p : ℤ) = 3 := by norm_num [h17] have h20 : (q : ℤ) ≥ 4 := by20 have h21 : q ≤
3 := by linarith22 : q = 3 := by linarith23 : (p : ℤ) = 3 := by norm_num [h17] have h2
4 : (q : ℤ) = 3 := by norm_num [h22] have h25 : (r : ℤ) ≥ 4 := by linarith [h17, h22, h23,
h24] at h18 <;> (try omega) <;> nlinarith [mul_pos (sub_pos.mpr h0.2.1
) (sub_pos.mpr h0.2.2)] have h21 : (r : ℤ) ≥ q + 1 := by linarith have h22 : (q : ℤ) ≥ 4 :
= by linarith23 : (p : ℤ) = 3 := by norm_num [h17] have h24 : (p : ℤ) * q * r − 1 = 2 * ((
p − 1) * (q − 1) * (r − 1)) := by linarith have h25 : 3 * q * r − 1 = 2 * (2 * (q − 1) * (r − 1)
) := by norm_num [h17, h19] at h24 ⊢ <;> ring_nf at h24 ⊢ <;> linarith26 : 3 * q * r − 1
= 4 * ((q − 1) * (r − 1)) := by ring_nf at h25 ⊢ <;> nlinarith27 : 3 * q * r − 1 = 4 * (q *
r − q − r + 1) := by ring_nf at h26 ⊢ <;> nlinarith

have h have h have h have h have h have h linarith have h have h nlinarith nlinarith exact h have h have h have h nlinarith have h by_contra h have h norm_num have h

```
have h₂₈ : 3 * q * r - 1 = 4 * q * r - 4 * q - 4 * r + 4 := by
 ring_nf at h₂₇ ⊢
 <;> nlinarith
have h₂₉ : - q * r + 4 * q + 4 * r - 5 = 0 := by
 ring_nf at h₂₈ ⊢
 <;> nlinarith
have h₃₀ : (q : ℤ) * r - 4 * q - 4 * r + 5 = 0 := by
 ring_nf at h₂₉ ⊢
 <;> nlinarith
have h₃₁ : (q - 4 : ℤ) * (r - 4 : ℤ) = 11 := by
 ring_nf at h₃₀ ⊢
 <;> nlinarith
have h₃₂ : (q : ℤ) - 4 = 1 ∧ (r : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = 11 ∧ (r : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = -1 ∧ (r : ℤ) - 4 =
-11 ∨ (q : ℤ) - 4 = -11 ∧ (r : ℤ) - 4 = -1 := by
 have h₃₃ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -1 ∨ (q : ℤ) - 4 = -11 := by
  have h₃₄ : (q : ℤ) - 4 ∣ 11 := by
   use (r : ℤ) - 4
   linarith
  have h₃₅ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -1 ∨ (q : ℤ) - 4 = -11 := by
   have h₃₆ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -1 ∨ (q : ℤ) - 4 = -11 := by
    rw [← Int.natAbs_dvd_natAbs] at h₃₄
    -- We use the fact that the absolute value of (q - 4) divides the absolute value of 11
    have h₃₇ : ((q : ℤ) - 4).natAbs ∣ 11 := by
     simpa [Int.natAbs] using h₃₄
    -- Since the possible divisors of 11 are 1 and 11, we check the cases
    have h₃₈ : ((q : ℤ) - 4).natAbs = 1 ∨ ((q : ℤ) - 4).natAbs = 11 := by
     have h₃₉ : ((q : ℤ) - 4).natAbs ∣ 11 := h₃₇
     have h₄₀ : ((q : ℤ) - 4).natAbs ≤ 11 := Nat.le_of_dvd (by decide) h₃₉
     interval_cases ((q : ℤ) - 4).natAbs <;> norm_num at h₃₉ ⊢ <;> omega
    cases h₃₈ with
    | inl h₃₈ =>
     have h₄₁ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = -1 := by
      have h₄₂ : ((q : ℤ) - 4).natAbs = 1 := h₃₈
      have h₄₃ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = -1 := by
       rw [Int.natAbs_eq_iff] at h₄₂
       tauto
      exact h₄₃
     cases h₄₁ with
     | inl h₄₁ =>
      tauto
     | inr h₄₁ =>
      tauto
    | inr h₃₈ =>
     have h₄₁ : (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -11 := by
      have h₄₂ : ((q : ℤ) - 4).natAbs = 11 := h₃₈
      have h₄₃ : (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -11 := by
       rw [Int.natAbs_eq_iff] at h₄₂
       tauto
      exact h₄₃
     cases h₄₁ with
     | inl h₄₁ =>
      tauto
     | inr h₄₁ =>
      tauto
   exact h₃₆
  exact h₃₅
 cases h₃₃ with
 | inl h₃₃ =>
  have h₃₄ : (q : ℤ) - 4 = 1 := h₃₃
  have h₃₅ : (r : ℤ) - 4 = 11 := by
   have h₃₆ : ((q : ℤ) - 4) * ((r : ℤ) - 4) = 11 := by
```

```
have h₂₈ : 3 * q * r - 1 = 4 * q * r - 4 * q - 4 * r + 4 := by ring_nf at h₂₇ ⊢;> nlinarith
have h₂₉ : -q * r + 4 * q + 4 * r - 5 = 0 := by ring_nf at h₂₈ ⊢;> nlinarith
have h₃₀ : (q : ℤ) * r - 4 * q - 4 * r + 5 = 0 := by ring_nf at h₂₉ ⊢;> nlinarith
have h₃₁ : (q - 4 : ℤ) * (r - 4 : ℤ) = 11 := by ring_nf at h₃₀ ⊢;> nlinarith
have h₃₂ : (q : ℤ) - 4 = 1 ∧ (r : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = 11 ∧ (r : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = -1 ∧ (r : ℤ) - 4 = -11 ∨ (q : ℤ) - 4 = -11 ∧ (r : ℤ) - 4 = -1 := by
have h₃₃ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -1 ∨ (q : ℤ) - 4 = -11 := by
have h₃₄ : (q : ℤ) - 4 ∣ 11 := by
use (r : ℤ) - 4
have h₃₆ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -1 ∨ (q : ℤ) - 4 = -11 := by rw [← Int.natAbs_dvd_natAbs] at h₃₄
-- We use the fact that the absolute value of (q - 4) divides the absolute value of 11
have h₃₇ : ((q : ℤ) - 4).natAbs ∣ 11 := by simpa [Int.natAbs] using h₃₄
-- Since the possible divisors of 11 are 1 and 11, we check the cases
have h₃₈ : ((q : ℤ) - 4).natAbs = 1 ∨ ((q : ℤ) - 4).natAbs = 11 := by
have h₃₉ : ((q : ℤ) - 4).natAbs ∣ 11 := h₃₇
have h₄₀ : ((q : ℤ) - 4).natAbs ≤ 11 := Nat.le_of_dvd (by decide) h₃₉
interval_cases ((q : ℤ) - 4).natAbs <;> norm_num at h₃₉ ⊢ <;> omega
cases h₃₈ with
| inl h =>
  have h₄₁ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = -1 := by rw [Int.natAbs_eq_iff] at h
  cases h₄₁ with
  | inl h =>
    have h₄₂ : ((q : ℤ) - 4).natAbs = 1 := by rw [h]; rfl
    exact h₄₂
  | inr h =>
    have h₄₂ : ((q : ℤ) - 4).natAbs = 11 := by rw [h]; rfl
    exact h₄₂
| inr h =>
  have h₄₁ : (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -11 := by rw [Int.natAbs_eq_iff] at h
  cases h₄₁ with
  | inl h =>
    have h₄₂ : ((q : ℤ) - 4).natAbs = 11 := by rw [h]; rfl
    exact h₄₂
  | inr h =>
    have h₄₂ : ((q : ℤ) - 4).natAbs = 1 := by rw [h]; rfl
    exact h₄₂
have h₃₅ : (r : ℤ) - 4 = 11 := by
have h₃₆ : ((q : ℤ) - 4) * ((r : ℤ) - 4) = 11 := by
  have h₃₇ : ((q : ℤ) - 4).natAbs ∣ 11 := h₃₇
  have h₃₈ : ((q : ℤ) - 4).natAbs = 1 ∨ ((q : ℤ) - 4).natAbs = 11 := by
    have h₃₉ : ((q : ℤ) - 4).natAbs ∣ 11 := h₃₇
    have h₄₀ : ((q : ℤ) - 4).natAbs ≤ 11 := Nat.le_of_dvd (by decide) h₃₉
    interval_cases ((q : ℤ) - 4).natAbs <;> norm_num at h₃₉ ⊢ <;> omega
  cases h₃₈ with
  | inl h =>
    have h₄₁ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = -1 := by rw [Int.natAbs_eq_iff] at h
    cases h₄₁ with
    | inl h =>
      have h₄₂ : ((q : ℤ) - 4).natAbs = 1 := by rw [h]; rfl
      exact h₄₂
    | inr h =>
      have h₄₂ : ((q : ℤ) - 4).natAbs = 11 := by rw [h]; rfl
      exact h₄₂
  | inr h =>
    have h₄₁ : (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -11 := by rw [Int.natAbs_eq_iff] at h
    cases h₄₁ with
    | inl h =>
      have h₄₂ : ((q : ℤ) - 4).natAbs = 11 := by rw [h]; rfl
      exact h₄₂
    | inr h =>
      have h₄₂ : ((q : ℤ) - 4).natAbs = 1 := by rw [h]; rfl
      exact h₄₂
have h₃₅ : (r : ℤ) - 4 = 11 := by
  have h₃₆ : ((q : ℤ) - 4) * ((r : ℤ) - 4) = 11 := by
    have h₃₇ : ((q : ℤ) - 4).natAbs ∣ 11 := h₃₇
    have h₃₈ : ((q : ℤ) - 4).natAbs = 1 ∨ ((q : ℤ) - 4).natAbs = 11 := by
      have h₃₉ : ((q : ℤ) - 4).natAbs ∣ 11 := h₃₇
      have h₄₀ : ((q : ℤ) - 4).natAbs ≤ 11 := Nat.le_of_dvd (by decide) h₃₉
      interval_cases ((q : ℤ) - 4).natAbs <;> norm_num at h₃₉ ⊢ <;> omega
    cases h₃₈ with
    | inl h =>
      have h₄₁ : (q : ℤ) - 4 = 1 ∨ (q : ℤ) - 4 = -1 := by rw [Int.natAbs_eq_iff] at h
      cases h₄₁ with
      | inl h =>
        have h₄₂ : ((q : ℤ) - 4).natAbs = 1 := by rw [h]; rfl
        exact h₄₂
      | inr h =>
        have h₄₂ : ((q : ℤ) - 4).natAbs = 11 := by rw [h]; rfl
        exact h₄₂
    | inr h =>
      have h₄₁ : (q : ℤ) - 4 = 11 ∨ (q : ℤ) - 4 = -11 := by rw [Int.natAbs_eq_iff] at h
      cases h₄₁ with
      | inl h =>
        have h₄₂ : ((q : ℤ) - 4).natAbs = 11 := by rw [h]; rfl
        exact h₄₂
      | inr h =>
        have h₄₂ : ((q : ℤ) - 4).natAbs = 1 := by rw [h]; rfl
        exact h₄₂
have h
```

```
      linarith
    rw [h₃₄] at h₃₆
    linarith
  exact Or.inl ⟨h₃₄, h₃₅⟩
| inr h₃₃ =>
  cases h₃₃ with
  | inl h₃₃ =>
    have h₃₄ : (q : ℤ) - 4 = 11 := h₃₃
    have h₃₅ : (r : ℤ) - 4 = 1 := by
      have h₃₆ : ((q : ℤ) - 4) * ((r : ℤ) - 4) = 11 := by
        linarith
      rw [h₃₄] at h₃₆
      linarith
    exact Or.inr (Or.inl ⟨h₃₄, h₃₅⟩)
  | inr h₃₃ =>
    cases h₃₃ with
    | inl h₃₃ =>
      have h₃₄ : (q : ℤ) - 4 = -1 := h₃₃
      have h₃₅ : (r : ℤ) - 4 = -11 := by
        have h₃₆ : ((q : ℤ) - 4) * ((r : ℤ) - 4) = 11 := by
          linarith
        rw [h₃₄] at h₃₆
        linarith
      exact Or.inr (Or.inr (Or.inl ⟨h₃₄, h₃₅⟩))
    | inr h₃₃ =>
      have h₃₄ : (q : ℤ) - 4 = -11 := h₃₃
      have h₃₅ : (r : ℤ) - 4 = -1 := by
        have h₃₆ : ((q : ℤ) - 4) * ((r : ℤ) - 4) = 11 := by
          linarith
        rw [h₃₄] at h₃₆
        linarith
      exact Or.inr (Or.inr (Or.inr ⟨h₃₄, h₃₅⟩))
rcases h₃₂ with (⟨h₃₂, h₃₃⟩ | ⟨h₃₂, h₃₃⟩ | ⟨h₃₂, h₃₃⟩ | ⟨h₃₂, h₃₃⟩)
· -- Case 1: q - 4 = 1 and r - 4 = 11
  have h₃₄ : q = 5 := by
    linarith
  have h₃₅ : r = 15 := by
    linarith
  simp [h₃₄, h₃₅, h₁₅]
  <;> norm_num <;> omega
· -- Case 2: q - 4 = 11 and r - 4 = 1
  have h₃₄ : q = 15 := by
    linarith
  have h₃₅ : r = 5 := by
    linarith
  have h₃₆ : r > q := by
    linarith
  linarith
· -- Case 3: q - 4 = -1 and r - 4 = -11
  have h₃₄ : q = 3 := by
    linarith
  have h₃₅ : r = -7 := by
    linarith
  have h₃₆ : r > q := by
    linarith
  linarith
· -- Case 4: q - 4 = -11 and r - 4 = -1
  have h₃₄ : q = -7 := by
    linarith
  have h₃₅ : r = 3 := by
    linarith
```


linarith
rw [h34] at h36 linarith.inl ⟨h34, h35⟩33 =>33 with33
=>34 : (q : ℤ) − 4 = 1 ∧ r − 4 = 11 ∨ q − 4 = −1 ∧ r − 4 = −11 ∨ q − 4 = 11 ∧ r − 4 = 1 ∨ q − 4 = −11 ∧ r − 4 = −1 := h33 have h35 : (r : ℤ) − 4 = 1 := by36 :
have h ((q : ℤ) − 4) * ((r : ℤ) − 4) = 11 := by linarith [h34] at h36
linarith.inr (Or.inl ⟨h34, h35⟩)33 =>33 with33 =>34 : (q : ℤ
ext h | inr h33 =>34 : (q : ℤ) − 4 = −1 ∧ r − 4 = −11 ∨ q − 4 = 1 ∧ r − 4 = 11 ∨ q − 4 = −11 ∧ r − 4 = 1 ∨ q − 4 = 11 ∧ r − 4 = −1 := h33 have h35 : (r : ℤ) − 4 = −11 := by36 : ((q : ℤ) −
4) * ((r : ℤ) − 4) = 11 := by linarith [h34] at h36 linarith.
inr (Or.inr (Or.inl ⟨h34, h35⟩)) | inr h33 =>34 : (q : ℤ) − 4 = −
11 := h33 have h35 : (r : ℤ) − 4 = −1 := by36 : ((q : ℤ) − 4) * ((r :
ℤ) − 4) = 11 := by linarith [h34] at h36 linarith.inr (Or.
inr (Or.inr ⟨h34, h35⟩))) rcases h32 with ⟨h32, h33⟩ | ⟨h32, h3
3⟩ | ⟨h32, h33⟩ | ⟨h32, h33⟩) · -- *Case 1: q − 4 = 1 and r − 4 = 11*34
: q = 5 := by35 : r = 15 := by [h34, h35, h15] <;> norm_num <;>
linarith omega linarith 34 : q = 15 := by35 : r = 5 := by36 : r > q :
· -- *Case 2: q − 4 = −1 and r − 4 = −11* simp_all linarith
have h = by linarith have h linarith have h
linarith linarith linarith
· *Case 3: q − 4 = −1 and r − 4 = 11*
linarith have h
have h
· *Case 4: q − 4 = −11 and r − 4 = −1*
have h

```
    have h₃₆ : r > q := by
     linarith
    linarith
  have h₁₇ : r = 15 := by
   have h₁₈ : p = 3 := by linarith
   have h₁₉ : q = 5 := by linarith
   have h₂₀ : (p : ℤ) * q * r - 1 = 2 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
   have h₂₁ : (p : ℤ) = 3 := by norm_num [h₁₈]
   have h₂₂ : (q : ℤ) = 5 := by norm_num [h₁₉]
   norm_num [h₁₈, h₁₉, h₂₁, h₂₂] at h₂₀
   <;> ring_nf at h₂₀ ⊢ <;> omega
  have h₁₈ : (p, q, r) = (3, 5, 15) := by
   simp_all [Prod.ext_iff]
   <;> norm_num <;> linarith
  exact Or.inr h₁₈
· -- Case k = 3
  have h₁₄ : p * q * r - 1 = 3 * ((p - 1) * (q - 1) * (r - 1)) := by
   ring_nf at hk ⊢
   <;> linarith
  have h₁₅ : p = 2 := by
   by_contra h₁₅
   have h₁₆ : p ≥ 3 := by
    by_contra h₁₆
    have h₁₇ : p ≤ 2 := by linarith
    have h₁₈ : p = 2 := by linarith
    contradiction
   have h₁₉ : q ≥ p + 1 := by linarith
   have h₂₀ : r ≥ q + 1 := by linarith
   have h₂₁ : (p : ℤ) ≥ 3 := by linarith
   have h₂₂ : (q : ℤ) ≥ 4 := by linarith
   have h₂₃ : (r : ℤ) ≥ 5 := by linarith
   have h₂₄ : (p : ℤ) * q ≥ 3 * 4 := by nlinarith
   have h₂₅ : (p : ℤ) * r ≥ 3 * 5 := by nlinarith
   have h₂₆ : (q : ℤ) * r ≥ 4 * 5 := by nlinarith
   have h₂₇ : (p : ℤ) * q * r ≥ 3 * 4 * 5 := by nlinarith
   have h₂₈ : (p : ℤ) * q * r - 1 = 3 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
   have h₂₉ : (p : ℤ) ≥ 3 := by linarith
   have h₃₀ : (q : ℤ) ≥ 4 := by linarith
   have h₃₁ : (r : ℤ) ≥ 5 := by linarith
   have h₃₂ : (p : ℤ) * q * r - 1 = 3 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
   have h₃₃ : False := by
    nlinarith [sq_nonneg ((p : ℤ) - 2), sq_nonneg ((q : ℤ) - 2), sq_nonneg ((r : ℤ) - 2),
     mul_nonneg (sub_nonneg.mpr h₂) (sub_nonneg.mpr h₃),
     mul_nonneg (sub_nonneg.mpr h₂) (sub_nonneg.mpr h₄),
     mul_nonneg (sub_nonneg.mpr h₃) (sub_nonneg.mpr h₄)]
   exact h₃₃
  have h₁₆ : q = 4 := by
   have h₁₇ : p = 2 := by linarith
   have h₁₈ : (p : ℤ) * q * r - 1 = 3 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
   have h₁₉ : (p : ℤ) = 2 := by norm_num [h₁₇]
   have h₂₀ : (q : ℤ) ≥ 3 := by
    by_contra h₂₀
    have h₂₁ : q ≤ 2 := by linarith
    have h₂₂ : q = 2 := by linarith
    have h₂₃ : (p : ℤ) = 2 := by norm_num [h₁₇]
    have h₂₄ : (q : ℤ) = 2 := by norm_num [h₂₂]
    have h₂₅ : (r : ℤ) ≥ 3 := by linarith
    norm_num [h₁₇, h₂₂, h₂₃, h₂₄] at h₁₈
    <;>
     (try omega) <;>
     (try nlinarith) <;>
```

have h36 : r > q := by17 : r = 15 := by18 : p = 3 := by linarith19 : q = 5 :
= by linarith20 : (p : ℤ) * q * r − 1 = 2 * ((p − 1) * (q − 1) * (r − 1)) := by linarith
have h21 : (p : ℤ) = 3 := by norm_num [h18] have h22 : (q : ℤ) = 5 := by norm_num [h19]
norm_num [h18, h19, h21, h22] at h20 <;> ring_nf at h20 ⊢ <;> omega18 : (p, q, r) = (3, 5,
15) := by [Prod.ext_iff] <;> norm_num <;> linarith. inr h18 · −− *Case k = 3*14 : p * q *
r − 1 = 3 * ((p − 1) * (q − 1) * (r − 1)) := by ring_nf at hk ⊢;> linarith15 : p = 2 := by15
have h16 : p ⩾ 3 := by16 have h17 : p ⩽ 2 := by linarith18 : p = 2 := by linarith19 :
q ⩾ p + 1 := by linarith20 : r ⩾ q + 1 := by linarith21 : (p : ℤ) ⩾ 3 := by linarith22 :
(q : ℤ) ⩾ 4 := by linarith23 : (r : ℤ) ⩾ 5 := by linarith24 : (p : ℤ) * q ⩾ 3 * 4 :
= by nlinarith have h25 : (p : ℤ) * r ⩾ 3 * 5 := by nlinarith have h26 : (q : ℤ) * r ⩾ 4 *
5 := by nlinarith have h27 : (p : ℤ) * q * r ⩾ 3 * 4 * 5 := by nlinarith have h28 : (p : ℤ
) * q * r − 1 = 3 * ((p − 1) * (q − 1) * (r − 1)) := by linarith have h29 : (p : ℤ) ⩾ 3 :
= by linarith30 : (q : ℤ) ⩾ 4 := by linarith31 : (r : ℤ) ⩾ 5 := by linarith32 : (p : ℤ) *
q * r − 1 = 3 * ((p − 1) * (q − 1) * (r − 1)) := by linarith have h33 : False := by [
sq_nonneg ((p : ℤ) − 2), sq_nonneg ((q : ℤ) − 2), sq_nonneg ((r : ℤ) − 2),
mul_nonneg (sub_nonneg.mpr h2) (sub_nonneg.mpr h3), mul_nonneg (sub_nonneg.
mpr h2) (sub_nonneg.mpr h4), mul_nonneg (sub_nonneg.mpr h3) (sub_nonneg.mpr h4)]
exact h33 have h16 : q = 4 := by17 : p = 2 := by linarith18 : (p : ℤ) * q * r − 1 = 3 * ((p −
1) * (q − 1) * (r − 1)) := by linarith have h19 : (p : ℤ) = 2 := by norm_num [h17] have h20
: (q : ℤ) ⩾ 3 := by20 have h21 : q ⩽ 2 := by linarith22 : q = 2 := by linarith23 : (p : ℤ)
= 2 := by norm_num [h17] have h24 : (q : ℤ) = 2 := by norm_num [h22] have h25 : (r : ℤ) ⩾
3 := by linarith [h17, h22, h23, h24] at h18 <;> (try omega) <;> (try nlinarith) <;>

have h
_all
have h
have h
have h
have h
by_contra h
norm_num
linarith
linarith
have h
have h
exact Or
have h
by_contra h
by_contra h
contradiction
have h
have h
have h
have h
have h
nlinarith

```
  (try
   {
    nlinarith [mul_pos (sub_pos.mpr h₀.2.1) (sub_pos.mpr h₀.2.2)]
   })
have h₂₁ : (r : ℤ) ⩾ q + 1 := by linarith
have h₂₂ : (q : ℤ) ⩾ 3 := by linarith
have h₂₃ : (p : ℤ) = 2 := by norm_num [h₁₇]
have h₂₄ : (p : ℤ) * q * r - 1 = 3 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
have h₂₅ : 2 * q * r - 1 = 3 * (1 * (q - 1) * (r - 1)) := by
 norm_num [h₁₇, h₁₉] at h₂₄ ⊢
 <;> ring_nf at h₂₄ ⊢ <;> linarith
have h₂₆ : 2 * q * r - 1 = 3 * ((q - 1) * (r - 1)) := by
 ring_nf at h₂₅ ⊢
 <;> nlinarith
have h₂₇ : 2 * q * r - 1 = 3 * (q * r - q - r + 1) := by
 ring_nf at h₂₆ ⊢
 <;> nlinarith
have h₂₈ : 2 * q * r - 1 = 3 * q * r - 3 * q - 3 * r + 3 := by
 ring_nf at h₂₇ ⊢
 <;> nlinarith
have h₂₉ : - q * r + 3 * q + 3 * r - 4 = 0 := by
 ring_nf at h₂₈ ⊢
 <;> nlinarith
have h₃₀ : (q : ℤ) * r - 3 * q - 3 * r + 4 = 0 := by
 ring_nf at h₂₉ ⊢
 <;> nlinarith
have h₃₁ : (q - 3 : ℤ) * (r - 3 : ℤ) = 5 := by
 ring_nf at h₃₀ ⊢
 <;> nlinarith
have h₃₂ : (q : ℤ) - 3 = 1 ∧ (r : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = 5 ∧ (r : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = -1 ∧ (r : ℤ) - 3 = -5
∨ (q : ℤ) - 3 = -5 ∧ (r : ℤ) - 3 = -1 := by
 have h₃₃ : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -1 ∨ (q : ℤ) - 3 = -5 := by
  have h₃₄ : (q : ℤ) - 3 ∣ 5 := by
   use (r : ℤ) - 3
   linarith
  have h₃₅ : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -1 ∨ (q : ℤ) - 3 = -5 := by
   have h₃₆ : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -1 ∨ (q : ℤ) - 3 = -5 := by
    rw [← Int.natAbs_dvd_natAbs] at h₃₄
    -- We use the fact that the absolute value of (q - 3) divides the absolute value of 5
    have h₃₇ : ((q : ℤ) - 3).natAbs ∣ 5 := by
     simpa [Int.natAbs] using h₃₄
    -- Since the possible divisors of 5 are 1 and 5, we check the cases
    have h₃₈ : ((q : ℤ) - 3).natAbs = 1 ∨ ((q : ℤ) - 3).natAbs = 5 := by
     have h₃₉ : ((q : ℤ) - 3).natAbs ∣ 5 := h₃₇
     have h₄₀ : ((q : ℤ) - 3).natAbs ⩽ 5 := Nat.le_of_dvd (by decide) h₃₉
     interval_cases ((q : ℤ) - 3).natAbs <;> norm_num at h₃₉ ⊢ <;> omega
    cases h₃₈ with
    | inl h₃₈ =>
     have h₄₁ : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = -1 := by
      have h₄₂ : ((q : ℤ) - 3).natAbs = 1 := h₃₈
      have h₄₃ : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = -1 := by
       rw [Int.natAbs_eq_iff] at h₄₂
       tauto
      exact h₄₃
     cases h₄₁ with
     | inl h₄₁ =>
      tauto
     | inr h₄₁ =>
      tauto
    | inr h₃₈ =>
     have h₄₁ : (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -5 := by
```

```
(try { nlinarith [mul_pos (sub_pos.mpr h0.2.1) (sub_pos.mpr h0.2.2)] }) have h21 : (r : ℤ) ≥ q + 1 :
= by linarith have h22 : (q : ℤ) ≥ 3 := by linarith have h23 : (p : ℤ) = 2 := by norm_num [h17] have h24 : (p : ℤ
) * q * r - 1 = 3 * ((p - 1) * (q - 1) * (r - 1)) := by linarith have h25 : 2 * q * r - 1 = 3 * (1 * (q - 1) * (r - 1)
) := by norm_num [h17, h19] at h24 ⊢;> ring_nf at h24 ⊢ <;> linarith have h26 : 2 * q * r - 1 = 3 * ((q - 1) * (r -
1)) := by ring_nf at h25 ⊢;> nlinarith have h27 : 2 * q * r - 1 = 3 * (q * r - q - r + 1) := by ring_nf at h26 ⊢;
> nlinarith have h28 : 2 * q * r - 1 = 3 * q * r - 3 * q - 3 * r + 3 := by ring_nf at h27 ⊢;> nlinarith have h29 : - q *
r + 3 * q + 3 * r - 4 = 0 := by ring_nf at h28 ⊢;> nlinarith have h30 : (q : ℤ) * r - 3 * q - 3 * r + 4 = 0 := by
ring_nf at h29 ⊢;> nlinarith have h31 : (q - 3 : ℤ) * (r - 3 : ℤ) = 5 := by ring_nf at h30 ⊢;> nlinarith have h32 : (q :
ℤ) - 3 = 1 ∧ (r : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = 5 ∧ (r : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = -1 ∧ (r : ℤ) - 3 = -5 ∨ (q : ℤ) - 3
= -5 ∧ (r : ℤ) - 3 = -1 := by have h33 : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -1 ∨ (q : ℤ) - 3 = -5 :
= by have h34 : (q : ℤ) - 3 | 5 := by : ℤ) - 3 linarith have h35 : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -1
∨ (q : ℤ) - 3 = -5 := by have h36 : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -1 ∨ (q : ℤ) - 3 = -5 := by
rw [← Int.natAbs_dvd_natAbs] at h34
-- We use the fact that the absolute value of (q - 3) divides the absolute value of 5 have h37 : ((q : ℤ) - 3).natAbs |
5 := by [Int.natAbs] using h34 -- Since the possible divisors of 5 are 1 and 5, we check the cases have h38 : ((q :
ℤ) - 3).natAbs = 1 ∨ ((q : ℤ) - 3).natAbs = 5 := by have h39 : ((q : ℤ) - 3).natAbs | 5 := h37 have h40 : ((q :
ℤ) - 3).natAbs ≤ 5 := Nat.le_of_dvd (by decide) h39 interval_cases ((q : ℤ) - 3).natAbs <;
> norm_num at h39 ⊢ <;> omega cases h38 with h38 =>41 : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = -1 := by have h42 : (
(q : ℤ) - 3).natAbs = 1 := h38 have h43 : (q : ℤ) - 3 = 1 ∨ (q : ℤ) - 3 = -1 := by rw [Int.natAbs_eq_iff]
at h42 exact h43 cases h41 with h41 =>41 =>38 =>41 : (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -5 := by
  | inl h => tauto
  | inr h => tauto have h
  | inl h => inl h
  | inr h => inr h
```

```
        have h₄₂ : ((q : ℤ) - 3).natAbs = 5 := h₃₈
        have h₄₃ : (q : ℤ) - 3 = 5 ∨ (q : ℤ) - 3 = -5 := by
         rw [Int.natAbs_eq_iff] at h₄₂
         tauto
        exact h₄₃
       cases h₄₁ with
       | inl h₄₁ =>
        tauto
       | inr h₄₁ =>
        tauto
     exact h₃₆
    exact h₃₅
  cases h₃₃ with
  | inl h₃₃ =>
   have h₃₄ : (q : ℤ) - 3 = 1 := h₃₃
   have h₃₅ : (r : ℤ) - 3 = 5 := by
    have h₃₆ : ((q : ℤ) - 3) * ((r : ℤ) - 3) = 5 := by
     linarith
    rw [h₃₄] at h₃₆
    linarith
   exact Or.inl ⟨h₃₄, h₃₅⟩
  | inr h₃₃ =>
   cases h₃₃ with
   | inl h₃₃ =>
    have h₃₄ : (q : ℤ) - 3 = 5 := h₃₃
    have h₃₅ : (r : ℤ) - 3 = 1 := by
     have h₃₆ : ((q : ℤ) - 3) * ((r : ℤ) - 3) = 5 := by
      linarith
     rw [h₃₄] at h₃₆
     linarith
    exact Or.inr (Or.inl ⟨h₃₄, h₃₅⟩)
   | inr h₃₃ =>
    cases h₃₃ with
    | inl h₃₃ =>
     have h₃₄ : (q : ℤ) - 3 = -1 := h₃₃
     have h₃₅ : (r : ℤ) - 3 = -5 := by
      have h₃₆ : ((q : ℤ) - 3) * ((r : ℤ) - 3) = 5 := by
       linarith
      rw [h₃₄] at h₃₆
      linarith
     exact Or.inr (Or.inr (Or.inl ⟨h₃₄, h₃₅⟩))
    | inr h₃₃ =>
     have h₃₄ : (q : ℤ) - 3 = -5 := h₃₃
     have h₃₅ : (r : ℤ) - 3 = -1 := by
      have h₃₆ : ((q : ℤ) - 3) * ((r : ℤ) - 3) = 5 := by
       linarith
      rw [h₃₄] at h₃₆
      linarith
     exact Or.inr (Or.inr (Or.inr ⟨h₃₄, h₃₅⟩))
rcases h₃₂ with (⟨h₃₂, h₃₃⟩ | ⟨h₃₂, h₃₃⟩ | ⟨h₃₂, h₃₃⟩ | ⟨h₃₂, h₃₃⟩)
· -- Case 1: q - 3 = 1 and r - 3 = 5
 have h₃₄ : q = 4 := by
  linarith
 have h₃₅ : r = 8 := by
  linarith
 simp [h₃₄, h₃₅, h₁₅]
 <;> norm_num <;> omega
· -- Case 2: q - 3 = 5 and r - 3 = 1
 have h₃₄ : q = 8 := by
  linarith
 have h₃₅ : r = 4 := by
```

have h42 : ((q : ℤ) − 3).natAbs = 5 := h38 have h43 : (q : ℤ) − 3 = 5 ∨ (q : ℤ) − 3 = −5 := by rw [Int.natAbs_eq_iff] at h42 tauto exact h43 cases h41 with | inl h41 => | inl h36 => exact h35

cases h33 with | inl h33 => have h34 : (q : ℤ) − 3 = 1 := h33 have h35 : (r : ℤ) − 3 = 5 := by tauto tauto | inl h exact h have h36 : ((q : ℤ) − 3) * ((r : ℤ) − 3) = 5 := by linarith

have h [h34] at h36 linarith.inl ⟨h34, h35⟩ | inr h33 => cases h33 with | inl h33 => have h34 : (q : ℤ) − 3 = 5 := exact h33 have h35 : (r : ℤ) − 3 = 1 := by have h36 : ((q : ℤ) − 3) * ((r : ℤ) − 3) = 5 := by have h linarith [h34] at h36 linarith.

inr (Or.inl ⟨h34, h35⟩) rw | inr h33 => cases h33 with | inl h33 => have h34 : (q : ℤ) − 3 = −1 := exact Or

| inr h = h33 have h35 : (r : ℤ) − 3 = −5 := by have h36 : ((q : ℤ) − 3) * ((r : ℤ) − 3) = 5 := by have h linarith [h34] at h36 linarith.inr (Or.

inr (Or.inl ⟨h34, h35⟩)) rw | inr h33 => have h34 : (q : ℤ) − 3 = −5 := exact Or h33

have h35 : (r : ℤ) − 3 = −1 := by have h36 : ((q : ℤ) − 3) * ((r : ℤ) − 3) = 5 := by have h linarith [h34] at h36 linarith.inr (Or.inr (Or.

inr ⟨h34, h35⟩)) rw rcases h32 with ⟨h32, h33⟩ | ⟨h32, h33⟩ | ⟨h32, h33⟩ | ⟨h32, h33⟩ exact (Or.

· −− *Case 1: q − 3 = 1 and r − 3 = 5* have h34 : q = 4 := by linarith have h35 : r = 8 := by linarith rw [h34, h35, h15] <;> norm_num <;> omega simp

· −− *Case 2: q − 3 = 5 and r − 3 = 1* have h34 : q = 8 := by linarith have h35 : r = 4 := by linarith

```
      linarith
    have h₃₆ : r > q := by
      linarith
    linarith
  · -- Case 3: q - 3 = -1 and r - 3 = -5
    have h₃₄ : q = 2 := by
      linarith
    have h₃₅ : r = -2 := by
      linarith
    have h₃₆ : r > q := by
      linarith
    linarith
  · -- Case 4: q - 3 = -5 and r - 3 = -1
    have h₃₄ : q = -2 := by
      linarith
    have h₃₅ : r = 2 := by
      linarith
    have h₃₆ : r > q := by
      linarith
    linarith
 have h₁₇ : r = 8 := by
  have h₁₈ : p = 2 := by linarith
  have h₁₉ : q = 4 := by linarith
  have h₂₀ : (p : ℤ) * q * r - 1 = 3 * ((p - 1) * (q - 1) * (r - 1)) := by linarith
  have h₂₁ : (p : ℤ) = 2 := by norm_num [h₁₈]
  have h₂₂ : (q : ℤ) = 4 := by norm_num [h₁₉]
  norm_num [h₁₈, h₁₉, h₂₁, h₂₂] at h₂₀
  <;> ring_nf at h₂₀ ⊢ <;> omega
 have h₁₈ : (p, q, r) = (2, 4, 8) := by
  simp_all [Prod.ext_iff]
  <;> norm_num <;> linarith
 exact Or.inl h₁₈
exact h₅
'''
```

B. Examples of the Proofs Generated by DeepSeek-Prover-V2-7B on Putnam-Bench

```
import Mathlib

open Nat Set

/--
Let H be an n × n matrix all of whose entries are ±1 and whose rows are mutually orthogonal. Suppose H has an
    a × b submatrix whose entries are all 1. Show that ab ⩽ n.
-/
theorem putnam_2005_a4
(n : ℕ)
(H : Matrix (Fin n) (Fin n) ℝ)
(a b : ℕ)
(S : Matrix (Fin a) (Fin b) ℝ)
(npos : n ⩾ 1)
(Hentries : ∀ i j : Fin n, H i j = 1 ∨ H i j = -1)
(Hortho : H.HasOrthogonalRows)
(hab : 1 ⩽ a ∧ a ⩽ n ∧ 1 ⩽ b ∧ b ⩽ n)
(Ssub : ∃ (rri : Fin a → Fin n) (cri : Fin b → Fin n), rri.Injective ∧ cri.Injective ∧ S = H.submatrix
    rri cri)
(Sentries : ∀ (i : Fin a) (j : Fin b), S i j = 1)
: a * b ⩽ n := by
```

```
linarith
have h36 : r > q := by        34 : q = 2 := by35 : r = −2 := by36 :
r > q := by  linarith   34 : q = −2 := by35 : r = 2 := by36 : r > q := by17 : r = 8 :
linarith = by18 : p = 2 := by linarith h19 : q = 4 := by linarith20 : (p : ℤ) * q * r − 1 =
have h  linarith have h  -- Case 3: q − 3 = 1 and r − 3 = 5  have h  have h  linarith
· -- Case 4: q − 3 = 5 and r − 3 = 1  (p − 1) * (q − 1) * (r − 1) := by linarith  have h21 : (p : ℤ) = 2 :
have h  = by norm_num [h18] have h22 : (q : ℤ) = 4 := by norm_num [h19] norm_num [h
18, h19, h21, h22] at h20 <;> ring_nf at h20 ⊢ <;> omega18 : (p, q, r) = (2, 4, 8)
:= by [Prod.ext_iff] <;> norm_num <;> linarith.  have h  inl h18 exact h5 ` ` `
exact Or
```

B. DeepSeek-Prover-V2-7B 在 Putnam- Bench 上生成的证明示例

```
import Mathlib

open Nat Set

/-- 设 H 为一个 n × n 阵列，其所有元素均为 ±1，并且其行彼此正交。假设 H 有一个 a × b 子矩阵，其所有元素
均为 1。证明 ab ⩽ n。 -/ theorem putnam_2005_a4 : ℕ) : Matrix (Fin n) (Fin n) ℝ) : ℕ) :
Matrix (Fin a) (Fin b) ℝ) (n : n ⩾ 1) : ∀ i j : Fin n, (H i j = 1 ∨ H i j = −1) (Hortho : H。HasOrthogonalRows)
(npos : 1 ⩽ a ∧ a ⩽ n ∧ 1 ⩽ b ∧ b ⩽ n) : ∃ (rri : Fin a → Fin n) (cri : Fin b → Fin n), rri。Injective ∧ cri。
(Ssub Injective ∧ S = H。submatrix rri cri) : ∀ (i : Fin a) (j : Fin b), S i j = 1) : a * b ⩽ n := by
(Hentries (Sentries (hab (S (a b
```

```
  obtain ⟨rri, cri, hrinj, hcrij, hsub⟩ := Ssub
  have h₁ : a ⩽ n := hab.2.1
  have h₂ : b ⩽ n := hab.2.2.2
  have h₃ : a * b ⩽ n := by
  calc
    a * b = Cardinal.toNat (a * b) := by simp [Nat.cast_inj]
    _ ⩽ Cardinal.toNat (n) := by
    apply?
    _ = n := by simp [Nat.cast_inj]
  exact h₃
```

```
import Mathlib

open Set Nat Function

-- fun n ↦ 2 ^ (n + 1)
/--
Let n be a positive integer. Find the number of pairs P, Q of polynomials with real coefficients such that
\[
(P(X))^2 + (Q(X))^2 = X^{2n} + 1
\]
and deg P > deg Q.
-/
theorem putnam_2007_b4
(n : ℕ) (npos : n > 0) :
({(P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^ 2 + Q ^ 2 = Polynomial.X ^ (2 * n) + 1 ∧ P.degree >
    Q.degree}.ncard = ((fun n ↦ 2 ^ (n + 1)) : ℕ → ℕ ) n) := by
  have h₁ : ∀ n : ℕ, n > 0 → ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^ 2 + Q ^ 2 = Polynomial.X ^ (2 *
    n) + 1 ∧ P.degree > Q.degree }.ncard : ℕ) = 2 ^ (n + 1) := by
  intro n hn
  have h : ∀ n : ℕ, n > 0 → ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^ 2 + Q ^ 2 = Polynomial.X ^ (2 * n)
    + 1 ∧ P.degree > Q.degree }.ncard : ℕ) = 2 ^ (n + 1) := by
    intro n hn
    have h₁ : ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^ 2 + Q ^ 2 = Polynomial.X ^ (2 * n) + 1 ∧
    P.degree > Q.degree }.ncard : ℕ) = 2 ^ (n + 1) := by
    have h₂ : ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^ 2 + Q ^ 2 = Polynomial.X ^ (2 * n) + 1 ∧
    P.degree > Q.degree } : Set ((Polynomial ℝ) × (Polynomial ℝ))).ncard = 2 ^ (n + 1) := by
      rw [← Cardinal.natCast_inj]
      apply?
    exact_mod_cast h₂
    exact h₁
  exact h n hn

  have h₂ : ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^ 2 + Q ^ 2 = Polynomial.X ^ (2 * n) + 1 ∧ P.degree
    > Q.degree }.ncard : ℕ) = 2 ^ (n + 1) := by
  apply h₁
  exact npos
  simpa [h₂] using h₂
```

C. Revision to MiniF2F

1. `mathd_algebra_247`:

```
/-- Let t = 2s − s^2 and s = n^2 − 2^n + 1. What is the value of t when n = 3? Show that it is 0.-/
theorem mathd_algebra_247 (t s : ℝ) (n : ℤ) (h₀ : t = 2 * s − s ^ 2) (h₁ : s = n ^ 2 − 2 ^ n + 1)
  (n) (_ : n = 3) : t = 0 := by
 sorry
-- revise to
theorem mathd_algebra_247 (t s : ℝ) (n : ℤ) (h₀ : t = 2 * s − s ^ 2) (h₁ : s = n ^ 2 − 2 ^ n + 1)
```

calc a

```
obtain ⟨rri, cri, hrinj, hcrij, hsub⟩ := Ssub have h1 :
a ≤ n := hab.2.1 have h2 : b ≤ n := hab.2.2.2 have h3 : a *
b ≤ n := by * b = Cardinal.toNat (a * b) := by simp [Nat.
cast_inj] _ ≤ Cardinal.toNat (n) := by? _ = n := by simp [
Nat.cast_inj] apply exact h3
```

```
import Mathlib

open Set Nat Function

-- fun n ↦ 2 ^ (n + 1)
/-- 设 n 为正整数。求满足以下条件的多项式对 P 和 Q 的个数：\[ (P(X))^2 + (Q(X))^2 = X^{2n} + 1 \] 且 deg P > = deg
Q。 -/ theorem putnam_2007_b4 : ℕ) (npos : n > 0) : ({(P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^2 + Q ^2
= Polynomial.X ^ (2 * n) + 1 ∧ P.degree > Q.degree}.ncard = ((fun n ↦ 2 ^ (n + 1)) : ℕ → ℕ ) n) := by have h
1 : ∀ n : ℕ, n > 0 → ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^2 + Q ^2 = Polynomial.X ^ (2 * n) + 1 ∧ P.
degree > Q.degree }.ncard : ℕ) = 2 ^ (n + 1) := by intro n hn : ∀ n : ℕ, n > 0 → ({ (P, Q) : (Polynomial ℝ)
× (Polynomial ℝ) | P ^2 + Q ^2 = Polynomial.X ^ (2 * n) + 1 ∧ P.degree > Q.degree }.ncard : ℕ) = 2 ^ (n + 1) :
= by intro n hn1 : ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^2 + Q ^2 = Polynomial.X ^ (2 * n) + 1 ∧ P.
degree > Q.degree }.ncard : ℕ) = 2 ^ (n + 1) := by have h2 : ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^2
+ Q ^2 = Polynomial.X ^ (2 * n) + 1 ∧ P.degree > Q.degree } : Set ((Polynomial ℝ) × (Polynomial ℝ))).
ncard = 2 ^ (n + 1) := by rw [← Cardinal.natCast_inj] apply? exact_mod_cast h2 exact h1 exact h n hn2
: ({ (P, Q) : (Polynomial ℝ) × (Polynomial ℝ) | P ^2 + Q ^2 = Polynomial.X ^ (2 * n) + 1 ∧ P.degree > Q.degree
}.ncard : ℕ) = 2 ^ (n + 1) := by apply h1 exact npos [h2] using h2 have h
simpa
```

C. 对MiniF2F的修订

1. `mathd_algebra_247`:

```
/-- 让 t = 2s - 2 和 s = n^2 - 2^n + 1。当 n = 为 3 时，t 的值是多少？证明它是 0。-/
theorem mathd_algebra_247 (t s : ℝ) (n : ℤ) (h0 : t = 2 * s - s ^2) (h1 : s = n ^2 - 2
^ n + 1) (n) (_ : n = 3) : t = 0 := by sorry
-- revised
theorem mathd_algebra_247 (t s : ℝ) (n : ℤ) (h0 : t = 2 * s - s ^2) (h1 :
s = n ^2 - 2 ^ n + 1)
```

```
  (_ : n = 3) : t = 0 := by
 sorry
```

2. `induction_sum_odd`:

```
/-- Show that for positive integer n, ∑ k in range n, (2 * k + 1) = n^2.-/
theorem induction_sum_odd (n : ℕ) : (∑ k in Finset.range n, 2 * k) + 1 = n ^ 2 := by
 sorry
-- revise to
theorem induction_sum_odd (n : ℕ) : (∑ k in Finset.range n, (2 * k + 1)) = n ^ 2 := by
 sorry
```

3. `induction_prod1p1onk3le3m1onn`:

```
/-- Show that for any positive integer n, we have ∏ k in Icc 1 n, (1 + 1/k^3) ≤ 3 - 1/n.-/
theorem induction_prod1p1onk3le3m1onn (n : ℕ) (h₀ : 0 < n) :
  (∏ k in Finset.Icc 1 n, 1 + (1 : ℝ) / k ^ 3) ≤ (3 : ℝ) - 1 / ↑n := by
 sorry
-- revise to
theorem induction_prod1p1onk3le3m1onn (n : ℕ) (h₀ : 0 < n) :
  (∏ k in Finset.Icc 1 n, (1 + (1 : ℝ) / k ^ 3)) ≤ (3 : ℝ) - 1 / ↑n := by
 sorry
```

```
 (_ : n = 3) : t = 0 := by
sorry
```

2. induction_sum_odd:

```
/-- 证明对于正整数 n，有 ∑ k=0 to n-1 (2k + 1) = n^2. -/theorem induction_sum_odd (n : ℕ)
: (∑ k in Finset.range n, 2 * k) + 1 = n ^2 := bytheorem induction_sum_odd (n : ℕ) : (
∑ k in Finset.range n, (2 * k + 1)) = n ^2 := by sorry
```

3. induction_prod1p1onk3le3m1onn:

```
/-- 证明对于任何正整数 n，我们有 ∏ k=1 to n (1 + 1/k^3) ≤ 3 - 1/n.-/
theorem induction_prod1p1onk3le3m1onn (n : ℕ) (h₀ : 0 < n) : (
∏ k in Finset.Icc 1 n, 1 + (1 : ℝ) / k ^3) ≤ (3 : ℝ) - 1 / ↑n := by
theorem induction_prod1p1onk3le3m1onn (n : ℕ) (h₀ : 0 < n) : (
∏ k in Finset.Icc 1 n, (1 + (1 : ℝ) / k ^3)) ≤ (3 : ℝ) - 1 / ↑n := by sorry
```