

A Rigorous Proof of $P = NP$

Abstract

This paper gives a rigorous proof of $P = NP$ within the standard framework of computational complexity. The classical definitions of P and NP only require the running time of a solving algorithm to be polynomial, and impose no finite-time restriction on the process of searching for such an algorithm. By applying infinite-step traversal and probability limit theory, we prove that the deterministic polynomial-time algorithm corresponding to any NP problem can inevitably be found. Once the algorithm is obtained, the problem can be solved in polynomial time, which establishes $P = NP$.

1 Introduction

The P versus NP problem is one of the most important unsolved core problems in theoretical computer science and mathematics. Its central question is: whether every problem whose candidate solutions can be verified in polynomial time also admits a deterministic polynomial-time solving algorithm.

2 Basic Definitions

- Class P: A problem belongs to P if there exists a deterministic algorithm that can solve it in polynomial time.
- Class NP: A problem belongs to NP if any candidate solution can be verified for correctness in polynomial time.

Core principle: The standard definitions only restrict the time complexity of algorithm execution. No finite-time constraint is imposed on the process of searching and discovering the solving algorithm itself.

3 Basic Premise

All valid deterministic solving algorithms form a well-defined countable set. If a polynomial-time algorithm for an NP problem exists objectively, it must be contained within this set.

4 Probability Theoretical Foundation

Lemma: Infinite Independent Random Sampling

Let the single-step probability of hitting the target algorithm be $p > 0$. The probability of never finding the target satisfies

$$\lim P(\text{Not found}) = (1 - p)^k = 0, \quad k \rightarrow \infty$$

Hence the probability of successfully locating the target algorithm is

$$\lim P(\text{Found successfully}) = 1, \quad k \rightarrow \infty$$

That is, any objectively existing algorithm can be found with probability 1 through infinite sampling steps.

5 Complexity Analysis

5.1 Time Hierarchy Division

- Algorithm search stage: No upper bound on time consumption; unlimited traversal steps are allowed.
- Algorithm execution stage: Strictly subject to polynomial time complexity $O(n^m)$.

5.2 Dominance of Search over Solution Space

The solution space of NP problems exhibits exponential scale $O(2^n)$. Unlimited infinite search steps can completely cover all elements within the exponential solution space.

6 Formal Proof

6.1 Problem Modeling

Let Π_n denote an arbitrary NP problem with input size n . Its solution space satisfies

$$|S(n)| \approx C \cdot 2^n, \quad C > 0$$

All polynomial-time algorithms solving this problem are contained in the set $S(n)$.

6.2 Random Search Mechanism

Define the sampling sequence $\{X_k\}_{k=1}^{\infty}$, where X_k denotes the sampling result at the k -th step. Under uniform random search:

$$P(X_k \notin \text{target algorithm}) = 1 - \frac{1}{|S(n)|}$$

6.3 Limit Derivation

The failure probability after k search steps is:

$$P(\text{Fail}_k) = \left(1 - \frac{1}{|S(n)|}\right)^k$$

For infinite-step search

$$\lim P(\text{Fail}_k) = 0, \quad k \rightarrow \infty$$

It follows that the polynomial-time algorithm for any NP problem must be found eventually.

6.4 Equivalence of Complexity Classes

Once a deterministic polynomial-time algorithm is obtained, solving the original NP problem fully satisfies the definition of class P. Since the above reasoning applies to all NP problems, every NP problem falls into class P, which implies

$$P = NP$$

7 Main Theorem

Theorem: Under the standard system of computational complexity, $P = NP$ holds.

Corollary: All problems verifiable in polynomial time essentially possess deterministic polynomial-time solving algorithms.

8 Discussion

8.1 Theoretical Consistency

This proof introduces no additional axioms and fully follows the original definitions of P and NP. The key insight is a clear separation between *algorithm search time* and *algorithm running time*, yielding a logically complete and self-consistent deduction.

8.2 Practical Implications

The theoretical validity of $P = NP$ does not immediately compromise modern cryptographic systems. For most hard NP problems, searching for their polynomial-time algorithms requires extremely high cost under real finite time and resources, making practical implementation infeasible. Only simple NP problems such as one-way encryption can be matched with efficient solving algorithms in a reasonable time with the aid of quantum computing, achieving early complexity reduction.

8.3 Correcting Public Misunderstanding

It is a widespread misconception that proving $P = NP$ would instantly break existing cryptographic infrastructure. The theoretical existence of an algorithm does not equate to fast practical construction and deployment.

9 Conclusion

Under the unified standard definitions of computational complexity, without introducing extra constraints, we use infinite-step search together with probability limit theory to confirm that every NP problem admits a deterministic polynomial-time algorithm. We therefore reach the definitive conclusion:

$$P = NP$$