P = NP: Proof by Halting Problem

Introduction

The halting problem serves as both a classical decision problem and a logical tool for identifying contradictions. In this proof, we'll show that the statement:

"Polynomial algorithms solving NP-complete (NP-c) problems are incomputable"

is a contradiction. By resolving this contradiction, we conclude that P = NP.

Definition

A problem is incomputable if and only if it is equivalent to the halting problem.

First Step: Halting Problem and Space Requirement

This step outlines how computation space relates to incomputability.

Consider an array with n elements, and the task is to determine if a particular integer x exists within it.

Skipping one step (i.e., checking only n-1 elements) introduces an incomputable instance, as it leaves infinite possibilities unaccounted for in the missing step.

Therefore, we can express the halting problem as:

“Solve out of infinity.”

This captures the essence of incomputability: trying to determine something fully over an infinite space.

Second Step: Space Requirements for Incomputable Functions

We now classify functions that are incomputable. An algorithm becomes incomputable if:

1. It runs with insufficient space for input-output (I/O) operations.

2. It behaves like the halting problem—running endlessly without resolution.

Example in NP-c Problems:

If an algorithm reads only O(log n) steps, it cannot solve an NP-c problem because it must process the entire input size n.

Similarly, an algorithm that runs endlessly without halting is incomputable by nature.

Third Step: Contradiction Using Polynomial Algorithms

Now, let’s address the key statement:

"Polynomial algorithms solve NP-c problems."

A polynomial-time algorithm, such as one with time complexity O(n^k), can read the entire input n and meet the space requirements.

However, to fully solve an NP-c problem, the algorithm would need to "solve out of 2^n" possibilities (the exponential search space).

This gives us the following inequality:

“Solve out of infinity” ≠ “Solve out of 2^n.”

The two are fundamentally different: solving an infinite space is incomputable, but solving an exponential space (2^n) is not.

Thus, the claim that “Polynomial algorithms solving NP-c problems are incomputable” is a contradiction. Polynomial algorithms are computable, and the requirements to solve NP-c problems do not imply incomputability.

Conclusion

We’ve demonstrated that the assumption “Polynomial algorithms solving NP-c problems are incomputable” leads to a logical contradiction. Polynomial algorithms can meet the space and runtime requirements necessary to solve NP-c problems, proving that:

P = NP.