Explanation of P, NP, NP-Complete, and the P vs NP Problem

111110542 林源茂

1. Class P (Polynomial Time)

- **Definition**: Class P consists of all decision problems (yes/no problems) that can be solved by a deterministic Turing machine in **polynomial time**.
  - Example: Sorting a list, finding the shortest path in a graph (Dijkstra's algorithm), and testing whether a number is prime.
- **Importance**: Problems in P are considered **efficiently solvable**, meaning their solutions can be computed within a reasonable amount of time as the size of the input increases.

Example:

- Problem: "Is this number divisible by 3?"
- Algorithm: Check divisibility using modular arithmetic, which is a polynomial-time algorithm.

2. Class NP (Nondeterministic Polynomial Time)

- **Definition**: Class NP consists of all decision problems for which a given solution can be **verified** in polynomial time by a deterministic Turing machine.
  - Example: Sudoku verification, the Traveling Salesman Problem (TSP), and the Boolean satisfiability problem (SAT).
- **Nondeterministic Meaning**: Imagine a "guessing machine" that can guess a solution and then verify it in polynomial time.

Key Difference Between P and NP:

- In P, the solution can be **found** in polynomial time.
- In NP, the solution can be **verified** in polynomial time, but finding the solution might take longer.

Example:

- Problem: "Is there a path through a graph that visits each node exactly once (Hamiltonian Path)?"
- Verification: If given a specific path, you can verify in polynomial time whether it satisfies the problem's conditions.

3. NP-Complete

- **Definition**: NP-Complete problems are a subset of NP problems that are the **"hardest" in NP**. If any NP-Complete problem can be solved in polynomial time, then **all NP problems can also be solved in polynomial time**.
    - NP-Complete problems are **both**:
        1. In NP (solution can be verified in polynomial time).
        2. NP-Hard (as hard as the hardest problems in NP).
- **Importance**: These problems are often used to explore the limits of computational efficiency.

Example:

- The **Boolean satisfiability problem (SAT)** was the first proven NP-Complete problem (Cook, 1971).
- Other NP-Complete problems:
    - Traveling Salesman Problem (TSP)
    - Knapsack Problem
    - Vertex Cover Problem

4. P vs NP Problem

- **Question**: Is P equal to NP? In other words:
    - Can every problem whose solution can be **verified** in polynomial time (NP) also be **solved** in polynomial time (P)?
- **Implications**:
    - If P = NP: Every problem that can be verified in polynomial time could also be solved efficiently, revolutionizing fields like cryptography, optimization, and artificial intelligence.
    - If P ≠ NP: Some problems are inherently harder to solve than to verify.

Current Status:

- The question remains **unsolved** and is one of the seven **Millennium Prize Problems**, with a $1 million reward for a correct solution (Clay Mathematics Institute, 2000).

Relationships Between P, NP, and NP-Complete

1. **P ⊆ NP**: Any problem that can be solved in polynomial time can also be verified in polynomial time.
2. If P = NP, then P = NP-Complete (all NP-Complete problems become solvable efficiently).
3. If P ≠ NP, then NP-Complete problems are harder than P problems and cannot be solved in polynomial time.