|  |  |
| --- | --- |
| P Versus Np Problems  Design and Analysis of Algorithm | CONTENTS   * Introduction * P vs NP problem * NP Hard & NP Complete * P=NP * Difference between P and NP * Real World Examples   Abhishek Kumar Suman    CSE -2nd Year |

**Introduction**

**What is P vs NP Problem?**

The P versus NP problem is to determine whether every language accepted by some non-deterministic algorithm in polynomial time is also accepted by some deterministic algorithm in polynomial time.

**What Does ‘P’ stand for?**

P (stands for Deterministic Polynomial Time), is the class of problems for which we have an efficient algorithm that is capable of producing a solution in polynomial time.

Things like searching a list, sorting a list, multiplication of matrices, balancing trees, etc are such type of problems, for which we have an efficient algorithm.

All such problems can be solved by a deterministic Turing machine within a polynomial amount of computation time.

**What does the term deterministic mean?**

When we know each and every statement of the algorithm, clearly and their working, we call it deterministic.

For instance, multiplication of two numbers, we can verify the result and solve it easily.

**What is ‘NP’?**

NP (stands for Non-Deterministic Polynomial Time), is the set of decision problems that are solvable in polynomial time by a Non-deterministic Turing Machine.

NP problems do not have a know algorithm that can produce a result in polynomial time. If we are given a solution to an NP problem, verifying that it is correct is easy and can be done in polynomial time or less. They are easy to check but hard to solve, they generally have an exponential time complexity such as O(nn).

Example: Solving a Sudoku Puzzle, unlocking a Smartphone, Sending messages over apps, etc

Problems like: -

0/1 Knapsack, Travelling Salesman, Sum of subsets. Graph Colouring. Hamiltonian Cycle, Finding Prime Factors of a large number take exponential time to solve.

* P is a subset of NP problems, as illustrated in Fig.1

.

**Figure 1**: P is a subset of NP.

**NP Hard** :-

A problem X is NP-hard if every problem Y ∈ NP **reduces** to X i.e. X is at least as hard to solve as every problem in NP (If P! = NP, then X does not belong to P)

**NP-Complete** :-

A problem L is NP-complete if :-

1. L is in P, and
2. an algorithm which solves L can be used to solve any problem L' in NP; that is, given an instance of L' we can create an instance of L that has a solution iff the instance of L' has a solution.

Formally speaking, every problem L' in NP is reducible to L.

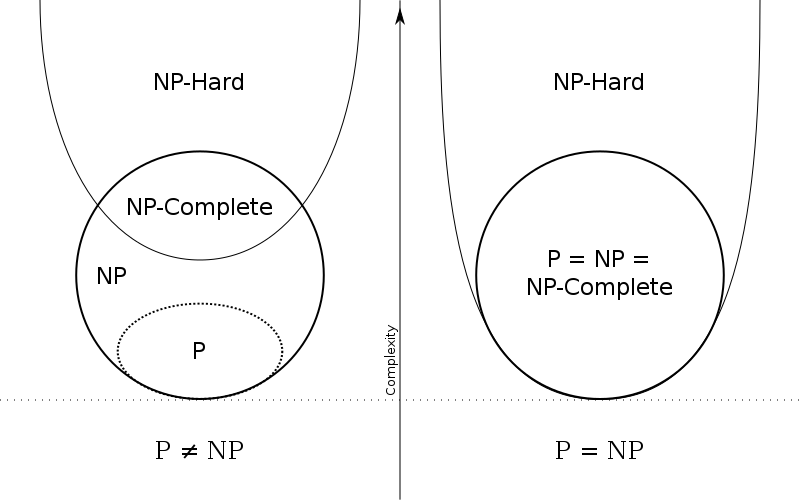


Figure 2 : Under the assumption that P≠NP, the existence of problems within NP but outside both P and NP-complete was established by Ladner

**DIFFERENCE BETWEEN P & NP**

Let’s take the example of adding two numbers, when the digits are less it is easy for us to compute, but the larger the numbers get the harder the computation becomes for us humans. But the computers can add larger numbers in a fairly simple amount of polynomial time quickly and efficiently. Such type of problems which can be solved by a computer are known as **P problems.**

Now let’s take a look at a little complex problem like prime factorization. We know that, every composite number can be expressed as a product of two or more prime factors. Our normal computers can handle this problem within seconds for numbers up to a billion. But as the numbers grow this problem becomes lot harder even for the fastest of computers. So, this is not solvable in Polynomial time.

But what if I provide the factors? Can we verify the solution? Like if I say, 50 can be factorized as 2\*5\*5. We can easily verify it by multiplying 2,5 and 5 real quick and find that it indeed produces 50.

In fact, multiplication is a P problem. Computers can multiply fairly large numbers in no time. So, factorization is not solvable in polynomial time but the solution is verifiable in polynomial time. These type of problems are known as **NP problems.**

**So From all of the things we have grasped, the differences are:-**

|  |  |  |
| --- | --- | --- |
| S No. | P Problems | NP Problems |
| 1 | These can be solved in polynomial time by deterministic algorithms. | These can be solved in non-deterministic polynomial time. |
| 2 | Such problems can be solved and verified in polynomial time. | NP problems solution cannot be obtained in polynomial time but if solution is given it can be verified in polynomial time. |
| 3 | P problems are subset of NP problems. | NP problems are a superset of P problems. |
| 4 | Eg. Searching, Sorting, Addition, Multiplication,etc. | Eg. Sudoku Puzzle ,Travelling Salesman Problem,etc. |

**Real World Examples**

* Suppose that a university is organizing accommodations in hostel for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided a list of pairs of incompatible students, and requested that no pair from this list appear in the final choice.
* A farmer wants to take 200 watermelons of different masses to the market. He needs to pack the watermelons into boxes. Each box can only hold 30 kilograms without breaking. The farmer needs to know if 10 boxes will be enough for her to carry all 200 watermelons to market.
* The vintage Super Mario game by Nintendo Inc. is itself a NP-hard.