

DAA - P & NP Class

In Computer Science, many problems are solved where the objective is to maximize or minimize some values, whereas in other problems we try to find whether there is a solution or not. Hence, the problems can be categorized as follows –

Optimization Problem

Optimization problems are those for which the objective is to maximize or minimize some values. For example,

- Finding the minimum number of colors needed to color a given graph.
- Finding the shortest path between two vertices in a graph.

Decision Problem

There are many problems for which the answer is a Yes or a No. These types of problems are known as **decision problems**. For example,

- Whether a given graph can be colored by only 4-colors.
- Finding Hamiltonian cycle in a graph is not a decision problem, whereas checking a graph is Hamiltonian or not is a decision problem.

What is Language?

Every decision problem can have only two answers, yes or no. Hence, a decision problem may belong to a language if it provides an answer 'yes' for a specific input. A language is the totality of inputs for which the answer is Yes. Most of the algorithms discussed in the previous chapters are **polynomial time algorithms**.

For input size n , if worst-case time complexity of an algorithm is $O(n^k)$, where k is a constant, the algorithm is a polynomial time algorithm.

Algorithms such as Matrix Chain Multiplication, Single Source Shortest Path, All Pair Shortest Path, Minimum Spanning Tree, etc. run in polynomial time. However there are many problems, such as traveling salesperson, optimal graph coloring, Hamiltonian cycles, finding the longest path in a graph, and satisfying a Boolean formula, for which no polynomial time algorithms is known. These problems belong to an interesting class of problems, called the **NP-Complete** problems, whose status is unknown.

In this context, we can categorize the problems as follows –

P-Class

The class P consists of those problems that are solvable in polynomial time, i.e. these problems can be solved in time $O(n^k)$ in worst-case, where k is constant.

These problems are called **tractable**, while others are called **intractable or superpolynomial**.

Formally, an algorithm is polynomial time algorithm, if there exists a polynomial $p(n)$ such that the algorithm can solve any instance of size n in a time $O(p(n))$.

Problem requiring $\Omega(n^{50})$ time to solve are essentially intractable for large n . Most known polynomial time algorithm run in time $O(n^k)$ for fairly low value of k .

The advantages in considering the class of polynomial-time algorithms is that all reasonable **deterministic single processor model of computation** can be simulated on each other with at most a polynomial slow-d

NP-Class

The class NP consists of those problems that are verifiable in polynomial time. NP is the class of decision problems for which it is easy to check the correctness of a claimed answer, with the aid of a little extra information. Hence, we aren't asking for a way to find a solution, but only to verify that an alleged solution really is correct.

Every problem in this class can be solved in exponential time using exhaustive search.

P versus NP

Every decision problem that is solvable by a deterministic polynomial time algorithm is also solvable by a polynomial time non-deterministic algorithm.

All problems in P can be solved with polynomial time algorithms, whereas all problems in $NP - P$ are intractable.

It is not known whether $P = NP$. However, many problems are known in NP with the property that if they belong to P, then it can be proved that $P = NP$.

If $P \neq NP$, there are problems in NP that are neither in P nor in NP-Complete.

The problem belongs to class **P** if it's easy to find a solution for the problem. The problem belongs to **NP**, if it's easy to check a solution that may have been very tedious to find.