P-NP PROBLEM

In Computer Science, generally the problems can be categorized as follows −

1. 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.

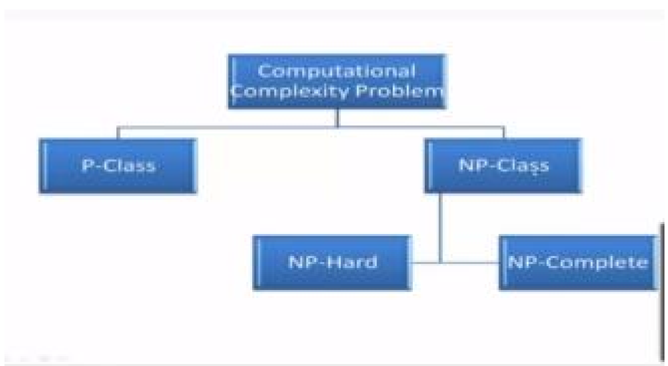
Finding the shortest path between two vertices in a graph.

2. 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.

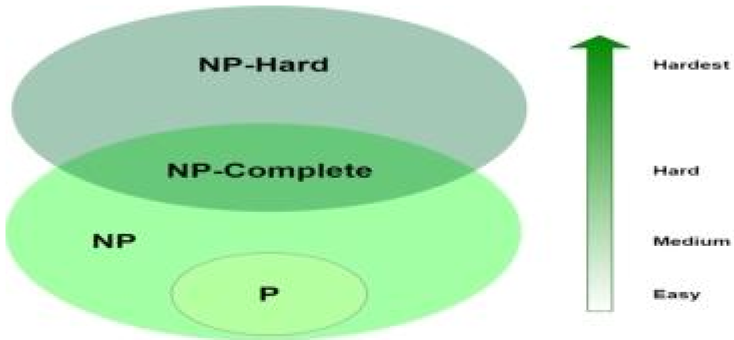
Finding Hamiltonian cycle in a graph is not a decision problem, whereas checking a graph is Hamiltonian or not is a decision problem.

In theoretical computer science, the classification and complexity of common problem definitions have two major sets; $\mathcal{P}$ which is "Polynomial" time and $\mathcal{NP}$ which "Non-deterministic Polynomial" time. There are also $\mathcal{NP}$-$Hard$ and $\mathcal{NP}$-$Complete$ sets, which we use to express more sophisticated problems.



In the case of rating from easy:

- Easy $\rightarrow \mathcal{P}$
- Medium -NP
- Hard $\rightarrow \mathcal{NP}$-$Complete$
- Hardest->NP-hard



Using the diagram, we assume that $\mathcal{P}$ and $\mathcal{NP}$ are not the same set, or, in other words, we assume that $\mathcal{P} \neq \mathcal{NP}$. This is our apparently-true, but yet-unproven assertion. Of course, another interesting aspect of this diagram is that we've got some overlap between $\mathcal{NP}$ and $\mathcal{NP}\text{-}Hard$. We call $\mathcal{NP}\text{-}Complete$ when the problem belongs to both of these sets.

P-Class : *o i.e. these problems can be solved in time $O(n^k)$, where k is constant and n is the size of the input to the problem.* **These problems can be solved and verified in polynomial time.**

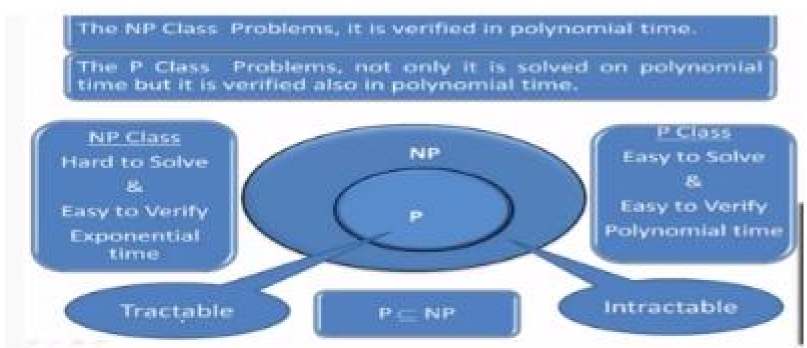
Example: Linear search,Binary search, insertion sort,merge sort

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))***.Most known polynomial time algorithm run in time $O(n^k)$ for fairly low value of ***k***.

NP-Class: A problem that can not be solved in polynomial time but solutoins can be verified in polynimial time using non-deterministic algorithm is known as NP(non-deterministic polynomial)algorithm. *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.*

Example: Traveling salesman problem,Sudoku Problem, Scheduling Problem.

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



P versus NP

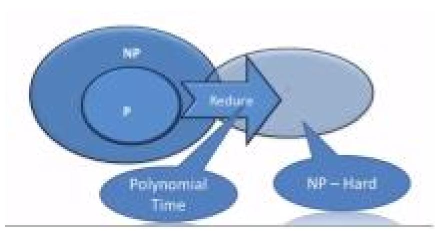
Frequently called the most important outstanding question in theoretical computer science, the equivalency of P and NP,there is a million dollars for proving — or disproving. Roughly speaking, **"P"-class problems are "easy" for computers to solve; that is, solutions to these problems can be computed in a reasonable amount of time compared to the complexity of the problem. Meanwhile, for "NP" problems, a solution might be very hard to find–perhaps requiring billions of years' worth of computation–but once found, it is easily checked.**

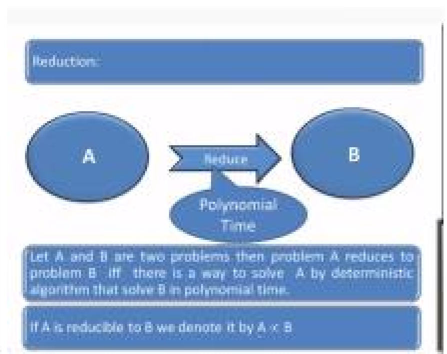
All problems in P can be solved with polynomial time algorithms, whereas all problems in *NP* can not be solved in polynomial time but solutins can be verified in polynomial time using non-deterministic algorithm.The problem belongs to class **P** if it's easy to find a solution for the problem. The problem belongs to **NP**, if it's easy to check a solution that may have been very tedious to find.

NP hard

A problem is NP-hard if the problem can be reduced using Polynomial time. Let A and B are two problems then problem A reduces to problem B iff there is a way to solve A by deterministic algorithm that solve B in polynomial time.

Example:Subset sum problem.



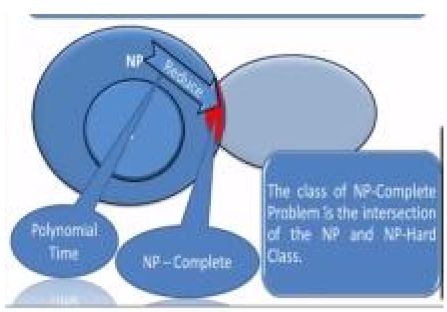


If A is reducable to B and B in P then A is in P
A is not in P implies B is not in P

NP complete

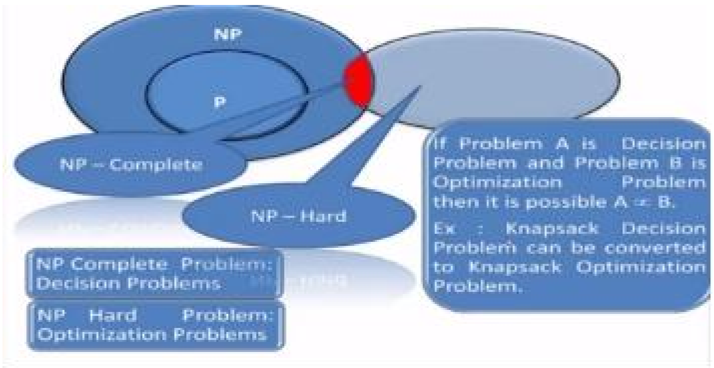
A problem is NP complete problem if it is in NP-hard and also in NP.It is quite likely to contain only intractable problem.The key notion behind the theory of NP-complete problems is the nondeterministic Turning machine.

Example: Graph coloring problem,Hamiltonian path problem,Longest path problem.



All NP -complete problems are NP- hard but all NP-hard problem is not NP complete.

- All NP complete problem is easy to verify but difficult to solve.
- All NP hard problem is difficult to solve and hard to verify.
- All NP problems those are reduced to another problem for solving it in polynomial time are called NP-hard problems; those NP -hard problems are easy to verify but difficult to solve are called NP-complete problems.
- All problems in NP-hard cannot be verified in polynomial time .A non-deterministic Turing machine can verify NP-Complete problem in polynomial time.
- A Problem X is *NP-Hard* if there is an NP-Complete problem Y, such that Y is reducible to X in polynomial time.A problem X is NP-Complete if there is an NP problem Y, such that Y is reducible to X in polynomial time.


NP
P
NP – Complete
NP – Hard
If Problem A is Decision Problem and Problem B is Optimization Problem then it is possible A ∝ B.
Ex : Knapsack Decision Problem can be converted to Knapsack Optimization Problem.
NP Complete Problem: Decision Problems
NP Hard Problem: Optimization Problems