DESIGN & ANALYSIS OF

ALGORITHMS ( CSB -252 )

ASSIGNMENT -5

BY-Ankit kr Rouniyar

Roll-181210011

Cse 2nd year.

**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(nk)*** 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 ***Ω(n50)*** time to solve are essentially intractable for large ***n***. Most known polynomial time algorithm run in time ***O(nk)*** 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.

**NP – Hard:**

This is the class of problems which are at least as hard as the hardest problems in NP. Problems belonging to this class may or may not be part of NP, that is, the hardest problems of NP belong to the intersection of NP and NP-Hard. Problems in NP-Hard may not even be decision problems.

Example of a problem which is NP-Hard but not NP is the problem of identifying a chess move in any given board state that is the best possible move to make.

**NP – Complete:**

This is the class problems which contains the set of all the hardest problems in NP. Every problem in NP-Complete must belong to NP and NP-Hard, which is not true for NP-Hard. NP-Complete is the intersection of NP and NP-Hard.

Example of a problem which is NP-Complete is the clique graph problem, where, in an undirected graph, the largest complete sub-graph is to be found.

P versus NP

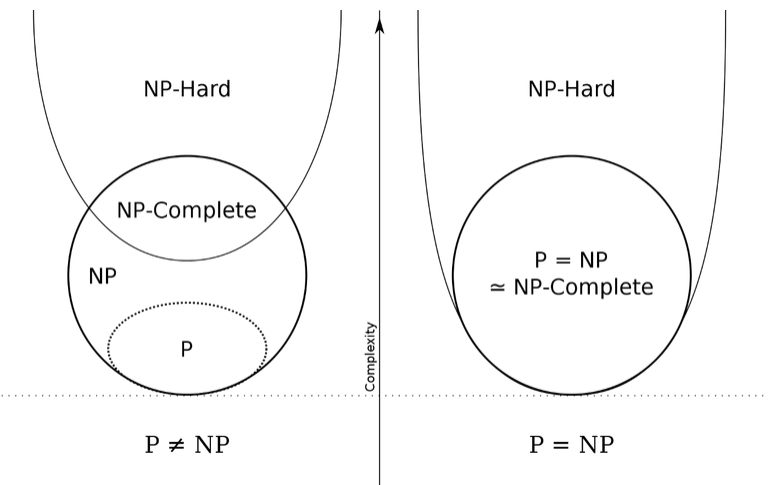
Every decision problem that is solvable by a deterministic polynomial time algorithm is also solvable by a polynomial time non-deterministic algorithm.

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 ≠ 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.



**Consequences:-**

**What happens if P does in fact, equal to NP?**

Now that we have defined all these different terms, a more important question to ask is — “Why does all this even matter?”. Having a solution to this problem can have profound consequences not only in academia, but also in practical situations. This includes:

* **Cryptography**— From the many passwords we maintain to the PIN number we use for the ATM, we rely on these codes to govern our everyday lives because they are easy to check but hard for people to crack.
* **Vehicle Routing** — Transportation and the movement of logistics would be optimized across the world, impacting many industries from transport to e-commerce to manufacturing.
* **Protein-Folding**-- Understanding or predicting how a given sequence of amino acids (a protein) will fold up to form its 3D shape will help in many areas such as drug design and perhaps even cure cancer.