Submitted by: deepika Kumari

(181210019)

**Difference between N and PN problem**

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.

NP-Class

* Every problem in this class can be solved in exponential time using exhaustive search.
* 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
* Here, we aren’t asking for a way to find a solution, but only to verify that an alleged solution really is correct.

## 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.

**Examples**

1. **P**

Given a connected graph G, can its vertices be colored using two colors so that no edge is monochromatic?

Algorithm: start with an arbitrary vertex, color it red and all of its neighbours blue and continue. Stop when you run out of vertices or you are forced to make an edge have both of its endpoints be the same color.

1. **NP**

Integer factorisation is in NP. This is the problem that given integers n and m, is there an integer f with 1 < f < m, such that f divides n (f is a small factor of n)?

This is a decision problem because the answers are yes or no. If someone hands us an instance of the problem (so they hand us integers n and m) and an integer f with 1 < f < m, and claim that f is a factor of n (the certificate), we can check the answer in polynomial time by performing the division n / f.

# **NP-Complete**

* NP-Complete is a complexity class which represents the set of all problems *X* in NP for which it is possible to reduce any other NP problem *Y* to *X* in polynomial time.
* Intuitively this means that we can solve Y quickly if we know how to solve X quickly. Precisely, Y is reducible to X, if there is a polynomial time algorithm f to transform instances y of Y to instances x = f(y) of X in polynomial time, with the property that the answer to y is yes, if and only if the answer to f(y) is yes.

# **NP-Hard**

* a problem *X* is NP-hard, if there is an NP-complete problem *Y*, such that *Y* is reducible to *X* in polynomial time