CLASSIFICATION OF PROBLEMS: P, NP, NP-Complete & NP-Hard

What is polynomial?

- A polynomial is a mathematical expression consisting of a sum of terms, each term including a variable or variables raised to a power and multiplied by a coefficient.
- The time $O(n^k)$, for some constant k, where n is the size of the input to the problem, is called polynomial time.

Polynomial Time: $O(n)$, $O(n^2 + n)$, $O(10^{100}n)$

Non-Polynomial Time: $O(e^{2n})$, $O(2^n)$, $O(e^{n/1000})$

CLASSIFICATION OF PROBLEMS

The subject of computational complexity theory is dedicated to classifying problems by how hard they are. There are many classifications, some of the most common and useful are the following:

1. The Class of P problems

- P stands for Polynomial.
- The problems that are **solvable** in polynomial time (that is the time $O(n^k)$, for some constant k, where n is the size of the input to the problem), are called class of **P problems**.

2. The Class of NP problems

- **NP** stands for **Non-deterministic Polynomial.**
- The problems that are **verifiable** in polynomial time, are called class of N**P problems**.
- Here the term verifiable mean is that if we were somehow given a certificate of a solution, then we would verify the certificate is correct in polynomial time.
- In other words, a problem is in NP if we can quickly (in polynomial time) test whether a solution is correct without worrying about how hard it might be to find the solution.
- Problems in NP are still relatively easy, if only we could guess the right solution, we could then quickly test it.
- Any problem in P is also in NP, since if a problem is in P then we can solve it in polynomial time without even being given a certificate. We believe P is a subset of NP.
- An example of a P type problem is finding the way from point A to point B on a map, Traveling salesperson ($O(n^2 2^n)$), knapsack ($O(2^{n/2})$)

3. The Class of NP-Complete & NP-Hard Problems

- A problem A can be **reduced** to another problem B if any instance of A can be rephrased as an instance of B, the solution to the instance of B provides a solution to the instance of A.
- A decision problem X is NP-complete if:
  a) X is in NP
  b) Every problem Y in NP is reducible to X in polynomial time.
  X can be shown to be in NP by demonstrating that a candidate solution to X can be verified in polynomial time.

- In other words, A problem X is NP-Complete if there is an NP problem Y, such that Y is reducible to X in polynomial time. NP-Complete problems are as hard as NP problems.
- Note that **a problem satisfying condition b)** is said to be **NP-hard**, whether or not it satisfies condition a)
- A problem is said to be NP-hard if an algorithm for solving it can be translated into one for solving any other NP-problem. It is much easier to show that a problem is NP than to show that it is NP-hard. **A problem which is both NP and NP-hard is called an NP-complete problem.**
- For example, the problem of solving linear equations in an indeterminate x reduces to the problem of solving quadratic equations. Given an instance $ax + b = 0$, we transform it to $0x^2 + ax + b = 0$, whose solution provides a solution to $ax + b = 0$.
- NP-complete problems are the hardest problems in NP. Formally, problem P is NP-complete if it is in NP and every problem in NP reduces to P.
- If an NP-complete problem were solvable in polynomial time, then every NP problem would be solvable in polynomial time.
- $P \neq NP$ widely believed (but still unproven).
- Example: Satisfiability, Travelling Salesman Problem, and Graph 3-Coloring. Finding the longest path between two vertices is NP-complete, even if the weight of each edge is 1. Determining whether a directed graph has a Hamiltonian cycle is NP-Complete.