NP-Completeness

Polynomial time algorithm: On inputs of size n, their worst case running time is $O(n^k)$ for some constant k. All problems can not be solved in polynomial time. There are problems such as Turing famous "Halting Problem" that can not be solved by any computer, no matter how much time allow. There are problems that can be solved, but not in time $O(n^k)$ for any constant k.

Class P: It contains all problems which can be solved by deterministic algorithms in polynomial time, that is in time $O(n^k)$ for some constant k, where n is the size of the input to the problem. Examples: Ordered search, Searching.

Class NP: The abbreviation NP refers to "Nondeterministic Polynomial time". The class NP consists of those problems that are solvable by nondeterministic algorithms in polynomial time and verifiable by deterministic algorithms in polynomial time. Any problem in P is also in NP. Examples: Subset Sum, TSP.

NP-Hard: Informally, NP-Hard problems are a class of decision problems that contains the problems that are at least as "hard" as any problem in NP, even though they may not be in NP themselves. NP-Hard problems are NP-Complete or "harder". If an NP-Hard problem can be solved in polynomial time, then all NP-Complete problems can be solved in polynomial time. All NP-Complete problems are NP-Hard, but some NP-Hard problems are not known to be NP-Complete. Examples: Subset Sum, TSP, Halting problem.

NP-Complete: Interesting class of problems whose status is unknown. No polynomial time algorithm has yet been discovered for an NP-Complete problem, nor has anyone yet been able to prove that no polynomial time algorithm can exist for any of them. A problem is in the class NPC if it is in NP and is as "hard" as any problem in NP. If any NP-Complete problem can be solved in polynomial time, then every problem in NP has a polynomial time algorithm. Examples: Subset Sum, TSP, Knapsack.