Unit 5

Complexity Classes:

The problems whose solutions are not yet found or not solvable in polynomial time , such problems are divided into classes known as complexity classes.

These classes are based on how much time and space they require to solve the problem and verify the solution.

- The time complexity of an algorithm is used to describe the number of steps required to solve a problem, but it can also be used to describe how long it takes to verify the answer.
- The space complexity of an algorithm describes how much memory is required for the algorithm to operate.

Types of Complexity Classes

1. **P Class**
2. **NP Class**
3. **CoNP Class**
4. **NP-hard**
5. **NP-complete**

P Class

The P in the P class stands for **Polynomial Time.** It is the collection of decision problems(problems with a “yes” or “no” answer) that can be solved by a deterministic machine in polynomial time.
Ex: Merge Sort

Characteristics:

- The solution to **P problem**s is easy to find.

- **P** is often a class of computational problems that are solvable and tractable. Tractable means that the problems can be solved in theory as well as in practice. But the problems that can be solved in theory but not in practice are known as intractable.

NP Class

The NP in NP class stands for Non-deterministic Polynomial Time. It is the collection of decision problems that can be solved by a non-deterministic machine in polynomial time.

Characteristics:

- The solutions of the NP class are hard to find since they are being solved by a non-deterministic machine but the solutions are easy to verify.
- Problems of NP can be verified by a Turing machine in polynomial time.

Example: Hamiltonian problem

Graph Coloring

Co-NP Class

Co-NP stands for the complement of NP Class. It means if the answer to a problem in Co-NP is No, then there is proof that can be checked in polynomial time.

Characteristics:

- If a problem X is in NP, then its complement X' is also in CoNP.
- For an NP and CoNP problem, there is no need to verify all the answers at once in polynomial time, there is a need to verify only one particular answer "yes" or "no" in polynomial time for a problem to be in NP or CoNP.

Example:

Prime Number

Integer factorization

NP-Hard Class

An NP-hard problem is at least as hard as the hardest problem in NP and it is a class of problems such that every problem in NP reduces to NP-hard.

Characteristics:

- All NP-hard problems are not in NP.
- It takes a long time to check them. This means if a solution for an NP-hard problem is given then it takes a long time to check whether it is right or not.
- A problem A is in NP-hard if, for every problem L in NP, there exists a polynomial-time reduction from L to A.

Ex: Halting Problem, No Hamiltonian Cycle.

NP-Complete Class

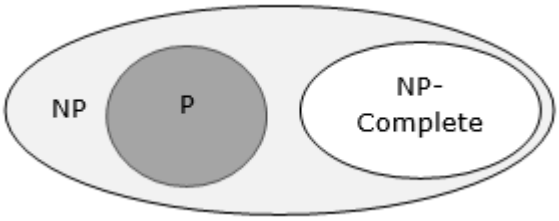
- NP-complete problems are special as any problem in NP class can be transformed or reduced into NP-complete problems in polynomial time.
- If one could solve an NP-complete problem in polynomial time, then one could also solve any NP problem in polynomial time.

Ex.

Hamiltonian Cycle

TSP

The traveling salesman problem consists of a salesman and a set of cities. The salesman has to visit each one of the cities starting from a certain one and returning to the same city. The challenge of the problem is that the traveling salesman wants to minimize the total length of the trip



NP Problem:

The NP problems set of problems whose solutions are hard to find but easy to verify and are solved by Non-deterministic machine in polynomial time.

NP-Hard Problem:

A Problem X is NP-Hard if there is an NP-Complete problem Y, such that Y is reducible to X in polynomial time. NP-Hard problems are as hard as NP-Complete problems. NP-Hard Problem need not be in NP class.
If every problem of NP can be polynomial time reduced to it called as NP Hard.

A lot of times takes the particular problem solve and reducing different problems.

example :

1. Hamiltonian cycle .
2. optimization problem .
3. Shortest path

NP-Complete Problem:

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. A problem is NP-Complete if it is a part of both NP and NP-Hard Problem. A non-deterministic Turing machine can solve NP-Complete problem in polynomial time.

A problem is np-complete when it is both np and np hard combines together.

this means np complete problems can be verified in polynomial time.

Example:

1. Decision problems.

NP-hard	NP-Complete
NP-Hard problems(say X) can be solved if and only if there is a NP-Complete problem(say Y) that can be reducible into X in polynomial time.	NP-Complete problems can be solved by a non-deterministic Algorithm/Turing Machine in polynomial time.
To solve this problem, it do not have to be in NP .	To solve this problem, it must be both NP and NP-hard problems.
Time is unknown in NP-Hard.	Time is known as it is fixed in NP-Hard.
NP-hard is not a decision problem.	NP-Complete is exclusively a decision problem.
Not all NP-hard problems are NP-complete.	All NP-complete problems are NP-hard
Do not have to be a Decision problem.	It is exclusively a Decision problem.
It is optimization problem used.	It is Decision problem used.
Example: Halting problem, Vertex cover problem, etc.	Example: Determine whether a graph has a Hamiltonian cycle, Determine whether a Boolean formula is satisfiable or not, Circuit-satisfiability problem, etc.