LESSON 2

Topic: Intractability II (P, NP, NP-Hard and NP-complete)

1) Introduction (5 minutes)

There are some unsolved problems in computer science, and these difficulties are grouped into classes called complexity classes. A Complexity Class is a group of issues with related complexity in complexity theory. These courses assist scientists in classifying difficulties according to the amount of time and space needed to solve problems and verify the solutions. It is the area of computing theory that deals with the tools needed to solve a problem.

Time and space, or how long an algorithm takes to solve a problem and how much memory it uses, are the two common resources.

The term "time complexity" is used to characterize both the number of steps needed to solve an issue and the amount of time it takes to validate the solution.

How much memory is needed for an algorithm to function is determined by its space complexity.

In order to organize problems of the same kind, complexity classes are helpful.

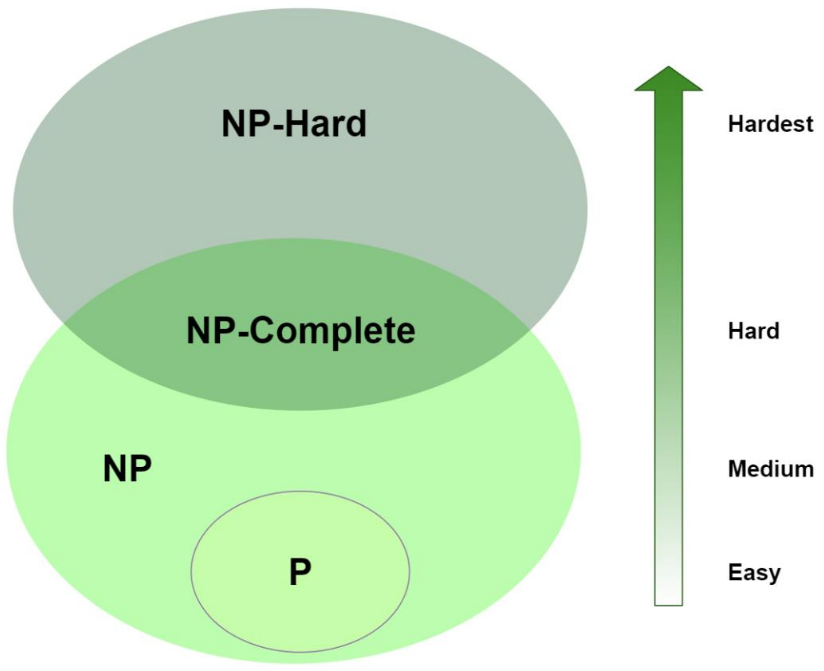
Types of Complexity Classes

1) P Class
2) NP Class
3) NP-hard
4) NP-complete

We can rate them from easy to hard as follows

- Easy - P
- Medium - NP
- Hard - NP Complete
- Hardest - NP Hardest

And we can visualize their relationship, too:



2) P Class Problems (5 minutes)

1. Deterministic algorithms can solve a set of issues known as P problems in a polynomial amount of time.
2. P problems are all decision-making problems that the deterministic Turing machine can answer in polynomial time.
3. They can be solved in a computationally acceptable amount of time for any instance of the problem and are straightforward to verify. These issues are often characterized as "tractable" issues.
4. In the worst scenario, it requires n comparisons to find an element in a list of size n. As the amount of the input rises, the number of comparisons grows linearly. Linear search is a P problem, then.
5. In reality, P problems make up the majority of the issues. A few examples of P problems include searching for an element in an array (O(n)), adding an entry to the end of a linked list (O(n)), sorting data using selection sort (O(n^2)), determining the height of a tree (O(log2n)), sorting data using merge sort (O(nlog2n), and matrix multiplication (O(n^3)).

6. If an algorithm of O(2^n) complexity is tested on a problem of size (n + 1), it takes twice as long. These issues do not fall under class P.
7. Any problems that can't be resolved in a polynomial amount of time are excluded. The knapsack problem cannot be solved in polynomial time with the brute force method. It is not a P problem as a result.
8. Many important problems exist for which a polynomial-time solution has not yet been discovered, nor has it been demonstrated that such a solution does not exist. Examples of such courses include TSP, Graph coloring, Partition Problem, Knapsack, etc.

Examples of P Problems:

- Insertion sort
- Merge sort
- Linear search
- Matrix multiplication
- Finding minimum and maximum elements from the array

3) NP Class Problems (10 minutes)

1. A group of problems known as NP are those that can be resolved in nondeterministic polynomial time. NP stands for Non-Deterministic Polynomial-Time, not non-polynomial.
2. There are two steps to the non-deterministic algorithm's operation.

A) Stage of Nondeterministic (guessing) Some solution string S is generated for input instance I, and it can be viewed as a potential solution.

B) Stage of Deterministic (verification): The deterministic algorithm receives the inputs I and S and returns "Yes" if S is a solution for input instance I.

3. NP-hard problems have a solution that cannot be found in polynomial time, but once found, the answer can be verified in polynomial time.
4. P NP, or all of P's difficulties are included in NP.
5. NP difficulties include the Knapsack problem (O(2n)), Traveling salesman problem (O(n!)), Tower of Hanoi (O(2n-1)), and Hamiltonian cycle (O(n!)).
6. NP-complete and NP-hard categories are used to further categorize NP problems.
7. The hardest problems are those that are NP-hard. Problems that are NP-complete are NP-hard, while the opposite is not true.
8. If NP-hard problems can be resolved in polynomial time, NP-complete problems can as well.

Examples of NP problems:

- Knapsack problem (O(2^n))

- Travelling salesman problem (O(n!))
- Tower of Hanoi ($O(2^n - 1)$)
- Hamiltonian cycle (O(n!))

Differences Between P and NP Problems are as follows:

Sr. No.	P Problems	NP Problems
1.	P problems are set of problems which can be solved in polynomial time by deterministic algorithms.	NP problems are problems which can be solved in nondeterministic polynomial time.
2.	P Problems can be solved and verified in polynomial time	The solution to NP problems cannot be obtained in polynomial time, but if the solution is given, it can be verified in polynomial time.
3.	P problems are a subset of NP problems	NP Problems are a superset of P problems
4.	All P problems are deterministic in nature	All the NP problems are non-deterministic in nature
5.	**Example:** Selection sort, Linear search	**Example:** TSP, Knapsack problem

4) NP Complete Class Problems (5 minutes)

1. If a problem is both NP and NP-hard, it is said to be NP-complete. The challenging NP problems are those that are NP-complete.
2. Since any NP class problem may be converted into an NP-complete problem in a polynomial amount of time, NP-complete problems are unique.
3. Any NP problem might be solved in polynomial time if one could solve an NP-complete problem in that amount of time.

Examples of NP Complete Problem:

• Decision version of 0/1 Knapsack.
• Hamiltonian Cycle.
• Satisfiability.
• Vertex cover.

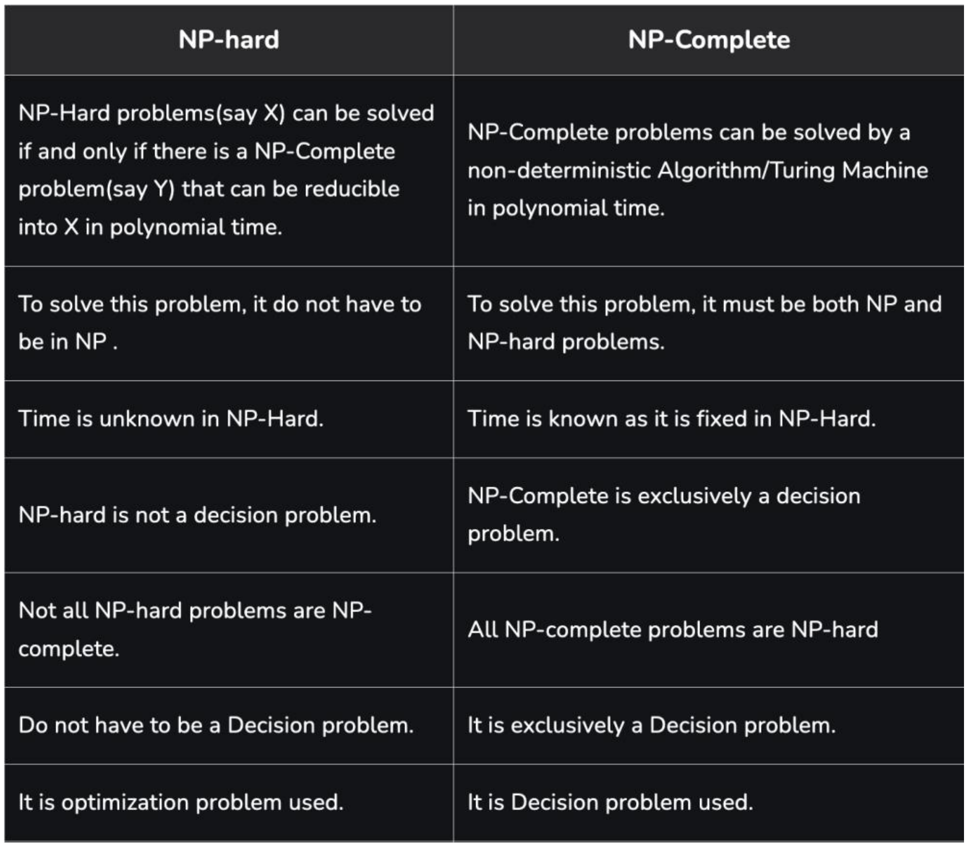
5) NP Hard Class Problems (5 minutes)

1. If problem Y is NP-Complete and reducible to X in polynomial time, then problem X is NP-Hard. The difficulty of NP-Hard and NP-Complete tasks is equal. NP class is not required for NP-Hard Problems.
2. If every NP problem can be solved in a polynomial amount of time, it is referred to as NP-Hard.
3. Many times, solving one problem leads to the reduction of other problems.

Examples of NP Hard Problems:

- Hamiltonian cycle
- Optimization problem
- Shortest path

Difference between NP-hard and NP-Complete:



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.

Video References (30 minutes)

- P vs. NP and the Computational Complexity Zoo
https://www.youtube.com/watch?v=YX40hbAHx3s&ab_channel=hackerdashery

- What Makes Mario NP-Hard? (Polynomial Reductions)
https://www.youtube.com/watch?v=oS8m9fSk-Wk&ab_channel=UndefinedBehavior

- **NP-Complete Explained**
https://www.youtube.com/watch?v=W9G_1xG77LE&t=307s&ab_channel=UndefinedBehavior