DSE 2256 DESIGN & ANALYSIS OF ALGORITHMS

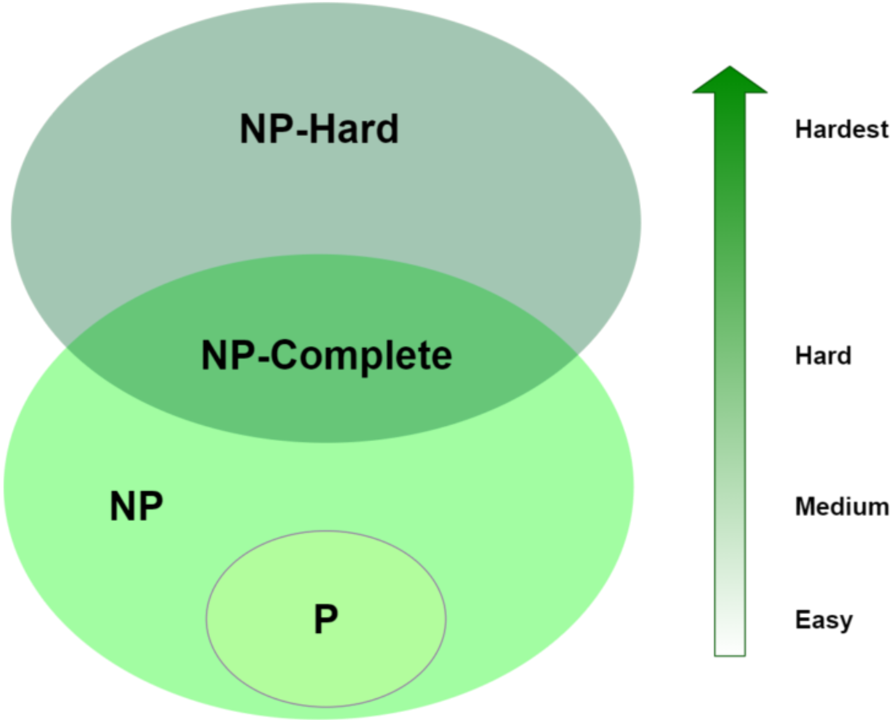
Lecture 40

Limitations of Algorithm Power
P, NP, NP-Complete, NP-Hard Problems

Instructors:

Dr. Savitha G,
Assistant Professor, DSCA, MIT, Manipal

Dr. Abhilash K. Pai,
Assistant Professor, DSCA, MIT, Manipal



Source: https://www.baeldung.com/

Key Terminologies and Definitions

1. **Decision Problems:** A problem that can be posed as a yes–no question of the input values.

2. **Deterministic and Non-Deterministic Algorithms**

   - In deterministic algorithm, for a given particular input, the computer will always produce the same output.
   - In non-deterministic algorithm, for the same input, the computer may produce different output in different runs.

Formal Definition of a Non-Deterministic Algorithm:

A non-deterministic algorithm is a two-stage procedure that takes as its input an instance I of a decision problem and does the following:

- **Nondeterministic ("guessing") stage**: An arbitrary string S is generated that can be thought of as a candidate solution to the given instance I.
- **Deterministic ("verification") stage**: A deterministic algorithm takes both I and S as its input and outputs yes if S represents a solution to instance I.

Key Terminologies and Definitions

Example of non-deterministic algorithm (For searching):

```
Algorithm NDSearch (int a, int n, int key)
{
    j= choice(a, n);
    if(A[j]==x) then
     {
        write(j);
        success();
     }
    write(0);
    failure();
}
```

Key Terminologies and Definitions

3. <u>Polynomial time algorithms</u>: An algorithm solves a problem in **polynomial time** if its worst-case time efficiency belongs to **O(p(n))** where **p(n)** is a polynomial of the problem's input size n.

<u>Examples:</u>

- Linear/Binary Search
- All sorting algoirthms
- Normal/ Strassen's Matrix Multiplication

Problems that can be solved in polynomial are called **tractable** (relatively easy)

Problems that cannot be solved in polynomial time are called **intractable** (relatively hard).

4. <u>Conjunctive Normal Form (CNF):</u> In Propositional logic, a statement is in CNF if it is a **conjunction** (AND) **of clauses**, where a clause is a **disjunction** (OR) of literals, and a literal is a variable or its negation.

$$(A \vee B) \wedge (!A \vee C)$$

$$A \vee B$$

Key Terminologies and Definitions

6. <u>CNF Satisfiability problem:</u> Given a Boolean formula of **n** variables $\mathbf{f(x_1,x_2,\ldots,x_n)}$, represented in CNF, the problem is to find the values of these variables, on which the formula takes on the value *true*.

5. <u>Problem Reduction:</u> A decision problem **D1** is said to be polynomially reducible to a decision problem **D2** (represented as D1 ≤ D2) if there exists a function **T** that transforms instances of **D1** to instances of **D2** such that:

- **T** maps all yes instances of D1 to yes instances of D2 and all no instances of D1 to no instances of D2
- **T** is computable by a polynomial time algorithm

Class P and Class NP

Class P is a class of decision problems that can be solved in polynomial time by (deterministic) algorithms.

This class of problems is called **deterministic polynomial time problems.**

Examples: Searching, Sorting, Shortest path in weighted graphs, Computing Minimum Spanning Tree, Matrix Multiplication etc.

Class NP is the class of decision problems that can be solved by nondeterministic polynomial algorithms.

This class of problems is called **non-deterministic polynomial time problems.**

Examples: Hamiltonian circuit problem, the partition problem, decision versions of the traveling salesman, the knapsack, graph coloring, and many hundreds of other difficult combinatorial optimization problems cataloged.

Class NP-Hard and NP-Complete

<u>Definition:</u> A decision problem **X** is said to be **NP-complete** if:

1. **X** belongs to class **NP**
2. **X** is **NP-Hard**

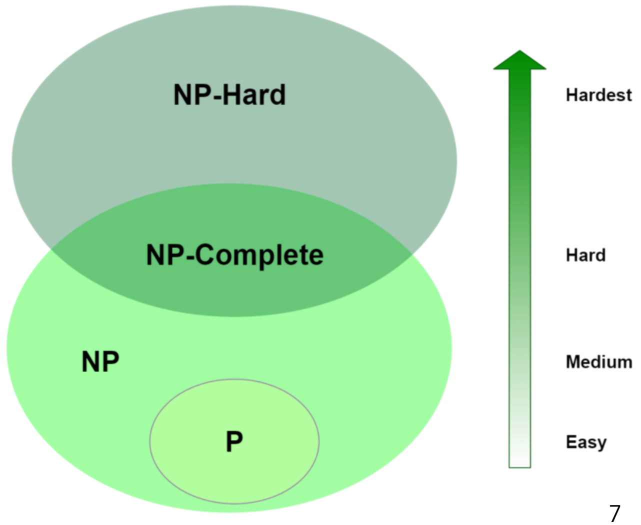
<u>Definition:</u> The decision problem **X** is **NP-Hard** if every problem **Y belonging to NP reduces to X.**

To prove the above point (2), it is enough to prove that:

- **X is in NP** and a **known problem** in **NP-Complete** can **be reduced to P**.

<u>Examples of NP-Hard problems:</u>

CNF satisfiability, Knapsack, Graph coloring, Travelling salesman Problem, Hamiltonian Circuit problem, Sudoku puzzle, Sub-set sum problem etc.



Thank you!

Any queries?