

Advanced Data Structures and Algorithms

NP-Hard and NP-Complete Problems

Dr G.Kalyani

**Department of Information Technology
VR Siddhartha Engineering College**

Topics

- **Basic Concepts**
- **Non-Deterministic Algorithms**
- **NP-Hard and NP-Complete Problems**

Categorization of Problems

- **Depending on problem characteristics**
- **Depending on time complexity**
- **Depending on the logic used to solve the problem**

Types of Problems

Depending on Problem Characteristics

- Any problem for which **answer is 0 or 1** is called **decision problem**, such algorithm is called decision algorithm.
- Any problem that tries to **identify all possible solutions** is called **enumeration problem**, such algorithm is called enumeration algorithm.
- Any problem that involves **identification of optimal solution** of given cost function is known as **optimization problem**, such algorithm is optimization algorithm.

Types of Problems Depending on Time Complexity

- An algorithm for a given problem is said to be **polynomial time algorithm** if its time complexity belongs $O(n^k)$, Where k is an integer.
- An algorithm for a given problem is said to be **non polynomial time algorithm** if it is Not a polynomial time algorithm. TSP($n^2 2^n$)

Types of Problems Depending on Logic Used

- A **Deterministic Algorithm** is an algorithm that, given a particular input, always produces the same output and follows the same sequence of operations. we can identify the how the logic works during the execution.
- A **Non-Deterministic Algorithm** is an algorithm that, given a particular input, may not always produces the same output even it follows the same sequence of operations. we can not identify the how the logic works during the execution.

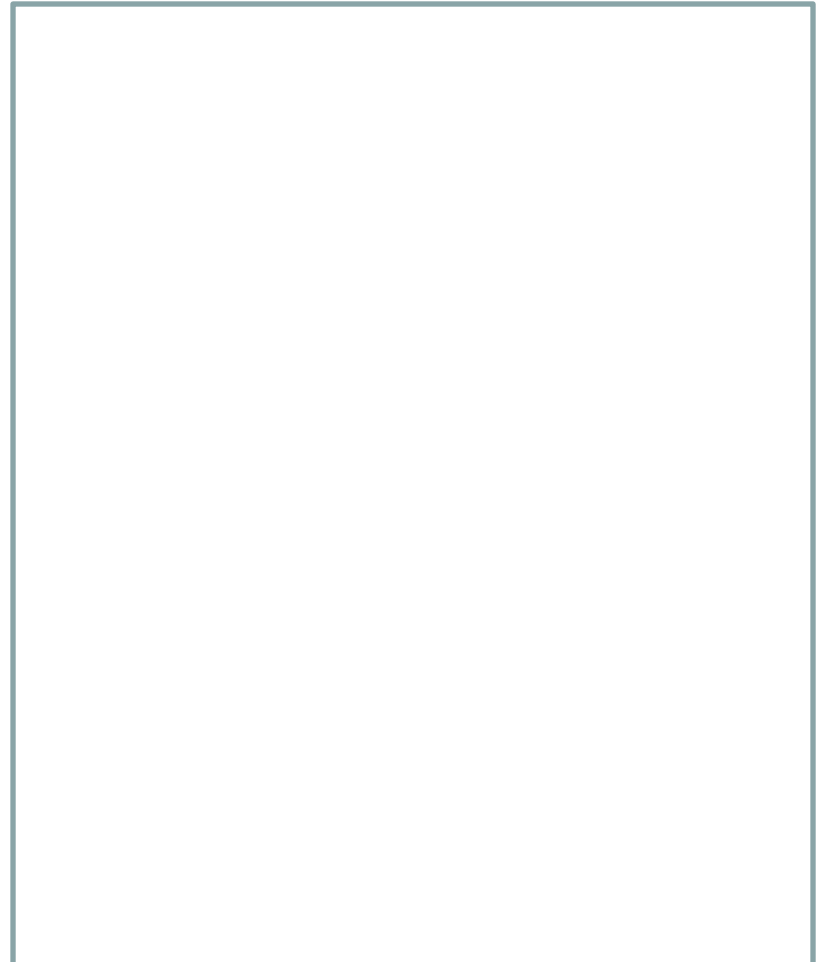
Deterministic & Non-Deterministic Search Algorithm

Deterministic

A is array of size n

```
for j= 1 to n
{
  If A[j]=key then
  {
    Write(j);
  }
}
Write(0);
```

Non-Deterministic



Deterministic & Nondeterministic Search Algorithm

Deterministic

```
A is array of size n

for j= 1 to n
{
    If A[j]=key then
    {
        Write(j);
    }
}
Write(0);
```

Non-Deterministic

```
A is array of size n
for i=1 to n
{
    j:=Choice(1,n)
    If A[j]=key then
    {
        Write(j);
        Success();
    }
}
Write(0);
Failure()
}
```


Nondeterministic Sorting Algorithm

```
1  Algorithm NSort( $A$ ,  $n$ )
2  // Sort  $n$  positive integers.
3  {
4      for  $i := 1$  to  $n$  do  $B[i] := 0$ ; // Initialize  $B[ ]$ .
5      for  $i := 1$  to  $n$  do
6          {
7               $j := \text{Choice}(1, n)$ ;
8              if  $B[j] \neq 0$  then Failure();
9               $B[j] := A[i]$ ;
10         }
11     for  $i := 1$  to  $n - 1$  do // Verify order.
12         if  $B[i] > B[i + 1]$  then Failure();
13     write ( $B[1 : n]$ );
14     Success();
15 }
```

Satisfiability(SAT) Problem

- Let x_1, x_2, \dots denote boolean variables (their value is either true or false).
- Let \bar{x}_i denote the negation of x_i .
- A literal is either a variable or its negation.
- A formula in the propositional calculus is an expression that can be constructed using literals and the operations **and** and **or**.
- Examples of such formulas are $(x_1 \wedge \bar{x}_2) \vee (x_3 \wedge \bar{x}_4)$ and $(x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2)$.
- The symbol \vee denotes or and \wedge denotes and.

Satisfiability(SAT) Problem contd..

- A formula is in **Conjunctive Normal Form (CNF)** if and only if it is represented as $\bigwedge_{i=1}^k c_i$ where the c_i are clauses each represented as $\bigvee l_{ij}$. The l_{ij} are literals.
- It is in **Disjunctive Normal Form(DNF)** if and only if it is represented as $\bigvee_{i=1}^k c_i$ and each clause c_i is represented as $\bigwedge l_{ij}$.
- Ex for DNF: $(x_1 \wedge x_2) \vee (x_3 \wedge \bar{x}_4)$
- Ex for CNF: $(x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2)$
- The satisfiability problem is to determine whether a formula is true for some assignment of truth values to the variables.
- CNF-satisfiability is the satisfiability problem for CNF formulas.

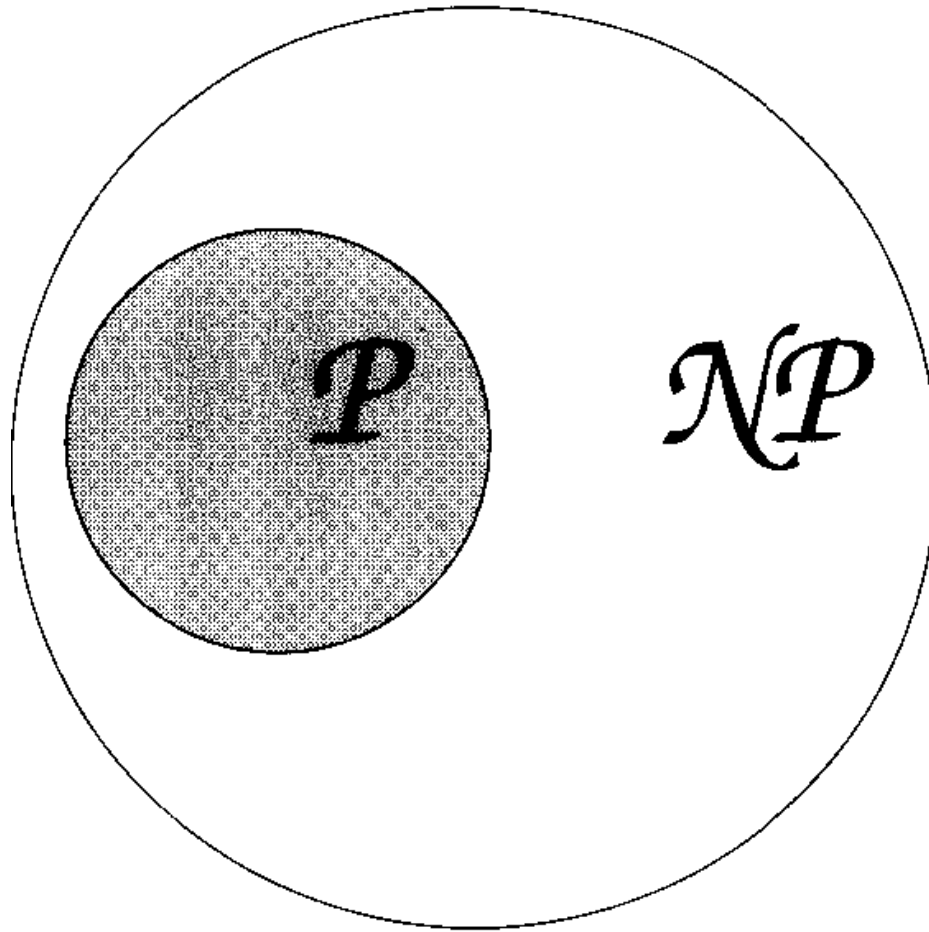
Nondeterministic Algorithm for SAT Problem

```
1  Algorithm Eval( $E$ ,  $n$ )
2  // Determine whether the propositional formula  $E$  is
3  // satisfiable. The variables are  $x_1, x_2, \dots, x_n$ .
4  {
5      for  $i := 1$  to  $n$  do // Choose a truth value assignment.
6           $x_i := \text{Choice}(\text{false}, \text{true});$ 
7          if  $E(x_1, \dots, x_n)$  then Success();
8          else Failure();
9  }
```

P and NP Problems

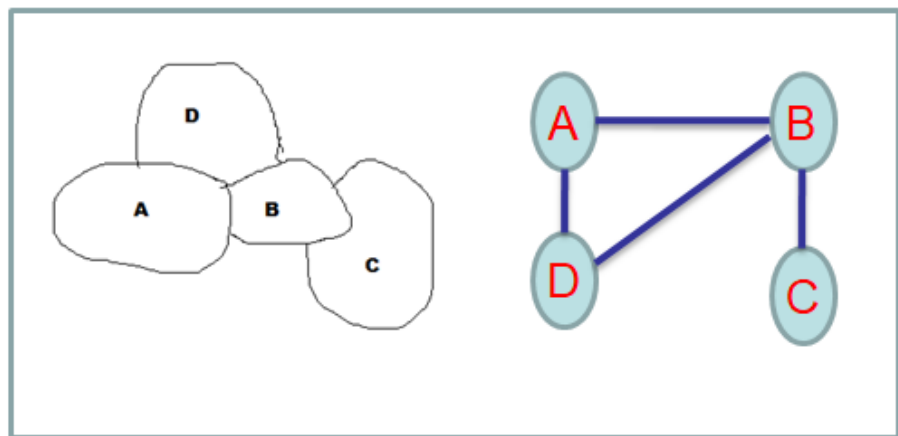
- **P:** The class of decision problems that are solvable in polynomial time by deterministic algorithms.
- **NP:** The class of decision problems that are solvable in polynomial time by nondeterministic algorithms.
- Since deterministic algorithms are just a special case of nondeterministic ones $P \subseteq NP$.
- The most famous unsolved problem in computer science, is whether $P = NP$ or $P \neq NP$.

Relationship between P and NP



Reducibility

Definition 11.4 Let L_1 and L_2 be problems. Problem L_1 *reduces* to L_2 (also written $L_1 \propto L_2$) if and only if there is a way to solve L_1 by a deterministic polynomial time algorithm using a deterministic algorithm that solves L_2 in polynomial time. \square

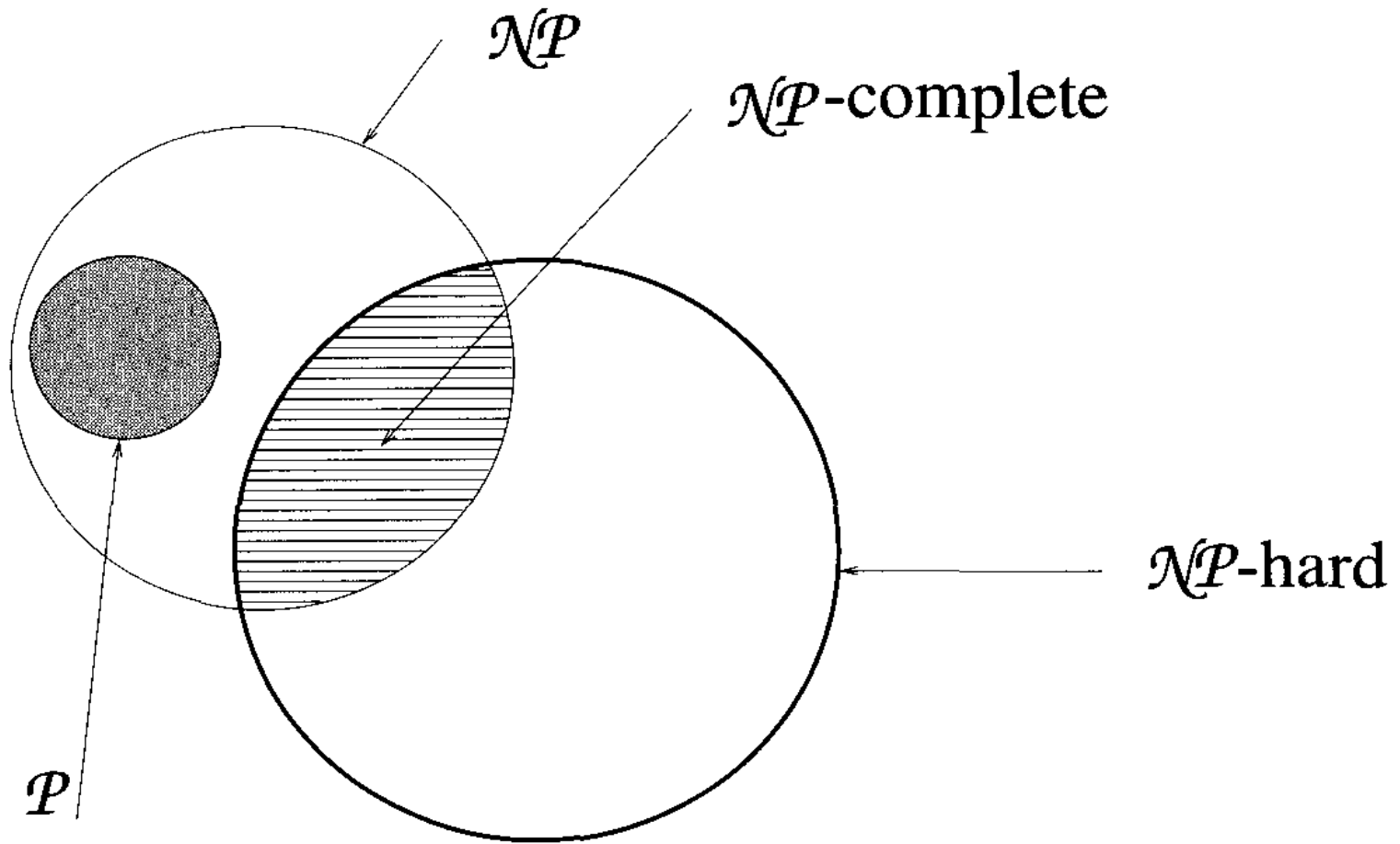


Graph
coloring
Problem

NP-Hard and NP-Complete

- **Definition of NP-Hard:** A problem L is NP-Hard if and only if satisfiability reduces to L (satisfiability α L).
- **Definition of NP-Complete:** A problem L is NP-Complete if and only if L is, NP-Hard and $L \in \text{NP}$.

Relationship between P, NP, NP-Hard and NP-Complete



Examples of NP-Hard and NP-Complete Problems

- The following problems are NP-Hard
 - The circuit-satisfiability problem
 - Set Cover
 - Vertex Cover
 - Travelling Salesman Problem
- Following are some NP-Complete problems:
 - Determining whether a graph has a Hamiltonian cycle
 - Determining whether a Boolean formula is satisfiable, etc.