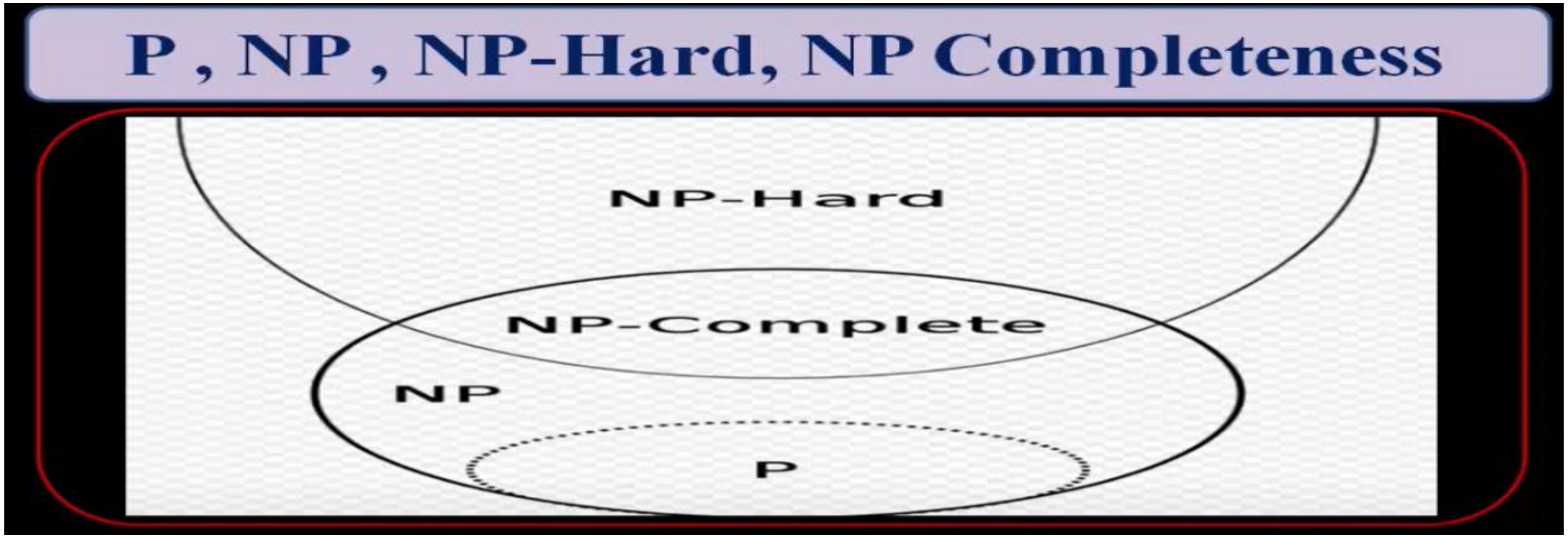


**NP - nondeterministic polynomial time**

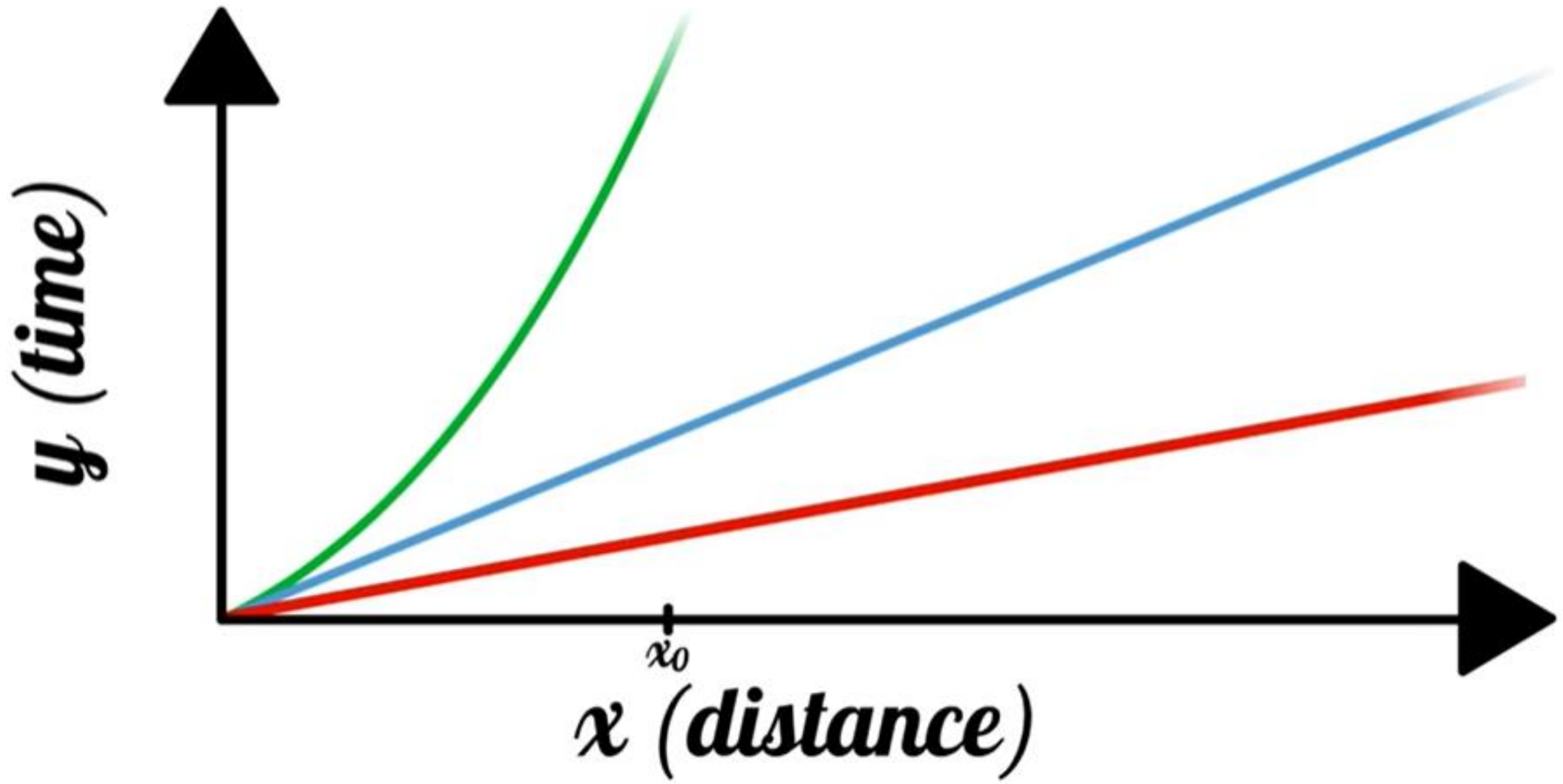
Dhaval Bhoi, Assistant Professor, CE, CSPIT, CHARUSAT

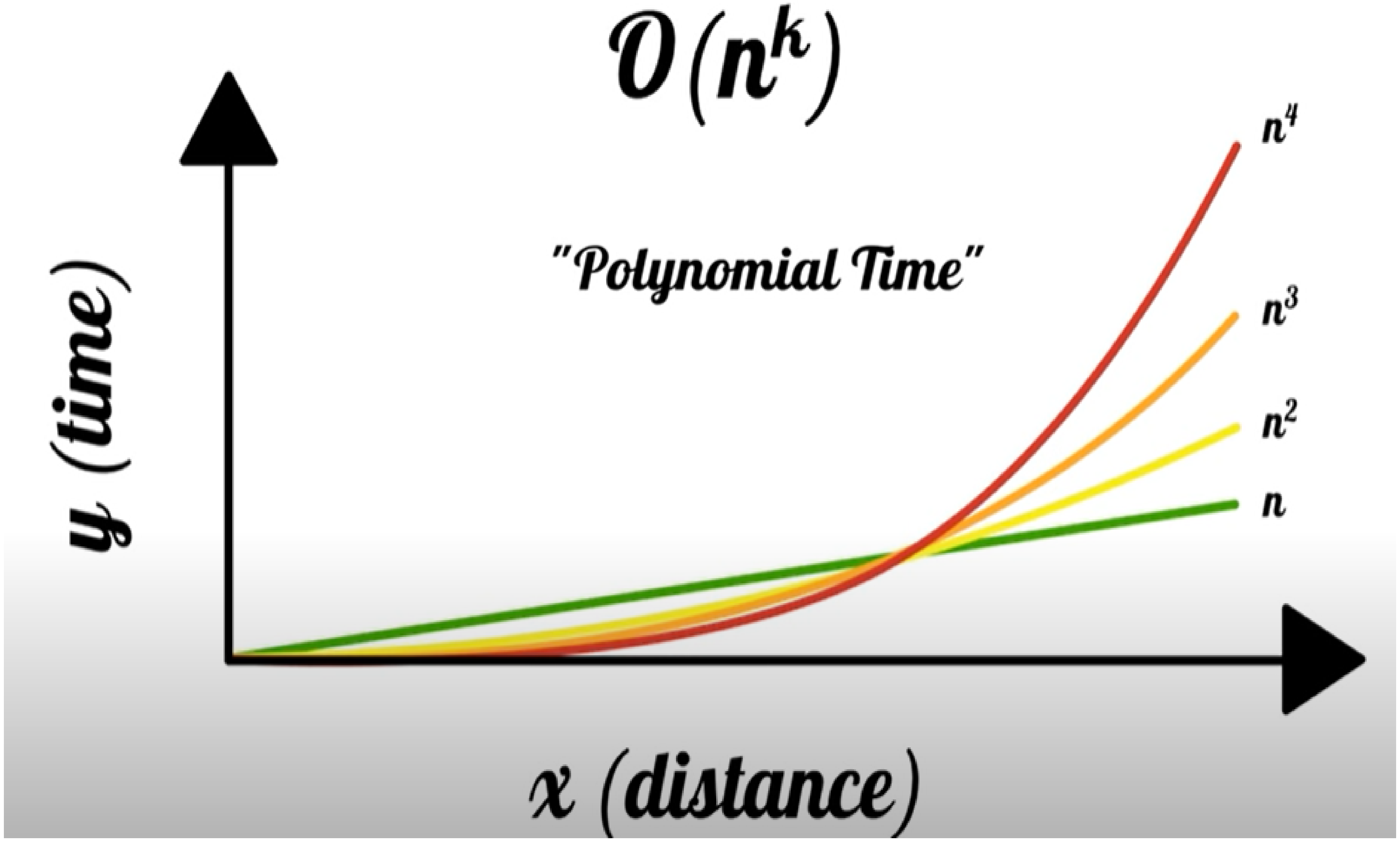
# Big O Asymptotic Analysis

$$f(x) \in O(g(x)) \text{ iff } \exists\, k, x_0 \text{ s.t. }$$
$$\forall x \geq x_0, f(x) \leq k \cdot g(x)$$

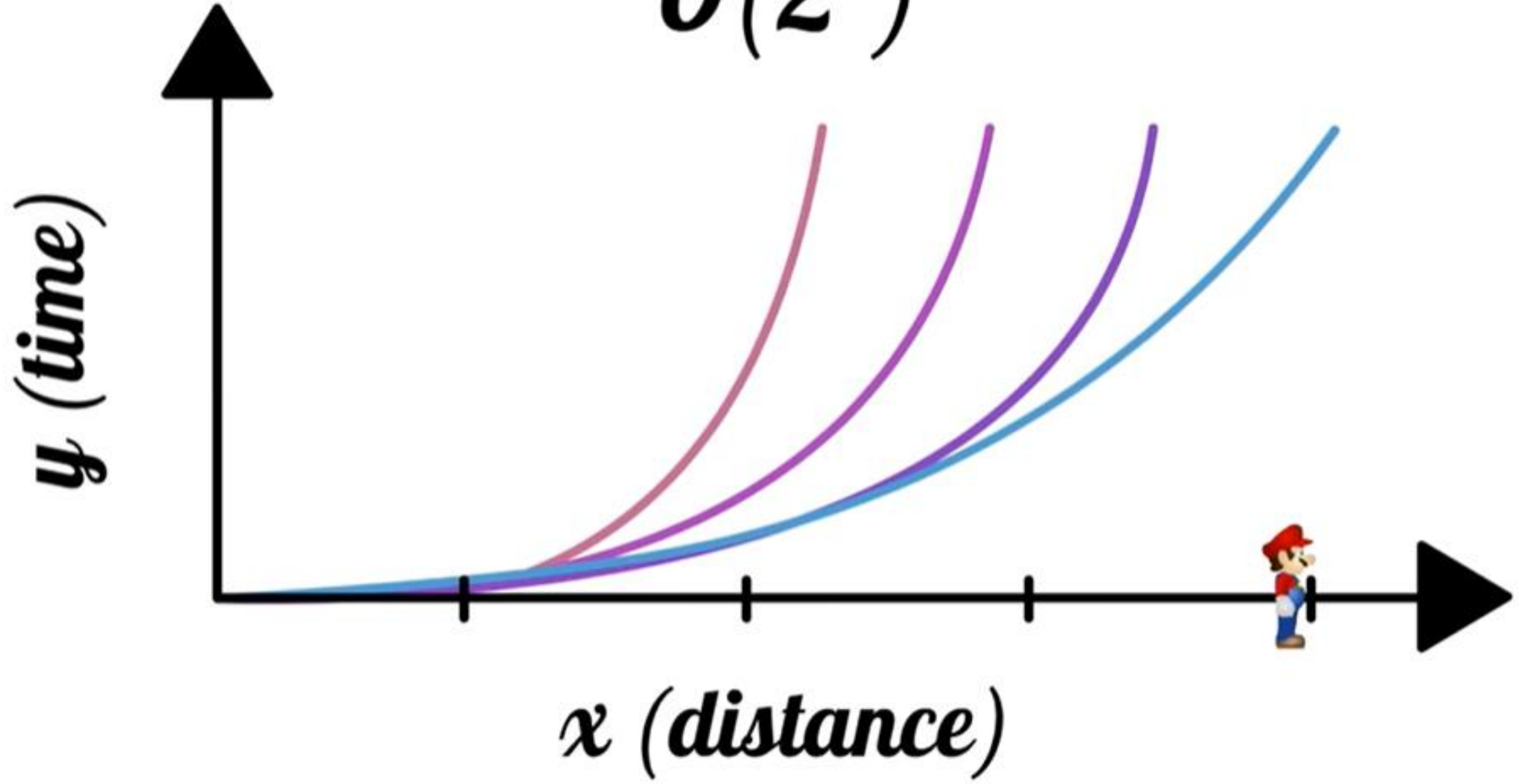
$$O(x^2) = O(5x^2 + 12x + 37)$$

**Definition:** $f(x) \in O(g(x))$ iff $\exists\, k, x_0$ s.t. $\forall x \geq x_0,\ f(x) \leq k \cdot g(x)$



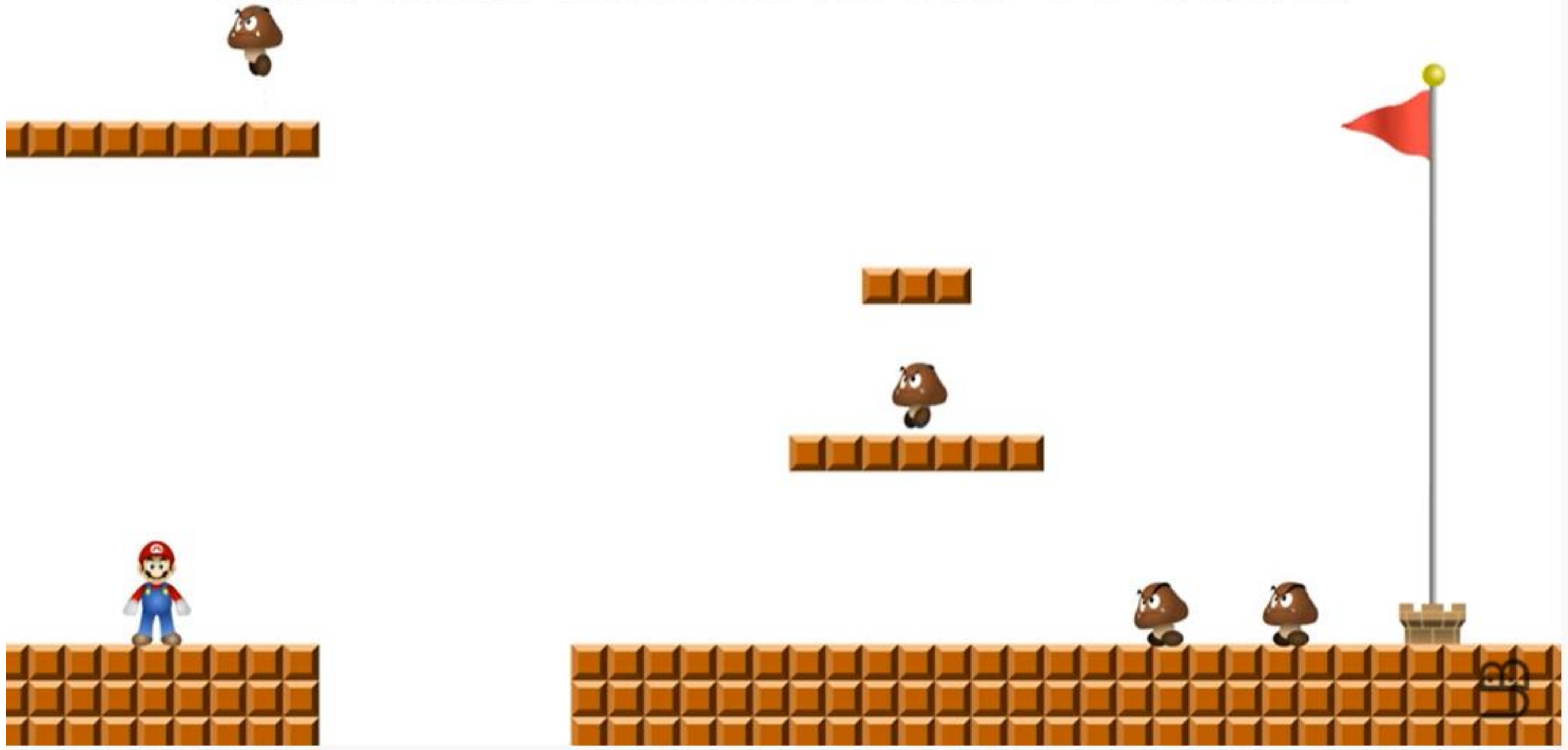

$\mathcal{O}(n^k)$
"Polynomial Time"
$n^4$
$n^3$
$n^2$
$n$
y (time)
x (distance)

## $\mathcal{O}(2^n)$

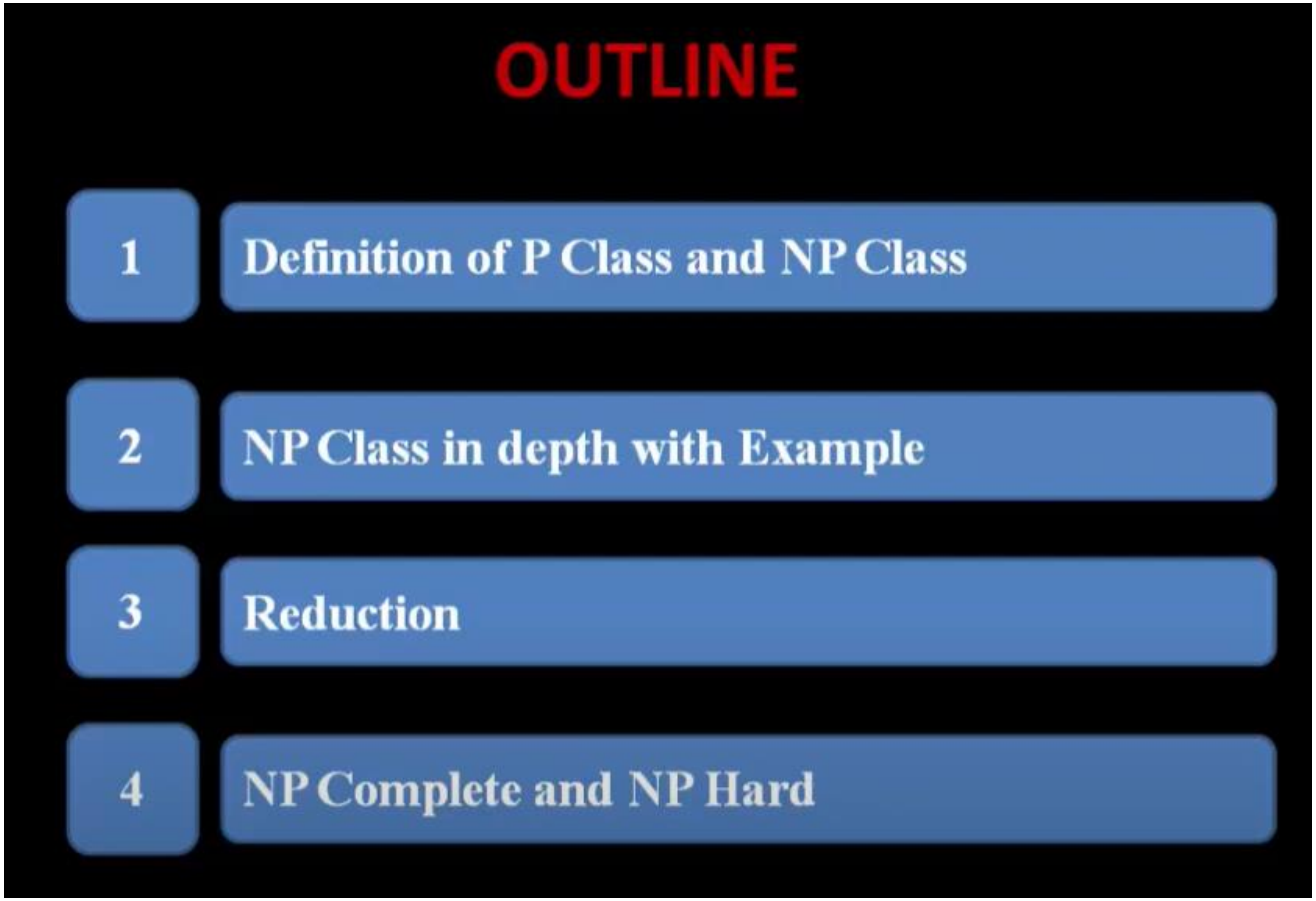


Yes
No

# Can this Mario level be beat?

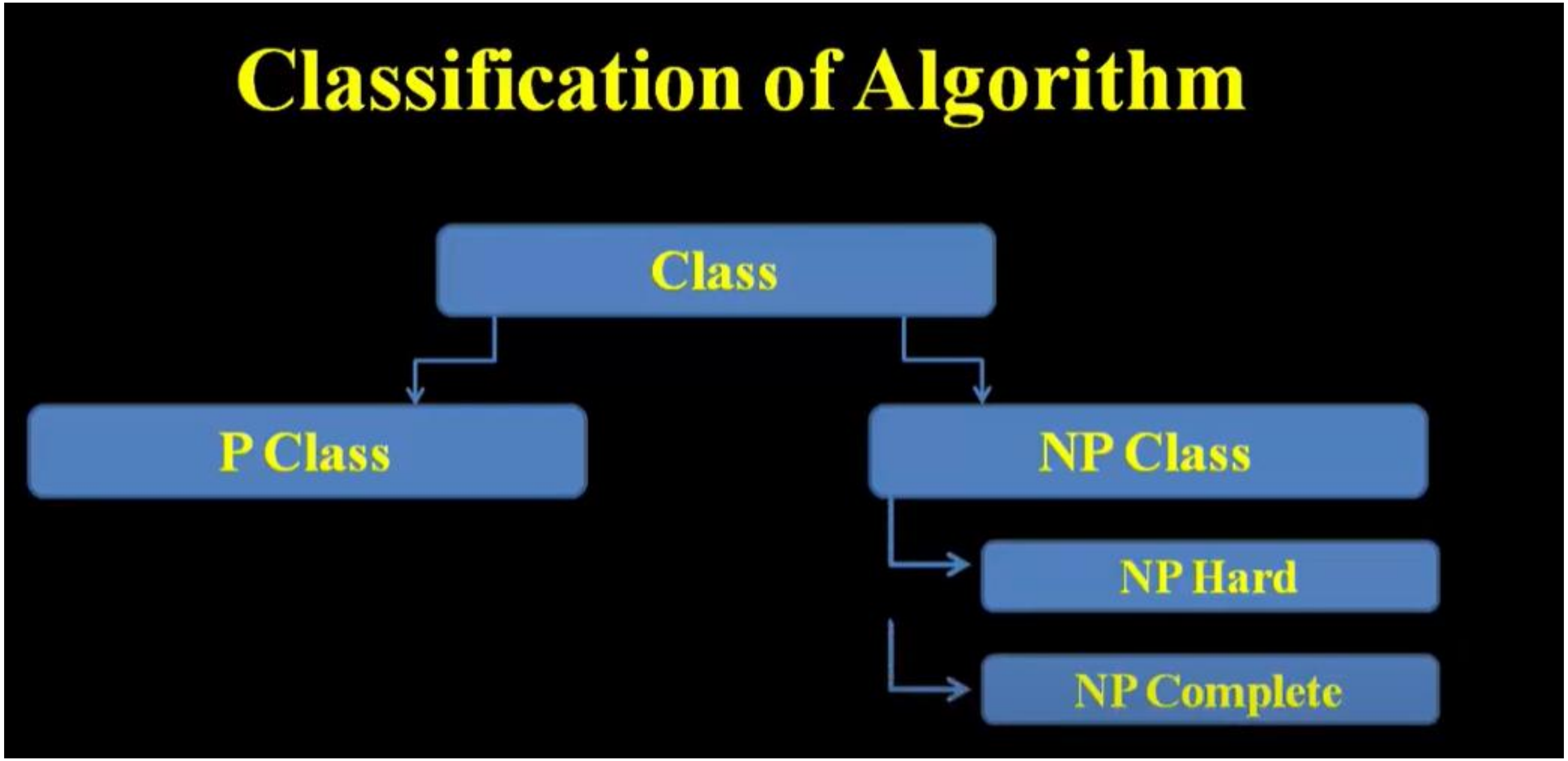


B
C
L
L
O
O
S
W
Y
Y
your name.



## OUTLINE

1. Definition of P Class and NP Class
2. NP Class in depth with Example
3. Reduction
4. NP Complete and NP Hard



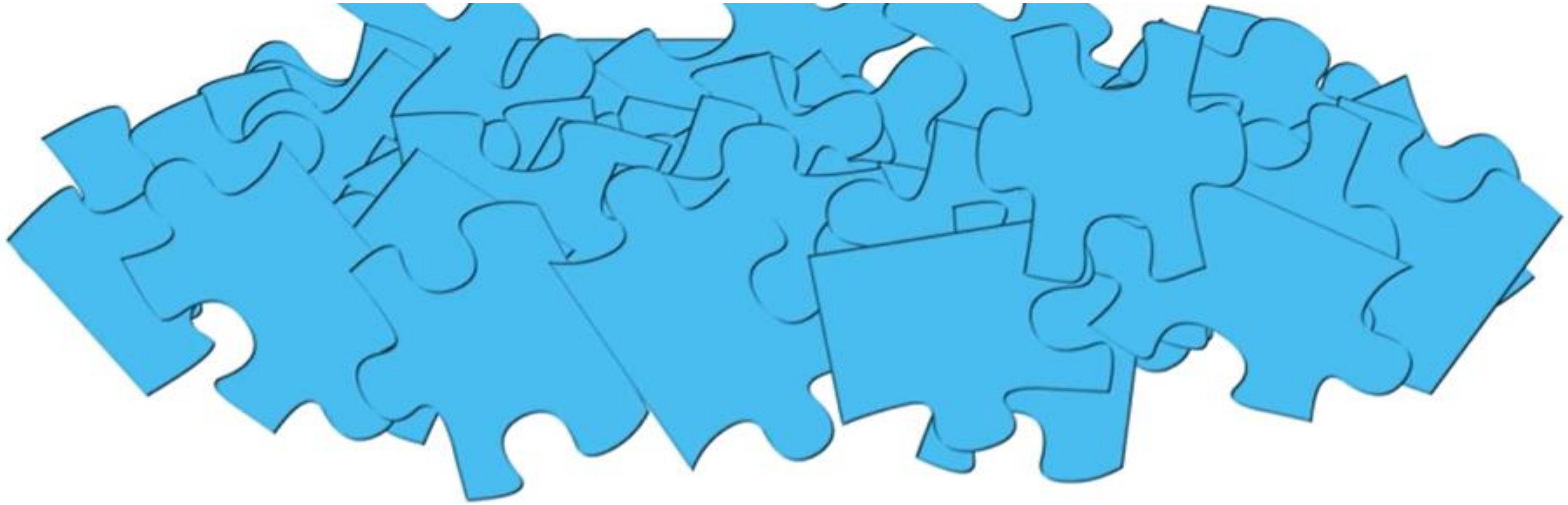
# Classification of Algorithm

$a + b$

## P

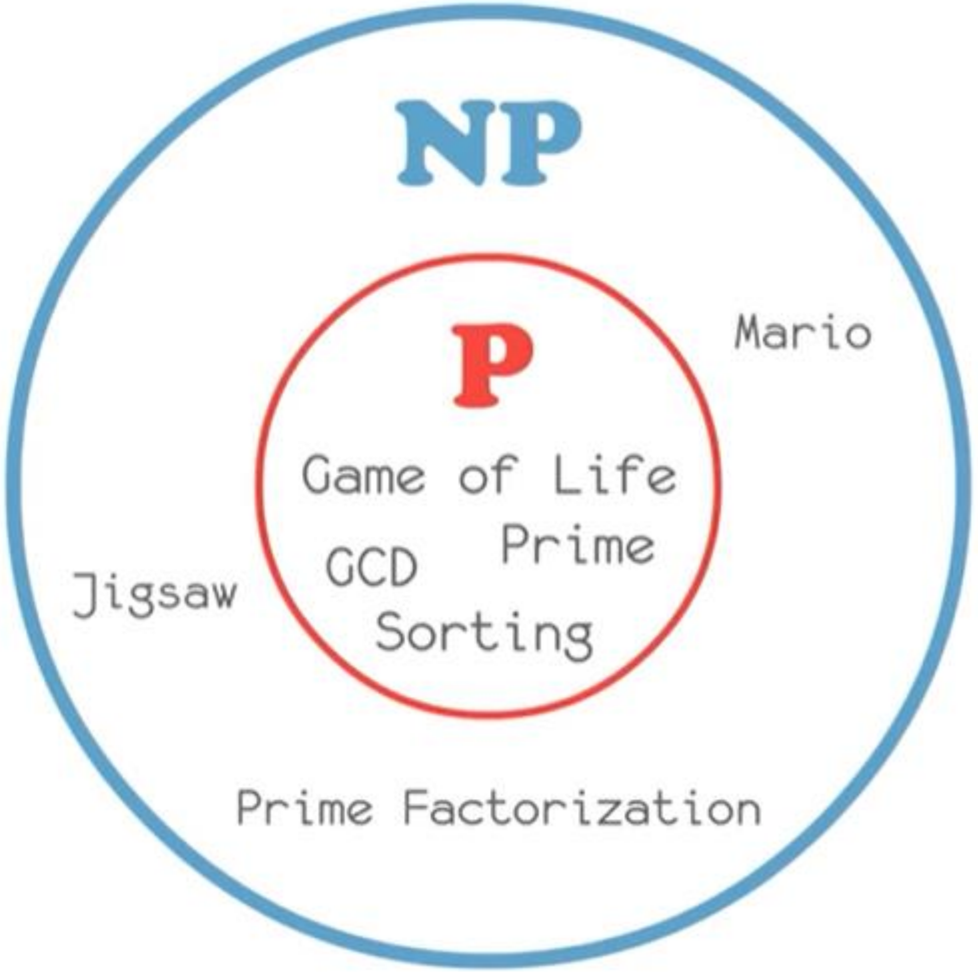
Easy to solve

Easy to verify



|  | P | NP |
|---|---|---|
| Easy to solve | ✓ | ? |
| Easy to verify | ✓ | ✓ |
| Examples | GCD, prime | Jigsaw |

$$\mathcal{O}(4^n \cdot n!)$$

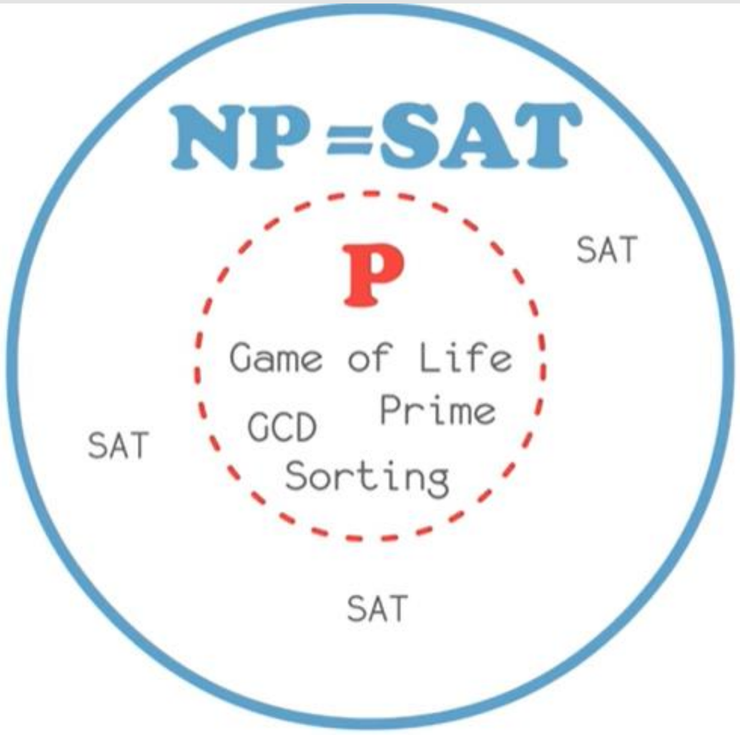


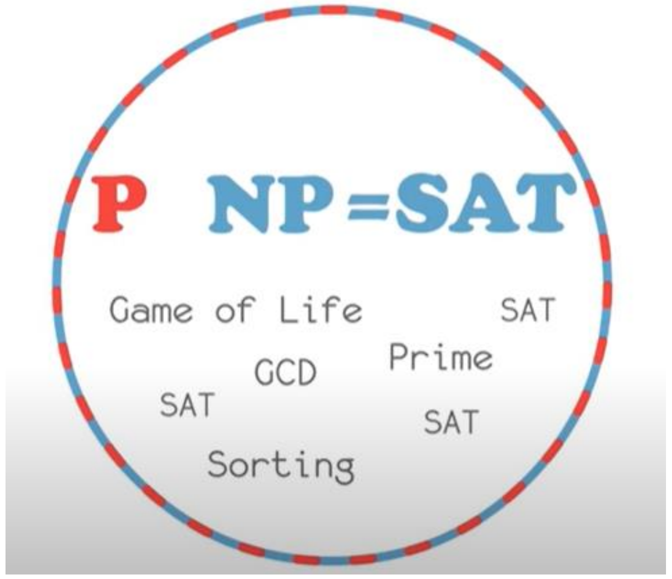
**P** *Easy*

**NP** *Hard?*

# Cook-Levin Theorem

SAT is the hardest problem in NP





# NP-Complete

Game of Life

Prime Factorization

Prime

Jigsaw

Sorting

GCD

Mario

# NP-Complete

Clique Linear Programming

Prime Factorization

Register Allocation

Sudoku

Jigsaw

Pokémon

Hamiltonian Path

Subset Sum

Residency Matching

GCD Vertex Cover

Graph Coloring

Protein Structure Prediction

Knapsack Problem

Game of Life

Battleship Math Proofs

Prime Rubik's Cube

Guarding Art Galleries

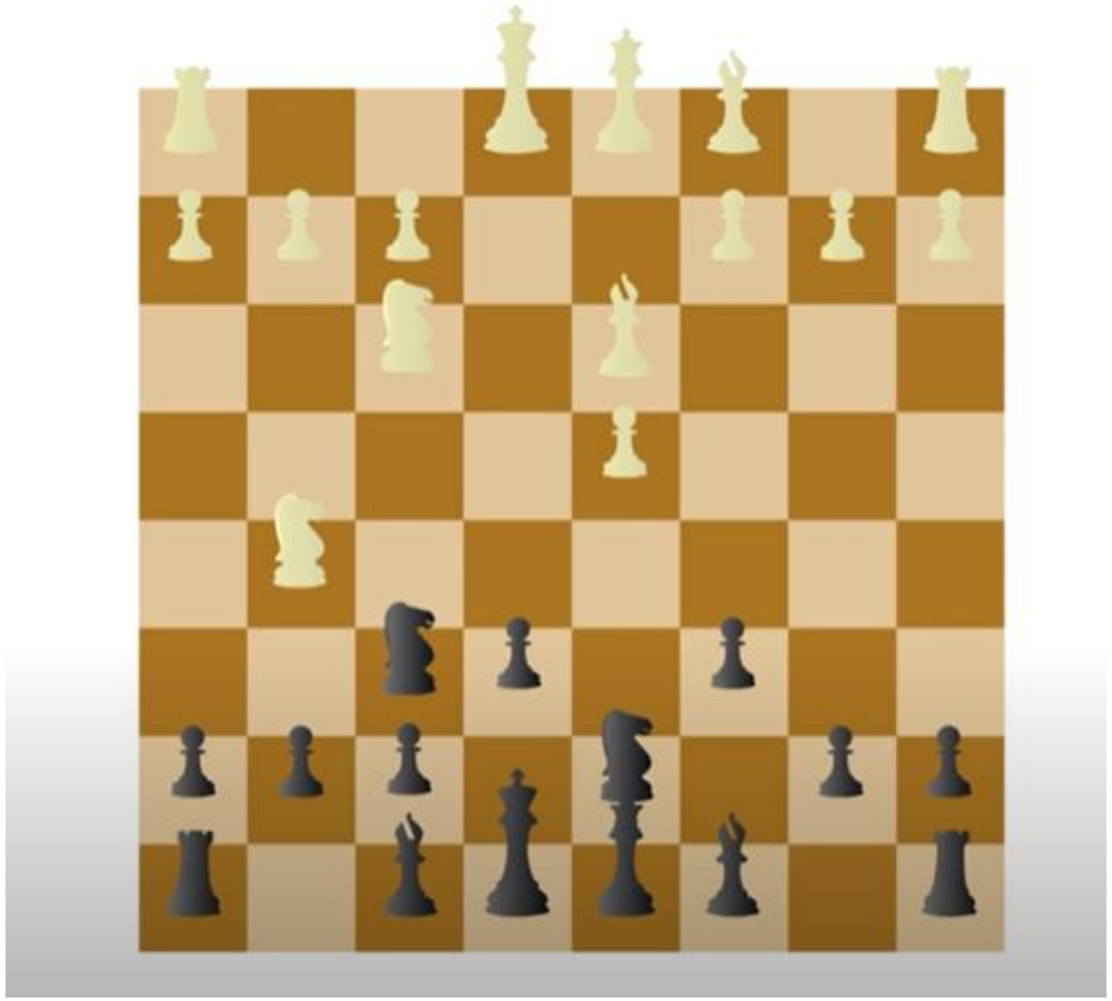
Sorting

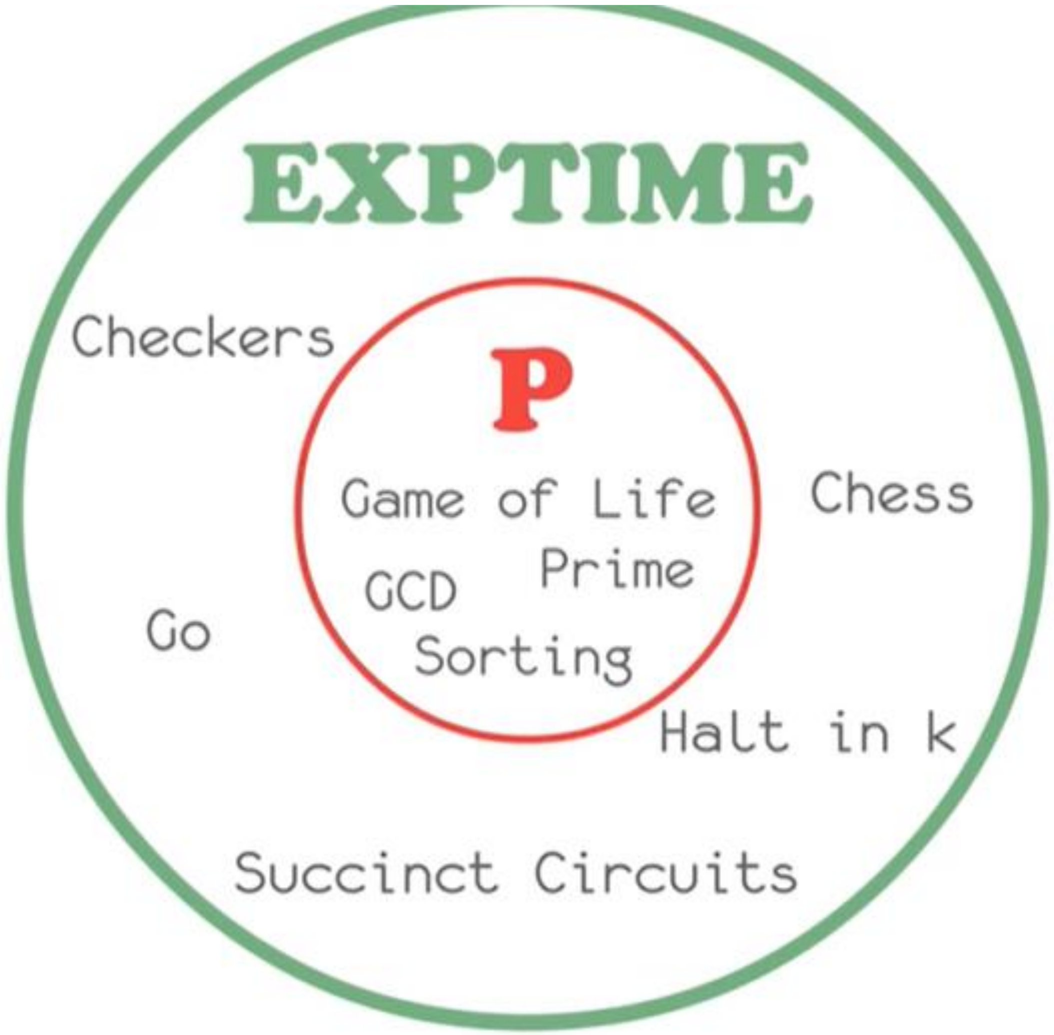
Metroid Circuit SAT

Mario Edit Distance

Candy Crush Saga

SAT




EXPTIME
Checkers
P
Game of Life
Chess
Prime
GCD
Go
Sorting
Halt in k
Succinct Circuits

P Class Problem:

A Problem which can be solved on polynomial time is known as P-Class Problem.

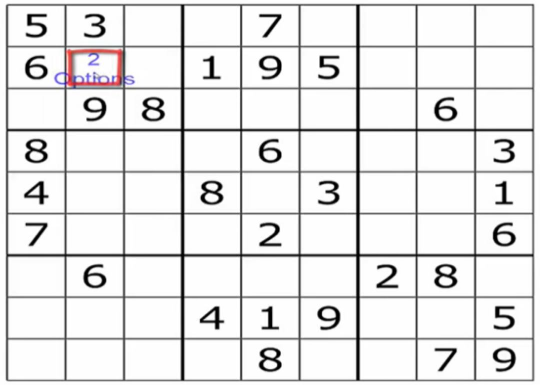
Ex: All sorting and searching algorithms.

NP Class Problem:

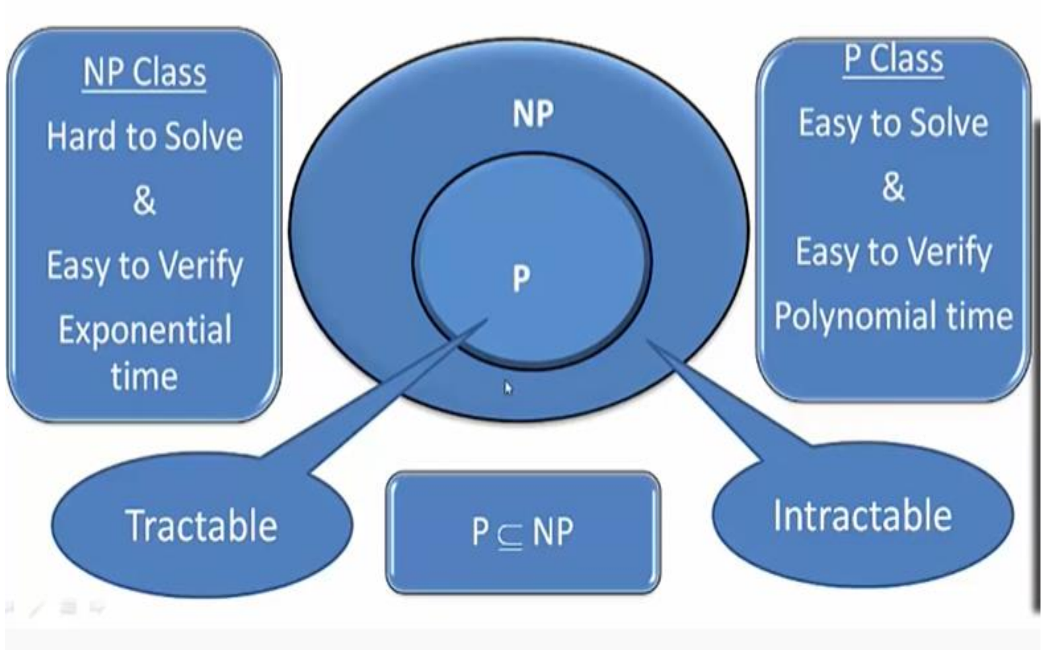
A Problem which cannot be solved on polynomial time but is verified in polynomial time is known as Non Deterministic Polynomial or NP-Class Problem.

Ex: Su-Do-Ku, Prime Factor, Scheduling, Travelling Salesman

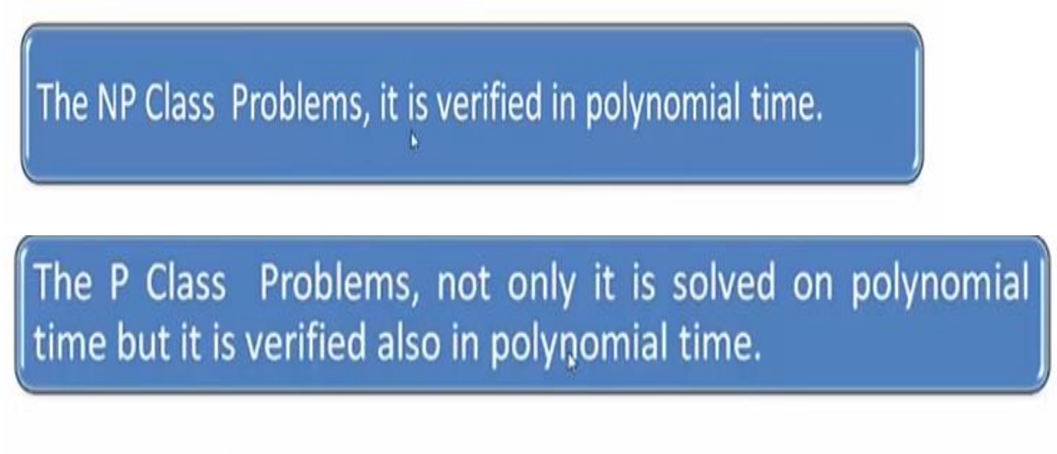
# Su-Do-ku



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | 2 Options | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

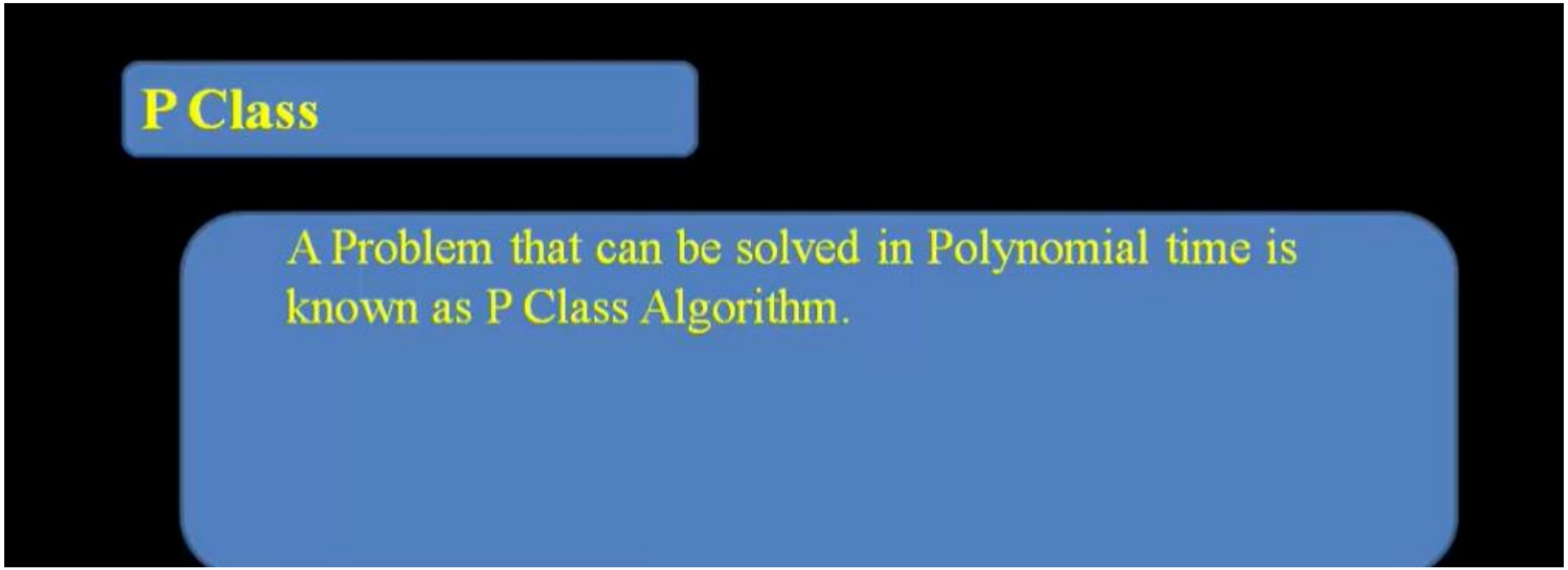


- Solving the problem is difficult but verifying is easy.



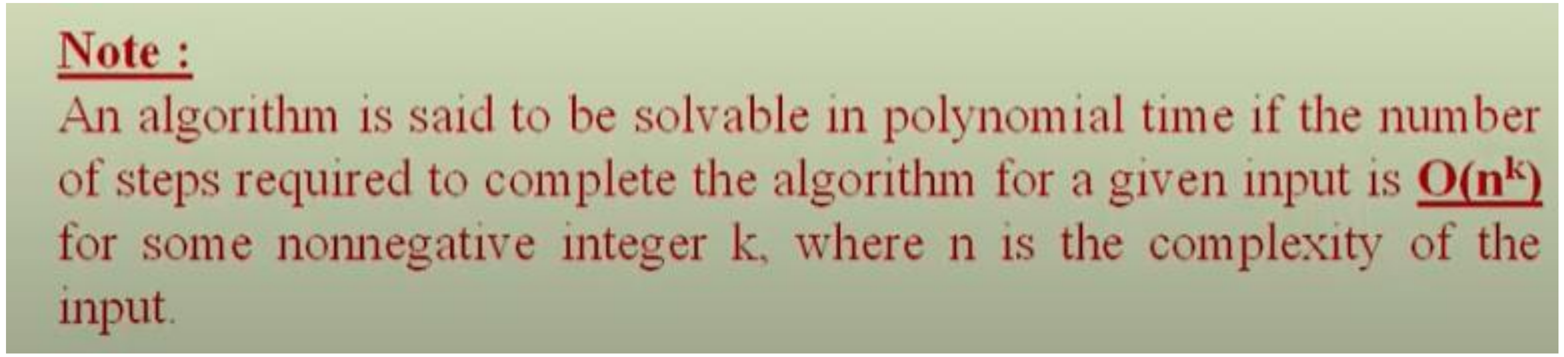
The NP Class Problems, it is verified in polynomial time.

The P Class Problems, not only it is solved on polynomial time but it is verified also in polynomial time.



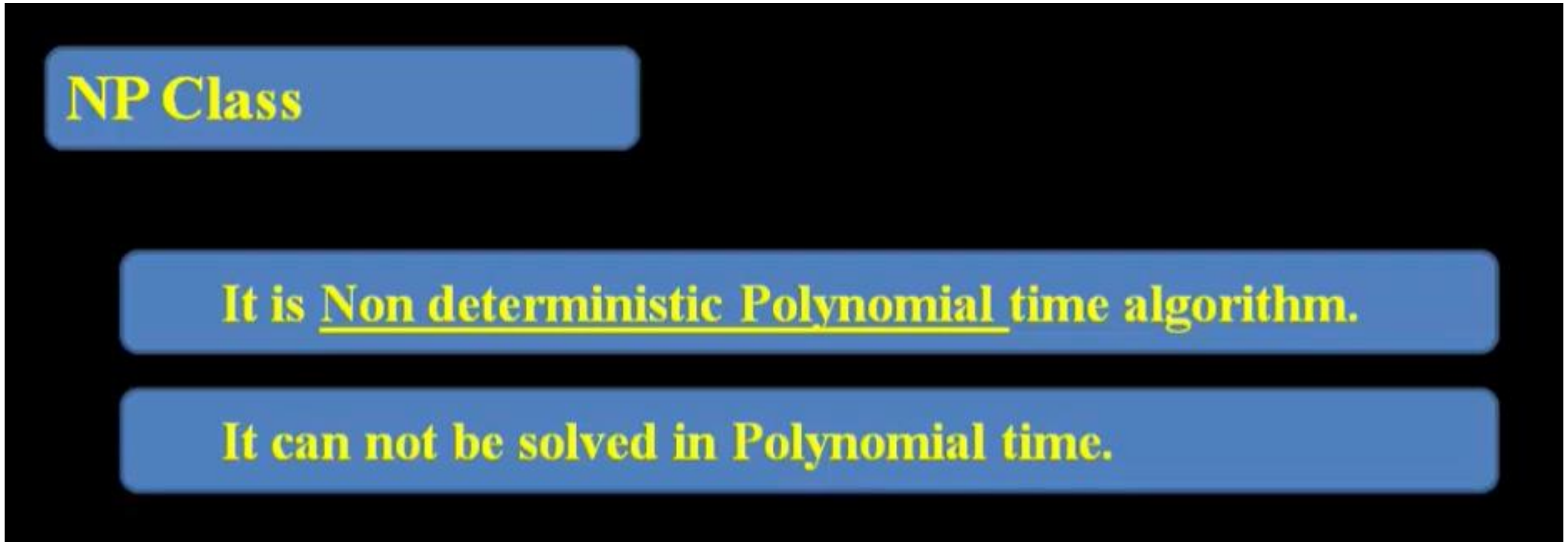
## P Class

A Problem that can be solved in Polynomial time is known as P Class Algorithm.



**Note :**

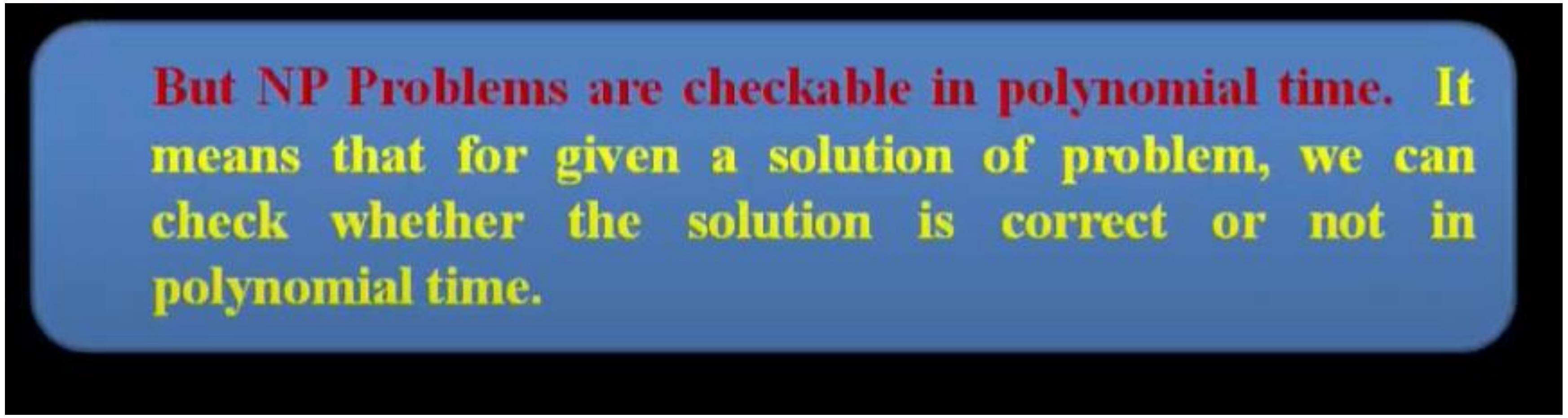
An algorithm is said to be solvable in polynomial time if the number of steps required to complete the algorithm for a given input is **$O(n^k)$** for some nonnegative integer k, where n is the complexity of the input.



## NP Class

It is Non deterministic Polynomial time algorithm.

It can not be solved in Polynomial time.



But NP Problems are checkable in polynomial time. It means that for given a solution of problem, we can check whether the solution is correct or not in polynomial time.

## Is P = NP ?

If you can prove P = NP Then

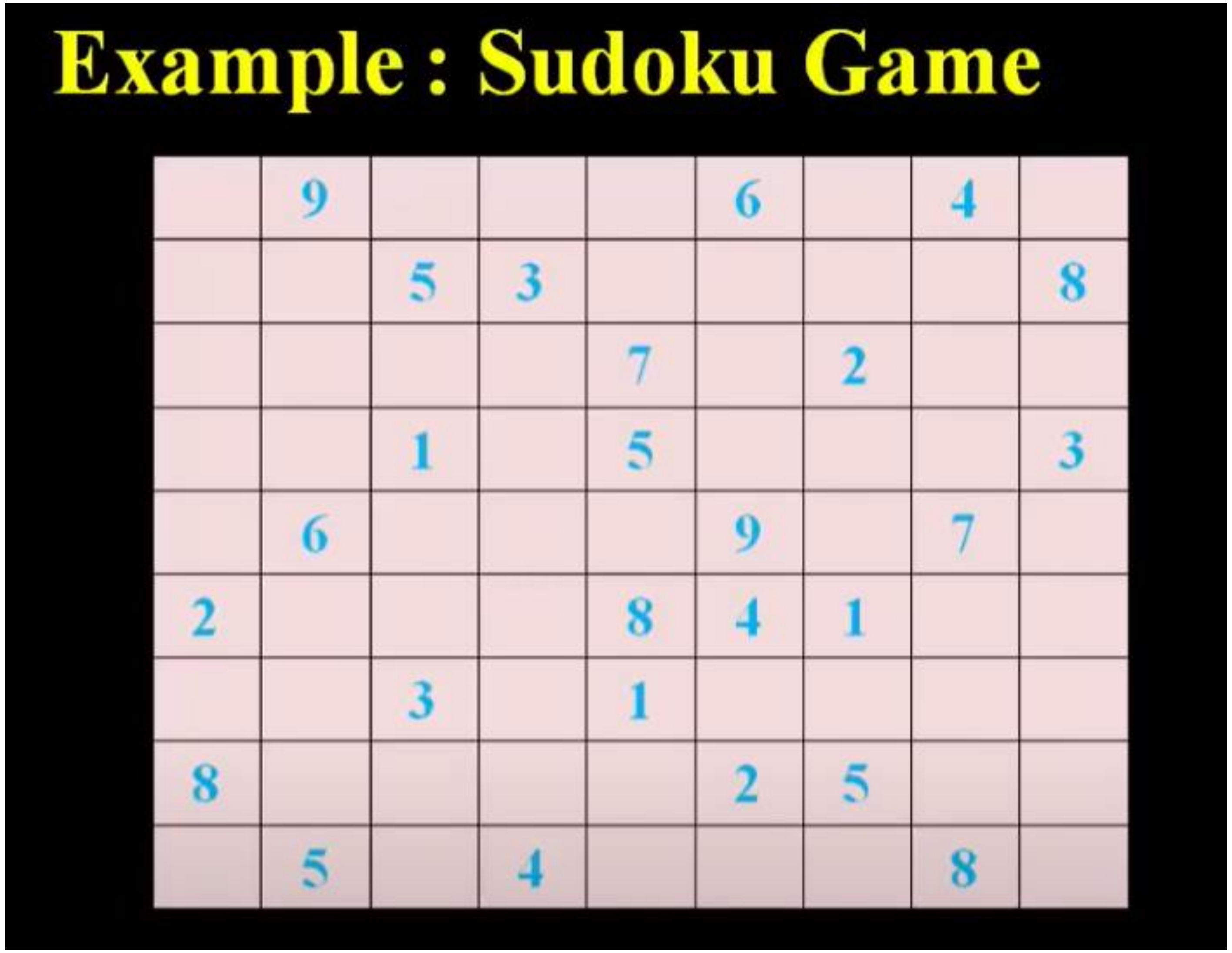
Information security or online security is vulnerable to attack,

Everything become more efficient such as

Transportation, Scheduling, understanding DNA etc.

If you can prove P $\neq$ NP Then

You can prove that there are some problems that can never be solved.



## Example : Sudoku Game

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 9 | | | | 6 | | 4 | |
| | | 5 | 3 | | | | | 8 |
| | | | | 7 | | 2 | | |
| | | 1 | | 5 | | | | 3 |
| | 6 | | | | 9 | | 7 | |
| 2 | | | | 8 | 4 | 1 | | |
| | | 3 | | 1 | | | | |
| 8 | | | | | 2 | 5 | | |
| | 5 | | 4 | | | | 8 | |

## Example : Sudoku Game

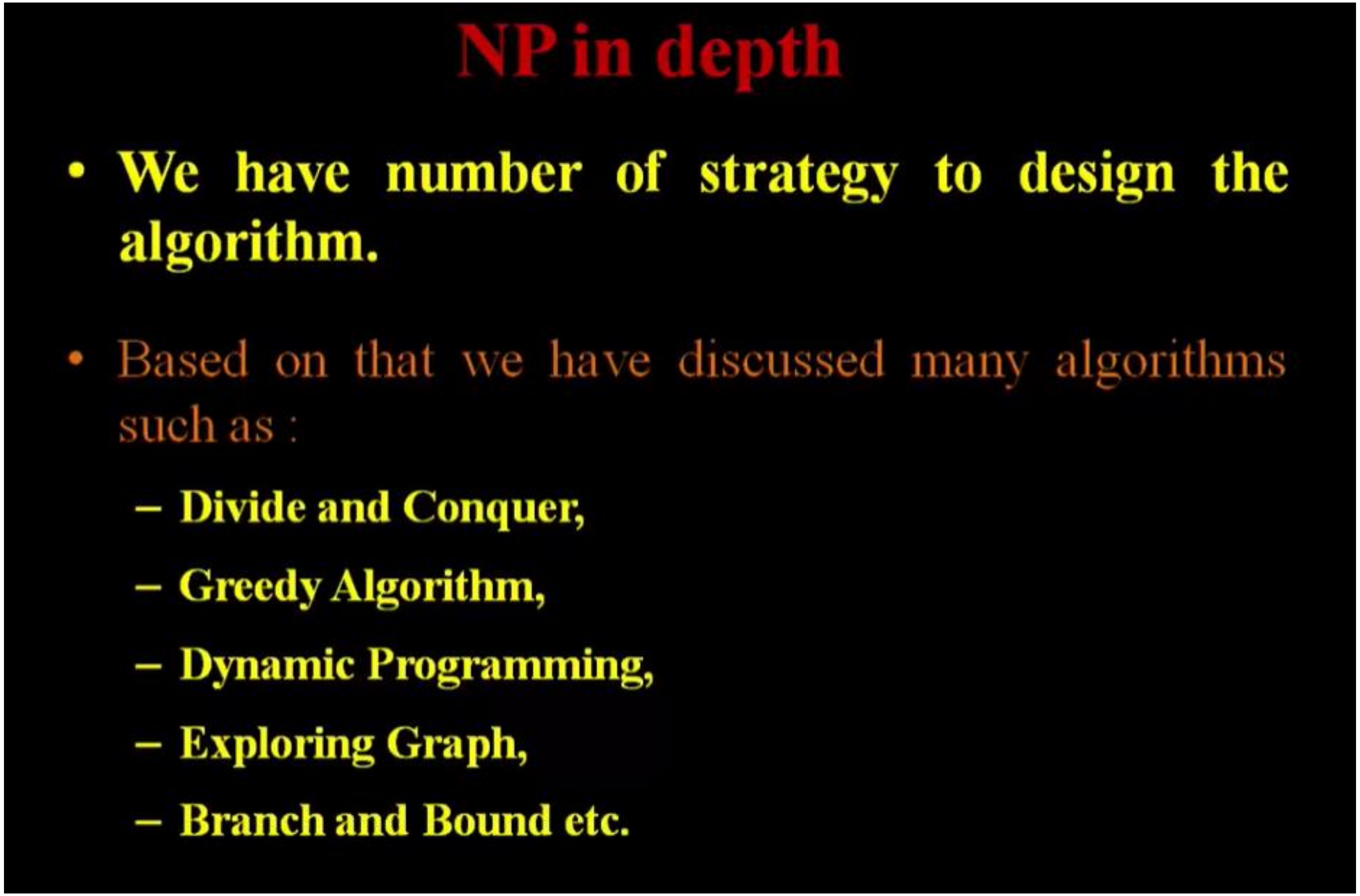
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 8 | 5 | 2 | 6 | 3 | 4 | 7 |
| 7 | 2 | 5 | 3 | 4 | 1 | 6 | 9 | 8 |
| 3 | 4 | 6 | 9 | 7 | 8 | 2 | 1 | 5 |
| 9 | 8 | 1 | 2 | 5 | 7 | 4 | 6 | 3 |
| 5 | 6 | 4 | 1 | 3 | 9 | 8 | 7 | 2 |
| 2 | 3 | 7 | 6 | 8 | 4 | 1 | 5 | 9 |
| 4 | 7 | 3 | 8 | 1 | 5 | 9 | 2 | 6 |
| 8 | 1 | 9 | 7 | 6 | 2 | 5 | 3 | 4 |
| 6 | 5 | 2 | 4 | 9 | 3 | 7 | 8 | 1 |

?

Why it is known as Non-Deterministic?

**Time Complexity is Exponential.**

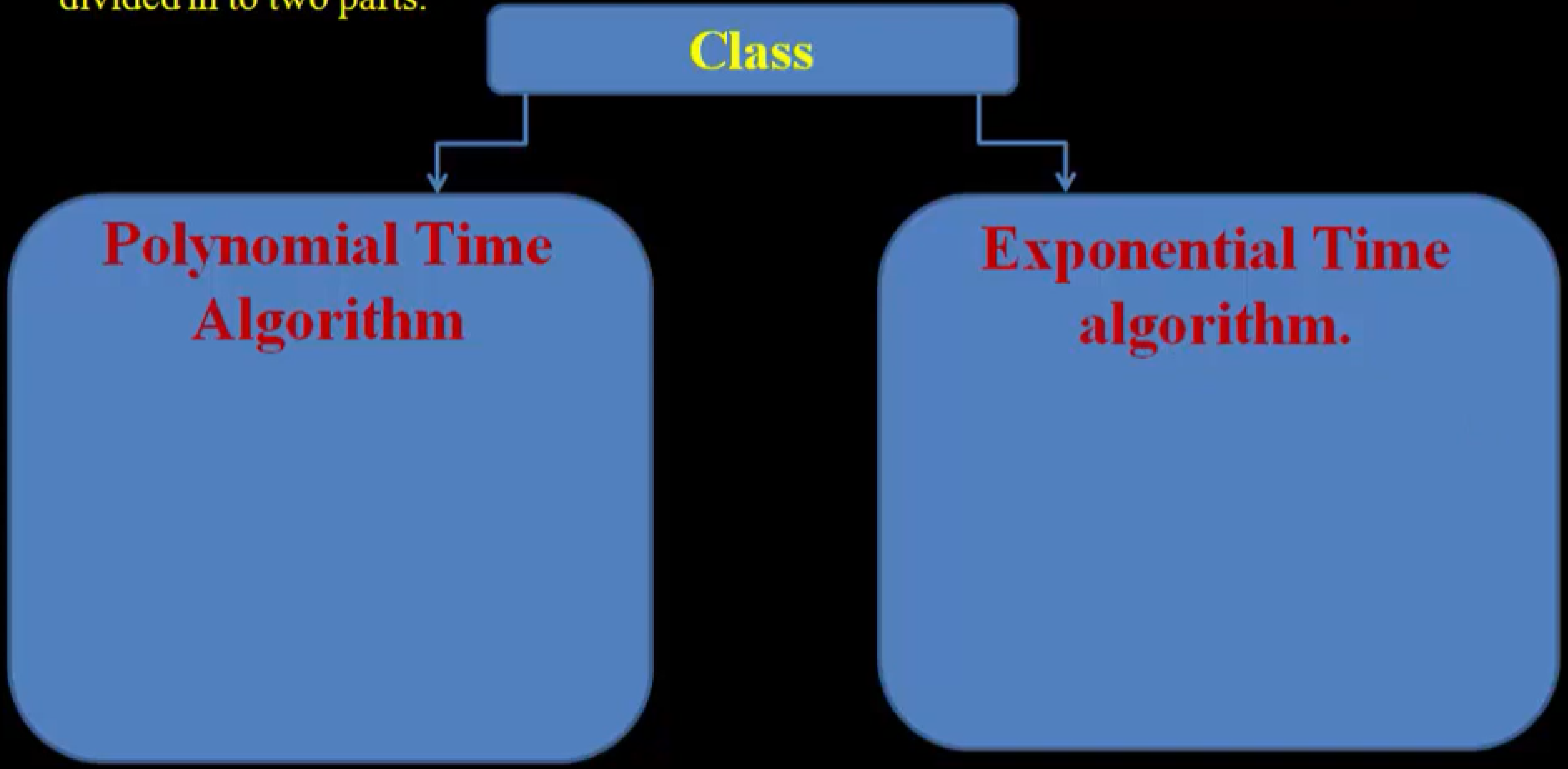
**But once we have solution then it is easy to check whether solution is correct or not. So verification can be possible in Polynomial time.**

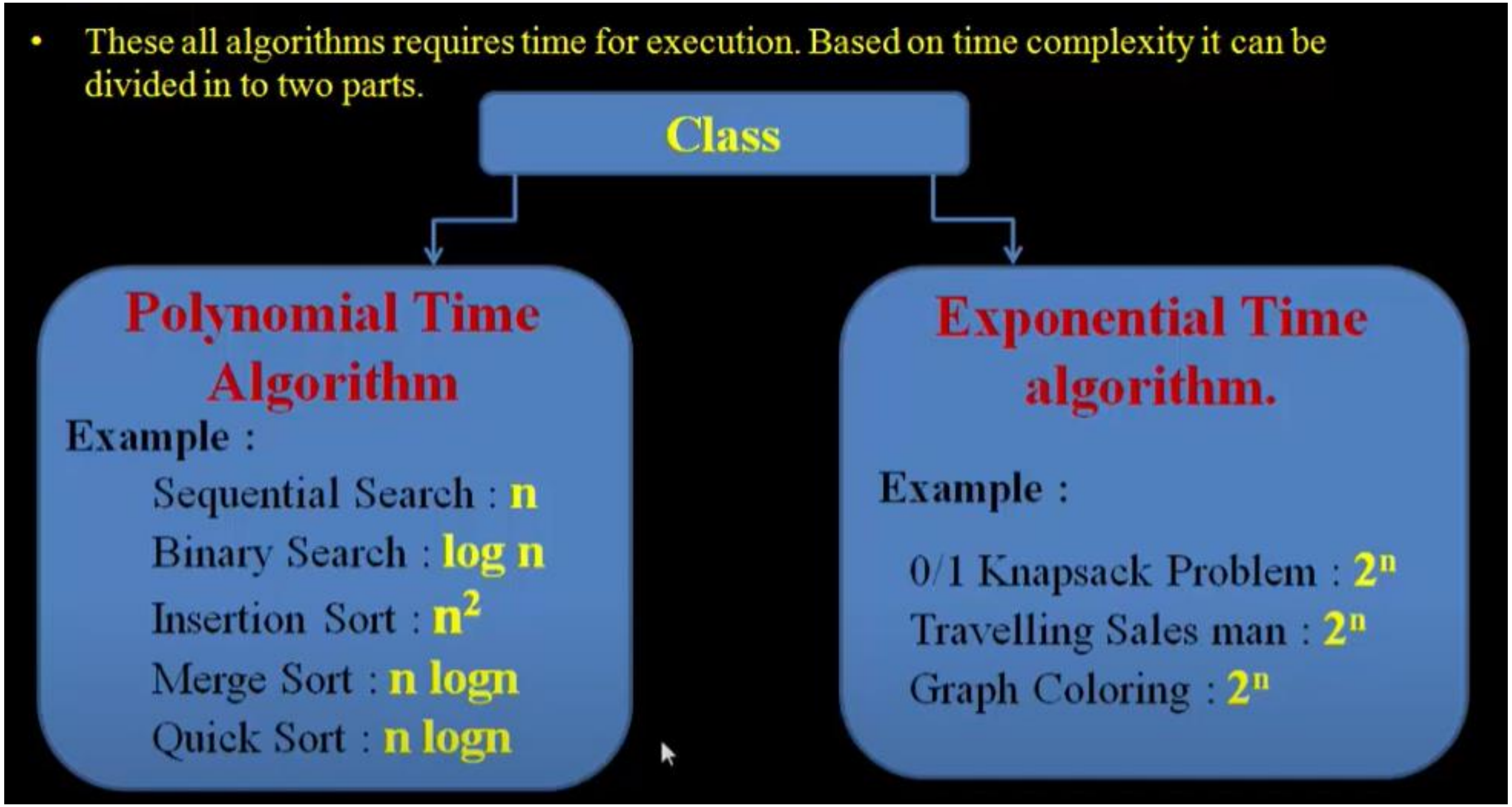


## NP in depth

- We have number of strategy to design the algorithm.
- Based on that we have discussed many algorithms such as :
  - Divide and Conquer,
  - Greedy Algorithm,
  - Dynamic Programming,
  - Exploring Graph,
  - Branch and Bound etc.

- These all algorithms requires time for execution. Based on time complexity it can be divided in to two parts.





- These all algorithms requires time for execution. Based on time complexity it can be divided in to two parts.

Class

**Polynomial Time Algorithm**

Example :

Sequential Search : $n$

Binary Search : $\log n$

Insertion Sort : $n^2$

Merge Sort : $n \log n$

Quick Sort : $n \log n$

**Exponential Time algorithm.**

Example :

0/1 Knapsack Problem : $2^n$

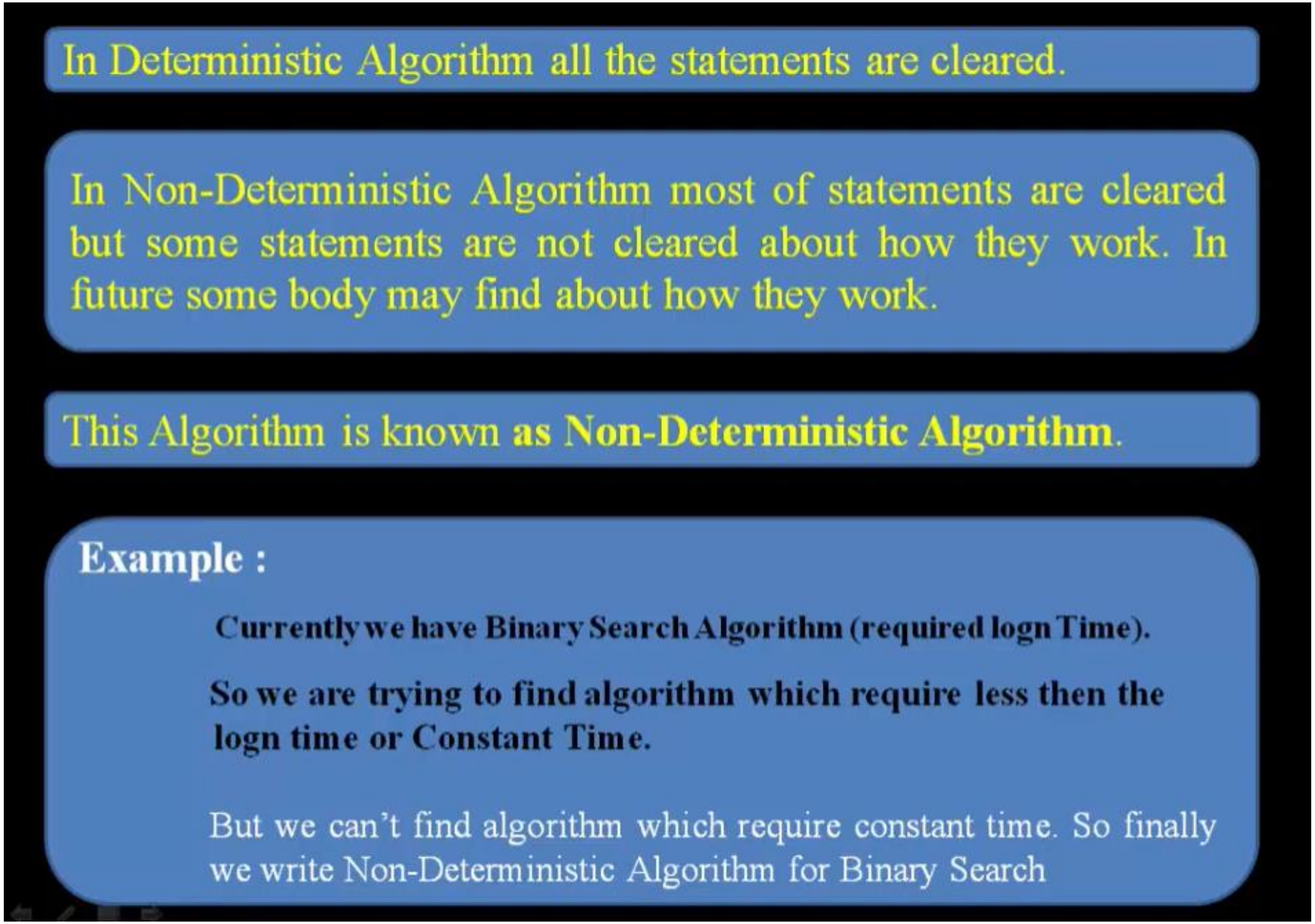
Travelling Sales man : $2^n$

Graph Coloring : $2^n$

So, we always try to solve Exponential Time algorithm in to Polynomial Time Algorithm.

So, we always try to solve Exponential Time algorithm in to Polynomial Time Algorithm.

If we can not write Polynomial Time Algorithm for Exponential Time Algorithm then we can write Non-Deterministic Algorithm (NP Class).



In Deterministic Algorithm all the statements are cleared.

In Non-Deterministic Algorithm most of statements are cleared but some statements are not cleared about how they work. In future some body may find about how they work.

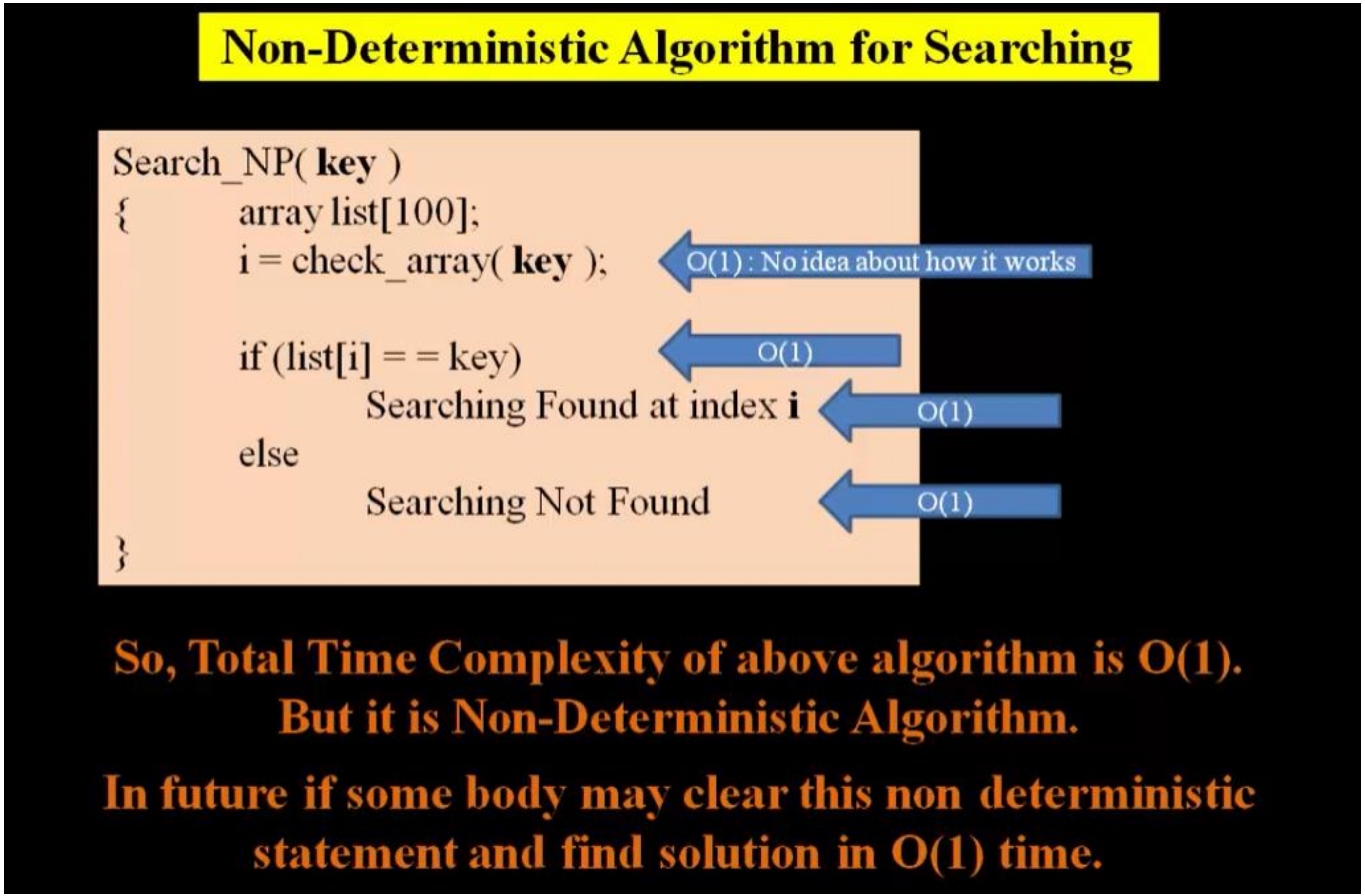
This Algorithm is known **as Non-Deterministic Algorithm**.

**Example :**

**Currently we have Binary Search Algorithm (required logn Time).**

**So we are trying to find algorithm which require less then the logn time or Constant Time.**

But we can't find algorithm which require constant time. So finally we write Non-Deterministic Algorithm for Binary Search



## Non-Deterministic Algorithm for Searching

```
Search_NP( key )
{       array list[100];
        i = check_array( key );        O(1) : No idea about how it works

        if (list[i] = = key)           O(1)
                Searching Found at index i        O(1)
        else
                Searching Not Found               O(1)
}
```

**So, Total Time Complexity of above algorithm is O(1).**
**But it is Non-Deterministic Algorithm.**

**In future if some body may clear this non deterministic statement and find solution in O(1) time.**

## Venn Diagram of P and NP

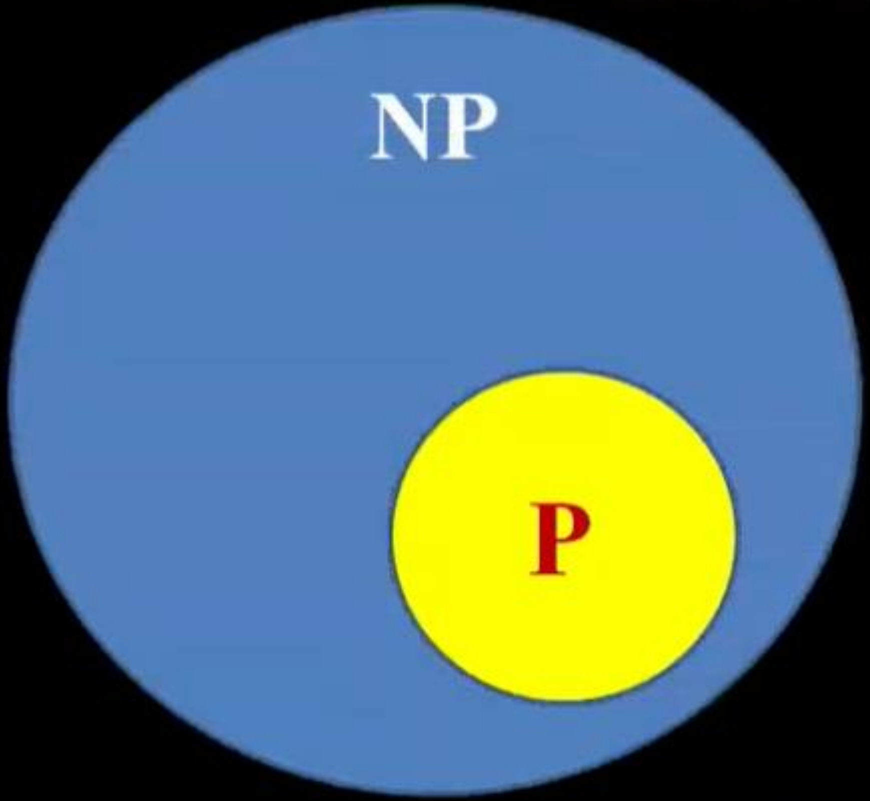
Ex. In past we had Sequential Search = O(n). At that time the algorithm which takes less then the n time was the Non-Deterministic Algorithm.

Now,
We have Binary Search algorithm which takes O(logn) time. So it is now Deterministic Polynomial Algorithm.

So, we have written Non-Deterministic Searching Algorithm which takes constant time O(1). But currently it is non-deterministic because we do not have idea how it is possible.

In future if any body find algorithm which take Constant time. Then again this non-deterministic algorithm will be Deterministic Polynomial Algorithm.

**Ex. In past we had Sequential Search = O(n). At that time the algorithm which takes less then the n time was the Non-Deterministic Algorithm.**

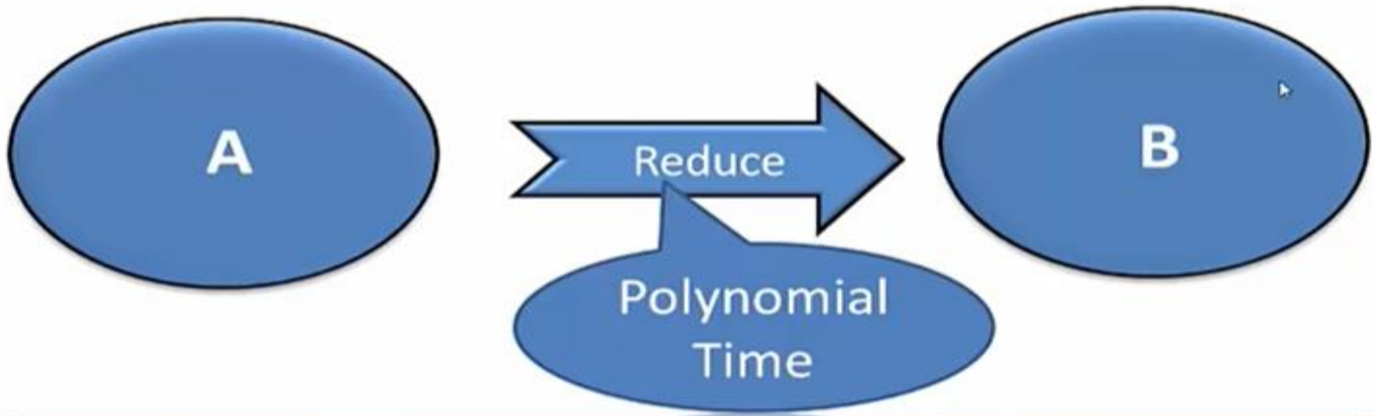


**Now,**
**We have Binary Search algorithm which takes O(logn) time. So it is now Deterministic Polynomial Algorithm.**

**So, we have written Non-Deterministic Searching Algorithm which takes constant time O(1). But currently it is non-deterministic because we do not have idea how it is possible.**

**In future if any body find algorithm which take Constant time. Then again this non-deterministic algorithm will be Deterministic Polynomial Algorithm.**

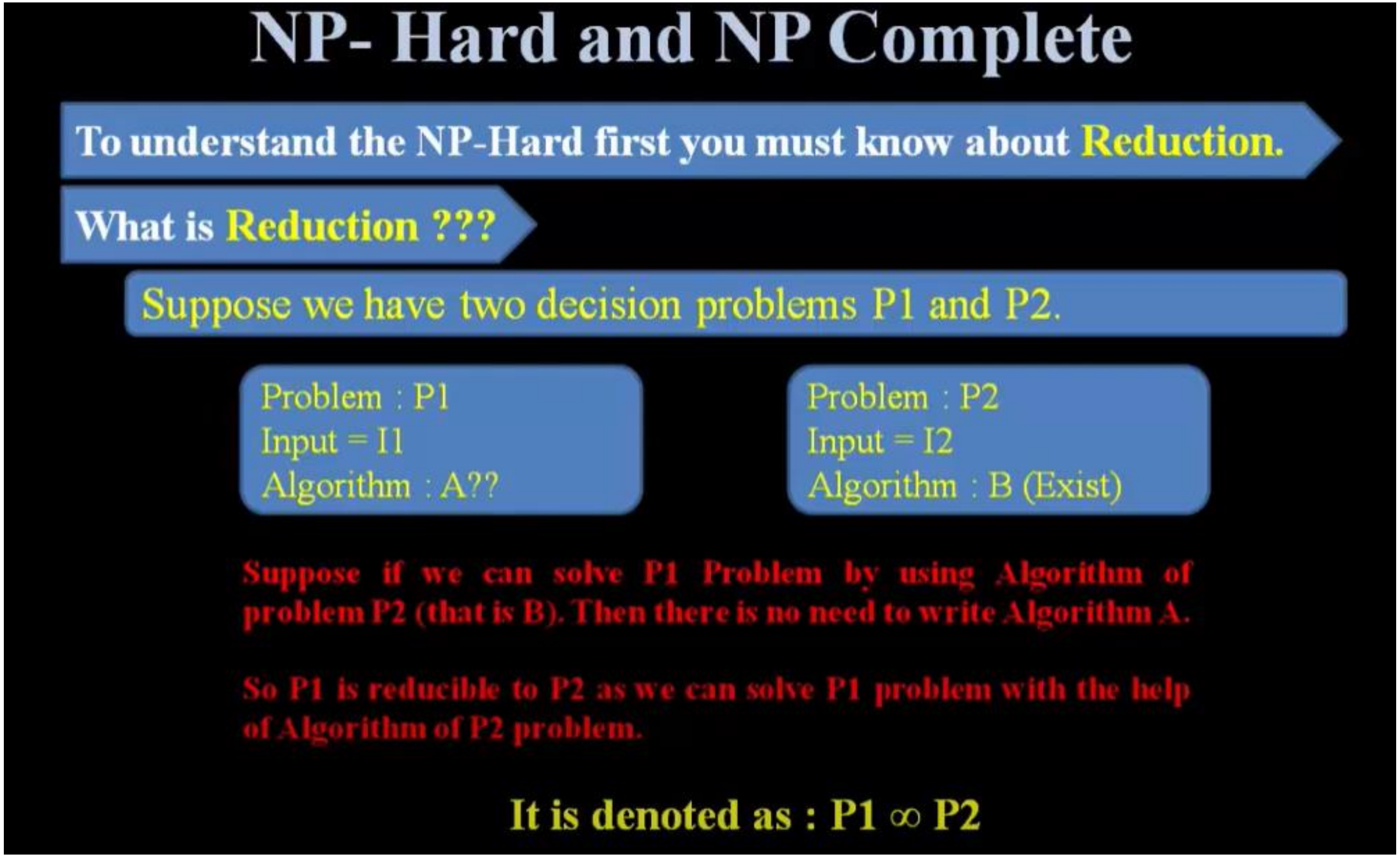
## Reduction:



Let A and B are two problems then problem A reduces to problem B iff there is a way to solve A by deterministic algorithm that solve B in polynomial time.

Properties:

1. if A is reducible to B and B in P then A in P.
2. A is not in P implies B is not in P



## NP- Hard and NP Complete

To understand the NP-Hard first you must know about **Reduction.**

What is **Reduction ???**

Suppose we have two decision problems P1 and P2.

Problem : P1
Input = I1
Algorithm : A??

Problem : P2
Input = I2
Algorithm : B (Exist)

**Suppose if we can solve P1 Problem by using Algorithm of problem P2 (that is B). Then there is no need to write Algorithm A.**

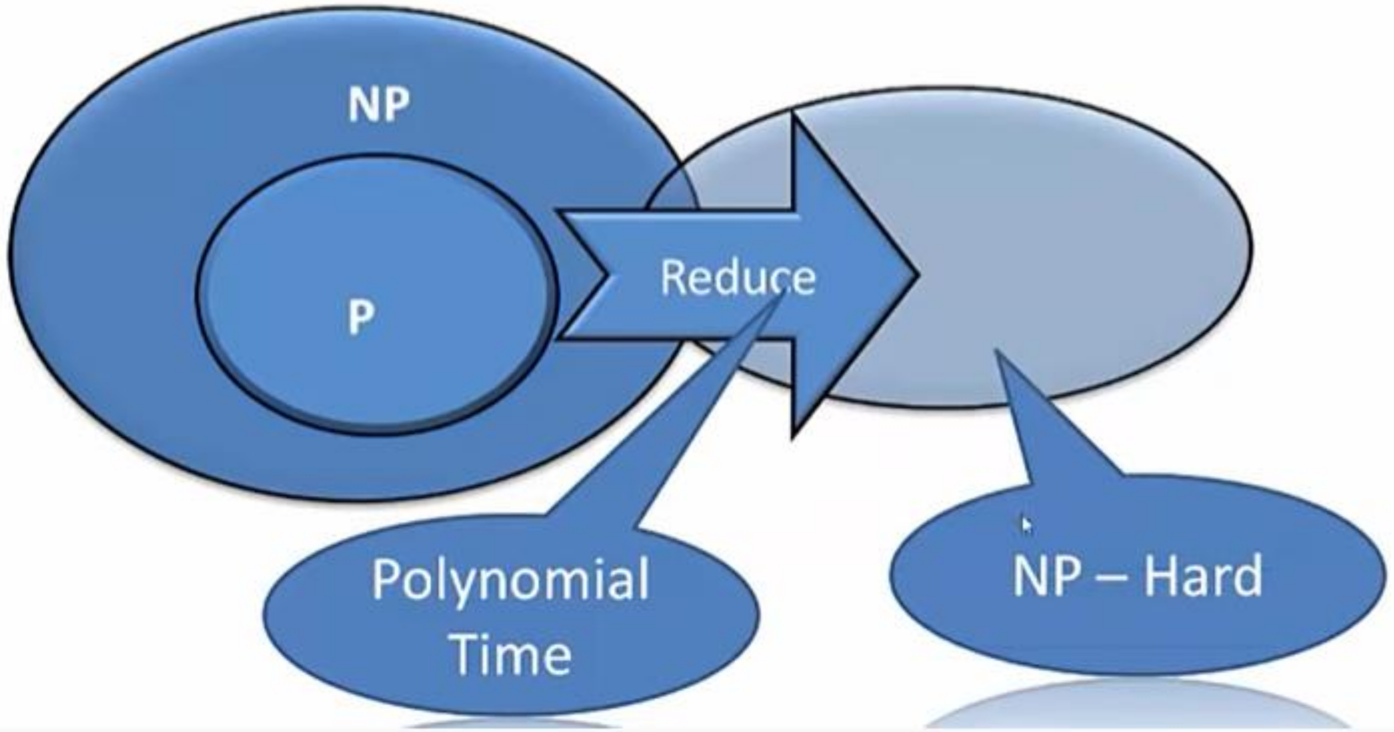
**So P1 is reducible to P2 as we can solve P1 problem with the help of Algorithm of P2 problem.**

**It is denoted as : P1 ∞ P2**

But this conversion cost must be Polynomial.

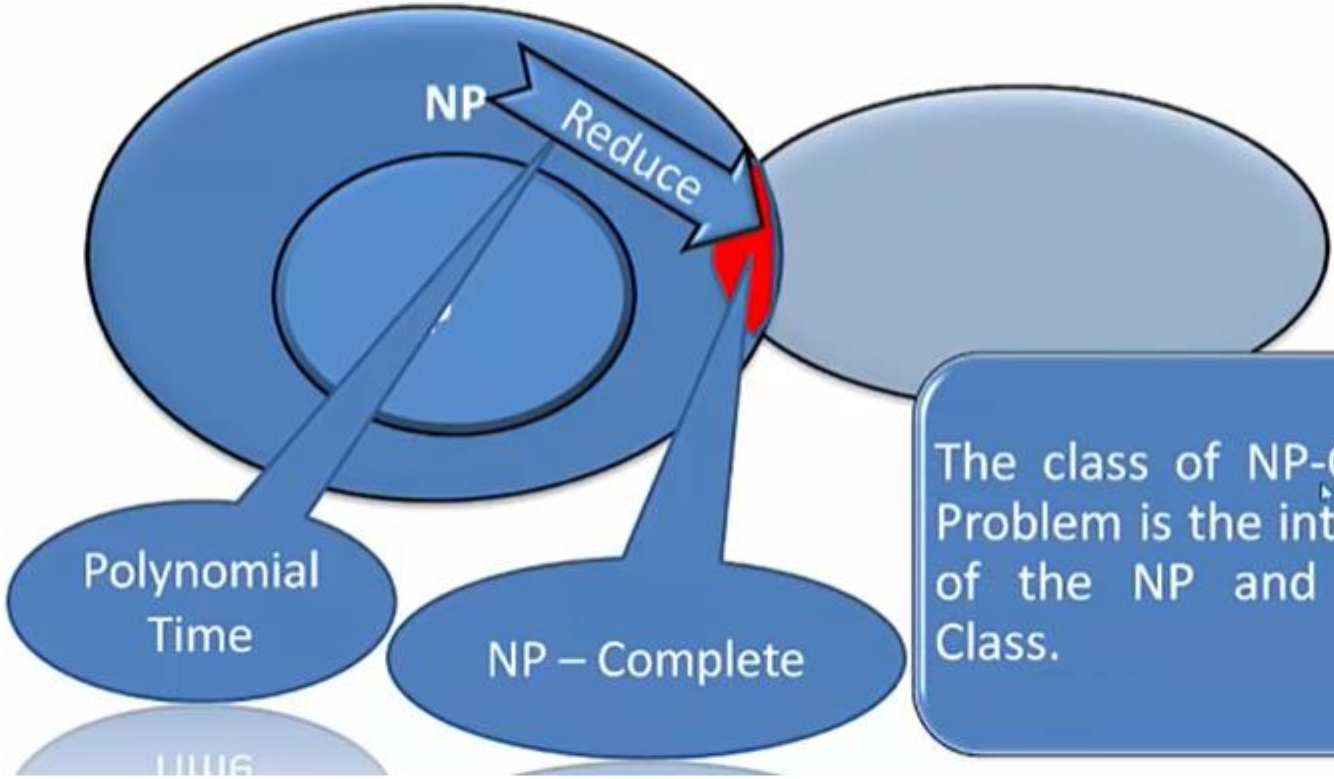
NP Hard Problem:

A Problem is NP-Hard if every problem in NP can be polynomial reduced to it.

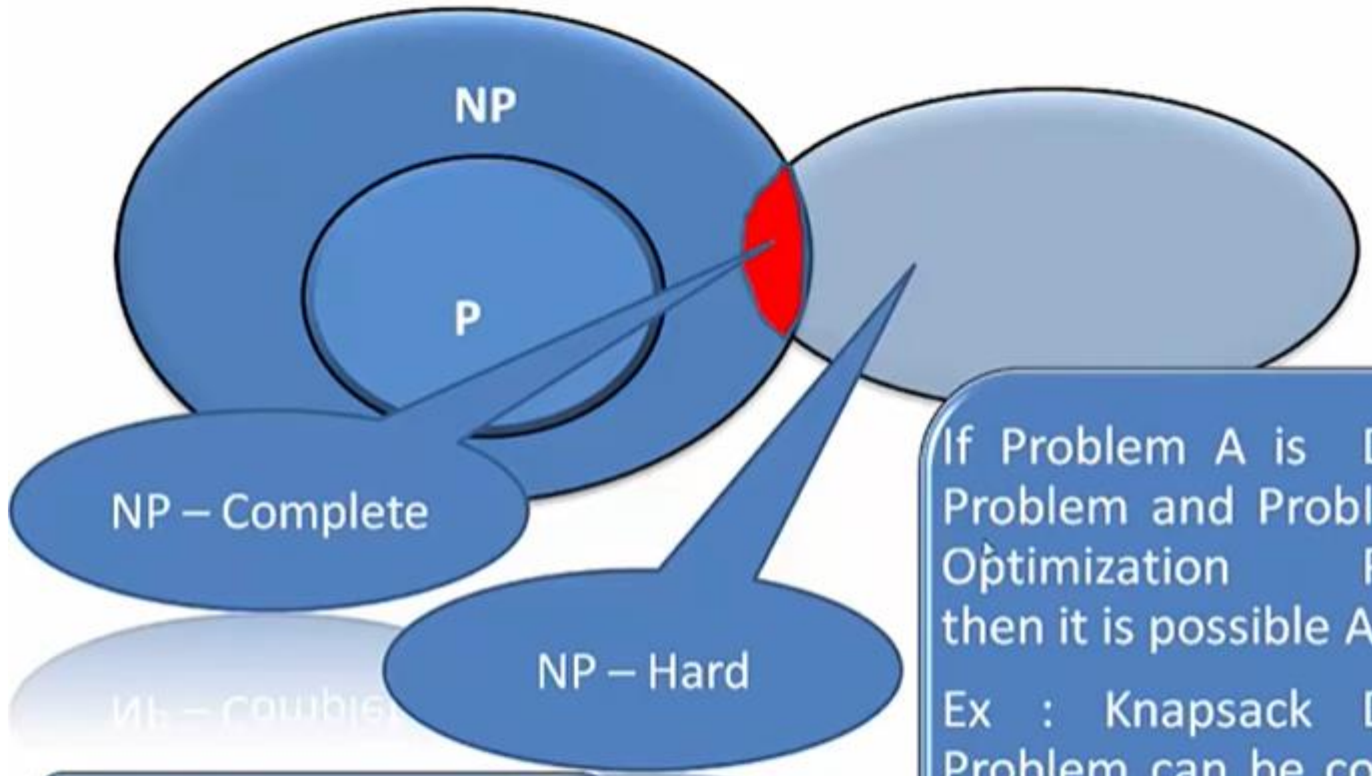


NP Complete Problem:

A Problem is NP-Complete if it is in NP and it is NP-Hard.



The class of NP-Complete Problem is the intersection of the NP and NP-Hard Class.

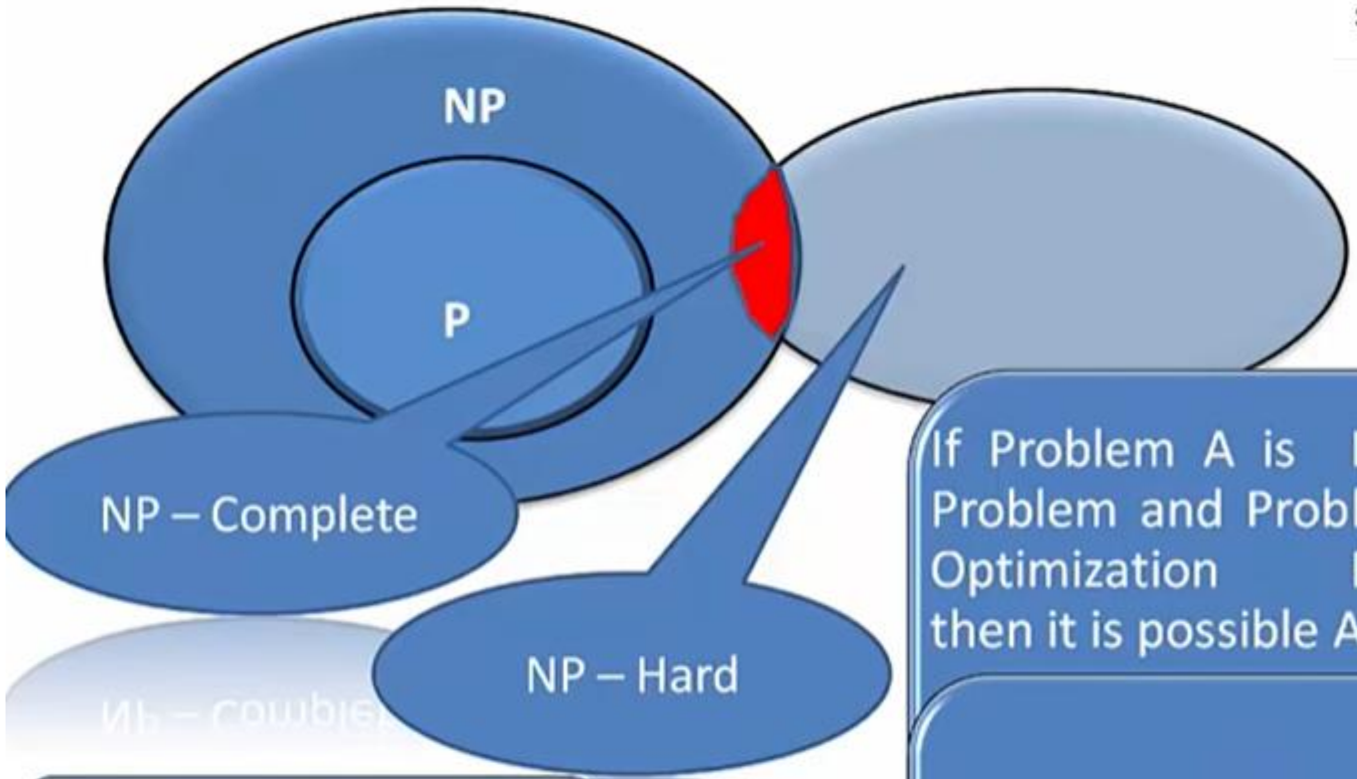


NP Complete Problem: Decision Problems

NP Hard Problem: Optimization Problems

If Problem A is Decision Problem and Problem B is Optimization Problem then it is possible A $\propto$ B.

Ex : Knapsack Decision Problem can be converted to Knapsack Optimization Problem.



NP Complete Problem: Decision Problems

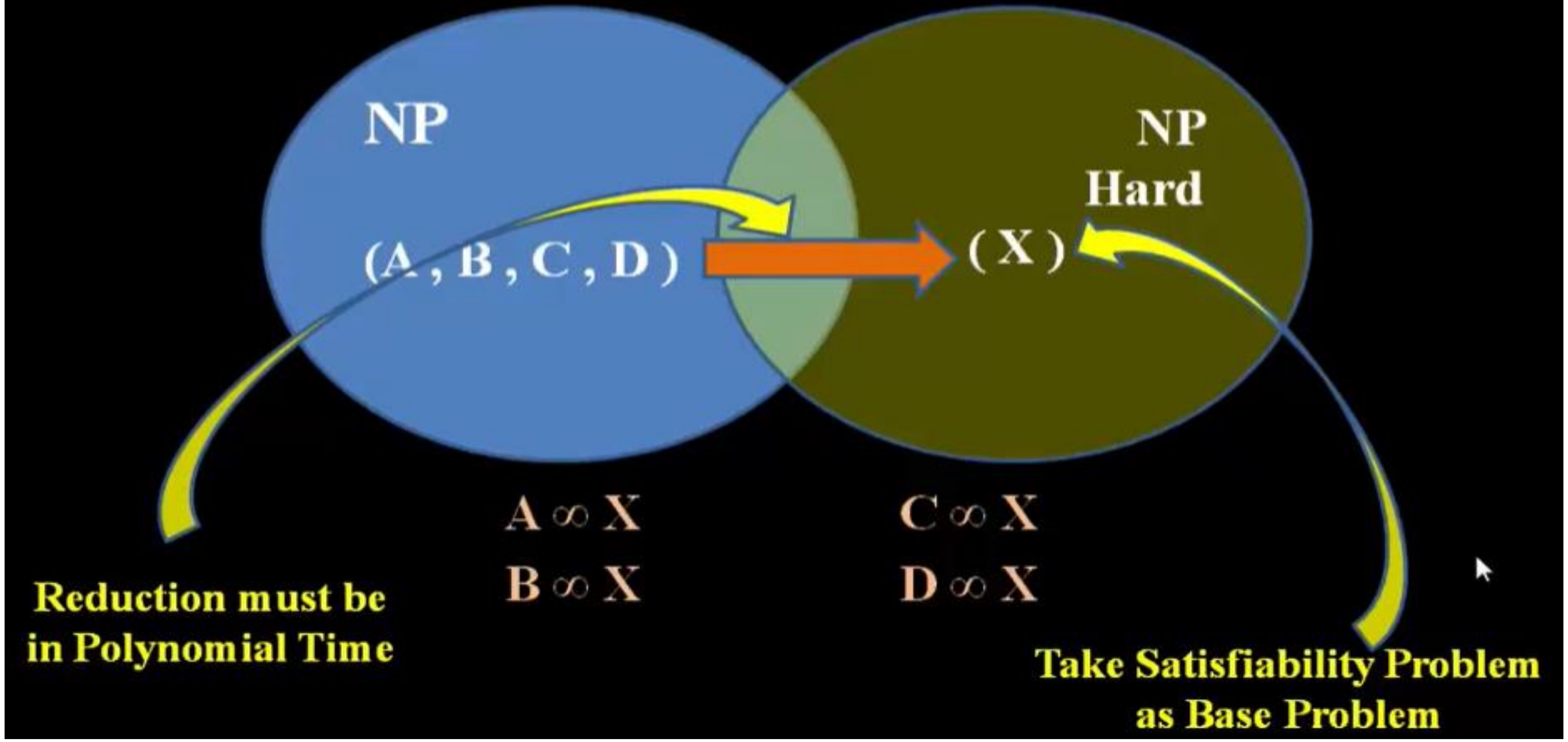
NP Hard Problem: Optimization Problems

If Problem A is Decision Problem and Problem B is Optimization Problem then it is possible $A \propto B$.

All NP-Complete Problems are NP- Hard but all NP-Hard Problems are not NP-Complete

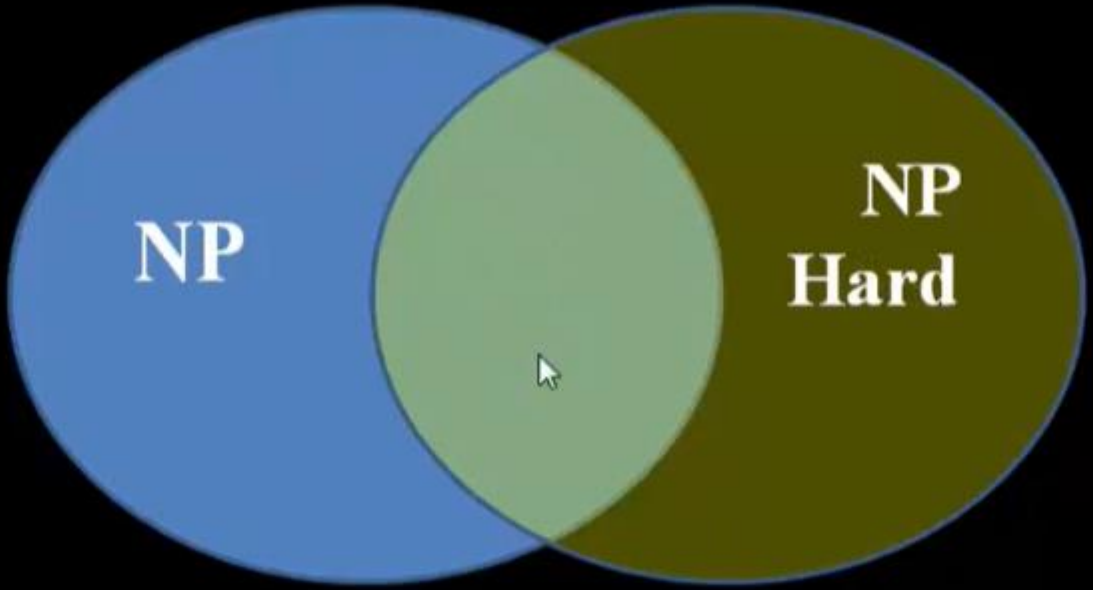
## • NP Hard :

A problem is **NP-hard** if all problems in NP are polynomial time reducible to it, even though it may not be in NP itself.

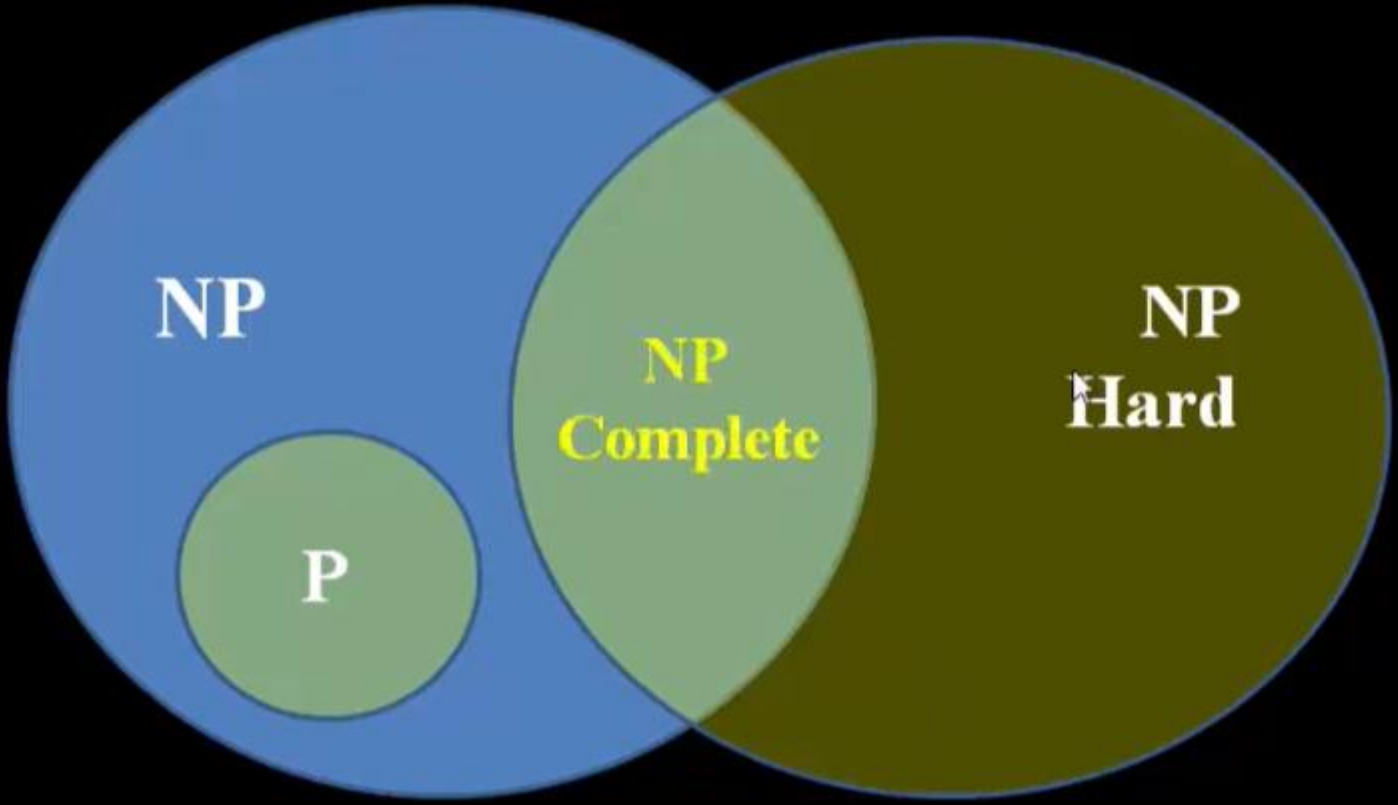

NP
NP Hard
(A , B , C , D )
( X )
A ∞ X
B ∞ X
C ∞ X
D ∞ X
Reduction must be in Polynomial Time
Take Satisfiability Problem as Base Problem

- **NP Complete :**

Any problem is NP-Complete if it is a part of both NP and NP-Hard Problem



## Venn Diagram of P, NP, NP-Hard, NP Compete


NP
P
NP Complete
NP Hard