P, NP, NP-Hard, NP-Complete

1) Deterministic algo. : No choice
2) Non-deterministic algo. : Choice, success, failure

Majority of the scientists & researchers are ~~a res~~ researching on this topic

Polynomial time Algo. :	Non-Polynomial time Algo. :
Linear search - n	0/1 Knapsack - 2^n
Binary search - $\log n$	TSP - 2^n
Insertion sort - n^2	Sum of subset - 2^n
Merge sort - $n \log n$	Graph coloring - 2^n
Matrix Multiplication - n^3	Hamiltonian cycle - 2^n

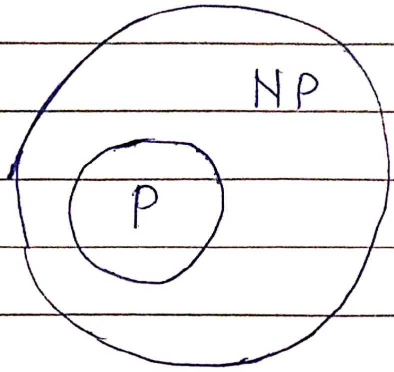
NP-Hard problems
We try to convert the TC of this algo. to linear TC

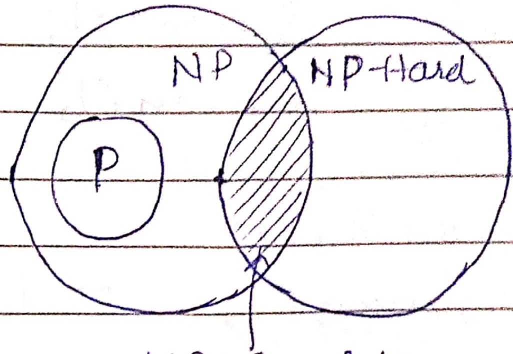
Deterministic Algo : Path of execution for Algorithm is same in every execution.

Non-Deterministic Algo. : Path of execution is not same for algorithm in every execution &

could take any random path for its execution.

NP → Polynomial TC & Non-deterministic Algo.
P → Polynomial TC & deterministic Algo.



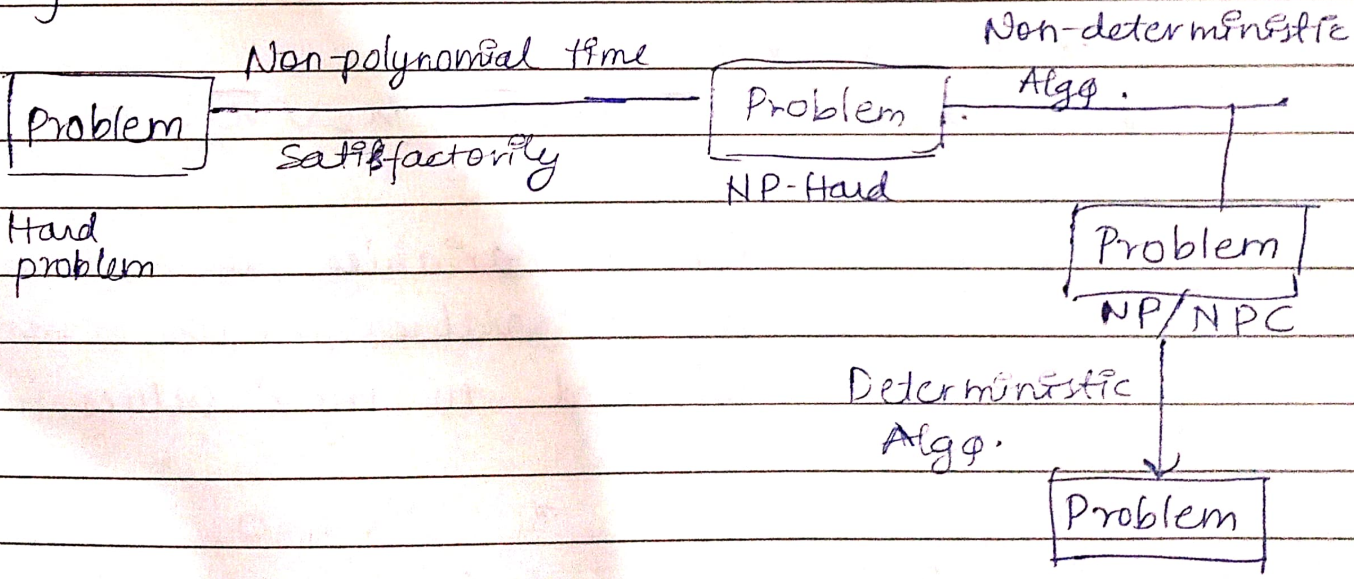


NP-Complete
(Solving NP-Hard problem using non-deterministic Algorithm in polynomial TC)

P ⊆ NP

P = NP (Cook's Theorem)

E.g.



There are 2 basic ideas to solve Non-polynomial time problem in polynomial time.

→ Non-Deterministic Algo. to solve them
→ Find similarity b/w problems

Non-Deterministic Algo:

~~Algorithm NSa~~

```
Algorithm NSearch (A, n, key) {
        j = choice()            — 1
    if (key = A[j]) {
            write(j);
            success();      — 1
    }
    write(0);
    failure();          — 1
}
```

$TC = O(1)$

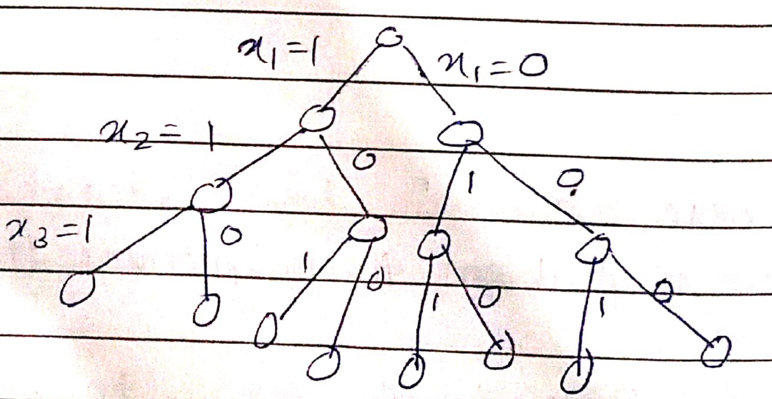
CNF Satisfiability

$x = \{x_1, x_2, x_3\}$

$CNF = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

for 3 variables we need to try 2^3 values, ∴ for n-variables we need to try 2^n values.



This is similar to 0/1 knapsack problem

$x_i = \{0/1, 0/1, 0/1\}$

NP-Hard Poly NP-Hard

Satisfiability I_1 Sat α 0/1 Knapsack

Sat α L_1
NP-Hard

Sat α L_1 $L_1 \alpha L_2$

NP-Hard

Cook's Theorem

Theorem: The satisfiability problem (SAT) is NP complete.

What is SAT?

A propositional logic formula ϕ is called satisfiable if there is some assignment to its variable that makes it evaluate to true.

→ $p \wedge q$ is satisfiable if $p = 1$ & $q = 1$

→ $p \wedge \neg p$ is not satisfiable

3SAT

A language is 3SAT $= \{\phi \mid \phi$ is satisfiable 3CNF formula$\}$

If it is in CNF & every clause has exactly 3 literals.

Theorem: SAT is NP-complete
→ Proof consists of 2 steps
1) Convert the ~~encod~~ execution of a polynomial time Non-deterministic Turing machine (NDTM) to a bunch of well formed formulae such that formulae satisfies iff machine accepts input.

2) Shows the sum of lengths of formulae is polynomial in the size of problem.

→ NP-Hard (L) – Can polynomially reduce any NP problem to L

→ NP-Complete – L ∈ NP

→ L ∈ NP → NDTM for L that runs in polynomial time

→ An NDTM is the only model we have for NP problem

→ SAT ∈ NP

→ ∴ If we can polynomially reduce an ~~arb~~ arbitrary polynomial NDTM to SAT. It means we have proven SAT is NP complete