UNIT - 5

NP-HARD, NP-COMPLETE

There are 2 types of groups to solve the problems. i) Problems that can be solved in polynomial time. (NP- Complete)

ii) Problems that cannot be solved in polynomial time (NP-Hard)

NP- Complete: A problem that is NP-complete has the property that it can be solved in polynomial time if & only if all other NP- complete problems can also be solved in polynomial time.

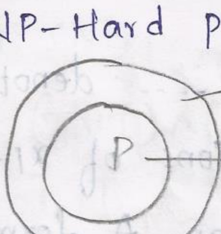
Ex: quick sort, binary search etc.

If an NP-Hard problems can be solved in polynomial time then all NP-complete problems can be solved in polynomial time.

All NP- complete problems are NP-Hard.

NP-complete $\subseteq$ NP-Hard.

But some NP-Hard problems are not NP-complete



NP (np-hard) (non-deterministic)

NP-complete (deterministic)

An algorithm with the property that result of every operation is uniquely defined is called <u>Deterministic</u>

Algorithms

→ An algorithms whose result of every operation of is not uniquely defined is called non-deterministic Algorithm

→ In order to specify non-deterministic problems, th we consider 3 functions

choice(s) : chooses one of element in given set s

failure() : returns unsuccessful completion

success() : returns successful completion

```
Ex:  j: = choice(i,n);
       if A[j]=x then
       {
         write(j);
         success();
       }
       write(0);
       Failure();
```

CNF ⇒ $\bigwedge_{i=1}^{k} c_i \;\; \bigvee l_{ij}$ → literals

DNF ⇒ $\bigvee_{i=1}^{k} c_i \;\; \bigwedge l_{ij}$ → literals

Satisfisability :

Let $x_1, x_2 \ldots$ denote boolean variables let $\bar{x}_i$ denotes negations of x_i. A literal is either a variable or its negation. A formula in the proposational calculus is an expression that can be constructed using literals and operations ~~and, or~~ "$\wedge$" or "$\vee$"

The symbol 'V' denotes OR ; '$\wedge$' denotes 'AND'. A formula in conjunctive normal form if & only if it is represented as $\bigwedge_{i=1}^{k} = C_i$ where C_i are the clauses each represented as $V_{L_{ij}}$ where L_{ij} are literals.

Literals

A formula is in disjunctive normal form if and only if it is represented as $\bigvee_{i=1}^{k} = C_i$ where C_i are clauses each represented as $\wedge_{L_{ij}}$ where L_{ij} are literals.

The satisfisability problem is to determine whether the formula is true for some assignment of truth values to variables.

Ex: $(x_1 \wedge x_2) \vee (x_3 \wedge \bar{x}_4)$ [DNF]

$(x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2)$

$(x_1 \wedge x_2) \vee (x_3 \wedge \bar{x}_4) \rightarrow DNF$

$(x_1 \vee \bar{x}_2) \wedge (x_3 \vee \bar{x}_4) \rightarrow CNF$

$x_1 = T$

$x_2 = T$ For DNF,

$x_3 = T$ $(T \wedge T) \vee (T \wedge T)$

$x_4 = F$ $T \vee T = T$

For CNF,

$(T \vee T) \wedge (T \vee F)$

$T \wedge T = T$

If CNF satisfies, then it is called satisfisability problem for CNF formulas. & vice-versa

Algorithm :- (Non-deterministic satisfiability)

Algorithm Eval (E, n)

{

for i := 1 to n do

x_i = choice (false, true);

if $E(x_1, x_2 \ldots x_n)$ then success();

else Failure();

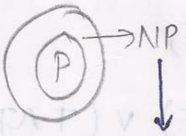
}

* Classes Of NP-HARD & NP-COMPLETE :-

(i) → P is a set of all decision problems soluble by the deterministic algorithms in polynomial time.

→ NP is a set of all decision problems soluble by the non-deterministic algorithms in polynomial time

$$P \subseteq NP \;, P = NP \cdot (\text{or})\; P \neq NP$$



Sorting, searching, all pairs shortest path

TSP, graph colouring.

ii) Let L_1, L_2 be problems. If problem L_1 reduces to L_2 i.e., $L_1 \propto L_2$ 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. i.e., if we have a polynomial time algorithm for

l_2 then we can solve l_1 in polynomial time.

$$l_1 \xrightarrow{\alpha} l_2 \qquad l_1 \alpha l_2$$

$$l_2 \xrightarrow{\alpha} l_3 \qquad l_2 \alpha l_3$$

$$l_1 \alpha l_3$$

$\therefore l_1 \xrightarrow{\alpha} l_3$ (According to transitive property)

iii) A problem L is NP-Hard if and only if satisfiability reduces to L.

A problem L is NP-complete if and only if L is NP-Hard & L $\in$ NP. ↳ NP-Hard & L $\in$ NP

iv) Two problems L_1 and L_2 are said to be polynomially equilant if and only if l_1 reduces to l_2 and l_2 reduces to l_1 i,e, a problem l_2 is NP-Hard since l_1 is some problem already known to be NP-Hard. Since it is using transitive relation it follows that satisfiability $\underset{\text{reduces}}{\alpha}$ l_1

$$l_1 \alpha l_2$$

then satisfiability α l_2.

COOK'S THEOREM:-

0\. * It "states that satisfisability is in P if and only if P = NP". According to definition of satisfiability we already seen that satisfiability is in NP. Hence, P = NP. i.e., satisfiability is also in P. In order to prove this following steps are considered.

1\. * To show how to obtain from any polynomial time non deterministic decision algorithms "A" and input "I" a formula Q(A,I) such that Q is satisfiable if and only if A has a successful termination with input i.

*. If length of I is 'n', and time complexity of A is p(n) for some polynomial time, then length of queue is given as $O(p^3(n)\log n) = O(p^4(n))$. The time needed to construct Q is $O(p^3(n)\log n)$.

2. *. A deterministic algorithm to determine outcome of A on any input I can be easily obtained.

→ Algorithm Z simply computes Q and then uses a deterministic algorithm for satisfiability problem to determine whether queue is satisfiable. If $O(q(m))$ is a time needed to determine whether a formula of length m is satisfiable then complexity of Z is $O(p^3(n)\log n) + q(p^3(n)\log n)$

→ If satisfiability is in P then q(m) is a polynomial function of M and complexity of Z becomes $O(r(n))$ for some polynomial R

→ Hence, if satisfiability is in P, then for every non-deterministic algorithm A in NP can obtain a deterministic Z in P so the above construction shows that if satisfisability is in P then P = N.