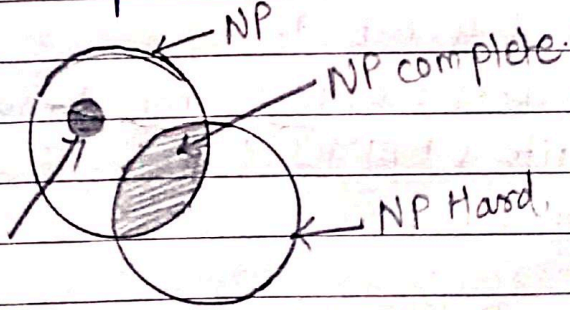
Tutorial No : 7

Name :
Date : / /

Q.1 Explain relationship between P, NP, NP- complete & NP-Hard problem.
Draw and explain commonly believed between P, NP, NP- complete and NP-Hard problem

→

1) P is the set of all decision problems solvable by deterministic algorithms in polynomial time.
2) NP is the set of all decision problems solvable by nondeterministic algorithms in polynomial time.
3) Since deterministic algorithms are just a special case of non-deterministic ones.
4) Conclude that P is subset of NP.
5) A problem that is NP-complete has the property. It can be solved in polynomial time, if & only if all other NP-complete problems can also be solved in polynomial time.
6) If an NP-Hard problem can be solved in polynomial time then all NP-complete problems can be solved in polynomial time.
7) All NP-complete problems are NP-Hard.
8) But some NP-Hard problems are not known to be NP-complete.
9) L is NP-Hard & L belongs to NP, then only A problem L is NP-complete.



Name :
Date : / /

Q. 2. What is non-deterministic algorithm? Explain non-deterministic search & sorting algorithms.

→

1) Algorithm contains some operations whose outcomes are not uniquely defined.
2) But limited to specified sets of possibilities it is allowed to choose any one of these outcomes.
3) Consider the problem of searching for an element x in a given set of elements A[1:n], n≥1. We are required to determine an index j such that A[j] = x or j=0 x is not in A. A non-deterministic for this is algorithm

4)
```
j := choice(1, n);
if A[j] = x then { write(j); success(); }
write(0); failure();
```

5) Sorting algorithm

```
Nsort (A, n)
{
    for i := 1 to n do B[i] = 0;
    For i := 1 to n do
    {
        j = choice (1, n);
        if B[j] ≠ 0 then failure ();
        B[j] := A[i];
    }
    for i := 1 to n-1 do
        if B[i] > B[i+1] then failure ();
    Write (B[1:n]);
    success();
}
```

Name :
Date : / /

Q.3. Write note on Cook's theorem.

→ 1) Satisfiability is in P if and only if P = NP
2) From satisfiability definition - satisfiability is in NP if P = NP the satisfiability is in P.
3) We have to show how to obtain from any polynomial time non-deterministic decision algorithm A & input I
4) A formula Q (A, I) such that Q is satisfiable.
5) iff A has a successful termination with input I.
6) Deterministic algorithm z to determine the outcome of ~~op~~ A on any input I can be obtained.
7) Algorithm z compute Q and then determine whether Q is satisfiable.
8) If Q obtained by deterministic algorithm z is satisfiable then P = NP.
9) As in both types of algorithms deterministic & non-deterministic for that problem satisfiability reduce to it.
10) Means - Matching truth table values at both sides.
It can be shown that if satisfiability is in P then P = NP

Q.4. Define / Differentiate / compare the following.
a) Deterministic & non-deterministic algorithm.
b) Decision & optimization problem.
c) P and NP problem.
d) NP-Hard & NP-Complete problems.
e) satisfiability & reducibility.

→

a) Deterministic & non-deteterministic algorithm :-
1) Deterministic -
Result of every operation is uniquely defined.

2) Non-deterministic :
Algorithm contains some operations whose outcomes are not uniquely defined. But limited to specified sets of possibilities it is allowed to choose any one of these outcomes.

3) Quick sort is example of deterministic polynomial time algorithm.

4) Randomized quick sort is example of non-deterministic polynomial time algorithm.

b] Decision & optimization problem :

1) Decision problem -
   i) Any problem for which the answer is either 0 or 1
   ii) decision problems are NP-complete problems.
   iii) Consider knapsack problem have 8 input m knapsack capacity.
   - it is very hard to calculate Optimized sol^n.
   - Is the 8rd element is part of solution.
     It is decision problem, it can be solved polynomial time.

2] Optimization problems :
   i) Any problem that involves the identification of an optimal value of a given cost function.
   ii) Optimization problem have complexity 2^n, n^n.
   iii) They are very hard to calculate hence it is called NP-Hard.
   iv) eg. knapsack problem with input 8 values.

Name :
Date : / /

c] P and NP problems -

i) P is the set of all decision problem solvable by deterministic algorithms in polynomial time.

2) NP - is the set of all decisions problems solvable by nondeterministic algorithms in polynomial time.

e] Satisfiability & Reducibility :

i) It is easy to obtain a polynomial time non-deterministic algorithm that terminates successfully iff a given propositional formula is satisfiable.

2) Let L_1 & L_2 be problems.

- Problem L_1 reduces to L_2, $L_1 \propto L_2$
- iff there is a way to solve L_1 by a deterministic algorithm that solves L_2 in polynomial time.
- Reducibility is transitive.

Q5. Explain NP-Hard graph problems.

→ 1) Pick a problem L_1 already known to be NP-Hard.

2) Show how to obtain an instance I' of L_2 from any instance I of L_1.

3) Show that from the solution of I' we can determine the solution to instance I of L_1.

4) Conclude from step (2) that L_1 is reducible to L_2.

5) Conclude from steps (1) & (3) & the transitivity of reducibility, that L_2 is NP-Hard.

Q.6. List and explain NP Hard graph problems.

→ List of NP-Hard graph problems -

1) Clique decision problem.

2) Node cover decision problem.

3) Chromatic number decision problem.

4) Directed Hamiltonian cycle.
5) Travelling salesperson Decision problem (TSP)
6) AND/OR Graph Decision Problem (AOG)

Q.6. i] Explain Clique decision problem & Node cover decision problem.
ii] Assume that node cover decision problem is NP-Hard.
iii] Prove that clique decision problem is also NP Hard using reducibility.
iv] Show that, the clique decision problem (CDP) is NP reducible to decision problem.

→

i] Clique decision problem -
1) Suppose a graph G = (V, E)
2) In this problem we have to find maximum subgroup have each vertex connected to each other.
3) Optimization problem to find maximum clique from graph.
4) Decision problem - determine weather G has a clique of size at least K for some given K.

Node Cover :
1] A set S subset of V
2] is a node cover for a graph G = (V, E)
3] if and only if all edges in E are incident to at least one vertex in S.
4] The size |S| of the cover is the number of vertices in S.

① Let G = (V, E) & K define an instance of CDP

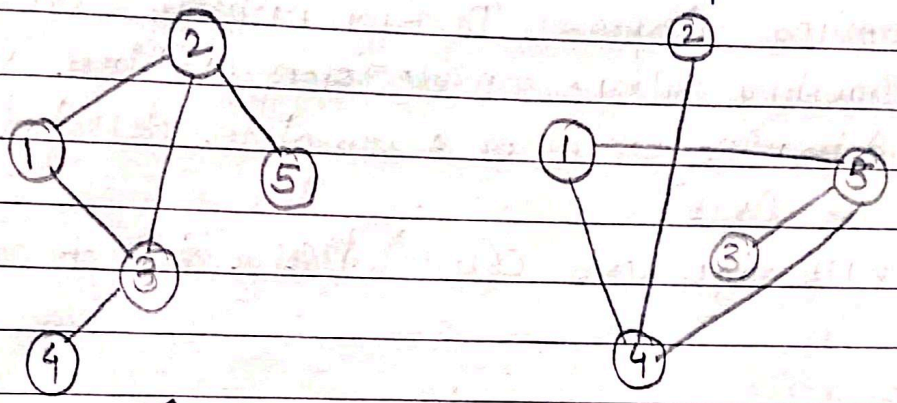
2) Assume that $|V| = n$

3) We construct a graph G' such that G' has a node cover of size at most $n-k$ iff G has a clique of size at least k

4) Graph G' is given.

5) by $G' = (V, \bar{E})$, where $\bar{E} = \{\{u, v\} \in V, v \in V \text{ \& } (u,v) \notin E\}$

6) The set G' is known as the complement of G.



G

7) G' has a node cover $\{4,3\}$ of size 2.

8) Since every edge of G' is incident either on the node 4 or on the node 5.

9) G has a clique of size $5-2=3$

10) Consisting of the nodes 1, 2 & 3.

iii] 1) The clique decision problem is reducible to node cover decision problem.

2) If CDP is NP-Hard

3) by the transitivity of reducibility.

4) Conclude that NCDP is also NP Hard.

Q.7. Explain Directed Hamiltonian Cycle (DHC) is reducible to the travelling salesperson decision problem (TSP)

Explain DHC & TSP

Assume TSP = NP Hard then prove DHC = NP Hard.

Name :
Date : / /

→ Directed Hamiltonian Cycle (DHC) -

1) A directed Hamiltonian cycle in a directed graph $G=(V,E)$ is directed cycle of length $m = |V|$.
2) The cycle goes through every vertex exactly once & then returns to the starting vertex.
3) The DHC problem is to determine weather G has a directed Hamiltonian cycle.

Travelling Salesperson Decision problem (TSP).

1) Traveling salesperson decision problem is to determine wheather a complete directed graph $G = (V, E)$
2) with edge costs $C(u,v)$ has a tour of cost at most 5.

DHC is reducible to the travelling salesperson decision problem (TSP)

1) From directed graph $G = (V, E)$ construct the complete directed graph $G' = (V, \bar{E})$
2) where, $E' = \{\{i,j\} \mid i \neq j\}$ & $C(i,j) = 1$ if $\langle i,j \rangle \in E$
3) $(i,j) = 2$ if $i \neq j$ & $\langle i,j \rangle \notin E$
4) Clearly G' has a tour of cost at most n iff G has a directed Hamiltonian cycle.

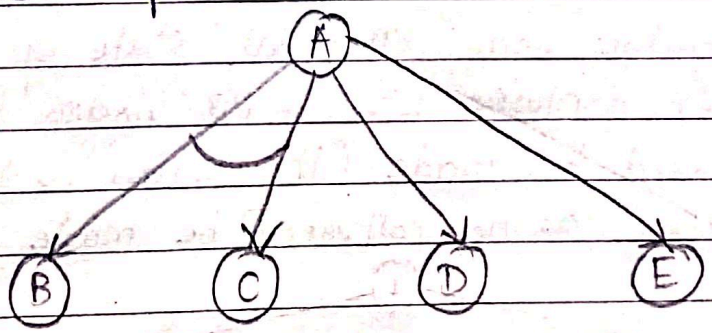
1) Assume TSP is NP Hard.
2) DHC is reducible to TSP
3) TSP is reducible to DHC.
4) By the transitivity of reducibility DHC is also NP Hard.

Name :
Date : / /

Q.8 Explain AND/OR graph decision problems.

→

1) Complex problems can be broken down into sets of sub problems such that
2) The solution of all or some of these results in the solution of the original problem.
3) These sub problems can be broken down further into sub-sub-problems & so on.
4) Until the only probems remaining are sufficiently primitive as to be trivially solvable
5) This breaking down of complex problem into several subproblem can be represented by a directed graphlike structure.
6) In which nodes represent problems & descendants of nodes represent the subproblems associated with them problem can be solved by solving either both the sub-problems B & C or the single subproblem D or E.



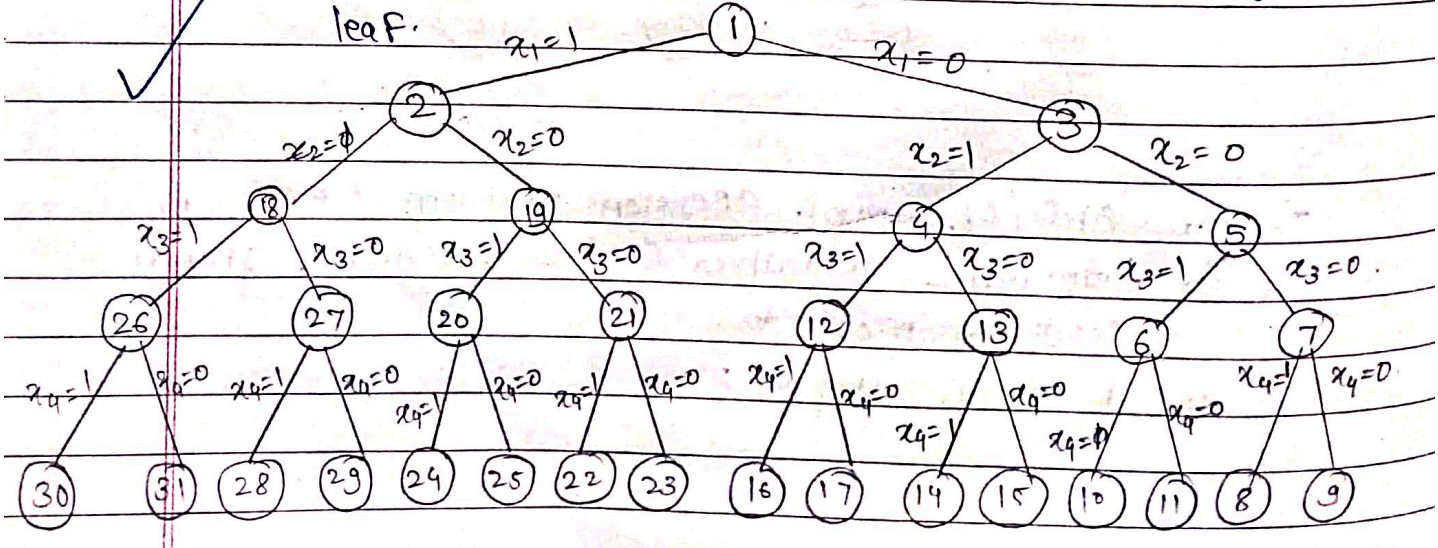
7) The AND/OR graph decision problem (AOG) is to determine whether G has a solution graph of cost at most k.
For k given input.

Name :
Date : / /

Q.9. Explain NP-Hard scheduling problems.

→

1) Sum of subset problem.
2) Suppose we are given n distinct positive numbers we desire to find all combinations of these numbers whose sums are m.
3) This is called the sum of subsets problem.
4) NP- Hard problem.
5) eg. Fig. shows the portion of the state space tree generated by function sumofsub while working on instance $n=6$, $m=30$, & $w[1:6] = \{5, 10, 12, 13, 15, 18\}$. The rectangular nodes list the values of s, k, & r on each of the calls to the sumofsub. Circular nodes represent points at which subsets with sums m are printed out. At nodes A, B & C the output is respectively $(1,1,0,0,1)$, $(1,0,1,1)$ & $(0,0,1,0,0)$. Note that the tree of figure contains only 23 rectangular nodes. The full state space tree for $n=6$ contains $2^6 - 1 = 63$ nodes from which calls could be made (this count excludes) the 64 leaf nodes as no call need be made from a leaf.



Name :
Date : / /

Q.10. Explain NP Hard code optimization problems.

→

1) The function of a compiler is to translate programs written in some source language into an equivalent assembly language or machine language program.
2) Thus, the C++ compiler on the sparc lo translates C++ programs into the machine language of this machine
3) We look at the problem of translating arithmetic expressions in a language such as C++ into assembly language code.
4) The translation clearly depends on the particular assembly language (and hence machine) being used.
5) To begin, we assume a very simple machine model.
6) We call this model machine A.
7) This machine has only one register called the accumulator.
8) All arithmetic has to be performed in this register.
9) ⊗ represents 4 binary operators such as +, -, *, &
10) Then the left operand must be accumulator.