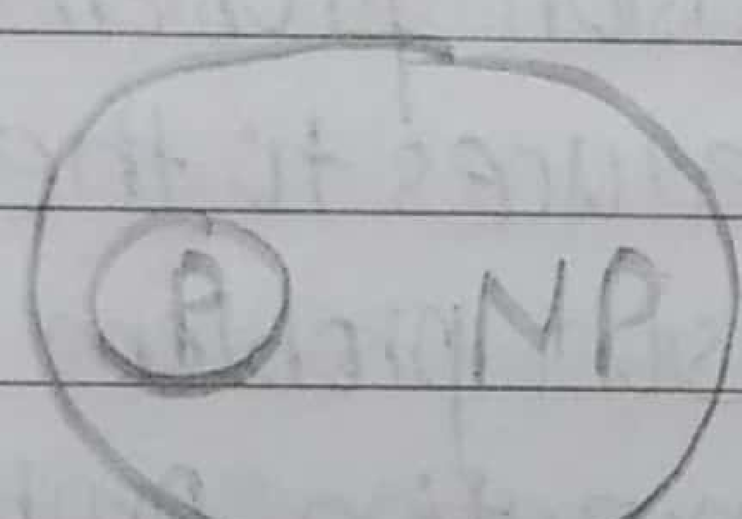
Page No.:
Date:

Tutorial No.-07

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

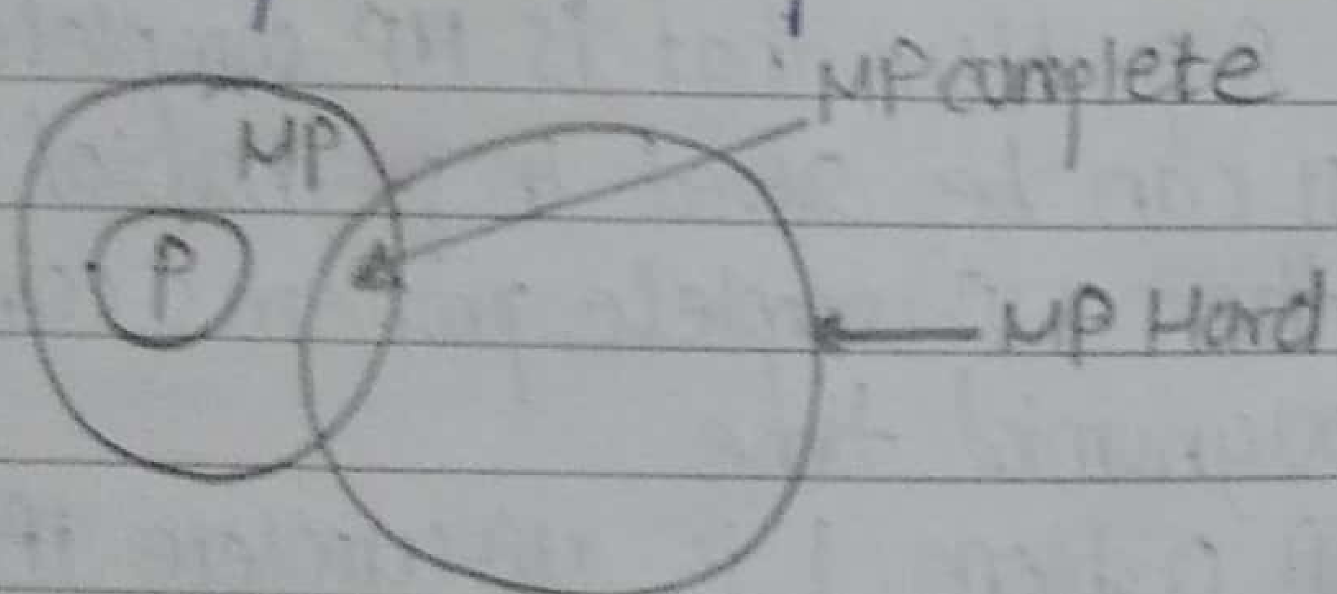
→ P is the set of all the problems solvable by deterministic algorithm in polynomial time.
NP is set of all the problems solvable by non-deterministics algorithm in polynomial time.
Since deterministic algorithm are just a special case of nondeterministic ones conclude that P is subset of NP.



A problem that is NP-complete has the property.
- It can be solved in polynomial time iff all other NP-complete problems can also be solved in polynomial time
- A problem L is NP-complete iff
  L is NP-hard and L belongs to NP.
- NP-complete problems are less hard than NP-hard. problem S. Smaller instance of NP-Hard problem Can form NP complete problem.
- Upto certain extent, it can be solved in polynomial time Eg. Decision problem instance of optimization Problem.
If an NP-hard problem can be solved in polynomial time then all NP-complete problems can be

solved in polynomial time.
- All NP- complete problems are NP-hard but some NP-hard problems are not known to be NP complete.
- A problem L is NP-hard iff satisfiability reduces to L
- A problem L is NP-hard iff it can be reduced satisfibality problem (SAT). As is NP-hard problem is also NP-hard.

Only decision problem can be NP-complete 0 or 1. As optimization problem may be NP-Hard iff L_1 is a decision problem and L_2 an optimization problem reduces to the knapsack.
Eg knapsack decision problem reduces to the knapsack optimization problem



Commonly belived NP-complete NP-hard problem relationship among P, NP.

Q.3 Write note on Cook's Theorem
→ Satisfiability is in P iff if P=NP from satisfiability definition - satisfiability is is NP
If P=NP then satisfiability is in P
We have to show to obtain from any polynomial time nondeterministic algorithm A and input I, a formula Q(A,I) such that Q is satisfiable iff A has a successful termination with input I.
Deterministic algorithm Z to determine the outcome of A any input I can be obtained.
Algorithm Z computes Q and then determine wheather Q is satisfiable
If Q obtained by determinstic algorithm wheather Q is satisfiable.
It's on both types of algorithms deterministic and nondeterministic for that problem satisfiability reduce to it.
Means-matching truth table values at the both sides.
It can be shown that if satisfiability is in P, then P=NP.

Q.4 Define/ Differentiate / Compare the following:
(a) Deterministic and non-deterministic algorithm.
→ Deterministic Algorithm:
- Result of every operation is uniquely defined.
- P is the set of all the problems solvable by deterministic algorithms in polynomial time.
Eg Normal quick sort.

Non-deterministic algorithm:
- Algorithm contain some operations whose outcomes are not uniquely defined.
- But limited to specified set of possibilities

- It is allowed to choose any one of these outcomes Eg. Randomized Quick Sort.

(b) Decision and Optimization Problem

Decision Problem

- Any problem for which the answer is either 0 or 1.

Optimization Problem

- Any problem that involves the identification of an optimal value of a given Cost function.

(c) P and NP Problems.

- P is the set of all the problems solvable by deterministics algorithm in polynomial time.
- NP is set of all the problems solvable by a non-deterministic algorithm in polynomial time.
- Since deterministic algorithms are just a special case of non-deterministic ones conclude that P is subset of NP optimization problems are very complex.

(d) NP-Hard and NP-Complete problems:

- If an NP-hard problem can be solvable in polynomial time then all NP-complete problems can be solved in polynomial time

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

All NP-complete problems are NP-hard but some NP-hard problems are not known to be NP-Complete.

(e) Satisfiability and Reducibility:

Reducibility

- ~~Let L~~
- It is easy to obtain in polynomial time non-deterministic algorithm that terminates successfully if and only if a given proportional formula is satisfiable.
- Let L_1 and L_2 be problems:

Problem L_1 reduces to L_2 ~~&~~ L_1 iff if there is a way to solve L_1 by a deterministic algorithm that solves L_2 in polynomial time

- Reducibility is transitive.

Q.2 What is nondeterministic algorithm? Explain non-deterministic search and sorting algorithm.

1. Algorithm contains some operation whose outcome are not uniquely defined.
2. But limited to specified sets of possibilities it is allowed to choose any one of these outcome
3. Consider the problem of searching for an element x in a given set of elements $A[1:n]$, $n \geq 1$. We are required to determine an index j such that $A[j] = x$ or $j = 0$ if 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
```
M sort (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[j];
   }
   for i=1 to n-1 do
      if B[i]>B[i+1] then failure();
   while (B[1:n]);
   success();
}
```

Q.5 Explain NP-Hard graph problem.

→ 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 reduciable to L_2

⑤ Conclude from steps ① & ③ and the transitivity of reducibility, that L_2 ~~...~~ is NP hard.

The above strategy is used to find a NP-Hard graph problem.

Q.5 List and Explain NP-Hard graph problems.

→ List of NP-Hard graph problem.

1) Clique Decision Problem
2) Node Cover Decision Problem (NCDP)
3) Chromatic Number Decision Problem (CNDP)
4) Directed Hamiltonian Problem.
5) Travelling Salesperson Decision Problem (TSDP)
6) AND/OR graph Decision Problem (AOG)

Q.6 ① Explain Clique decision problem and Node cover decision problem

(ii) Assume that Node cover decision problem is NP-Hard, prove that Clique decision problem is also NP-Hard using reducibility

(iii) Show that, the clique decision problem (CDP) is reducable to the node cover decision problem

→ (i)

Clique Decision Problem

① Suppose a graph $G=(V,E)$

② In this problem we have to find maximum subgraph 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 of S.

④ The size |S| of the cover is the numbers of vertex in S.

iii) MCDP ⟶ NP-Hard

prove CDP ⟶ NP-Hard

① Let $G=(V,E)$ and k defines 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 if and only if G has a clique of size at least k.

④ Graph G' is given by $G'=(V,\bar{E})$ where $\bar{E} = \{ \{u,v\} \mid u \in V, v \in V \ \& \ (u,v) \notin E \}$

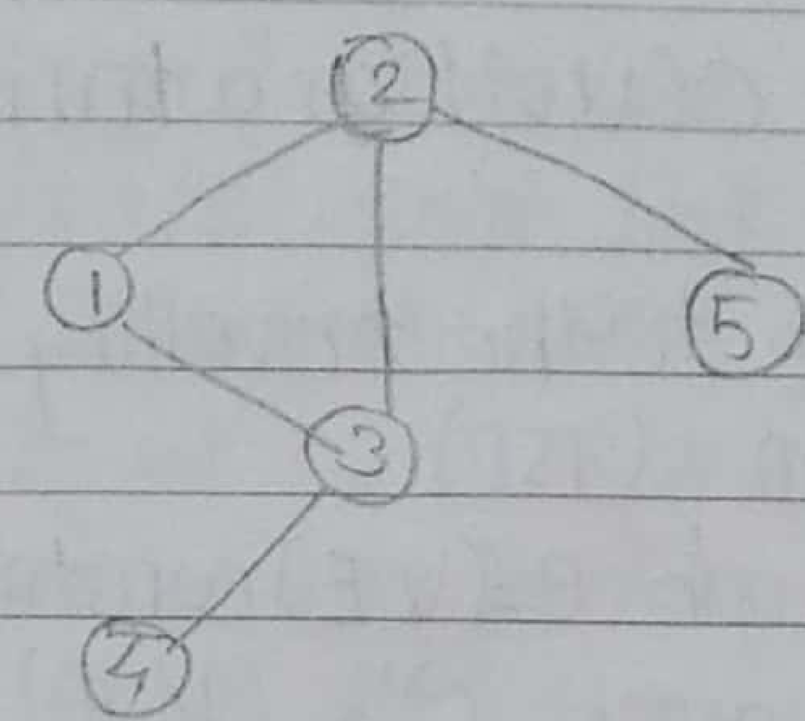
⑤ The set G' is known as complement of G

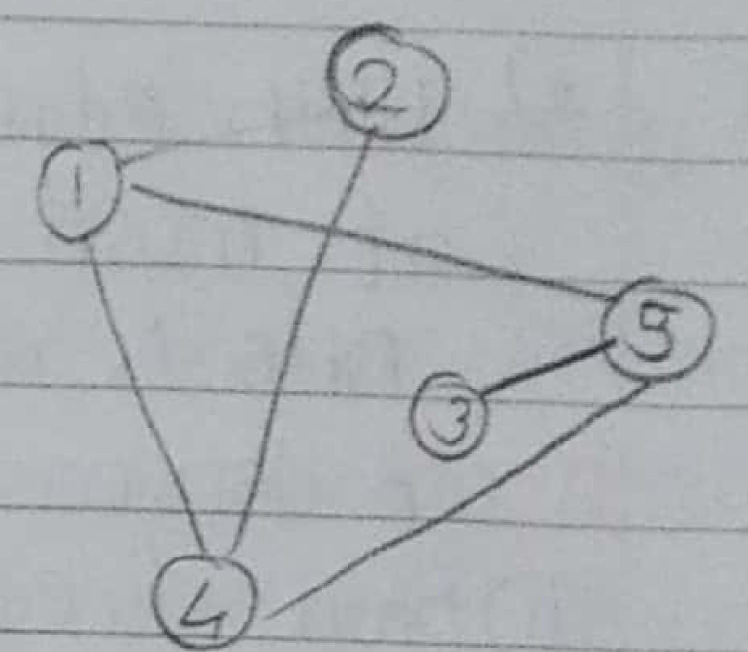
⑥ G' has a node cover {4,3} of size 2

⑦ Since every edge of G' is incident either on the node 4 or on the node G'.

⑧ G has a clique of size 5-2=3

⑨ Consisting of the node 1,2,&3.





ii) 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 MCDP is also NP-Hard

Q.7 ① Explain Directed Hamiltonian Cycle (DHC) is reducible to travelling salesperson decision problem

⟶ i) Explain DHC and TSP

ii) Assume TSP = NP Hard then prove DHC ≠ NP Hard

⟶ 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 starting vertex.

③ The DHC problem is to determine wheather G has a directed Hamiltonian cycle

· Travelling salesperson decision problem (TSP)

1) Travelling 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

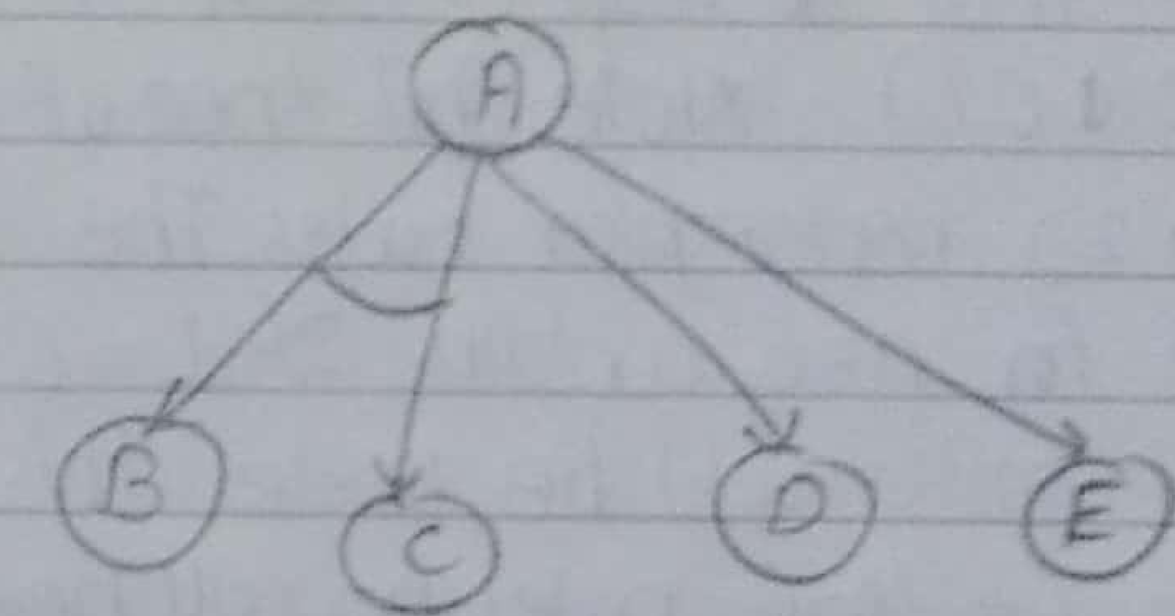
DHC is reducible to the travelling salesperson decision problem (TSP)

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

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

Q 8 Explain AND/OR graph decision problem

→ 1. Complex problems can be broken down into series of sub problems such that
2. The solution of all or some of these results in the solutions of the original problem
3. These subproblems can be broken down further into sub-sub-problems
4. Until the only problems remaining are sufficiently primitive as to be ~~to~~ trivially solvable
5. This breaking down of complex problem into several subproblems can be represented by a directed graph like structure
6. In which nodes represents problems & descendents of node represent the subproblems associated with them problem can be solvable by solving either both the sub-problem B and C or the single subproblem D or E

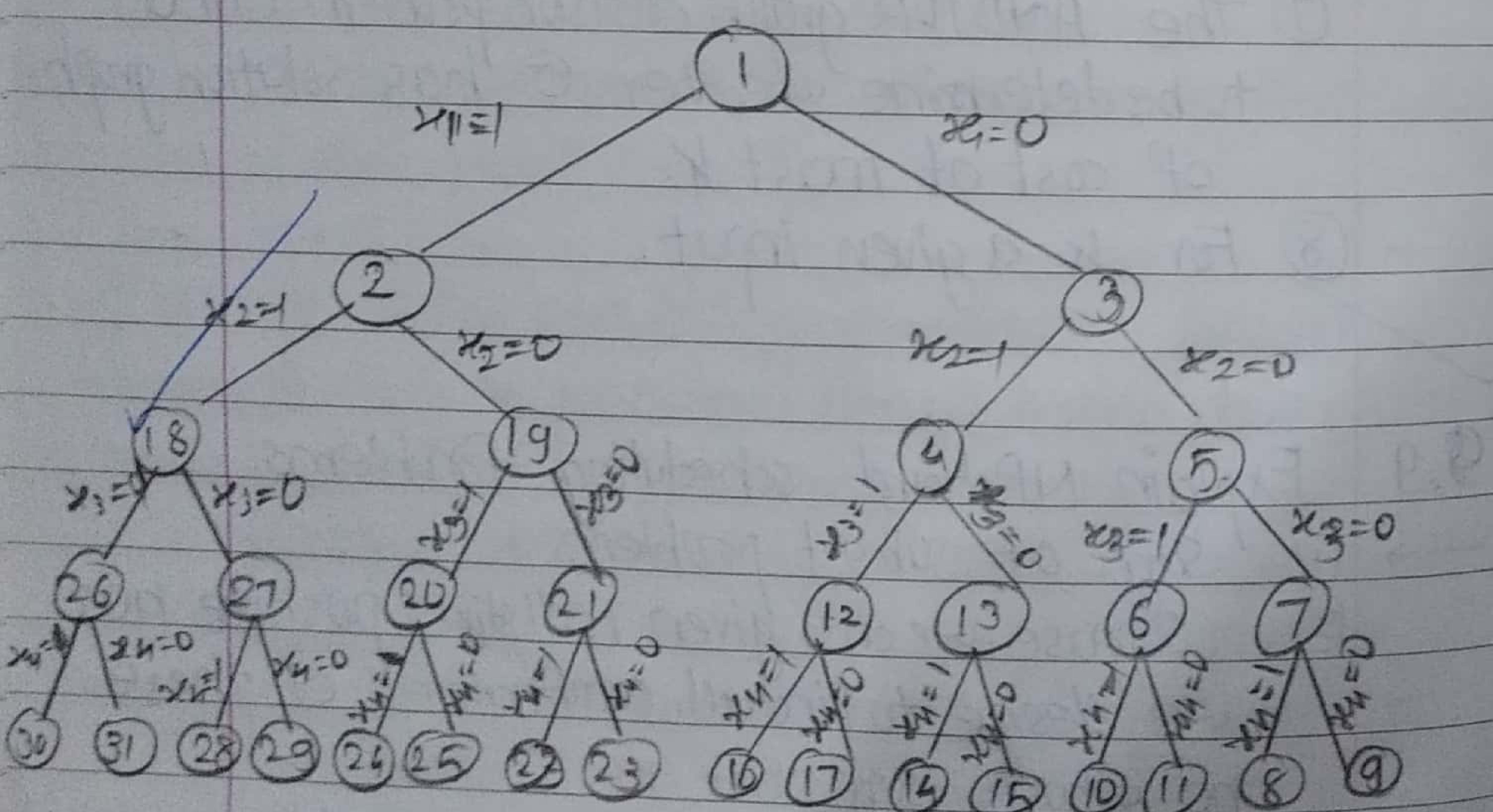


7. The AND/OR graph decision problem (AOG) to be determine whether G has solution graph of cost at most k.
8. For k a given input.

Q.9 Explain NP-Hard scheduling Problems.

→ 1. Sum of subset problem
2. Suppose we are given n distinct positive nos we desire to find all combinations of these no. whose sums are m
3. This is called the sum of subset problem NP Hard Problem
4. eg. fig shows the portion of the stack space

are generate a by function sum of subset while working on instance $n=6$, $m=30$ & $w[1:6] = \{1,5,10,12,13,15,18\}$. The rectangular nodes list the values of s, k and r on each of the calls to sum of sub. Circular nodes represent points at which ~~sb~~ subsets with sums m are printed out. At nodes A, B and C the output is note that ~~th~~ resp (1,1,0,0,1), (1,0,1,1) and (0,0,1,0,0). Note that tree of figure contains only 23 rectangular nodes. The full state space tree for $n=6$ contain $2^6-1=63$ nodes from which calls could be made (this count excludes the 64 leaf nodes no call need to be match from a leaf)



Q.10 Explain NP Hard Code Optimization problem

→ ① The function of a compiler is to translate programs written in some source language, into an equivalent assembly language or machine language problems.

② Thus, the C++ Compiler on the spare 10 translates C++ programs into the machine language of this machine

③ We took at the problem of translating arithmatic expression in a language such as C++ into assembly language code.

④ The translation clearly depend 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 arithmatic has to be performed in this register.

⑨ & represents a binary operator such that +, −, *, & /

⑩ Then the left operand ⊗ must be in the accumulation.

C
Bhagate