NP Completeness

Date:
Page No.:

→ Polynomial-time Algorithms: It is an algorithm in which on inputs of size n, worst-case running time is $O(n^k)$ for a constant k.

→ Not all the problems can be solved in polynomial time.

→ Example: Turing's Halting problem.

→ Some problems can't be solved by any computer no matter how much time is provided - such problems are called undecidable.

→ Some problems can be solved but not in $O(n^k)$ time.

→ A decision problem is a problem in which the solutions are either yes or no / 0 or 1 / True or false.

→ A problem is polynomial-time solvable if there is an algorithm to solves it in time $O(n^k)$, for some constant k.

→ The problems are categorized into
1. P class problem.
2. NP class problem.
3. NP-complete class problem.

P class: The set of decision problems that are solvable in polynomial time.

→ problems in P are also called tractable.

→ problems are not in P are called intractable. It can be solved in reasonable time only for small inputs.

NP (Nondeterministic polynomial) class:

→ The problems which are verifiable in polynomial time is called NP problem.

→ If we have a "certificate" of solution, we could verify that the certificate is correct in time polynomial to the size of the input.

Verification Algorithm

A verification algorithm is an algorithm A that takes two inputs:
1. An ordinary input x (coded in binary)
2. A certificate y
and outputs 1 on certain combinations of x and y.

→ verification algorithm A verifies an input string x if there exists a certificate y such that -

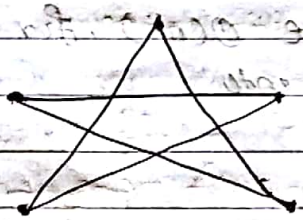
$$A(x, y) = 1$$

⇒ Example: Hamiltonian Cycle

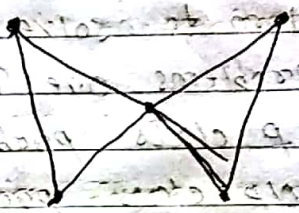
* In a directed graph $G = (V, E)$, determine a simple cycle that contains each vertex in V. i.e. each vertex can only be visited only once except
* certificate: sequence: $\langle v_1, v_2, v_3, \ldots, v_{|V|} \rangle$
* verification:

(1) $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, |V|$

(2) $(v_{|V|}, v_1) \in E$



(Hamiltonian)



(Not Hamiltonian)

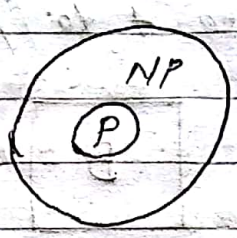
→ Examples of NP problems:

1. HAM-CYCLE = {⟨G⟩ | G is a Hamiltonian graph}
2. CIRCUIT-SAT = {⟨C⟩ | C is a satisfiable boolean circuit}
3. SAT = {⟨φ⟩ | φ is satisfiable boolean formula}
4. CNF-SAT = {⟨φ⟩ | φ is a satisfiable boolean formula in CNF}
5. 3-CNF-SAT = {⟨φ⟩ | φ is a satisfiable boolean formula in CNF}
6. CLIQUE = {⟨G, K⟩ | G is an undirected graph with a clique of size K}
7. VERTEX-COVER = {⟨G, K⟩ | undirected graph G has a vertex cover of size K}
8. TSP = {⟨G, C, K⟩ | $G = \langle V, E \rangle$ is a complete graph, $C: V \times V \to Z$ is a cost function, $K \in Z$, and G has a travelling salesman tour with cost at most K}
9. SUBSET-SUM = {⟨S, t⟩ | there is a subset $S' \subseteq S$ such that $t = \sum S'$}

Page No.:

Note: A p class problem belongs to NP

$P \subseteq NP$



Deterministic Algorithm:

An algorithm is said to be deterministic if the result of every operation of an algorithm is unique.

```
if (a > b)
    printf("a is greater");
else
    printf("b is greater");
```

Non-Deterministic Algorithm:

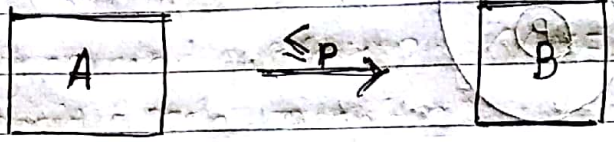
An algorithm is said to be non-deterministic if the result of every operation of an algorithm is not unique.

```
for(i=0; i<n; i++)
    if(a[i] == k)
    {   printf("Successful search");
        printf("Key element is found at position", i);
    }
    else
        printf("Unsuccessful search");
```

Reduction:

→ Reduction is a tool for comparison between problems.

→ If we can translate problem A to problem B

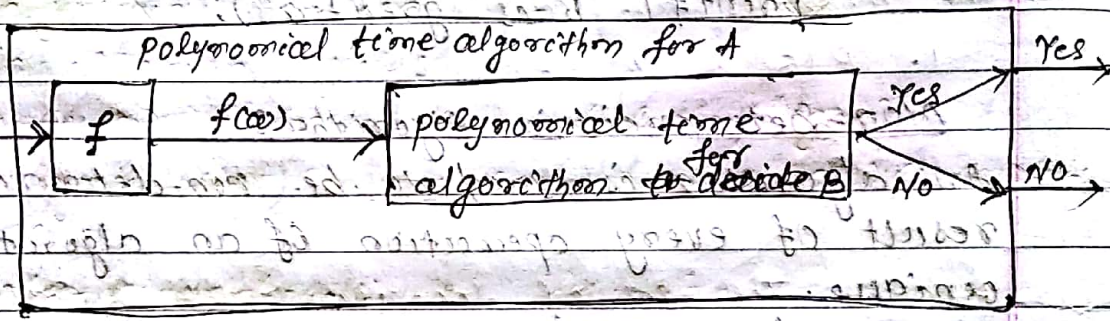


Then B is at least as hard as A.

→ Definition: Language A is polynomial time reducible (i.e. $A \leq_P B$) to language B if a polynomial time computable function $f: \Sigma^* \to \Sigma^*$ exists where for every w,

$w \in A \Leftrightarrow f(w) \in B$ (∵ w is input to an Algo A)

Here f is called the polynomial time reduction of A to B



→ To solve a decision problem A in polynomial time

1. Use a polynomial time reduction algorithm to transform A into B.
2. Run a known polynomial time algorithm for B.
3. Use the answer for B as the answer for A.

→ Given two problems A, B, we say that A is reducible to B ($A \leq_p B$) if:

1. There exist a function f that converts the input of A to inputs of B in polynomial time.
2. $A(i) = YES \Leftrightarrow B(f(i)) = YES$

How to Reduce

1. construct f
2. show f is polynomial time computable.
3. prove f is a reduction i.e. show
   (i) If $w \in A$ then $f(w) \in B$
   (ii) If $f(w) \in B$ then $w \in A$

NP-Completeness

→ A problem B is NP complete if:

1. $B \in NP$
2. $A \leq_p B$ for all $A \in NP$

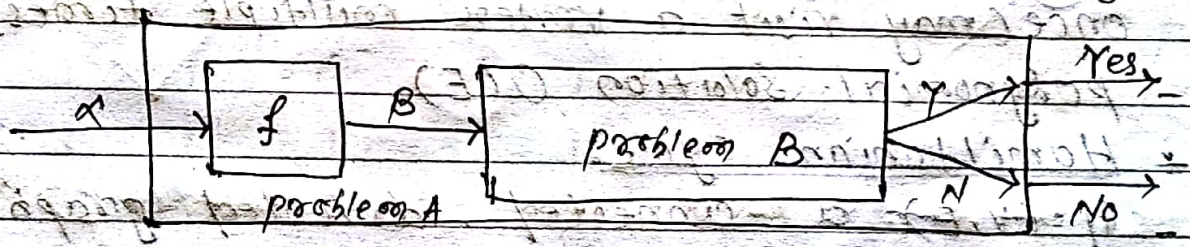
→ If B satisfies only property (2) we say that B is NP-hard.

→ problems which can't be solved in a polynomial-time are called NP-hard problem.

→ No polynomial algorithm has been discovered for an NP-complete problem.

→ There is no polynomial time algorithm exist for any NP complete problem.

Reduction and NP-completeness



→ Suppose we know:

* No polynomial time algorithm exist for problem A.
* we have a polynomial reduction f from A to B

→ No polynomial-time algorithm exists for B.

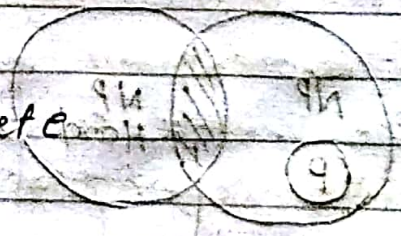
proof of NP-completeness:

Theorem: If A is NP-complete and $A \leq_p B$

$\Rightarrow$ B is NP-hard

In addition, if $B \in NP$

$\Rightarrow$ B is NP-complete



proof:

Assume that $B \in P$

Since $A \leq_p B \Rightarrow A \in P$ (contradiction!)

$\Rightarrow$ B is NP-hard

P and NP-complete Problems

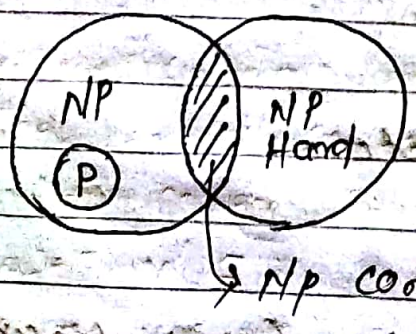
① * Shortest-path problem:
- Given a graph $G=(V,E)$ find the shortest path from a source to all other vertices
- polynomial solution: $O(VE)$

* Longest path problem:
- Given a graph $G=(V,E)$ find a longest path from a source to all other vertices
- NP-complete

② * Euler Tour:
- $G=(V,E)$ a connected, directed graph find a cycle that traverses each edge of G exactly once (may visit a vertex multiple times)
- polynomial solution $O(E)$

* Hamiltonian cycle:
- $G=(V,E)$ a connected, directed graph find a cycle that visits each vertex of G exactly once.
- NP-complete

Relationship Between P, NP and NP-complete



Page No.:

Satisfiability (SAT)

A boolean formula is satisfiable if there is a way to assign truth values (0 or 1) to the variables such that the final result is 1.

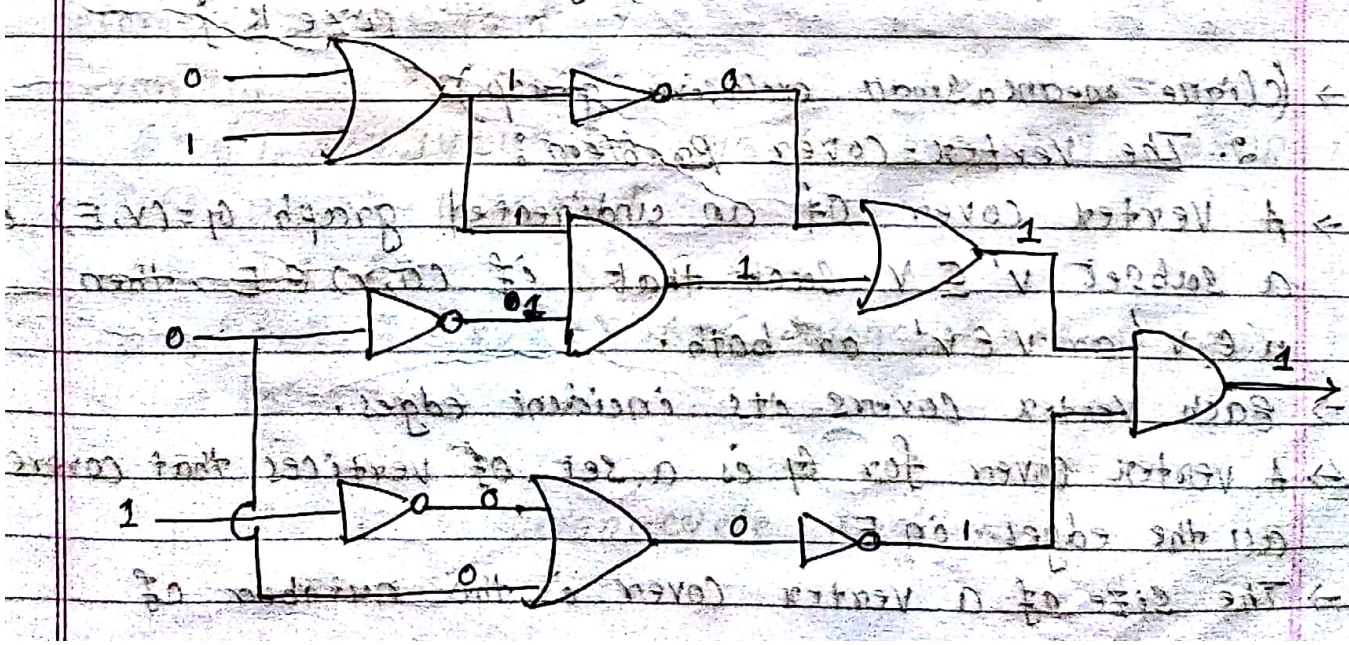
Example: $f(x,y,z) = (x \wedge (y \vee \bar{z})) \vee (\bar{y} \wedge z \wedge \bar{x})$

x	y	z	$x \wedge (y \vee \bar{z})$	$(\bar{y} \wedge z \wedge \bar{x})$	$f(x,y,z)$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1

The assignment $x=1, y=1, z=0$ makes $f(x,y,z)$ true. Hence it is satisfiable.

CIRCUIT SATISFIABILITY

A combinational circuit is called circuit satisfiable if for set of inputs, output of this circuit should always be one.



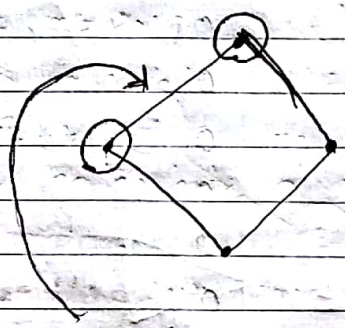
NP-Complete Problems

1. The clique problem: A clique is an undirected graph $G=(V,E)$ is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in E.

→ A clique is a complete subgraph of G.

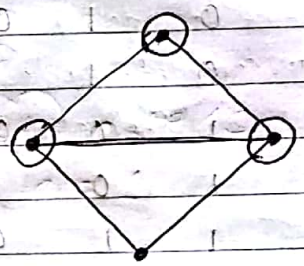
→ clique is a subset of vertices in V all connected to each other by edges in E (i.e. forming a complete graph)

→ The size of a clique is the number of vertices it contains.



clique(G,2) = Yes

clique(G,3) = No



clique(G,3) = Yes

clique(G,4) = No

→ The clique problem is the optimization problem of finding a clique of maximum size in a graph.

→ CLIQUE = {⟨G,K⟩ : G is a graph with a clique of size K}

→ (clique means a small exclusive group.)

2. The Vertex-Cover Problem:

→ A vertex cover of an undirected graph $G=(V,E)$ is a subset $V' \subseteq V$ such that if $(u,v) \in E$, then $u \in V'$ or $v \in V'$ or both.

→ Each vertex covers its incident edges.

→ A vertex cover for G is a set of vertices that covers all the edges in E.

→ The size of a vertex cover is the number of vertices in it.

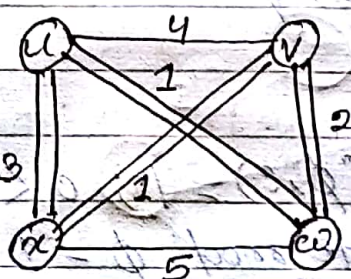
→ The vertex-cover problem is to find a vertex cover of minimum size in a given graph.

→ VERTEX-COVER = {⟨G,K⟩ : graph G has a vertex cover of size K}

3. The Traveling-Salesman Problem:

→ In traveling salesman problem, a salesman wish to make a tour n cities or hamiltonian cycle means visiting each city exactly once and finishing at the city he starts from.

→ There is an integer cost c(i,j) to travel from city i to city j and the salesman wishes to make the tour whose total cost is minimum, where the total cost is the sum of the individual costs along the edges of the tour.



→ minimum cost of tour is 7.

TSP = {⟨G, c, k⟩ : G = (V, E) is a complete graph, c is a function from V × V → Z, k ∈ Z and G has a travelling-salesman tour with cost at most k}

4. The Subset-Sum Problem:

→ In the subset-sum problem, a finite set S ⊂ N and a target t ∈ N.

→ Here we find whether there is a subset S' ⊆ S whose elements sum to t

→ Example:

S = {1, 2, 7, 14, 49, 98, 343, 686, 2409, 2793, 16808, 17206, 117705, 117993}

t = 138457

S' = {1, 2, 7, 98, 343, 686, 2409, 17206, 117705} is a solution.

5. The Hamiltonian-cycle problem.