

**UNIT V NP COMPLETE AND NP HARD**

NP-Completeness: Polynomial Time – Polynomial-Time Verification – NP- Completeness and Reducibility – NP-Completeness Proofs – NP-Complete Problems.

---

**What was the first problem proved as NP-Complete?**

There must be some first NP-Complete problem proved by the definition of NP-Complete problems. SAT (Boolean satisfiability problem) is the first NP-Complete problem proved by Cook (See CLRS book for proof).

It is always useful to know about NP-Completeness even for engineers. Suppose you are asked to write an efficient algorithm to solve an extremely important problem for your company. After a lot of thinking, you can only come up with an exponential time approach which is impractical. If you don't know about NP-Completeness, you can only say that I could not come up with an efficient algorithm. If you know about NP-Completeness and prove that the problem is NP-complete, you can proudly say that the polynomial-time solution is unlikely to exist. If there is a polynomial-time solution possible, then that solution solves a big problem of computer science many scientists have been trying for years.

**NP COMPLETENESS PROBLEM AND PROOFS****Prove that the circuit-satisfiability problem belongs to the class NP and also NP-Hard.**

**Satisfiability Problem:** SAT(Boolean Satisfiability Problem) is the problem of determining if there exists an interpretation that satisfies a given boolean formula. It asks whether the variables of a given boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE. If this is the case, the formula is called satisfiable. On the other hand, if no such assignment exists, the function expressed by the formula is FALSE for all possible variable assignments and the formula is unsatisfiable.

**Problem:** Given a boolean formula  $f$ , the problem is to identify if the formula  $f$  has a satisfying assignment or not.

**Explanation:** An instance of the problem is an input specified to the problem. An instance of the problem is a boolean formula  $f$ . Since an NP-complete problem is a problem which is both NP and NP-Hard, the proof or statement that a problem is NP-Complete consists of two parts:

The problem itself is in NP class.

All other problems in NP class can be polynomial-time reducible to that. (B is polynomial-time reducible to C is denoted as  $B \leq PC$ )

If the 2nd condition is only satisfied then the problem is called NP-Hard.

But it is not possible to reduce every NP problem into another NP problem to show its NP-Completeness all the time i.e., to show a problem is NP-complete then prove that the problem is in NP and any NP-Complete problem is reducible to that i.e. if B is NP-Complete and  $B \leq PC$  For C in NP, then C is NP-Complete. Thus, it can be verified that the SAT Problem is NP-Complete using the following propositions:

**SAT is in NP:**

If any problem is in NP, then given a ‘certificate’, which is a solution to the problem and an instance of the problem(a boolean formula f) we will be able to check (identify if the solution is correct or not) certificate in polynomial time. This can be done by checking if the given assignment of variables satisfies the boolean formula.

**SAT is NP-Hard:**

In order to prove that this problem is NP-Hard then reduce a known problem, Circuit-SAT in this case to our problem. The boolean circuit C can be corrected into a boolean formula as:

For every input wire, add a new variable  $y_i$ .

For every output wire, add a new variable Z.

An equation is prepared for each gate.

These sets of equations are separated by  $\cap$  values and adding  $\cap Z$  at the end.

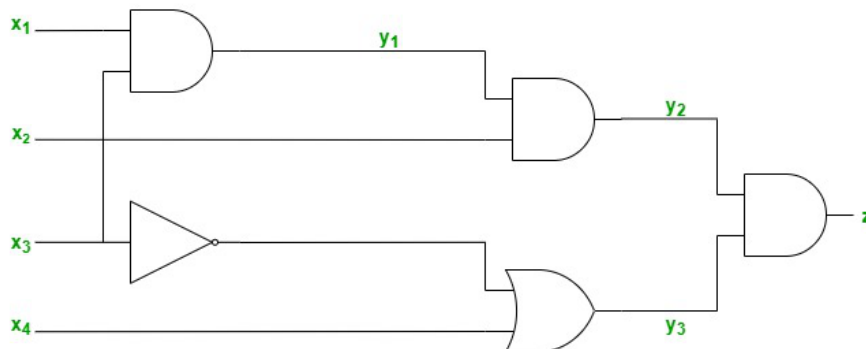
This transformation can be done in linear time. The following propositions now hold:

If there is a set of input, variable values satisfying the circuit then it can derive an assignment for the formula f that satisfies the formula. This can be simulated by computing the output of every gate in the circuit.

If there is a satisfying assignment for the formula f, this can satisfy the boolean circuit after the removal of the newly added variables.

**For Example:**

If below is the circuit then:



Therefore, the SAT Problem is NP-Complete.

## Prove that Sparse Graph is NP-Complete

**Problem:** Given graph  $G = (V, E)$  and two integers  $a$  and  $b$ . A set of a number of vertices of  $G$  such that there are at most  $b$  edges between them is known as the Sparse Subgraph of graph  $G$ .

**Explanation:** Sparse Subgraph problem is defined as follows:

- Input: An undirected graph  $G = (V, E)$  and variables  $a, b$ .
- Output: A set  $S$  which is a subset of  $V$  where  $|S| = a$  and there are at most  $b$  edges between pairs of vertices in  $S$ . Report “NO”, if no such set exists.

To prove a problem NP Complete, there are two steps involved:

Prove given problem belong to NP Class

- All other problems in the NP class can be polynomial time reducible to that problem. (This is the prove of being NP-Hard)

Now it is not possible to reduce every NP problem to another NP problem to prove it's NP completeness all the time. That's why we show that any known NP complete problem is reducible to that problem in polynomial time.

### **Proof:**

#### **1. Sparse Graph belongs to NP Class:**

A problem is said to be in NP Class if the solution for the problem can be verified in polynomial time.

Given an input  $G = (V, E)$  and two integer variables  $a$  and  $b$ .

- For a given solution  $S$ , it takes  $O(|V|)$  time to verify that  $|S| = a$ .
- To check that the number of edges between any pair of vertices in  $S$  is at most  $b$ , we need to run  $O(|V|^2)$  algorithm.

So, verification of a solution for Sparse Graph takes at most  $O(|V|^2)$  which is polynomial in nature so Sparse Graph belongs to NP Class.

#### **2. Sparse Graph is an NP-Hard problem:**

Now we need to show Sparse Graph is at least as hard as a known NP-Complete Problem by reduction technique.

Here the known problem is going to be the Independent Set problem which is already known to be NP-complete

We are going to show the reduction from **Independent Set -> Sparse Graph**.

**Input Conversion:** We need to convert the input from Independent Set to the input of the Sparse Graph.

**Independent Set Input:** An undirected graph  $G(V, E)$  and integer  $k$ .

**Sparse Graph Input:** An undirected graph  $G'(V, E)$  and two integers  $a$  and  $b$ .

We are going to transform the input from Independent Set to Sparse Graph such that

- $G' = G(V, E)$
- $a = k$
- $b = 0$  since we need to have the maximum Independent Set

This conversion is going to take  $O(1)$  time. So it's polynomial in nature.

**Output Conversion:** We need to convert the solution from Sparse Graph to the solution for the Independent Set problem.

Solution of Sparse Graph will result in a set  $a$  which would be an Independent Set of size  $k$  as  $k = a$ . So direct output from Sparse Graph can be used by Independent Set. Since no conversion is required, it's again polynomial in nature.

**Correctness:**

Forward implication: Consider any Independent Set  $S$ . This is a Sparse graph as well, since there is no edges between vertices of  $S$  ( $b \leq 0$ ) and  $|S| = k = a$

Reverse implication: A given Sparse Graph solution  $S$ , it is an Independent Set as well (number of edges between vertices is zero, and  $|S| = k = a$ ).

So, this means **Sparse Graph has a solution  $\leftrightarrow$  Independent Set has a solution.**

The complete **reduction takes polynomial time** and Independent Set is an NP complete problem. So Sparse Graph is also an NP complete problem.

**Conclusion:**

Hence we can conclude that Sparse Graph is an NP Complete problem.

**Hamiltonian Cycle:**

A cycle in an undirected graph  $G=(V, E)$  traverses every vertex exactly once.

**Problem Statement:** Given a graph  $G=(V, E)$ , the problem is to determine if graph  $G$  contains a Hamiltonian cycle consisting of all the vertices belonging to  $V$ .

**Explanation:** An instance of the problem is an input specified to the problem. An instance of the Independent Set problem is a graph  $G=(V, E)$ , and the problem is to check whether the graph can have a Hamiltonian Cycle in  $G$ . Since an NP-Complete problem, by definition, is a problem which is both in NP and NP-hard, the proof for the statement that a problem is NP-Complete consists of two parts:

1. The problem itself is in NP class.
2. All other problems in NP class can be polynomial-time reducible to that.

If the 2nd condition is only satisfied then the problem is called NP-Hard. But it is not possible to reduce every NP problem into another NP problem to show its NP-Completeness all the time. That is why if we want to show a problem is NP-Complete, we just show that the problem is in NP and if any NP-Complete problem is reducible to that, then we are done

Hamiltonian Cycle is in NP If any problem is in NP, then, given a ‘certificate’, which is a solution to the problem and an instance of the problem (a graph  $G$  and a positive integer  $k$ , in this case), we will be able to verify (check whether the solution given is correct or not) the certificate in polynomial time. The certificate is a sequence of vertices forming the Hamiltonian Cycle in the graph. We can validate this solution by verifying that all the vertices belong to the graph and each pair of vertices belonging to the solution are adjacent. This can be done in polynomial time, that is  $O(V + E)$  using the following strategy for graph  $G(V, E)$ :

flag=true

For every pair  $\{u, v\}$  in the subset  $V'$ :

    Check that these two have an edge between them

    If there is no edge, set flag to false and break

If flag is true:

    Solution is correct

Else:

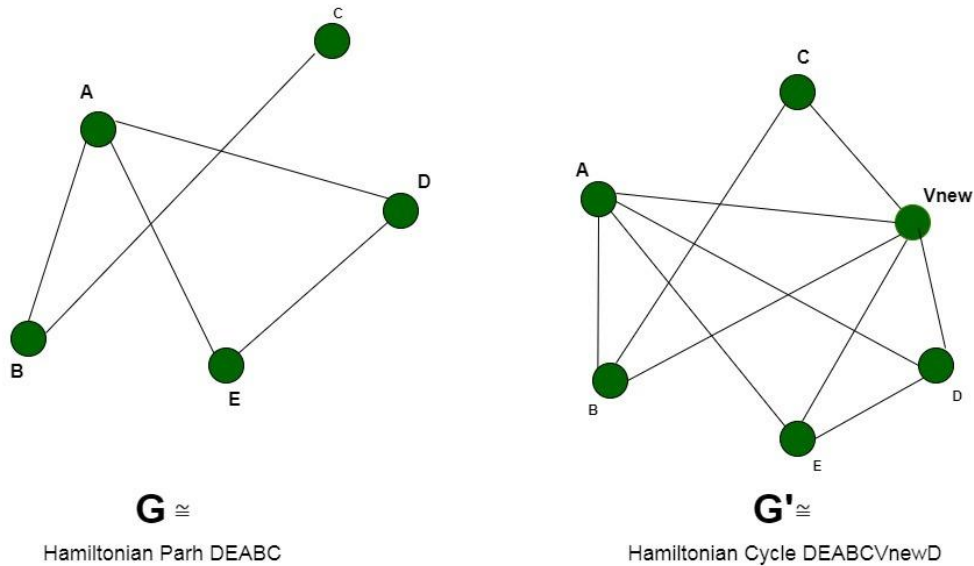
    Solution is incorrect

Hamiltonian Cycle is NP Hard In order to prove the Hamiltonian Cycle is NP-Hard, we will have to reduce a known NP-Hard problem to this problem. We will carry out a reduction from the Hamiltonian Path problem to the Hamiltonian Cycle problem. Every instance of the Hamiltonian Path problem consisting of a graph  $G = (V, E)$  as the input can be converted to a Hamiltonian Cycle problem consisting of graph  $G' = (V', E')$ . We will construct the graph  $G'$  in the following way:

- $V' =$  Add vertices  $V$  of the original graph  $G$  and add an additional vertex  $V_{new}$  such that all the vertices connected of the graph are connected to this vertex. The number of vertices increases by 1,  $V' = V + 1$ .
- $E' =$  Add edges  $E$  of the original graph  $G$  and add new edges between the newly added vertex and the original vertices of the graph. The number of edges increases by the number of vertices  $V$ , that is,  $E' = E + V$ .
- Let us assume that the graph  $G$  contains a hamiltonian path covering the  $V$  vertices of the graph starting at a random vertex say  $V_{start}$  and ending at  $V_{end}$ , now since we connected all the vertices to an arbitrary new vertex  $V_{new}$  in  $G'$ . We extend the

original Hamiltonian Path to a Hamiltonian Cycle by using the edges  $V_{end}$  to  $V_{new}$  and  $V_{new}$  to  $V_{start}$  respectively. The graph  $G'$  now contains the closed cycle traversing all vertices once.

- We assume that the graph  $G'$  has a Hamiltonian Cycle passing through all the vertices, inclusive of  $V_{new}$ . Now to convert it to a Hamiltonian Path, we remove the edges corresponding to the vertex  $V_{new}$  in the cycle. The resultant path will cover the vertices  $V$  of the graph and will cover them exactly once.



Thus we can say that the graph  $G'$  contains a Hamiltonian Cycle if graph  $G$  contains a Hamiltonian Path. Therefore, any instance of the Hamiltonian Cycle problem can be reduced to an instance of the Hamiltonian Path problem. Thus, the Hamiltonian Cycle is NP-Hard. Conclusion: Since, the Hamiltonian Cycle is both, a NP-Problem and NP-Hard. Therefore, it is a NP-Complete problem.