MTech CSE – 1st Semester
Student ID: **A125023**
Student Name: **SRIJITA VERMA**

Question

Is the 3-SAT (3-CNF-SAT) problem NP-hard? Justify your answer.

Answer

Short Answer

Yes, the 3-SAT (3-CNF-SAT) problem is NP-hard. In fact, 3-SAT is NP-complete.

Background and Definitions

Boolean Satisfiability (SAT)

The SAT problem asks whether a given Boolean formula has a truth assignment that makes the formula evaluate to TRUE. SAT was the first problem proven to be NP-complete, as shown by the Cook–Levin Theorem.

3-SAT (3-CNF-SAT)

In 3-SAT, the Boolean formula is restricted to:

- Conjunctive Normal Form (CNF), and
- each clause contains exactly three literals.

Formally, a 3-SAT formula has the form:

$$\varphi = \bigwedge_{i=1}^{m} (\ell_{i1} \lor \ell_{i2} \lor \ell_{i3}),$$

where each literal ℓ_{ij} is either a variable or its negation.

Meaning of NP-Hardness

A problem P is *NP-hard* if every problem in **NP** can be reduced to P in polynomial time. If a problem is both NP-hard and belongs to **NP**, then it is *NP-complete*.

Membership of 3-SAT in NP

Given a truth assignment:

- each clause can be checked in constant time,
- all clauses can be checked in linear time.

Thus, a satisfying assignment is a polynomial-time verifiable certificate. Therefore,

$$\text{3-SAT} \in \mathbf{NP}.$$

Polynomial Reduction from SAT to 3-SAT

To establish NP-hardness, we show that:

$$\text{SAT} \leq_p \text{3-SAT}.$$

That is, any CNF formula can be transformed into an equivalent 3-CNF formula in polynomial time.

Clause Transformation

Consider a CNF clause of arbitrary length:

$$(x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_k).$$

This clause can be replaced by a conjunction of 3-literal clauses by introducing new variables:

$$(x_1 \vee x_2 \vee y_1) \ \wedge \ (\neg y_1 \vee x_3 \vee y_2) \ \wedge \ \cdots \ \wedge \ (\neg y_{k-3} \vee x_{k-1} \vee x_k).$$

This transformation satisfies the following properties:

- satisfiability is preserved,
- the number of new variables and clauses grows linearly,
- the transformation runs in polynomial time.

Worked Reduction Example: SAT to 3-SAT

Consider the CNF formula:

$$\varphi = (x_1 \vee x_2 \vee x_3 \vee x_4) \ \wedge \ (\neg x_1 \vee x_2) \ \wedge \ (x_3).$$

This formula is in CNF, but not all clauses contain exactly three literals. We now convert it into an equivalent 3-CNF formula.

Step 1: Handling Clauses with More Than Three Literals

The clause

$$(x_1 \vee x_2 \vee x_3 \vee x_4)$$

contains four literals. Introduce a new variable y_1 and rewrite it as:

$$(x_1 \vee x_2 \vee y_1) \ \wedge \ (\neg y_1 \vee x_3 \vee x_4).$$

This transformation preserves satisfiability:

- if the original clause is satisfiable, the new clauses are satisfiable;
- if the new clauses are satisfiable, at least one of the original literals must be true.

Step 2: Handling Clauses with Fewer Than Three Literals

The clause

$$(\neg x_1 \vee x_2)$$

contains two literals. We duplicate one literal to obtain:

$$(\neg x_1 \vee x_2 \vee x_2).$$

This duplication does not change the logical meaning of the clause.

Step 3: Handling Single-Literal Clauses

The clause

$$(x_3)$$

contains a single literal. We duplicate it to obtain:

$$(x_3 \vee x_3 \vee x_3).$$

Again, satisfiability is preserved.

Step 4: Final 3-CNF Formula

The resulting 3-CNF formula is:

$$\varphi' = (x_1 \vee x_2 \vee y_1) \ \wedge \ (\neg y_1 \vee x_3 \vee x_4) \ \wedge \ (\neg x_1 \vee x_2 \vee x_2) \ \wedge \ (x_3 \vee x_3 \vee x_3).$$

The formula φ' is satisfiable if and only if the original formula φ is satisfiable.

Step 5: Complexity of the Reduction

Each clause is transformed using a constant number of new variables and clauses. The total size of the formula increases linearly, and the transformation can be performed in polynomial time.

Therefore,

$$\text{SAT} \leq_p \text{3-SAT}.$$

Consequence

Since:

- SAT is NP-complete, and
- SAT reduces to 3-SAT in polynomial time,

we conclude that:

$$\text{3-SAT is NP-hard.}$$

Final Classification

We have shown that:

$$\text{3-SAT} \in \mathbf{NP} \quad \text{and} \quad \text{SAT} \leq_p \text{3-SAT}.$$

Therefore,

$$\text{3-SAT is NP-complete,}$$

and hence NP-hard.

Why This Result Is Important

The NP-hardness of 3-SAT is fundamental because:

- many NP-hardness proofs reduce from 3-SAT,
- it serves as a standard starting point for reductions,
- it shows that even very restricted Boolean formulas remain computationally hard.

Despite each clause having only three literals, the problem retains the full difficulty of SAT.

Intuition

SAT is hard because it encodes arbitrary logical constraints. Restricting clauses to length three does not reduce expressive power; it only standardizes the structure. Thus, 3-SAT captures the essence of NP-hardness in a highly controlled form.

Final Remark

Although 3-SAT is NP-hard, special cases such as 2-SAT are solvable in polynomial time. This highlights how small syntactic restrictions can dramatically change computational complexity.