Try doing one of the other polynomial-time reductions we've mentioned, e.g.,

$$\text{CUBIC SUBGRAPH} \leq_P \text{SATISFIABILITY}$$

Perform a polynomial-time reduction from the **Cubic Subgraph** problem to the **Satisfiability (SAT)** problem.

The **Cubic Subgraph** problem is about finding a subgraph of a graph G that is a cubic graph (every vertex has a degree of 3).

To do this reduction, we'll construct a SAT instance such that it is satisfiable if and only if the original graph G contains a cubic subgraph.

1. **Input**: We start with a graph G as the input for the **Cubic Subgraph** problem.

2. **Construction**: We'll construct a Boolean formula in SAT that models the condition of finding a cubic subgraph within G.

3. **Boolean Variables**: For each vertex in G, we create a set of Boolean variables to represent the presence of edges to its neighbors. For each vertex V in G and each pair of its neighbors U and W, we introduce a variable X_VUW, which is true if there's an edge between V and both U and W (i.e., V is connected to a cubic subgraph through U and W).

4. **Constraints**:

- For each vertex V in G, we add constraints to ensure that it is connected to exactly three other vertices in the subgraph. This is expressed as a SAT clause using the variables X_VUW for all possible pairs of neighbors U and W of V.

- We also include clauses to ensure that if a vertex V is in the cubic subgraph, it must have exactly three neighbors (each pair of neighbors implies the presence of an edge).

5. **Satisfiability Check**: We convert the logical formula into CNF format and check if the formula is satisfiable.

If the SAT instance is satisfiable, it means that there exists a cubic subgraph within the original graph G. The assignment of Boolean values to the variables will indicate which vertices and edges belong to this cubic subgraph.

This reduction is polynomial-time because the number of Boolean variables and constraints is linear in the size of the input graph G, making it a polynomial-time reduction from **Cubic Subgraph** to **SAT**.

Therefore, if you can efficiently solve the **SAT** problem, you can also solve the **Cubic Subgraph** problem, and this demonstrates that the **Cubic Subgraph** problem is NP-hard.