Example 12.15 **Vertex Cover Problem:** It is one type of optimization problem in computer science. Given a graph G, the aim is to find the minimum vertex cover $\mathrm{C} \subseteq \mathrm{G}$. Given a graph $G = \{V, E\}$ and an integer $k \leq V$, the problem is to find V', subset of at most k vertices, such that each edge of G is incident to at least one vertex in V'.
The algorithm can be constructed by assuming a set of k vertices C for a given graph G. We have to check whether there exists a vertex cover of $\leq \mathrm{k}$ vertices. The checking can be done by the following steps.
i) Check whether C contains at most k number of vertices.
ii) Check whether C is a vertex cover of G.

Input $G = \{V, E\}, C = \{V'\}$ where $|V'| \leq k$
for $i = 1$ to $|E|$

check the edges ends on V' i
remove them from $\{\mathrm{E}\}$.
End

Thus, this checking operation is performed in $\mathrm{O}\left(|\mathrm{V}'| + |\mathrm{E}|\ \right)$ i.e. in polynomial time. A graph with n number of vertices can have 2^n number of such vertex combination. For each of the combinations, the same checking algorithm is performed. Among these, the set of vertices with minimum number is the minimum vertex cover of G. So, a minimum vertex cover algorithm is carried out by a non-deterministic Turing machine in polynomial time.

Example 12.16 Prove that 3 SAT is NP complete.

Solution:

i) 3 SAT is clearly in NP, given an instance of a circuit C and a 'satisfying' setting of Boolean variables. A simulation of the circuit can easily check in polynomial time whether the output is 1 or not.

ii) A SAT problem φ is called 3 SAT if the number of literals in each clause is exactly 3. It is already shown how a SAT formula with clauses less than or more than 3 literals can be converted to a formula with clauses each having exactly 3 literals. This conversion can be made in polynomial time. Hence, SAT can be reduced to 3 SAT in polynomial time. So 3 SAT is NP complete.