ADVANCED ALGORITHMS

ASSIGNMENT – 1

NAME : SANMAT SANJAYAKUMAR PAYAGOUDAR

SRN : PES1UG20CS385

1. **TURING MACHINE :**

   A simple mathematical model of computation, having a set of rules, which manipulates symbols on a strip of tape using those rule, and capable of implementing any computer algorithm.

   COMPLEXITY CLASS :

   P : Set of all decision problems which can be solved using deterministic Turing machine in a polynomial amount of computation time or polynomial time.

   NP : Set of all the decision problems which can be solved using non-deterministic Turing machine in a polynomial amount of computation time or polynomial time.

   NP Completeness : A problem is said to be NP complete if a brute-force algorithm can find a solution by trying all possible solutions and verifying the correctness of each solution.

2. **SAT :**
   SAT is the problem of determining if a Boolean formula is satisfiable or unsatisfiable.
   - Satisfiable : If the Boolean variables can be assigned values such that the formula turns out to be TRUE, then we say that the formula is satisfiable.
   - Unsatisfiable : If it is not possible to assign such values, then we say that the formula is unsatisfiable.

   Proof to show that SAT is NP :

   We are given a circuit with satisfying set of inputs. Output of each gate in circuit can be computed in a definite and constant time. Hence, the output of the circuit is verifiable in polynomial time.

   - So, given an input to SAT and a solution S, we can check whether each literal is evaluated to TRUE/FALSE.
   - And there are n literals in a clause, so time complexity is O(n) per clause.
   - And there are m clauses, so total running time is O(nm), which is polynomial in nature.

   So SAT can be verified in polytime.

3. **<u>Chromatic numbering</u> :**

Here we check whether there a possible way of assigning colors to each vertices with each neighbor colored differently.

In 3-coloring problem we check whether there a possible way of assigning colors to each vertices using only 3 different colors with each neighbor colored differently.

<u>3-coloring problem is NP-Complete.</u>

An NP-Complete problem is a problem which is both NP and NP-Hard. So to prove that a problem is NP-Complete it should satisfy these 2 statements :

1. The problem is in NP class.
2. The problem is in NP-Hard.

<u>3-coloring problem is in NP :</u> For the solution of any problem belonging to NP it can be verified that it is in polynomial time. This can be done in the following way :

For a edge between A and B in graph G verify that color of A is not equal to color of B

Hench, the solution can be checked for correctness in the polynomial-time of the graph with respect to its edge.

<u>3-coloring problem is NP-Hard :</u> In order to prove that 3-coloring problem is NP-Hard we are reducing a known NP-Hard problem, that is 3-SAT problem to 3-coloring problem.

Let's assume that the 3-SAT problem has a 3-SAT formula of 'm' clauses on 'n' variables denoted by $x_1, x_2, \ldots, x_n$. The graph can then be constructed from the formula in the following way :

1. For every variable x_i Construct a vertex v_i In the graph and a vertex $\mathbf{v_{i'}}$ denoting the negation of the variable x_i.
2. For each clause c in m, add 5 vertices corresponding to values c1, c1,….., c5.
3. Three vertices of different colors are additionally added to denote the values True, False, and Base (T, F, B) respectively.
4. Edges are added among these three additional vertices T, F, B to form a triangle.
5. Edges are added among the vertices v_i and $v_{i'}$ and Base (B) to form a triangle.

The following constraints are true for graph G:

1. For each of the pairs of vertices v_i and $v_{i'}$, either one is assigned a TRUE value and the other, FALSE.
2. For each clause c in m clauses, at least one of the literal has to hold TRUE value for the value to be true.

Now the reduction can be proved by the following two propositions **:**

Let us assume that the 3-SAT formula has a satisfying assignment, then in every clause, at least one of the literals $\mathbf{x_i}$ has to be true, therefore, the corresponding $\mathbf{v_i}$ can be assigned to a TRUE color and $\mathbf{v_{i'}}$ to FALSE. Now, extending this, for each clause the corresponding OR-gadget graph can be 3-colored. Hence, the graph can be 3-colored.

Let us consider that the graph G is 3-colorable, so if the vertex vi is assigned to the true color, correspondingly the variable x_i is assigned to true. This will form a legal truth assignment. Also, for any clause C_j = (x V y V z), it cannot be that all the three literals x, y, z are False. Because in this case, the output of the OR-gadget graph for C_j has to be colored False. This is a contradiction because the output is connected to Base and False. Hence, there exists a satisfying assignment to the 3-SAT clause.

Conclusion: Therefore, 3-coloring is an NP-Complete problem.