AYÇA ELİF AKTAŞ - 27802

PROGRAM 1

1. Reading Input: The program starts by reading a .col file to extract vertices and edges.
2. Checking Colorability:
   - **k_colorable() function** is called with the number of vertices n, the list of edges, and the number of colors k.
   - Inside this function, **create_vertex_coloring_cnf()** constructs the CNF (Conjunctive Normal Form) formula required for the SAT solver:
   - The constructed CNF formula is passed to a SAT solver.
3. SAT Solver Execution:
   - The solver attempts to find a solution that satisfies the CNF constraints.
   - If a solution exists, it means the graph is k-colorable, and the solver outputs the model representing this solution.
4. Interpreting the Model:
   - If the graph is found to be k-colorable, **interpret_model()** translates the SAT solver's model into a vertex-color mapping, indicating which vertex is assigned which color.
5. Output:
   - The program returns whether the graph is k-colorable and, if so, provides the specific coloring of the vertices.

This workflow efficiently determines if a graph can be colored with k colors without any two adjacent vertices sharing the same color, and if possible, identifies the coloring

PROGRAM 2

1. Initialize: Set k to 1 to start testing colorability from the minimum possible number of colors.
2. Colorability Check Loop: Continuously check if the graph can be colored with k colors using the **k_colorable() function**, which:
   a. Calls **create_vertex_coloring_cnf()** to generate CNF constraints based on the current value of k.
   b. Uses a SAT solver to determine if the CNF constraints are satisfiable, indicating the graph is colorable with k colors.
   c. If the SAT solver finds a solution, the graph is k-colorable; otherwise, it's not.
3. Result Handling:
   a. If the graph is k-colorable (is_colorable is True):
      i. Use **interpret_model()** to convert the SAT solver's model into a human-readable vertex-color mapping.
      ii. Print and return the coloring and the value of k as the chromatic number of the graph.
   b. If the graph is not k-colorable: Print a message indicating this and increment k to test the next higher number of colors.
4. Repeat: The loop continues until a satisfiable coloring is found, thus determining the chromatic number as the smallest k for which the graph is k-colorable.

This workflow effectively finds the smallest number of colors needed to color a graph such that no two adjacent vertices share the same color, thus determining the graph's chromatic number.