CSCI 2300 — Algo
Homework 10
Hayden Fuller

Question: 3-COLOR. Given a graph, is there a way to color the vertices red, green, and blue so that no adjacent vertices have the same color?
Show that 3-satisfiability reduces to graph 3-colorability. Give a step by step construction [25 points] and proof [25 points]

Construction:
We can transform a 3-satisfiability problem into a graph 3-colorability problem. By employing 3-colorability as logic gates, we can reduce each clause $(x_a \vee x_b \vee x_c)$ in the CNF to a graph structure. To achieve this, we assign colors to represent boolean values: Red for True, Green for False, and Blue for Neutral (which holds no boolean significance but aids in mimicking logic gates). First, we construct a control gadget to ensure adjacent vertices have different colors, resulting in the Blue node being Neutral.
Following that, we create a control structure to assign either Red (True) or Green (False) to variables x_n and $\neg x_n$, respectively, thus enforcing a truth assignment. This step is interchangeable because we have associated each available literal's truth value with a specific color.
Taking advantage of the constraint that adjacent colors must be distinct, we construct a logic gadget that exhibits logic gate behavior.
$E = \{(X, a_1), (Y, a_2), (a_1, a_2), (a_1, C), (a_2, C)\}$
Suppose a clause consists of arbitrary values x, y, and z. In that case, we construct a 3-colorable graph where the clause vertex is set to Red (True). Since x, y, and z are boolean literals (inputs), they must be assigned either Red or Green, excluding Blue.
Setting the clause vertex C to Red results in the 3-colorability of the entire logic gadget behaving like a logic OR gate. This specific variant is referred to as the OR gadget. This substructure becomes non-3-colorable only when both x and y are Green (False).
Since a 3-satisfiability clause contains three literals $(x_1 \vee x_2 \vee x_3)$, it can be equivalently represented as $((x_1 \vee x_2) \vee x_3)$. Using this property, we construct a clause gadget that determines the truth value of a clause based on its 3-colorability.
To build the final graph G, we utilize the earlier constructed control structure to assign values to boolean literals. We then connect each clause in the original boolean formula to either X, Y, or Z in the clause gadget.

Proof:
Through exhaustive examination, we can demonstrate that the OR gadget is exclusively 3-colorable only when either X, Y, or both are assigned Red (True). This process can be easily verified by testing all possible combinations: (F, F), (T, F), (F, T), and (T, T) for (X, Y). Due to its simplicity, I will not include the detailed typing here.
To establish the correctness of the transformation from 3-satisfiability, we aim to prove that $(x_1 \vee x_2 \vee x_3)$ is equivalent to the 3-colorability of the graph G, denoted as colorable(G). It's important to note that the final structure consists of a logic gadget nested within the OR variant.
The original clause implies that colorable(G) is False if and only if x1, x2, and x3 are all False (F).
First, we can demonstrate that the general logic gadget is always 3-colorable regardless of the unspecified vertex C, considering any combination of X and Y.
Upon observation, the clause vertex C must be the same as X or Y when X = Y.

Based on Proof 1, we established that the OR gadget is consistently 3-colorable when at least one of the inputs is True.

By substituting the value of the sub-gadget's vertex C into the OR gadget, we deduce that for the entire clause gadget to be 3-colorable, Z must be True when X, Y = F. However, when either X or Y (or both) is True, the entire clause gadget is 3-colorable, irrespective of the value of Z, as the smaller logic gadget is 3-colorable when its vertex C = T.

At this stage, we have demonstrated that the graph G is 3-colorable if and only if the clause is true, which occurs when any of X, Y, or Z is True. This conclusion is derived from the construction process and a thorough explanation of why they would have the same truth table.