Part I: Graph Coloring

At some point in your childhood, chances are you were given a blank map—of the United States, of Africa, of the whole world—and you tried to color in each state or each country so that if two regions shared a border, you used different colors for them. Maybe you tried to figure out how many colors you needed. Could you color in the U.S. map using only three different colored crayons? What if you had four colors?

In computer science, we call this question—at minimum, how many colors are needed so that no two adjacent regions are the same color?—the *Graph Coloring* problem. The problem is formally defined as follows:

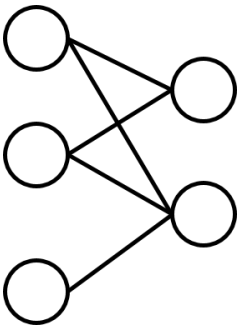
Graph k-coloring
Input:

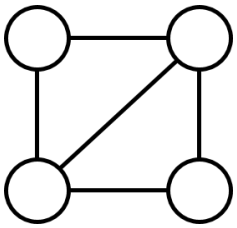
- An undirected, unweighted graph $G = (V, E)$
- A positive integer k

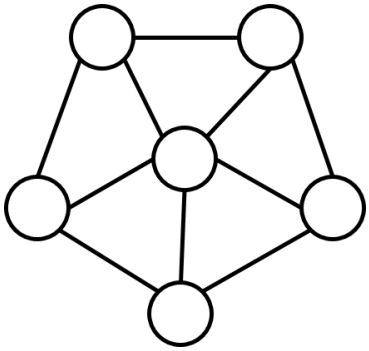
Output: Is it possible to color each node in G using at most k colors, such that if edge (u, v) exists, then nodes u and v are different colors?

Note that here we've defined a decision problem, not an optimization problem. But if we can solve the decision problem, we can solve the corresponding optimization problem (what is the smallest number of colors required?) by repeatedly solving the decision problem for different values of k.

For each of the graphs below, is the graph 2-colorable? 3-colorable? 4-colorable? Is there a simple rule that tells you how many colors you will need?







Recall from last time the *3-SAT* problem:

3-SAT
Input:

- A set of boolean variables $X = \{x_1, \ldots, x_n\}$
- A set of clauses $C_1, \ldots, C_k$, each of which is the disjunction of three of the variables in X

Output: Is the set of clauses satisfiable? That is, is there an assignment of value true or false to each of the variables in X such that each of the k clauses evaluates to true?

On Wednesday, we found a reduction from 3-SAT to Independent Set. But often (in fact, usually) it is possible to reduce the same problem to multiple different problems. *One way* of solving 3-SAT is to turn an instance of 3-SAT into an instance of Independent Set, and then solving the Independent Set problem. As we will see, another way of solving 3-SAT is to turn an instance of 3-SAT into an instance of Graph 3-Coloring, and then solving the Graph 3-Coloring problem. That is, we will show the following theorem:

Theorem 1. *3-SAT $\leq_P$ Graph 3-Coloring.*

To prove this theorem, we will take an instance of [ ] and turn it into an instance of [ ].

Once we have proved this theorem, we will know that:

1. If [ ] can be solved in polynomial time, then so can [ ].

2. If [ ] can't be solved in polynomial time, then neither can [ ].

To reduce from 3-SAT to Independent Set, we took a clause-centric view. This time, we will take a variable-centric view. Review with your group what these two perspectives were. In each view, we imagined having to make a choice. What choices did we need to make?

At this point, ask me for the next page.

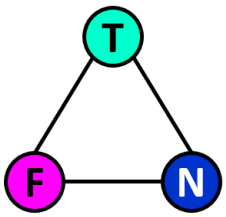
Part II: Reducing 3-SAT to 3-Coloring

The three colors that we use to color the nodes in the graph will be teal, fuchsia, and navy. We will imagine these colors corresponding to "true," "false," and "null."

We want the nodes and edges in the graph we construct to encode the constraints that:
1. We cannot set both x_i and $\bar{x}_i$ to be true; if x_i is true then $\bar{x}_i$ must be false, and vice versa.
2. In every clause, there must be at least one variable that is set to true.

Draw a "variable gadget" that enforces the constraint that we cannot set both x_i and $\bar{x}_i$ to true.

Now consider the following gadget:



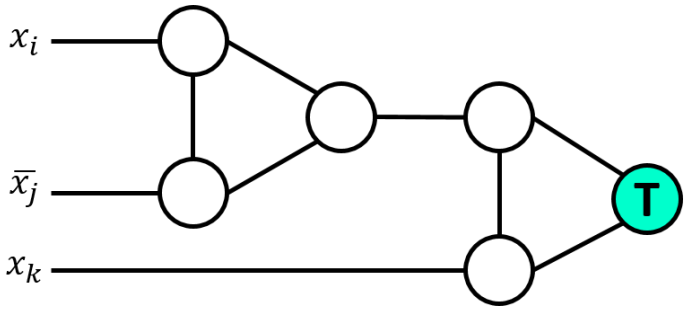
Connect a single copy of this gadget to all of your variable gadgets in a way that enforces the constraint that each node corresponding to a term must be set to teal (true) or fuchsia (false); we cannot set variable values to null (navy).

At this point show me your work and ask me for the next page.

Part III: A Clause Gadget

We have constructed a graph in which every 3-coloring corresponds to a truth assignment to the variables, and *any* truth assignment can be represented as a valid 3-coloring. We want it to be possible to represent a truth assignment as a valid 3-coloring only if it is a *satisfying* assignment.

To accomplish this, we will add one gadget to our graph for each clause. Consider the following gadget, which corresponds to the clause $x_i \bigvee \bar{x_j} \bigvee x_k$:



Each of the left-most nodes in the gadget has an edge to one of the terms in the corresponding clause. The teal node labeled T is the *same node* as the teal node in the gadget from the previous page. This is a single node that is shared among all of the clause gadgets.

Show that if the clause is satisfied, then there is a valid 3-coloring of this gadget. (There are seven possible truth assignments to the three variables that satisfy the clause. You do not need to check all of the, but you should check two or three to convince yourself that this works.)

Show that if the graph can be 3-colored, then there is a satisfying assignment of truth values to the variables. Do this by assuming that there is a valid 3-coloring in which all three "term" inputs to the gadget are colored fuchsia (false), and derive a contradiction.