I. Exam Questions. Ask them now.
Answer them

II. 2-Color It’s a well known fact that graph 3-Colorability is NP-complete. Fortunately, however, 2-COLOR is in P. In fact, it is $O(n)$. How can we use DFS to find if a graph is 2-colorable?

Begin at an arbitrary node, define a DFS function which takes in a color which will be the color it colors the current node as. Now DFS to all the neighbors with the other color. If we ever have a neighbor of the same color in the DFS, we say it is not 2-colorable. Otherwise continue for all connected components.

III. 2-SAT It’s also a well known fact that 3-SAT (Given a boolean formula of the form $(x_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_3 \lor x_1 \lor x_4) \land \ldots$, determining if there is a satisfying assignment) is NP-complete. Also fortunately, however, 2-SAT is in P. In fact, it too is $O(n)$, though perhaps this fact is less obvious than the case for 2-COLOR.

1. Consider constructing the implication graph of a 2-SAT formula. We want the property a directed edge represents an implication; i.e. a directed edge $(u, v)$ implies that if $u$ is true, then $v$ must be true. How would you go about doing this (Hint: for each variable, we want 2 vertices corresponding to $x_i$ and $\bar{x}_i$)

   For each clause $(x \lor y)$, make directed edges $\bar{x} \rightarrow y$ and $\bar{y} \rightarrow x$.

2. Show that if there is some $x_i$ and $\bar{x}_i$ are strongly connected, then the formula is unsatisfiable.

   This means that $x_i \leftrightarrow \bar{x}_i$, but we know that only exactly one can be true, so this is a contradiction.

3. Show that for each strongly connected component, either all of the predicates in the component are true or false.

   Assume there exists some node $v$ which is true. Since the component is strongly connected, then all other vertices $u$ in the component have some directed path $v \rightarrow u$, which means that $u$ must be true as well. Thus, the entire component is all true or all false.

4. Consider the DAG of strongly connected components. Assuming $x_i$ and $\bar{x}_i$ are always in different components, come up with a ‘coloring’ technique for the DAG such that we never get a $T \rightarrow F$ edge (Hint: consider the sources/sinks of this graph). This implies the inverse of (2), meaning the formula is satisfiable iff $\forall x_i$, $x_i$ and $\bar{x}_i$ are in different components.

   Color any sinks (components with no outgoing edges) true, and color the complement component (which must be a source as outgoing edges become incoming edges in the complement component) false (note if $x, y$ are strongly connected, then $\bar{x}, \bar{y}$ are in strongly connected as well, so it makes sense to talk about complement components). Now note that any ‘valid’ coloring on the rest of the graph
is also a valid coloring on this new graph, as for directed edge \((u, v)\), \(u\) being false does not constrain \(v\). Similarly, \(v\) being true does not constrain \(u\). Thus, we can delete these two nodes and repeat. Since this is a DAG, there will always be a sink, so this works. To do this in linear time, we toposort the DAG and visit components in reverse topological order. We color similarly to above, and mark all components and their complements when visited, and ignore any visited already visited nodes in this traversal.

5. Verify this indeed runs in linear time

We can find strongly connected components in linear time based on lecture. We can check component membership for each variable and its complement in constant time, so checking for all variables is linear time. Finally, if we want to find the truth assignment, we can find a topological sorting in linear time, and our coloring just iterates on that.