CS 361 Meeting 34 — 12/3/18

Announcements
1. Homework 11 due Monday. Office hours adjusted for non-Friday deadline:
   - Tom: Monday and Tuesday 2-3:30.
   - TAs: Monday 8-10 and Tuesday 8-11.
2. Tentative plans for final: Goal — no crazy time allocation problems, but no all-nighters either. Proposal — 24-hour pickup and return but max of 12 hours from start to end of work.

Review
1. Last time, we reviewed the proof of the Cook-Levin theorem and showed that the problem of determining whether a graph contained a k-clique was NP-complete by reducing 3SAT to k-clique.

Graph Coloring
1. Another problem that is NP-complete is the problem of determining how many colors are required to color the nodes of a graph in such a way that no two adjacent nodes are assigned the same color.
   - This (seemingly silly) problem has many important applications:
     - If you were building a compiler for a language with k registers and you were generating code for a method with n > k variables you might still hope you could keep all the variables in registers by analyzing the program to figure out which variables were unused at various points of the program. Given the results of such an analysis (commonly called use-def chaining) you would build a graph with one node for each variable and edges between any two variable that are both needed simultaneously at any program point. If this graph is k-colorable, then k registers is enough.
2. Stated as a language

\[
k-\text{COLOR} = \{ \langle G, k \rangle \mid G \text{ is a graph that can be colored with } k \text{ colors} \}
\]

3. We will demonstrate that \( k – \text{COLOR} \) is NP-complete by describing a mapping \( f(\phi) = \langle G, k \rangle \) that can be computed in polynomial time on a deterministic TM and such that \( \phi \in 3\text{-SAT} \iff \langle G, k \rangle \in k-\text{COLOR} \) with \( k = 1 + \) the number of variables used in \( \phi \).

4. The graph we will construct has three components (gadgets?). The first two only depend on the number of variables used in the formula:
   - A core of \( k \) nodes (where \( k = 1 + \) the number of variables in \( \phi \) that form a clique. This subgraph by itself ensures that the graph is not colorable by any few colors than one more than the number of variables in \( \phi \). The figure below shows examples of this subgraph for formulas with 2, 3, and 4 variables.
   - A pair of nodes for each variable, \( x_i \) in \( \phi \) one corresponding to the literal \( x_i \) and the other corresponding to \( \overline{x_i} \). Each of these nodes should be connected to the other and to all of the \( c_j \) nodes except \( c_i \). The diagram below illustrates how this construction works in the case that \( \phi \) contains three variables.
5. The goal of these two components is to give us two ways to color each literal while staying within the limit of $n + 1$ colors. Since the two nodes for any literal $x_i$ are connected to one another, they must be assigned different colors. Since they are each connected to $c_j$ nodes except $c_i$, the only choices left for these two nodes are the false color, $F$, and the color of $c_i$.

6. Now, for each clause, $t_k$ in $\phi$, we add one additional node and connect this node to $F$ and to all of the $x_i$ and $\overline{x}_i$ nodes that do NOT occur in the clause. Since the node for $t_k$ will be connected to the nodes for both $x_i$ and $\overline{x}_i$ if neither of these literals appears in $t_k$, it cannot be colored with either $c_i$ or $F$. On the other hand, if $x_i$ appears in $t_k$, the node for $t_k$ will be connected to the node for $\overline{x}_i$ but not to the node for $x_i$. A truth assignment in which $x_i$ is true will correspond to a coloring in which node $x_i$ has color $c_i$ while node $\overline{x}_i$ will be $F$. In this situation, it will be possible to color the node for $t_k$ with color $c_i$. On the other hand, in the coloring corresponding to an assignment in which all of the literals in $t_k$ are false, for each literal in $t_k$, there will be an edge to the node for the version of the literal $x_i$ or $\overline{x}_i$ that has been colored $c_j$. So, $t_k$ will be adjacent to node colored with $F$ and all the $c_i$s requiring an extra, k+2nd color to color the entire graph.

7. For example, if the clause in question were $x_1 \lor \overline{x}_2 \lor x_3$, then we could create a node $t_1$ for it and connect $t_1$ as shown below (the remaining nodes have been dramatically repositioned to make it easy to draw the new node and its edges).

8. In addition to repositioning the nodes, we have colored the nodes other than $t_1$ in one of the $2^3 = 8$ possible ways allowed by the graph’s structure. If you are looking at a copy of these notes printed in grayscale, don’t worry. While the construction requires multiple colors, remember that white (the color of the $F$ node) means false and any other color (i.e., gray if your copy is not colored) means true.

9. The colors in the diagram correspond to a truth assignment in which $x_1$ and $x_2$ are true while $x_3$ is false. The formula we used as an example, $x_1 \lor \overline{x}_2 \lor x_3$, would therefore include just one true literal, $x_1$. Because of the way the graph has been constructed, $t_1$ an be colored with the color of this true literal, $c_1$.

10. In general, if we connect the node for some clause $t_i$ as described, and color the variable nodes to encode some truth assignment, then if $t_i$ includes some literal involving $x_j$ that has the value true given the assignment it will be possible to color $t_i$ with $c_i$ since $t_i$ will not be connected to the node for the colored literal $x_j$ or $\overline{x}_j$. On the other hand, if the clause contained no literal that is assigned true by the assignment, then $t_i$ will be connected to nodes colored using $F$ and all
of the $c_i$’s. Therefore, we will have to use $n + 2$ colors to complete the coloring of $t_i$. 