23 More NP-Hard Problems (April 26)

In this lecture, I'll describe some more NP-hardness reductions, mostly involving graphs.

23.1 Independent Set (from Clique)

An *independent set* is a collection of vertices is a graph with no edges between them. The INDEPENDENTSET problem is to find the largest independent set in a given graph.

There is an easy proof that INDEPENDENTSET is NP-hard, using a reduction from CLIQUE. Any graph G has a *complement* $\overline{G}$ with the same vertices, but with exactly the opposite set of edges—(u, v) is an edge in $\overline{G}$ if and only if it is *not* an edge in G. A set of vertices forms a clique in G if and only if the same vertices are an independent set in $\overline{G}$. Thus, we can compute the largest clique in a graph simply by computing the largest independent set in the complement of the graph.

$$\begin{tikzcd}
\boxed{\text{graph } G} \arrow[r, "O(n)"] & \boxed{\text{complement graph } \overline{G}} \arrow[d, Rightarrow, "\textsc{IndependentSet}"] \\
\boxed{\text{largest clique}} & \boxed{\text{largest independent set}} \arrow[l, "\text{trivial}"']
\end{tikzcd}$$

23.2 Vertex Cover (from Independent Set)

A *vertex cover* of a graph is a set of vertices that touches every edge in the graph. The VERTEX-COVER problem is to find the smallest vertex cover in a given graph.

Again, the proof of NP-hardness is simple, and relies on just one fact: If I is an independent set in a graph $G = (V, E)$, then $V \setminus I$ is a vertex cover. Thus, to find the *largest* independent set, we just need to find the vertices that aren't in the *smallest* vertex cover of the same graph.

$$\begin{tikzcd}
\boxed{\text{graph } G = (V, E)} \arrow[r, "\text{trivial}"] & \boxed{\text{graph } G = (V, E)} \arrow[d, Rightarrow, "\textsc{VertexCover}"] \\
\boxed{\text{largest independent set } V \setminus S} & \boxed{\text{smallest vertex cover } S} \arrow[l, "O(n)"']
\end{tikzcd}$$

23.3 Graph Coloring (from 3SAT)

A c-*coloring* of a graph is a map $C : V \to \{1, 2, \ldots, c\}$ that assigns one of c 'colors' to each vertex, so that every edge has two different colors at its endpoints. The graph coloring problem is to find the smallest possible number of colors in a legal coloring. To show that this problem is NP-hard, it's enough to consider the special case 3COLORABLE: Given a graph, does it have a 3-coloring?

To prove that 3COLORABLE is NP-hard, we use a reduction from 3SAT. Given a 3CNF formula, we produce a graph as follows. The graph consists of a *truth* gadget, one *variable* gadget for each variable in the formula, and one *clause* gadget for each clause in the formula.

The truth gadget is just a triangle with three vertices T, F, and X, which intuitively stand for TRUE, FALSE, and OTHER. Since these vertices are all connected, they must have different colors in any 3-coloring. For the sake of convenience, we will name those colors TRUE, FALSE, and OTHER. Thus, when we say that a node is colored TRUE, all we mean is that it must be colored the same as the node T.