

GRAPH COLORING WITH A SAT SOLVER — A NIFTY ASSIGNMENT

Brent Yorgey
Dept. of Mathematics and Computer Science
Hendrix College
1600 Washington Ave., Conway, AR 72032
yorgey@hendrix.edu

Introduction and Goals

Towards the end of our Algorithms course, we spend a few days talking about NP-completeness from an algorithmic perspective. This assignment comes at the end of the unit on NP-completeness and serves three main purposes:

1. To give students hands-on experience with an NP-complete problem (graph coloring), polynomial reduction, and certificates.
2. To expose students to the idea and practice of SAT solving.
3. To give students a more practical understanding of the abstract, often implicitly assumed idea that everything can be encoded as sequences of bits.

Assignment

The *chromatic number* of a graph is defined as the smallest $k \geq 0$ such that the graph is k -colorable, that is, so that one of k colors can be assigned to each vertex in such a way that no adjacent vertices have the same color. Finding the chromatic number of a graph by brute force is infeasible even for relatively small graphs; in fact, the problem of determining whether a given graph has a k -coloring is NP-complete for $k \geq 3$. However, the problem can be reduced to that of finding a satisfying assignment for a Boolean formula that encodes the requirements for a valid graph coloring. Although also NP-complete, in practice such Boolean satisfiability problems can be solved with surprising efficiency by dedicated solvers. Students must implement the reduction themselves, turning a given graph into an appropriate collection of Boolean formulae in a particular format, which can then be handed off to a free, open-source, industrial-strength SAT solver. The students then use the output to produce a *certificate* exhibiting a valid k -coloring of the graph.

Preventing academic dishonesty presents an interesting challenge, since there would be no way to detect copying of certificates. Cheating can be prevented by giving each student their own individual graph; however, it would be tedious to generate many unique graphs and keep track of which graph is assigned to which student. Instead, each graph is generated deterministically from a unique seed value such as the student's name. Implementing this graph generation algorithm is a nontrivial example of compactly encoding a complex object (namely, a graph) as a bit sequence, and forms an integral part of the assignment. The seed value is run through a one-way hash function and the resulting 128-bit string interpreted as an undirected simple graph with 16 vertices according to a specified encoding scheme.