6.890: Algorithmic Lower Bounds — Prof. Erik Demaine
Fall 2014

Problem Set 2

Due: Wednesday, October 8th, 2014

Problem 1. The R&D department of IncompeTech, Inc. has been looking into light-based circuits. In their years of research, they've developed a single type of gate. The gate takes two inputs — each either red light, green light, or blue light — and outputs a single beam of light whose color is determined by the following table:

		input 1		
		$\boldsymbol{R}$	$\boldsymbol{G}$	$\boldsymbol{B}$
input 2	$\boldsymbol{R}$	G	R	B
	$\boldsymbol{G}$	R	B	R
	$\boldsymbol{B}$	B	R	G

IncompeTech's circuits take the form of a directed acyclic graph in which every node has in-degree 0 or in-degree 2. There are two types of nodes with in-degree 0: the inputs $x_1, \ldots, x_n$ and the constants R, G, and B. The nodes with in-degree 2 represent instances of the gate shown above. One such node produces the output (R, G, or B) of the overall circuit.

The R&D department of IncompeTech has designed several circuits which are supposed to be able to output the color red for an appropriate (unknown) selection of inputs, but the QA department hasn't figured out how to test for this. Can this property be checked in polynomial time, or is IncompeTech struggling with an NP-hard problem?

Solution: Let $f(\cdot, \cdot)$ be the function given by the problem, and define a new function $g(x) = f(x, x)$. By examining f, we see that:

$$g(x) = \begin{cases} B & \text{if } x = G \\ G & \text{otherwise} \end{cases}$$

Thus, $g(x)$ seems to behave a little bit like a negation: changing G to B and vice versa. So suppose that we assign G to be the "true" value and B to be the "false" value. Then our next goal is to construct either an AND gate or an OR gate. To make our construction simpler, we can "sanitize" our inputs $x_1, \ldots, x_n$ by applying $g(g(x))$, thus ensuring that the inputs to our AND and OR gates will be either G or B, with no need to worry about what happens when R is used.

To construct the desired gate, we start by examining the function $f(x, y)$ when its inputs are restricted to $x, y \in \{G, B\}$. That table is as follows:

	G	B
G	B	R
B	R	G

The nice thing about this truth table is that the three cases are nicely differentiated: B if there are two Gs, G if there are two Bs, and R if there is one G and one B. The not-so-nice thing is that there are three different outputs. But we know that $g(x)$ maps three outputs down to two. So we examine the function $h(x, y) = g(f(x, y)) = f(f(x, y), f(x, y))$:

	G	B
G	G	G
B	G	B

This is precisely what we want: if G represents true and B represents false, then $h(x, y)$ is an OR gate. This, combined with the NOT gate, allows us to represent arbitrary boolean circuits.

But we are not quite done: the problem asks whether the output is R, but our "true" value is G. To fix this, we add one extra gate at the end of the computation: $s(x) = f(R, x)$. This will transform an input of G to R, while leaving an input of B alone. Hence, we can simulate a boolean circuit with this tri-valued circuit, thus showing that there is a polynomial-time reduction from Circuit-SAT to this problem. Thus, the problem is NP-hard. □

Problem 2. For each of the following planar edge-coloring problems, either show that the problem is NP-hard, or show that there exists a polynomial-time algorithm for the problem (e.g., by reducing to shortest paths, minimum spanning tree, matching, network flow, etc.).

(a) Given a 3-regular planar undirected multigraph (allowing parallel edges and self-loops), color each edge either red or blue such that, at each vertex, the multiset of colors of the incident edges is $\{R, B, B\}$.

Solution: The set of red edges has the property that there is exactly one incident to each node. This is precisely the definition of a perfect matching. Hence, we may apply Edmonds' matching algorithm. If the result is a perfect matching, color the edges in the matching red, and color all other edges blue. If the result is not a perfect matching, then it is impossible to color the graph. □

(b) Given a 3-regular planar undirected multigraph (allowing parallel edges and self-loops), color each edge either red or blue such that, at each vertex, the multiset of colors of the incident edges is either $\{R, B, B\}$, $\{R, R, R\}$, or $\{B, B, B\}$.

Solution: This is even easier: just color every edge red. Then every vertex will have $\{R, R, R\}$ as the multiset of its incident edge colors, which is precisely what we wanted. □

(c) Suppose that you are given an planar undirected multigraph (allowing parallel edges and self-loops) such that each vertex has degree 3 or degree 6. Color each edge either red or blue such that, at each vertex, the multiset of colors of the incident edges is either $\{R, R, B, B, B, B\}$, $\{B, B, B\}$, or $\{R, R, R\}$.

Solution: Reduction from planar All-Positive 1-in-3 SAT. Given a variable from the 1-in-3 SAT instance that appears in k clauses, we construct $k - 2$ vertices of degree 3, and connect them in a chain. Because all of these vertices are connected, the edges they are incident to must either be all red, or all blue. The other $3(k - 2) - 2(k - 3) = 3k - 6 - 2k + 6 = k$ connections emanating from these nodes will be used to connect the variable gadget to the k clauses that the variable belongs to. Note that this variable gadget is planar (because all vertices are arranged in a chain), and all vertices are on the outside face of the gadget, so when we replace the variable node from the original graph of clauses and variables, we will maintain planarity.

To construct a clause gadget, we use two vertices: one of degree 6, and one of degree 3, connected to each other by three parallel edges. Because the vertex of degree 6 cannot be incident to three red edges, the vertex of degree 3 must be a $\{B, B, B\}$ vertex. So the remaining three edges emerging from the clause gadget must be $\{R, R, B\}$ — exactly one of them must be blue. Hence, if we connect our clause gadgets to our variable gadgets as specified by the original planar 1-in-3 SAT problem, we know that the resulting graph can be colored in this way if and only if the original 1-in-3 SAT problem is satisfiable. Furthermore, it is plain to see that this graph is planar. We have already seen that the graph remains planar when each variable node is replaced with a chain of nodes of degree 3, and our clause gadget consists of adding one neighbor to the original clause node, which can be done by placing the neighbor in one of the adjacent faces. Hence, the problem is NP-hard for planar graphs. □

Problem 3. Suppose that you are given an arbitrary planar graph, and a subset $Q \subseteq V$ of the vertices in the graph labeled "exactly 3 blue". Let $N(v)$ denote the set of neighbors of a vertex v. We want to color each vertex in the graph either red or blue such that, if we examine a vertex v labeled "exactly 3 blue", the number of blue nodes among $\{v\} \cup N(V)$ is exactly 3. (Unlabeled vertices are unconstrained.) Can this problem be solved in polynomial time, or is it NP-hard?

Solution: Reduction from Planar All-Positive 1-in-3 SAT. Start with the graph of clauses and variables. For each clause C_i, add a single neighboring vertex s_i, and add a third vertex t_i that is connected to s_i by an edge (s_i, t_i). (Note that this can be done without changing the planarity of the graph by placing both added nodes in one of the faces adjacent to the clause node.) Label both s_i and C_i with "exactly 3 blue". By construction, there are exactly three nodes in $\{s_i\} \cup N(s_i)$: s_i, t_i, and C_i. Hence, all three nodes must be blue. This means that, in $\{C_i\} \cup N(s_i)$, we know that there are exactly two non-variable nodes, both of which are blue: C_i and s_i. So exactly one of vertices incident to the clause must be blue. By using blue to represent the value of true, this ensures that we have correctly encapsulated the 1-in-3 SAT constraint. □

Problem 4. *Graduating is Hard, or: Why Does 6.890 Conflict With So Many Things?*
Students at the Miskatonic Institute of Technology sometimes find it hard to figure out whether they will graduate on time. The Institute has a set $\mathcal{C}$ of classes it offers, though not all classes are taught every semester. To graduate with a specific major, you must complete all of the courses required for that major. The courses you can take may also have additional restrictions, such as conflicts or prerequisites, described below.

(a) The major in Computational Metaphysics simply requires completing a specific set of required courses. Each course $c \in \mathcal{C}$ is offered in a certain set of semesters S_c. In each semester i, certain pairs of classes conflict, as given by an undirected graph $(\mathcal{C}, X_i)$. In each semester, you can take any set of classes that are offered during that semester and have no pairwise conflicts. Prove that it is NP-hard to decide whether you can graduate with a major in Computational Metaphysics.

Solution: Reduction from 3-SAT. We create one class per variable and one class per clause, which are all required. Each variable class v_i is held in two semesters, $2i - 1$ and $2i$, representing true and false respectively. Clause classes c_i are held during the three semesters corresponding to the negation of the literals they contain, and conflict with the variable class in that semester (but not with other clause classes).

Suppose that we have a satisfying assignment, and wish to convert that to a schedule. For each variable $x_i = T$, take the corresponding variable class v_i during semester $2i-1$. For each variable $x_i = F$, take the corresponding variable class v_i during semester $2i$. For each clause, we know that there is at least one literal in the clause that is set to true; therefore, at least one of the negations of the three literals must be false, and there will be no variable class during the semester corresponding to that literal. This allows us to take the corresponding clause class during that semester.

Conversely, suppose that we have a schedule that allows us to graduate. We can convert it to an assignment by setting $x_i = T$ if and only if v_i is held during semester $2_i - 1$. The clause class c_i must be scheduled during one of the three semesters it is offered. Suppose that chose to take it during some semester $2j-1$. Then, because it conflicts with all variable classes, we could not have taken v_j in the same semester, which means that $x_i = F$ in our assignment. The fact that c_i was scheduled during semester $2j-1$ in the first place means that the corresponding clause contains the literal $\overline{x_j}$, so the clause will be satisfied. A similar argument applies if we were able to take the clause class during some semester $2j$. □

(b) You notice that none of the classes required for a major in Eldritch Lore ever have conflicts with each other. However, due to the nature of the material, students are restricted to taking no more than four courses in any given semester. As before, each course $c \in \mathcal{C}$ is offered in a set of semesters S_c. In addition, each course $c \in \mathcal{C}$ has a prerequisite list $P_c \subseteq \mathcal{C}$. A student must have taken all of the prerequisites in P_c before they can take c. The graph of prerequisites form a (potentially disconnected) directed acyclic graph over the courses. Each course $c \in \mathcal{C}$ also has a list of courses R_c which can be used as substitutions for c, which fulfill all the prerequisite and major requirements that c does. Prove that it is NP-hard to decide whether you can graduate with a major in Eldritch Lore.

Solution: Once again we reduce from 3SAT. Each clause is represented by a required course in the major. The clauses have three possible substitutions, so we only need to take one of those classes to satisfy the clause. Each of the substitutions represents a literal and has it's corresponding variable course as a prerequisite. Each variable is represented by two courses, one for true and one for false. Each of these has four prerequisites, all eight of which are only offered in a single semester. This ensures that one can only fulfill the prerequisites for either the true or the false instance of the variable. Variable courses are not required (except as prerequisites for the clause courses), and have no substitutions. The variables prerequisite semesters are all offered sequentially, followed by the variables, and finally the literals. □

Problem 5. An ε-imperfect 2-coloring is a 2-coloring of an undirected graph in which no more than an ε fraction of the edges have endpoints of the same color. Show that it is NP-complete to find an ε-imperfect 2-coloring for some constant ε strictly between 0 and 1 (i.e., ε cannot have any dependence on the input but can be whatever constant you want other than 0 or 1).

Solution: We reduce from Monotone NAE-SAT where every clause has exactly 3 variables, all 3 of which are unique. We let ε equal $\frac{1}{6}$. We represent each clause by three nodes connected in a triangle. Thus if all three nodes are the same color we have three monochromatic edges; however, if only two nodes are the same color we will only have one monochromatic edge. Next, we create a node for every variable and connect it to every node representing an instance of that variable. If any of the literals are the same color as the variable it creates a monochromatic edge. The number of edges in this graph is exactly six times the number of clauses. The graph must contain at least one monochromatic edge in each clause. Thus the graph is only $\frac{1}{6}$-imperfect 2-colorable if no other edges are monochromatic. This can only occur if all variable nodes are the opposite color of their literals, and thus all the literal are the same color, and if all the clauses have two different colors in them. We pick one color to be true and one to be false, when translating to NAE-SAT. $\square$