

Solution Set 5

Posted: February 28

1. (a) First, note that 3-COLORABLE is in NP because given an assignment of colors to the vertices of the graph G , we can verify in polynomial time that each edge has different colors at its endpoints.

To show that 3-COLORABLE is NP-hard, we reduce from 3SAT. Given an instance ϕ of 3SAT, our reduction produces the following graph G : we have three special vertices A, B, C , and one vertex for each literal, x_i and $\neg x_i$. We have a triangle on $A, x_i, \neg x_i$ for each i . Notice that any 3-coloring of the graph so far must assign distinct colors to B and C , which we will call (suggestively) T and F , respectively. Also each pair x_i and $\neg x_i$ must be colored with T and F , respectively, or F and T , respectively. We can thus think of the coloring of the “literal vertices” as a truth assignment.

Now, for each clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ appearing in ϕ , we add a copy of the gadget from the problem set. We identify the three grey vertices on the left with the three vertices ℓ_1, ℓ_2 , and ℓ_3 . We identify the grey vertex on the right with the vertex B . This reduction runs in polynomial time.

If we started with a YES instance of 3SAT, then we claim that the reduction produces a YES instance of 3-COLORABLE. Consider a satisfying assignment for ϕ . We color A, B , and C with the colors RED, T and F , respectively. We then color x_i and $\neg x_i$ with colors T and F , respectively, if x_i is true in the satisfying assignment, and we color x_i and $\neg x_i$ with colors F and T , respectively, if x_i is false in the satisfying assignment. So far this is a valid 3-coloring. Now notice that every one of the clause gadgets has among its left three grey nodes at least one node that is colored T (since every clause has at least one true literal in the satisfying assignment). Moreover, its right grey node is colored T . Thus using the observation in the problem set we can extend the 3-coloring to a 3-coloring of the clause gadgets, obtaining a 3-coloring of the entire graph G .

Now, if G is a YES instance of 3-COLORABLE, then we claim that the reduction started with a satisfiable formula ϕ . Suppose we have a 3-coloring of G , then A, B , and C must be colored with three distinct colors, and let us call the color assigned to B “ T ” and the color assigned to C “ F ”. As noted each pair x_i and $\neg x_i$ must be colored with T and F respectively, or F and T respectively. For each clause gadget, the rightmost grey node is colored with T , and so by the observation in the problem set, the only way it can be 3-colored is if at least one of its leftmost grey nodes are colored with T . But this means that we can set x_i to true if it is colored T and x_i to false if it is colored F , and the resulting assignment must satisfy every clause. Thus the formula ϕ is satisfiable, as required.

- (b) First, note that K-COLORABLE is in NP because given an assignment of colors to the vertices of the graph G , we can verify in polynomial time that each edge has different colors at its endpoints.

To show that k -COLORABLE is NP-hard (for $k \geq 4$), we reduce from 3-COLORABLE. Given an instance G of 3-COLORABLE, our reduction produces the following graph G' : we add $k - 3$ nodes to G , and connect every pair of them with an edge. We also add an edge between every node in G and every one of the $k - 3$ additional nodes. The reduction runs in polynomial time.

It is easy to see that if G is 3-colorable then we can use the same coloring to color with 3 colors all of the nodes of G' except for the additional nodes; we then use the remaining $k - 3$ colors for the additional nodes.

In the other direction, if G' is k -colorable, then we observe that we must use $k - 3$ distinct colors for the $k - 3$ additional nodes. None of the remaining nodes can use any of these colors, because there is an edge between every one of the $k - 3$ added nodes and every node that came from G . Thus the remaining nodes must be 3-colorable, which implies that G is 3-colorable.

2. First, observe that SUBGROUP ISOMORPHISM is in NP, because if we are given a specification of the subgraph of G and the mapping between its vertices and the vertices of H , we can verify in polynomial time that H is indeed isomorphic to the specified subgraph of G .

To show SUBGRAPH ISOMORPHISM is NP-hard, we reduce from CLIQUE. Given an instance (G, k) of CLIQUE, we produce the following instance of SUBGRAPH ISOMORPHISM: $(G, H = K_k)$, where K_k is the complete graph on k vertices. This reduction runs in polynomial time.

It is clear that G contains a clique of size k if and only if G contains a subgraph isomorphic to H (these are just two ways of saying the same thing). Thus we have shown that SUBGRAPH ISOMORPHISM is NP-hard, as desired.

3. $(3, 3)$ -SAT is in NP for the same reason 3-SAT is. We show that it is NP-hard by reducing from 3-SAT.

Given a 3-CNF formula ϕ , we perform the following transformation to obtain 3-CNF ϕ' : for each x_i we replace the m_i occurrences of x_i with fresh variables $y_{i,1}, y_{i,2}, \dots, y_{i,m_i}$, and we add the following m_i clauses:

$$\begin{aligned} &(\neg y_{i,1} \vee y_{i,2}) \\ &(\neg y_{i,2} \vee y_{i,3}) \\ &(\neg y_{i,3} \vee y_{i,4}) \\ &\quad \dots \\ &(\neg y_{i,m_i-1} \vee y_{i,m_i}) \\ &(\neg y_{i,m_i} \vee y_{i,1}) \end{aligned}$$

Note that the extra clauses are logically equivalent to:

$$y_{i,1} \Rightarrow y_{i,2} \Rightarrow \dots \Rightarrow y_{i,m_i} \Rightarrow y_{i,1}$$

and hence any assignment satisfying ϕ' must set all of these variables to the same value. Such an assignment can be turned into a satisfying assignment for ϕ by setting $x_i = y_{i,1}$ for all i .

Similarly, any assignment satisfying ϕ can be turned into a satisfying assignment for ϕ' by setting $y_{i,j} = x_i$ for all i, j . This shows that “yes maps to yes” and “no maps to no,” and thus (3, 3)-SAT is NP-hard as required.

4. We first prove the following lemma regarding the 10 clauses given on the problem set: any assignment to x, y, z that sets at least one to true can be extended to an assignment to x, y, z, w that satisfies at most 7 of the 10 clauses, while the assignment to x, y, z that sets all of them to false cannot be extended to an assignment to x, y, z, w that satisfy more than 6 of the 10 clauses. Moreover it is impossible to satisfy more than 7 of the 10 clauses simultaneously.

The second part is easier: if x, y, z are all false, then setting w to true satisfies 4 clauses, while setting w to false satisfies 6 clauses. For the first part, observe that the clauses are symmetric in x, y, z . Thus we need only consider 3 cases: (a) exactly one of x, y, z is true, (b) exactly 2 of x, y, z are true, and (c) exactly 3 of x, y, z are true. In case (a), we can set w to false to satisfy 1 clause in the first row, 3 in the second row, and 3 in the last row, for a total of 7. In case (b), we can set w to true to satisfy 3 clauses in the first row, 2 in the second row, and 2 in the last row, for a total of 7. In case (c) we can set w to be true to satisfy 4 clauses in the first row, no clauses in the second row, and 3 clauses in the last row, for a total of 7. In each of these three cases, we see that setting w the other way doesn't help: in case (a) we would satisfy only 6 clauses; in case (b) we would satisfy 7 clauses; in case (c) we would satisfy only 6 clauses. This proves the “moreover” part of the lemma.

Now we proceed with proving MAX2SAT is NP-complete. Observe that it is in NP, because given a formula ϕ and an integer k , together with an assignment A , it is easy to verify in polynomial time that the assignment satisfies at least k of the clauses of ϕ .

Now, we show MAX2SAT is NP-hard by reducing from 3SAT. Given an instance ϕ of 3SAT with m clauses, we produce a 2-CNF formula ϕ' by replacing each clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ with the 10 clauses in the problem set, with ℓ_1, ℓ_2, ℓ_3 in place of x, y, z . We use a fresh variable w for each clause of ϕ . The reduction sets the bound $k = 7m$. This reduction runs in polynomial time (it produces a 2-CNF ϕ' with $10m$ clauses).

Suppose ϕ is satisfiable by an assignment A . Then by the lemma above, we can extend A to an assignment to ϕ' that satisfies at least $k = 7m$ clauses of ϕ' simultaneously, which implies that (ϕ', k) is a YES instance of MAX2SAT.

Now, suppose that there is an assignment that simultaneously satisfies at least $k = 7m$ clauses of ϕ' . Since the maximum number of clauses that can be satisfied within each group of 10 clauses is 7, this means that *every* group of 10 clauses has 7 clauses satisfied by the assignment. But by the lemma this means that this same assignment must satisfy every clause of ϕ , because if it didn't satisfy even a single clause, then the associated group of 10 clauses of ϕ' would only have 6 clauses satisfied by the current assignment. Thus we conclude that ϕ must be satisfiable.