Bram Schuijff
CSCI 387, §S01/02
Homework 6
April 1st, 2025 by 10:30am

1. To demonstrate that DOUBLESAT is NP-complete, we show first that DOUBLESAT is in NP and then that SAT is polynomial-time reducible to DOUBLESAT. To show that DOUBLESAT is in NP, we can produce a polynomial time verifier for DOUBLESAT. Given a witness, because we know that SAT is in NP, we can run the polynomial time verifier for SAT and pass it the boolean expression and our first assignment. If it rejects, reject. If it accepts, we run the polynomial time verifier for SAT on our boolean expression and the second assignment. If it rejects, reject. If it accepts, we accept because there are at least two satisfying assignments. This is clearly polynomial time because SAT is in NP, meaning it has a polynomial time verifier. Therefore, DOUBLESAT is in NP.

   To demonstrate that DOUBLESAT is NP-complete, we perform a polynomial-time reduction from SAT to DOUBLESAT. Given a boolean expression X, we can add an additional variable D such that D is not present in X. Then, our new boolean expression becomes

   $$Y = X \wedge (D \vee \neg D)$$

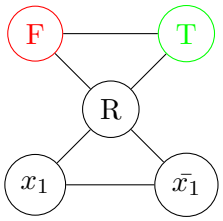
   Clearly, if X has a satisfying assignment, then Y has at least two satisfying assignments: one where D is False and one where D is True. Meanwhile, if X has no satisfying assignments, then Y will not either. This reduction is clearly polynomial time because producing Y only takes as long as it takes to read X plus a constant factor for appending our new clause. Therefore, because DOUBLESAT is in NP and it reduces in polynomial time to SAT and SAT is NP-complete, DOUBLESAT is NP-complete.

2. If P = NP, then all languages in NP are also in P. This satisfies the first condition for a language being NP-complete. Now, we need to either demonstrate that any language in P is polynomial-time reducible to an NP-complete language or that any language in P is polynomial-time reducible to any other language in P. We will do the latter. Let A, B be two languages in P such that $A, B \notin \{\varnothing, \Sigma^*\}$. Let us select two string $b, \bar{b}$ such that $b \in B$ and $\bar{b} \notin B$. To produce a polynomial-time reduction to B, for any string w, run A on w. Because A is in P, this can be done in polynomial time. If A rejects, $f(w) = \bar{a}$; otherwise, $f(w) = a$. Because a and $\bar{a}$ are constant size, transcribing them can be done in polynomial time. Thus, this produces a reduction from any language in P to any other language that isn't in P so long as neither is the empty set or Σ^*.

   The reason $\varnothing$ and Σ^* are not NP-complete if P = NP is because of the definition of polynomial-time mapping reducibility which is that for every w, $w \in A \iff f(w) \in B$ for two languages A, B. Therefore, for any other language A such that $A \notin \{\varnothing, \Sigma^*\}$, suppose we are trying to construct a mapping to $\varnothing$. Then, because $A \neq \varnothing$, there must exist a string $a \in A$. However, by its nature, we cannot assign an $f(a) \in \varnothing$ which violates the if any only if condition of the definition of polynomial-time mapping reducibility. Similarly, because $A \neq \Sigma^*$, there must exist at least one string $\bar{a} \notin A$. However, we cannot assign an $f(\bar{a}) \notin \Sigma^*$ which once again violates the if and only if condition. Therefore, $\varnothing$ and Σ^* are not NP-complete even if P = NP.

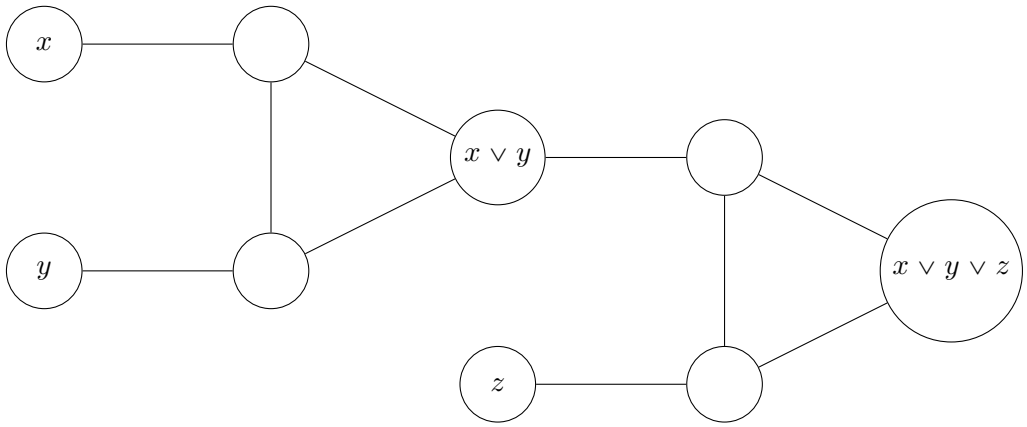
3. We demonstrated that 3COLOR is in NP in question 4 of homework 5. Now, all that remains is to show that 3COLOR is NP-complete. We do so by creating a reduction from 3SAT to 3COLOR.

   First, we need a variable gadget. We can establish the following:



Where F is the color that denotes False, T is the color that denotes True, and R is the color that denotes whatever our third color is. This construction guarantees that either x_1 or $\bar{x_1}$ is True while the other is False. Therefore, if our graph has a valid coloring, then for each variable in our boolean expression, either it is True or its negation is True.

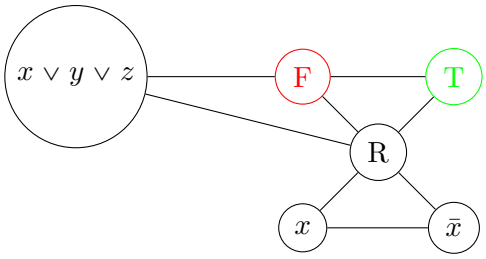
For our clause gadget, we already have an OR gadget which we can chain together as follows:



In this construction, there exists a coloring whereby $x \vee y \vee z$ is colored as True if one of x, y, z is colored True. Meanwhile, if all of x, y, z are colored False, there are no ways for $x \vee y \vee z$ to be True. To see why this is the case, suppose without loss of generality that between x, y, x is colored True. We can do this without loss of generality because the gadget is symmetrical between x and y. Then, if the node to the right of x is colored to False and the node to the right of y is colored R, we can color $x \vee y$ as True. Because the gadget is symmetrical for $x \vee y$ and z and the gadget is the same as the one that contained $x \vee y$, we conclude that if one of x, y, z is True, this construction yields a coloring such that $x \vee y \vee z$ is True.

Suppose now that all of x, y, z are False. Then, one of the nodes to the right of x or y must be colored True and the other must be colored as R. This leaves $x \vee y$ to be colored as False. Similarly, one of the nodes to the right of $x \vee y$ or z must be colored True and the other must be colored R, leaving $x \vee y \vee z$ to be colored as False. Therefore, in all cases, $x \vee y \vee z$ is False if all of x, y, z are False.

All that remains is to connect our OR gadget to the rest of our graph. We do so through the following construction:



We attach the final node in all other clause widgets to R and F in the same way. Similarly, we attach all variable widgets to R in the same way. I have omitted them from this graph for readability.

This reduces 3COLOR to 3SAT because a valid coloring will ensure that every clause in our boolean expression evaluates to True and that there are no contradictory statements. Therefore, because 3SAT is NP-complete and there exists a polynomial-time 3COLOR is NP-complete.

4. To demonstrate that this SET-SPLITTING is in NP, we can see immediately that a decider is produced by nondeterministically assigning each element of S either red or blue. This is certainly polynomial time in $|S|$ and therefore SET-SPLITTING is in NP.

   To demonstrate that SET-SPLITTING is NP-complete, we perform a reduction from 3SAT. We are trying to now produce a set S and a collection of subsets $C_1, C_2, ..., C_k$ of S that model a 3SAT problem. To do so, if we have a 3SAT problem with n distinct variables, we let $|S| = 2n$ such that $S = \{x_1, \bar{x_1}, x_2, \bar{x_2}, ..., x_n, \bar{x_n}\}$. Then, our variable widget is that we have n subsets $V_1, V_2, ..., V_n$ such that $V_i = \{x_i, \bar{x_i}\}$ for each $i \in \{1, ..., n\}$. This guarantees that in a valid split, either x_i is red and $\bar{x_i}$ is blue or x_i is blue and $\bar{x_i}$ is red.

   For our clause widget, it is slightly challenging. Say for instance that we arbitrarily select blue as being True and red as being False. Then, if we simply choose subsets that contain each of the variables in a clause, then the case where each variable in a clause is True would be an invalid solution. Thus, we have an additional subset $I = \{T, F\}$. From there, if our boolean expression has m clauses, we simply create m additional subsets $C_1, C_2, ..., C_m$ such that $C_j = \{a, b, c, F\}$ where a, b, c are the variables in the j-th clause for all $j \in \{1, ..., m\}$. Then, whichever color F is not assigned corresponds to a value of True in that coloring. In addition, we can arbitrarily select either blue or red as True because if there exists a valid coloring where F is assigned red, then we can simply flip the coloring of every element and obtain another valid coloring where F is assigned blue instead. Then, if there is a solution to this set splitting problem, there can be no contradictory statements—either x_i or $\bar{x_i}$ is True—and in each clause, at least 1 variable is True. In conclusion, we produce $2n + m + 1$ subsets.

   Therefore, if there exists a valid set split for this construction, there exists a valid assignment for any 3SAT boolean expression. We know that this reduction is polynomial time because reading through a boolean expression in 3-cnf takes $O(m)$ time if m is the number of clauses in that expression. Copying every variable and its negation into S takes $O(n)$ time if n is the number of distinct variables. Creating the variable widget subsets again also takes $O(n)$ time. Finally, creating the clause widgets and I takes at most $O(m)$ time. Taken together, these terms are polynomial in the length of the 3SAT expression. Therefore, because SET-SPLITTING is in NP and reduces in polynomial time to 3SAT and 3SAT is NP-complete, we have that SET-SPLITTING is NP-complete.

5. To formulate this problem as a language, we have:

$$\text{EXAM-SCHEDULE} = \{\langle \mathcal{S}, H, k\rangle \mid \mathcal{S} = \{S_1, S_2, ..., S_l\} \text{ is a finite collection of subsets of } H \text{ for some } l > 0 \text{ such that each element of } H \text{ must be colored in one of } k \text{ colors so that no } S_i \text{ has two elements colored in the same color.}\}$$

One might recognize this problem as an instance of k-COLOR which is a generalization of 3COLOR. To formally define k-COLOR, we do:

$$k\text{-COLOR} = \{\langle G, k\rangle \mid G \text{ is colorable with } k \text{ colors}\}$$

To prove that EXAM-SCHEDULE is NP-complete, we first demonstrate that k-COLOR is NP-complete. To do this, we first show that k-COLOR is in NP. We produce a polynomial time verifier which, when given a witness, first checks each vertex to ensure that there are at most k colors in our graph which can be done in $O(|V|)$ time. If there are more than k colors, reject. Otherwise, it goes through each edge in our graph and checks whether there are any two adjacent vertices that share the same color. If there are, reject. Otherwise, accept. There are at most $O|V|^2)$ edges in our graph and therefore this verifier is polynomial time in the size of our graph.

To demonstrate that k-COLOR is NP-complete, we perform a reduction from 3COLOR. We know that 3COLOR is NP-complete by problem 3. Given a graph $G = (E, V)$, we can simply pass $k = 3$ and our reduction is complete. Therefore, there exists a polynomial-time reduction from 3COLOR to k-COLOR. Thus, k-COLOR is NP-complete.

Now, to demonstrate that EXAM-SCHEDULE is NP-complete, we first show that it is in NP. To do so, we produce a polynomial-time verifier for EXAM-SCHEDULE. Given $\langle \mathcal{S}, H, k, w\rangle$, we go through each $h \in H$ to make sure that there are at most k colors which can be done in $O(|H|)$ time. If there are more than k colors, reject. Otherwise, we go through each $S_i \in \mathcal{S}$ and ensure that there are no two elements in S_i with the same color. If there are, reject. Otherwise, accept. This can be done in $O(|\mathcal{S}| \cdot |H|)$ time. Therefore, this verifier is polynomial-time in the length of our input and EXAM-SCHEDULE is in NP.

To demonstrate that EXAM-SCHEDULE is NP-complete, we perform a reduction from k-COLOR. Given $\langle G, k\rangle$, we let $|H| = |V|$. Then, for each $v_i \in V$, for each e_j attached to v_i, if $j > |\mathcal{S}|$, add a new S_j to $\mathcal{S}$ and add j to S_j. Because $j < |V|$, we guarantee that this value is in H and therefore $S_j \subseteq H$. Otherwise, add j to S_j.