CS5371 Theory of Computation

Homework 5 (Suggested Solution)

1. Prove that if $P = NP$, then $PATH$ is NP-complete.

   Ans. If $P = NP$, we claim that every language in NP can be reduced to $PATH$ in polynomial time. Then, together with the fact that $PATH$ is in NP, we have $PATH$ is NP-complete.

   To prove our claim, we shall show SAT can be reduced to $PATH$ in polynomial time. Firstly, since $P = NP$, there exists a decider D for SAT that runs in polynomial time. Based on this, consider the following TM F that computes a reduction f from SAT to $PATH$:

   F = "On input $\langle \phi \rangle$,
   1. Run D on $\langle \phi \rangle$.
   2. If D accepts $\langle \phi \rangle$, construct a graph G containing two vertices s and t, with an edge $\{s, t\}$ joining them.
   3. Otherwise, if D rejects $\langle \phi \rangle$, construct a graph G with two isolated vertices s and t.
   4. In either case, output $\langle G, s, t \rangle$."

   It is easy to check that $\langle \phi \rangle \in SAT \Leftrightarrow \langle G, s, t \rangle \in PATH$. Also, the above reduction takes polynomial time. This completes the proof of the claim.

2. Let $LPATH$ denote the language:

   $$LPATH = \{\langle G, s, t, k \rangle \mid G \text{ contains a simple path of length at least } k \text{ from } s \text{ to } t\}.$$

   Ans. Firstly, $LPATH$ is in NP because a certificate for $\langle G, s, t, k \rangle$ simply consists of the sequence of edges in a simple path from s to t with length at least k, so that for this kind of certificate, we can find a corresponding polynomial time DTM verifier.

   To further show that every NP problem can be reduced $LPATH$ in polynomial time, we shall reduce $HAMPATH$ to $LPATH$. Consider the following TM F that computes a reduction f from $HAMPATH$ to $LPATH$:

   F = "On input $\langle G, s, t \rangle$,
   1. Output $\langle G, s, t, n-1 \rangle$, where n is the number of vertices in G."

   Firstly, if there is a hamiltonian path from s to t in G, the path would have length $n - 1$ so that $\langle G, s, t, n-1 \rangle$ is in $LPATH$. On the other hand, if $\langle G, s, t, n-1 \rangle$ is in $LPATH$, the simple path from s to t has length $n - 1$, so that it must be hamiltonian. Thus,

   $$\langle G, s, t \rangle \in HAMPATH \Leftrightarrow \langle G, s, t, n-1 \rangle \in LPATH.$$

   Also, it is obvious that the above reduction runs in polynomial time. This implies $HAMPATH$ is polynomial-time reducible to $LPATH$. Thus, $LPATH$ is NP-complete.

3. Let S be a finite set and $C = \{C_1, C_2, \ldots, C_k\}$ be a collection of subsets of S, for some $k > 0$. We say S is *two-colorable* with respect to C if we can color the elements of S in either red or blue, such that each subset C_i contains at least a red element and at least a blue element.

   Let *2COLOR* denote the language:

   $$\mathit{2COLOR} = \{\langle S, C\rangle \mid S \text{ is two-colorable with respect to } C\}.$$

   Show that *2COLOR* is NP-complete.

   Ans. It is easy to show that *2COLOR* is in NP (how?). To show that every NP language can be reduced to *2COLOR* in polynomial time, we shall use reduction from $\neq SAT$.

   Consider the following TM F that computes a reduction from $\neq SAT$ to *2COLOR*:

   $F =$ "On input formula $\langle \psi \rangle$,
   1. For each variable x in ψ, create two variables s_x and s'_x in S.
      Also, create a subset $\{s_x, s'_x\}$ of C.
   2. For each clause γ_i in ψ, create a subset c_i of C such that
      (i) if $x \in \gamma_i$, $\quad s_x \in c_i$;
      (ii) if $\neg x \in \gamma_i$, $\quad s'_x \in c_i$.
   3. Output $\langle S, C\rangle$."

   Firstly, if there is a satisfying not-all-equal assignment (say, A) for ψ, it is easy to obtain a 2-coloring for the variables in S as follows: If x is assigned true in A, we color s_x to red and s'_x to blue; otherwise, we color s_x to blue and s'_x to red. Under this coloring, each subset $\{s_x, s'_x\}$ must contain 2 colors, while each subset c_i also contains 2 colors (because γ_i is not-all-equal under the assignment A). Thus, $\langle S, C\rangle$ is in *2COLOR*.

   On the other hand, if $\langle S, C\rangle$ is in *2COLOR*, we can obtain a satisfying not-all-equal assignment for ψ as follows: Fix a 2-coloring scheme for $\langle S, C\rangle$. If s_x is colored red, assign x to true in ψ. Otherwise, assign x to false. Since c_i contains two colors, the corresponding clause γ_i in ψ must be not-all-equal under the above assignment. This implies that every clause in ψ will be not-all-equal, so that ψ has a satisfying not-all-equal assignment.

   In summary, we have

   $$\langle \psi \rangle \in \neq SAT \quad \Leftrightarrow \quad \langle S, C\rangle \in \mathit{2COLOR}.$$

   Also, the above reduction takes polynomial time to run. Thus, $\neq SAT \leq_P \mathit{2COLOR}$, so that *2COLOR* is NP-complete.

4. (Further Studies: No marks) Let ϕ be a cnf-formula. An assignment to the variables of ϕ is called *not-all-equal* if in each clause, at least one literal is TRUE and at least one literal is FALSE.

   Let $\neq SAT$ be the language:

   $$\neq SAT = \{\langle \phi \rangle \mid \phi \text{ is a cnf-formula which has a satisfying not-all-equal assignment}\}.$$

   Show that $\neq SAT$ is NP-complete.

Ans. It is easy to check that $\neq SAT$ is in NP. It remains to show that every NP language is polynomial-time reducible to $\neq SAT$. To do so, we shall reduce *CNF-SAT* to $\neq SAT$.

Before that, we first notice that for any formula ϕ, if A is a satisfying not-all-equal assignment, then the negation of A is also a satisfying not-all-equal assignment.[1] For instance, let

$$\phi = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) \wedge (x \vee y \vee \neg z).$$

Then, $A = (x = 0, y = 1, z = 0)$ is a satisfying not-all-equal assignment. On the other hand, the negation of A, which is $(x = 1, y = 0, z = 1)$, is also a satisfying not-all-equal assignment.

Now, the reduction is as follows. Let

$$C_i = (x_1 \vee x_2 \vee \cdots \vee x_k)$$

be the ith clause in an instance of *CNF-SAT*. We shall replace clause C_i with two clauses

$$D_i = (x_1 \vee x_2 \vee \cdots \vee x_{k-1} \vee z_i) \quad \text{and} \quad E_i = (\neg z_i \vee x_k \vee b),$$

where z_i is a new variable corresponding to C_i, and b is a global variable shared by other D_j's and E_j's.

Let ϕ be the original cnf-formula, and ψ be the transformed cnf-formula. First, if the original formula ϕ is satisfiable, it is easy to obtain a satisfying not-all-equal assignment for the transformed formula ψ as follows:

(a) Use the same assignment for the variables that appear in ϕ;
(b) For clause D_i, set $z_i = \neg(x_1 \vee x_2 \vee \cdots \vee x_{k-1})$;
(c) Set b to be false;

Under this assignment, for each i, the clause D_i must be not-all-equal. Also, we know that either x_k is true or $(x_1 \vee x_2 \vee \cdots \vee x_{k-1})$ is true (why?). The latter case implies that z_i is false. Then, in both cases, we know that E_i must be not-all-equal (because b is set to false). Thus, $\langle\phi\rangle$ is in *CNF-SAT* implies $\langle\psi\rangle$ is in $\neq SAT$.

On the other hand, if $\langle\psi\rangle$ is in $\neq SAT$, let A be a satisfying not-all-equal assignment for ψ. If b is set to false in A, we claim that with the same assignment for the variables that appear in ϕ, ϕ will become satisfied. Consider C_i: if x_k is true, C_i is satisfied immediately. Otherwise, we know that E_i is not-all-equal, so that z_i is true. In this case, $\neg z_i$ is false in D_i so that $(x_1 \vee x_2 \vee \cdots \vee x_{k-1})$ must be true. This in turn would imply C_i is satisfied in ϕ. In summary, if b is set to false in A, then ϕ is satisfiable.

Next, if b is set to true in A, we know that the negation of A is also a satisfying not-all-equal assignment for ψ. Then, we can proceed with the same reasoning and show that ϕ is also satisfiable (using the negated assignment).

Thus, $\langle\psi\rangle$ is in $\neq SAT$ implies $\langle\phi\rangle$ is in *CNF-SAT*, so that

$$\langle\phi\rangle \in \textit{CNF-SAT} \quad \Leftrightarrow \quad \langle\psi\rangle \in \neq SAT.$$

Finally, the reduction takes polynomial time to run, so that we have proven *CNF-SAT* $\leq_P$ $\neq SAT$. This completes the proof.

[1] The proof is very straightforward: a literal is assigned true in A if and only if it is assigned false in the negation of A. Since A guarantees each clause has at least one false, the negation of A thus guarantees each clause has at least one true so that it is also satisfying. Moreover, A guarantees each clause has at least one true, so that the negation of A guarantees each clause must be not-all-equal.