Homework 6

Problem 1. Prove that if P = NP, then NP = coNP.

Solution. Suppose P = NP. Then for any problem $A \in$ NP, $A \in$ P. So there exists a TM T_1 that can decide A in polynomial time. We construct a new TM T_2, which simulates T_1 and gives the opposite answer (when T_1 accepts, T_2 rejects, and vice versa). By definition T_2 will decide $\bar{A}$. Since T_1 runs in polynomial time, T_2 also runs in polynomial time. Therefore, $\bar{A} \in \mathrm{P} = \mathrm{NP}$. So $A \in$ coNP, indicating NP $\subset$ coNP. Meanwhile, for any $A \in$ coNP, we have $\bar{A} \in \mathrm{NP} \subset \mathrm{coNP}$. So $A \in$ NP, which means coNP $\subset$ NP. Therefore, NP = coNP.

Problem 2. For every 2-SAT instance φ of n variables, define graph G_φ of $2n$ vertices as follows. For each variable x_i in φ, G_φ has two vertices labeled by x_i and $\neg x_i$ respectively. There is a directed edge $\ell_i \to \ell_j$ if $(\neg\ell_i) \vee \ell_j$ or $\ell_j \vee (\neg\ell_i)$ is a clause of φ. For notational convenience, for literal $\ell_i = \neg x_{k_i}$, $\neg\ell_i$ is defined to be x_{k_i}. Prove that φ is unsatisfiable if and only if there exist paths from x_j to $\neg x_j$ and from $\neg x_j$ to x_j in G_φ for some j. Use the above fact to show that 2-SAT $\in$ P.

Solution. φ is satisfiable if and only if there exists an assignment of values to variables x_i ensuring all clauses to be true. If there exists a path from x_j to $\neg x_j$, suppose the vertices on the path are $\ell_1, \ell_2, ..., \ell_k$. Then it means that $\neg x_j \vee \ell_1, \neg\ell_1 \vee \ell_2, ..., \neg\ell_k \vee \neg x_j$ are all clauses in φ. Now we attempt to assign values to variables to satisfy all these clauses. If $x_j = 1$, then since $\neg x_j \vee \ell_1 = 1$, $\ell_1 = 1$. Similarly we can deduce that $\ell_2 = 1, ..., \ell_k = 1$. Then $\neg\ell_k = 0$, which contradicts with $\neg\ell_k \vee \neg x_j = 1$. So φ satisfiable $\Rightarrow x_j = 0$. Meanwhile, if there also exists a path from $\neg x_j$ to x_j, we can similarly deduce that $x_j = 1$, which arises contradiction. So the existence of both paths implies φ is unsatisfiable.

If $\forall j$, G_φ does not contain circles including x_j and $\neg x_j$, we will prove that φ is satisfiable. $\forall j$, if there exists a path from x_j to $\neg x_j$, we need to assign x_j to 0 to avoid contradiction. Similarly, if there exists a path from $\neg x_j$ to x_j, we need to assign x_j to 1. For all reachable paths $\ell_i \to \ell_j$ in G_φ, if $\ell_i = 1$ assign $\ell_j = 1$. After assigning all variables we can assign, randomly assign a variable to 0 or 1 and repeat the process. Such assigning process will guarantee all clauses are made true, and will satisfy φ. We will prove that this process will not raise contradiction.

In the first assigning stage, if some ℓ_1, ℓ_2 all need to be assigned 1, they are generated from some vertices that reach there own negatives. Assume $\ell_1 \to y$ and $\ell_2 \to \neg y$, and $\neg\ell_1 \to \ell_1, \neg\ell_2 \to \ell_2$. Then there happens to be $y \to \neg\ell_2$ and $\neg y \to \neg\ell_1$, which makes a circle containing y and $\neg y$. In the second assigning stage, if ℓ is assigned 1 and $\ell \to y$ and $\ell \to \neg y$, then there are also $\neg y \to \neg\ell$ and $y \to \neg\ell$, which makes a path $\ell \to \neg\ell$. However, this indicates that ℓ should be assigned in the first assigning stage.

Therefore, if there are no such circles, φ is satisfiable. So we've proved the equivalence.

Thus, we can consider a graph algorithm for 2-SAT using G_φ. For all variable x, test the reachability of $x \to \neg x$ and $\neg x \to x$. This test can be down by Dijkstra algorithm in polynomial time. If there exists some x that the two path exist simultaneously, then φ is not satisfiable. Otherwise, φ is satisfiable. We need to test n times in total, indicating the whole algorithm is still in polynomial time.

Problem 3. The Lehmer's theorem states that a natural number n is a prime number if and only if the following two conditions hold:

1. There is number a such that $a^{n-1} \equiv 1 \pmod{n}$.
2. For every prime factor q of $n-1$, $a^{(n-1)/q} \not\equiv 1 \pmod{n}$.

Use this theorem to show that PRIME $\in$ NP $\cap$ coNP. (Hint: To prove PRIME $\in$ NP, you may need to use recursively defined witness.)

Solution. First we will prove PRIME $\in$ coNP. For any integer a, we can verify it's not a prime by giving a divider of a, say q, where $q \neq 1$ and $q \neq a$, as proof string. In polynomial time we calculate $a\%q$, if it's zero it means that a has a divider and thus is not a prime.

Then we will prove PRIME $\in$ NP. We will prove that there exists a polynomial $t(n) \geq n^3$, for any prime p, we can verify that p is a prime in $O(t(\log p))$ time. Use induction on p. For $p = 2$, it's obviously true. Assume it's true for all primes less Than p, we will prove it's true for p.

We construct the verifier using the given property of prime numbers. The proof string contains the number a and the prime factor decomposition of $p-1$: $p - 1 = \prod_i q_i^{\alpha_i}$. The verifier will do the following:

1. Verify $a^{p-1} \equiv 1 \pmod{p}$. Using the fast-exponentiation algorithm this can be done in $O(\log p)$ time.

2. For the given prime factor decomposition of $p-1$, verify that it is indeed the correct decomposition. As $\prod_i q_i^{\alpha_i} = p - 1$, we have $\sum_i \log(q_i) \leq \log(p-1)$.

    (a) By the induction hypothesis, we can verify q_i indeed are primes in $O(t(\log q_i))$ time. Verifying all q_i can be done in $O(\sum_i t(\log q_i)) \leq O(t(\sum_i \log q_i)) \leq O(t(\log(p-1))) \leq O(t(\log p))$, where the second inequality holds because $(\sum a_i)^m \leq \sum a_i^m$ for positive a_i and m.

    (b) Verify the equality $\prod_i q_i^{\alpha_i} = p - 1$ holds. This can be done in at most $\sum_i (\log(p))^3 \leq (\log(p))^3$, since there are at most $\log p$ prime factors.

3. For all the given q_i, verify that $a^{(p-1)/q_i} \not\equiv 1 \pmod{p}$. Each can be done in $O(\log p)$ time, and since there are at most $\log p$ prime factors we can finish the verification in $O((\log p)^2)$ time.

Therefore, the total verification will cost at most $O(t(\log p) + (\log p)^3) = O(t(\log p))$ time, as $t(n) \geq n^3$, which completes the inductive step's proof,

By induction, we know that for all p prime, we can verify it in polynomial time, indicating PRIME $\in$ NP. So PRIME $\in$ NP $\cap$ coNP.