1 Problem 34.2-3

Show that if HAM-CYCLE $\in$ P, then the problem of listing the vertices of a hamiltonian cycle, in order, is polynomial-time solvable.

**Solution:** Assume that HAM-CYCLE $\in$ P, then we can find the hamiltonian cycle in such a way:

1. for all node $v \in V$
2. let $E_v$ be the edges incident to $v$.
3. for each pair $\{e_1, e_2\} \in E_v$
4. construct graph $G' = (V, (E - E_v) \cup \{e_1, e_2\})$
5. if $G'$ contains a hamiltonian cycle
6. record the pair of $\{e_1, e_2\}$
7. break
8. if no pair in $E_v$ is recorded
9. return false
10. print(all pair of $\{e_1, e_2\}$)
11. return true

Because line 5 only takes polinomial time to determine if there is a hamiltonian cycle, this algorithm is also polynomial-time solvable.

2 Problem 34.2-10

Prove that if $NP \neq co-NP$, then $P \neq NP$.

**Solution:** Prove it by its contrapositive: if $NP = P$, then $NP = co-NP$. Assume $NP = P$, given $L \in co-NP, \bar{L} \in NP = P$. Because $\bar{L} \in P$ and the languages in $P$ are closed under complement, so $\bar{L} = L \in P$. Therefore $co-NP = P = NP$.

3 Problem 34.3-2

Show that the $\leq_P$ relation is a transitive relation on languages. That is to show if $L_1 \leq_P L_2$, and $L_2 \leq_P L_3$, then $L_1 \leq_P L_3$

**Solution:**
Proof. Let $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$, i.e. there exist polynomial-time computable reduction functions $f_1 : \{0,1\}^* \rightarrow \{0,1\}^*$ and $f_2 : \{0,1\}^* \rightarrow \{0,1\}^*$ such that

\[ x \in L_1 \iff f_1(x) \in L_2 \]

\[ x \in L_2 \iff f_2(x) \in L_3 \]

Define $f_3 = f_1 \circ f_1$, then $L_3$ is a polynomial-time computable function: $\{0,1\}^* \rightarrow \{0,1\}^*$, and

\[ x \in L_1 \iff f_3(x) \in L_3 \]

holds. Hence $L_1 \leq_P L_3$.

4 Problem 34.4-7

Let 2-CNF-SAT be the set of satisfiable boolean formulas in CNF with exactly 2 literals per clause. Show that 2-CNF-SAT $\in$ P. Make your algorithm as efficient as possible.

Solution: Considering a 2-CNF formula $\Psi$ with $n$ variables and $m$ clauses, we need to show that 2-SAT is polynomial-time decidable by constructing a graph and using path searches in the graph. Create a graph $G(V,E)$ with $2n$ vertices. Each vertex represents a true or not true literal for each variable in $\Psi$.

For any clause $(a \lor b)$ in $\Psi$, create a directed edge from $\neg a$ to $b$ and from $\neg b$ to $a$.

For any clause $(\neg a \lor b)$ in $\Psi$, create a directed edge from $a$ to $b$ and from $\neg b$ to $\neg a$.

For any clause $(\neg a \lor \neg b)$ in $\Psi$, create a directed edge from $a$ to $\neg b$ and from $b$ to $\neg a$.

A 2-CNF formula $\Psi$ is unsatisfiable iff there exists a variable $x$, such that:

1. there is a path from $x$ to $\neg x$ in the graph
2. there is a path from $\neg x$ to $x$ in the graph

Check if such path exists by a graph traversal algorithms like DFS. It takes polynomial time of $O(V+E)$. 

2