Due: Thursday, April 20 by 5:00pm - (Old date was Wednesday, April 19)

Reading:
Chapter 7 (Time Complexity), section 7.1-7.3 pages 247-270

Problems:

1. The vertex cover problem (VC) is defined below.

Show that the VC problem is in NP.

Vertex Cover Problem:
A vertex cover C of a graph G is a set of vertices of G where every edge of G contains at least one vertex from C. The size of C is just the number of vertices in the set C.

VC = \{(G,k) | G is a graph and k is an integer and G has a vertex cover of size at most k\}

First state what you need to do to show VC is in NP. Then show it.

Answer: We need to exhibit a polynomial time verifier V (i.e. an algorithm) which uses a witness w together with a possible element of VC to verify if the VC instance is true or false.

Specifically for the VC problem, V takes as input a possible element of VC, (G,k) and a string w, a possible witness to the fact that that (G,k) has a vertex cover of size at most k.

V then checks if w is a list of \(\leq k\) vertices and if that set of vertices w in fact covers every edge in G. If this is true then V accepts ((G,k),w). If not, it rejects.

2. Show that NP is closed under intersection.

Answer: Let \(L_1, L_2 \in NP\), and \(M_1, M_2\) NTMs that decide \(L_1, L_2\) in polynomial time. We construct NTM \(M\) that decides \(L_1 \cap L_2\) in polynomial time.

\(M = \)"on input w:

1. Run \(M_1\) on w, since \(M_1\) is a NTM, this will cause the execution to branch out into a tree as \(M_1\) would.
2. For every branch of the execution where \(M_1\) rejects, reject.
3. For every branch of the execution where \(M_1\) accepts, Run \(M_2\) on w.
4. For every branch of the execution where \( M_2 \) accepts, accept. For branches where \( M_2 \) rejects, reject.”

The execution of \( M \) will be a large tree, the top part of the tree will match the execution tree of \( M_1 \). Computation paths (branches of execution) were \( M_1 \) accepts will be extended with sub-trees matching the execution tree of \( M_2 \). The longest possible computation path \( M \) can have is made out of the longest possible path of \( M_1 \) extended with the longest possible path of \( M_2 \). Since \( M_1 \) and \( M_2 \) run in polynomial time, the length of the longest paths in their execution is polynomial in the input size. The sum of two polynomials is a polynomial, therefore, \( M \) runs in polynomial time.

\( M \) accepts if and only if both \( M_1 \) and \( M_2 \) had at least one accepting computation path. If either \( M_1 \) or \( M_2 \) had only rejecting paths, then \( M \) will reject as well. Therefore \( M \) decides \( L_1 \cap L_2 \).

3. This problem concerns the question of whether NP is closed under complement, meaning that if \( L \) is in NP is it also true that \( \overline{L} \) = the complement of \( L \) is in NP.

In class we showed that P is closed under complement. This was done by noting that Time \((n^k)\) is closed under complement for any integer \( k \). And this followed from the fact that if we simple reverse accept and reject for the output of a Time\((n^k)\) algorithm we get a Time\((n^k)\) for \( \overline{L} \) = complement of \( L \).

Here is the same attempt to show that NP is closed under complement:

We show that NTime \((n^k)\) is closed under complement for any integer \( k \). This follows from the fact that if we simple reverse accept and reject for the output of a NTime\((n^k)\) algorithm we get a NTime\((n^k)\) for the complement of \( L \).

This argument, which works for P and Time\((n^k)\), does not work for NTime \((n^k)\). Explain why.

**Answer:** Let \( N \) be a NTM that runs in NTime\((n^k)\) and accepts a language \( L \). If we reverse the accepting and rejecting states in \( N \) then this does not always result in the complement of \( L \) being the language accepted.

That is, let \( N \) be a NTM that runs in NP-time and accepts a language \( L \).

What strings are accepted by an NTM \( R \) where \( R \)’s program duplicates \( N \)’s except that \( R \) has \( N \)’s accepting and rejecting states switched in its program?

Consider some input string \( x \) which is accepted by \( N \). Then \( R(x) \) might either be accepted or rejected. (it would be rejected only if EVERY computation path of \( N(x) \) accepted).

Now consider some input \( y \) which \( N \) rejected. In this case \( R(y) \) accepts, it cannot reject since it halts in the accepting state on the computation path or paths on which \( N(y) \) rejects.
So, in general the set of string accepted by R includes all string that N rejected and also some that N accepted. That is \( L \) complement \( \subset \{ s \mid s \text{ is a string accepted by NTM } R \} \), and is generally not equal to it. And also \( \{ s \mid s \text{ is a string rejected by NTM } R \} \subset L \), but not always equal to it.

4. Prove that \( \leq^p m \) is transitive.
   This means: Prove that if \( A \leq^p m B \) and \( B \leq^p m C \) then \( A \leq^p m C \).

   **Answer:** From \( A \leq^p m B \) we know that there is a polynomial time function \( f \) where \( x \in A \) if and only if \( f(x) \in B \).

   From \( B \leq^p m C \) we know that there is a polynomial time function \( g \) where \( x \in B \) if and only if \( g(x) \in C \).

   We need to show: \( A \leq^p m C \), that is we show that there is a polynomial time function \( h \) where \( x \in A \) if and only if \( h(x) \in C \).

   We define \( h \) as \( h(x) = g(f(x)) \). From this definition we have \( x \in A \) if and only if \( f(x) \in B \) if and only if \( g(f(x)) \in C \). Finally as we saw in the last homework problem in HW 4, \( h(x) = g(f(x)) \) is polynomial time computable since \( f \) and \( g \) are.

5. (Question of whether NP is closed under the relation \( \leq^p m \))
   Assume that \( A \leq^p m B \) and \( B \) is in NP. Is it true that then \( A \) is in NP?

   If true, give a proof, if false, say why it is false.

   **Answer:** This is true. Briefly the proof is: \( B \) has a verifier \( V \) since \( B \) is in NP.

   Let \( f \) be the function reducing \( A \) to \( B \). To prove \( A \) is in NP we need to find a verifier \( V' \) for \( A \).

   Here is how the algorithm: \( V' \) runs on an inputs \((x,w)\), where \( x \) is a possible element of \( A \) and \( w \) is a possible witness for \( V \), the verifier for \( B \).

   \( V'(x,w) \) computes \( f(x) \) and then runs \( V \) on input \((f(x),w)\).

   If \( V(f(x),w) \) accepts then \( V' \) accepts \((x,w)\), else if rejects.

   Clearly \( V' \) runs in \( P \) and it is straightforward to check that \( V' \) is a verifier for \( A \), so \( A \) is in NP.
Alternate Answer: Another way to prove this is to come up with some NTM that decides $A$ in polynomial time. We will also have to use the function reducing $A$ to $B$.

Since $B \in NP$, then there exists NTM $M_B$ that decides $B$ in polynomial time. Since $A \leq^P_m B$, then there must exist a function $f$ such that $w \in A \equiv f(w) \in B$ and $f$ can be computed in polynomial time. (check definition of poly-time reduction in the book).

We will construct NTM $M_A$ that decides $A$ in polynomial time as follows:\
$M_A = “$ on input $w:"
1. compute $g = f(w)$ (this can be carried out in polynomial time).
2. Run NTM $M_B$ on $w$, for every computation path of $M_B$ that accept, accept. For other computation paths, reject.”

$M_A$ accepts $w$ if there exists one computation path of $M_B$ that accepts $f(w)$. In other words, $M_A$ accepts $w$ if and only if $M_B$ accepts $f(w)$. Since $L(M_B) = B$, then $M_A$ accepts $w$ if and only if $f(w) \in B$, which is the case if and only if $w \in A$ (by the definition of reduction). Therefore $L(M_A) = A$.

$M_A$ runs in polynomial time, since both steps in its description can run in polynomial time, and adding to polynomials results in a polynomial.

6. Show that the problem max-cut, as defined in problem 7.25 on page 296 is in NP.

First state what you need to do to show this. Then show it.

Answer: To show it is in NP we need to define a verifier $V$ for max-cut.

This verifier $V$ takes as input $((G,k), w)$ where $(G,k)$ is an instant of max-cut (a graph $G = (V, E)$ and a number $k$) and $w$, the witness, is a set of vertices forming the cut for $G$ (which may have size at least $k$).

$V$ checks if $w$ forms a cut in $G$ whose size is bigger than or equal to $k$, if so it accepts, else it rejects. Hence $V$ is a verifier for max-cut and it is not difficult to check it is computable in polynomial time.

One way to check this property is to check that for every edge $e = (v_1, v_2)$ in the cut $w$, $v_1$ and $v_2$ are two unconnected nodes in the graph $(V, E - w)$. To perform this check, we first need to remove every edge in the cut $w$ from the graph (can clearly be done in polynomial time), then we need to run a graph traversal algorithm (DFS or BFS) starting from $v_1$ and check that $v_2$ is not reached, $w$ can have at most $|E|$ edges, and the DFS/BFS need $O(|V| + |E|)$, so this can also be carried out in polynomial time ($O(|E| \times (|V| + |E|))$).