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.

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

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:** The proof of this fact is very similar to that of closure of P under intersection which
is proved in the previous HW. It is omitted here.

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 $\text{NTIME}(n^k)$ 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 $\text{NTIME}(n^k)$ algorithm we get a $\text{NTIME}(n^k)$ for the complement of $L$.

This argument, which works for $P$ and $\text{Time}(n^k)$, does not work for $\text{NTIME}(n^k)$. Explain why.

Answer: Let $N$ be a NTM that runs in $\text{NTIME}(n^k)$ 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 $\text{NP}$-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 $\subset \{ s \mid s$ is a string accepted by NTM $R \}$, and is generally not equal to it. And also $\{ s \mid s$ is a string rejected by NTM $R \}$ $\subset L$, but not always equal to it.

4. Prove that $\leq_P$ is transitive.
This means: Prove that if $A \leq_P B$ and $B \leq_P C$ then $A \leq_P C$.

Answer: From $A \leq_P B$ we know that there is a polynomial time function $f$ where $x \in A$ if and only if $f(x) \in B$.

From $B \leq_P C$ 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: $A \leq_P C$, 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 $\leq_P$)

Assume that $A \leq_P B$ 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.
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 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.