PROBLEMS

Part 1: 4 of the following 5 problems will be graded. You can choose which one to leave out. There are 40 points total. (10 points each).

1. Consider the independent set (IS) problem which we discussed in class and appeared in HW 5.
   Show that if $P=NP$ then there is an algorithm $A$ running in polynomial time which takes as input a graph $G$ and which outputs a largest independent set $I$ in $G$. (Largest means having the most pairwise independent vertices possible, and it may not be unique.)
   Hint: Take a look at the solution to problem 7.40 on page 327 in the text.
   Answer: If you assume that $P = NP$, then $IS \in P$, and you can test whether $G$ contains an independent set of size $k$ in polynomial time, for any value of $k$. By testing whether $G$ contains an independent set of each size, from 1 to the number of nodes in $G$, you can determine the size $t$ of a maximum independent set in $G$ in polynomial time. Once you know $t$, you can find an independent set with $t$ nodes as follows. For each node $x$ of $G$, remove $x$ and calculate the resulting maximum independent set size. If the resulting size decreases, replace $x$ and continue with the next node. If the resulting size is still $t$, keep $x$ permanently removed and continue with the next node. When you have considered all nodes in this way, the remaining nodes are an independent set of size $t$.

2. You may assume that $A$ and $B$ below are both sets of Boolean strings.
   (i). Prove that $SPACE (n^3)$ is closed under difference. That is, if $A$ and $B$ are both in $SPACE (n^3)$ then so is $A-B$.
   (ii). Prove that $SPACE (n^3)$ is closed under complement. That is, if $A$ is in $SPACE (n^3)$ then so is the complement of $A$. You can use part (i) of this problem if you like.
   Answer:
   i. $A$ and $B$ are both in $SPACE(n^3)$, so let $M_A$ decide $A$ in $O(n^3)$ space and $M_B$ decide $B$ in $O(n^3)$ space.
   $M_{A-B} =$“On input $x$:
   1. Run $M_A$ on $x$. If $M_A$ rejects, reject.
   2. Run $M_B$ on $x$. If $M_B$ rejects, accept. Otherwise, reject.”

   $M_{A-B}$ decides $A - B$ since if $x \in A - B$ then $x \in A$ and $x \notin B$, so $M_A$ accepts $x$ and $M_B$ rejects $x$. Otherwise, $x \notin A - B$ and $M_{A-B}$ rejects $x$.  

1
\(M_{A-B}\) requires the amount of space that \(M_A\) requires plus the amount of space that \(M_B\) requires or really just the maximum of the two, since after running \(M_A\) everything except \(x\) can be deleted from the tape and \(M_B\) can reuse the same space. Since \(M_A\) and \(M_B\) are in \(\text{SPACE}(n^3)\), \(M_{A-B}\) is as well.

ii. \(\overline{A} = \Sigma^* - A\) so if \(\Sigma^*\) and \(A\) are in \(\text{SPACE}(n^3)\), then so is \(\overline{A}\) by i. \(\Sigma^*\) is decided by the machine which ignores its input and immediately accepts. This machine requires no space on the work tape, so \(\Sigma^* \in \text{SPACE}(n^3)\). Therefore, \(\overline{A}\) is also in \(\text{SPACE}(n^3)\).

3. Explain briefly why the clique problem is in \(\text{PSPACE}\).

What I am asking for is a description of how you can solve an instance of the clique problem using only an amount of space (memory) which is some polynomial in the size of the instance.

Answer:
The brute force algorithm for clique shows that clique is in \(\text{PSPACE}\). On input \((G, k)\), check whether any subset of \(V\) is a clique of size \(k\) in \(G\). This algorithm only requires enough space on the work tape to store one subset \(C\) at a time. After checking whether \(C\) is a clique of size \(k\), \(C\) can be deleted from the tape and the same cells can be reused to store the next subset of \(V\). \(C\) has at most \(n\) vertices in it, so \(C\) requires \(O(n)\) tape cells to store. Thus, clique is in \(\text{PSPACE}\).

4. Prove that \(\text{NP}\) is a subset of \(\text{PSPACE}\).

Answer: In class and in problem 2 above we discussed why \(\text{SAT}\) and also clique (both \(\text{NP}\)-complete problems) are in polynomial space (\(\text{PSPACE}\)). This problem is similar to those but now you have to show that an arbitrary \(\text{NP}\) language is in \(\text{PSPACE}\). Still the idea is the same.

We take an \(\text{NP}\) language \(K\) and we describe an algorithm to decide \(K\) which uses only polynomial much space.

Since \(K\) is in \(\text{NP}\) what we know about \(K\) is that it has a polynomial time verifier \(V(w,c)\).

The desired algorithm now simulates the computations of \(V(w,c)\) for all possible certificates \(c\). Each of these computations, being in \(\text{P}\) (poly time) can be carried out in \(\text{PSPACE}\). And we can simple carry out these computations on every different \(c\)’s. If we find an \(c\) where \(V(w,c)\) accepts then we know \(w\) is in \(K\) and we accept. Otherwise for all possible \(c\) \(V(w,c)\) rejects then we know that \(w\) is not in \(K\), and our algorithm rejects.

5. The definition of \(\text{PSPACE}\)-complete is very similar to \(\text{NP}\)-complete. It can be found on page 337 of the textbook. And an example of such a problem can be found on page 339.

Prove that if \(C\) is a \(\text{PSPACE}\)-complete language and \(C \in \text{NP}\) then \(\text{NP} = \text{PSPACE}\).

Answer: Proving any two sets to be equal requires showing that each is a subset of the other one.

By problem 4 we know \(\text{NP} \subseteq \text{PSPACE}\). So left to prove is that \(\text{PSPACE} \subseteq \text{NP}\).

We are given a language \(C\) which is \(\text{PSPACE}\)-complete and \(C \in \text{NP}\).

Now starting with any language \(L\) in \(\text{PSPACE}\) we prove it is also in \(\text{NP}\).
Let \( L \) be any a language in \( \text{PSPACE} \). Then \( L \leq^P_m C \) by \( C \)'s \( \text{PSPACE} \)-completeness. We also have \( C \in \text{NP} \).

and so \( L \) is in \( \text{NP} \) by problem 3(ii) of Hw 5. Hence \( \text{PSPACE} \subseteq \text{NP} \) as needed.

Part 2: These problems are good practice and you should try them. They will not be graded.

1. Give an example of a bin packing algorithm which always finds a packing which uses no more than twice the optimal number of bins in its packings.
   (This is called a 2-approximation algorithm for bin packing.)
   Maybe you can show that the first fit bin packing algorithm gives a 2-approximation ?

2. Give an example of a propositional formula with exactly three variables which is true for every possible truth assignment. (Such a formula is called a tautology.)

3. Show that the intersection of a \( \text{P} \) problem and an \( \text{NP} \) problem is in \( \text{NP} \).

4. Do you think that \( \text{NP} \) is closed under complement ? That is, if a language \( L \) is in \( \text{NP} \) then it’s complement must also be in \( \text{NP} \) ?
   I’m not looking for a proof here just some intuition about the problem, for example explaining why the “usual” reason that sets like \( \text{P} \) are closed under complement does not work for \( \text{NP} \).

5. Page 323, problem 7.12