Solution Set for CPSC 468/568 Exam 1

Question 1

(a) No, A is decidable. To decide whether \((\alpha, x)\) is in A, simulate \(M_\alpha\) on input \(x\) for \(t=|x|^3\) steps, output 1 if the simulation has halted by the \(t^{th}\) step, and output 0 otherwise.

(b) Turing Machine \(M\) is oblivious if, for every input \(x\in\{0,1\}^*\) and \(i\in\mathbb{N}\), the location of each of \(M\)'s heads at the \(i^{th}\) step of execution on input \(x\) is a function only of \(|x|\) and \(i\).

(c) <This is Theorem 4 in [BGS75]. The proof is very similar to the proof given in class and in the book that there is an \(O\) relative to which \(P\) is not equal to \(NP\). I am still thinking about how to write a version of the answer that makes clear why this proof is very similar to but not exactly the same as the one in the book.>

Question 2

(a) VC is clearly in NP, because one can guess a subset \(V'\) of \(V(G)\) and verify in polynomial time that it is of size at most \(k\) and is a vertex cover. Furthermore, the mapping from \((G, k)\) to \((G, |V(G)| - k)\) is a reduction from IND-SET to VC, because \(W\) is an independent set of \(G\) if and only if \(V(G) - W\) is a vertex cover of \(G\).

(b) SUBGRAPH-ISO is clearly in NP, because one can guess a subset \(V_3\) of \(V_1\), a subset \(E_3\) of \(E_1\), and a mapping \(f\) from \(V_2\) to \(V_3\) and then verify in polynomial time that \(f\) is an isomorphism from \(H\) to \((V_3, E_3)\). Furthermore, there is obviously a polynomial-time reduction from IND-SET to SUGRAPH-ISO, because IND-SET is a special case of SUBGRAPH-ISO in which \(H\) is a graph with \(k\) vertices and no edges. Thus SUBGRAPH-ISO is at least as hard as any set in NP.

(c) DOMINATING SET is clearly in NP, because one can guess a subset \(V'\) of \(V(G)\) and verify in polynomial time that it is of size at most \(k\) and is a dominating set. The following polynomial-time mapping is a reduction of VC to DOMINATING SET. Let \(((V, E), k)\) be an instance of VC; the corresponding instance of DOMINATING SET is \(((U, E'), k)\). The set \(U\) contains a vertex \(v'\) for each \(v\) in \(V\) that is the endpoint of at least one edge in \(E\) and a vertex \(v_e\) for each \(e\) in \(E\). The set \(E'\) contains an edge \((u', v')\) for every pair of nodes \(u\) and \(v\) that correspond to nodes \(u\) and \(v\) in \(V\) such that \((u, v)\) is in \(E\) and edges \((v_e, x)\) and \((v_e, w)\) such that \(e = (x, w) \in E\).

Let \(V'\) be a vertex cover of \((V, E)\), and let \(V''\) be the set of nodes in \(U\) that correspond to vertices in \(V'\). We show that \(V''\) is a dominating set of \((U, E')\). Consider a vertex \(v\) in \(U - V''\). If \(v\) corresponds to a node in \(V\), then, by construction, it is not an isolated vertex and, by definition of \(E'\) and of “vertex cover,” must be adjacent to at least one vertex in \(V''\). If \(v\) corresponds to an edge \((x, w)\) in \(E\), then, again by definition of \(E'\) and “vertex cover,” it must be adjacent to a node in \(V''\) that corresponds to \(x\) or \(w\).

Let \(V'''\) be a dominating set of \((U, E')\). We show how to map it to a set \(V'\) that is no bigger and that is a vertex cover of \((V, E)\). If \(v' \in V'''\) corresponds to a vertex \(v\) in \(V\), then put \(v\) into \(V'\). If \(v_e \in V'''\) corresponds to an edge \(e = (x, w)\) in \(E\), then put \(x\) into \(V'\). Note
that $|V'| \leq |V''|$, because at most one vertex is put into $V'$ for each vertex in $V''$. Let $(y, z)$ be an edge in $E$. If $y'$ or $z'$ is in $V''$, then $y$ or $z$ is in $V'$, and $(y, z)$ is covered. If neither $y'$ nor $z'$ is in $V''$, then $y$ was put into $V'$, and once again $(y, z)$ is covered. In either case, $V'$ is a vertex cover of $(V, E)$.

**Question 3**

(a) Implicitly logspace-computable reductions are defined in Definition 4.14. To study NL, we want to look at logspace reductions; in particular, we want to look at logspace reductions from various languages in NL to the NL-complete language PATH. For such a reduction to make sense, it must be able to map a string of length $n$ to a PATH instance of size $p(n)$, for some polynomial $p$, but a deterministic-logspace machine cannot even write down such a target instance if $p(n) = o(\log n)$. An implicitly logspace-computable reduction, however, can produce any bit of the desired target instance, and that suffices for the study of NL.

(b) Read-once certificates are defined in Subsection 4.4.1. Recall that the sequence of choices made by an NP machine $M$ during an accepting computation can be regarded as a polynomial-length certificate $u$ that the input $x$ is in $L(M)$; a deterministic machine can verify the pair $(x, u)$ in time polynomial in $|x|$. We would like to say something similar about NL. However, the sequence $u$ of choices made during an accepting computation of a nondeterministic-logspace machine on input $x$ will, in general, be of length polynomial in $|x|$; hence a deterministic-logspace verifier could not even store the pair $(x, u)$ if it is restricted to space logarithmic in $|x|$. Thus, we require that the verifier access the certification $u$ in a read-once fashion.

(c) Let $M$ be a nondeterministic, $s(n)$-space-bounded Turing Machine and $x \in \{0, 1\}^n$ be an input. Consider the configuration graph $G_{M,x}$ (as defined in Section 4.1). Because $M$ uses space $O(s(n))$ on input $x$, it can enter $2^{O(s(n))}$ distinct configurations; thus, the total number of vertices in $G_{M,x}$ is $2^{O(s(n))}$. Furthermore, each vertex can be represented by a string of length $O(s(n))$, and it has outdegree at most two, because we can assume without loss of generality that the nondeterministic choices of $M$ are binary. Thus, $G_{M,x}$ can be explicitly constructed in its entirety in deterministic time $2^{O(s(n))}$. Using a standard linear-time algorithm for reachability in a directed graph, one can determine whether there is a directed path in $G_{M,x}$ from the start configuration to the accept configuration. This entire process takes time $2^{O(s(n))}$ and gives a correct answer regardless of whether $M$ accepts or rejects $x$. Therefore, both $L(M)$ and its complement can be decided in $\text{DTIME}(2^{O(s(n))})$.

**Question 4**

(a) False. P/poly contains undecidable sets, but NP does not. What is unknown is whether NP is contained in P/poly; by the Karp-Lipton theorem, we know that, if indeed NP is contained in P/poly, then the PH collapses at the second level.

(b) True, by Savitch’s Theorem.
(c) Unknown. The proof of Savitch’s Theorem tells us that \( \text{NSPACE}(n) \) is contained in \( \text{DSPACE}(n^2) \), but we do not know whether this bound is tight.

(d) True, by Corollary 4.19 of the Immerman-Szlepcsenyi theorem.

(e) Unknown. It is unknown whether \( \text{PSPACE} \) is contained in \( \text{PH} \), and it is unknown whether \( \text{PH} \) has a complete set; as shown in HW3, if \( \text{PH} \) has a complete set, then it collapses at the \( i \)th level, for some \( i \).

**Question 5**

(a) A directed path, starting at \( s \), with an odd number of nodes.

(b) A directed path, starting at \( s \), with an even number of nodes.

(c) Let \( (G, s) \) be a GEOGRAPHY instance of size \( n \). Note that \( |V(G)| = N < n \). Consider the function \( f(i, x, X) \), where \( i \) is either I or II, the node \( x \) is the one just chosen by player I, and \( X \) is the set of nodes that have already been chosen; then, for all \( x \),

\[
\begin{align*}
&f(I, x, V(G)) = \text{YES} \quad \text{and} \quad f(II, x, V(G)) = \text{NO}. \\
&(\text{If one gets to the point at which I (resp. II) moved last (by choosing } x \text{) and all nodes have been chosen, then the original input is a yes (resp. no) instance of GEOGRAPHY.})
\end{align*}
\]

If \( X \neq V(G) \), then \( f \) is defined recursively:

\[
\begin{align*}
&f(I, x, X) = \text{YES if and only if, for all } y \text{ such that } (x, y) \in E(G) \text{ and } y \text{ not in } X, \\
&f(II, y, X \cup \{y\}) = \text{YES}; \\
&f(II, x, X) = \text{YES if and only if, for at least one } y \text{ such that } (x, y) \in E(G) \text{ and } y \text{ not in } X, \\
&f(I, y, X \cup \{y\}) = \text{YES}.
\end{align*}
\]

We can decide whether \( (G, s) \) is a yes instance of GEOGRAPHY by computing \( f(I, s, \{s\}) \). The space required to compute \( f(i, x, X) \) is upper bounded by \( N \) (one bit for each answer returned by a recursive call) plus the space required for one recursive call – space used for the first recursive call can be reused for the second, third, etc. Specifying the input to the recursive call requires space at most \( 2N \), and the depth of the recursion is at most \( N \). The overall space complexity is thus at most \( 3N^2 = O(n^2) \).

**Question 6**

(a) A language \( L \) is in DP if it is of the form \( L_1 \cap L_2 \), where \( L_1 \in \text{NP} \) and \( L_2 \in \text{coNP} \).

(b) (ii) is known. Any language \( L \) in \( \text{NP} \) or \( \text{coNP} \) is clearly in DP, because \( L = L \cap \{0,1\}^* \), and \( \{0,1\}^* \) is in both \( \text{NP} \) and \( \text{coNP} \).

(c) AND and OR gates in \( \text{NC}^i \) circuits have fan-in two, whereas AND and OR gates in \( \text{AC}^i \) circuits have unbounded fan-in.

(d) For each \( x \in \{0,1\}^* \), the string \( 1x \) can be interpreted as an integer; furthermore, the mapping from strings to integers that maps \( x \) to \( 1x \) is injective. Let \( L \) be a language in \( \text{NEXP} \). Then the unary language \( L' = \{1^{|x|} : x \in L\} \) is in \( \text{NP} \). If \( M \) is a deterministic machine that verifies membership of \( x \) in \( L \), then \( M \) can also be used to verify membership of \( 1^{|x|} \) in \( L' \); furthermore, if \( M \) runs in time exponential in \( |x| \), then it runs in time polynomial in \( |1^{|x|}| \). Now, because \( L' \) is a unary language in \( \text{NP} \), by hypothesis it is...
in P. To decide whether $x$ is in $L$, run a deterministic polynomial-time machine that
decides membership of $1^{1x}$ in $L'$; the running time of this machine is exponential in $|x|$. 