MidTerm (with Solution)

THIS IS AN OPEN BOOK, ONE-HOUR IN-CLASS EXAM. READ THE QUESTIONS CAREFULLY. WE STRONGLY SUGGEST REWRITING YOUR FINAL ANSWER – IT ALWAYS GET BETTER THE NEXT TIME! PLEASE WRITE IN COMPLETE ENGLISH SENTENCES. GOOD LUCK!

Problem 1 (Short Questions) [10 points each]
State whether each of the following statements is TRUE, FALSE or UNKNOWN. You must provide a brief (say, two or three lines) of justification.

(A) There exists a language $A \subseteq \Sigma^*$ such that $len(A) \notin NP^A$. Recall that $len(A) = \{x \in \Sigma^* : |x| = |w| \text{ for some } w \in A\}$.
SOLUTION: FALSE. For any A, the language $len(A) \in NP^A$ because on input x, we can guess the string w of length equal to $|x|$ and ask the oracle if $w \in A$.
PITFALL: some of you said that we can modify the proof (in homework) which shows that there is an A such that $len(A) \notin P^A$. This is a very understandable error to make!

(B) There exist languages A, B such that $P^A = NP^A$ and $P^B \neq NP^B$.
SOLUTION: TRUE. If A is $PSPACE$-complete, a homework problem shows $P^A = NP^A$. Another homework problem shows that there is a B such that $len(B) \notin P^B$, and hence $P^B \neq NP^B$.

(C) $NTIME(2^n) = NTIME_r(2^n)$.
SOLUTION: TRUE. 2^n is time constructable, so we can "clock" any machine to reject if the machine does not accept within 2^n steps.

(D) The characteristic class $DTIME(n^6)$ has an efficient universal acceptor.
SOLUTION: UNKNOWN. To use the usual trick of simulating all Turing machines that runs in time n^6, the best result known says we need $O(n^6 \log n)$ time (this is the Hennie-Stearns tape reduction result). But we need to improve this to $O(n^6)$ to get the result stated.
FOOD FOR THOUGHT: how come P has an efficient universal acceptor?

(E) The characteristic class IP has an efficient universal acceptor.
SOLUTION: UNKNOWN. We do not know how to simulate all and only those polynomial time $\{\oplus, \vee\}$-machines that have "all bounded-error".
PITFALL: many students ansy TRUE because $IP = PSPACE$ and since $PSPACE$ has an efficient universal acceptor. The problem is that IP, as a characteristic class, is distinct from the characteristic class $PSPACE$.

Problem 2 (Separation Techniques) [20 points]

Recall $DEXPTIME = DTIME(2^{n^{O(1)}})$ and $NEXPTIME = NTIME(2^{n^{O(1)}})$. Show that if $P = NP$ then $DEXPTIME = NEXPTIME$.

HINT: this is an upward translation result.

SOLUTION: It is enough to show that $NEXPTIME \subseteq DEXPTIME$. Let L_0 be accepted by some machine N in nondeterminisitic time 2^{n^k} (for some constant $k \geq$

1). Consider the language $L_1 = \{x\#^m : x \in L_0 : m = 2^{|x|^k}\}$. Hence $L_1 \in NP$ since we can recognize if $x\#^m \in L_1$ by simulating N on input x. The time taken is $2^{|x|^k} \le |x\#^m|$ (which is actually linear time). Since $P = NP$, there is some deterministic machine M that accept L_1 in polynomial n^ℓ for some $\ell \ge 1$. Consider the following deterministic machine M' for recognizing L_0: on input x of length n, we first compute $w = x\#^m$ ($m = 2^{n^k}$) and then use M to decide if $w \in L_1$. M takes time $|w|^\ell = (n + 2^{n^k})^\ell = O(2^{n^{k+1}})$. We accept iff M accepts. It is easy to see that L(M')$= L_0$ and M' runs in time $O(2^{n^{k+1}})$. Hence $L_0 \in DEXPTIME$.

Problem 3 (Alternation) [20 points]

Let $G = (V, E)$ be any directed acyclic graph with goal node $i_0 \in V$. If T is a pebbling tree for (G, i_0) with pebbling time t, show that we can pebble (G, i_0) using $1 + t$ pebbles.

SOLUTION: We need some preliminary concepts:

- Recall the terminology in our lecture notes: since we are simultaneously discussing two graphs T and G, we shall call the nodes of T "vertices", so as not to confuse them with the "nodes" of G.
- Recall that if (u, v) is an edge of G, we call u a *predecessor* of v. A source node is one without any predecessor. Also, call u an *ancestor* of v if there is a path in G from u to v. Thus the ancestor relation is the reflexive transitive closure of the predecessor relation. We will need a simple technical lemma:
- Suppose $W \subseteq V$ is a set of nodes in $G = (V, E)$. To "pebble (G, i_0) relative to W" (or "pebble $(G, i_0; W)$" means to play the pebble game with i_0 as goal as usual, except that we assume that the nodes in W already have pebbles which are never removed. Moreover these pebbles in W are free – we are not charged for them when counting the number of pebbles used.
- Suppose T is a pebbling tree for (G, i_0). Let A_0 be the set of ancestors of of i_0. Suppose T' is obtained from T as follows: at each vertex u of T, we replace the pebbling set $X(u)$ at u by $X(u) \cap A_0$. CLAIM 1: T' is also a pebbling tree for (G, i_0).
  The proof of CLAIM 1 is left as an easy exercise. Call T' the "ancestor restricted" version of T.

We now prove the main result by induction on the number of vertices in T. In view of CLAIM 1 we may assume that the ancestor-restricted version of T is T itself. Let u be the root of T with label $[i_0, X, Y]$.
BASIS: suppose T has no other vertices. Then i_0 is a source node in G and by definition, we can pebble i_0 using 1 pebble. But $1 \le 1 + t$.
INDUCTION: let $|X| = k$ and the subtrees rooted at the children of u are $T_1, \ldots, T_k$. Let the challenge node at the root of T_j be x_j. Let $time(T)$ be the pebbling time of the pebbling tree T. Then we have

$$time(T) \ge k + \max_{j=1,\ldots,k} \{time(T_j)\}.$$

Let us topologically sort $x_1, \ldots, x_k$, and re-number the indices so that if x_i is an ancestor of x_j then $i < j$. Let $W_j = \{x_1, \ldots, x_{j-1}\}$. Is T_j a pebbling tree for (G, x_j)? It is not. However, we have:

CLAIM 1: For $j = 1, \ldots, k$, T_j is a pebbling tree for $(G, x_j; W_j)$.

To see this, let us introduce the notation $C_T(u)$ for the current set at vertex u when u is viewed as a vertex of T, Similarly, $C_j(u)$ is the current set at u when u is viewed as a vertex of T_j. We note that

$$C_T(u) \subseteq C_j(u) \cup W_k.$$

It follows that T_j is a pebbling tree for $(G, x_j; W_k)$ since T is a pebbling tree for (G, i_0). It easily follows that T_j is a pebbling tree for $(G, x_j; W_j)$.

Our strategy is now simple to describe: We pebble $x_1, x_2, \ldots, x_k$ in turn. For each j, we pebble $(G, x_j; W_j)$ using the strategy described by T_j. After x_j is pebbled, we will remove all pebbles on the graph but leave all the pebbles in W_{j+1}. At the end of this process, we are left with k pebbles on $W_{k+1} = \{x_1, \ldots, x_k\}$. The maximum number of pebbles used is

$$\leq (k-1) + 1 + \max_{j=1,\ldots,k} \{time(T_j)\}.$$

Finally, if i_0 is not among $x_1, \ldots, x_k$, we need one more pebble to pebble i_0 (this is possible since all predecessors of i_0 are among $x_1, \ldots, x_k$). So the total number of pebbles is

$$\leq 1 + (k-1) + 1 + \max_{j=1,\ldots,k} \{time(T_j)\}.$$

This is bounded by $time(T)$.