CS 142 & Math 167
Fall 2008
Homework 8
Due Thursday, November 6 by 5 PM

PLEASE note that this assignment may be turned in on Thursday at 5 PM at Ran's office rather than at the usual time. This is to allow you to stay up late to watch the election returns on Tuesday evening. Also note that the last problem suggests yet another possible student project: Razborov's Amazing Theorem!

1. **[50 Points] IP $\subseteq$ PSPACE!** In this problem, you will show that IP $\subseteq$ PSPACE. In class on Monday, we'll show that PSPACE $\subseteq$ IP leading to the astonishing results that IP = PSPACE. Both directions of the proof are very cool!

   Here we go! First, let's define IP a bit more formally than we did in class. We are given an input string w of length n. The Prover, P, has unbounded computational power while the Verifier, V, is a deterministic polynomial time TM. The Prover and Verifier take turns sending messages to one another in each "stage". In even stages, beginning with stage 0, V spends polynomial time computing and generating a message for P. In odd stages, P generates a message for V with no bounds on the resources used. Of course, the message that is generated is (without loss of generality) only polynomial in the length of w since V doesn't have time to read very long messages. The number of message exchanges is polynomial in the length of w. We can assume, without loss of generality, that the number of stages is *exactly* $q(n)$ for some polynomial $q(n)$. The last message is an "accept" or "reject" emitted by V.

   The Verifier has a source of private random bits (also of some polynomial length because longer sequences would be too long to read in their entirety). These random bits are on a read-only read-once tape: After the machine has read a bit, its tape head is forced to move right to the next random bit. In other words, this tape of random bits is like a coin that can be flipped on demand. It can be convenient to think about these random bits as just a source of a fair coin.

   Finally, we assume that there is a shared "transcript" tape that both V and P can read that has the collective transcript of the messages that were exchanged thus far between the two parties. V and P are allowed to read this transcript as part of their computation.

We write $V(w, r, m_1, \ldots, m_i) = m_{i+1}$ to mean that string m_{i+1} is the string emitted by V in stage i (where i is even) based on the string w, the random bit string r, and the transcript of messages $m_1, \ldots, m_i$ to date. Similarly, we write $P(w, m_1, \ldots, m_i) = m_{i+1}$ to mean that string m_{i+1} is the string emitted by P on stage i (where i is odd).

We say that a language L is in IP if *there exists* a verifier V and prover P such that for every $w \in L$, the probability (taken over all random bit sequences r) that V accepts w is at least $\frac{2}{3}$ and if $w \notin L$ then *for all* provers $\hat{P}$, w is accepted by V with probability at most $\frac{1}{3}$.

Alright, so now we'd like to show that IP $\subseteq$ PSPACE. We'll effectively simulate the Verifier and Prover in polynomial space. **How can we do this when the Prover is unbounded in computational power?!** To answer this, keep in mind the "exists" and "for all" quantifiers in the definition of acceptance of strings in L. In particular, we don't really need to know the "good" prover that we had in mind when building the protocol but rather we can determine the highest probability by which any prover could make the verifier accept the given string. That is, we can determine the behavior of an optimal prover on our own!

Let's let M_j denote a message transcript with j messages, $m_1, \ldots, m_j$. We write $\text{Prob}_{V,P}(w, M_j)$ to denote the probability that V accepts when interacting with P on string w, *starting part way through the protocol from message transcript* M_j. This probability is taken over all possible random strings.

We can now define $\text{Prob}_V(w, M_j) = \max_P \text{Prob}_{V,P}(w, M_j)$. This is the maximum probability of V accepting w (again taken over all random strings) over all possible Provers P. Notice that some transcripts M_j are impossible because there is no sequence of random bits that would make the verifier emit its part of M_j. Also note that our objective is to determine whether or not $\text{Prob}_V(w, M_0) > \frac{2}{3}$.

The plan is to do this in PSPACE using recursion! To that end, let's simplify our notation by making w implicit (not carrying it around in our notation). Let $\pi(M_j)$ denote the probability that V accepts w (when interacting with the best conceivable Prover) when the transcript begins with M_j. If the number of stages is $q = q(n)$, then computing $\pi(M_q)$ for a given transcript of length q is not too difficult. Our goal is to compute $\pi(M_0)$.

Explain in detail how your algorithm works, why it is correct, and why it uses only polynomial space. Conclude that IP = PSPACE.

2. **[25 Points] Examining Complexity Theory with Circuits.** This problem gives us a first glimpse of the a subarea of complexity theory called "circuit complexity." The size of a circuit is the number of logic gates that it contains. A *family of circuits* is an infinite sequence $\mathcal{C} = (C_0, C_1, \ldots)$ of boolean circuits, where C_n has n input variables. We say that a language $L \subseteq \{0,1\}^*$ *has polynomial circuits* if there is a family of circuits $\mathcal{C} = (C_0, C_1, \ldots)$ such that the following are true: First, the size of C_n is at most $p(n)$ for some fixed polynomial p. Second, for all binary strings x, $x \in L$ if and only if $C_{|x|}$ outputs **true** when the $|x|$ bits of x are input into the corresponding input variables of $C_{|x|}$.

   (a) Prove that all languages in the class P have polynomial circuits.

   (b) Is the converse true? That is, do all polynomial circuits correspond to languages in P? No! Prove this by showing that there exist undecidable languages that have polynomial circuits.

   (c) OK, maybe we should refine our notion of polynomial circuits so that we don't get such crazy results! A family of circuits $\mathcal{C} = (C_0, C_1, \ldots)$ is said to be *uniform* if there is a $O(\log n)$ space transducer (TM) which on input 1^n outputs C_n on its write-only output tape. We now say that a language L *has uniformly polynomial circuits* if there is a uniform family of polynomial circuits $(C_0, C_1, \ldots)$ that decides L. Prove that a language L has uniformly polynomial circuits if and only if $L \in P$.

3. **[25 Points] Monotone Circuits.** A boolean function f on n variables is a function $f : \{\textbf{true}, \textbf{false}\}^n \to \{\textbf{true}, \textbf{false}\}$. A boolean function is said to be *monotone* if it has the following property: If one of the inputs changes from **false** to **true**, the value of the function cannot change from **true** to **false**.

   (a) Show that a function f is monotone if and only if it can be expressed as a circuit with only AND and OR gates. (Such a circuit is, not surprisingly, called a "monotone circuit").

   (b) The CLIQUE problem is said to be *monotone* because given an instance G, k where G is encoded as a 0-1 adjacency matrix, if an entry changes from 0 to 1 (**false** to **true**), the answer to this instance cannot go from **true** to **false**. We would like to have a family of *monotone* circuits (circuits using only AND and OR gates) such that for all instances of CLIQUE with n vertices and the decision variable (clique target size k) there is a corresponding monotone circuit with n^2 input gates. Explain

how such a monotone circuit would work. The size of the circuit size should be polynomial in n when k is a constant. *Is it possible to build monotone circuits for* CLIQUE *which have size polynomial in n regardless of the value of k? A very famous theorem due to Razborov says "NO"! This would be a nice student project. The proof of the theorem uses some nifty combinatorics.*