Solutions to Practice Midterm 2

1. Consider the following time-bounded variant of Kolmogorov complexity, written $K_L(x)$, and defined to be the shortest string $\langle M, w, t\rangle$ where t is a positive integer written in binary, and M is a TM that on input w halts with x on its tape within t steps.

   (a) Show that $K_L(x)$ is computable (by describing an algorithm that on input x outputs $K_L(x)$).

   (b) Prove that for all positive integers n, there exists a string x of length n such that $K(x) = O(\log n)$ and $K_L(x) \geq n$. (In fact, there is an algorithm that on input n finds such a x.)

   Solution Outline:

   (a) Here's an algorithm for computing $K_L(x)$. On input x,

      1. Go through all binary strings s in lexicographic order, and for each such s, parse s as $\langle M, w, t\rangle$ for some TM M, input w and integer t. If s fails to parse, move to the next such s.
      2. Simulate M on input w for up to t steps. If it halts within t steps with x on its tape, output $|s|$.

   (b) By a counting argument, it is easy to see that for every n, there exists a string x_n of length at least n such that $K_L(x_n) \geq n$. Choose x_n to be the lexicographically first such string. Now, consider the machine T that on input an integer n written as a binary string, enumerates over all binary strings s in lexicographic order, computes $K_L(s)$, and outputs the first s such that $K_L(s) \geq n$. Then, $T(n) = x_n$, so $\langle T, n\rangle$ is a description for x_n and thus $K(x_n) = O(\log n)$.

2. (Sipser 7.41) For a cnf-formula ϕ with m variables and c clauses (that is, ϕ is the AND of c clauses, each of which is an OR of several variables), show that you can construct in polynomial time an NFA with $O(cm)$ states that accepts all nonsatisfying assignments, represented as Boolean strings of length m. Conclude that the problem of minimizing NFAs (that is, on input a NFA, find the NFA with the smallest number of states that recognizes the same language) cannot be done in polynomial time unless $\mathbf{P} = \mathbf{NP}$.

   Solution Outline: On input ϕ, construct a NFA N that nondeterministically picks one of the c clauses (via ϵ-transitions), reads the input of length m, and accepts if it does not satisfy the clause, and rejects otherwise. In addition, N also accepts all inputs of length not equal to m. For each clause, we need $O(m)$ states, so N has $O(cm)$ states. It is clear that N can be computed in polynomial time. In addition, for any nonsatisfying assignment a, at least one clause is not satisfied, so N accepts a. Conversely, if N accepts a, some clause is not satisfied, so a is a nonsatisfying assignment. Hence, N accepts all the nonsatisfying assignments of ϕ.

   Next, suppose the problem of minimizing NFAs can be done in polynomial time. Then, consider the polynomial-time algorithm that on input a 3cnf formula ϕ with m clauses, constructs a NFA N that accepts all the nonsatisfying assignments of ϕ. Observe that N accepts all binary strings iff ϕ is not satisfiable. Now, run the NFA minimizing algorithm to produce a new NFA N'. If N' contains exactly one state and accepts all binary strings, reject ϕ; otherwise, accept ϕ. This yields a polynomial-time algorithm for $3SAT$, and hence $\mathbf{P} = \mathbf{NP}$.

3. (Sipser 7.33) Prove that the following language is NP-hard

$$D = \{\langle p \rangle \mid p \text{ is a polynomial in several variables having an integral root}\}$$

(The problem is in fact, undecidable. Turing first published the notion of a Turing machine and formalization of algorithms to prove the undecidability of this very problem.)

SOLUTION OUTLINE: We reduce 3SAT to D as follows. For each clause c_i, we define a polynomial $p_i(x_1, \ldots, x_n)$ such that $p_i(x_1, \ldots, x_n) = 0$ iff there is a way of assigning values $0/1$ to the variables in c_i such that the clause is satisfied. For (say) $c_i = (x_2 \vee \overline{x}_5 \vee x_7)$, we have $p_i(x_1, \ldots, x_n) = (1 - x_2)x_5(1 - x_7)$, which is zero if and only if $x_2 = 1$, $x_5 = 0$ or $x_7 = 1$. Interpreting 1 as `true` and 0 as `false`, this is consistent with the formula.

We then define $P(x_1, \ldots, x_n) = \sum_{i=1}^{m}(p_i(x_1, \ldots, x_n))^2$, where m is the total number of clauses. Since, P can be zero only when each of the individual p_i's is zero, an integral root of P gives a satisfying assignment to the given formula and vice-versa.