CS 174 Homework Assignment 7 (due Thursday, April 18)

1. Consider the Karp-Rabin randomized pattern matching algorithm discussed in class. In practice, we might want to eliminate the possibility of false matches in this algorithm entirely. To do this, we could make the algorithm *test* any match before reporting it. If it is found to be a false match, the algorithm could simply restart with a new random prime p. The resulting algorithm never makes an error.

   (a) Show that its expected running time is at most $\frac{c}{c-1}T \approx T$, where T is the running time of the original algorithm

   (b) Show that the probability it runs for at least $(\ell+1)T$ time is at most $c^{-\ell}$

2. Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ be *multisets* over the universe $U = \{0, \ldots, m-1\}$, i.e., each of X and Y consists of n not necessarily distinct elements of U, given in some arbitrary order. Suppose we want to test whether $X = Y$, in the sense that X and Y both contain exactly equal numbers of all elements in U. Clearly this problem can be solved in time $O(n \log n)$ by sorting the members of X and Y and then comparing them. Here we investigate a more efficient randomized algorithm.

   (a) By considering the polynomial $Q_X(z) = (z - x_1)(z - x_2)\ldots(z - x_n)$ (and a similar polynomial $Q_Y(z)$ for Y), design a randomized algorithm based on the Schwartz-Zippel technique for testing whether $X = Y$. Your algorithm should always output "yes" when $X = Y$, and should output "no" with probability at least $1/2$ when $X \neq Y$. You should specify the values of all quantities used in your algorithm (in terms of n).

   (b) Assuming that each arithmetic operation can be performed in constant time, what is the running time of your algorithm as a function of n?

3. Consider a directed acyclic graph (DAG) in which nodes are labeled with Boolean variables from the set $\{x_1, x_2, \ldots, x_n\}$. Call these nodes *variable nodes*. Each variable node has two outgoing edges, one of which is labeled with a 0 and the other of which is labeled with a 1. There are also two additional nodes in the graph, called *output nodes*, each of which is labeled with a 0 or a 1. Finally, we also designate one of the variable nodes as a *start node*.

   We call such a DAG a *read-once branching program* if each variable x_i can appear at most once on any path from the start node to an output node.

   A read-once branching program represents a Boolean function by associating assignments with paths. Given an assignment to the Boolean variables, begin at the start node and follow the path through the graph given by the assignment. That is, if $x_3 = 0$, then, upon arriving at a node labeled x_3, follow the edge labeled 0. The value of the function

corresponding to the assignment is given by the label of the output node at the end of this path.

Propose a randomized algorithm that decides whether two read-once branching programs represent the same Boolean function. [Hint: Convert this problem into an equality test between a pair of multivariable polynomials. A given read-once branching program can be converted into a polynomial by associating a polynomial with each node and each edge in the graph. If the node labeled x is associated with a polynomial p, associate xp with the outgoing 1-edge and $(1-x)p$ with the outgoing 0-edge. The polynomials on edges incoming to a node are summed. The polynomial associated to the graph is that associated with the output node labeled 1.]