Computer Science 2300: Homework 5

Due: April 7, 2011

Note: Please use rigorous, formal arguments. If you are asked to provide an algorithm then you may either write pseudocode similar to the pseudocode in the DPV text, or provide a clear description in English. You **must** also provide an argument for why the algorithm is correct, and an analysis of the running time. We encourage you to collaborate with other students, while respecting the collaboration policy. Please write the names of all the other students you collaborated with on the homework. **Hardcopies are required by submission time. E-mailed versions will not be accepted.**

1. (5 points) Given a uniform hash function, derive the probability that there will be no collisions when inserting n elements into a hash table of size m.

2. (5 points). Suppose you want to also support deletions in a hash table. Suppose you implemented deletion in an open-addressed hash table when using linear probing by the following simple method: When searching for key x, compute $h(x)$, the hash of x. Then search linearly, starting from position $h(x)$ until you find the element with key x. Then delete that element from the hash table. Give an example of a situation where this would lead to incorrect future results when using the hash table.

3. (10 points) Suppose social security numbers are generated uniformly at random (with replacement: this is obviously a massive oversimplification because two people could then have the same social security number). How many people need to be in a room before it is more likely than not that two people in the room have the same last four digits of their SSNs?

4. (15 points total)

   (a) (5 points) Assume you have access to a function that, in constant time, generates a real number uniformly at random between 0 and 1. Suppose you are given a discrete probability distribution, that is:
   A set of n real numbers $p_1, p_2, \ldots, p_n$ such that $\sum_{i=1}^{n} p_i = 1$.
   Give an algorithm that generates a random integer x between 1 and n satisfying the condition that $\Pr(x = i) = p_i$, and runs in time $O(n + \log n)$ (the $\log n$ is left in there as a hint). So, for example, if $n = 3$ with $p_1 = 0.3, p_2 = 0.5, p_3 = 0.2$ then it should generate 1 with probability 0.3, 2 with probability 0.5, and 3 with probability 0.2.

   (b) (10 points) Let's generalize this. Again, suppose you are given a discrete probability distribution $p_1, p_2, \ldots, p_n$ such that $\sum_{i=1}^{n} p_i = 1$, and access to a function that generates a real number uniformly at random between 0 and 1 in constant time. Give an algorithm to generate k integers $x_1, \ldots, x_k$ between 1 and n so that for each j,

$\Pr(x_j = m) = p_m$. Your algorithm should run in time $O(n + k \log k)$. (Note that an obvious generalization of Part (a) would give you a running time of $O(n + k \log n)$. This version will be be useful in situations where $n \gg k$).

5. (15 points, based on CLRS Problem 11.2-6). Suppose you have stored n keys in a hash table of size m, with collisions resolved by chaining, and you know the length of each chain. Let L be the length of the longest chain. Each entry in the hash table is of course associated with a key. Design an algorithm that selects a key uniformly at random among the keys in the hash table (remember that you don't have direct access to the keys, only to the hash table). You algorithm should run in expected time $O(L(1 + 1/\alpha))$ where $\alpha = n/m$. You must prove the running time bound!