Computer Science 2300: Homework 5

Due: April 9, 2012

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. (10 points) DPV Problem 5.17 (longest codeword)
2. (10 points) DPV Problem [5.31 in the online version] / [5.32 in the print version] (efficient customer processing: **be sure to show why the algorithm is optimal**)
3. (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 $O(n)$. 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 $O(n + k \log k)$.
4. (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!