Assignment 1 CSC263
Due: Oct 2, 2008

1. Suppose two jobs are to be scheduled to run on a computer. Job A takes a units of time. With probability p job B takes b units of time and with probability $1-p$ it takes $b' > b$ units of time. The length of time for job B is not known until it has been executing for b units of time. Then either it has terminated or will neeed another $b' - b$ units of time.

   Pre-emption is allowed. This means that a job can be stopped before it is finished and then resumed later. We will assume there is no overhead for doing this.

   What is the best way to schedule these jobs (starting at time 0) so as to minimize the expected sum of the completion times of the two jobs? What is this minimum? Note that your answers will depend on the relationships between the values of the parameters a, b, b', and p.

   In your analysis, carefully define the probability space and any random variables that you use.

2. Consider the following hybrid data structure for the dictionary abstract data type:

   A set is represented by a linked list of arrays, each of size $2^k - 1$. An array contains at least one real element (i.e., element in the set) and zero or more dummy elements (i.e., ∞) stored in nondecreasing order. The real elements in an array are all less than the real elements in the next array in the linked list.

   SEARCH is performed by examining the first element of each array in the linked list until the only array that can contain the key is found and then performing binary search on that array.

   (a) Write a pseudocode algorithm for SEARCH as described above. State appropriate preconditions and postconditions for your algorithm and appropriate loop invariants for any loops.

   (b) What is the worst case running time of your SEARCH algorithm? For running time, count the number of comparisons performed.

   (c) What is the average case running time of your SEARCH algorithm, assuming that every array in the list contains $2^k - 1$ real elements? For the probability distribution, assume that each element in the set is equally likely to be sought.

   (d) Write a pseudocode algorithm for INSERT that, except in the case when the set contains fewer than 2^{k-1} elements, maintains that each array in the linked list contains at least 2^{k-1} elements. State appropriate preconditions and postconditions for your algorithm and appropriate loop invariants for any loops.

   (e) What is the worst case running time of your INSERT algorithm? For running time, count the number of comparisons and the number of movements of the real elements performed.

   (f) For what choices of n and k does your INSERT algorithm for this data structure have better worst case running time than the INSERT algorithm for a dictionary represented by a sorted linked list?

3. A *half-height-balanced tree* is a leaf-oriented binary search tree (i.e., a strict binary tree with all dictionary elements stored at leaves) with the following properties:

   - Each internal node v stores
     - a pointer $p(v)$ to its parent or nil (if it is the root),
     - a pointer $r(v)$ to the rightmost leaf in its left subtree,
     - $l(v) =$ the length of the longest path from v down to a leaf, and
     - $s(v) =$ the length of the shortest path from v down to a leaf.
   - Each leaf node stores one dictionary element and a pointer to its parent or nil (if it is the root).
   - $l(v) \leq 2s(v)$ for all internal nodes v.

   (a) Prove that a half-height-balanced tree with n leaves has $O(\log n)$ height.

   (b) Give an algorithm to INSERT a new element into a half-height-balanced tree, given a pointer to the root of the tree. At most how many rotations are required to rebalance the tree? Explain why your algorithm is correct.