CS 261 – Spring 2013 – Midterm

Name:

Student ID:

1:

2:

3:

4:

5:

6:

Total:

1. (20 points) Suppose that we implement a simplified version of the dynamic array data structure in which the amount of memory allocated for the dynamic array is always the smallest power of two that is at least as large as the number of items in the array. If we try to do an increase-length (insertion) operation that would cause the number of items in the array to be larger than this power of two, we reallocate a new piece of memory twice as long as the previous one, and if we try to do a decrease-length (deletion) operation that would cause the number of remaining items to be smaller than half of this power of two, we reallocate a new piece of memory half as large.

   (a) Using the potential function $\Phi = 2 \times (\text{number of items in array}) - (\text{allocated size})$, show that the amortized time per insertion (for a sequence of insertions without any deletions, starting from an empty array) is constant.

   (b) Using the different potential function $\Phi' = (\text{allocated size}) - (\text{number of items in array})$, show that the amortized time per deletion (for a sequence of deletions without any insertions, starting from an array with exactly 2^k items in it, for some chosen value k) is constant.

   (c) Do these two pieces of analysis together imply that this version of the dynamic array uses constant amortized time for insertions and deletions mixed together? Why or why not?

2. (20 points) The time analysis of Fibonacci heaps uses the fact that certain shapes of trees cannot be formed by the Fibonacci heap operations. Draw an example of a tree that cannot be formed as part of a Fibonacci heap. (You do not need to draw the priorities of the items stored in the tree, just the shape of the tree itself.)

3. (20 points) Suppose that we are inserting some number of keys into a Cuckoo hash data structure in which each of the two tables has room for ten items.

   (a) Give an example of three keys and an assignment of hash values for each of these keys that, when inserted, would be certain to cause the cuckoo hash structure to fail and need to be rebuilt.

   (b) If the hash functions we are using are 3-independent, for tables of this size, what is the probability that any particular triple of keys has hash values that cause the structure to fail?

4. (20 points) If we are doing tabulation hashing for 32-bit keys, broken into 8-bit bytes, how many random numbers are needed to initialize the tables used in the hash function?

5. (20 points) Suppose that we are using a software library that includes a binary search tree structure that allows us to insert or delete data items with associated keys, and to answer predecessor or successor queries that find the item with the smallest key larger than a query value q. However, the library does not give us access to the internal structure of the tree, so we can't directly follow paths in it: we have to use the operations we are already given.

   Rather than using this structure as a binary search tree, we wish to use it as a priority queue. Describe (in English or pseudocode) how to use the predecessor or successor operations to find the highest priority item in the structure, assuming that the keys are being used as priorities and that lower key numbers correspond to higher priorities. If you need them, you may assume the existence of special key values $-\infty$ and $+\infty$ that are less than (greater than, respectively) all data keys.

6. (20 points) If we have an unsorted array of n items, and we search it by scanning from the start rather than by using binary search, the time to search for the item in position i will be $O(i+1)$. In particular, searching for items that happen to be near the start of the array will be faster than $O(\log n)$. Describe an assignment of weights w_i for the items such that the $O(\log(W/w_i))$ time bound for splay trees (with $W = \sum w_i$) is at least as fast as this $O(i+1)$ time bound for scanning an array.

You may use this page (or the back of the other pages) as scratch paper.