Homework 1

CS6043 Design and Analysis of Algorithms II
Fall 2009

Due: Monday, 10/12/09 in class
Maximum Score: 125 points

Note: This assignment has two pages.

1. (10 points) Textbook Exercise 17.2-3 (page 412). (The assumption for the running time of the basic operations is the same as in the textbook: setting/resetting/checking a bit takes $O(1)$ time each.) You should give both an algorithm and an analysis. **(10 points)**

2. (35 points) Textbook Problem 17-2 (page 426), with the following modifications:
a. Same question as in the textbook. **(5 points)**
b. Same question as in the textbook. **(10 points)**
c. Design and analyze an algorithm to perform DELETE. Excluding the time for searching the item to be deleted, your algorithm should use $O(n)$ worst-case time but only $O(1)$ amortized time per deletion for re-structuring the data structure. You should still use only linear storage at any time, and make sure that the running times of SEARCH and INSERT still stay the same.
(**Hint:** Consider a *lazy deletion* scheme, where you either mark an item as deleted or actually perform deletion, in appropriate situations.) **(20 points)**
(**Comment:** In many applications the searching step in DELETE can actually be skipped by providing a pointer to each item. For example, each item has an ID and a key. We maintain a "look-up table" consisting of item ID's and each item ID has a pointer to its representative stored in the data structure in question, which is ordered by the key values of the items.)

3. (37 points) Textbook Problem 17-3 (page 427), with the following modifications:
a. Same question as in the textbook. **(5 points)**
b. Same question as in the textbook. **(5 points)**
c. Same question as in the textbook. **(2 points)**
d. Suppose rebuilding an m-node subtree takes $d \cdot m$ time for some constant d, and this time cost can be paid by the same amount (i.e., $d \cdot m$) of potential. Derive a large enough value for c in terms of d and α so that rebuilding a subtree that is not α-balanced will take $O(1)$ amortized time.
(**Hint:** Recall that amortized cost = actual cost + $\Delta\Phi(T)$, where $\Delta\Phi(T)$ is the potential difference after the rebuilding. Let $\Phi'(T) = \Phi(T)/c$ (i.e., removing the term c from $\Phi(T)$). Derive a lower bound for $|\Delta\Phi'(T)|$, and give a value for c such that $c \cdot |\Delta\Phi'(T)|$ is big enough to pay for the actual cost.) **(15 points)**
e. Same question as in the textbook.
(**Hint:** For each of insertion and deletion, you should consider two cases: (1) rebuilding does not occur, and (2) rebuilding does occur.) **(10 points)**

4. (23 points) Your task in this question is to provide more operations supported by binomial heaps. Assume that n is the current number of nodes in the binomial heap in question.

a. UPDATE-KEY**.**
In addition to decreasing the key value, we would like to support a general UPDATE-KEY operation that also allows us to increase a key value. (1) In the DECREASE-KEY operation, the item whose key is decreased "bubbles up" along a leaf-to-root path. Give an algorithm for increasing the key value of an item, by "pushing down" the item analogous to DECREASE-KEY. Analyze the worst-case running time. (2) Give a more efficient algorithm that performs UPDATE-KEY in worst-case $O(\log n)$ time. **($5 + 3 = 8$ points)**

b. SPLIT**.**
The SPLIT operation can be considered as an inverse operation of UNION. SPLIT takes as an input a number k, where $k < n$, and split the current binomial heap H with n nodes into two binomial heaps H_1 and H_2 with k nodes and $n - k$ nodes respectively. The original heap H is destroyed. The choice of which nodes of H go to H_1 and which nodes go to H_2 can be arbitrary. Give an algorithm to perform SPLIT and analyze its performance. The worst-case running time should be $O(\lg n)$.
(**Hint:** Consider subtracting two binary numbers.) **(15 points)**

5. (20 points) Textbook Problem 19-2 (page 474).
a. Look at lines 7 and 9 of the algorithm MST(G), and consider the two cases $i \neq j$ and $i = j$. Argue that the algorithm works correctly for both cases, namely, argue that the operations performed in each case result in a correct final answer. **(7 points)**
b. Answer the questions in the textbook. **(13 points)**