## CSE 373 Problem Set 3 Due: May 18, 1998

1. What is the minimum number of keys that can be stored in a B-tree of order $m$ and height $h$? A tree with a single node is of height 0.

2. Text, problem 4.36, page 143.

3. Text, problem 4.23, page 141.

4. Text, problem 4.24, page 141.

5. We define the operations of $Split(x, T)$ and $Join(T_1, T_2)$ for splay trees. $Split(x, T)$ takes a splay tree $T$ and key $x$ (not necessarily contained in $T$) and returns two splay trees $T_1$ and $T_2$ such that $T_1$ contains all the keys in $T$ that are less than or equal to $x$ and $T_2$ contains all the keys in $T$ that are greater than $x$. Join operates on two splay trees $T_1$ and $T_2$ where all the keys in $T_1$ are less than all the keys in $T_2$. It combines $T_1$ and $T_2$ into a splay tree $T$ which contains all the keys in either $T_1$ or $T_2$.

   (a) Recall that the operations $Find(x, T)$ and $Insert(x, T)$ on a splay tree $T$ bring the key $x$ to the root using zig-zag, zig-zig and zig operations. Describe how to perform the operations $Split(x, T)$ and $Join(T_1, T_2)$ using the operations $Find(x, T)$ and $Insert(x, T)$ as subroutines.

   (b) Describe an algorithm that uses split and join operations to do range deletion. Range deletion takes a splay tree $T$ together with keys $x$ and $y$, where $x \leq y$, and returns a splay tree $T'$ which contains all the keys $z$ in $T$ except those where $x \leq z \leq y$. In other words, range deletion removes all the keys between $x$ and $y$, inclusive. You can use an abstract drawing or pseudocode to describe your algorithm. Your algorithm should use just a few split and join operations to do the job of range deletion. Explain why your algorithm is correct.

6. Consider a hash table with 11 locations, in which locations $0, 1, 5, 8$ and 10 are occupied, and the other locations are unoccupied.

   (a) Suppose hashing with linear probing is being used, with hash function $h$. Suppose key $x$ is inserted into this table, and ends up in location 2. What can you say about the value of $h(x)$?

   (b) Same question for quadratic probing.

7. Double hashing is a closed hashing method that requires two hash functions $h_0(x)$ and $k(x)$. The sequence of addresses probed when inserting key $x$ into the hash table is $h_0(x), (h_0(x) + k(x)) \bmod M, (h_0(x) + 2k(x)) \bmod M, \cdots$. Show the steps in inserting the following sequence

of keys when $M = 7$, $h_0(x) = x \bmod 7$ and $k(x) = (x \bmod 6) + 1$: $73, 19, 56, 45, 84, 23, 43$.

8. A spell checker reads an input file and prints out all the words not contained in an online dictionary. A simple strategy for building a spell checker is to insert each word of the dictionary into a hash table. Then, whenever a file is to be spell-checked, a find operation on the hash table is performed on each word of the input file.

   An alternative method, which is not foolproof but will catch most spelling errors, is to choose a hash function $h$ which maps the universe of all possible words to the range $\{0, 1, \cdots, M-1\}$, and then construct an array $H$ of bits, where, for $i = 0, 1, \cdots, M-1$, $H[i]$ is equal to 1 if there exists a word $x$ in the dictionary such that $h(x) = i$. Then, whenever a file is to be spell checked, the spell checker computes hash address $h(y)$ of each word in the input file. If $H[h(y)] = 0$ then it declares that $y$ is not in the dictionary; if $H[h(y)] = 1$ then it declares that $y$ is in the dictionary.

   (a) How would you construct the array $H$?

   (b) Does this algorithm ever falsely declare that a word is in the dictionary?

   (c) Does this algorithm ever falsely declare that a word is not in the dictionary?

   (d) Assume that, whenever a word is hashed into $H$ for the first time, its hash address is equally likely to be any address in $H$. How frequently will the algorithm make an error if there are $30,000$ words in the dictionary and $M = 300,000$?

9. (a) Show the result of inserting $10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13$ and $2$, one at a time, into an initially empty binary heap.

   (b) Show the steps in applying Heapsort to the above keys, assuming they are initially stored in an array in the order of their occurrence above.

10. Weiss, problem 6.15