CSE 326: Data Structures
Assignment #6
February 27, 1998
due: Wednesday, March 11

1. (a) Suppose n is odd. You are given a splay tree on n nodes such that the path from the root to the key 1 passes through nodes with keys in the order

$$n, n-1, \ldots, 7, 6, 5, 4, 3, 2, 1.$$

   Show the splay trees before and after splaying on the key 1.

   (b) How many dollars must be paid just for rotations in part (a)? (I.e., for this part ignore dollars spent to maintain the Money Invariant.)

   (c) The Investment Lemma says we are paid only $3\lfloor \log_2 n \rfloor + 1$ new dollars for the splay of part (a), so anything in excess of this in part (b) must come from money stored in the splay tree before the splay. For $1 \le i \le n$, what was the rank of the node with key i before the splay? For $1 \le i \le n$, what is the rank of the node with key i after the splay? From which nodes can we take money to pay for the rotations? Show that these nodes have enough excess money to pay for all the rotations.

2. In this question you will explore what happens if you implement a dictionary as a splay tree, and do m dictionary operations starting from an arbitrary z-node splay tree T rather than the empty tree.

   (a) Suppose that someone invested the minimum number of dollars dictated by the Money Invariant in each node of T. Show that the total number of dollars invested in T would be $O(z \log z)$.

   (b) Show carefully that any m dictionary operations starting from T take a total time of $O(z \log z + m \log n)$, where n is the maximum size the dictionary attains during these operations. Note that in the proof given in lecture and the text, money invested in the tree represents unused time saved from earlier operations. With this interpretation, you wouldn't be able to assume that there is any money invested in T initially, since there haven't been any earlier operations. (What makes the empty tree a good starting place is that no initial investment is needed in order for the Money Invariant to hold immediately.)

3. Insert the integers 13, 92, 22, 8, 48, 34, 19, 77, 21, 55 (in this order) into an initially empty hash table of size 11 using the hash function $h(x) = x \bmod 11$,

   (a) using separate chaining.

   (b) using open addressing with linear probing.

(c) using open addressing with double hashing, where $h_2(x) = 1 + (x \bmod 10)$.

(d) If a hash table is going to be this full (i.e., $n \approx m$) most of the time, which of these methods is likely to be fastest, and why?

4. Consider the universal class consisting of the hash functions

$$h_{a,b}(x) = ((ax + b) \bmod 71) \bmod 11$$

for $0 < a < 71$ and $0 \leq b < 71$. Let x_1 and x_2 be any two distinct keys in $\{0, 1, \ldots, 70\}$. Justify your answers to the following. (Hint: you definitely don't want to start listing all the (a, b) pairs, as there are 4970 of them. Instead, use what you know from the proof of the Universal Classes of Hash Functions Theorem.)

   (a) Exactly how many of the $70 \cdot 71$ (a, b) pairs hash both x_1 and x_2 into bucket 4 (where the buckets are numbered $0, 1, \ldots, 10$ according to the value of $h_{a,b}(x)$)? Exactly how many hash both x_1 and x_2 into bucket 5?

   (b) Exactly how many ordered pairs of distinct numbers (q, r) are there such that $0 \leq q, r < 71$ and $q \equiv r \pmod{11}$? Compare your answer to the upper bound $N(N-1)/m$ proved in the Theorem.

   (c) What is the exact probability that a randomly chosen hash function $h_{a,b}$ will cause x_1 and x_2 to collide? Compare your answer to the upper bound $1/m$ proved in the Theorem.

5. Page 331, problem 1. Show the partially ordered tree after each Insert and after each DeleteMin. Also show the heap in table form after the last Insert.

6. Another useful heap operation is ChangePriority(i, K, h), which changes the key stored in heap entry $(\text{Table}(h))[i]$ to K.

   (a) Write an algorithm that implements the operation ChangePriority.

   (b) Show that your algorithm runs in time $O(\log n)$, where n is the number of entries in the heap.