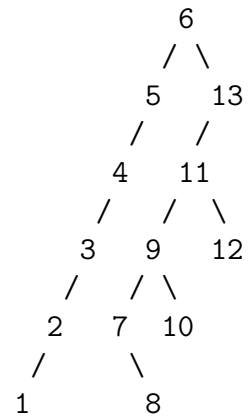
CSE 326: Data Structures
Assignment #5
November 15, 2004
due: Monday, November 22

1. Suppose you are splaying on the key K, which is not present in the splay tree. A student asked whether you should always splay whichever of the immediate predecessor or immediate successor is closer to K. Suppose that this was indeed the way the splay algorithm was done. Construct a splay tree with the n keys $1, 2, 3, \ldots, n-2, n-1, n+2$ and the property that there is a key K such that LookUp(K, T) would require n key comparisons, but would not change the shape of T at all. Therefore repeating this same LookUp m times would take time $O(mn)$ instead of $O(m \log n)$.

2. Consider the following splay tree T.



   (a) Show the results of each of the rotations to splay the key 9 to the root.

   (b) In part (a), let T_1 be your splay tree after the first Case I, II, or III rotation, and T_2 be your splay tree after the second. Go back and label the nodes of the 3 trees T, T_1, and T_2 with their ranks.

   (c) Suppose the Money Invariant holds for T, and suppose you were only paid \$2 to splay the key 9 in T. According to the proof of the Cost of Splay Steps Lemma, for which of your two rotations (T to T_1, or T_1 to T_2) would you have to take \$1 from the tree, and from which node would the \$1 come?

   (d) Exactly how would the \$2 you are paid for the splay in part (c) be used?

3. (a) Suppose n is odd. You are given a splay tree on n nodes such that the path from the root to the key 0 passes through nodes with keys in the order

$$-\frac{n-1}{2}, \frac{n-1}{2}, -\frac{n-3}{2}, \frac{n-3}{2}, \ldots, -2, 2, -1, 1, 0.$$

   Show the splay trees before and after splaying on the key 0.

(b) How many dollars must be paid just for rotations in part (a)? (That is, 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. Answer the following questions precisely. For $-\frac{n-1}{2} \le i \le \frac{n-1}{2}$, what was the rank of the node with key i before the splay? For $-\frac{n-1}{2} \le i \le \frac{n-1}{2}$, 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.

4. We showed that you can delete key K from a splay tree if you are paid $7\lfloor \log n \rfloor + 2$ dollars (in addition to the dollars that the Money Invariant states are already in the tree). Recall that the breakdown for this figure was $3\lfloor \log n \rfloor + 1$ dollars to splay on K, another $3\lfloor \log n \rfloor + 1$ dollars to splay the left subtree on $+\infty$, and $\lfloor \log n \rfloor$ extra dollars to invest in the new root because it takes on new descendents.

   Prove that any Delete in a splay tree can be accomplished (maintaining the Money Invariant and paying \$1 per rotation, of course) if the client pays only $5\lfloor \log n \rfloor + 2$ dollars. (Hint: prove that $3\lfloor \log n \rfloor + 1$ dollars is sufficient for the Concat and that another $\lfloor \log n \rfloor$ dollars can be saved in the other part of Delete. In both cases the trick is to be careful about the investment at the root.)

5. Insert the integers 87, 19, 25, 55, 36, 46, 88, 7, 67, 21 (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, and you have a hash function that spreads the entries over the hash table reasonably well, which of these methods is likely to be fastest, and why?