CSE 326: Data Structures
Assignment #5
May 1, 1996
due: Friday, May 10

Implement the dictionary abstract data type using splay trees. Here are the details:

1. Implement the procedures Rotate, Splay, LookUp, Insert, Concat, and Delete as described in Algorithm 7.3 and Section 7.3. Also implement a procedure to display the splay tree after each dictionary operation, so that you can watch its shape change. (A fully parenthesized inorder traversal, as on page 222, line 7 from the bottom, will suffice if you don't want to attempt anything more picturesque.) With the exception of Splay and your possibly picturesque display procedure, these procedures are very short and easy, so don't be concerned about the number of procedures you have to implement.

   Design Splay recursively, so that the recursion stack will do the job of remembering the path back to the root. (Hint: Because the type of rotation done depends on the path to P from the grandparent of P, you need some ancestral context after returning from a recursive call. Design your procedure so that when the recursive call returns, P is either at depth 1 or depth 2 from the current root. If it is at depth 1, then simply return without doing any rotation; if at depth 2, then do the appropriate two rotations before returning. You may want your recursive call to return a pointer to P, to help you find P from the current root. Notice that when all the recursion ends, P may be left at depth 1 of the entire tree, so some cleanup may be necessary.)

   To implement Concat, you can use any convenient key from T_2 in place of $+\infty$.

2. Run some experiments with your dictionary package, trying to find a sequence of operations from an initially empty tree that causes some of its operations to take $\Omega(n)$ time (even though the average time must be $O(\log n)$). What happens to the shape of the tree after a few such expensive operations? Include a short report on these experiments in your `README` file.