Algorithms Core 15-750 PROBLEM SET # 2 Spring 1998

DUE: 10:30AM Wednesday 18 February 1998

1 Amortized Analysis

Show how to implement a linear-time dynamic data structure supporting the First-In-First-Out (FIFO) queue operations: `Enqueue`, `Dequeue` and `Minimum`. The `Minimum` operation should report the smallest value currently in the queue. [Hints: (1) Aim for $O(1)$ amortized costs. (2) You may want to use two arrays.]

2 Maintaining a list with reversals

Consider a data structure that represents an ordered list of elements under the following three types of operations:

access(k):	Return the kth element of the list (in its current order).
insert(k, x):	Insert x (a new element) after the kth element in the current version of the list.
reverse(i, j)	Reverse the order of the ith through jth elements.

For example, if the initial list is $[a, b, c, d, e]$, then access(2) returns b. After reverse(2,4), the represented list becomes $[a, d, c, b, e]$, and then access(2) returns d.

2.1 Reversals using Splay Trees

Show how to modify splay tree construction so that each operation runs in $O(\log n)$ amortized time, where n is the (current) number of elements in the list. The list starts out empty.

Hint: First consider how to implement access and insert using splay trees. Then think about a special case of reverse in which the $[i, j]$ range is represented by a whole subtree. Use these ideas to solve the real problem. Remember, if you store extra information in the tree, you must state how this information can be maintained under various restructuring operations.

(This data structure is useful in efficiently implementing the Lin Kernighan heuristic for the travelling salesman problem. This is a good idea for a project in this course.)

2.2 Reversals using Treaps

Show how these three operations can also be implemented in $O(logn)$ expected time using treaps.