Burt Rosenberg

Extra Problem A

Out: 23 November, 1993
Due: 3 December, 1993

Write a program to perform a weird form of sorting called *shaker sort.* Use a doubly linked list data structure. The algorithm works by making repeated passes over the data, from front to back, then from back to front, taking out of the data increasing subsequences. With each pass, the extracted subsequence is merged onto an output list. This list is initially empty and is otherwise kept in increasing order.

Extract the increasing subsequence from the list in the most obvious manner possible. Depending on the direction of the scan, select the first item in the list. If presently the scan direction is forward, the first item is the list head, if the scan direction is backwards, the first item is the list tail. Continue in the direction of the scan, selecting the next item larger than the first, and then the next item larger than that, and so on.

For instance, here is the original data, in a doubly linked list:

$$\text{Input} = \langle 23, 26, 1, 94, 88, 11, 83, 30, 82, 52 \rangle$$

The first pass forward extracts:

$$\langle 23, 26, 94 \rangle$$

and merges this onto an initially empty list. Next, what remains from the original list:

$$\text{Input}' = \langle 1, 88, 11, 83, 30, 82, 52 \rangle$$

is scanned from back to front and we extract:

$$\langle 52, 82, 83, 88 \rangle$$

note that we can reverse the order during the extraction. This is merged onto the list:

$$\text{Output} = \langle 23, 26, 94 \rangle$$

to give a list:

$$\text{Output}' = \langle 23, 26, 52, 82, 83, 88, 94 \rangle.$$

The remains of the original list:

$$\text{Input}'' = \langle 1, 11, 30 \rangle$$

is scanned again, front to back, and the extracted sequence is the entire list. (If there was more remaining we would scan again back-to-front, then front-to-back, etc.) It is merged into the output list and we are done:

$$\text{Output}'' = \langle 1, 11, 23, 26, 30, 52, 82, 83, 88, 94 \rangle.$$

Implement this algorithm using queues, stacks, and doubly-linked lists. The run time of the algorithm depends upon home many times you must scan the input list. If can be shown that in a list of n items, there is either an increasing subsequence of length $\sqrt{n}$ or a decreasing subsequence of length $\sqrt{n}$. Hence in each possibly $O(n)$ pair of passes through the list, $\sqrt{n}$ elements are removed. Hence there can be at most $\sqrt{n}$ pair of passes through the list.