6.851 Advanced Data Structures (Spring'14)

Prof. Erik Demaine TAs: Timothy Kaler, Aaron Sidford

Problem 6 *Due: Monday, Mar. 31*

Be sure to read the instructions on the assignments section of the class web page.

Remember that your solution write-up must be your own and self-contained. You must cite any references you use, and just writing "this was solved in paper ..." is *not* a valid solution.

Motivation

Recall that, in fractional cascading, every list stores two types of nodes: nodes containing actual elements in the list (call them *red nodes*) and nodes containing elements propagated from preceding lists (call them *black nodes*). For every red node, we had pointer to the nearest black nodes, and for every black node, we had a pointer to the nearest red nodes. For the static case, these pointers were easily computed, but in the dynamic case, more work needs to be done. For this week's problem set, we phrase maintaining these pointers as a self-contained subproblem for you to solve.

Problem

Describe and analyze a dynamic data structure for maintaining a dynamic ordered list of n elements. The elements have two colors, ***red*** and ***black***, and your data structure should allow for querying for the preceding and succeeding elements of either color. More precisely, your data structure should support the following operations:

insertAfter$(x, d, c \in \{\textbf{\textit{red}}, \textbf{\textit{black}}\})$**:** Make a new node y of color c with data d. Insert node y immediately after node x, and return a reference to y.

delete(x)**:** Delete node x.

prev$(x, c \in \{\textbf{\textit{red}}, \textbf{\textit{black}}\})$**:** Return the first node of color c that occurs before node x.

next$(x, c \in \{\textbf{\textit{red}}, \textbf{\textit{black}}\})$**:** Return the first node of color c that occurs after node x.

Your data structure should work in the word-RAM model and support insertAfter, delete, next, and prev operations in $O(\log \log n)$ amortized time per operation. The total space usage of your data structure should be $O(n)$.

You may use any standard results we saw in class regarding van Emde Boas or list labeling. Note that the data values cannot be manipulated (not even compared), and thus serve no useful purpose for data structuring.