CS 360 100P
The Mysterious Square-Root Data Structure

Revision Date: January 10, 2013

Topics to learn

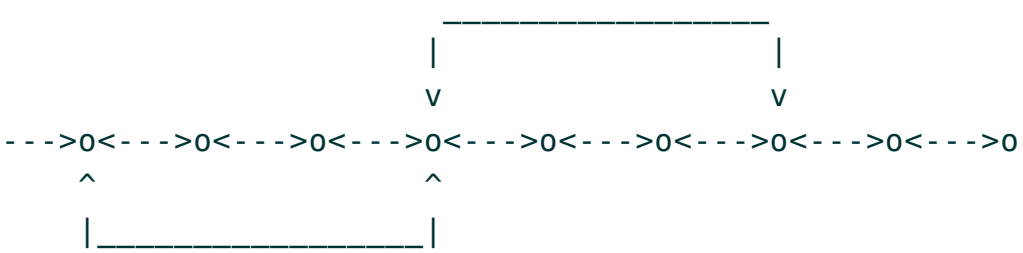
Please read up on the following subjects:

- doubly-linked lists
- forward/skip pointers

Tasks to perform

This a group project. You are to implement a data structure whose *insert*, *delete* and *find* operations have a time complexity of *theta(sqrt(n))* time. One way to do this is to maintain a data structure that keeps *n* items organized as *sqrt(n)* columns, each with *sqrt(()n)* items.

You should maintain a sorted linked-list of elements with forward pointers to every *z* elements, where *z* is $(z - 1)^2 < n <= z^2$. Here is the state of the structure after the ninth element is inserted:



The elements with the forward pointers are known as base elements. You will need to keep an

updated count of how many elements are between a one base element and the next. If an insertion causes that number to become too large (i.e. *count* is $z + 1$), the forward pointers will need to be adjusted. Of course, this adjustment should take *O(sqrt(n))* time.

When a column has too many items, either the largest item in the column is shifted to the column on the right or the smallest item in the column is shifted to the column on the left. Shifting an item to an adjacent column has to take *O(1)* time for the *insert* time bound to be met.

For deletion, disassembly is the reverse of assembly. Do as little work as possible for each deletion.

Hints:

- Your node should have four pointers: *next*, *previous*, *nextColumn*, and *previousColumn*. It should also have a count. Keep a dummy head and a dummy tail node. When the structure is filled, expand the number of columns by one, placing some of the largest elements in the newly created column.
- When a column gets too large, move its largest element to the column to the right, repeatedly, until you end up with a column that is not overfilled. If the last column is overfilled as a result, repeat the process, moving its smallest element to the column to the left, repeatedly.

  Of course, you are not ever moving elements, just adjusting column pointers.

You must implement this data structure using the Java programming language. You must provide a makefile.

Testing

Provide a complete mini-interpreter that tests your data structure in such a way that your Mentor can easily ascertain the correctness of your solution.

Other concepts

Through pictures (on the board is fine), describe your algorithm to your mentor before starting your coding.

Be able to describe to your mentor the complexity of your *insert* and *find* methods.