CSCA48, Assignment #1-2
Implementation of Matrix Class
due 11:55pm July 2, 2017

Introduction

We are now fast-forwarding in time several steps after you presented your designs from phase 1 to the client. A few iterations of tweaking and discussion have repeated, and we now have a much better idea of the ADT that we need to implement. You may notice that some things have been added, and some taken away, this is normal for a project like this (and shows why it's a good idea to settle on a design before you spend large chunks of time developing the actual code). The ADT is almost finalized, and now we can start to work on the implementation.

The ADT

The starter code provided in `a1.py` represents an almost complete ADT. The class hierarchy is settled, and all of the methods have been outlined, at least at their highest level. We've decided which exception types we need, but not exactly when and where they should be raised, and we haven't figured out yet which methods will need to be over-ridden in children classes. These final elements of the design should be your first task.

Implementation

Once we're ready to implement the actual Matrix class and its sub-classes, we need to think about a sensible data structure that we can use. Obviously we could simply have lists of lists, but this seems impractical. What if the client wants to create a 1000 x 1000 matrix, but only cares about 1 or 2 values. We'd be using up millions of times more space than we actually need. Fortunately, we've learned about linked lists in lecture. Maybe they can help us.

Linked Data Structure

Suppose we have the matrix:

$$\begin{pmatrix} A & D & G \\ B & E & H \\ C & F & I \end{pmatrix}$$

Each element can be held in a node like in a linked list, with A pointing to B, etc. This would be fine if all we cared about were the columns, but sometimes we're going to need to deal with rows, so maybe each node could have a link to the node, beside it as well (i.e, A could also link to D). We could create a linked data structure, that can hold a matrix of elements, like so:

$$\begin{tikzcd}
\boxed{\text{A}} \arrow[r] \arrow[d] & \boxed{\text{D}} \arrow[r] \arrow[d] & \boxed{\text{G}} \arrow[d] \\
\boxed{\text{B}} \arrow[r] \arrow[d] & \boxed{\text{E}} \arrow[r] \arrow[d] & \boxed{\text{H}} \arrow[d] \\
\boxed{\text{C}} \arrow[r] & \boxed{\text{F}} \arrow[r] & \boxed{\text{I}}
\end{tikzcd}$$

This is a nice start, but doesn't help us with our initial problem of saving space, since every node has to be created for us to be able to figure out where we are in the matrix. If we didn't have a D node, we wouldn't know if the item to the right of A was supposed to be in column 1 or 2 or 20. So let's add a set of index nodes that tell us the index of each row or column. Since we want to have a single point of entry into the data structure, we'll also add an extra node with no data, but that can be used as the head of the structure.

$$\begin{tikzcd}
\boxed{\text{None}} \arrow[r] \arrow[d] & \boxed{0} \arrow[r] \arrow[d] & \boxed{1} \arrow[r] \arrow[d] & \boxed{2} \arrow[d] \\
\boxed{0} \arrow[r] \arrow[d] & \boxed{\text{A}} \arrow[r] \arrow[d] & \boxed{\text{D}} \arrow[r] \arrow[d] & \boxed{\text{G}} \arrow[d] \\
\boxed{1} \arrow[r] \arrow[d] & \boxed{\text{B}} \arrow[r] \arrow[d] & \boxed{\text{E}} \arrow[r] \arrow[d] & \boxed{\text{H}} \arrow[d] \\
\boxed{2} \arrow[r] & \boxed{\text{C}} \arrow[r] & \boxed{\text{F}} \arrow[r] & \boxed{\text{I}}
\end{tikzcd}$$

So now let's see what happens when we don't have all the nodes. Let's say that our matrix is all 0s, except for A, G and F.

$$\begin{pmatrix} A & 0 & G \\ 0 & 0 & 0 \\ 0 & F & 0 \end{pmatrix}$$

We could create the matrix like this:

$$\begin{tikzcd}
\boxed{\text{None}} \arrow[r] \arrow[d] & \boxed{0} \arrow[r] \arrow[d] & \boxed{1} \arrow[r] \arrow[dd] & \boxed{2} \arrow[d] \\
\boxed{0} \arrow[r] \arrow[d] & \boxed{A} \arrow[rr] & & \boxed{G} \\
\boxed{2} \arrow[rr] & & \boxed{F} &
\end{tikzcd}$$

Notice now that we only need to create nodes for the data that actually exists in our matrix. Any node that doesn't exist, we know must have a value of 0 (or whatever other default value we want to ascribe).

Make sure you fully understand this data structure before moving on. Try drawing out what it would look like with different matrices).

Your Task

Your task is to complete the code in `a1.py` using the linked data structure described above. In particular, you may not use any other Python data structure (e.g., lists, dictionaries) in completing your code. You may add other methods if you wish, but you must at least implement the ADT as provided in the starter code.

Hints & Tips

- Plan everything before you write anything
- Draw lots of pictures
- Think about boundary cases (inserting at the beginning/end of a row/col, empty matrices, etc)
- If you're stuck, you can go with the simpler approach of just creating every node, even those with default values, this won't get you full marks, but it's better than not having any of your code work
- Searching in a 2-dimensional system like this can be very confusing. Finding the intersection of a row and a column is difficult. Maybe there's an easier way? (remember: you can add to the Node class as well).
- Don't forget to include your representation invariants
- Remember that this must be 100% your own work. You may ask for clarification on Piazza, but you should not discuss this assignment with your classmates, or look for help elsewhere
  - If you don't understand something about the matrix mathematics, ask me, and I'll explain it or post a tutorial on Piazza (I don't want to test your linear algebra knowledge).