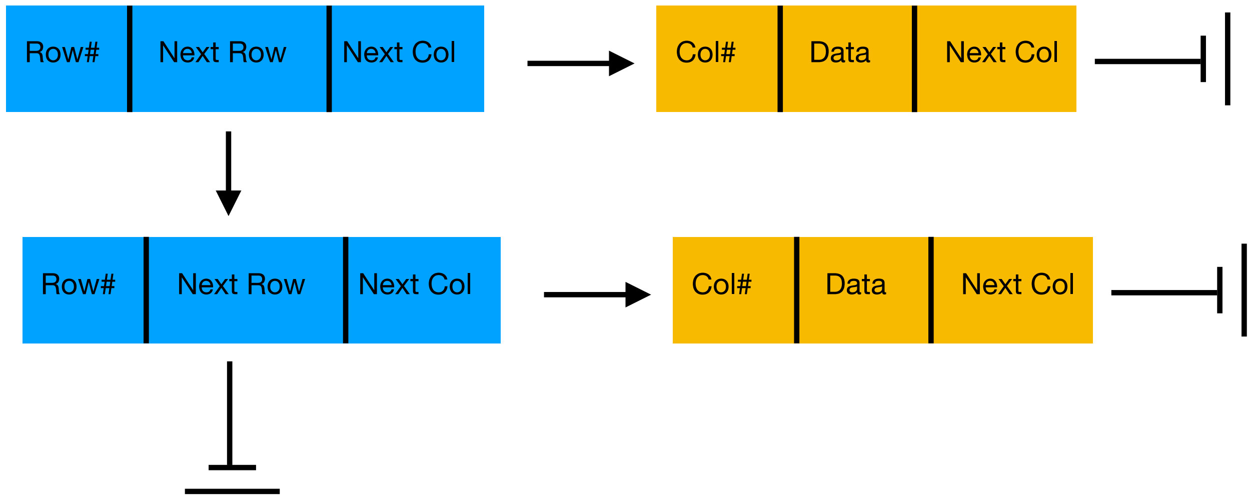
Daniel Castro
Commentary

In my implementation of the sparse matrix I decided to go with an approach similar to a linked list. Row nodes were created which store the row number and point to the next row if there is one, and the next column entry in that row. The column entries were created with column nodes that stored the column number, data to be entered in the matrix, and point to the next column entry in the row if there is one.



(*Image above is a visualization of the Row Nodes and Column Nodes)*

O(1) operations:

clear - clear is O(1) because when the method is called it simply sets the head node to a new node, losing access to all previously added nodes.

setSize - setSize is O(1) because when the method is called it changes the values of numRows and numCols of the sparse matrix.

getNumRows, getNumCols - getNumRows and getNumCols are O(1) because the methods merely return the value of the number of rows and the number of columns in the matrix.

O(N) operations:

addElement, removeElement, getElement - The addElement, removeElement and getElement methods at the worst case would take O(N) because they all contain while loops. addElement's worst case would be if the row of the element to be added was greater than all the rows currently in the matrix. addElement would have to iterate through all the other rows and then iterate through the column entries in that row. removeElement's worst case would be if the element to be removed was the last column entry in the last row. removeElement would need to iterate to the final row and then iterate to the final column entry in that row. getElements's worst case would be if the element to retrieve was the last element in the matrix. getElement would need to iterate through all the rows to get to the final row and then iterate through all the column entries in that row.

toString - toString will always take O(N) because it must iterate through the entire matrix and then append each row number, column number, and data entry to the string that will be returned.

O(N^2) operations:

addMatrices - addMatrices is O(N^2) because at the worst case the matrix may be an N x N matrix and each element will have to be visited. The method contains two nested for loops that iterate through each row and column.

O(N^3) operations:

multiplyMatrices - multiplyMatrices is O(N^3) because at the worst case N elements must be visited in the N x N matrices. The method contains three nested for loops which iterate N times for the N x N matrices.