Johnny Li

COP 3530

Project 1 Redo Recursive Addition and Multiplication of a Sparse Matrix

Commentary

My matrix implementation involves creating a one-dimensional list, using the ‘ArrayList’ function, that will store all the sets of the linked list. Since the matrix is defined to be an nxm matrix, each of the indexes of the list holds a single row of linked list, creating n rows of linked lists that represents the entire rows of the matrix. There will be m nodes in each linked list to represent the number of columns where only nodes with values other than zero exist. Each node of the list contains the data value, the column position, and a pointer to reference the next node. This implementation allows for the creation of an nxm virtual matrix that can ignore the zero values in it in a relatively simple way. The implementation created the linked list on a needed basis thus removing the possible excess links formed from a defined size design which saves memory and runtime spent. The computational complexity of the entire program is O(n3) due to the three-degree nested ‘for’ loop on the matrices multiplication that transposes and transverse the matrices. The computational complexity of the of the operation the setSize is O(n) since it implements a ‘for’ loop to build the array list and linked list in it. Similarly, the getElement and toString operations are also O(n) since they require searching through the linked list for the data value. The operation removeElement is O(n) since it must search through the entire linked list in order to find the link to be removed. The operation addElement is O(1) as it simply adds a node at the end of the list. The operation getNumRows and getNumCols is also O(1) as it purely returns the size of the list. The clear operation is O(n2) as it has to remove all values in the linked list, the columns, for all rows thus require a two-degree nested ‘for’ loop. The SparseInterface addMatrices operation is an O(n2) as it has transverse all the rows and columns, thus require a two-degree nested ‘for’ loop, for both matrices and add the values in them. The SparseInterface multiplyMatrices operation is an O(n3) as it implements a three-degree nested ‘for’ loop on the matrices multiplication that transposes and transverse the matrices to multiple the values and sum the products. As a note, the best case complexity for many of the operations is when the parameters of matrices are invalid, therefore, resulting in a complexity of O(1) as they fail the check. On the other hand, the worst case complexity for many of the operations is when they must run through the entire matrices.