Johnny Li

COP 3530

Project 1 Recursive Determinant 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 a 5x5 matrix then each of the indexes of the list holds a single row of linked list, creating five rows of linked list that represents the row of the matrix. For each node of the list contain the data value, the column position, and a pointer to reference the next node. This implementation allows for the creation of a 5x5 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 preset design which saves memory and runtime spent. The computational complexity of the entire program is O(n!) due to the recursive nature of the program. The computational complexity of the of the operations includes the setSize as Ω (n) since it implements a for loop to build the list and linked list in it. The operation removeElement is Ω (n) since it must search through the entire linked list in order to find the link to be removed. The operation addElement is Ω(1) as it simply adds a node at the end of the list. The operation getSize is also Ω(1) as it purely returns the size of the list as well as clear where it takes the head node equal to null. SparseInterface minor and determinant operations are Ω (n) as there is a repetitive loop that to solve the minor matrix and get its determinant.