Ricardo Golac Barriga

COP3530 Data Structures & Algorithms

**Project 1 Documentation**

**Why did you choose the implementation you did?**

I decided to store the matrix in a linked list that held the value, row, and column of each element. I only added nonzero elements. I then checked whether the matrix was a square, using counters altered when reading in the data, and threw an error if a square matrix was not entered. Using a single linked list seemed like the best option because I didn’t want to keep track of multiple lists and using other data structures didn’t make sense.

In determining the matrix, I wanted to get down the matrix to a 2x2 base case because its simple do solve arithmetically. So, anything bigger than a 2x2 matrix will recursively call the determinant function until 2x2 matrices are called.

**What did you learn from doing this assignment?**

I learned about the benefits of a sparse matrix memory wise. It wasn’t as difficult to implement as I thought it would be, and I learned about dealing with nonexistent values using a variety of counters and other tricks.

**What is the computational complexity of the operations in your matrix implementation?**

Reading in the data is dependent on the number of elements in the matrix so its complexity is O(n). Checking whether or not the matrix is a square is O(1) because it’s a simple if statement that checks some math with counters created while reading in the elements. Determining the determinant for a null matrix or 1x1 matrix has a complexity of O(1) because it either returns a zero or the only value in the array.

The complexity of finding the determinant of a 2x2 matrix takes O(n) and O(1) per each nonzero value in the first row in the matrix. O(n) because I employed a search function that looks through the entire list to find a specific value. In total is should take O(4n + 4) which simplifies down to O(n) for a 2x2 matrix. For anything bigger each row would recursively find get the minor until I find a 2x2 submatrix. So, for any square matrix it would take O(n^(s-1)) where s is the size of the total rows of the matrix.

**What was the hardest part of this assignment?**

The hardest part initially was getting my head around the sparse matrix idea conceptually. It took me a while to figure out how to do the determinant calculations without storing zero values, but I realized that using some counters and other programming tricks made it easy. After that, making the minor was a bit of work; making the new smaller matrix was easy but it took me a while to make sure I changed the row and column values of each node correctly.