**ANALYSIS OF THE DIFFERENT DATA STRUCTURES:**

**Name:** Spencer Davis

**Class:** Data Structures

**Professor:** Dr. Rouse

**Date:** 09-15-2018

=====================================================================

|  |  |
| --- | --- |
| **Normal Array**: | When I ran this on my PC the standard array was the fastest to input all the numbers in to the array. I think this data structure ran the fastest is because all the indexes are next to each other in memory. Also, the array uses indexing, which is something that none of the other data structures used in this test have. This is the data structure I would use for this task. |
| **Array List:** | The array list was the second fastest when run on my PC. I think this was slower than the standard array because it had to continue to declare new arrays and move the data over. This would be a good data structure for this task if you didn’t know how many numbers you were taking in from a file. |
| **Linked List:** | The linked list was the third fastest. The difference in speed from the standard array to the linked list was less than a second. That being said, I wouldn’t use this data structure for this task. If you were trying to do something with the numbers after you take them in from the file, this wouldn’t be practical with the sheer volume of data that was taken in. Finding different spots in memory, and pointing each node to the next node was the downfall of the speed in this data type. |
| **Backward Array List:** | This was TERRIBLE. Running on my PC this was 927 times slower than the fastest data structure. I am guessing this is attributed to the fact that every time an element is added to the list the entire list has to be shifted. I would never use this data structure. |