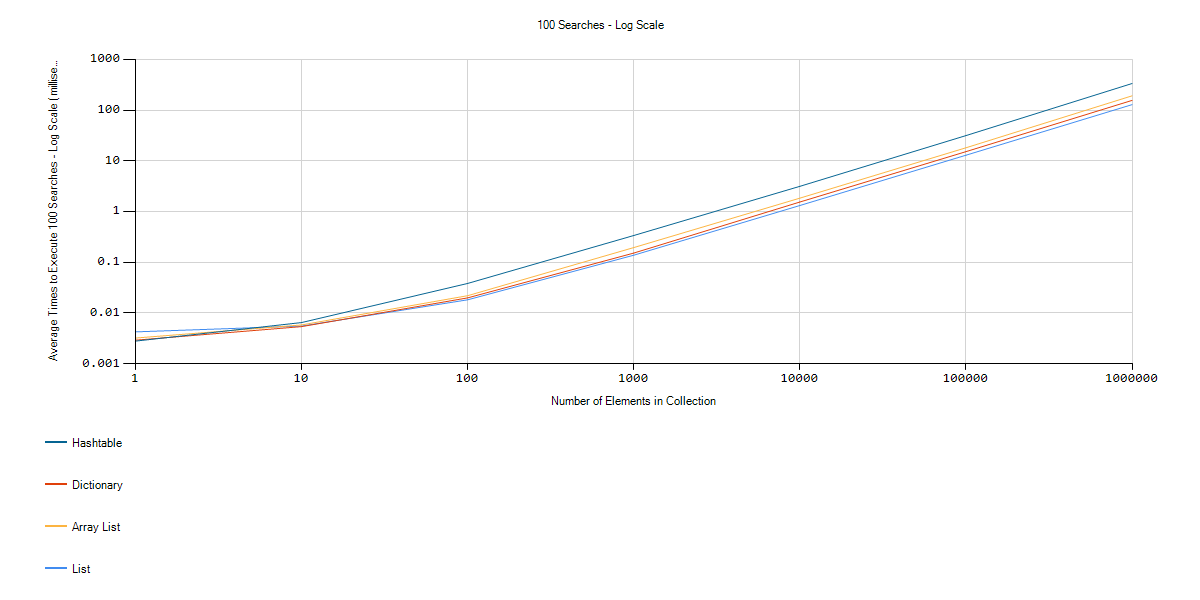
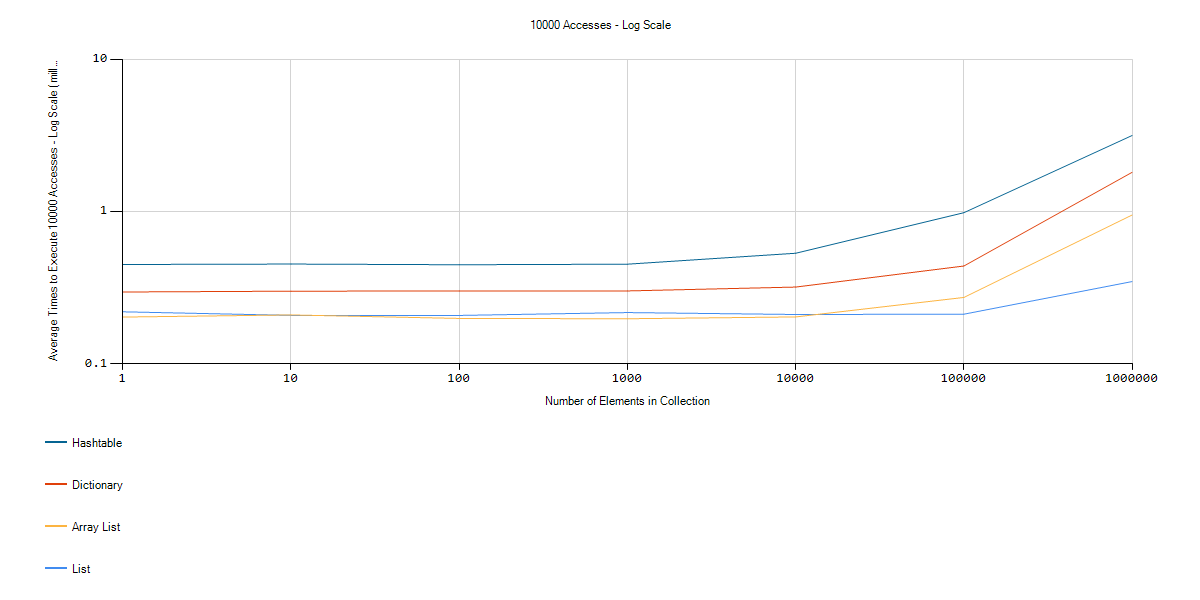
Based off the results that I gathered, I am quite confident making the following statements:

1. A list was the fastest for both searching and accessing, where as a hashtable was the slowest in both cases.
2. The difference in access speed across the four containers became more profound as the size of the collection increased
3. All four container classes had consistent search times, relative to the size of the collection when the search was conducted.

This data partially aligns with what I would have expected going in. Based on research I did before the assignment, I would have expected List and Dictionary to be the fastest, as they both had strictly defined data types, and then I would have expected hashtable and arraylist to be the slower of the four, as they are both non restricted data collection types. As best as I could tell, the act of having to cast objects back and forth between object types and integer types (boxing and unboxing) should have caused the hashtable and arraylist to be the slowest collection types. This was the case for searching, but for direct access, dictionary was slower than the array list.

**Graphed Results**



The results are not easy to see in this format, I have attached the images along with the document. To achieve the above results, I performed the following tests:

1. All four collections were filled with numbers, increasing from zero up to the size of that particular test, 1 to 100000.
2. On each collection type, I performed 10000 direct accesses, and 100 searches.
3. I performed each set of tests in point two 100 times, and then took the averages of all 100 times.
4. I performed the above combination of tests on each collection type for seven different sizes, ranging from 1 to 1000000.

The accesses method for each collection type was the contains method for search, and a direct index access for the access test. Not all four collections contained the same set of functions, so in each case, I looked at the MSDN page for that collection to find the most similar function.