Sort Method Analysis

This assignment actually blew my initial expectations. I fully expected “Quick Sort” to be the quickest, just based upon its name, which turned out to be quite the fallacy, since it turned out to be the slowest method for this many simple records. For my testing, I uploaded a file with 10,000 unique numbers into the array. The output was as follows, from quickest to slowest:

Sort Method	Duration (in milliseconds)
Insertion Sort	.000963
Merge Sort	.000964
Bubble Sort	.094001
Selection Sort	.104034
Quick Sort	.110038

I was also surprised to see that the difference in time was negligible between Insertion and Merge Sorts, even though the Merge Sort seems to go through more processes per the required code. I also ran my tests on a system with a large amount of RAM and solid state hard drive, so the timing of the Merge Sort might actually be a lot slower than my test results, depending on the system in which the task is performed.

While this test used 10,000 items for sorting, it does not provide for a solid decision as to the best sorting method. Clearly, the best sorting method would change based on the amount of items and memory utilization of the overall program. So, as a best practice, it might be a good idea to keep these methods in a code library so that they can easily be implemented into a program and run as a test in order to select the best one. Of course, it would be best to test this prior to developing the remainder of the code.

In summary, it is advisable to become familiar with these different sort methods and their overall behavior on a variety of data and programs in order to recommend the best method during a planning session.