This is one sample output of my program:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vector Size | Qsort | MergeSort | HeapSort | InsertionSort |
| 1,000 | 17,263 | 6,519 | 6,834 | 248,625 |
| 10,000 | 233,124 | 90,023 | 86,718 | 25,054,123 |
| 20,000 | 487,184 | 190,365 | 172,385 | 99,892,073 |

The results that I get are clear as to what complexity each algorithm has. The Quick Sort algorithm is a little slower in comparison to Merge Sort and Heap Sort, but that is acceptable since Quick Sort has a worst case complexity of n^2, and average and best case of n \* log n, and given the random numbers that I fill my vectors with the algorithm did not perform so well with them. On the other hand, the Merge Sort and Heap Sort algorithm do very well at a n \* log n complexity throughout the whole time, although the Heap Sort algorithm does a little bit better than the Merge Sort algorithm with larger numbers. Lastly, Insertion Sort performs worst out of all the algorithms having almost 100 million comparisons for 20,000 values, because of the slowness of Insertion Sort the running of my program took almost 30 minutes to be able to perform Insertion Sort on a vector of 20,000 values.