**Merge Sort Report**

CS6301.g42

David Tan, Khaled Al-naami, Peter Farago, Yu Lin

The running times for different data sizes and algorithms are shown below. These running times represent averages taken over several runs of each algorithm for each data size.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Alg.***  ***Size*** | Int Type ***MergeSort***  (milliseconds) | Generic Type ***MergeSort*** (milliseconds) | Generic Type ***InsertionSort*** (milliseconds) |
| 1 M | 183 | 454 | 1951828 |
| 5 M | 872 | 2309 | n/a |
| 9 M | 1654 | 4326 | n/a |
| 13 M | 2380 | 6816 | n/a |
| 16 M | 2916 | 7683 | n/a |

Table 1 Average Running Time of MergeSort and InsertionSort

Now we show how much data each algorithm can handle, given specific time limits:

|  |  |  |  |
| --- | --- | --- | --- |
| ***Alg.***  ***Time*** | Int Type ***MergeSort***  (M = Million) | Generic Type ***MergeSort***  (M = Million) | Generic Type ***InsertionSort***  (K = Thousand) |
| 100ms | About 0.6M | About 0.1M | About 7k |
| 200ms | About 1.2M | About 0.3M | About 11k |
| 300ms | About 1.8M | About 0.7M | About 14K |
| 400ms | About 2.4M | About 0.85M | About 16K |
| 600ms | About 3.5M | About 1.35M | About 19k |

Table 2 Capable Data Size of each algorithm given specific time limits

Obviously, according to the Table 1 and Table 2, the insertion sort, as an algorithm, is much slower than merge sort whose asymptotic running time is . The time consumed by insertion sort is not measurable after the data size becomes larger than 5 million elements. Even with 5 million elements, insertion sort took over 6 hours to execute.

There is another more interesting point. From a quick glance we can see that the merge sort for int arrays is at least twice as fast as the merge sort for generic types. This is expected since the merge sort for generic types has to compare objects by repeatedly calling the function. This takes much more time than a single comparison between two ints.