Problem 5.1 (d)

Using: ***for(int*** *i = 0; i < 1000; i++);*

Lomuto Partition Algorithm elapsed time: 4094 milliseconds

Hoare Partition Algorithm elapsed time: 29 milliseconds

Median-Of-Three Algorithm elapsed time: 35 milliseconds

Using: ***for(int*** *i = 0; i < 10000; i++);*

Lomuto Partition Algorithm elapsed time: 40590 milliseconds

Hoare Partition Algorithm elapsed time: 297 milliseconds

Median-Of-Three Algorithm elapsed time: 341 milliseconds

Using: ***for(int*** *i = 0; i < 100000; i++);*

Lomuto Partition Algorithm elapsed time: 480272 milliseconds

Hoare Partition Algorithm elapsed time: 3511 milliseconds

Median-Of-Three Algorithm elapsed time: 3962 milliseconds

In conclusion:

Hoare-Partition Algorithm is more efficient than Lomuto-Partition because it does three times fewer swaps on average. It also creates an efficient partition even when all the values are equal. Still the wors-case (when all elements are already sorted or reversed) for both, Lomuto & Hoare – Partition is $O(n^2)$ and $O(nlg(n))$ for best-case. Considering this (worst-case), the Median-Of-Three Partition is more efficient. It gives a better estimate of the true median. The expected number of comparisons needed to sort size *S* elements is 1.188 $nlg(n)$ (Found this value in Wikipedia), which is faster than a random number generator.

Work: Albrit Bendo