7.1.  
 d) Lomoto Partition: 8.03221e-05  
 Hoare Partition: 5.73433e-05  
 Median Of Three Partition: 7.89144e-05

The best partition is the Hoare one, because it makes the least number of swaps, so performs faster than the others. Also Lomoto Partition is slightly behind the Median Of Three, because in Lomoto we may accidentally pick the lowest number as a pivot and the division of array will not be efficient.

7.2

The time complexity of QuickSort\_Modified algorithm depends on the partition scheme and the selection of pivot elements. It selects two pivot elements instead of one, and divides the array into three parts instead of two. The first part contains elements smaller than the first pivot, the second part contains elements between the two pivots, and the third part contains elements larger than the second pivot. The selection of pivot elements in this implementation is not optimal. It always selects the first two elements of the array as pivots, and swaps them if they are not in the correct order. This can lead to worst-case behavior if the array is already sorted or reverse sorted, as it will always select the smallest and largest elements as pivots. In this case, the algorithm will perform n-1 comparisons for each recursive call, leading to O(n^2) time complexity. On average this implementation has an expected time complexity of O(n\*log(n)). However, this depends on the selection of pivot elements, and the distribution of input elements. If the pivot elements divide the array into roughly equal parts, the algorithm will have a good performance. If the pivot elements divide the array into highly imbalanced parts, the algorithm will have a poor performance. Therefore, the best case of this QuickSort\_Modified algorithm occurs when the pivot elements divide the array into equal parts, and the worst case occurs when the array is already sorted or reverse sorted. The average case performance depends on the distribution of input elements, and the selection of pivot elements.