Experiments

1). Plotted the run time of the Insertion sort algorithm and the Merge sort algorithm on arrays of integers.

Machine used: Intel Core i3 5005U 2 GHz Processor Dell Insprion 15 3000 Series
Operating System - Ubuntu 17.04

Case 1 - Analysis when array is not already sorted.

Input Size - 10
Time taken by insertionSort: 6 microseconds
Time taken by mergeSort: 9 microseconds

Input Size - 100
Time taken by insertionSort: 34 microseconds
Time taken by mergeSort: 33 microseconds

Input Size - 146
Time taken by insertionSort: 82 microseconds
Time taken by mergeSort: 82 microseconds

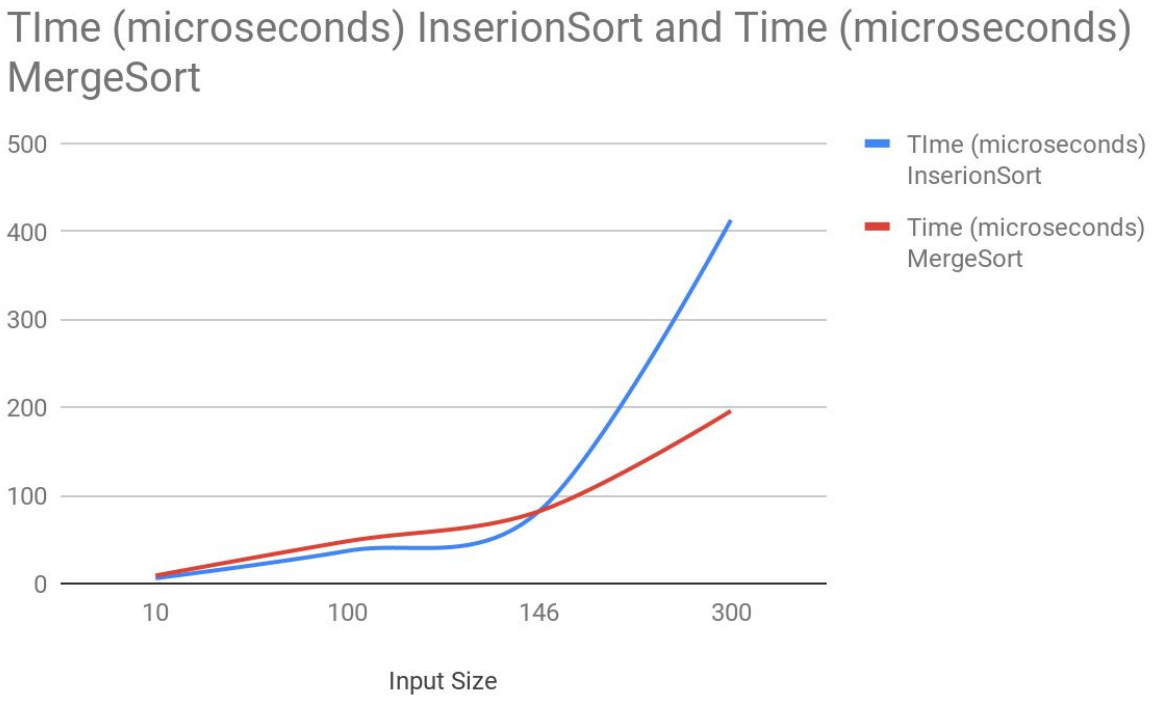
Input Size - 300
Time taken by insertionSort: 413 microseconds
Time taken by mergeSort: 196 microseconds

Input Size - 1000
Time taken by insertionSort: 2805 microseconds
Time taken by mergeSort: 533 microseconds

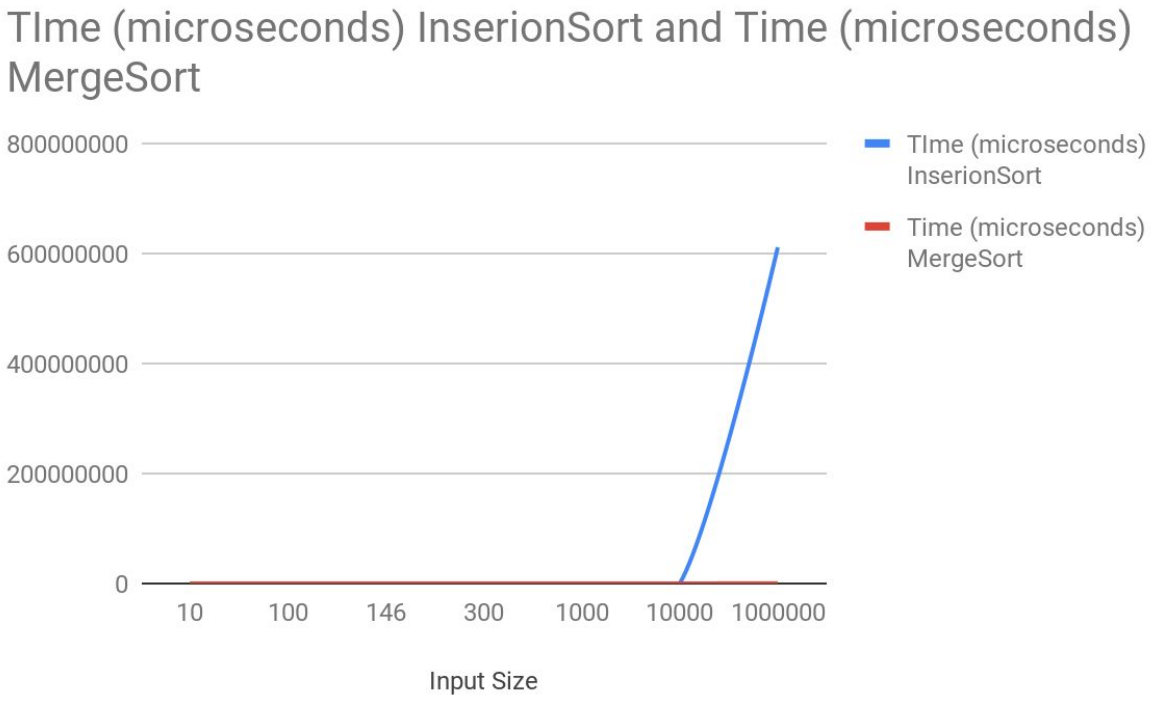
Input Size - 10000
Time taken by insertionSort: 67973 microseconds
Time taken by mergeSort: 1469 microseconds

Input Size - 1000000
Time taken by insertionSort: 611158321 microseconds
Time taken by mergeSort: 118826 microseconds

In the below chart graph is plotted for input sizes of 10,100, 146 and 300. For input size greater than 146, the merge Sort starts performing better than insertion Sort. So for input size greater than 146 we should prefer Merge Sort over Insertion Sort.

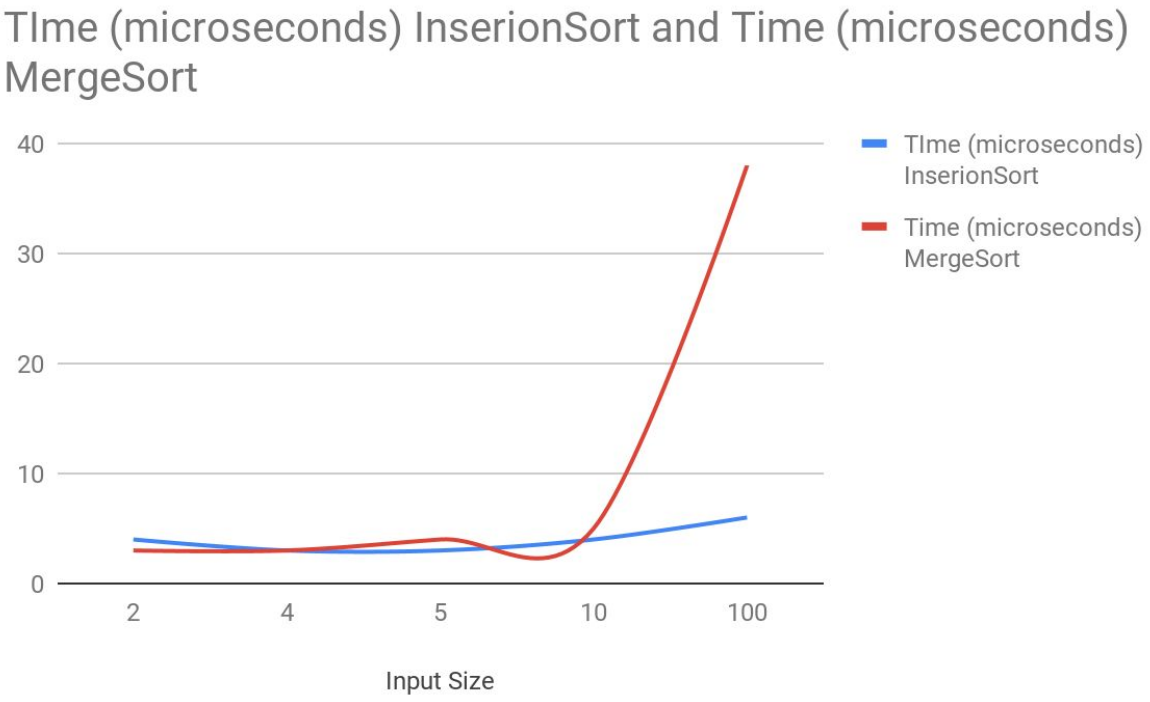


The below graph is for input sizes from 10 to 10,000 and it reveals how the time taken by insertion sort increases sharply after a particular input size. Thus, for larger inputs merge sort is preferred.



Case 2 - Analysis when array is already sorted.

In the below chart graph is plotted for input sizes of 2, 4, 5, 10, 100 and 1000. For input size greater than 10, the insertion Sort starts performing better than merge Sort if the array is already sorted.

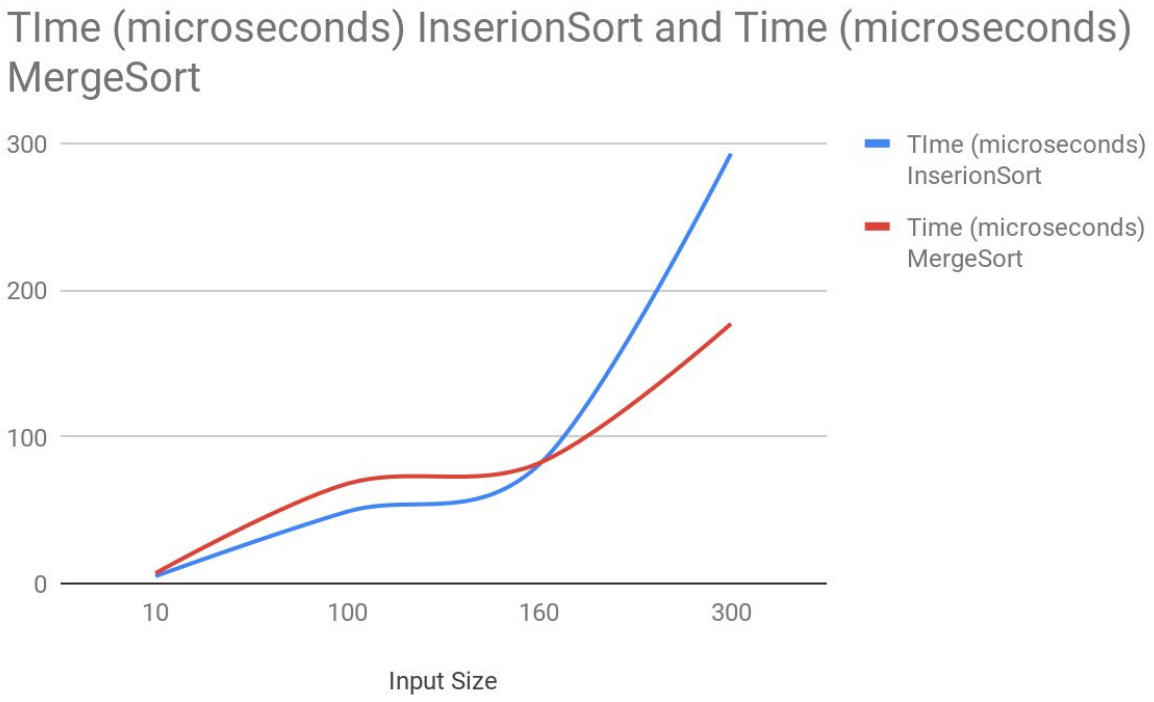


2). Plotted the run time of maximum finding on arrays of integers.

Machine used: Intel Core i3 5005U 2 GHz Processor Dell Inspiron 15 3000 Series
Operating System - Ubuntu 17.04

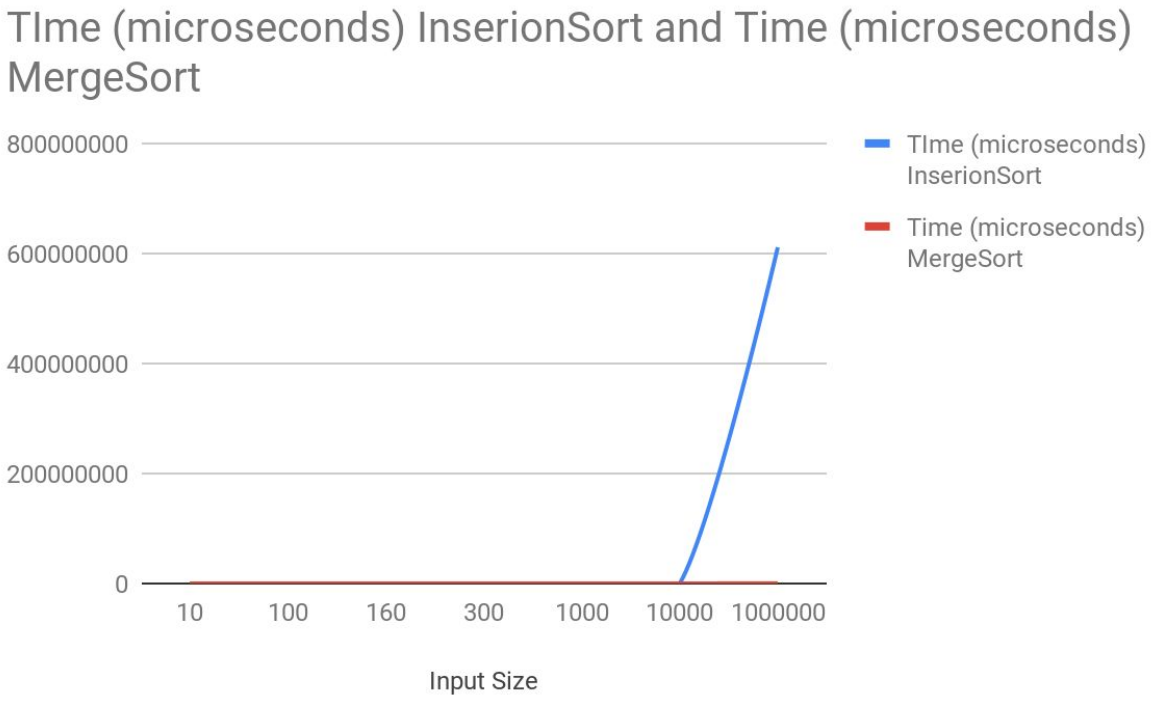
Since after sorting, selection takes constant amount of time. Therefore the time taken for sorting the arrays in increasing order is approximately equal to the time taken to select maximum (which includes time of sorting the array in increasing order).

Input Size	TIme (microseconds) InserionSort	Time (microseconds) MergeSort
10	5	7
100	49	68
160	81	82
300	293	177
1000	2815	543
10000	69500	1479



In the below chart graph is plotted for input sizes of 10,100, 160 and 300. For input size greater than 160, the merge Sort starts performing better than insertion Sort. So for input size greater than 160 we should prefer Merge Sort over Insertion Sort.

The below graph is for input sizes from 10 to 10,000 and it reveals how the insertion sort time increases sharply after a particular input size. Thus, for larger inputs merge sort is preferred.



Conclusion

For larger inputs merge sort is better than insertion sort if the array is not sorted already. The insertion sort has a best case when array is already sorted and in this case it takes linear amount of time to sort whereas insertion sort takes O(nlogn) time. In average cases insertion sort takes O(n^2) time and takes much more time than merge sort which takes O(nlogn) time.