Analysis: Time Complexity of Sorting Algorithms

Abstract:

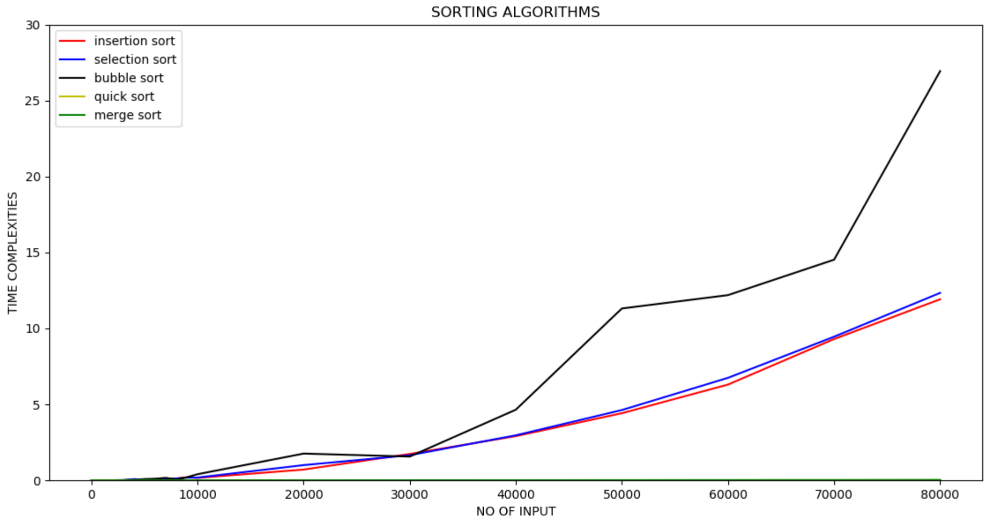
In this Assignment, the time complexities of different sorting algorithms like insertion sort, selection sort, bubble sort (modified), quick sort, and merge sort are calculated and a comparison is made between them.

About the Program:

In the Code the elements of the array are generated at random and are imputed to all the sorting algorithms through a file. The Algorithm the sorts the elements .The execution time of each algorithm is calculated using the clock () [ctime.h library] function, by calling it be before and after the algorithm and the finding its difference .The output of the program is again saved back to the file. (I used cpp for the code)

About the Graph:

The graph is plotted using Python. The time Complexities were plotted against the no of elements the algorithm sorted. And the following Graph was obtained.



Analysis of Sorting Algorithms.

Observation:

From the above graph we can see that the time complexity of bubble sort increases more rapidly than selection sort and insertion sort .As the no of elements increases the time complexity also increases exponentially. Selection sort and insertion sort work fairly similar .Their time complexities increases exponentially but less than bubble sort. We can see that in the range between n=0 to 30,000 the three algorithms work equally well ,but after 30,000 bubble sort increases more rapidly than the other two.

Now we come to the Quick sort and the merge sort their time complexities are very close to zero. With increase in no of elements the time complexity does not increases appreciably.

Theoretically:

Algorithm	Worst case time complexities
Insertion sort	Big O(n*n)
Selection sort	Big O(n*n)
Bubble sort	Big O(n*n)
Quick sort	nlogn
Merge sort	nlogn

Result:

The best sorting Algorithms are Quick Sort and Merge sort.

When the no of elements are large .Other algorithms are fairly good when no of elements are less.