HACETTEPE UNIVERSITY

Department of Computer Engineering

SPRING 2019

BBM 204 PROGRAMMING LAB. ASSIGNMENT 1 REPORT

Authors:	Burcak Asal
	Merve Ozdes
	Alaettin Ucan
Name and Surname:	Ege Berke Balseven
Number:	21590776
Subject:	Analysis of Algorithms
Due Date:	25.03.2019

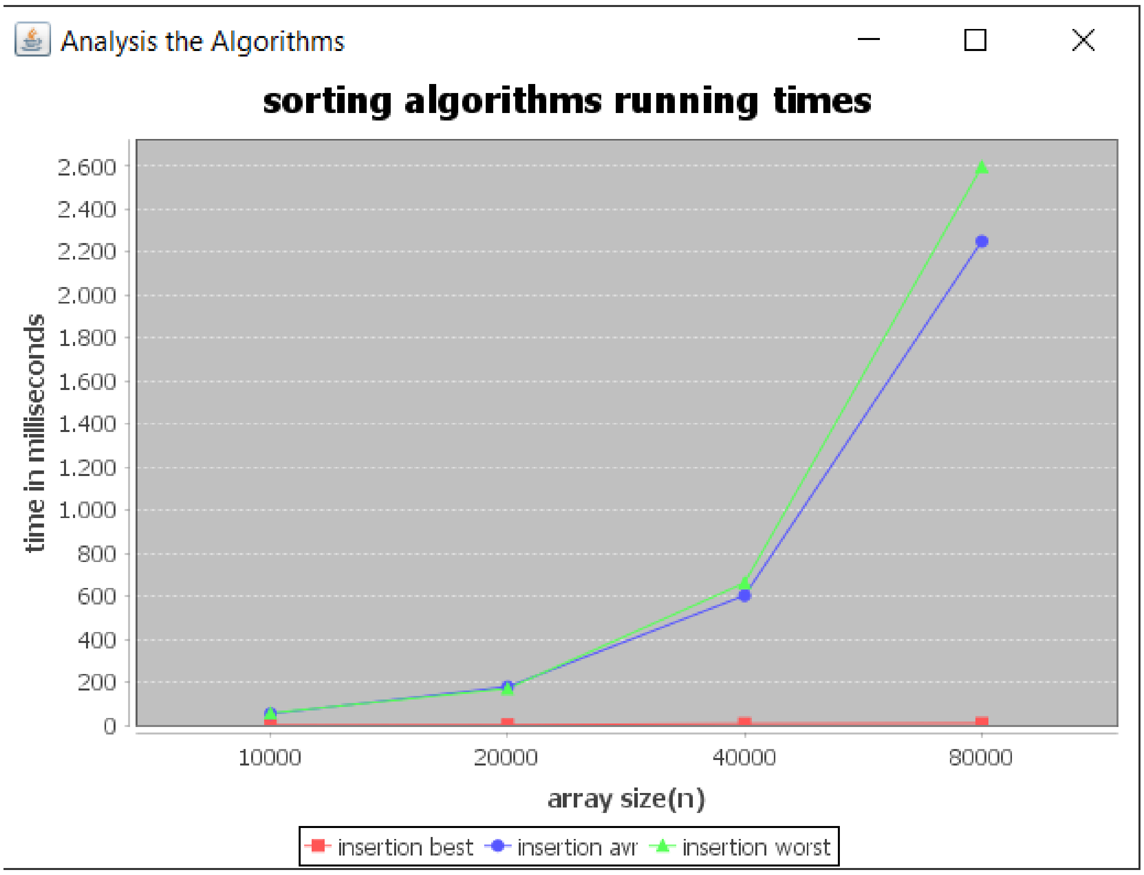
1 PROBLEM DEFINITION

Searching for specific data in a data repository is a feature that is present in almost all software systems today. To search for the data, sorting is one of the most effective methods. But there are many different methods, algorithms, to do this. There are many different effects to sorting the data such as the type of the data, the size of the data, and the way the data is stored in memory. The objective of this report is to analyze and compare the performance of five sorting algorithms organizing an array of integers in ascending, descending or random order. Ascending order corresponds the best case, descending order corresponds the worst case and random order corresponds the average case. The five algorithms used in this analysis were insertion sort ,merge sort, radix sort, selection sort and binary search algorithms. The algorithms sorted the same order arrays containing 10000, 20000, 40000 and 80000 integers.

2 FINDINGS

I only showed in the report the graphics printed in the code, because the extra library was giving the error in the dev.cs.hacettepe. It needed to be loaded. That's why I submitted the parts that are related to the graphics as a comment line. However, when run, output.txt will be displayed instead of graphics.
Time Complexity symbols:
Best $= \Omega(), Average = \Theta(), Worst = O()$

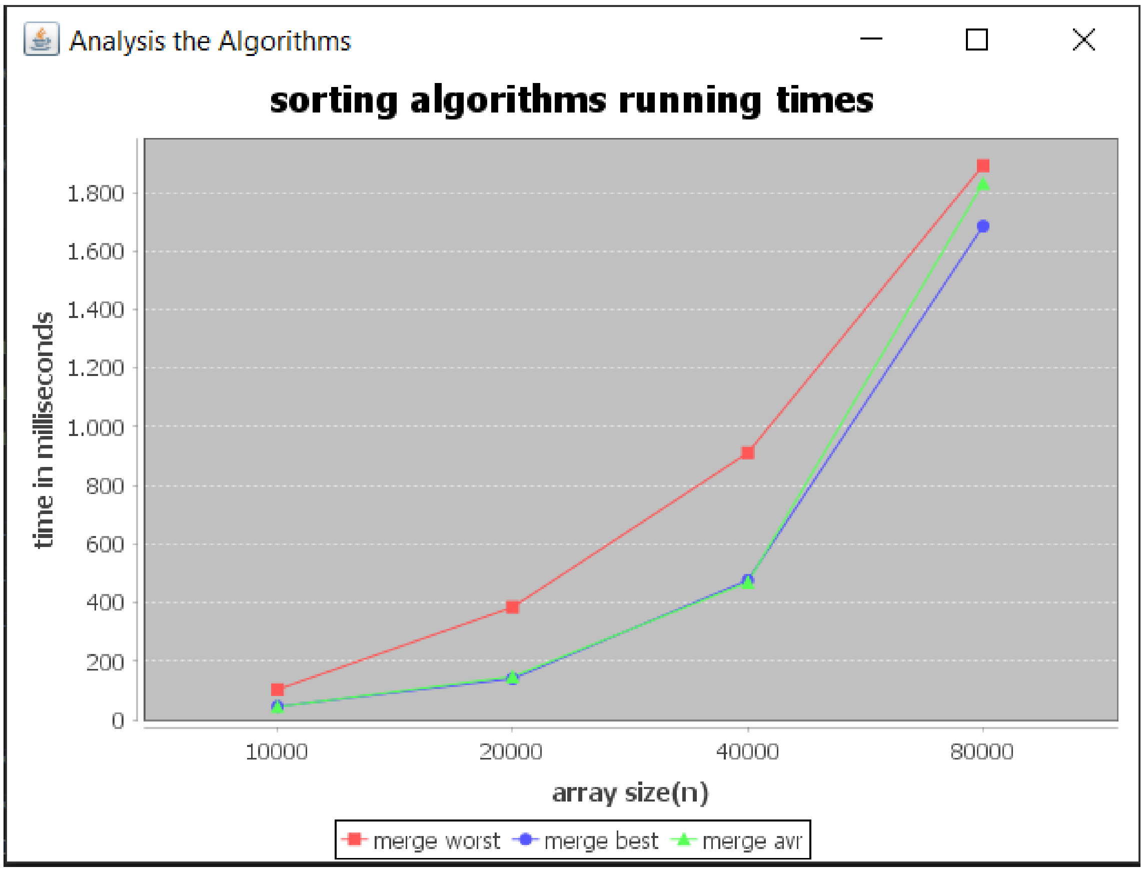
2.1 Insertion sort algorithm



$\Omega(n), \Theta(n \wedge 2), O(n \wedge 2)$

Insertion sort gives expected results in the best and worst cases. Because best case has O(n) time complexity. Best and average cases are increases quadratic. Perhaps due to the ssd and processing power on the computer the values in the O(n) time complexities are little low.

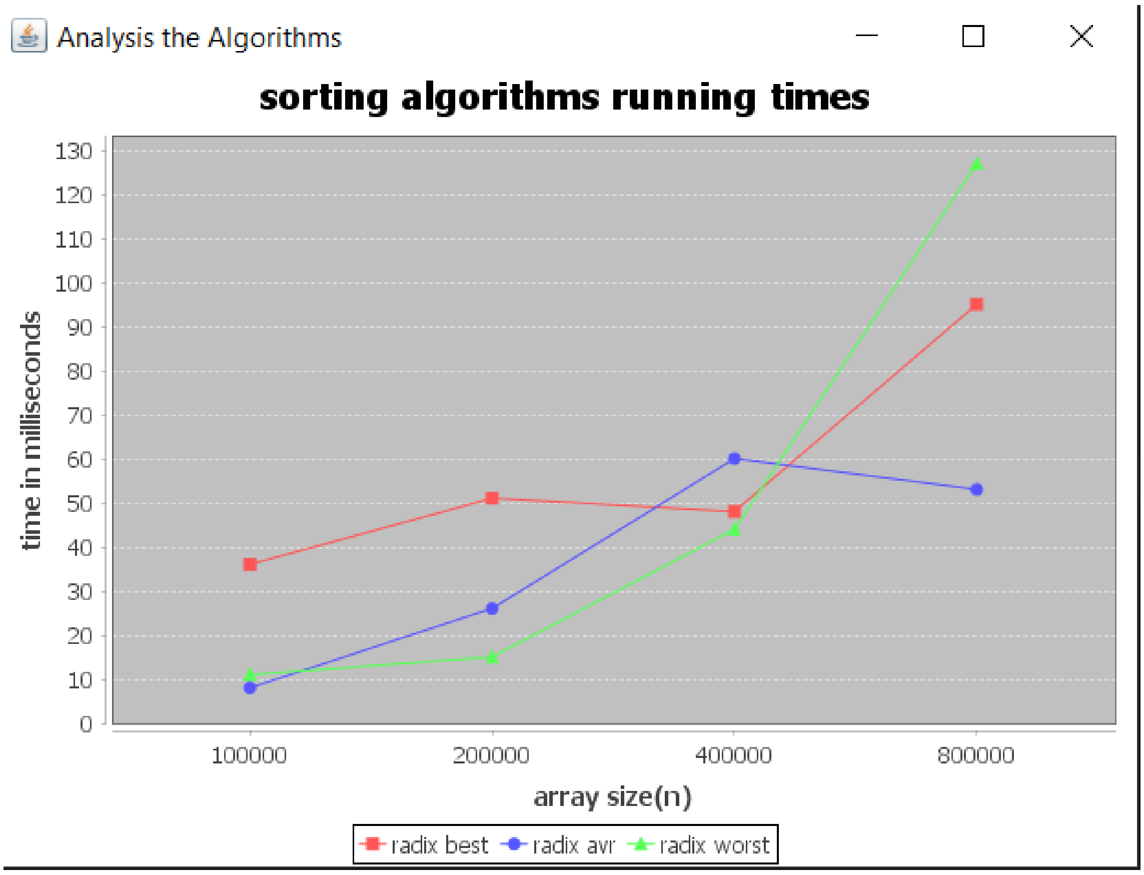
2.2 Merge sort algorithm



$\Omega(nlog(n)), \Theta(nlog(n)), O(nlog(n))$

The results in the merge sort may look like O(n2), but when looked carefully, the values does not increase as quadratic like insertion sort and the results are not high as insertion sort. Also the results are close to each other and 0(nlogn) for all cases.

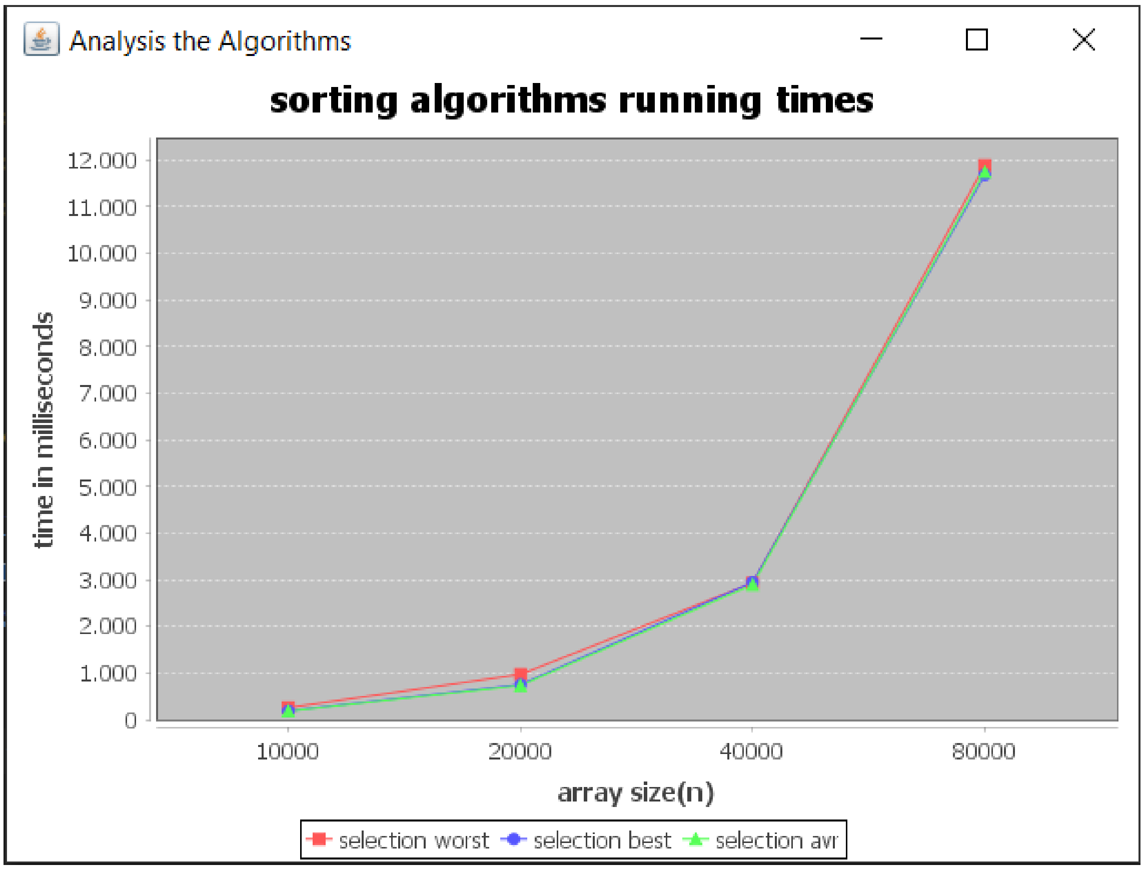
2.3 Radix sort algorithm



$\Omega(nk), \Theta(nk), O(nk)$

In radix sort algoritm, results are not much satisfactory. It seems little complicated. But expected time complexity values are seems true. Values are little low and increases as line because of the O(nk).

2.4 Selection sort algorithm

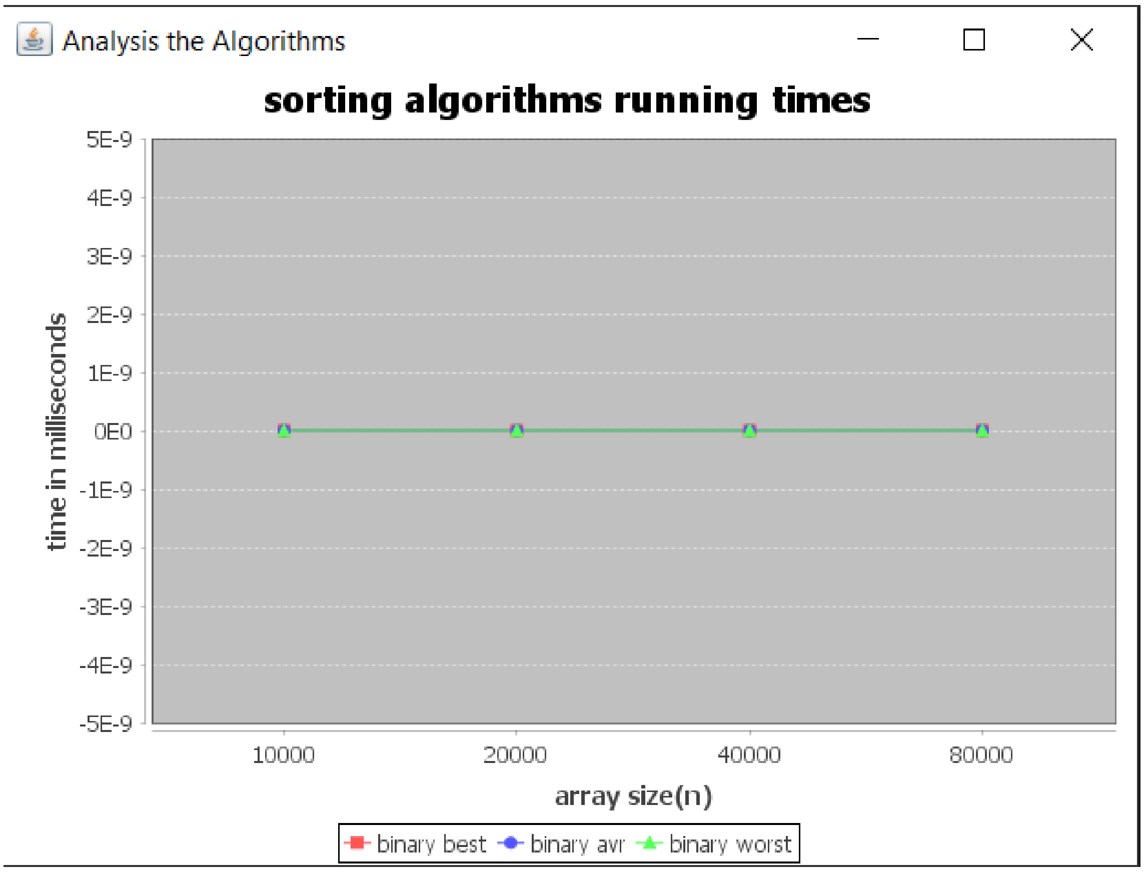


$\Omega(n \wedge 2), \Theta(n \wedge 2), O(n \wedge 2)$

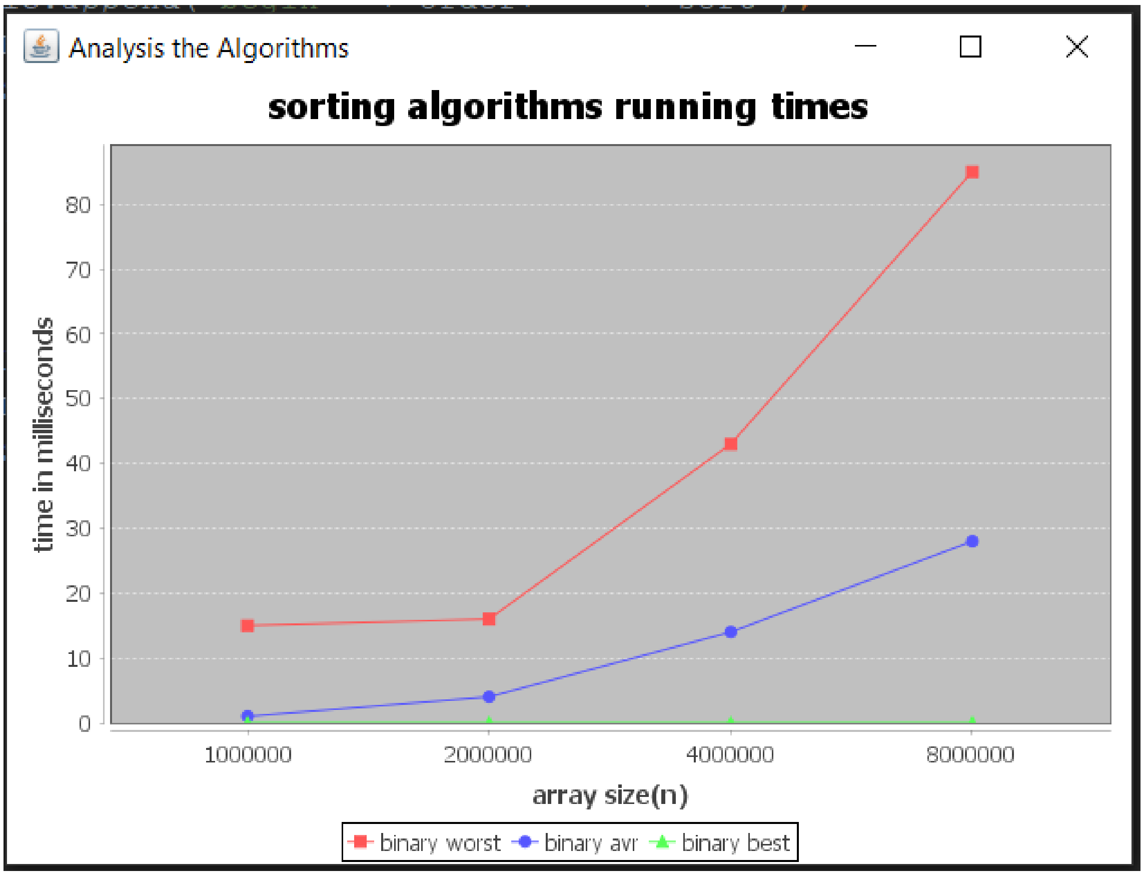
In selection sort we see expected results. They hit around the expected O(n2) times.

2.5 Binary Search algorithm

$\Omega(1), \Theta(log(n)), O(log(n))$



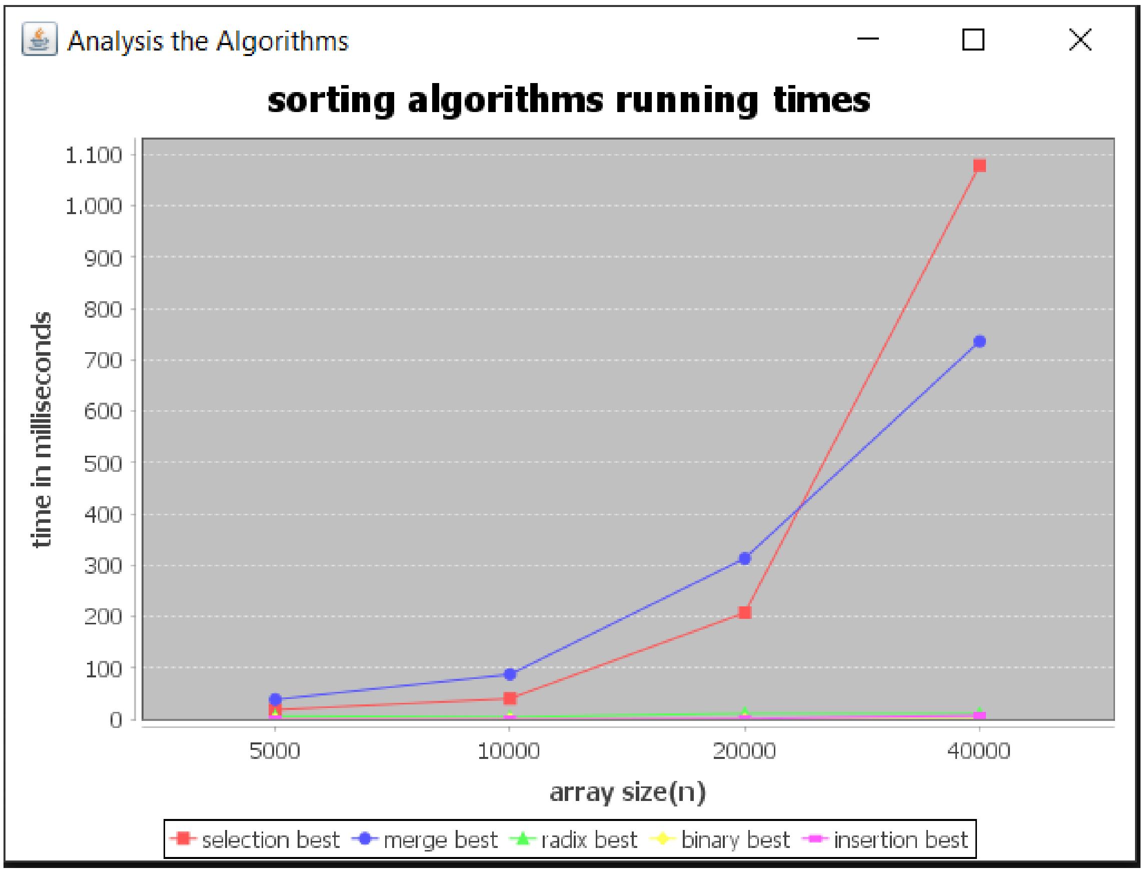
If i use the same array i used in other algorithms, it is normal for binary search to have zero time results. Because the array size was not too high to prevent other algorithms from working too long.



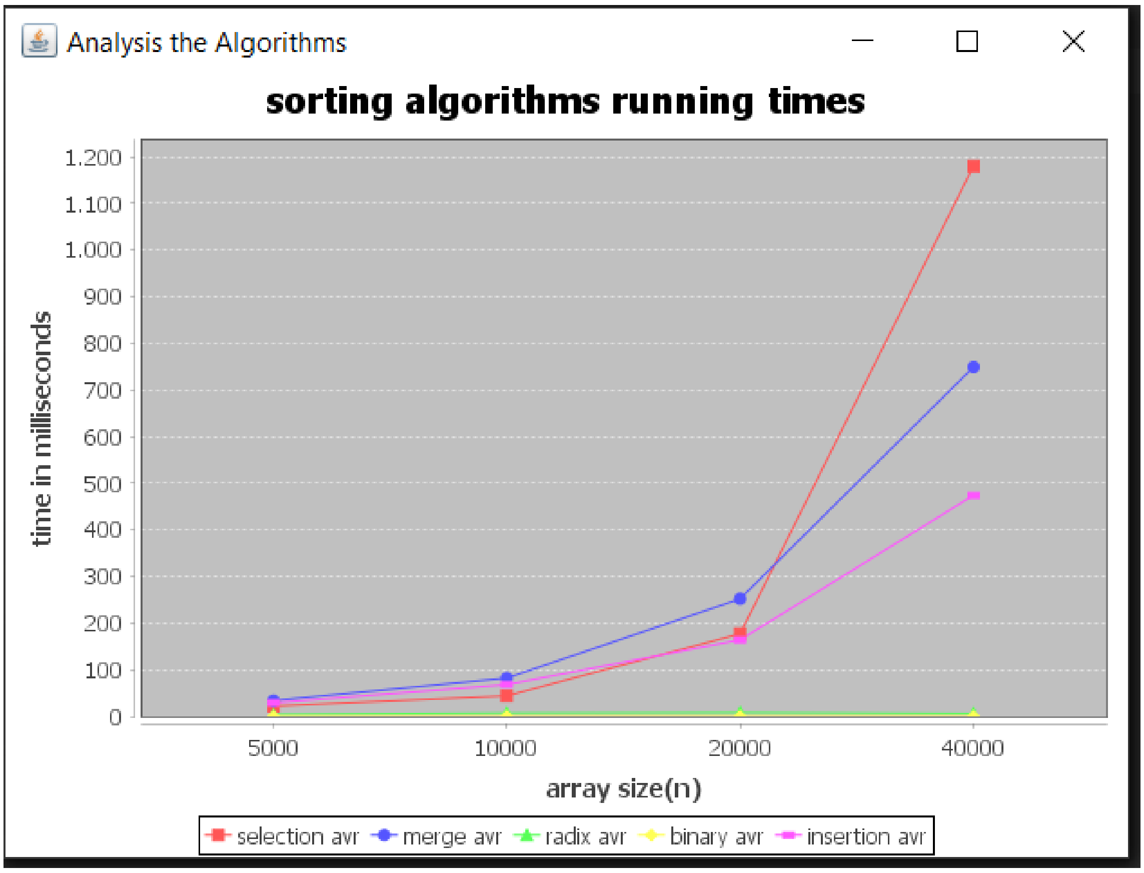
In these graph, i increase the array size to millions. Because time complexity is zero in low size arrays. But at best case, the value is still 0, because the middle element is the best case and it founds directly. In the worst case I chose the last element.

3 DISCUSSION

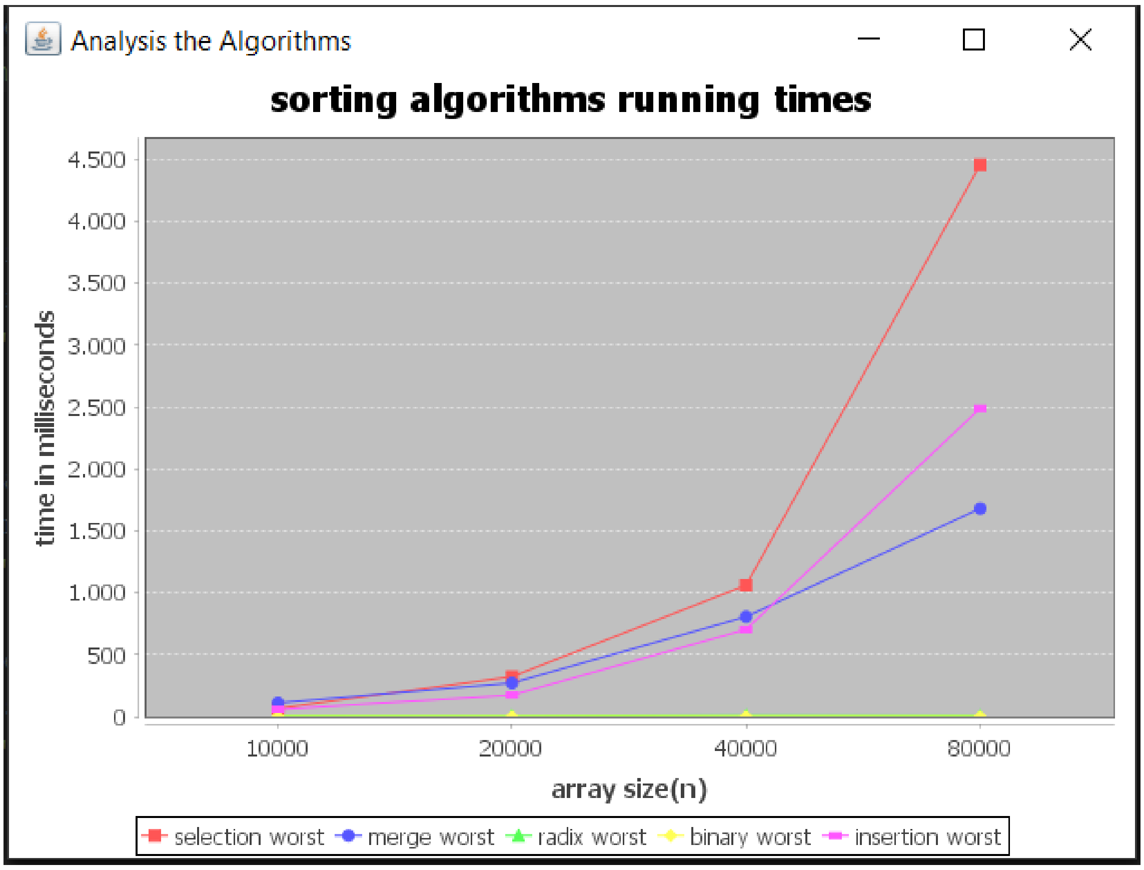
Algorithms behaved generally expected. But the difference in theoretical and experimental time complexity may be due to deviations in time runs and the relatively small size of files tested, further testing and larger inputs will most likely reduce inaccuracies and cause the true time complexity of the algorithm to be more apparent and behave as expected. The other factors, such as memory usage, code reusability, CPU usage, the algorithm's code, etc.
Here are the comparison of all algorithms in the best, worst and average cases. These charts show us the complexity more clearly. For example, the difference between O(n2) and O(n) or O(logn) is clear.



This chart shows best case performances. For example insertion sort does not shows as high value here. But below average and worst cases it shows. Because insertion different in best case. We can easily see the value differences in quadratic and linear time complexities.



This chart shows average case performances.



This chart shows worst case performances. Selection and insertion time complexity O(n2) can be seen clear and merge O(nlogn) also.