**Analysis and Comparison of Sorting Algorithms**

**(Selection,Heap, and Hybrid) Using JavA**

**1.Introduction:**

in this project

**2.Background**

A.Selection Algorithm

B.Heap Algorithm

C.Hybrid Algorithm

**3. METHODOLOGY**

**A. Experimental Setup This section presents the used machine and platform**

**1. Used Machine:**

Windows HP Core i5 & 8GB RAM

**2.Tools Used**

**1. Apache NetBeans :**

to write code to all three algorithms and show results

**2.Excel application:**

to analyze results and drawing charts to results to all algorithm

**B.Data Generation**

We are use three inputs to algorithms first generate random array and order it in sorted and third I reverse in decreasing order.

The array types can be in three orders:

**• Increasing (Best Case):** Use the same array after sorting. This array is considered the best case.

• **Random (Average Case):** Generate an array of unsorted elements using a random method of package "java.util.Random".

• **Decreasing (Worst Case):** Generate a reversed array with decreasing sorted elements.

This figure that shows how to generate size to array that store szies to array that we generate it by randomly.

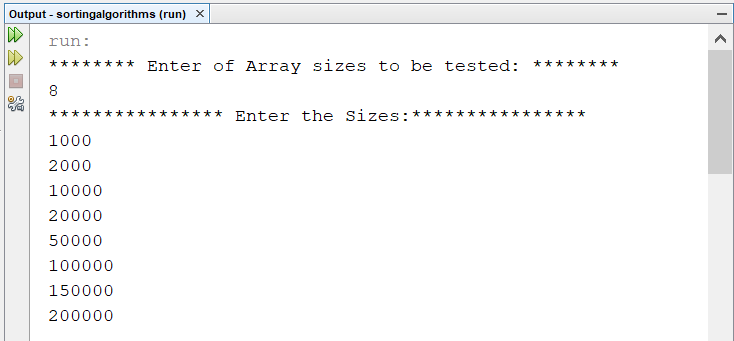


Figure 1: Generate Random Arrays

**C. Timing Mechanism:**

I use by java to calculating running time to all three algorithm

System.nanoTime();

This to calculate when starting and ending execution to algorithm and subtract start from end and result is time to execute this algorithm.

**4.Results**

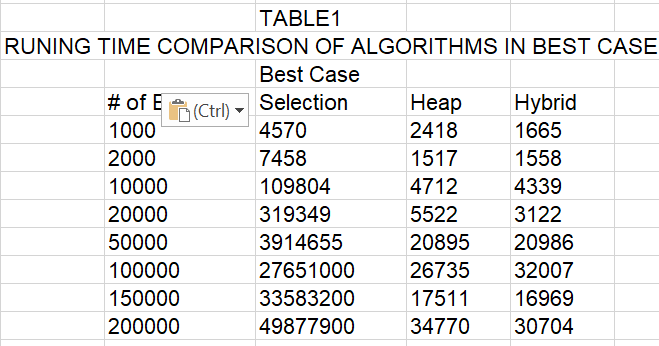
We hav three cases and I want to calculate and analyse results to three algorithms in three cases

**A.Best Case Running Time:**

This table show the results in running time to three algorithms when array is increasing based on the number of elements and the used algorithm.

The results show that as the number of elements increases, the execution time also increases.

However, the Hybrid sort and also Heap as the very similar has the lowest running time in this case.

****

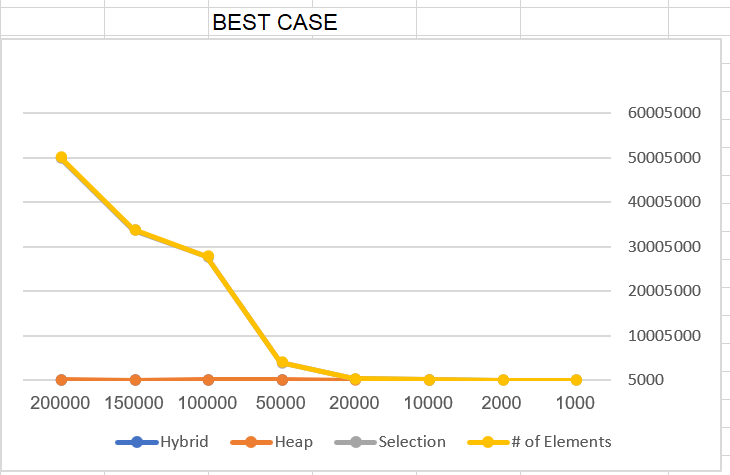
****

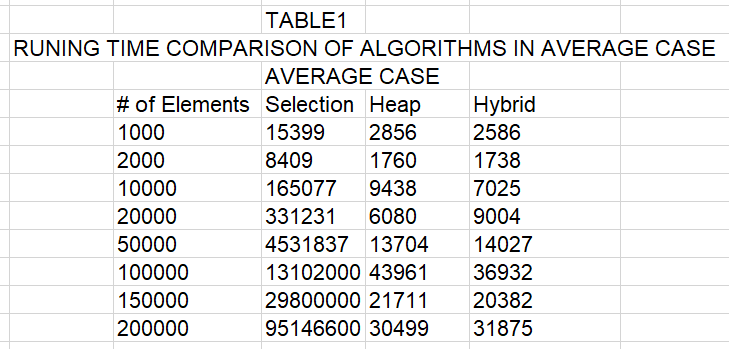
Fig. 2. Best case of three sorts

**B. Average Case Running Time:**

In this case show to three algorithms when array is random to three algorithms when array is increasing based on the number of elements and the used algorithm.

The results show that as the number of elements increases, the execution time also increases.

The results is selection sort algorithm is the highest running time in this case and Heap algorithm is lowest is running and is the best in running time.



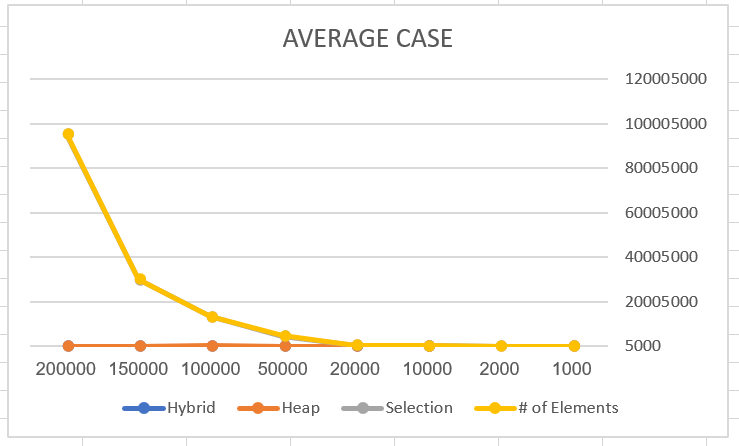


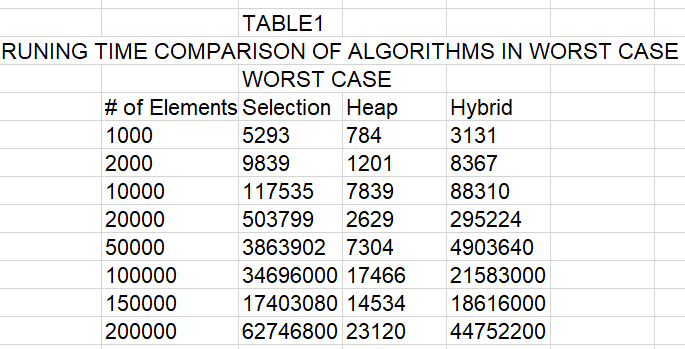
Fig. 3. Average case of three sorts

**C. Worst Case Running Time:**

In this case the arrays is in descending order algorithms when array is increasing based on the number of elements and the used algorithm.

The results show that as the number of elements increases, the execution time also increases.

The best algorithm in this case is heap because it is lowest values in running time as show in bleow table



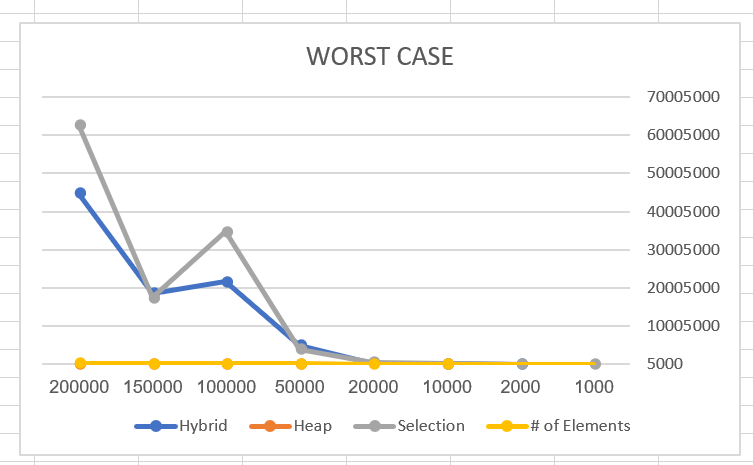


Fig. 4. Worst case of three sorts

Asthe general the best algorithm is heap algorithm from three algorithms and selection sort is worst because the bigest in running time to it as above tables in three cases and this algorithms is very similar in all cases becase time complexity in best,average and worst case is big-oh O(n^2) and heap is big-oh O(nlogn)

**5. Discussion of the best performance sorting algorithm:**

In this project the time in microseconds

**A.Theoretical VS Experimental Results Comparison**

We compre experimintal and theoretical results as this start from selection algorithm

**1.Selection sort algorithm:**

The time complexity in all cases is big-oh O(n^2)

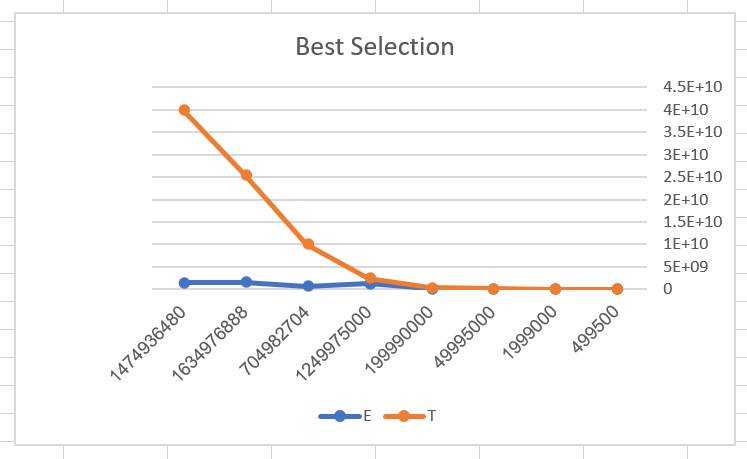


Fig. 5. Best case of selection sort

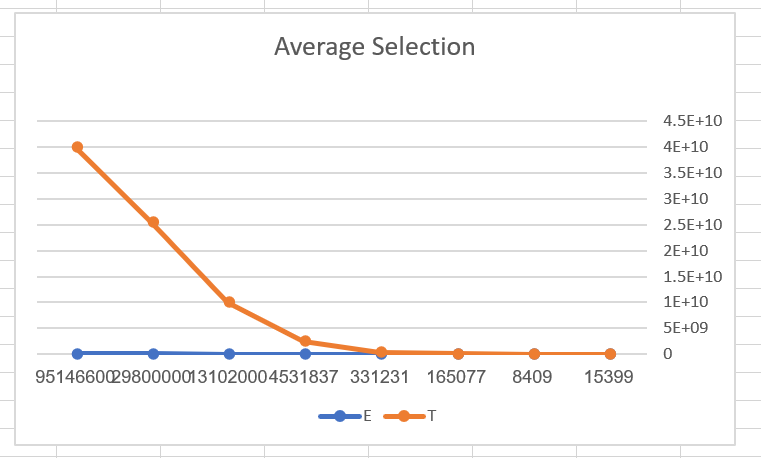


Fig. 6. Average case of selection sort

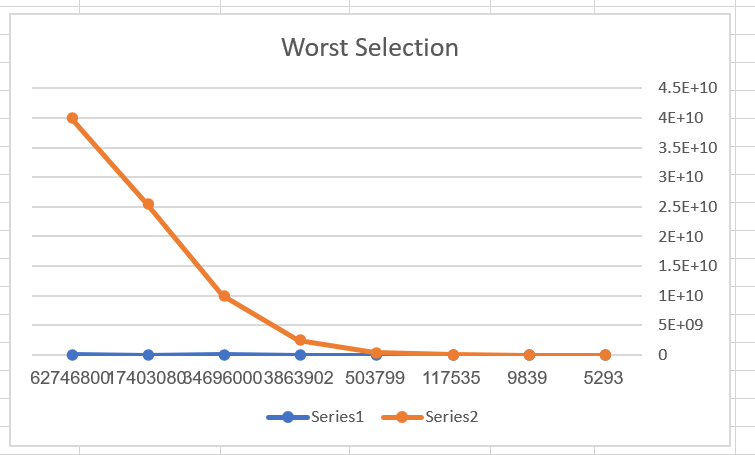
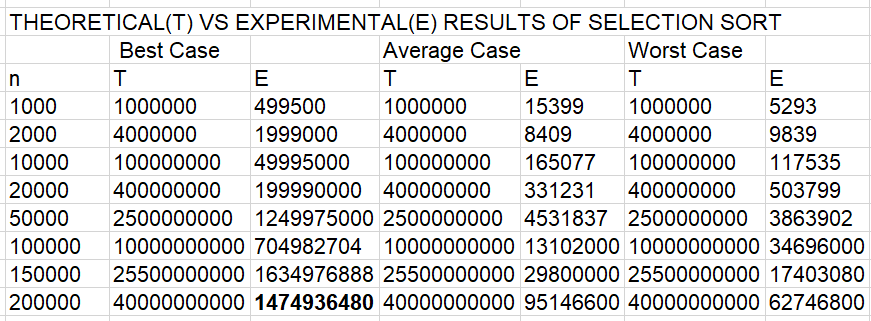


Fig. 7. Worst case of selection sort

THEORETICAL(T) VS EXPERIMENTAL(E) RESULTS OF SELECTION SORT



**2.Heap Sort Algorithm**

In this algorithm we compre experimintal and theoretical results as this we visulaize by compare it by drawing in best,average and worst case to this algorithm and the time complexity is to heap is big-oh O(nlogn) we calculate Theoritical by O(nlogn) to base 2

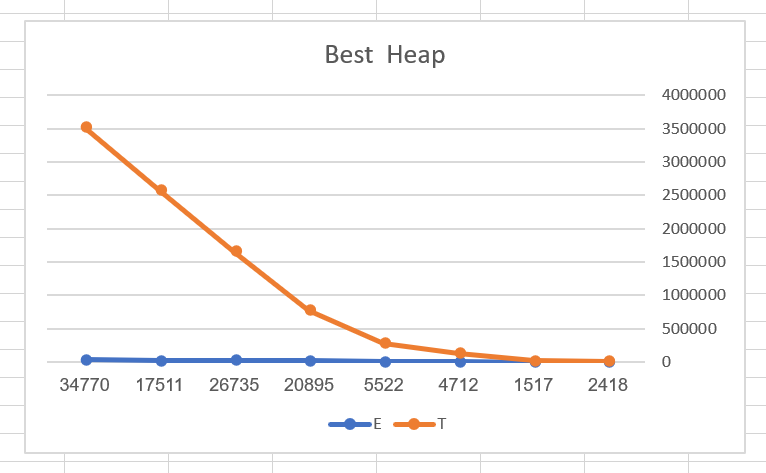


Fig. 8. Best case of Heap sort

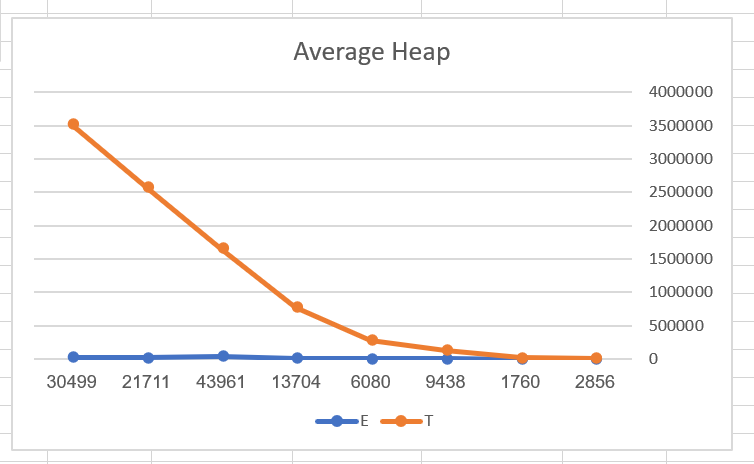


Fig. 9.Average case of Heap sort

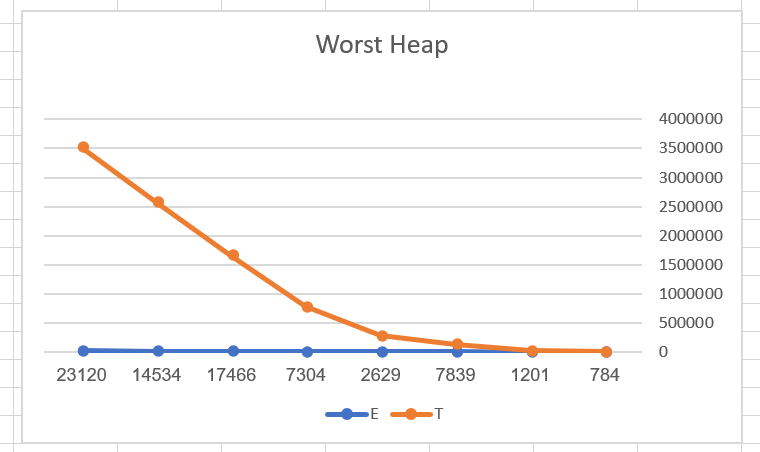
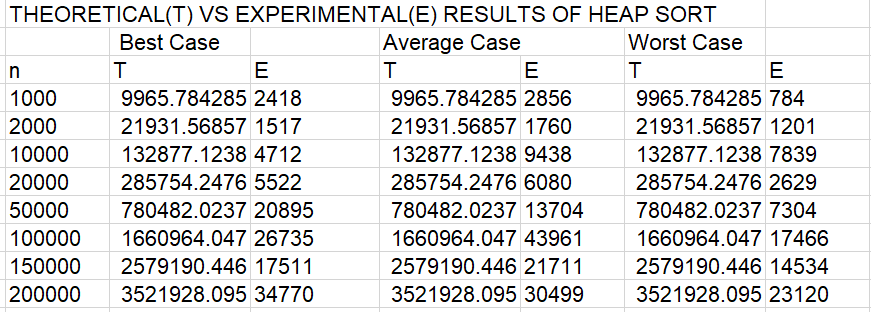


Fig. 10. Worst case of Heap sort

THEORETICAL(T) VS EXPERIMENTAL(E) RESULTS OF HEAP SORT



3.Hybrid Algorithm

In this algorithm that composed two algorithm (Selection,Heap)

we compre experimintal and theoretical results as this we visulaize by compare it by drawing in best,average and worst case to this algorithm and the time complexity is to heap is big-oh O(nlogn) in heap and when caluculating theoretical values to Selection we use big-oh O(n^2) HYBRID is as very similar in best,average as heap and when worst case is as selection sort and is the heigest value to running time .

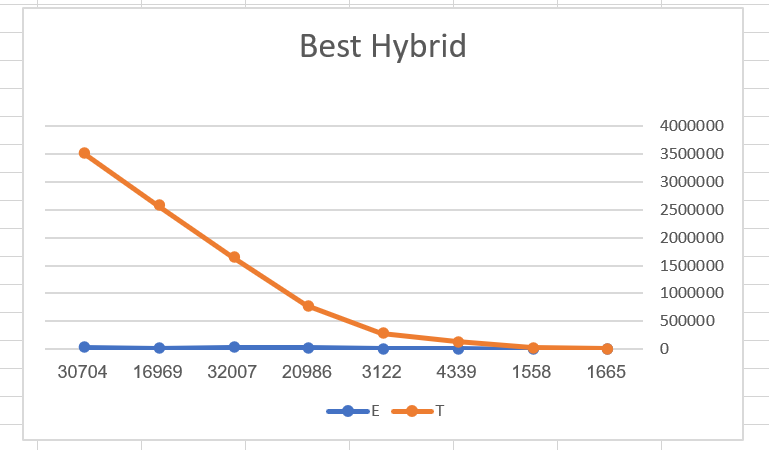


Fig. 11. Best case of Hybrid sort

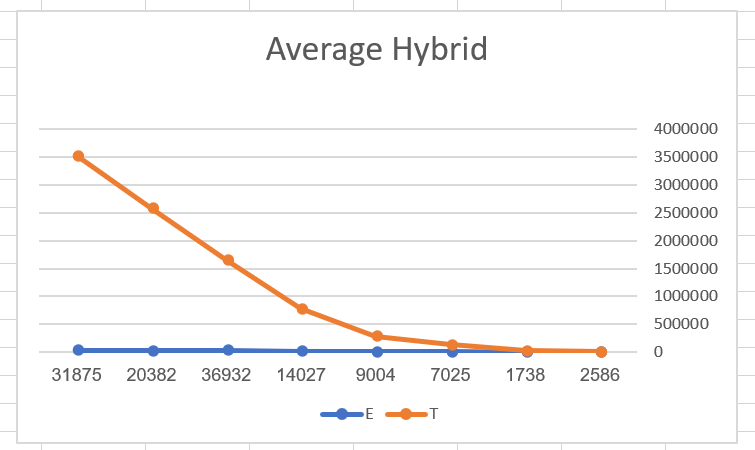


Fig. 12. Average case of Hybrid sort

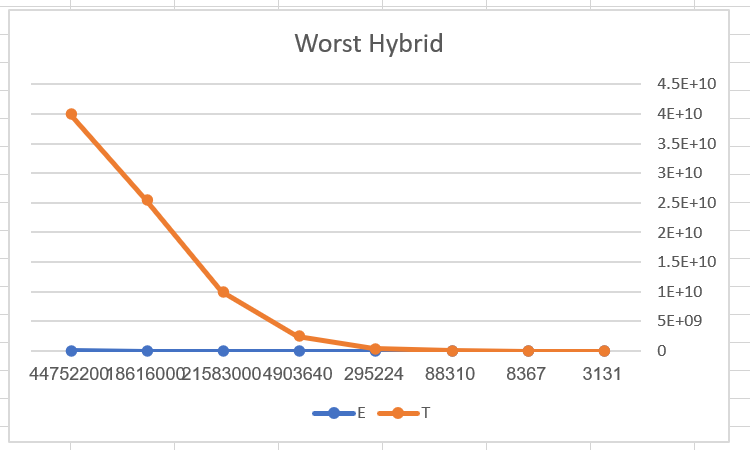
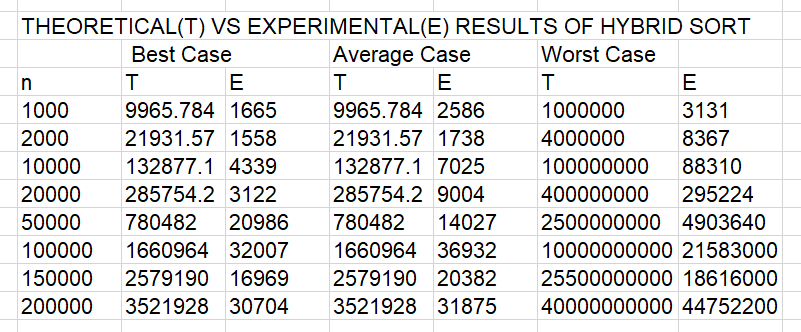


Fig. 13. Worst case of Hybrid sort

THEORETICAL(T) VS EXPERIMENTAL(E) RESULTS OF HYBRID SORT



**6. Number of Elements vs Number of Comparisons vs Running Time**

Plot (time vs. n and #comp vs. n in one plot

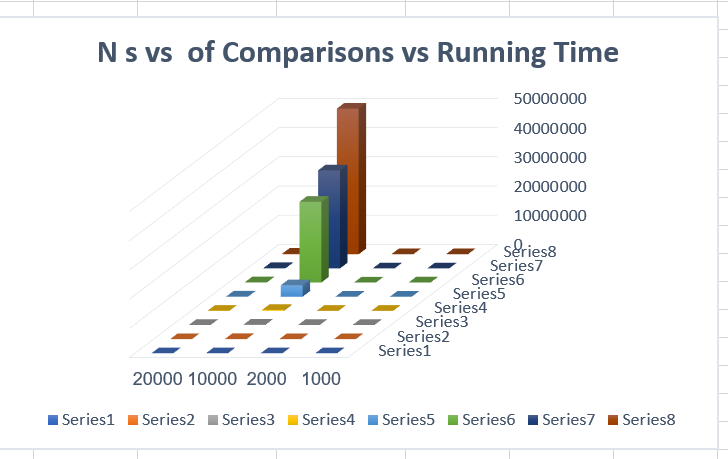
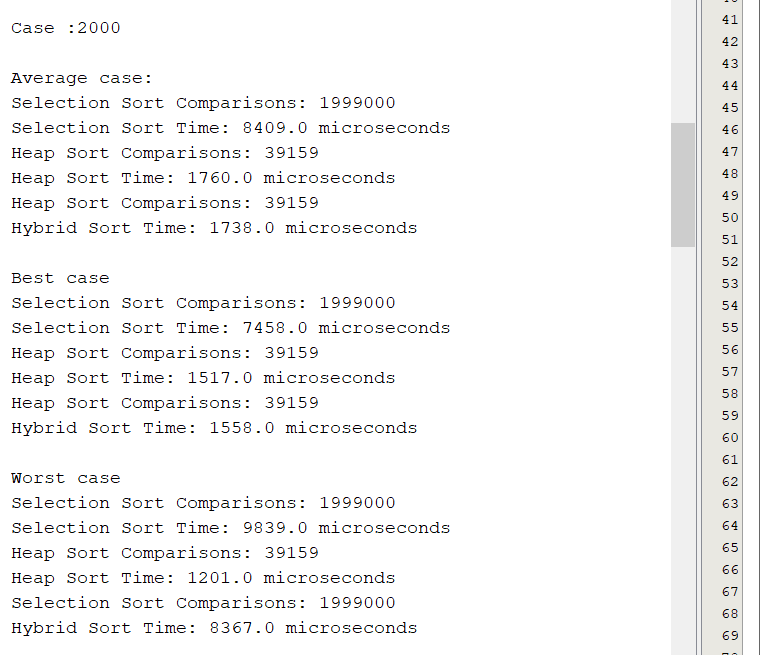
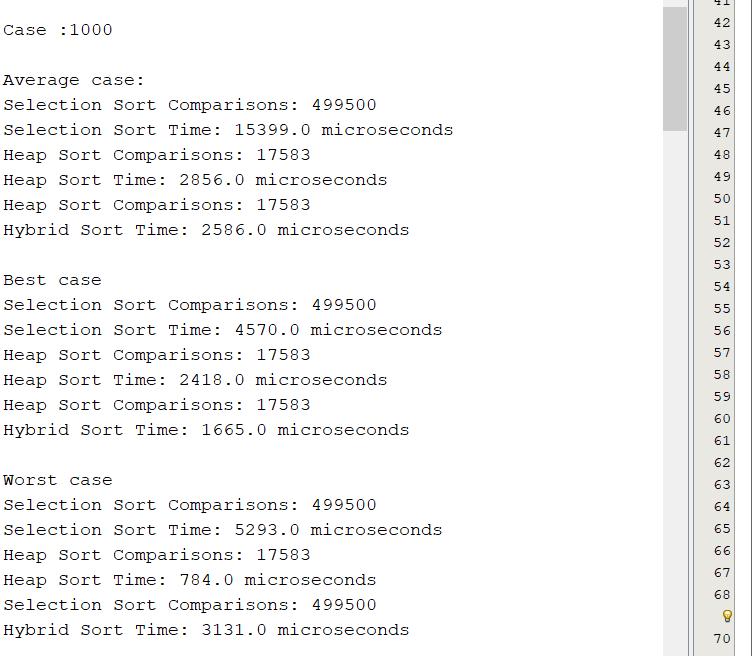
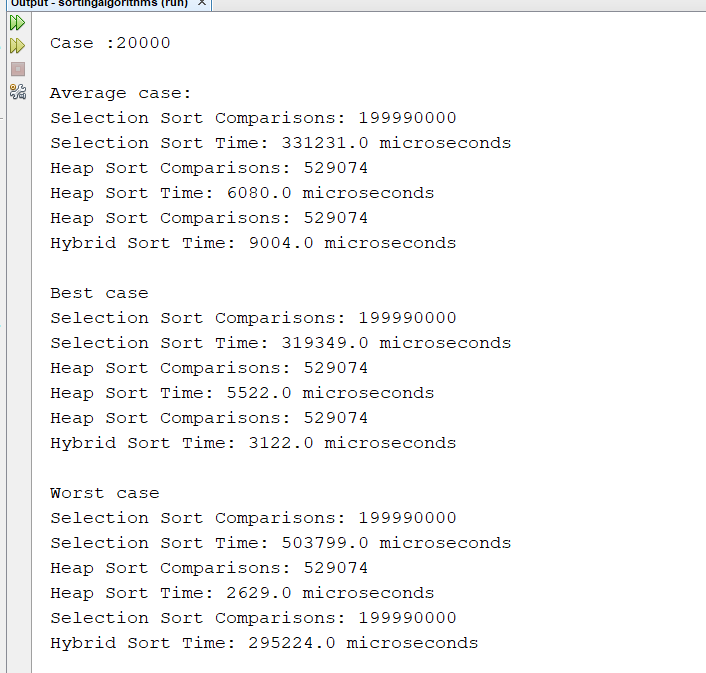


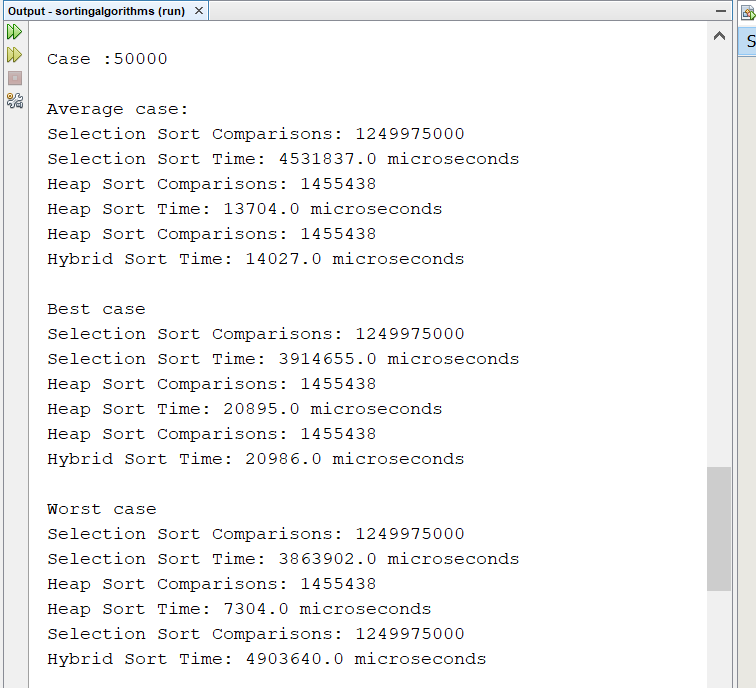
Fig. 14 . # of Elements vs # of Comparisons vs Running Time

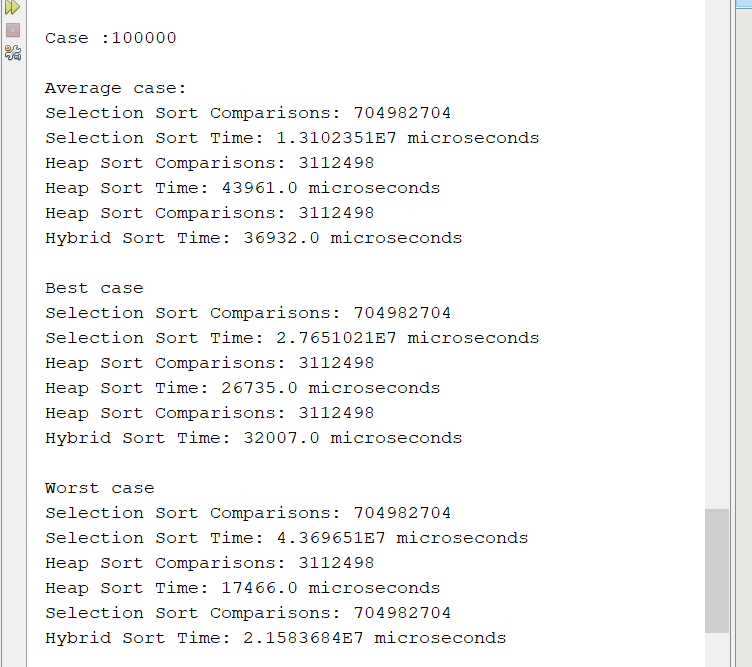
We note that when number of elements that I need to sort it the number of compersion also inceased and running time is increased also this make we sure it is relationship between them

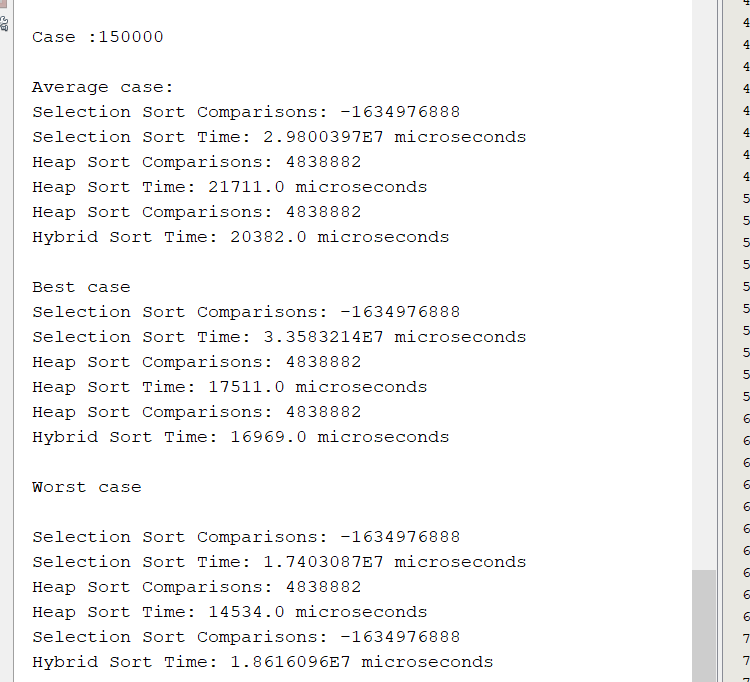
**ScreenShot To All Cases And Output**

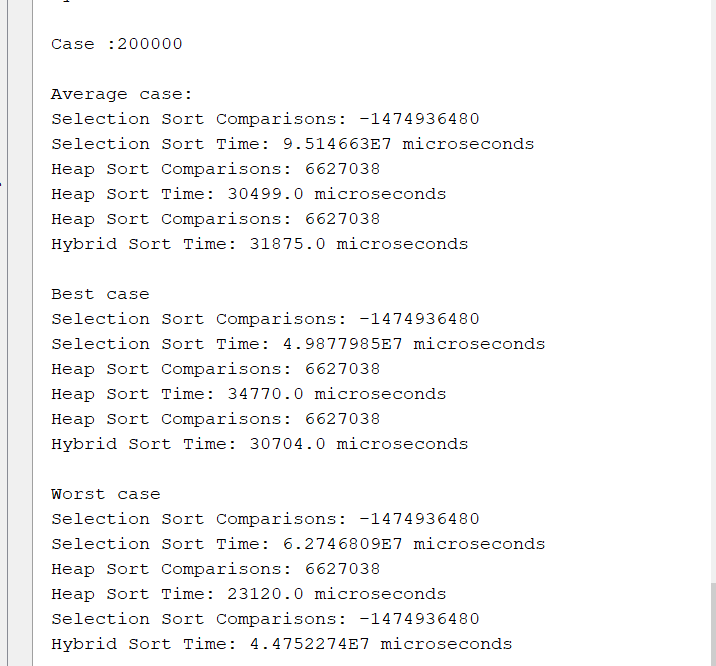
****

****

****

****

****

****