Program Structures and Algorithms
Spring 2023(SEC –3)

NAME: Akshay Parab
NUID: 002766150

Task:

In this assignment, I have performed the task to determine the best predictor of total execution time for sorting algorithms, whether it is comparisons, swaps/copies, hits (array accesses), or something else. The benchmarks for merge sort, (dual pivot) quick sort, and heap sort have been run. Randomly generated arrays between 10,000 and 256,000 elements were sorted, with the size doubling each time. The number of runs were chosen automatically as Sort Benchmark was used. For each experiment (a sort method of a given size), I have run it twice, once for the instrumentation and once without instrumentation for timing purposes. The Benchmark and/or Timer classes were used. I have attached conclusions supported with evidence from the benchmarks.

Relationship Conclusion:

The best predictor of total execution time for sorting algorithms can depend on a variety of factors, including the specific implementation and the properties of the data being sorted. In this experiment I have performed sorting on strings.

Here is a table showing the best predictor of total execution time for each sorting algorithm as per the experiment data:

Sorting Algorithm	**Best Predictor**
Heap Sort	Number of Hits
Merge Sort	Number of Copies
Quick Sort Dual Pivot	Number of Swaps

Evidence to support that conclusion:

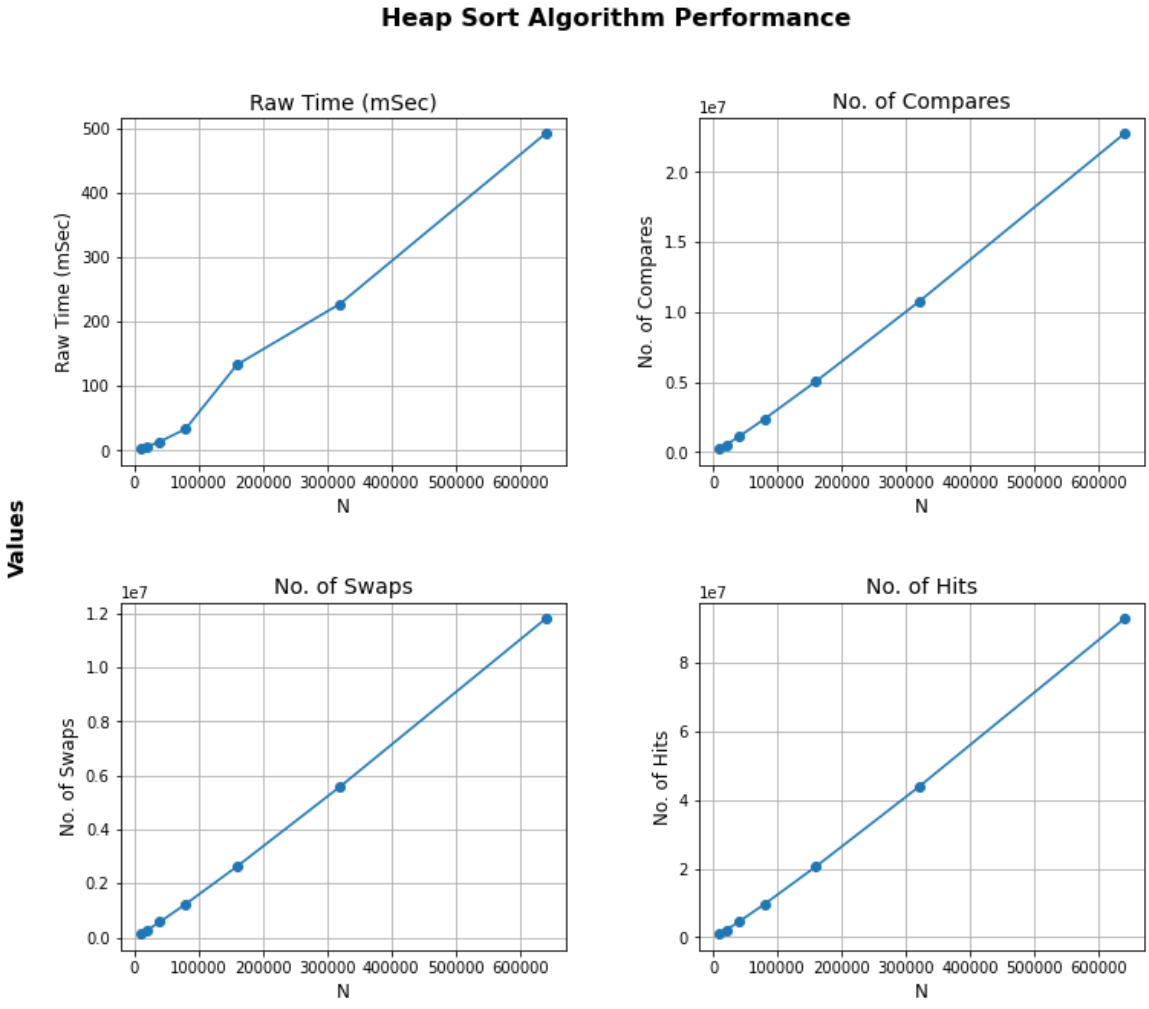
Heap Sort						
N	**Raw Time (mSec)**	**Normalized Time**	**No. of Compares**	**No. of Swaps**	**No. of Hits**	**No. of Copies**
10000	**3.02**	**4.24**	**235429**	**129948**	**968194**	**0**
20000	**6.42**	**4.16**	**510715**	**268410**	**2095158**	**0**
40000	**14.08**	**4.24**	**1101303**	**576485**	**4509222**	**0**
80000	**33.65**	**4.72**	**2363060**	**1233708**	**9661116**	**0**
160000	**133.24**	**8.76**	**5046578**	**2627430**	**20602896**	**0**
320000	**227.31**	**7.03**	**10731022**	**5574030**	**43758228**	**0**
640000	**491.16**	**7.17**	**22743819**	**11789118**	**92644114**	**0**

Merge Sort						
N	**Raw Time (mSec)**	**Normalized Time**	**No. of Compares**	**No. of Swaps**	**No. of Hits**	**No. of Copies**
10000	**1.85**	**2.6**	**101210**	**0**	**440000**	**220000**
20000	**4.21**	**2.73**	**222635**	**0**	**960000**	**480000**
40000	**8.99**	**2.71**	**486183**	**0**	**2080000**	**1040000**
80000	**16.45**	**2.31**	**1052221**	**0**	**4480000**	**2240000**

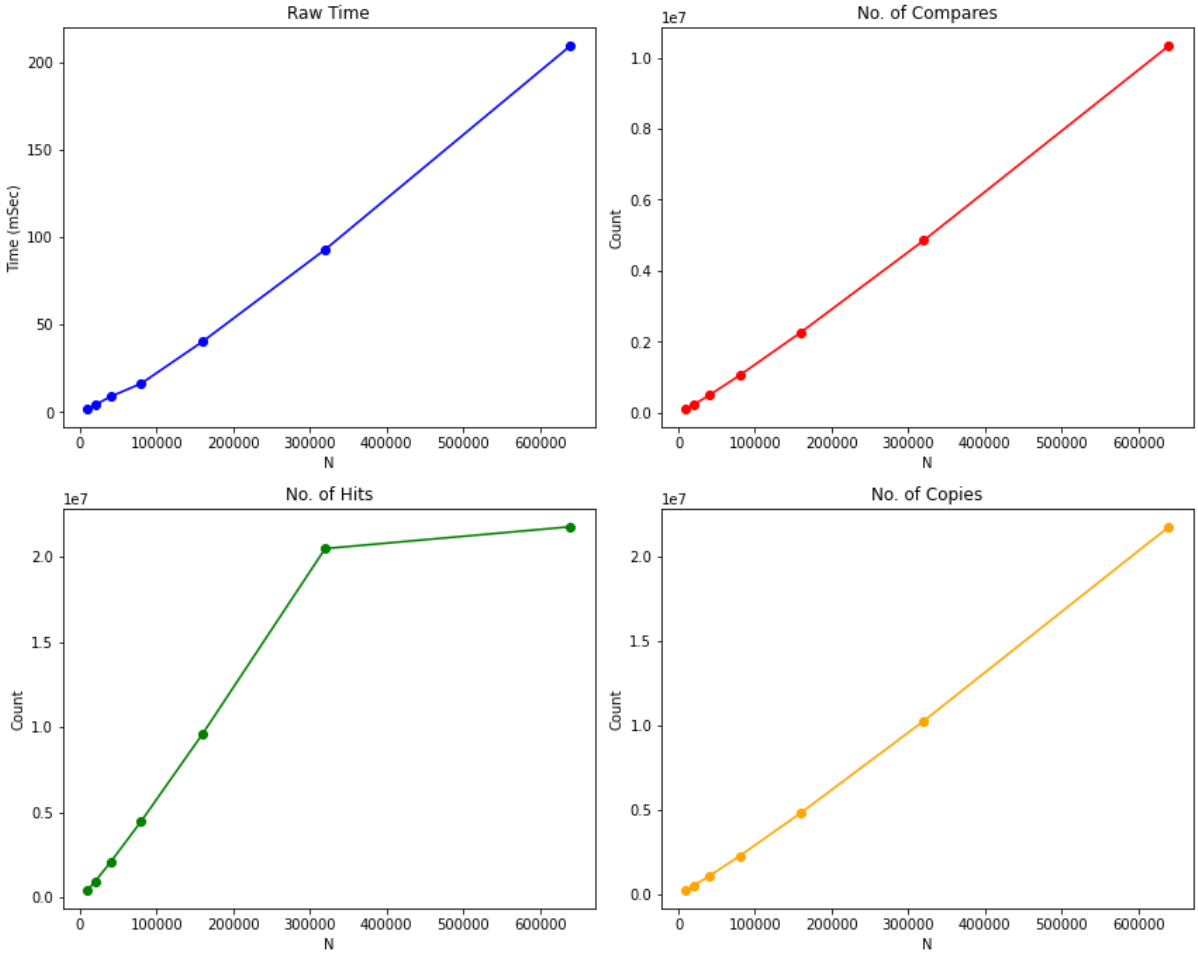
160000	**40.36**	**2.65**	**2264284**	**0**	**9600000**	**4800000**
320000	**92.92**	**2.87**	**4849461**	**0**	**20480000**	**10240000**
640000	**209.22**	**3.05**	**10336980**	**0**	**10336980**	**21760000**

Quick Sort Dual Pivot						
N	**Raw Time (mSec)**	**Normalized Time**	**No. of Compares**	**No. of Swaps**	**No. of Hits**	**No. of Copies**
10000	**2.52**	**3.55**	**151633**	**65416**	**406739**	**0**
20000	**5.46**	**3.54**	**318198**	**134594**	**863581**	**0**
40000	**11.37**	**3.42**	**750066**	**293763**	**1938449**	**0**
80000	**20.91**	**2.93**	**1539672**	**678636**	**4279536**	**0**
160000	**44.19**	**2.91**	**3380552**	**1341132**	**8798462**	**0**
320000	**104.1**	**3.22**	**6824571**	**2922694**	**18615778**	**0**
640000	**213.74**	**3.12**	**14514179**	**6125328**	**39229747**	**0**

Graphical Representation:



Merge Sort Algorithm Performance



Quick Sort Dual Pivot Performance