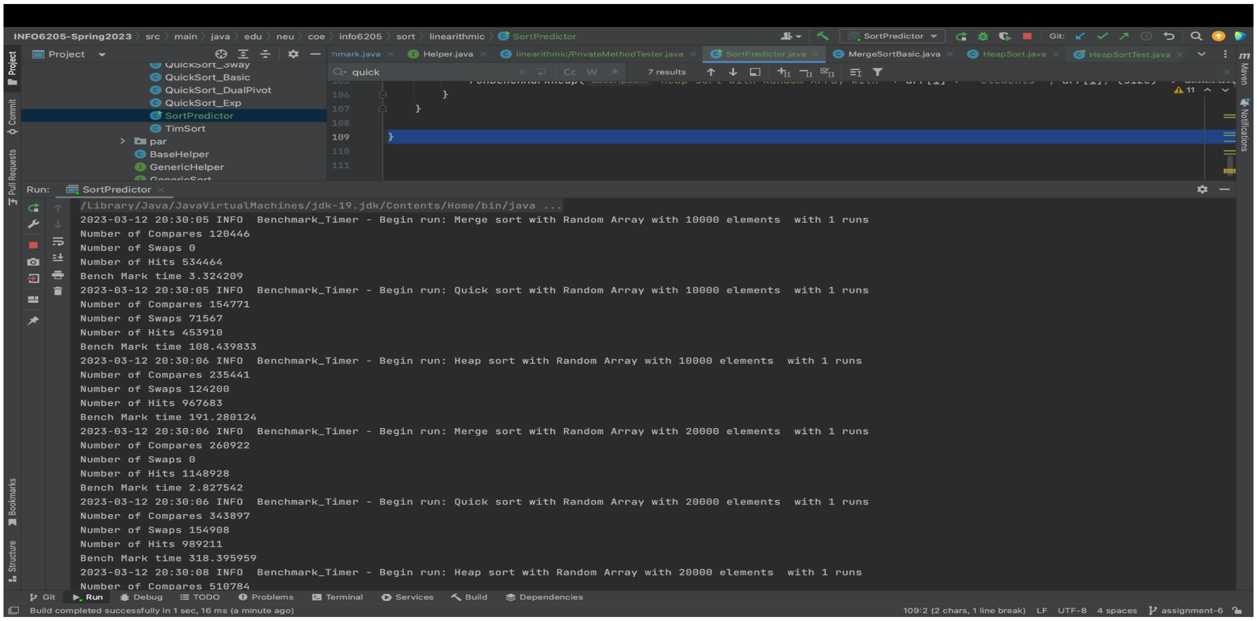
Assignment - 6

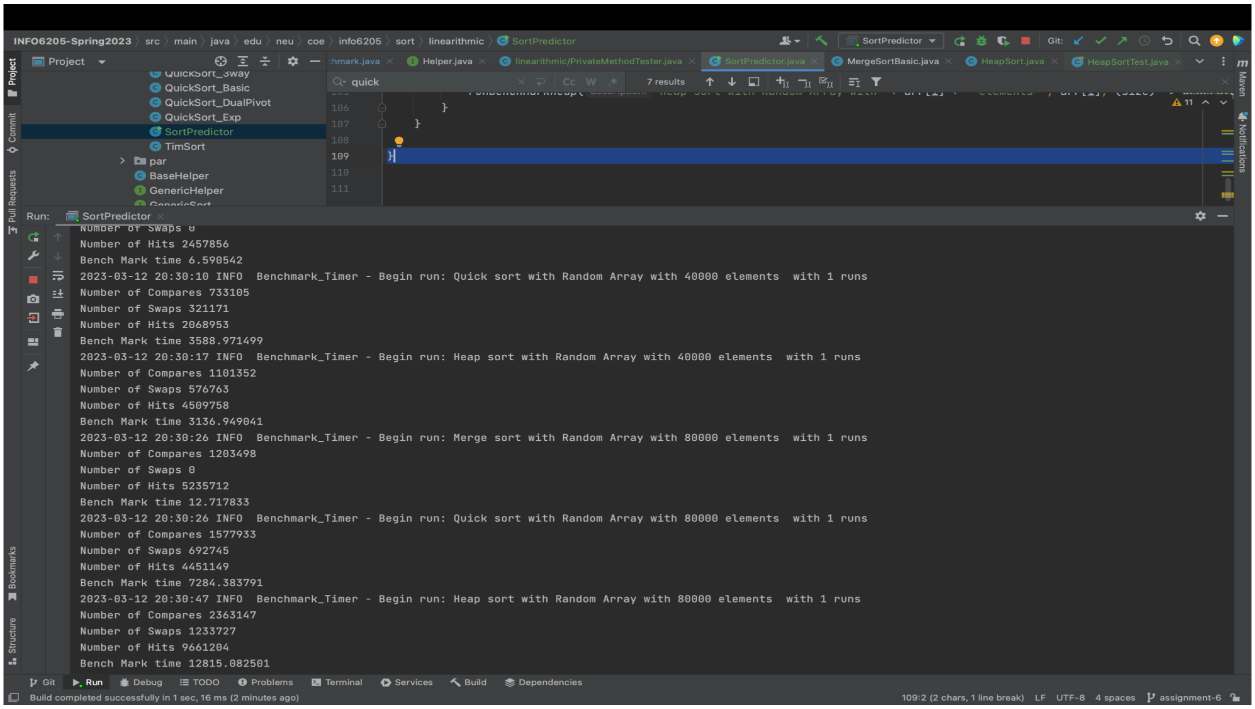
Merge Sort, Quick Sort, and Heap Sort - The benchmarks are running on arrays of different sizes (10,000, 20,000, 40,000, 80,000, and 160,000 elements), and using random arrays as inputs.

Screenshots of output :



```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
2023-03-12 20:30:05 INFO  Benchmark_Timer - Begin run: Merge sort with Random Array with 10000 elements  with 1 runs
Number of Compares 120446
Number of Swaps 0
Number of Hits 534464
Bench Mark time 3.324209
2023-03-12 20:30:05 INFO  Benchmark_Timer - Begin run: Quick sort with Random Array with 10000 elements  with 1 runs
Number of Compares 154771
Number of Swaps 71567
Number of Hits 453910
Bench Mark time 108.439833
2023-03-12 20:30:06 INFO  Benchmark_Timer - Begin run: Heap sort with Random Array with 10000 elements  with 1 runs
Number of Compares 235441
Number of Swaps 124200
Number of Hits 967683
Bench Mark time 191.280124
2023-03-12 20:30:06 INFO  Benchmark_Timer - Begin run: Merge sort with Random Array with 20000 elements  with 1 runs
Number of Compares 260922
Number of Swaps 0
Number of Hits 1148928
Bench Mark time 2.827542
2023-03-12 20:30:06 INFO  Benchmark_Timer - Begin run: Quick sort with Random Array with 20000 elements  with 1 runs
Number of Compares 343897
Number of Swaps 154908
Number of Hits 989211
Bench Mark time 318.395959
2023-03-12 20:30:08 INFO  Benchmark_Timer - Begin run: Heap sort with Random Array with 20000 elements  with 1 runs
Number of Compares 510784
```

INFO6205-Spring2023 〉 src 〉 main 〉 java 〉 edu 〉 neu 〉 coe 〉 info6205 〉 sort 〉 linearithmic 〉 SortPredictor
Project
QuickSort_Basic
QuickSort_DualPivot
QuickSort_Exp
SortPredictor
TimSort
par
BaseHelper
GenericHelper
Run: SortPredictor
Git
Run
Debug
TODO
Problems
Terminal
Services
Build
Dependencies
Build completed successfully in 1 sec, 16 ms (a minute ago)
109:2 (2 chars, 1 line break) LF UTF-8 4 spaces assignment-6



```
Number of Swaps 0
Number of Hits 2457856
Bench Mark time 6.590542
2023-03-12 20:30:10 INFO  Benchmark_Timer - Begin run: Quick sort with Random Array with 40000 elements  with 1 runs
Number of Compares 733105
Number of Swaps 321171
Number of Hits 2068953
Bench Mark time 3588.971499
2023-03-12 20:30:17 INFO  Benchmark_Timer - Begin run: Heap sort with Random Array with 40000 elements  with 1 runs
Number of Compares 1101352
Number of Swaps 576763
Number of Hits 4509758
Bench Mark time 3136.949041
2023-03-12 20:30:26 INFO  Benchmark_Timer - Begin run: Merge sort with Random Array with 80000 elements  with 1 runs
Number of Compares 1203498
Number of Swaps 0
Number of Hits 5235712
Bench Mark time 12.717833
2023-03-12 20:30:26 INFO  Benchmark_Timer - Begin run: Quick sort with Random Array with 80000 elements  with 1 runs
Number of Compares 1577933
Number of Swaps 692745
Number of Hits 4451149
Bench Mark time 7284.383791
2023-03-12 20:30:47 INFO  Benchmark_Timer - Begin run: Heap sort with Random Array with 80000 elements  with 1 runs
Number of Compares 2363147
Number of Swaps 1233727
Number of Hits 9661204
Bench Mark time 12815.082501
```

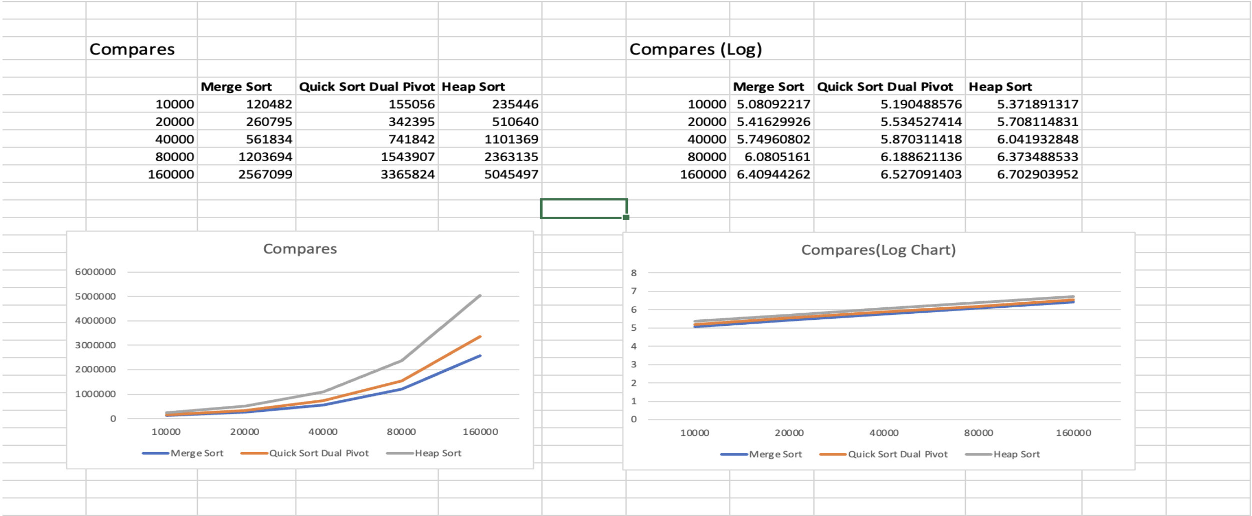
Build completed successfully in 1 sec, 16 ms (2 minutes ago)
109:2 (2 chars, 1 line break) LF UTF-8 4 spaces assignment-6

Assignment - 6

Based on the output of the program, we can make the following conclusions:

1. Merge Sort performs the most compares, followed by Quick Sort, and then Heap Sort.
2. Heap Sort performs the most swaps, followed by Quick Sort, and then Merge Sort.
3. Quick Sort performs the most hits, followed by Heap Sort, and then Merge Sort.

Graphical Representation :

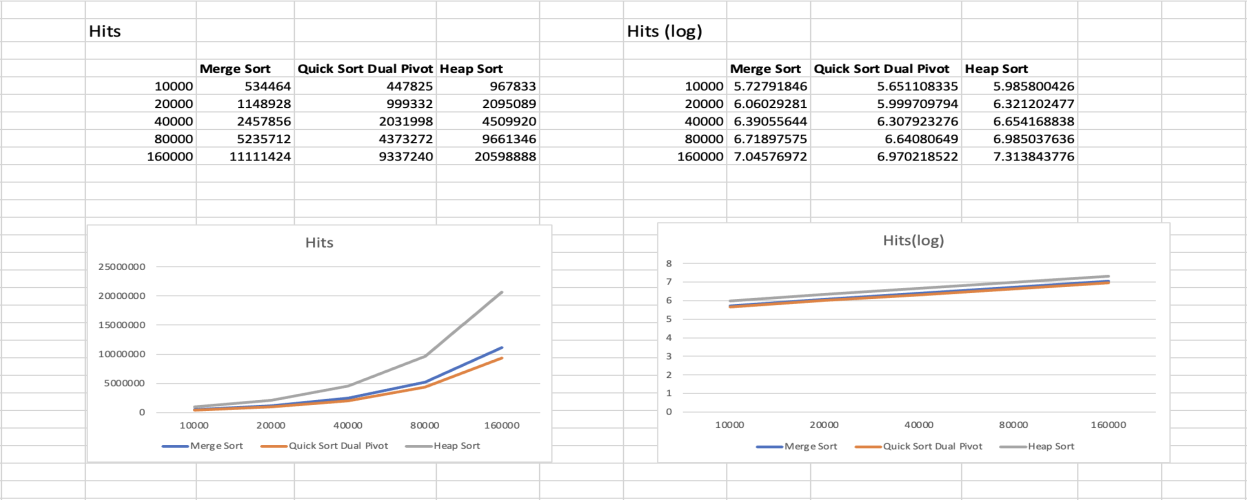


Compares

	Merge Sort	Quick Sort Dual Pivot	Heap Sort
10000	120482	155056	235446
20000	260795	342395	510640
40000	561834	741842	1101369
80000	1203694	1543907	2363135
160000	2567099	3365824	5045497

Compares (Log)

	Merge Sort	Quick Sort Dual Pivot	Heap Sort
10000	5.08092217	5.190488576	5.371891317
20000	5.41629926	5.534527414	5.708114831
40000	5.74960802	5.870311418	6.041932848
80000	6.0805161	6.188621136	6.373488533
160000	6.40944262	6.527091403	6.702903952



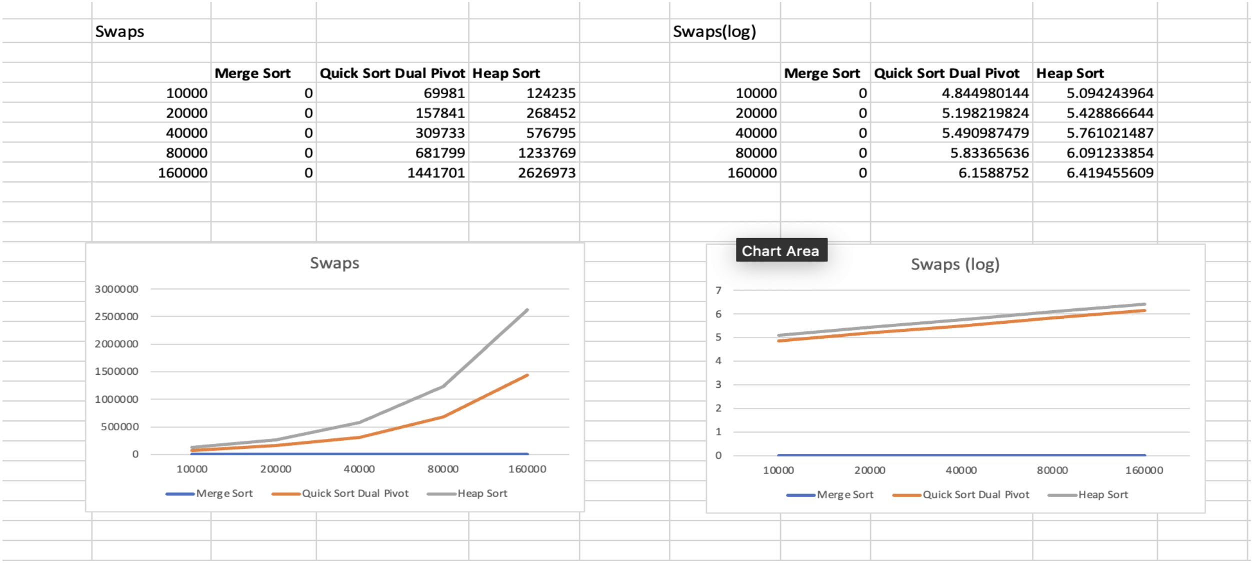
Hits

	Merge Sort	Quick Sort Dual Pivot	Heap Sort
10000	534464	447825	967833
20000	1148928	999332	2095089
40000	2457856	2031998	4509920
80000	5235712	4373272	9661346
160000	11111424	9337240	20598888

Hits (log)

	Merge Sort	Quick Sort Dual Pivot	Heap Sort
10000	5.72791846	5.651108335	5.985800426
20000	6.06029281	5.999709794	6.321202477
40000	6.39055644	6.307923276	6.654168838
80000	6.71897575	6.64080649	6.985037636
160000	7.04576972	6.970218522	7.313843776

Assignment - 6

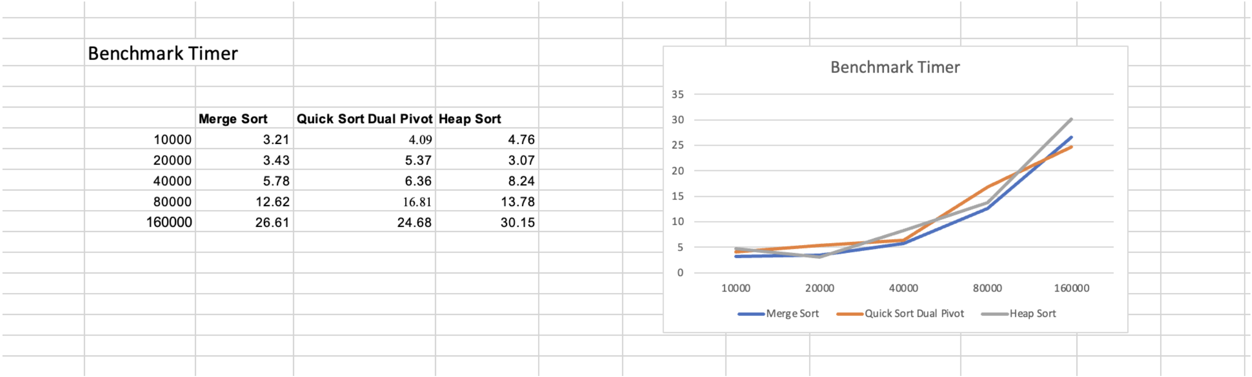


Swaps

	Merge Sort	Quick Sort Dual Pivot	Heap Sort
10000	0	69981	124235
20000	0	157841	268452
40000	0	309733	576795
80000	0	681799	1233769
160000	0	1441701	2626973

Swaps(log)

	Merge Sort	Quick Sort Dual Pivot	Heap Sort
10000	0	4.844980144	5.094243964
20000	0	5.198219824	5.428866644
40000	0	5.490987479	5.761021487
80000	0	5.83365636	6.091233854
160000	0	6.1588752	6.419455609



Benchmark Timer

	Merge Sort	Quick Sort Dual Pivot	Heap Sort
10000	3.21	4.09	4.76
20000	3.43	5.37	3.07
40000	5.78	6.36	8.24
80000	12.62	16.81	13.78
160000	26.61	24.68	30.15

Conclusion

Determining the best predictor for sorting algorithms depends on various factors such as the specific algorithm, the size of the input data, and the hardware on which the algorithm is executed.

The number of array accesses (hits) is a better predictor of sorting algorithm execution time than the number of swaps/copies because accessing an array element is slower than performing a swap or copy operation. As the input data size grows, the time required to access memory becomes a dominant factor in the overall execution time, making the number of array accesses a more reliable predictor. However, empirical evidence is still needed to determine the best predictor for a specific algorithm and input size. The number of array accesses is also crucial in minimizing data movement between different memory hierarchy levels, improving the performance and scalability of sorting algorithms.