Program Structures and Algorithms
Spring 2023(SEC – 01)
Assignment-6

NAME: **Harshini Venkata Chalam**
NUID: **002934047**

Task:
To determine- for sorting algorithms--what is the best predictor of total execution time: comparisons, swaps/copies, hits (array accesses).

Relationship Conclusion:

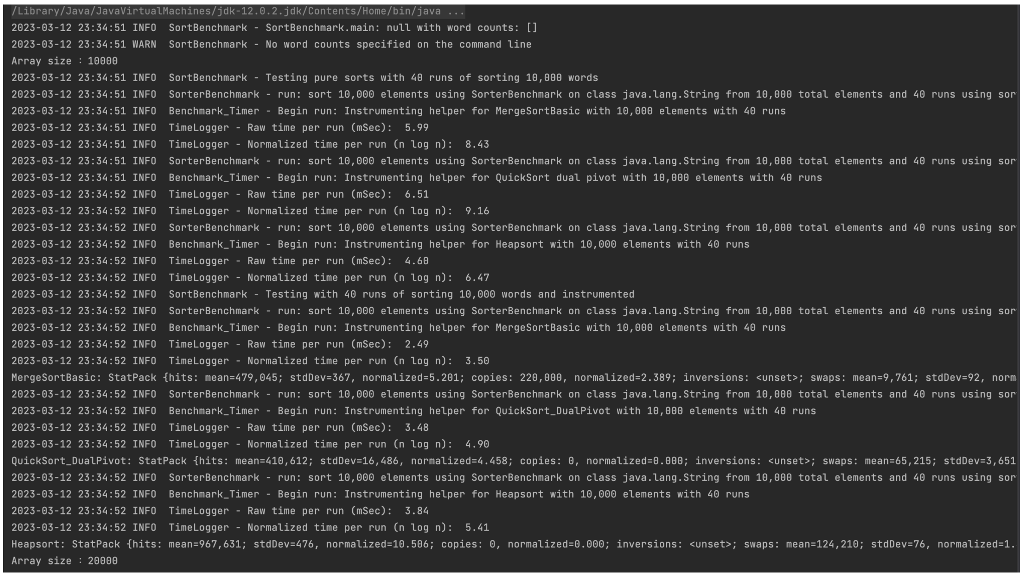
Based on the data obtained from the SortBenchmark implementation values it can be concluded or inferred that 'hits' is the best predictor of total execution time followed the comparisons and then swaps especially if the data set being sorted is large.
It can be observed that the number of it hits are increased as the time increases from the graphs plotted below.

On comparing the graphs - raw time of each of the sorting algorithms with the swaps/compares and hits it can be seen that the log-log charts time and hits linearly increases and hence that could be one of the best predictor along with number of compares .

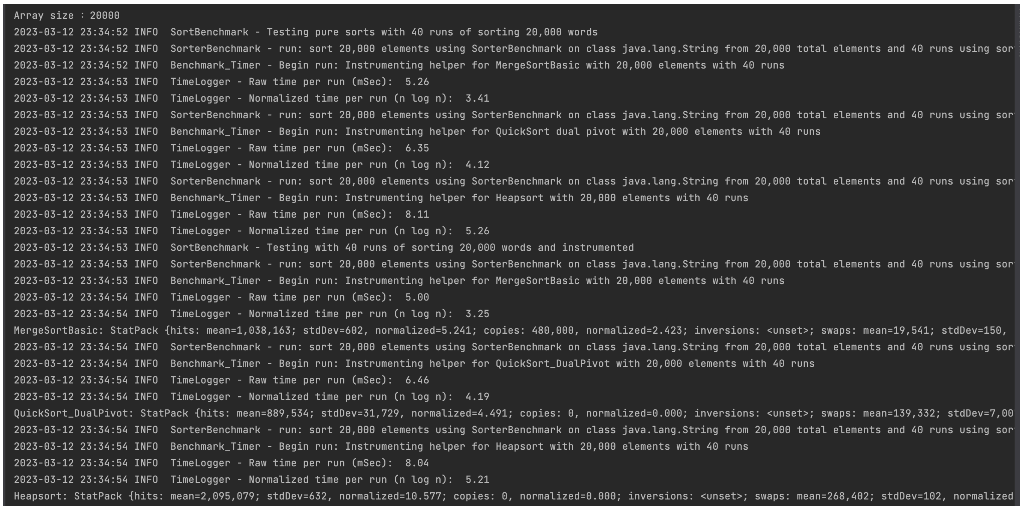
Compared to all other predictors that were taken into consideration, the data acquired from implementing the program suggests that the predictor variable "hits" has the biggest impact on the overall runtime of the program. This suggests that the quantity of hits, which is the quantity of times a specific element or value is accessed during program execution, has a significant impact on the amount of time it takes for the program to finish.
To improve the program's efficiency, minimizing the number of memory accesses can directly reduce the total time required for sorting.

Also, it was shown that although not as strongly as the "hits" variable, the predictor "comparisons" has a substantial impact on the program execution time. This shows that program execution time may also be influenced by the number of comparisons between components or values.

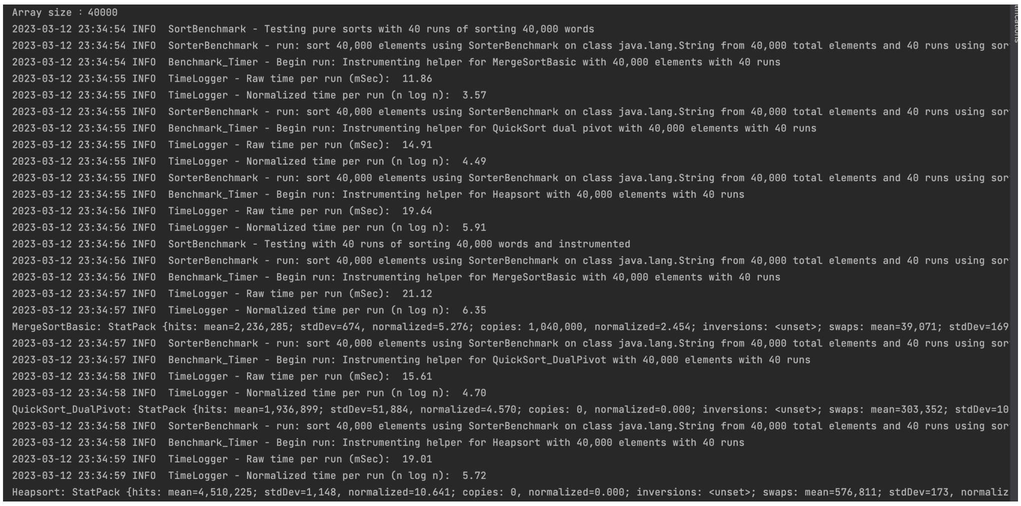
Evidence to support that conclusion:



```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
2023-03-12 23:34:51 INFO  SortBenchmark - SortBenchmark.main: null with word counts: []
2023-03-12 23:34:51 WARN  SortBenchmark - No word counts specified on the command line
Array size : 10000
2023-03-12 23:34:51 INFO  SortBenchmark - Testing pure sorts with 40 runs of sorting 10,000 words
2023-03-12 23:34:51 INFO  SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 10,000 total elements and 40 runs using sor
2023-03-12 23:34:51 INFO  Benchmark_Timer - Begin run: Instrumenting helper for MergeSortBasic with 10,000 elements with 40 runs
2023-03-12 23:34:51 INFO  TimeLogger - Raw time per run (mSec):  5.99
2023-03-12 23:34:51 INFO  TimeLogger - Normalized time per run (n log n):  8.43
2023-03-12 23:34:51 INFO  SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 10,000 total elements and 40 runs using sor
2023-03-12 23:34:51 INFO  Benchmark_Timer - Begin run: Instrumenting helper for QuickSort dual pivot with 10,000 elements with 40 runs
2023-03-12 23:34:52 INFO  TimeLogger - Raw time per run (mSec):  6.51
2023-03-12 23:34:52 INFO  TimeLogger - Normalized time per run (n log n):  9.16
2023-03-12 23:34:52 INFO  SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 10,000 total elements and 40 runs using sor
2023-03-12 23:34:52 INFO  Benchmark_Timer - Begin run: Instrumenting helper for Heapsort with 10,000 elements with 40 runs
2023-03-12 23:34:52 INFO  TimeLogger - Raw time per run (mSec):  4.60
2023-03-12 23:34:52 INFO  TimeLogger - Normalized time per run (n log n):  6.47
2023-03-12 23:34:52 INFO  SortBenchmark - Testing with 40 runs of sorting 10,000 words and instrumented
2023-03-12 23:34:52 INFO  SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 10,000 total elements and 40 runs using sor
2023-03-12 23:34:52 INFO  Benchmark_Timer - Begin run: Instrumenting helper for MergeSortBasic with 10,000 elements with 40 runs
2023-03-12 23:34:52 INFO  TimeLogger - Raw time per run (mSec):  2.49
2023-03-12 23:34:52 INFO  TimeLogger - Normalized time per run (n log n):  3.50
MergeSortBasic: StatPack {hits: mean=479,045; stdDev=367, normalized=5.201; copies: 220,000, normalized=2.389; inversions: <unset>; swaps: mean=9,761; stdDev=92, norm
2023-03-12 23:34:52 INFO  SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 10,000 total elements and 40 runs using sor
2023-03-12 23:34:52 INFO  Benchmark_Timer - Begin run: Instrumenting helper for QuickSort_DualPivot with 10,000 elements with 40 runs
2023-03-12 23:34:52 INFO  TimeLogger - Raw time per run (mSec):  3.48
2023-03-12 23:34:52 INFO  TimeLogger - Normalized time per run (n log n):  4.90
QuickSort_DualPivot: StatPack {hits: mean=410,612; stdDev=16,486, normalized=4.458; copies: 0, normalized=0.000; inversions: <unset>; swaps: mean=65,215; stdDev=3,651
2023-03-12 23:34:52 INFO  SorterBenchmark - run: sort 10,000 elements using SorterBenchmark on class java.lang.String from 10,000 total elements and 40 runs using sor
2023-03-12 23:34:52 INFO  Benchmark_Timer - Begin run: Instrumenting helper for Heapsort with 10,000 elements with 40 runs
2023-03-12 23:34:52 INFO  TimeLogger - Raw time per run (mSec):  3.84
2023-03-12 23:34:52 INFO  TimeLogger - Normalized time per run (n log n):  5.41
Heapsort: StatPack {hits: mean=967,631; stdDev=476, normalized=10.506; copies: 0, normalized=0.000; inversions: <unset>; swaps: mean=124,210; stdDev=76, normalized=1.
Array size : 20000
```



```
Array size : 20000
2023-03-12 23:34:52 INFO  SortBenchmark - Testing pure sorts with 40 runs of sorting 20,000 words
2023-03-12 23:34:52 INFO  SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 20,000 total elements and 40 runs using sor
2023-03-12 23:34:52 INFO  Benchmark_Timer - Begin run: Instrumenting helper for MergeSortBasic with 20,000 elements with 40 runs
2023-03-12 23:34:53 INFO  TimeLogger - Raw time per run (mSec):  5.26
2023-03-12 23:34:53 INFO  TimeLogger - Normalized time per run (n log n):  3.41
2023-03-12 23:34:53 INFO  SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 20,000 total elements and 40 runs using sor
2023-03-12 23:34:53 INFO  Benchmark_Timer - Begin run: Instrumenting helper for QuickSort dual pivot with 20,000 elements with 40 runs
2023-03-12 23:34:53 INFO  TimeLogger - Raw time per run (mSec):  6.35
2023-03-12 23:34:53 INFO  TimeLogger - Normalized time per run (n log n):  4.12
2023-03-12 23:34:53 INFO  SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 20,000 total elements and 40 runs using sor
2023-03-12 23:34:53 INFO  Benchmark_Timer - Begin run: Instrumenting helper for Heapsort with 20,000 elements with 40 runs
2023-03-12 23:34:53 INFO  TimeLogger - Raw time per run (mSec):  8.11
2023-03-12 23:34:53 INFO  TimeLogger - Normalized time per run (n log n):  5.26
2023-03-12 23:34:53 INFO  SortBenchmark - Testing with 40 runs of sorting 20,000 words and instrumented
2023-03-12 23:34:53 INFO  SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 20,000 total elements and 40 runs using sor
2023-03-12 23:34:53 INFO  Benchmark_Timer - Begin run: Instrumenting helper for MergeSortBasic with 20,000 elements with 40 runs
2023-03-12 23:34:54 INFO  TimeLogger - Raw time per run (mSec):  5.00
2023-03-12 23:34:54 INFO  TimeLogger - Normalized time per run (n log n):  3.25
MergeSortBasic: StatPack {hits: mean=1,038,163; stdDev=602, normalized=5.241; copies: 480,000, normalized=2.423; inversions: <unset>; swaps: mean=19,541; stdDev=150,
2023-03-12 23:34:54 INFO  SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 20,000 total elements and 40 runs using sor
2023-03-12 23:34:54 INFO  Benchmark_Timer - Begin run: Instrumenting helper for QuickSort_DualPivot with 20,000 elements with 40 runs
2023-03-12 23:34:54 INFO  TimeLogger - Raw time per run (mSec):  6.46
2023-03-12 23:34:54 INFO  TimeLogger - Normalized time per run (n log n):  4.19
QuickSort_DualPivot: StatPack {hits: mean=889,534; stdDev=31,729, normalized=4.491; copies: 0, normalized=0.000; inversions: <unset>; swaps: mean=139,332; stdDev=7,00
2023-03-12 23:34:54 INFO  SorterBenchmark - run: sort 20,000 elements using SorterBenchmark on class java.lang.String from 20,000 total elements and 40 runs using sor
2023-03-12 23:34:54 INFO  Benchmark_Timer - Begin run: Instrumenting helper for Heapsort with 20,000 elements with 40 runs
2023-03-12 23:34:54 INFO  TimeLogger - Raw time per run (mSec):  8.04
2023-03-12 23:34:54 INFO  TimeLogger - Normalized time per run (n log n):  5.21
Heapsort: StatPack {hits: mean=2,095,079; stdDev=632, normalized=10.577; copies: 0, normalized=0.000; inversions: <unset>; swaps: mean=268,402; stdDev=102, normalized
```



```
Array size : 40000
2023-03-12 23:34:54 INFO  SortBenchmark - Testing pure sorts with 40 runs of sorting 40,000 words
2023-03-12 23:34:54 INFO  SorterBenchmark - run: sort 40,000 elements using SorterBenchmark on class java.lang.String from 40,000 total elements and 40 runs using sor
2023-03-12 23:34:54 INFO  Benchmark_Timer - Begin run: Instrumenting helper for MergeSortBasic with 40,000 elements with 40 runs
2023-03-12 23:34:55 INFO  TimeLogger - Raw time per run (mSec):  11.86
2023-03-12 23:34:55 INFO  TimeLogger - Normalized time per run (n log n):  3.57
2023-03-12 23:34:55 INFO  SorterBenchmark - run: sort 40,000 elements using SorterBenchmark on class java.lang.String from 40,000 total elements and 40 runs using sor
2023-03-12 23:34:55 INFO  Benchmark_Timer - Begin run: Instrumenting helper for QuickSort dual pivot with 40,000 elements with 40 runs
2023-03-12 23:34:55 INFO  TimeLogger - Raw time per run (mSec):  14.91
2023-03-12 23:34:55 INFO  TimeLogger - Normalized time per run (n log n):  4.49
2023-03-12 23:34:55 INFO  SorterBenchmark - run: sort 40,000 elements using SorterBenchmark on class java.lang.String from 40,000 total elements and 40 runs using sor
2023-03-12 23:34:55 INFO  Benchmark_Timer - Begin run: Instrumenting helper for Heapsort with 40,000 elements with 40 runs
2023-03-12 23:34:56 INFO  TimeLogger - Raw time per run (mSec):  19.64
2023-03-12 23:34:56 INFO  TimeLogger - Normalized time per run (n log n):  5.91
2023-03-12 23:34:56 INFO  SortBenchmark - Testing with 40 runs of sorting 40,000 words and instrumented
2023-03-12 23:34:56 INFO  SorterBenchmark - run: sort 40,000 elements using SorterBenchmark on class java.lang.String from 40,000 total elements and 40 runs using sor
2023-03-12 23:34:56 INFO  Benchmark_Timer - Begin run: Instrumenting helper for MergeSortBasic with 40,000 elements with 40 runs
2023-03-12 23:34:57 INFO  TimeLogger - Raw time per run (mSec):  21.12
2023-03-12 23:34:57 INFO  TimeLogger - Normalized time per run (n log n):  6.35
MergeSortBasic: StatPack {hits: mean=2,236,285; stdDev=674, normalized=5.276; copies: 1,040,000, normalized=2.454; inversions: <unset>; swaps: mean=39,071; stdDev=169
2023-03-12 23:34:57 INFO  SorterBenchmark - run: sort 40,000 elements using SorterBenchmark on class java.lang.String from 40,000 total elements and 40 runs using sor
2023-03-12 23:34:57 INFO  Benchmark_Timer - Begin run: Instrumenting helper for QuickSort_DualPivot with 40,000 elements with 40 runs
2023-03-12 23:34:58 INFO  TimeLogger - Raw time per run (mSec):  15.61
2023-03-12 23:34:58 INFO  TimeLogger - Normalized time per run (n log n):  4.70
QuickSort_DualPivot: StatPack {hits: mean=1,936,899; stdDev=51,884, normalized=4.570; copies: 0, normalized=0.000; inversions: <unset>; swaps: mean=303,352; stdDev=10
2023-03-12 23:34:58 INFO  SorterBenchmark - run: sort 40,000 elements using SorterBenchmark on class java.lang.String from 40,000 total elements and 40 runs using sor
2023-03-12 23:34:58 INFO  Benchmark_Timer - Begin run: Instrumenting helper for Heapsort with 40,000 elements with 40 runs
2023-03-12 23:34:59 INFO  TimeLogger - Raw time per run (mSec):  19.01
2023-03-12 23:34:59 INFO  TimeLogger - Normalized time per run (n log n):  5.72
Heapsort: StatPack {hits: mean=4,510,225; stdDev=1,148, normalized=10.641; copies: 0, normalized=0.000; inversions: <unset>; swaps: mean=576,811; stdDev=173, normaliz
```

MergeSortBasic

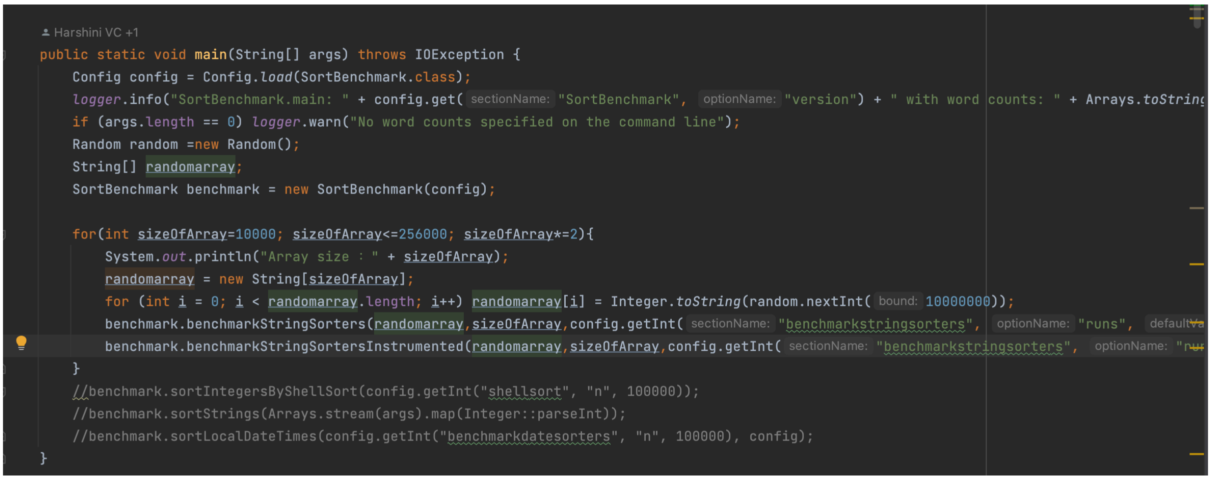
ArraySize	Hits	Swaps	Compares	Time
10000	478,981	9,745	121,489	5.99
20000	1,038,206	19,551	262,994	5.26
40000	2,236,197	39,049	566,056	11.86
80000	4,792,163	78,041	1,211,978	29.45
160000	10,225,143	156,286	2,584,005	60.94

QuickSortDualPivot

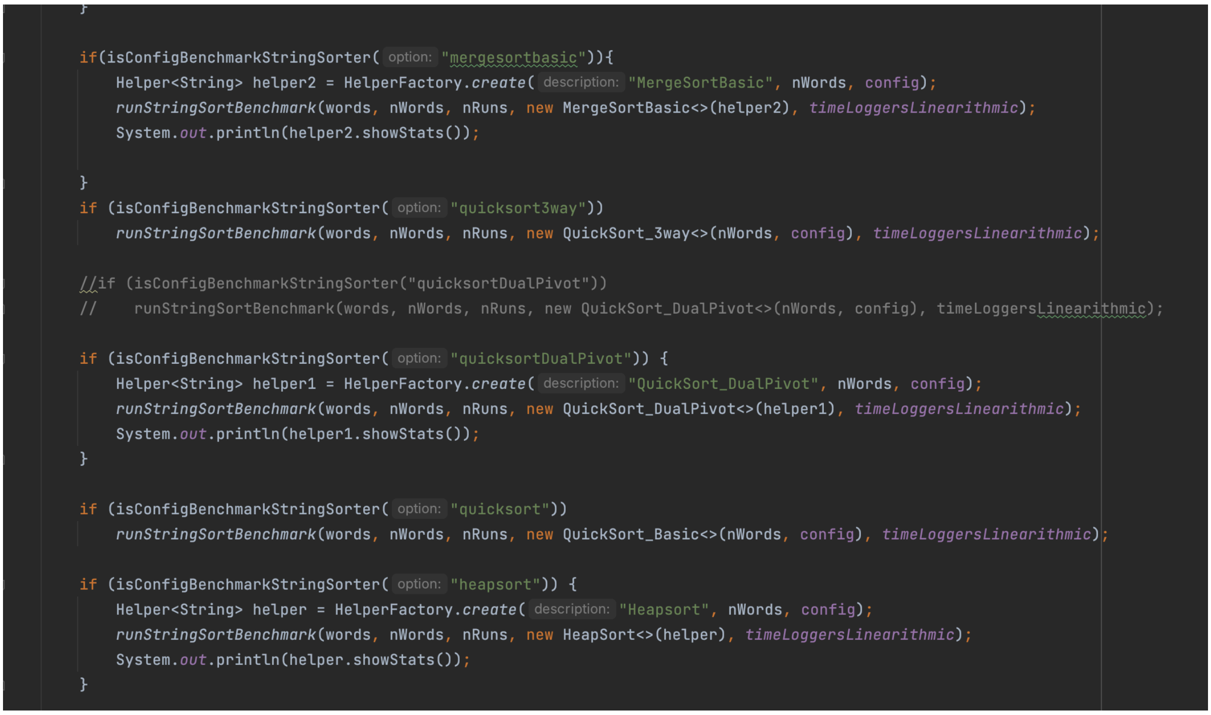
ArraySize	Hits	Swaps	Compares	Time
10000	411,697	65,120	155,548	6.51
20000	898,130	140,043	338,518	6.35
40000	1,921,465	303,304	725,594	19.64
80000	4,239,749	672,753	1,583,484	30.78
160000	8,911,442	1,398,162	3,388,300	66.67

HeapSort

ArraySize	Hits	Swaps	Compares	Time(mS)
10000	967,603	124,211	235,380	4.6
20000	2,095,008	268,384	510,736	8.11
40000	4,510,221	576,788	1,101,534	21.12
80000	9,660,331	1,233,589	2,362,987	41.22
160000	20,599,776	2,627,047	5,045,794	87.58



```
Harshini VC +1
public static void main(String[] args) throws IOException {
    Config config = Config.load(SortBenchmark.class);
    logger.info("SortBenchmark.main: " + config.get( sectionName: "SortBenchmark",  optionName: "version") + " with word counts: " + Arrays.toStrin
    if (args.length == 0) logger.warn("No word counts specified on the command line");
    Random random =new Random();
    String[] randomarray;
    SortBenchmark benchmark = new SortBenchmark(config);

    for(int sizeOfArray=10000; sizeOfArray<=256000; sizeOfArray*=2){
        System.out.println("Array size : " + sizeOfArray);
        randomarray = new String[sizeOfArray];
        for (int i = 0; i < randomarray.length; i++) randomarray[i] = Integer.toString(random.nextInt( bound: 10000000));
        benchmark.benchmarkStringSorters(randomarray,sizeOfArray,config.getInt( sectionName: "benchmarkstringsorters",  optionName: "runs",  defaultVa
        benchmark.benchmarkStringSortersInstrumented(randomarray,sizeOfArray,config.getInt( sectionName: "benchmarkstringsorters",  optionName: "ru
    }
    //benchmark.sortIntegersByShellSort(config.getInt("shellsort", "n", 100000));
    //benchmark.sortStrings(Arrays.stream(args).map(Integer::parseInt));
    //benchmark.sortLocalDateTimes(config.getInt("benchmarkdatesorters", "n", 100000), config);
}
```

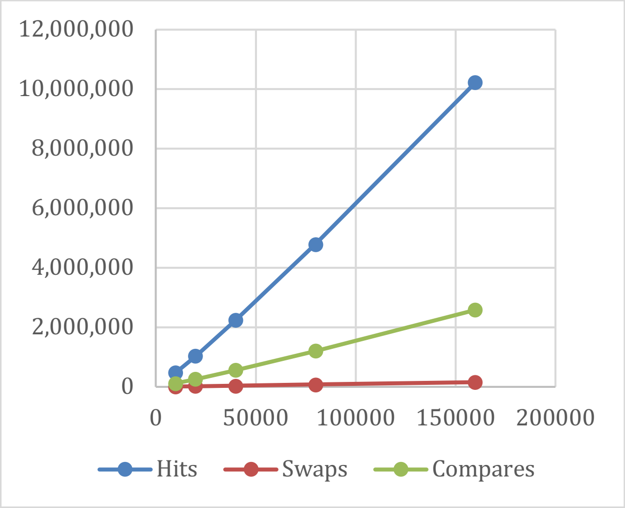


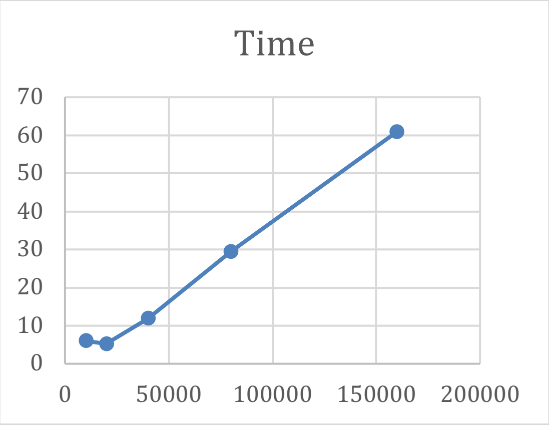
```
    }

    if(isConfigBenchmarkStringSorter( option: "mergesortbasic")){
        Helper<String> helper2 = HelperFactory.create( description: "MergeSortBasic", nWords, config);
        runStringSortBenchmark(words, nWords, nRuns, new MergeSortBasic<>(helper2), timeLoggersLinearithmic);
        System.out.println(helper2.showStats());

    }
    if (isConfigBenchmarkStringSorter( option: "quicksort3way"))
        runStringSortBenchmark(words, nWords, nRuns, new QuickSort_3way<>(nWords, config), timeLoggersLinearithmic);

    //if (isConfigBenchmarkStringSorter("quicksortDualPivot"))
    //    runStringSortBenchmark(words, nWords, nRuns, new QuickSort_DualPivot<>(nWords, config), timeLoggersLinearithmic);

    if (isConfigBenchmarkStringSorter( option: "quicksortDualPivot")) {
        Helper<String> helper1 = HelperFactory.create( description: "QuickSort_DualPivot", nWords, config);
        runStringSortBenchmark(words, nWords, nRuns, new QuickSort_DualPivot<>(helper1), timeLoggersLinearithmic);
        System.out.println(helper1.showStats());
    }

    if (isConfigBenchmarkStringSorter( option: "quicksort"))
        runStringSortBenchmark(words, nWords, nRuns, new QuickSort_Basic<>(nWords, config), timeLoggersLinearithmic);

    if (isConfigBenchmarkStringSorter( option: "heapsort")) {
        Helper<String> helper = HelperFactory.create( description: "Heapsort", nWords, config);
        runStringSortBenchmark(words, nWords, nRuns, new HeapSort<>(helper), timeLoggersLinearithmic);
        System.out.println(helper.showStats());
    }
```

Graphical Representation:

MergeSortBasic

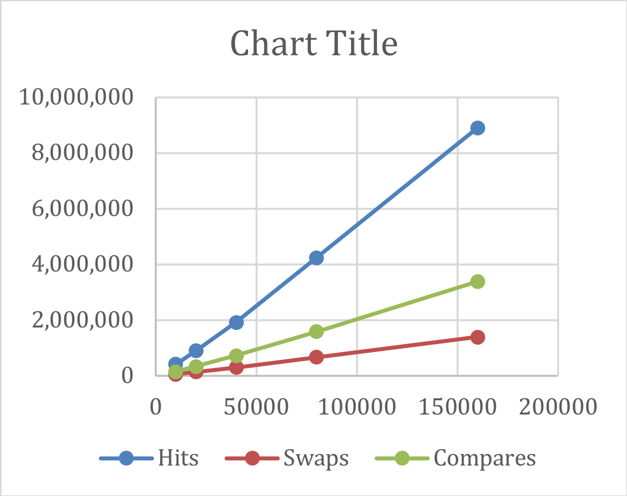
ArraySize	Hits	Swaps	Compares	Time
10000	478,981	9,745	121,489	5.99
20000	1,038,206	19,551	262,994	5.26
40000	2,236,197	39,049	566,056	11.86
80000	4,792,163	78,041	1,211,978	29.45
160000	10,225,143	156,286	2,584,005	60.94

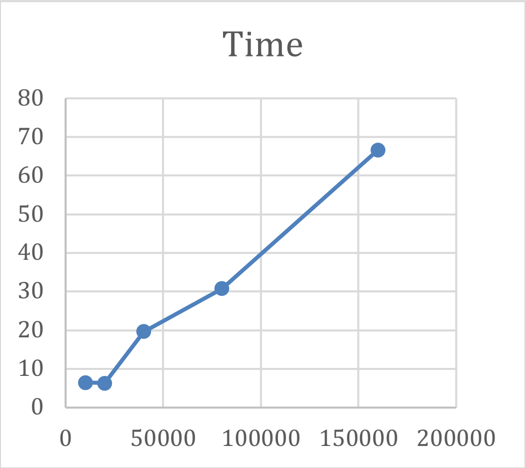




QuickSortDualPivot

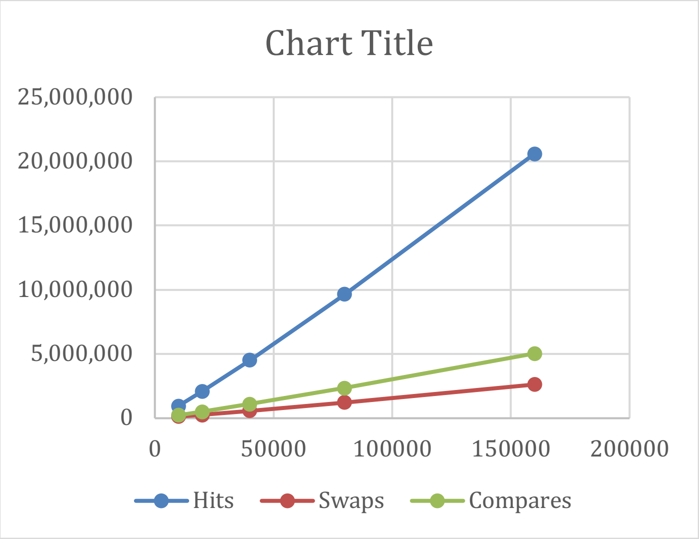
ArraySize	Hits	Swaps	Compares	Time
10000	411,697	65,120	155,548	6.51
20000	898,130	140,043	338,518	6.35
40000	1,921,465	303,304	725,594	19.64
80000	4,239,749	672,753	1,583,484	30.78
160000	8,911,442	1,398,162	3,388,300	66.67

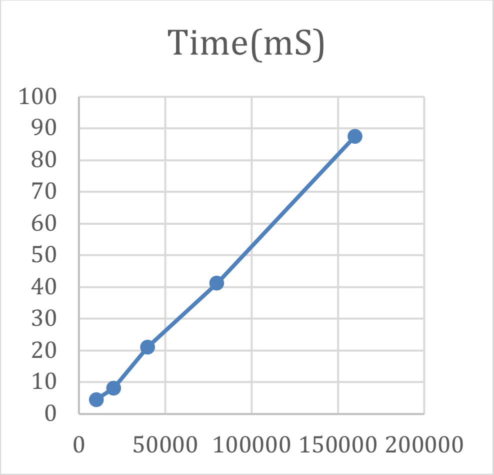




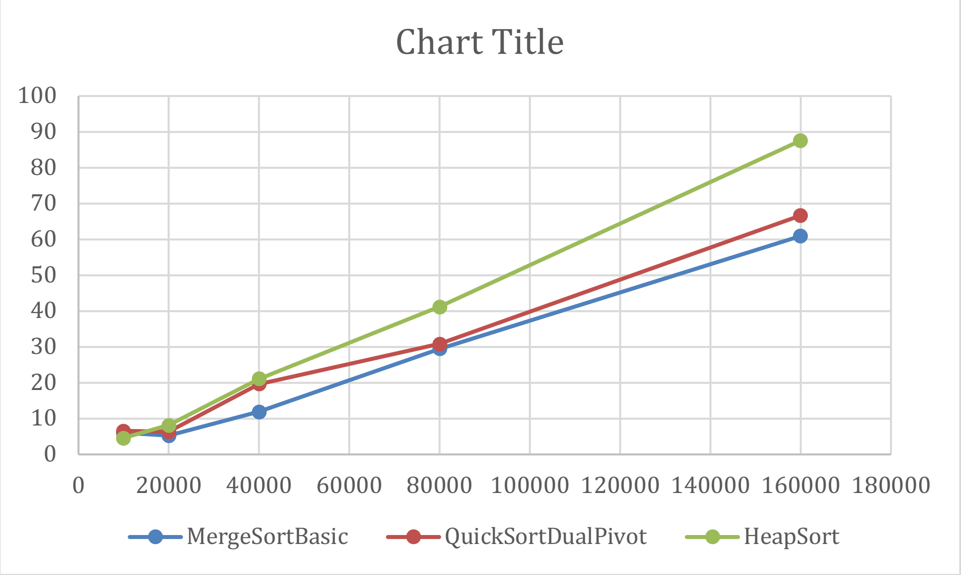
HeapSort

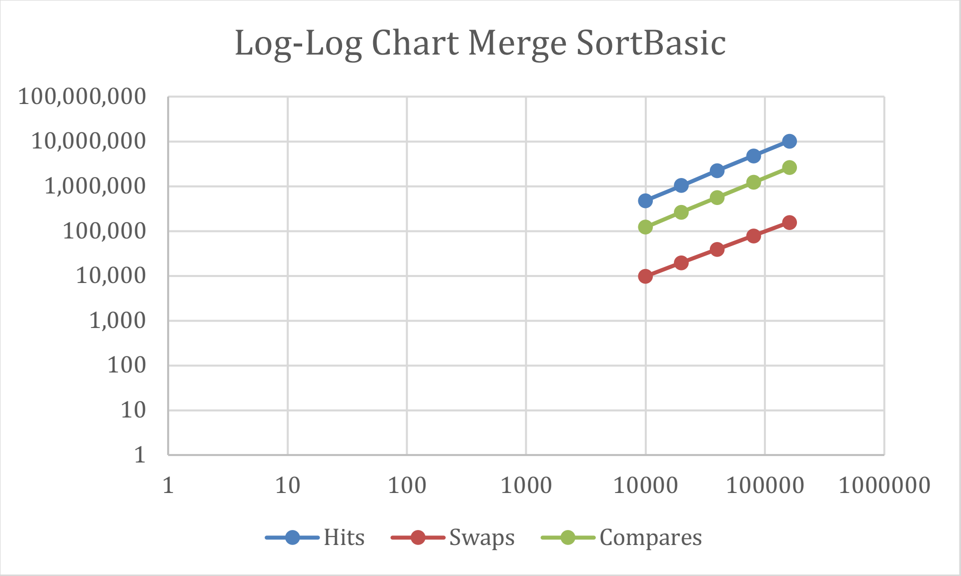
ArraySize	Hits	Swaps	Compares	Time(mS)
10000	967,603	124,211	235,380	4.6
20000	2,095,008	268,384	510,736	8.11
40000	4,510,221	576,788	1,101,534	21.12
80000	9,660,331	1,233,589	2,362,987	41.22
160000	20,599,776	2,627,047	5,045,794	87.58

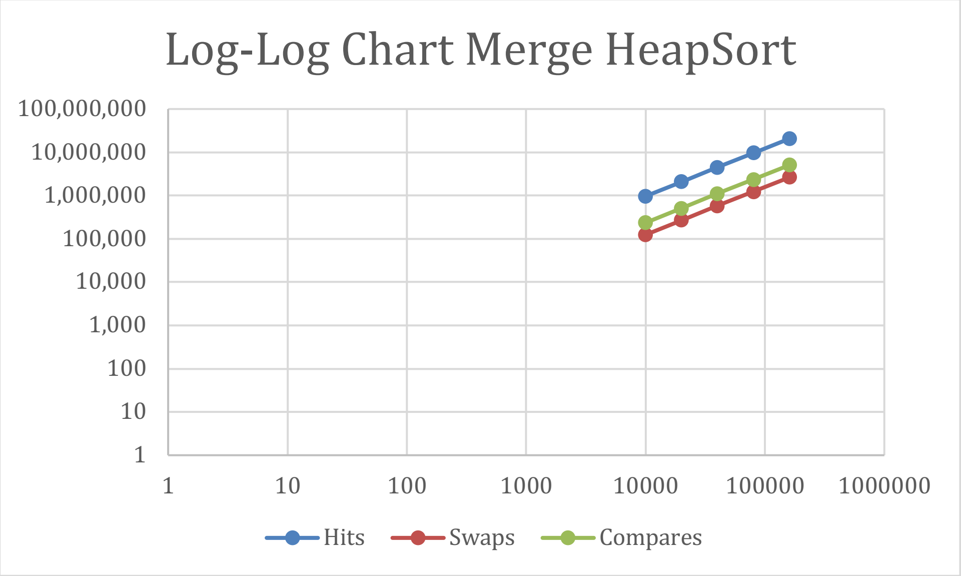


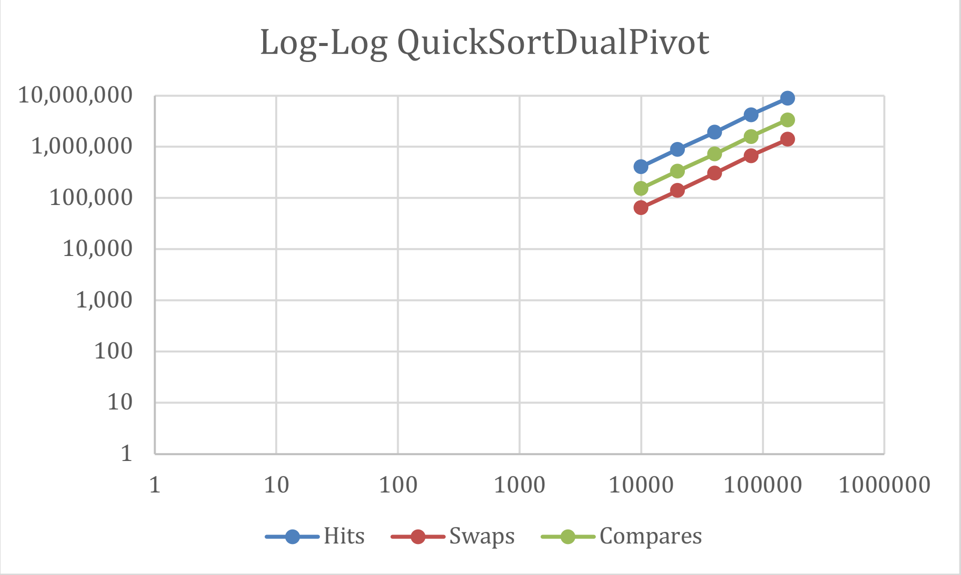


ArraySize	**MergeSortBasic**	**QuickSortDualPivot**	**HeapSort**
10000	5.99	6.51	4.6
20000	5.26	6.35	8.11
40000	11.86	19.64	21.12
80000	29.45	30.78	41.22
160000	60.94	66.67	87.58

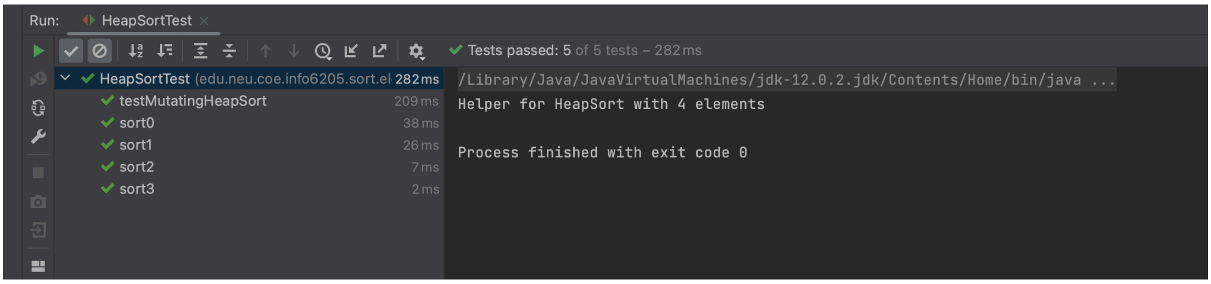



Log-Log Chart Merge SortBasic
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
1
10
100
1000
10000
100000
1000000
Hits
Swaps
Compares


Log-Log Chart Merge HeapSort
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
1
10
100
1000
10000
100000
1000000
Hits
Swaps
Compares


Log-Log QuickSortDualPivot
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
1
10
100
1000
10000
100000
1000000
Hits
Swaps
Compares

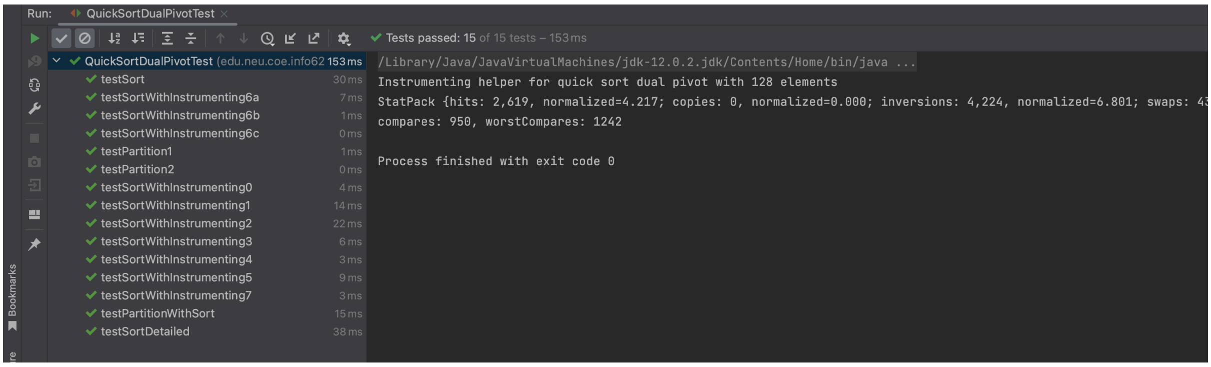
Unit Test Screenshots:



✓ Tests passed: 5 of 5 tests – 282 ms

```
✓ HeapSortTest (edu.neu.coe.info6205.sort.el 282 ms
    ✓ testMutatingHeapSort                   209 ms
    ✓ sort0                                   38 ms
    ✓ sort1                                   26 ms
    ✓ sort2                                    7 ms
    ✓ sort3                                    2 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
Helper for HeapSort with 4 elements

Process finished with exit code 0
```



✓ Tests passed: 15 of 15 tests – 153 ms

```
✓ QuickSortDualPivotTest (edu.neu.coe.info62 153 ms
    ✓ testSort                               30 ms
    ✓ testSortWithInstrumenting6a             7 ms
    ✓ testSortWithInstrumenting6b             1 ms
    ✓ testSortWithInstrumenting6c             0 ms
    ✓ testPartition1                          1 ms
    ✓ testPartition2                          0 ms
    ✓ testSortWithInstrumenting0              4 ms
    ✓ testSortWithInstrumenting1             14 ms
    ✓ testSortWithInstrumenting2             22 ms
    ✓ testSortWithInstrumenting3              6 ms
    ✓ testSortWithInstrumenting4              3 ms
    ✓ testSortWithInstrumenting5              9 ms
    ✓ testSortWithInstrumenting7              3 ms
    ✓ testPartitionWithSort                  15 ms
    ✓ testSortDetailed                       38 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
Instrumenting helper for quick sort dual pivot with 128 elements
StatPack {hits: 2,619, normalized=4.217; copies: 0, normalized=0.000; inversions: 4,224, normalized=6.801; swaps: 43
compares: 950, worstCompares: 1242

Process finished with exit code 0
```