NAME: Sharun Kumar Kakkad Sasikumar
NUID: 002774079

Task: to determine the best predictor of total execution time for sorting algorithms by running benchmarks for merge sort, dual-pivot quick sort, and heap sort.

Relationship Conclusion:
Through analysis of the statistics from the instrumentation techniques and comparing them vs the runtimes when instrumentation is not used, it can be safely concluded that Hits, Swaps, and compares can be used as predictors for the runtime for merge sort, dual pivot quick sort and heap sort, with number of hits giving a more accurate prediction.
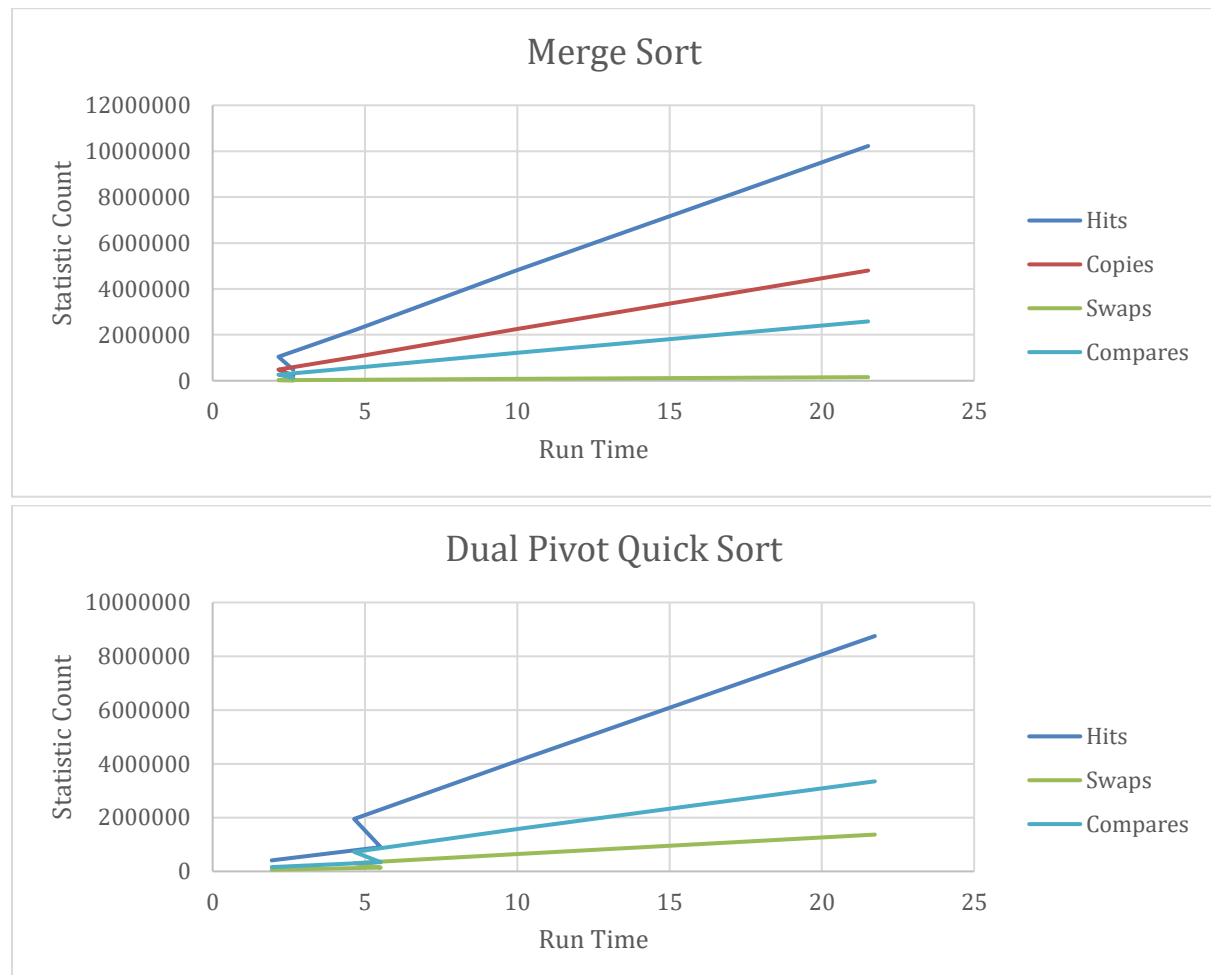
- Merge Sort: $\frac{runtime}{hits} * 1e5 \cong 0.24$

- Dual Pivot Quick Sort: $\frac{runtime}{hits} * 1e5 \cong 0.24$

- Heap Sort: $\frac{runtime}{hits} * 1e5 \cong 0.14$
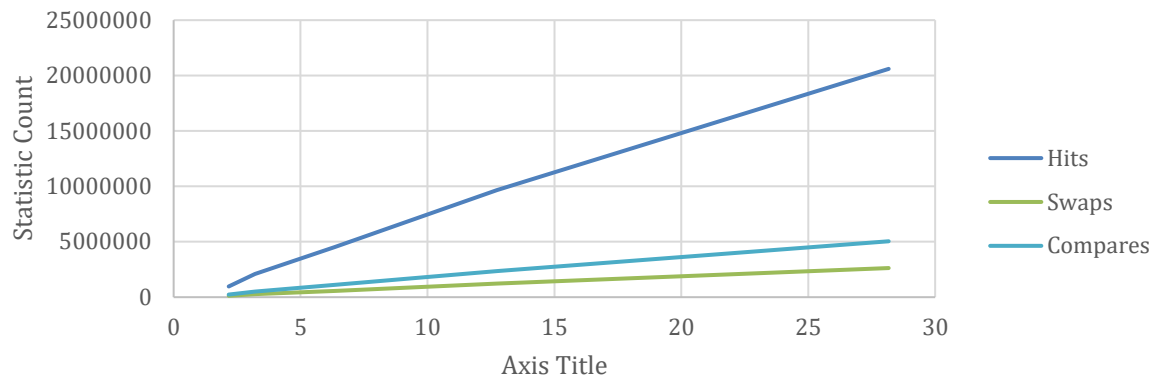
Code:
Submitted to GitHub Repository: https://github.com/sharunkumar-ks/INFO6205/pull/5/files

Graphical Representation:
Complete data is available in the **Hits as time predictor.xlsx** file.

Heap Sort

Unit Tests:

```
/*
 * Copyright (c) 2017. Phasmid Software
 */

package edu.neu.coe.info6205.sort.elementary;

import ...

/ALL/
public class HeapSortTest {

    no usages  akshaymatad
    @Test
    public void sort0() throws Exception {
        final List<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        Integer[] xs = list.toArray(new Integer[0]);
```

config.ini:
```
cutoff =

[instrumenting]
# The options in this section apply only i
# This slows everything down a lot so kee
inversions = 0
swaps = true
compares = true
copies = true
fixes = true
hits = true

[benchmarkstringsorters]
words = 1000
runs = 1000
mergesort = false
mergesortbasic = true
timsort = false
quicksort = false
```

Run — HeapSortTest:
```
"C:\Program Files\Java\jdk-19\bin\java.exe" ...
Helper for HeapSort with 4 elements

Process finished with exit code 0
```

Tests passed: 5 of 5 tests – 1 sec 372 ms
- testMutatingHeapSort 692 ms
- sort0 471 ms
- sort1 6 ms
- sort2 200 ms
- sort3 1 ms