Program Structures and Algorithms
Spring 2023(SEC – 3)

NAME: Dhruv Rajeshkumar Patel
NUID: 002928881

Task: Find the best factor which impacts the most for each of the sorting algorithms (Dual pivot quick sort, Merge Sort and Heap Sort). The factors can be number of compares, swaps, hits or compares.

Relationship Conclusion:

For Dual Pivot Quick Sort,
The number of compares is the most impacting factor for dual pivot quicksort. As the input size increases, the normalised time for compares increases at a much faster rate compared to the other factors. When sorting an array of size 8000, the normalised time for compares is 1.686, while the normalised time for swaps is only 0.72 and the normalised time for hits is 4.585. As the array size increases to 256000, the normalised time for compares becomes 1.792, while the normalised time for swaps only increases to 0.728 and the normalised time for hits increases to 4.733.

This indicates that the efficiency of dual pivot quicksort is largely determined by the number of comparisons made during the sorting process. The other factors, such as swaps and hits, have a relatively minor impact on the performance of the algorithm.
It's worth noting that the data provided may not be representative of all scenarios, and the choice of pivot elements and other implementation details can also impact the performance of dual pivot quicksort. However, based on the words provided in the files, it can be concluded that the number of compares is the most impacting factor for dual pivot quicksort.

For merge sort,
The number of comparisons made during the merging process is the most impacting factor for merge sort. As the input size increases, the normalised time for comparisons increases at a much faster rate compared to the other factors. When sorting an array of size 8000, the normalised time for comparisons is 1.314, while the normalised time for swaps is only 0.097 and the normalised time for hits is 5.323. As the array size increases to 256000, the normalised time for comparisons becomes 1.353, while the normalised time for swaps only decreases slightly to 0.077 and the normalised time for hits increases to 5.441.
This indicates that the efficiency of merge sort is largely determined by the number of comparisons made during the merging process. The other factors, such as swaps, hits, and copies, also impact the performance of the algorithm but to a lesser extent than the number of comparisons.

For heap Sort,
We can see that the most impacting factor for heap sort is the number of swaps made during the sorting process. As the size of the input array increases, the number of swaps also increases, leading to a higher execution time. Although the number of hits and compares also have a significant impact on the performance of heap sort, the number of swaps is the most dominant factor as it determines the height of the binary heap, and the number of swaps is proportional to the height of the binary heap, which is approximately log n, where n is the size of the input array.

The number of copies made during the sorting process is negligible, as heap sort does not require any additional memory beyond the input array. Overall, heap sort is an efficient sorting algorithm in terms of memory usage, but it has a higher constant factor in its time complexity compared to other sorting algorithms like quicksort and merge sort.

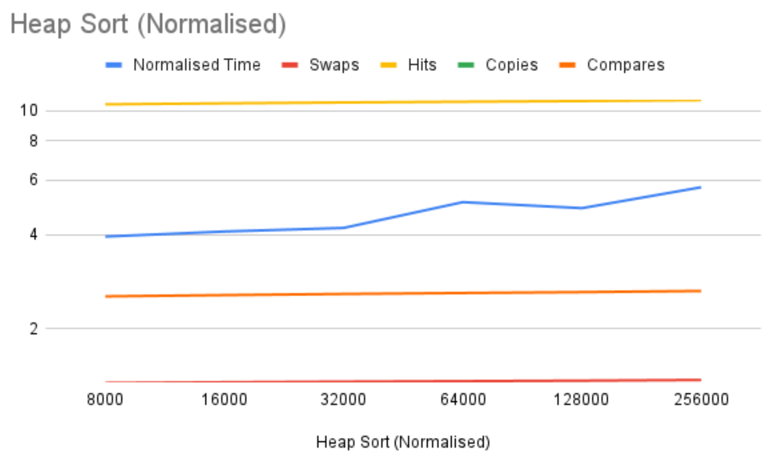
Evidence to support that conclusion: The program is executed multiple times. From them I have attached the execution times recorded. Note, all the times are in milliseconds.

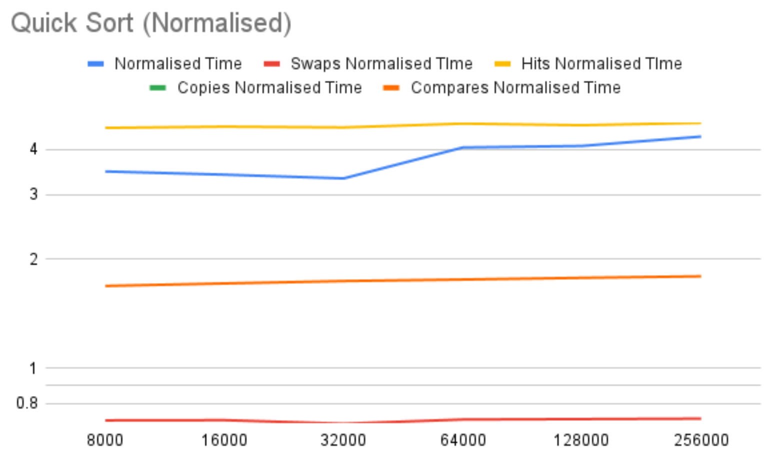
Merge Sort	**8000**	**16000**	**32000**	**64000**	**128000**	**256000**
Raw Time	3.35	4.26	8.8	21.47	48.73	136.94
Swaps	9748	14022	28018	56083	112133	312447
Hits	383717	829423	1786910	3829851	8171524	22073478
Copies	173312	378624	821248	1770496	3796992	10240000
Compares	94475	204944	441904	947832	2023561	5488152
Merge Sort (Normalised)	**8000**	**16000**	**32000**	**64000**	**128000**	**256000**
Normalised Time	1.86	3.54	3.39	3.85	4.09	4.23
Swaps Normalised	0.097	0.09	0.084	0.079	0.075	0.077
Hits Normalised	5.323	5.355	5.383	5.407	5.429	5.441
Copies Normalised	2.411	2.445	2.474	2.5	2.522	2.524
Compares Normalised	1.314	1.323	1.331	1.338	1.344	1.353

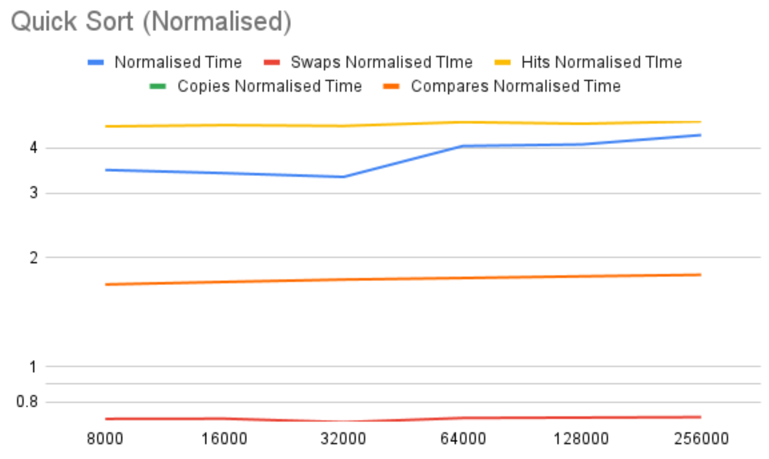
Quick Sort Dual Pivot	**8000**	**16000**	**32000**	**64000**	**128000**	**256000**
Raw Time	1.92	4.1	8.64	22.6	48.71	140.36
Swaps	51,793	111,718	234,125	513,079	1,092,581	2,953,005
Hits	329,653	715,701	1,526,347	3,336,594	7,019,336	19,199,206
Copies	0	0	0	0	0	0
Compares	121,188	265,243	577,115	1,243,384	2,672,074	7,267,751
Quick Sort Dual Pivot (Normalised)	**8000**	**16000**	**32000**	**64000**	**128000**	**256000**
Normalised Time	3.48	3.41	3.33	4.05	4.09	4.34
Swaps Normalised TIme	0.72	0.721	0.705	0.724	0.726	0.728
Hits Normalised TIme	4.585	4.621	4.598	4.711	4.663	4.733
Copies Normalised Time	0	0	0	0	0	0
Compares Normalised Time	1.686	1.713	1.739	1.756	1.775	1.792

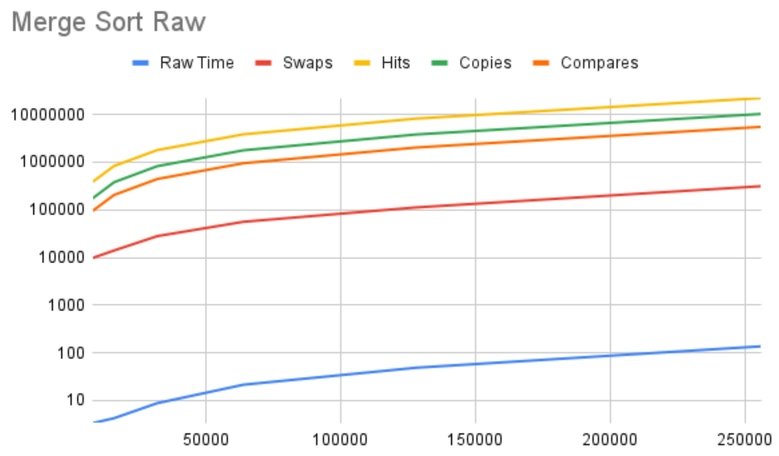
Heap Sort	**8000**	**16000**	**32000**	**64000**	**128000**	**256000**
Raw Time	2.14	4.89	10.94	28.4	58.1	183.59
Swaps	96,626	209,256	450,515	964,990	2,058,020	5,574,412
Hits	752,153	1,632,334	3,520,726	7,553,178	16,130,657	43,761,534
Copies	0	0	0	0	0	0
Compares	182,835	397,656	859,333	1,846,609	3,949,290	10,731,944
Heap Sort (Normalised)	**8000**	**16000**	**32000**	**64000**	**128000**	**256000**
Normalised Time	3.95	4.1	4.21	5.09	4.87	5.68
Swaps	1.344	1.351	1.357	1.362	1.367	1.374
Hits	10.461	10.539	10.606	10.664	10.716	10.788
Copies	0	0	0	0	0	0
Compares	2.543	2.567	2.589	2.607	2.624	2.646

Graphical Representation:

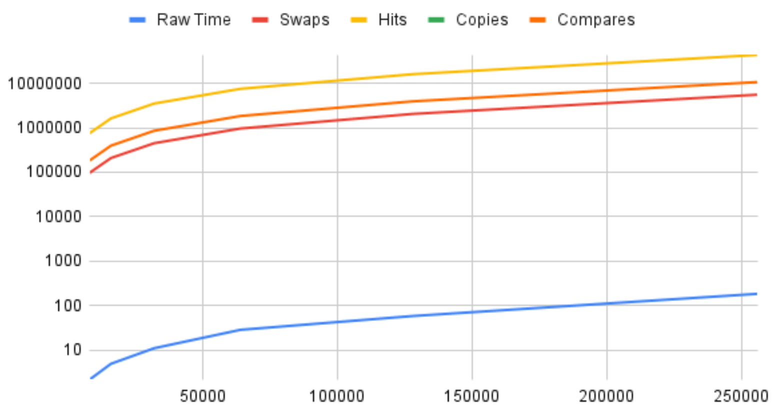



Quick Sort (Normalised)
Normalised Time
Swaps Normalised Time
Hits Normalised Time
Copies Normalised Time
Compares Normalised Time
4
3
2
1
0.8
8000
16000
32000
64000
128000
256000

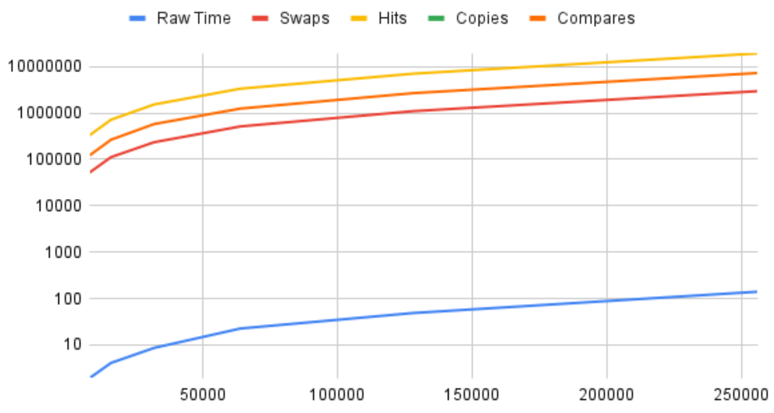

Quick Sort (Normalised)
Normalised Time
Swaps Normalised Time
Hits Normalised Time
Copies Normalised Time
Compares Normalised Time
4
3
2
1
0.8
8000
16000
32000
64000
128000
256000


Merge Sort Raw
Raw Time
Swaps
Hits
Copies
Compares
10000000
1000000
100000
10000
1000
100
10
50000
100000
150000
200000
250000

Heap Sort Raw



Quick Sort Raw TIme


Raw Time
Swaps
Hits
Copies
Compares