Aliza Siddiqui

CSC 1351

Time Complexity Analysis of Different Sorting Algorithms

As the world grows more complex, humans need to process a massive amount of data at an efficient speed. As computer scientists, it is our duty to identify the best algorithms that are not computationally expensive and can appropriately get the tasks done --- such as sorting algorithms.

Currently, the most popular sorting algorithms include Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and so on. In this case, I compare eight different sorting algorithms in terms of their time complexity according to the amount of data (size of list).

Bubble sort , Selection sort, and Insertion sort are popular due to their simplicity however their performance becomes lacking when it comes to larger lists of items. Merge and Quick sort are significantly better when it comes to larger amounts of data however, both algorithms require space due to their nature: Merge Sort requires more memory space and Quick Sort requires stack space due to recursion.

According to the data, Heap Sort, Quick Sort, and Merge Sort time complexities are very close in value.

Time Complexity Analysis of Sorting Algorithms

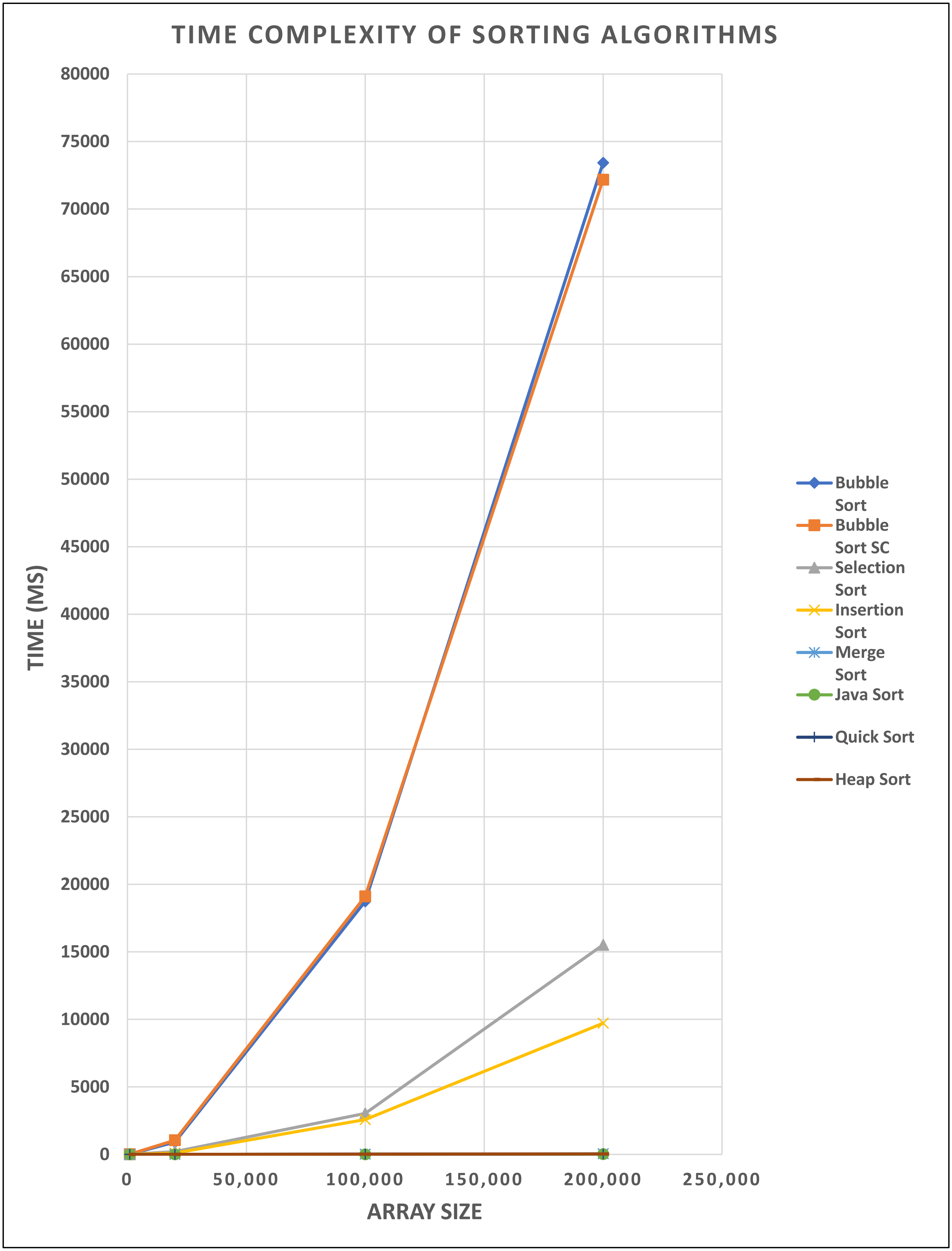
Time (milliseconds)

Array Size	1000	20,000	100,000	200,000
Bubble Sort	16	902	18,740	73,431
Bubble CS	14	1,049	19,117	72,186
Selection Sort	5	208	3,029	15,520
Insertion Sort	6	93	2,587	9,713
Merge Sort	1	7	24	46
Quick Sort	1	5	22	34
Java Sort	1	7	8	20
Heap Sort	1	6	20	35

1.) The Time complexity of the Heap Sort is known to be O(n log n) for both best- and worst-case scenarios which closely mirrors the complexity of Merge and Quick Sort.

2.) The Heap sort is unique in that it utilizes a binary tree and a heap which allows it to work without using additional memory. Using the list, it first builds the tree, then transforms the heap into a maximum heap (where the parent nodes are always greater than or equal to the child nodes), swaps the root node with the last node and finally removes the root from the heap. From here, it repeats this process until the list is sorted.
While Quick and Merge Sort require some memory to work, Heap Sort has minimal memory usage apart from holding the initial list of items to be sorted. Also, Heap sort time complexity is O(n log n) for both best- and worst-case scenarios while Quick Sort can go up to O(n^2) time

depending on the list. So, while the time complexity of this sort may closely mirror that of the Merge and Quick Sort, Heap sort is known to be the most efficient sorting algorithm of all.


TIME COMPLEXITY OF SORTING ALGORITHMS
TIME (MS)
80000
75000
70000
65000
60000
55000
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0
50,000
100,000
150,000
200,000
250,000
ARRAY SIZE
Bubble Sort
Bubble Sort SC
Selection Sort
Insertion Sort
Merge Sort
Java Sort
Quick Sort
Heap Sort