Comparison between quick, merge and heap sort

Azizbek Mukhammadjonov

December 20, 2022

1. Introduction

In this paper, we investigate a classical problem: sorting. The problem is to arrange an array of n integers according to some total order which is computed in O(1). We use < as the total order. In this paper we compare three comparison-based sorting algorithms: quick sort, merge sort, and heap sort

1.1 Quick sort

The Quick sort algorithm works by selecting a pivot element from the input array and partitioning the array around it. All elements less than the pivot are placed to the left of it and all elements greater than the pivot are placed to the left of it and all elements greater then the pivot are placed to the right. The pivot is then placed in its correct position in the sorted array. The algorithm is then applied recursively to the left and right halves of the array until it is fully sorted.

1.2 Heap sort

The Heap Sort algorithm begins by constructing a max heap from the input array. This is done by repeatedly inserting each element into the heap and ensuring that the max heap property is maintained. Once the heap is constructed, the algorithm repeatedly removes the root node (which is the largest element) from the heap and inserts it into the sorted array. This process continues until the heap is empty, at which point the sorted array will contain all of the elements in ascending order.

1.3 Merge sort

The Merge Sort algorithm works by dividing the input array into two halves, sorting each half using the same algorithm, and then merging the two halves back together in sorted order. The merging process is done by comparing the elements of the two halves and inserting them into a temporary array in sorted order. Once all of the elements have been inserted into the temporary array, they are copied back into the original array. This process is repeated recursively until the entire array is sorted.

2. Methodology

The C++ implementation of the algorithms was tested on sorted and randomly shuffled arrays of increasing sizes ranging from 0 to 2500. The test was repeated 100 times for each array size. The results show the average time it took for each algorithm to sort the given input data.

3. Results

Graphs of results for the average case of all sorting algorithms are presented in Figure 1 and Figure . The Heap sort takes several times longer than the other sorting algorithms. Quick sort takes the least time and merge sort takes little bit longer that quick sort .

4. Conclusion

Overall, these three algorithms provide different trade-offs in terms of efficiency and complexity. Merge sort and Quick sort are the best sorting methods when it comes to time , especially the Quck sort. On the other hand Heap sort takes more time that other methods .

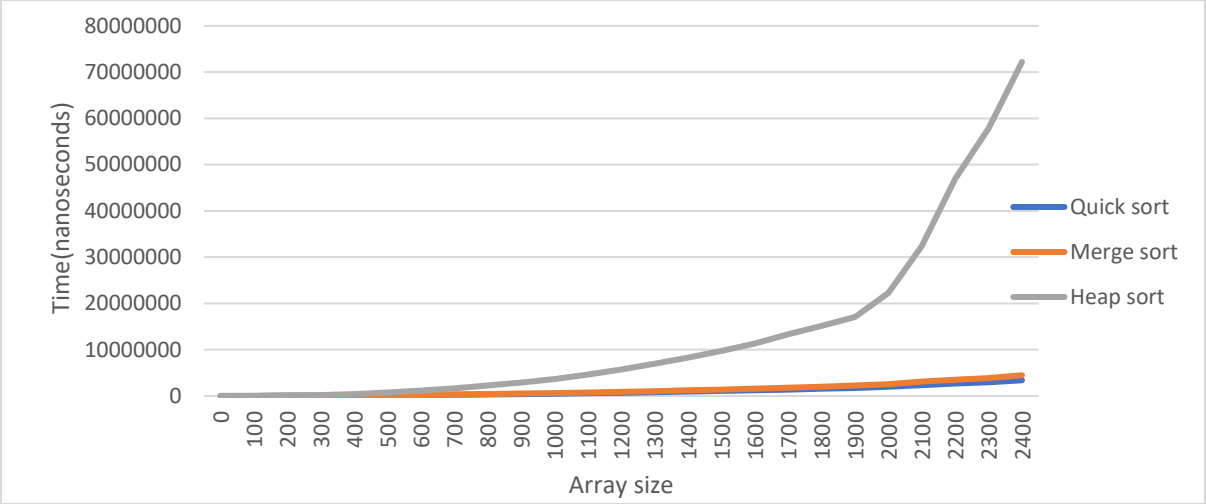

Figure 1

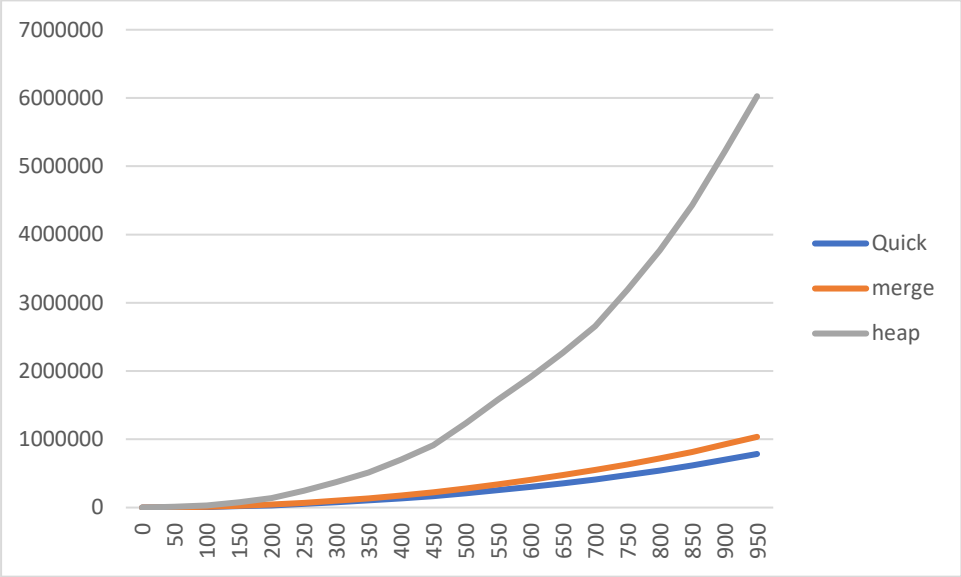

Figure 2

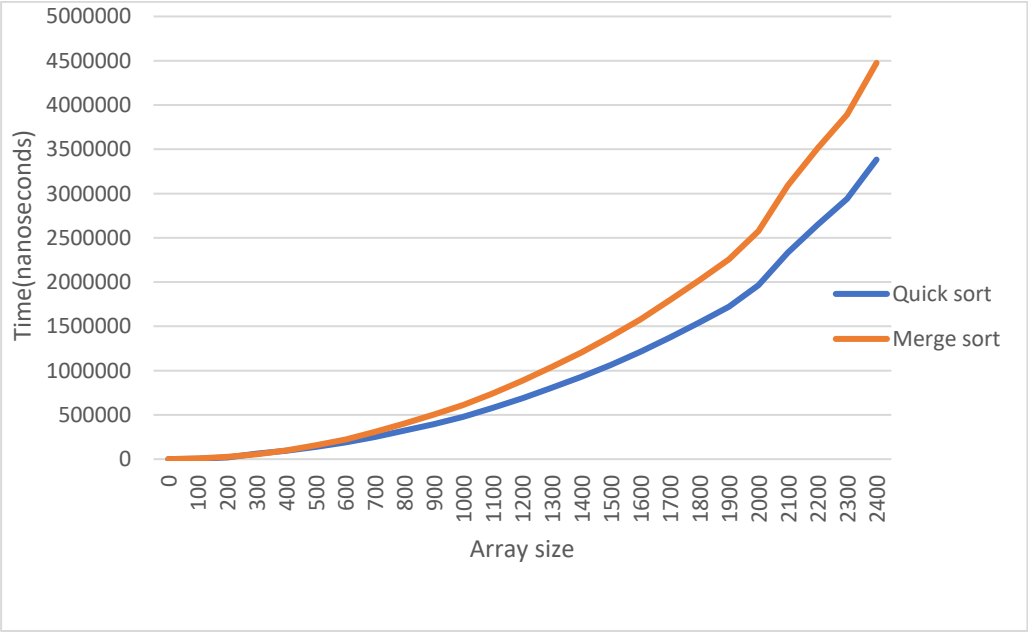

Figure 3(only quick and merge sort)