Comparison between quicksort, heapsort and mergesort algorithms

Najmiddin Nazarmatov

20.12.2022

1.Introduction

In this paper we are going to investigate the common problem: sorting arrays. To do it we will compare four most used algorithms: quicksort, heapsort, mergesort and insertion sort.

1.1. Quicksort

Quick sort is a popular and efficient sorting algorithm that works by selecting a "pivot" element from the input array, partitioning the array around the pivot, and then recursively sorting the sub-arrays on either side of the pivot. It has an average case time complexity of O(n*log(n)), which makes it well-suited for sorting large arrays and lists.

1.2. Heapsort

Heap sort is a comparison-based sorting algorithm that works by building a binary heap data structure from the input array, and then repeatedly extracting the maximum element (for a max-heap) or the minimum element (for a min-heap) and placing it at the end of the sorted array. It has a time complexity of O(n*log(n)), which makes it well-suited for sorting large arrays and lists.

1.3. Mergesort

Merge sort is a comparison-based sorting algorithm that works by dividing the input array into smaller sub-arrays, sorting the sub-arrays independently, and then merging the sorted sub-arrays back together to form a sorted array. It has a time complexity of O(n*log(n)), which makes it well-suited for sorting large arrays and lists.

1.4.Insertion Sort

Insertion sort is a simple sorting algorithm that works by iterating through an input list, taking one element at a time and inserting it into its correct position in a sorted list. Insertion sort has a time complexity of O(n^2) in the worst case, making it less efficient than other sorting algorithms. However, it is a stable sort and has a low overhead, making it a good choice for sorting small lists or partially sorted lists.

2. Methodology

The task was completed in C++. Results are created for randomly shuffled array. The algorithms were tested on arrays with size from 100 to 10000 with 100 steps. Each size of array was tested 100 times. Each algorithm was given the same input array. The result for each array is average time it took to run the method.

3.Results

Graphs of results for average case of all sorting algorithms are presented in Figure 1 and Figure 2. As shown in both these figures, quicksort and heapsort is much faster than both insertion sort and mergesort. While insertion sort is significantly slower than all remaining algorithms, Figure 2 shows us it can be even faster than quicksort in small sized arrays. Since it is hard to see comparison between other three methods, lets have a closer look into them, without insertion sort (Figure 3). Here we can see that quicksort is more than 2 times faster than heapsort and more than 4 times faster than mergesort.

4.Conclusion

Overall, quicksort performed best in all aspects of sorting array. Heapsort also performed very well, but mergesort’s performance is very poor. Considering all these three algorithms have an average of O(n*log(n)) time complexity, these results are completely unexpected. As insertion sort has O(n^2) time complexity, it is predictible that it performes significantly worse than other methods. But according to results, we conclude that “Quicksort” is the best of these three algorithms.

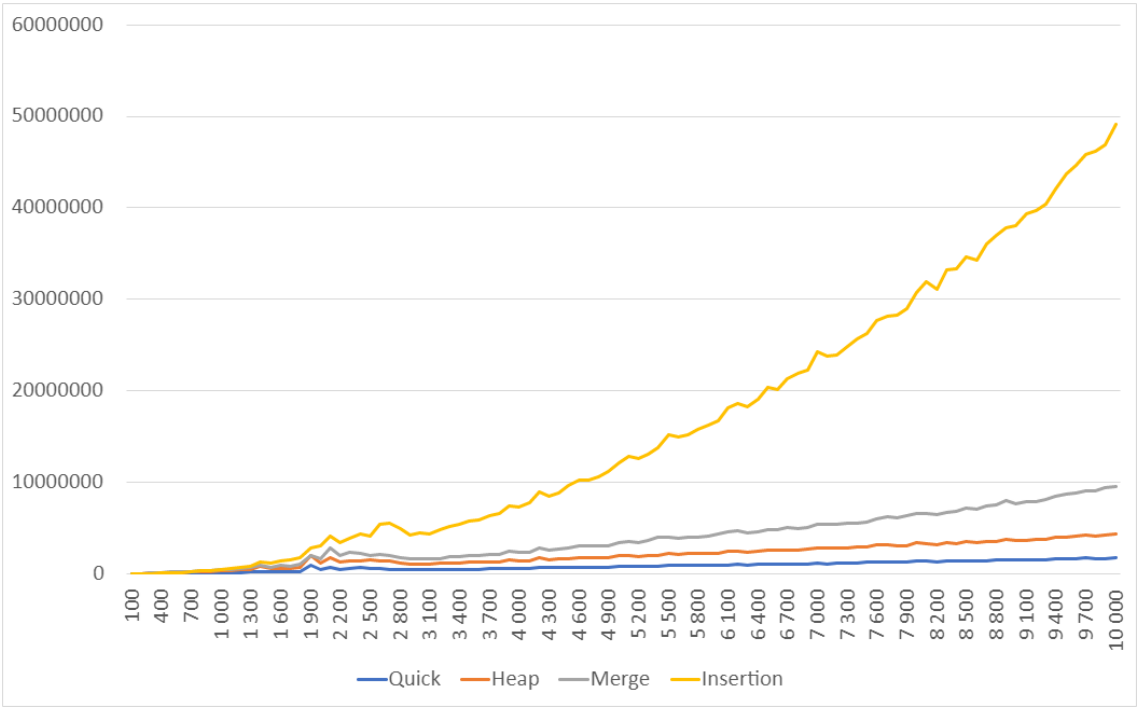

Figure 1. Array size from 100 to 10000

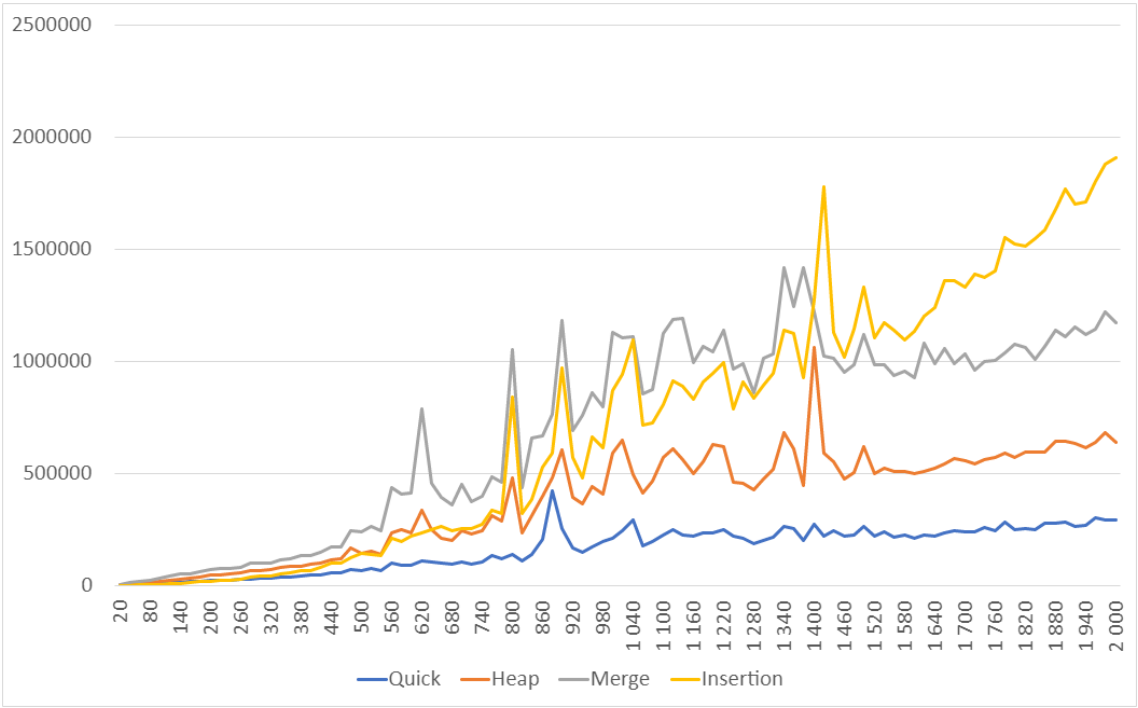

Figure 2. Array size from 10 to 1000

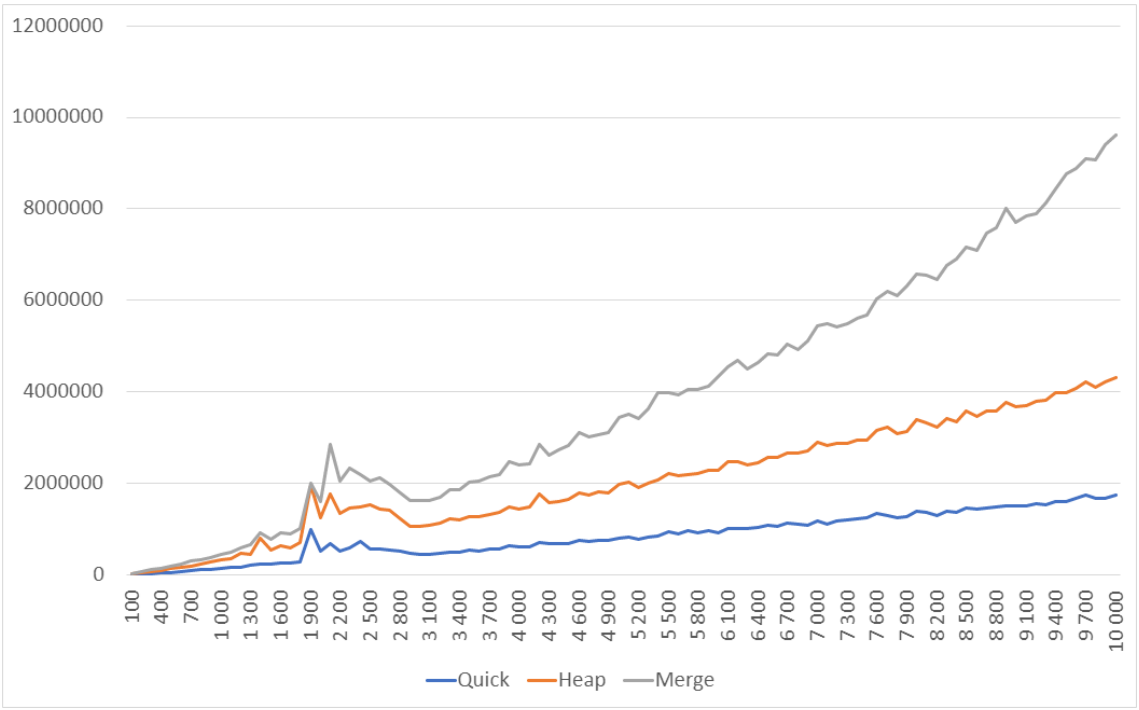

Figure 3. Quicksort, heapsort and mergesort excluding insertion sort