Sorting. Comparison between Insertion, Quick, Radix and Heap sorts

Roman Rypiak

December 10, 2023

I. Introduction

In this paper we analyze the regular sorting problem – arranging an array of non-negative integers in the ascending order. We shall use four different algorithms, such as: insertion sort, quick sort, radix sort and heap sort.

i. insertion sort

Insertion Sort is a simple sorting algorithm that builds the final sorted array one item at a time. When it comes to larger inputs, insertion sort is much less efficient than other to-be-mentioned algorithms.

The algorithm starts with both unsorted and sorted parts (latest, at the very beginning, contains only one element – the first element of the list). For each element in the unsorted portion, it is compared with elements in the sorted portion. The element is then moved to its correct position in the sorted part, rearranging other elements if necessary. The entire process is repeated until the complete list becomes sorted.

- Average and Worst cases: O(n^2)
- Best case (*already sorted*): O(n)

ii. quick sort

Quick sort chooses a pivot element from the array. It then rearranges the array such that smaller elements than the pivot are on the left side, and greater than the pivot - on the right. Pivot is placed on its final sorted position. This process is applied recursively on both left and right sides until the list is sorted.

- Average case: O(n * log(n))
- Worst case (*pivot is poorly chosen*): O(n^2)

iii. radix sort

Radix Sort is a non-comparative sorting algorithm that works by placing elements into cells according to their individual digits. It processes the numbers from the least significant digit (LSD) to the most significant digit (MSD). Unfortunately, because it does not compare the elements directly, this algorithm could not be applied to negative integers (well, technically it could but then the implementation would be more complex).

- Average case: O(k * n) - where k is the maximum number of digits, and n is the number of elements.

iv. heap sort

Heap Sort is a sorting algorithm that transforms an input array into a heap and then repeatedly extracts the maximum element to build the sorted array. It requires only a constant amount of additional memory.

- Average case: O(n * log(n))

II. Methodology

The algorithms were implemented in Java. The results were generated in two categories: sorted and unsorted arrays. They were tested on arrays of size from 100, 200, 300, …, 10000. In both categories, each algorithm was given the same input data. Each array size was tested 100 times. The result for each size is the average time it took for each algorithm to sort the input data.

III. Results

Graphs of results for the average case of all sorting algorithms are presented in Figures 1 and 2. When it comes to the input array that was unsorted (Figure 1) from the very beginning, insertion sort takes much more time than other algorithms which. The graph that displays data for the arrays with the size up to two hundreds (Figure 3) shows us that insertion started linearly taking more time after 100-th element. The radix sort though, was the most efficient for unsorted input. I removed from the graph insertion sort (Figure 4) so the rest of the algorithms could be compared more peculiarly, and it looks like heap sort is less efficient than radix sort in few times.

What concerns sorted input (Figure 2), the worst one for such a scenario is quick sort with time complexity O(n^2) with insertion sort being extremely fast having O(n) time complexity. On Figure 5 it is seen that radix sort is a bit slower in this case but still faster than heap in approximately two times even though when it comes to smaller input (Figure 6) they have begun pretty similarly.

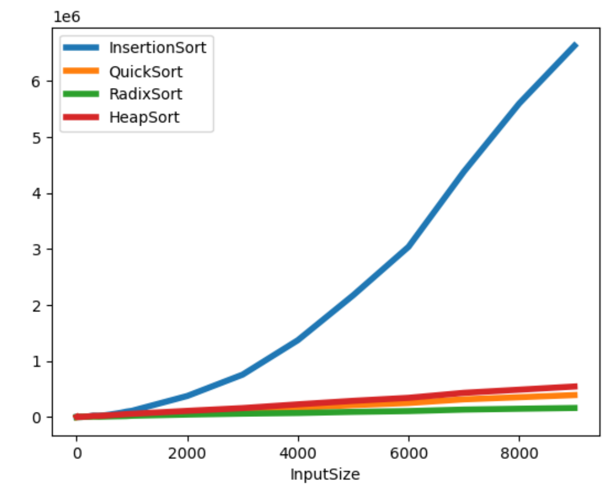

Figure 1. Unsorted array

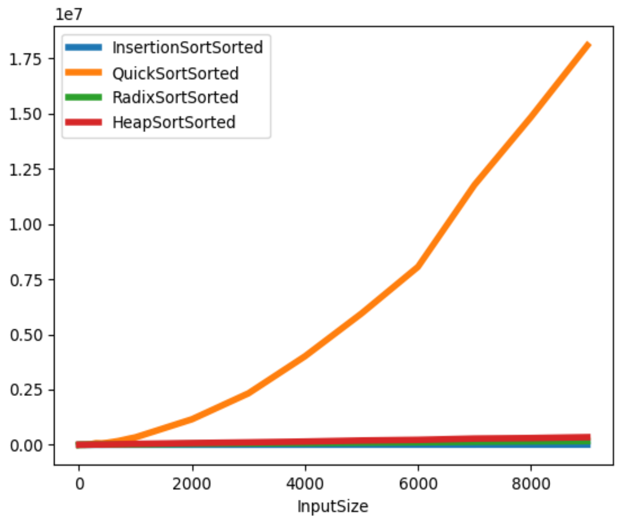

Figure 2. Sorted array

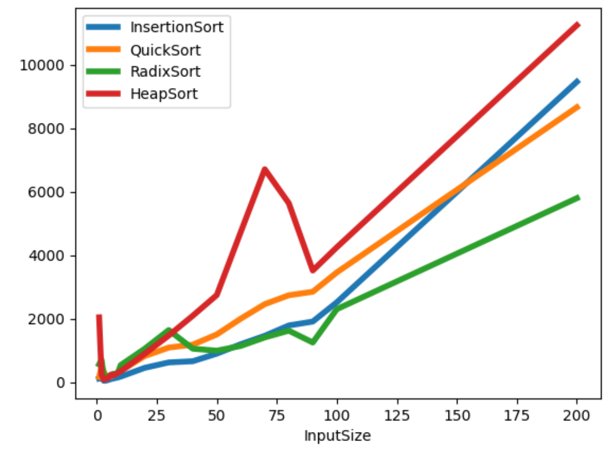

Figure 3. Unsorted array up to 200 elements

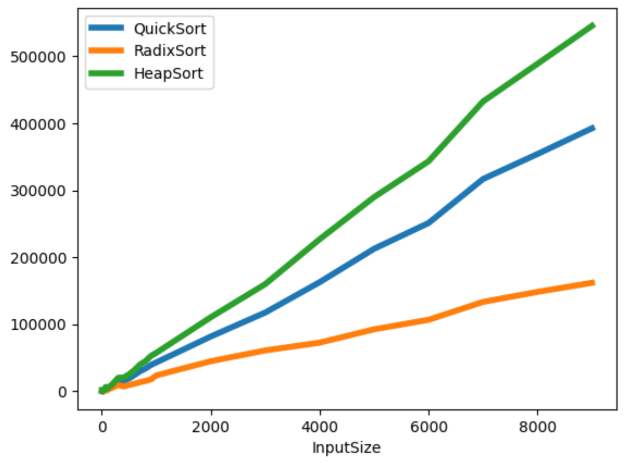

Figure 4. Unsorted array without InsertionSort

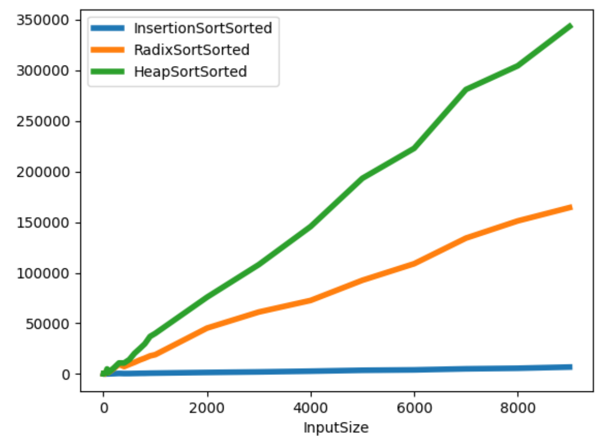

Figure 5. Sorted array without QuickSort

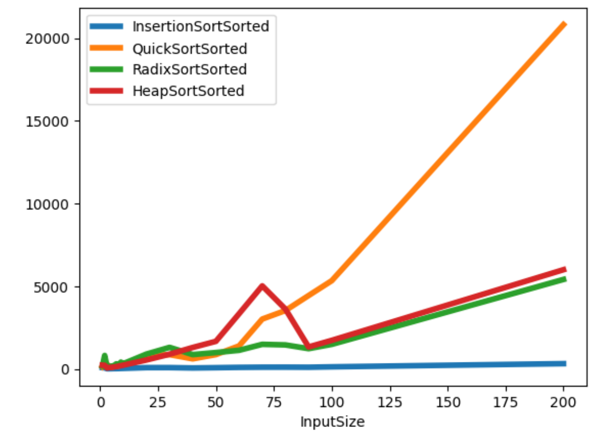

Figure 6. SortedArray up to 200 elements

IV. Conclusion

We conclude that using the tests described in methodology and implementations provided in the repository, radix sort is the best algorithm (for non-negative integers) when it comes to unsorted input and insertion sort for already sorted input.