Comparison between insertion sort, heap sort, quicksort, and radix sort

By Emin Rahimov, University of Lodz

1. Abstract

There are various sorting algorithms in programming. These algorithms should be explored to determine the best among them. The main purpose is to achieve efficient solutions. In this case, I have worked on the performance of sorting algorithms to identify the fastest one based on specific conditions.

2. Quick sort

According to the QuickSort algorithm, the program chooses a pivot element from the array. This can be the first, last, or any element of the array. I always prefer the first element as the pivot. The rearrangement process should then be performed: swap elements smaller than the pivot to the left and elements greater than the pivot to the right. After that, the same process is applied recursively, and the recursion stops when the arrays have only one element. QuickSort is efficient in practice and has an average time complexity of O(n log n). However, its worst-case time complexity is O(n^2) when the pivot selection consistently results in unbalanced partitions.

3. Heap sort

Heap is a binary tree structure, but we should not forget that it has differences from a binary search tree. There are two types: maximum and minimum. So what difference does having a maximum and minimum make? In the maximum heap type, the data is placed in a decreasing order from top to bottom, in other words, the element in the root node is the largest, while in the minimum heap, the data is placed in an increasing order from top to bottom, that is, the element in the root is the smallest. In this way, it provides the opportunity to sort data in ascending or descending order. When deletion is done using maximum heap, the root node is deleted and placed at the end of the array and sorted from smallest to largest, we can do the opposite by using minimum heap.

4. **Insertion sort**

Insertion Sort is a sorting algorithm used in computer science that creates a sorted array element by element at each step. When working with large arrays, quick sort works less efficiently than more advanced sorting algorithms such as merge sort and heap sort. The performance of insert sort is O(n2). The reason for this is that as many passes are required as the number of elements in the array, and at worst, as many shifts are required as the number of elements in each pass. So the worst case of insertion sort is a reverse sorted list.

5. **Radix sort**

Radix Sort, also known as Radix Sort, is a sorting method designed to arrange arrays of integers either in ascending order, from smallest to largest, or in descending order, from largest to smallest. It operates by sorting numbers according to their digits, making it one of the algorithms that focus on digit-based sorting. The Radix Sort algorithm is notably efficient and has been developed based on a binary (base 2) system. This algorithm is commonly referred to as radix sort or digit sort.

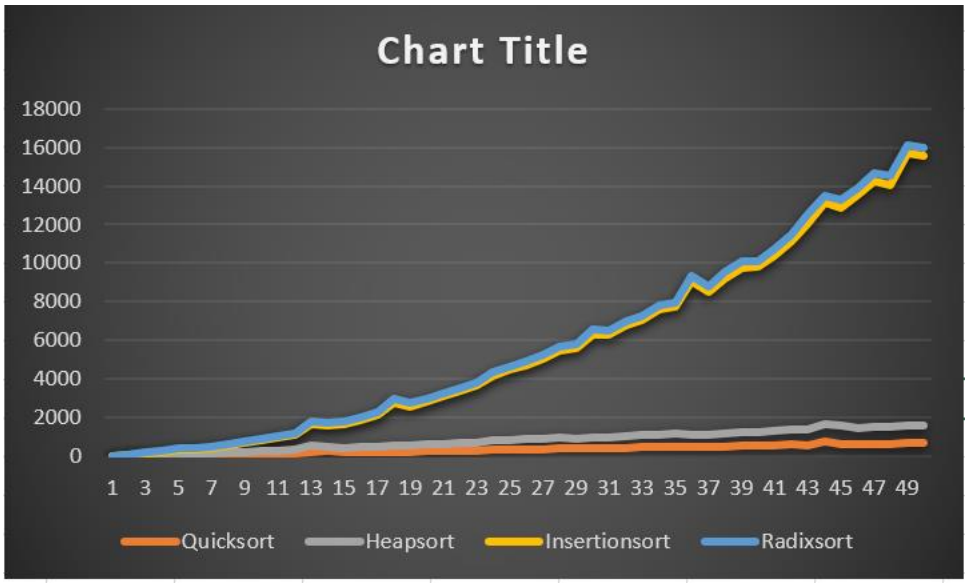

Illustration of comparison of sorting algorithms with high size array with random elements.

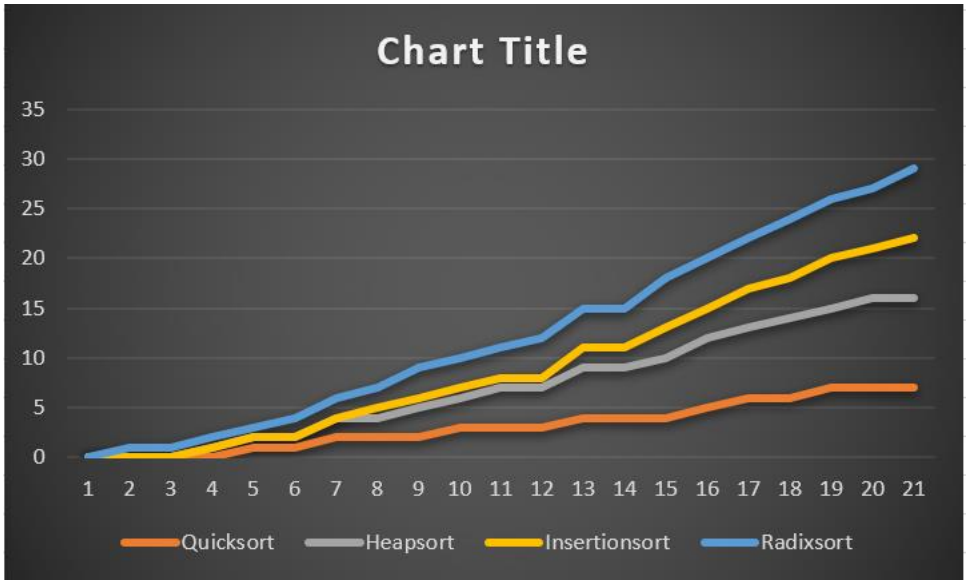

Illustration of comparison of sorting algorithms with small size array with random elements.

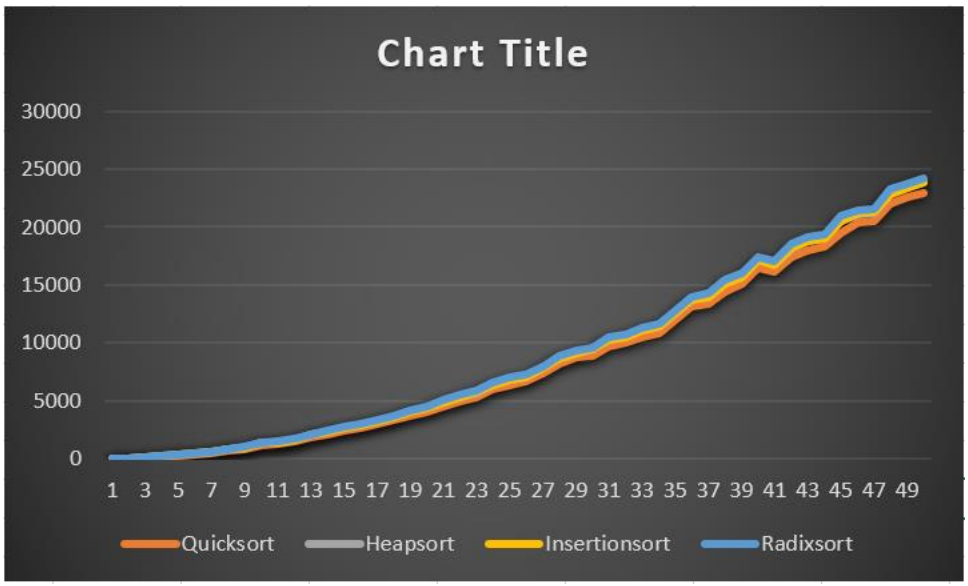

Illustration of comparison of sorting algorithms with high size sorted array

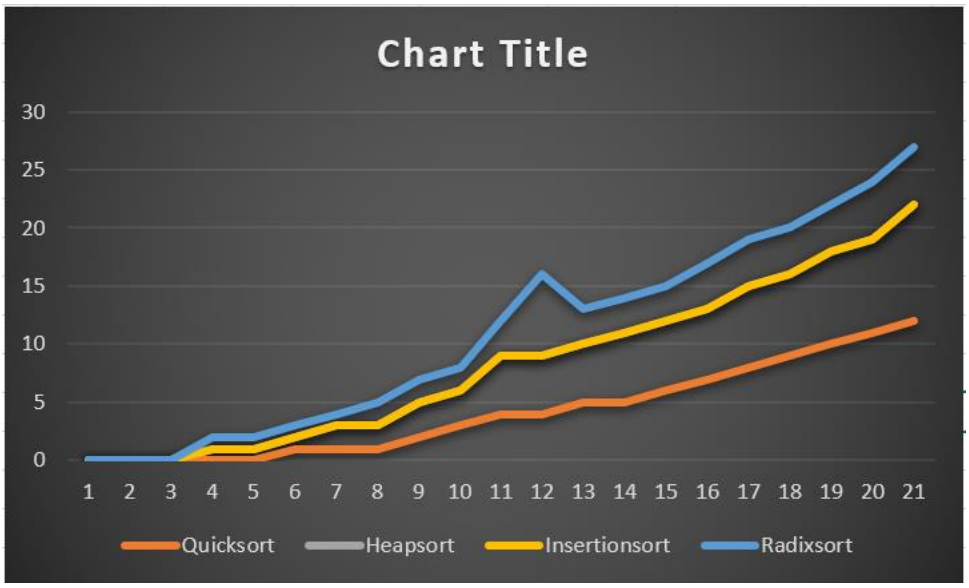

Illustration of comparison of algorithms with small size sorted array.

Conclusion:

It has been observed that the radix and insertion algorithms take more time compared to others when sorting high-dimensional arrays. For small sized arrays, the fastest algorithm is quick sort. It is obvious that the reordering time of the previously sorted arrays is very close to each other. I would also like to point out that different sequences are randomly given to each algorithm separately.