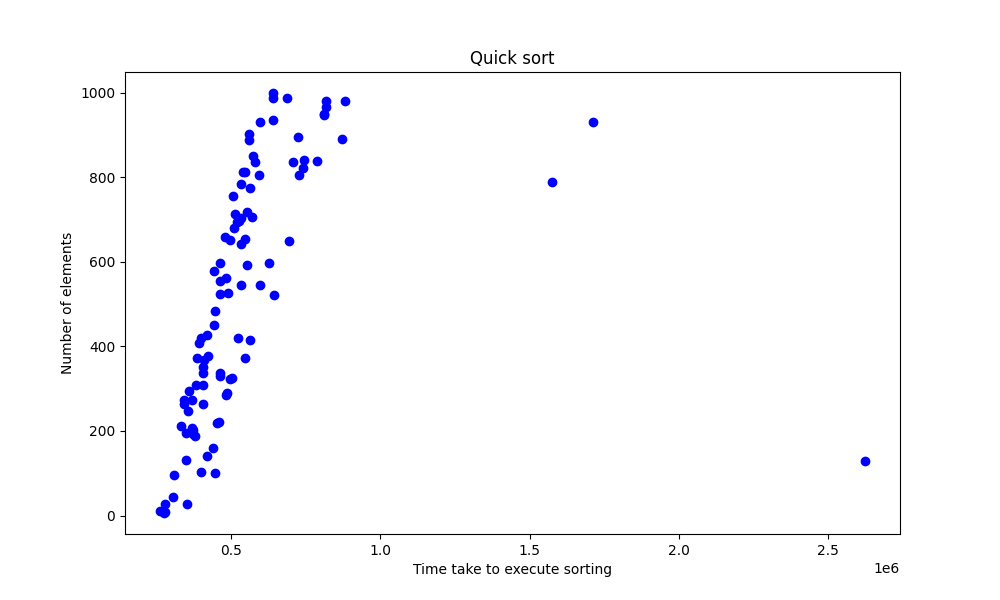
The task is to compare different sorting algorithms, namely quick sort, heap sort, insertion sort, radix sort.

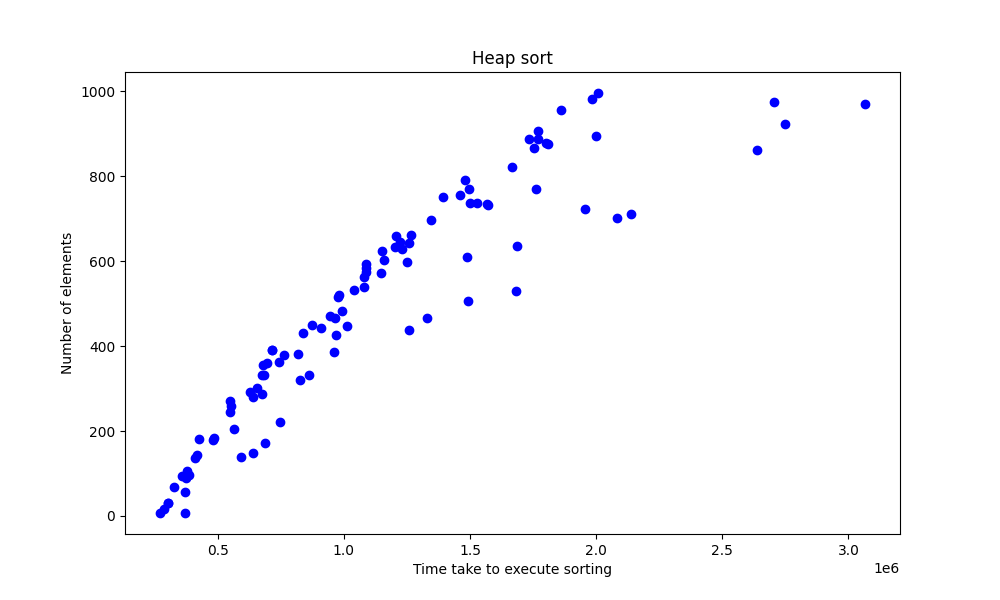
Quick sort

QS is a sorting algorithm, which works recursively by distributing all elements to two logical parts and applying the distribution again. One element is chosen as a pivot, while others are compared with it and divided into two groups, namely with all numbers that are bigger than the pivot or less respectively. It is efficient in most of the cases, however it takes quadratic time in the worst case.



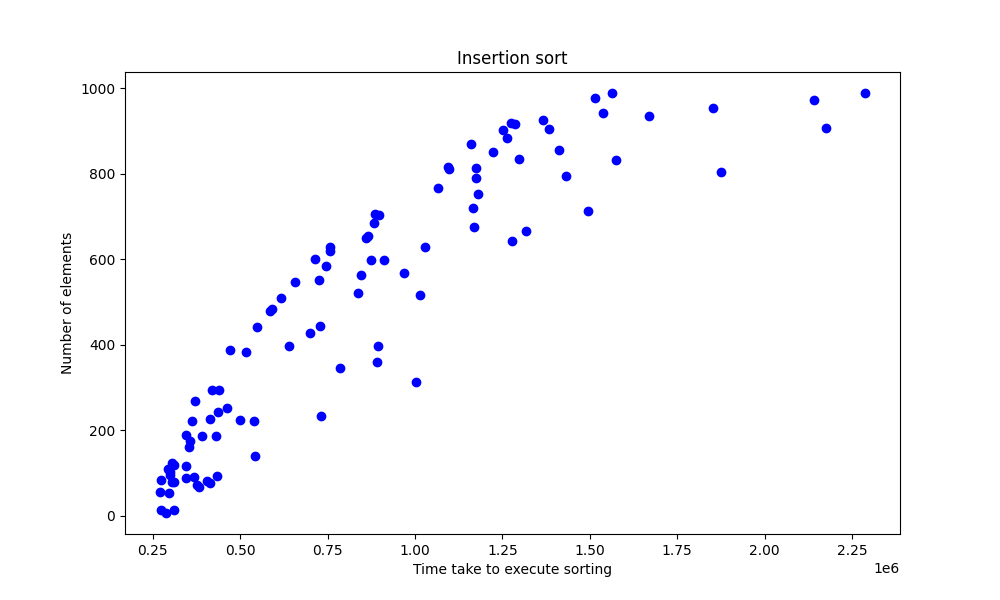
Heap sort

HS is an algorithm that uses a heap structure to sort numbers quickly. The model of a heap helps contributes to fasten the way of searching for the max element. It works similarly to selection sort, but provides more efficient way for finding the next max element. It has consistent time complexity O(nlogn), but complex in implementation.



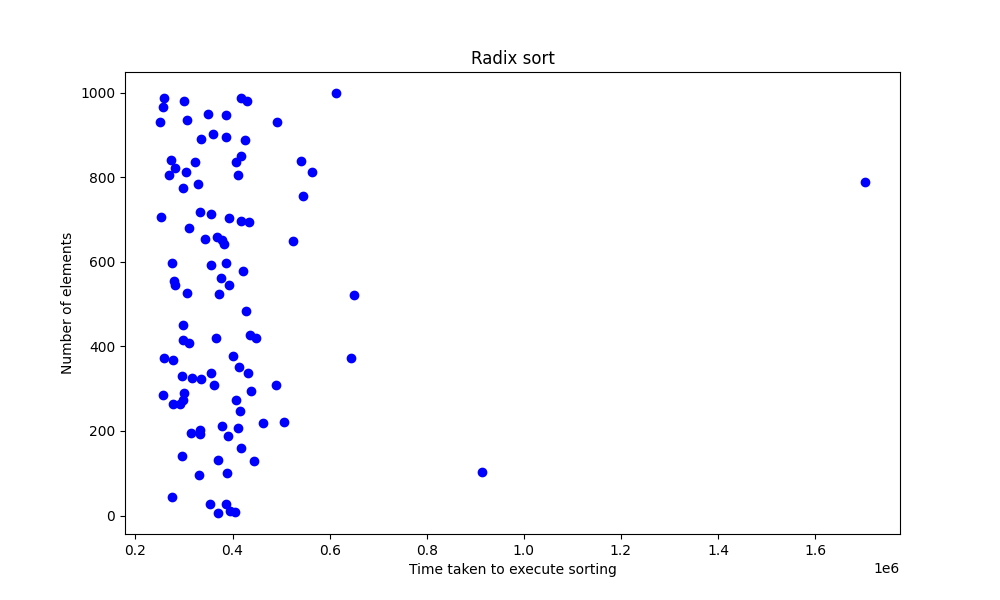
Insertion sort

IS repeatedly takes an unsorted element and inserts it into its correct place in the sorted portion of the array. This is done by comparing the unsorted element to the elements in the sorted portion and shifting those elements to the right until the correct place for the unsorted element is found. It is simple in implementation, but has quadratic computational performance for the worst and average cases.



Radix sort

RS is a non-comparison sorting algorithms that works with digits of elements. It starts from the least significant and than moves to the biggest one. It requires additional memory usage (for temporary containers), but it is good because of the time complexity and therefore for dealing with huge numbers. However, it is not suitable for non-numeric data and cumbersome in modifying to work with negative numbers.



|  |  |  |  |
| --- | --- | --- | --- |
| Sorting Algorithm | Best Case | Average Case | Worst Case |
| Quick Sort | O(n log n) | O(n log n) | O(n^2) |
| Heap Sort | O(n log n) | O(n log n) | O(n log n) |
| Insertion Sort | O(n) | O(n^2) | O(n^2) |
| Radix Sort | O(nk) | O(nk) | O(nk) |

Where k is a number of digits.

The C++ language was used to implement these algorithms. Sizes of vectors were picked up randomly in the range (0, 1000), and the constrains for numbers values were mainly in (-1000, 1000) diapason. Graphs were generated from that data.

To summarize, the best sorting algorithms are quick and radix sort. The first one is understandable in implementation and has really good performance. And the second one is the best in time complexity and wins in computing large numbers. However, it cannot be used for negative numbers or any other not numerical data that can be compared.

So, quick sort is the most clear, robust and great in sorting. But the radix sort is the algorithm that should be used for big numerical datasets.