**Sorting algorithms: comparison between insertion,**

**heap,quick and radix sort**

Bakhodir Soliev

November 4, 2023

**1 Introduction**

In this paper, i will conduct a comparative analysis of four sorting algorithms: Insertion,Heap,Quick and Radix Sort.The objective is to determine the most efficient sorting algorithm for different input sizes.The problem is to arrange an array of n integers,where 'n' represents the number of elements in the array.

* 1. **Insertion Sort**

Insert sort inserts elements repeatedly into an ordered order. Inserting an element into an ordered sequence is done by moving all elements,which are one unit higher than the value entered in the index than the currently occupied one, starting with the largest.Initially, the array's first element is considered as the already sorted portion. Subsequently, the algorithm proceeds with 'n - 1' insertions, beginning from the second element and continuing until the last element. On average, the time complexity of Insertion Sort is O(n^2),but it performs better than other O(n^2) sorting algorithms in practice for small data.During these (n - 1) insertions, the algorithm calculates the number of movements and comparisons involved and so we are finding worst-case scenarios for evaluating the efficiency of Insertion Sort,with the worst-case time complexity being O(n^2).However, it has a best case = O(n),only when all elements are sorted.The algorithm is stable and in-place.

* 1. **Quick Sort**

The quick sort algorithm is a sorting method, which it involves partitioning an array into smaller sub-arrays by choosing a pivotal element.First, we choose any element from the array and call it the pivot. Then, we place all the elements smaller than the pivot on the left side of the pivot, and all the elements greater than the pivot on the right side of the pivot.The quick sort method is then called recursively on the left and right sides of the array until the array is completely sorted.In the average case, the time complexity of quicksort is O(n\*(log(n)),making it one of the most efficient sorting algorithms.In the average case,the time complexity of quicksort is O(n\*log(n)),making it one of the most efficient sorting algorithms.In the best case, the time complexity is also O(n\*log(n)).This occurs when the pivot element is always chosen as the median element in the array,resulting in the array being divided into two subarrays of equal size.In the worst case, however,quicksort can have a time complexity of O(n^2).This happens when the pivot element is either the smallest or largest element in the array causing one subarray to have all the elements and the other subarray to be empty.In this case, each recursive call only reduces the size of the array by one element ,resulting in n recursive calls and a time complexity of O(n^2).As a result, due to the need for additional space for recursive calls and partitioning,quicksort cannot be classified as an in-place algorithm.

* 1. **Heap Sort**

Heap sort is a comparison-based sorting algorithm that utilizes the concept of a binary heap data structure.It has a time complexity of O(n\*log(n)) in all cases,making it an efficient sorting algorithm.The heap sort algorithm works by first building a max heap or a min heap from the given array.In a max heap,the parent node is always greater than or equal to its child nodes,while in a min heap,the parent node is always smaller than or equal to its child nodes.Once the heap is constructed,the largest (in the case of a max heap) or smallest (in the case of min heap) element is moved to the end of the array and removed from the heap.This process is repeated until all elements are removed from the heap,resulting in a sorted array.Heap sort is an in-place sorting algorithm,meaning it doesn't require additional memory beyond the input array.However,it is not a stable algorithm,as it may change the relative order of elements with equal keys.

* 1. **Radix Sort**

Radix sort is a non-comparative sorting algorithm that works by sorting elements by their individual digits or charachters.The radix sort algorithm sorts elements by first grouping them by the least significant digit,then by the second least significant digit,and so on untill all digits have been considered.This process results in a sorted array.Radix sort can be performed in two ways: least significant digit radix sort and most significant digit radix sort.LSD radix sort begins with the least significant digit,while MSD radix sort begins with the most significant digit.Radix sort is a stable sorting algorithm,meaning it preserves the relative order of elements with equal keys and radix sort is not in-place algorithm.Best case complexity - its where the array is already sorted.The best-case time complexity of Radix Sort is O(n+k).In the average case - it happens when the array elements are shuffled that is not properly ascending and not properly descending and the average time is O(n\*k). The worst case complexity could be when the array elements are required to be sorted in reverse order.The worst case is as in average case is O(n\*k).

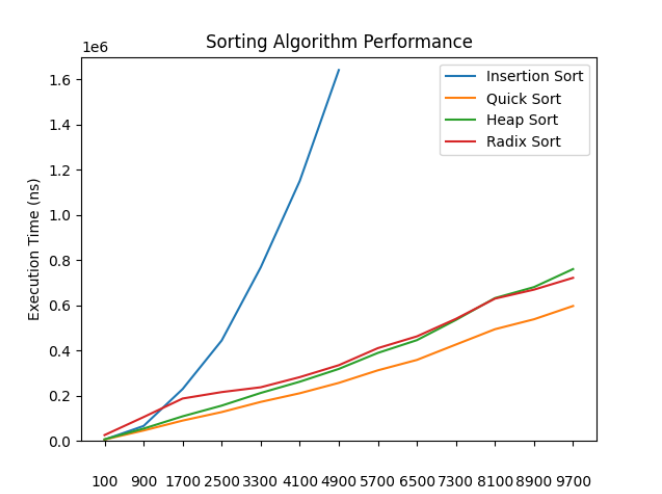
**2 Methodology**

The algorithms were implemented in Java.The results were generated for only one categoriy of randomly shuffled arrays of values.The algorithms were tested on arrays of sizes from 10,100,500,1000,...,10000.So in the Result chapter, will be provided the graphs according to all comparisons.Each array size was tested 1000 times.In each algorithm was given the same input.

**3 Result**

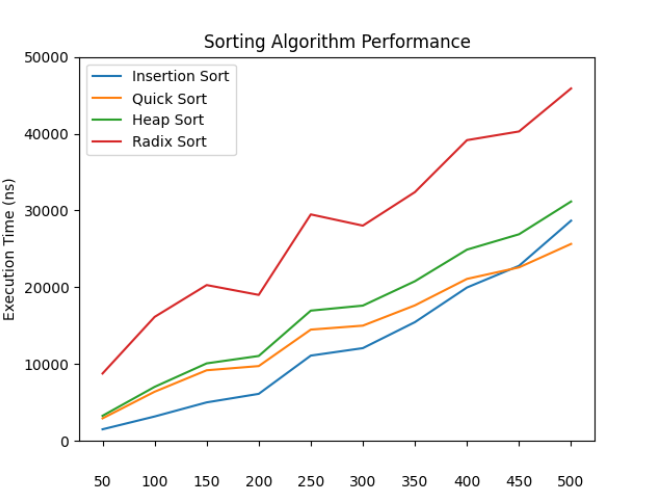
**Comparison of four sorting algorithms from 100 to 9700 with 800 step**

The graph below displays the average sorting time for unsorted arrays.

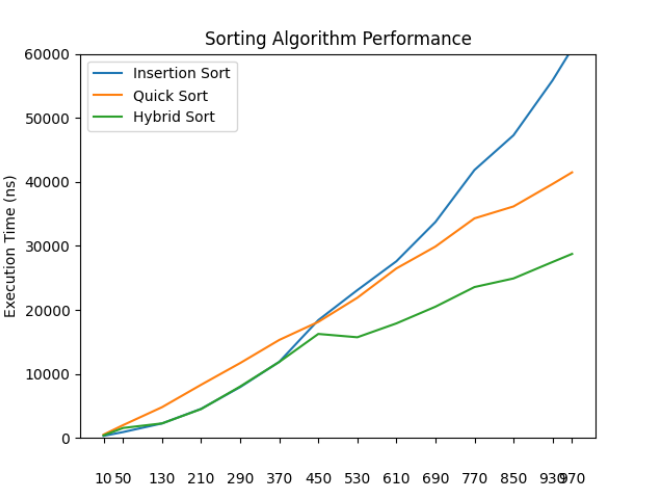


The graph clearly shows that quick sort is time-efficient comparing with four sorting algorithms,and insertion sort is most time consuming algorithm when working with big datasets,so it was checked for the values up to 4900 because checking the greater values is taking a lot of time for measuring .Radix and Heap sorting algorithms displayed almost similar and we can say that with big data sets they are time-efficient respectively.

**Small array sortings from 50 to 500 with 50 step**

****

As we can see for the small array sizes insertion sort displayed the time-efficiency among other sorting algorithms till 450,The intersection between quick sort and insertion sort in 450 size and after 450 quick sort works faster than insertion sort.So the hybrid sort was added:



In conclusion,quick sort is the fastest one for the arrays with big datasets and insertion sort is time-efficient for the arrays with less than 450 elements,so thats why hybrid sort was constructed of quick sort and insertion sort in order to create an optimized sorting algorithm.