Comparison between Quick Sort, Insertion Sort, Heap Sort and Radix Sort.

Adam Heybatov

04.12.2023

1. Introduction

Sorting data efficiently is crucial in computer science. This project centers on comparing several popular sorting methods: Quick Sort, Insertion Sort, Heap Sort, and Radix Sort. We will illustrate strength of each sorting algorithm with our graphs. At the end from our research we will make our own Hybrid Sort algorithm.

* 1. Quick Sort

Quick Sort is a popular sorting algorithm known for its efficiency and simplicity. It works by selecting a 'pivot' element and partitioning the array around it, sorting smaller elements to one side and larger elements to the other. This process is recursively applied to the partitions until the entire array is sorted. In the best and average cases, Quick Sort typically exhibits a time complexity of O(n log n).However in the worst case where the pivot selection consistently divides the array in an unbalanced manner, the time complexity can degrade to O(n^2).

* 1. Insertion Sort

Insertion Sort is a simple sorting algorithm ideal for small datasets or nearly sorted arrays. It works by iteratively taking each element and placing it in its correct position among the already sorted elements, shifting greater elements to make space for insertion. The time complexity of Insertion Sort is O(n^2) in the worst-case scenario, where the array is in reverse order or nearly sorted. However, in the best-case scenario—when the array is already sorted—the time complexity reduces to O(n) as it requires minimal comparisons and shifts.

* 1. Radix Sort

Radix Sort is a non-comparative sorting algorithm specialized in sorting numbers, particularly integers, by processing digits individually. It sorts elements by grouping them based on significant digits, from the least significant to the most significant, using counting sort or similar methods. Its time complexity is O(nk), where n is the number of elements and k is the number of digits in the largest number in the array. Worst-Case Time Complexity. In radix sort, the worst case is when all elements have the same number of digits except one, which has a significantly large number of digits. If the number of digits in the largest element equals n, the runtime is O. (n2).

* 1. Heap Sort

Heap Sort is an efficient and in-place sorting algorithm that uses a binary heap data structure. It transforms the array into a max heap, repeatedly extracts the maximum element from the heap, and reconstructs the heap until all elements are sorted. Its time complexity is consistent with O(n log n) in both the best and worst cases, making it highly efficient for large datasets.

Graph 1.1

As you can see in the graph 1.1 insertion sort takes too much time for big data. Therefor I will remove insertion sort from the graph to see what is happening with other sorts and I will show it in graph 1.2.

Graph 1.2

Graph 1.3

In Graph 1.2 we remove Insertion Sort and Heap Sort. As we can see Radix sort has hegemony in big data. However, Sort and Radix Sort are approximately the same till some size. Therefore, We decreased the size from 10000 to 3000 and we check again from 100 to 3000 in Graph 1.3. In this graph shows us that quick sort algorithm is the best option for the size from 100 to 1400. Now we will find what happens size from 1 to 50.

Graph 1.4

As we can see in small data Insertion Sort and Quick Sort seems efficient however Heap Sort and Radix Sort takes more time than Insertion Sort and Quick Sort. In Graph 1.5 I will remove those algorithms.

Graph 1.5

Now we see in Graph 1.5 Insertion Sort is the best option for size from 1 – 30.

* 1. Hybrid Sort (conclusion)

From result of the research we will use Insertion Sort from size 1 – 30, Quick Sort from 30 – 1400 and from 1400 – 10000 Radix Sort.