Programming Assignment 4

Sorting

By
Kishlay Kumar and K Pramod Krishna
E-A 008: Data Structures and Algorithms
Prof. Aakash Tyagi
TA : Raghav Awasty

Introduction

We will implement three different versions of a sorting and study their performance. Namely:

(1) Radix Sort

(2) Merge Sort

(3) Quick Sort

Theoretical Analysis

Sorting: A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of elements in the respective data structure. Divide and Conquer sorting algorithms like quicksort and mergesort breaks a problem into subproblems that are similar to the original problem, recursively solves the subproblems, and finally combines the solutions to the subproblems to solve the original problem.

<u>Radix Sort</u>

Radix Sort takes $O(d*(n+b))$ time where b is the base for representing numbers, for example, for the decimal system, b is 10. What is the value of d? If k is the maximum possible value, then d would be $O(logb(k))$. So overall time complexity is $O((n+b) * logb(k))$. Which looks more than the time complexity of comparison-based sorting algorithms for a large k. Let us first limit k. Let $k <= nc$ where c is a constant. In that case, the complexity becomes $O(nLog(n))$. But it still doesn’t beat comparison-based sorting algorithms.

<u>Merge Sort</u>

The Merge Sort algorithm is a sorting algorithm that is based on the Divide and Conquer paradigm. In this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner. The time complexity of Merge Sort is $O(Nlog(N))$ in all 3 cases (worst, average, and best) as merge sort always divides the array into two halves and takes linear time to merge two halves.

Quick Sort

Like Merge Sort, Quicksort is a Divide and Conquer algorithm. It picks an element as a pivot and partitions the given array around the picked pivot. There are many different versions of quicksort that pick pivot in different ways. The best case occurs when the partition process always picks the middle element as the pivot. Although the worst-case time complexity of Quicksort is $O(N^2)$ which is more than many other sorting algorithms like Merge Sort and Heap Sort, Quicksort is faster in practice

Experimental Setup

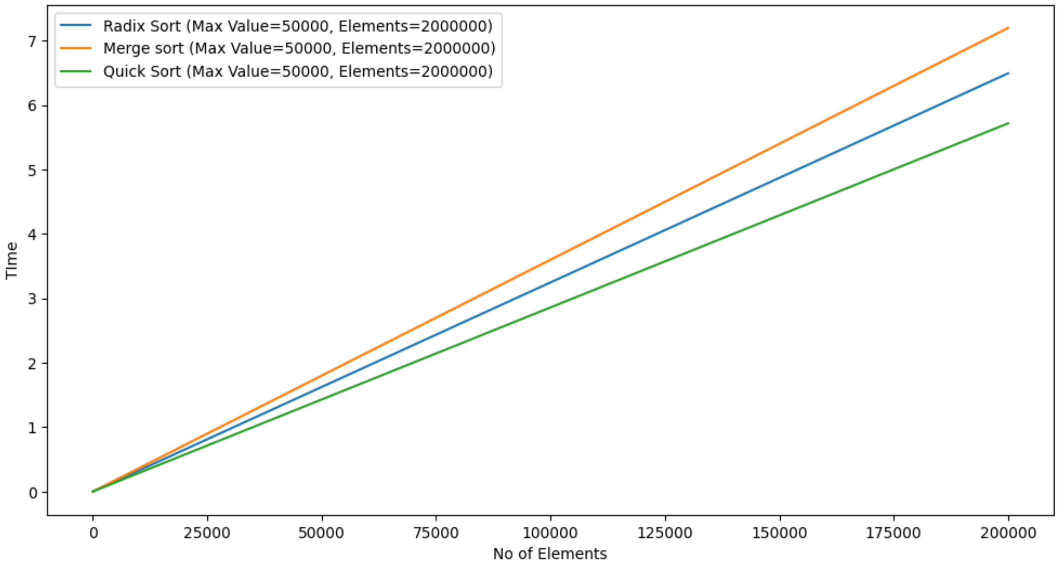
We used a machine running windows 11 OS with the following specs:

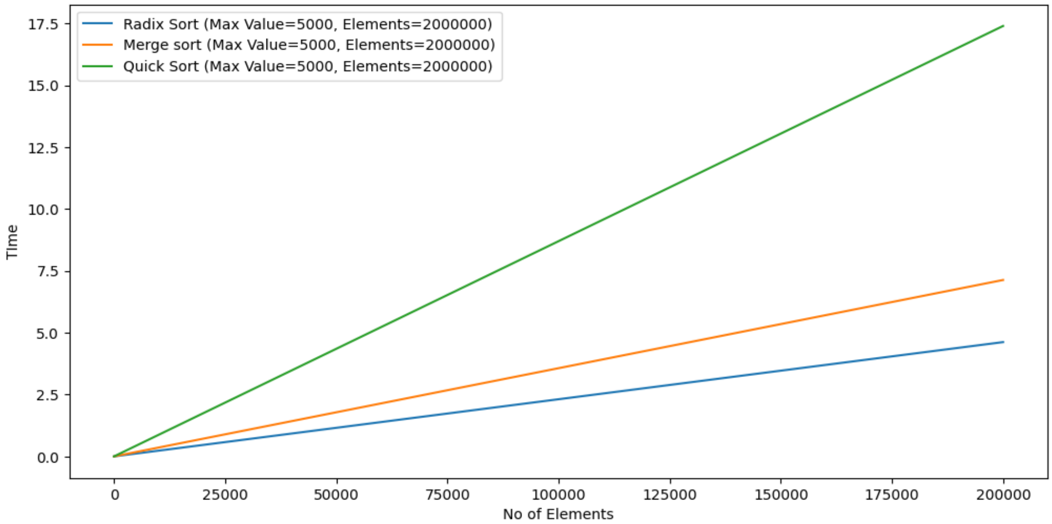
System Model	Pulse GL66 11UEK
System Type	x64-based PC
System SKU	1581.3
Processor	11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, 2301 Mhz, 8 Core(s), 16 Logical Processor(s)
BIOS Version/Date	American Megatrends International, LLC. E1581IMS.30F, 07-12-2021
SMBIOS Version	3.3
Embedded Controller Version	255.255
BIOS Mode	UEFI
BaseBoard Manufacturer	Micro-Star International Co., Ltd.
BaseBoard Product	MS-1581
BaseBoard Version	REV:1.0
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\WINDOWS
System Directory	C:\WINDOWS\system32
Boot Device	\Device\HarddiskVolume1
Locale	United States
Hardware Abstraction Layer	Version = "10.0.22621.819"
User Name	MSI\kishl
Time Zone	India Standard Time
Installed Physical Memory (RA...	16.0 GB
Total Physical Memory	15.7 GB
Available Physical Memory	4.68 GB
Total Virtual Memory	30.0 GB
Available Virtual Memory	13.1 GB

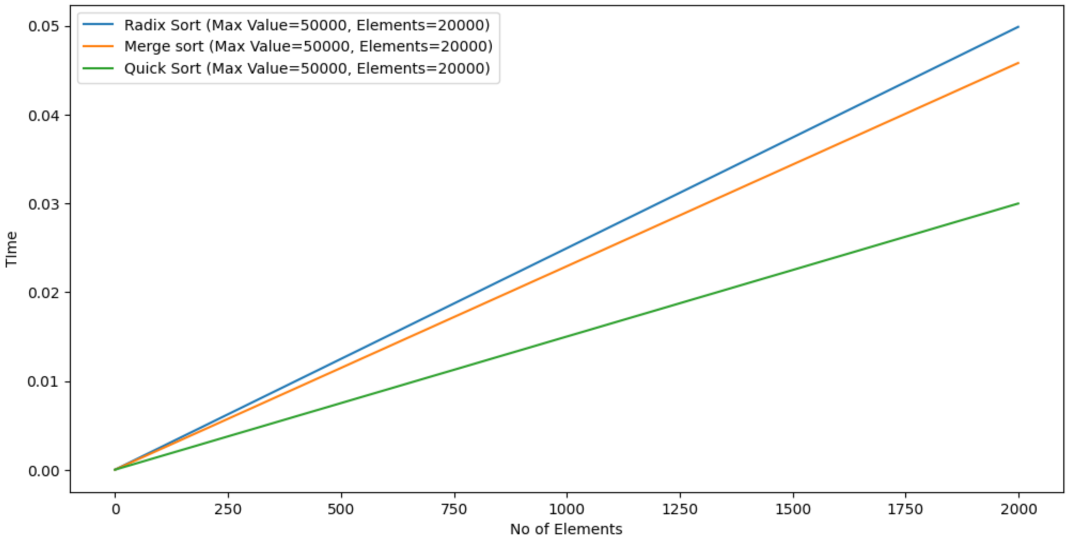
Test Inputs & Process:

We tested the sorting times for various input sizes (200000, 2000000) as well as different length of digits. Recorded the time before and after sorting and linearly separated it and then visualized it using matplotlib. Following are the results of the experiment.

Experimental Results







Observations:

- Radix sort performs better when thew number of digits is less even if the number of elements is more.

- Quicksort performs best when there are more elements, and the number of digits is also large.

- Mergesort performs average is all cases.