National Institute of Technology
Karnataka, Surathkal

Assignment Report: Sorting Analysis

CS700

Algorithms And Complexity

Submitted To:
Dr. Vani M
Associate Professor
Computer Science
Department

Submitted By:
Dileep Reddy Gosu
242CS017
M Tech
1st Semester

Contents

1 Introduction

Comparison of Sorting Algorithms

Comparing sorting algorithms involves evaluating their performance in terms of time complexity, space complexity, and practical efficiency. Here, we compare six popular sorting algorithms: **Insertion Sort**, **Bubble Sort**, **Merge Sort**, **Heap Sort**, **Radix Sort**, and **Quick Sort**.

Insertion Sort

- **Time Complexity**:
  - Best: $O(n)$
  - Average: $O(n^2)$
  - Worst: $O(n^2)$

Bubble Sort

- **Time Complexity**:
  - Best: $O(n)$
  - Average: $O(n^2)$
  - Worst: $O(n^2)$

Merge Sort

- **Time Complexity**:
  - Best: $O(n \log n)$
  - Average: $O(n \log n)$
  - Worst: $O(n \log n)$

Heap Sort

- **Time Complexity**:
  - Best: $O(n \log n)$
  - Average: $O(n \log n)$
  - Worst: $O(n \log n)$

Radix Sort

- **Time Complexity**:
  - Best: $O(nk)$
  - Average: $O(nk)$
  - Worst: $O(nk)$

Quick Sort

- **Time Complexity**:
  - Best: $O(n \log n)$
  - Average: $O(n \log n)$
  - Worst: $O(n^2)$

2 Data Generation and Experimental Setup

1. **What kind of machine did you use?**

Component	Details
Processor	Intel Core i5-9300H
Graphics	NVIDIA GeForce GTX 1650
Memory	8 GB DDR4 2666 MHz
Storage	512 GB PCIe NVMe SSD
Operating System	Windows 10 Home

Table 1: Acer Nitro 5 (AN515-54) System Specifications

2. **What timing mechanism did you use?**
   c++ Standard Timing Mechanism Before the Function Starts it starts the timer in milli seconds and end the timer when the function return back to the file

3. **How many times did you repeat each experiment?**
   Repeated the experiment so many times untill it is stabled to one saturated point.aproximately 35-40 times

4. **What times are reported?**

Best Case Performance (in ms)

	10000	25000	50000	75000	100000
Bubble sort	0.208718	1.37705	6.10409	11.9284	21.0614
Insertion Sort	3e-05	0.000167	0.000512	0.000464	0.000637
Merge Sort	0.011543	0.012729	0.040524	0.038125	0.051745
Quick sort	0.000584	0.001652	0.004649	0.004851	0.007082
heap Sort	0.002906	0.007884	0.027251	0.02628	0.037392
Radix sort	0.001623	0.004189	0.009033	0.012066	0.020337

Table 2: Best Case Performance for All Sorting Algorithms

Worst Case Performance (in ms)

	10000	25000	50000	75000	100000
Bubble sort	0.628138	3.92506	15.5298	35.1707	63.9983
Insertion sort	0.287589	1.83405	7.36959	16.8773	29.0017
Merge sort	0.004695	0.012246	0.025961	0.039539	0.050406
Quick sort	0.000643	0.001797	0.003806	0.005412	0.008124
Heap sort	0.002686	0.007428	0.016556	0.025633	0.033584
Radix sort	0.001668	0.004107	0.008229	0.012365	0.015738

Table 3: Worst Case Performance for All Sorting Algorithms

Average Case Performance (in ms)

	10000	25000	50000	75000	100000
Bubble sort	0.562817	3.51982	14.1775	33.4867	57.9221
Insertion sort	0.148834	0.885714	3.58435	8.17295	14.6388
Merge sort	0.005827	0.014728	0.030547	0.047165	0.06333
Quick sort	0.001694	0.00443	0.009743	0.015645	0.020526
Heap sort	0.003181	0.008644	0.018904	0.032311	0.040858
Radix sort	0.00164	0.003982	0.008112	0.013451	0.016183

Table 4: Average Case Performance for All Sorting Algorithms

5. **How did you select inputs?**
   Randomly with uniform distribution of positive integers in the range of 1 and 1000000

6. **Did you use the same inputs for all sorting algorithms?**
   yes,I used the same iinputs for all the algorithms

3 Which of the Three Versions of Quicksort Seems to Perform the Best?

1. **Graph the best case running time as a function of input size n for all three versions.**

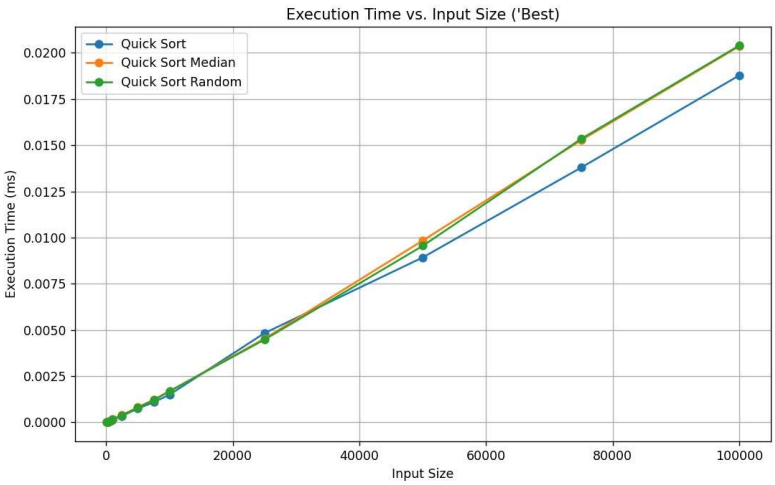


Figure 1: Best Case for the Quick Sorting Algorithms.

This graph determines the best case of the quick sort algorithm which chooses the pivot element as low represent in blue colour and median of first,last and middle and random element in the array. the best case will come when we pass a random file as a input which will be equivalent to $O(nlogn)$ in all cases.

2. **Graph the Average case running time as a function of input size n for all three versions.**

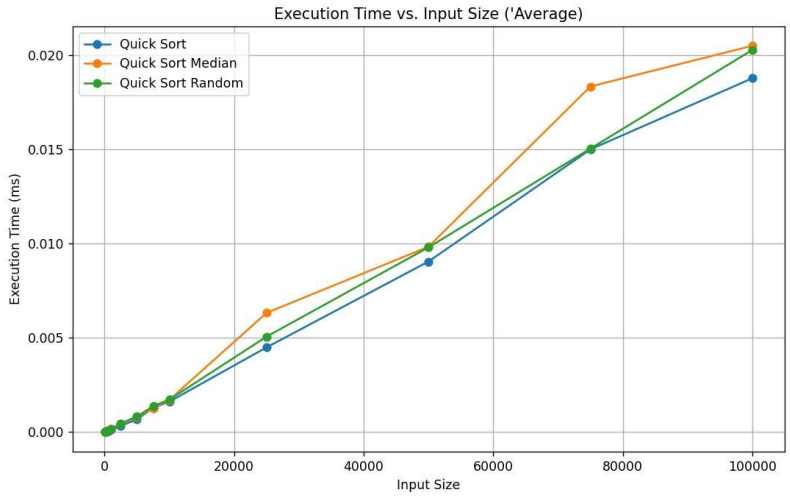
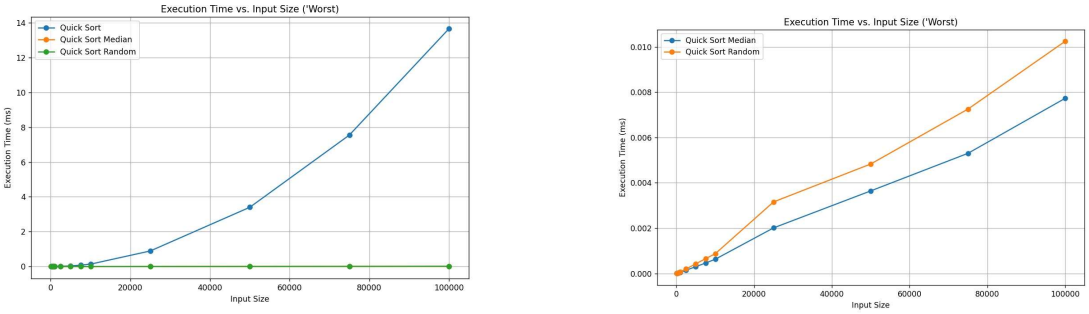


Figure 2: Average Case for all the Sorting Algorithms.

This graph determines the Average case of the quick sort algorithm which chooses the pivot element as low represent in blue colour and median of first,last and middle and random element in the array. the best case will come when we pass a random file as a input which will be equivalent to $O(nlogn)$ in all cases.

3. **Graph the Worst case running time as a function of input size n for all three versions.**



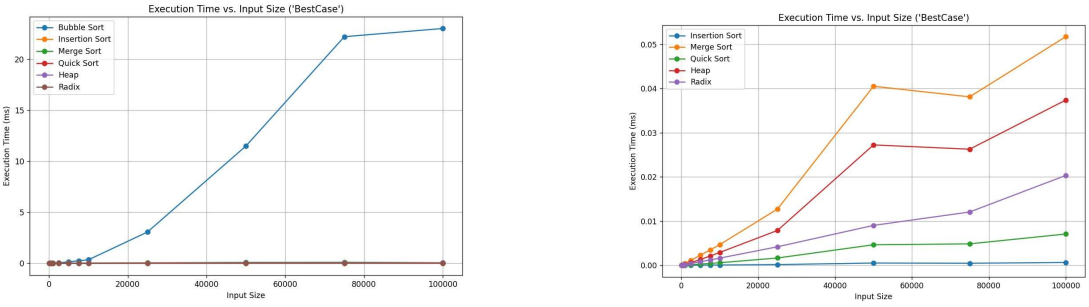
(a) Worst Case (b) Worst case without 1st case

Figure 3: Worst Case for all Quick Sort Algorithms.

This graph determines the worst case of the quick sort algorithm which chooses the pivot element as low represent in blue colour and median of first,last and middle and random element in the array. the worst case will come when we pass a sorted or nearly sorted array as a input which will be equivalent to $O(n^2)$ in the pivot as a low but the median and random element will reduce to O(NlogN).

4 Which of the Six Sorts Seems to Perform the Best (Consider the Best Version of Quicksort)?

1. **Graph the best case running time as a function of input size n for the six sorts.**

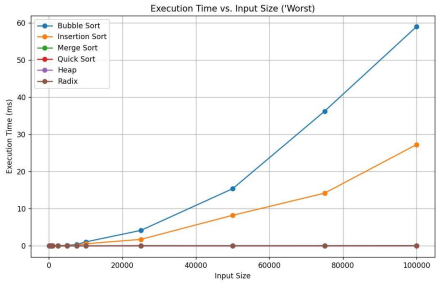


(a) Best Case (b) without Bubble Sort

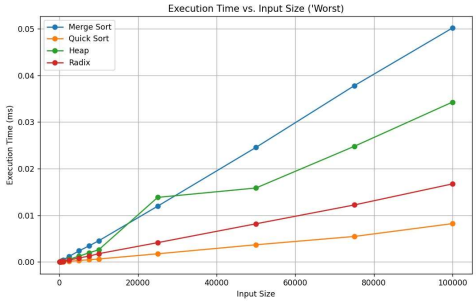
Figure 4: Best Case for all the Sorting Algorithms.

This graph determines the Best Case of the All sorting algorithms The insertion sort performs best in O(N). thereby Merge sort,Quick sort, Heap sort,Radix sort follows the same

2. **Graph the worst case running time as a function of input size n for the six sorts.**



(a) Worst Case

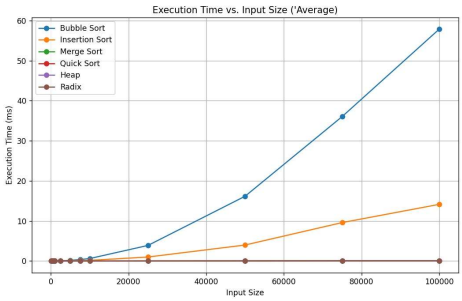


(b) Without Bubble and Insertion Sort

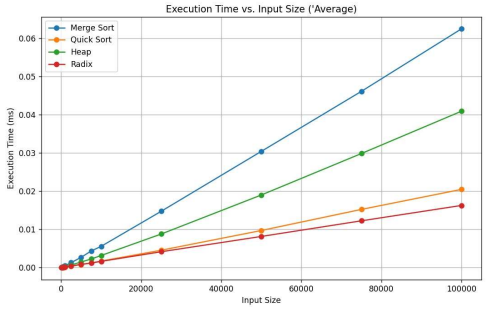
Figure 5: Worst Case for all the Sorting Algorithms.

This graph determines the Worst Case of the All sorting algorithms The Quick sort performs best in O(NlogN). thereby Merge sort, Heap sort,Radix sort follows after Quick sort.

3. **Graph the average case running time as a function of input size n for the six sorts.**



(a) Average Case



(b) without Bubbe and Insertion sort

Figure 6: Average Case for all the Sorting Algorithms.

This graph determines the Average Case of the All sorting algorithms The Radix sort performs best in O($N * k$) here it is depends on the length of the number. thereby Merge sort, Heap sort,Quick sort follows after Radix with least being the Bubble and Insertion sort.

5 Analyze your data to see if the number of comparisons is correlated with execution time plot (time #comparisons) vs. n and refer to these plots in your answer

- **Quicksort:** Showing correlation value nearly 0.99,0.99,0.99 in best and worst case but in average case it is showing 0.99,0.99,0.99 for all 3 types of quicksort variations and for all Algorithms combining it is the fastest algorithm
- **Merge and Heap sort:** Merge sort and heap sort are showing same trend in all the cases

- **Insertion Sort & Bubble sort:** A stronger correlation with n but with higher execution times compared to Merge and Heap Sort
- **Radix Sort:** Shown a linear trend bcox all the number are nearly of same digits

Input Size	Bubble Sort	Insertion Sort	Merge Sort	Quick Sort	Heap Sort
100	4950	99	356	480	1081
250	31125	249	1011	1503	3363
500	124750	499	2272	3498	7756
750	280875	749	3769	5737	12657
1000	499500	999	5044	7987	17583
2500	3.12375e+06	2499	14752	23417	51040
5000	1.24975e+07	4999	32004	51822	112173
7500	2.81212e+07	7499	49432	81831	176524
10000	4.9995e+07	9999	69008	113632	244474
25000	3.12488e+08	24999	188476	317257	677702
50000	1.24998e+09	49999	401952	684515	1.45561e+06
75000	2.81246e+09	74999	624316	1.06944e+06	2.26835e+06
100000	4.99995e+09	99999	853904	1.46937e+06	3.1127e+06

Table 5: Best Case Comparisons

These are comparisons for the Worst case of all sorting algorithms. and these are correlated with time.

Input Size	Bubble Sort	Insertion Sort	Merge Sort	Quick Sort	Heap Sort
100	4950	4950	316	480	944
250	31125	31125	983	1503	3007
500	124750	124750	2216	3498	7010
750	280875	280875	3457	5737	11438
1000	499500	499500	4932	7987	15965
2500	3123750	3123750	13656	23417	46694
5000	12497500	12497500	29811	51823	103223
7500	28121250	28121250	47388	81828	163721
10000	49995000	49995000	64640	113632	226684
25000	312487500	312487500	178995	317263	633684
50000	1249975000	1249975000	383324	684953	1365990
75000	2812462500	2812462500	596063	1069440	2138700
100000	4999950000	4999950000	818227	1519550	2926670

Table 6: Worst Case Comparisons

These are comparisons for the Worst case of all sorting algorithms. and these are correlated with time.

Input Size	Bubble Sort	Insertion Sort	Merge Sort	Quick Sort	Heap Sort
100	4950	2479	545	530	1036
250	31125	15398	1677	1813	3197
500	124750	60839	3869	3940	7410
750	280875	138198	6211	6720	12074
1000	499500	249815	8712	9499	16830
2500	3123750	1536270	25152	26701	48900
5000	12497500	6405760	55310	58440	107615
7500	28121250	14118300	87123	95171	170130
10000	49995000	25325100	120401	131235	235380
25000	312487500	155534000	334256	368362	654907
50000	1249975000	623506000	718386	794975	1409880
75000	2812462500	1408100000	1121440	1233540	2200550
100000	4999950000	2513850000	1536310	1707820	3019080

Table 7: Random Comparisons

These are comparisons for the Average case of all sorting algorithms. and these are correlated with time.

Code: Acces to Github repo