Sorting Algorithms Analysis

Yash Narnaware

Entry Number: 2024AIM1011

Course Code: CS506

Department of Computer Science and Engineering

System Specifications

Processor: AMD Ryzen 7 3750H

RAM: 16 GB

OS: Windows 11

Chapter 1

Sorting Algorithms

1.1 Bubble Sort

The time complexity of Bubble sort is $O(n)^2$ because we have 2 nested for loops (can be while loops also) in its algorithm. Best case is when array is already sorted in that case time complexity becomes $O(n)$ which wee achieve by keeping track of if swapping is happend in that iteration. And worst case is when array is revese sorted.

1.2 Selection Sort

The time complexity of selection sort in best worst and average case is $O(n)^2$ because it has 2 nested loops one for choosing minimum and 1 for traversing through array.

1.3 Merge Sort

The time complexity for merge sort is $O(n \log n)$ in best worst and average cases because in any case we have to divide array in half sizes which takes $O(\log n)$ time and merging the sorted arrays take $O(n)$ time. So we are dividing array in half till they become sorted(singleton array) and then combine the sorted arrays which altogether takes $O(n \log n)$ time.

1.4 Insertion Sort

The time complexity of insertion sort is $O(n^2)$ in worst and average case and is $O(n)$ in best case. The best cae is when array is already sorted in this case we don't need to find the correct place for current element in preceeding array. The worst case occures when array is rverse sorted in this case we have go to the start of preceeding array to place the current element.

1.5 Quick Sort

The time complexity of quick sort in best and average case is $O(n \log n)$ and in worst case is $O(n^2)$.Dividing based on pivot takes $O(n)$ time and passing the divided arrays to quick sort again takes $O(\log n)$ time so altogether time complexity becomes $O(n \log n)$. The best case occurs when we select array median as a pivot for partitioning and worst case is when we select the pivot such that array gets divided into 1 and N-1 elements.

Chapter 2

Comparing Sorting Algorithms

I have taken arrays of different sizes and different conditions like sorted,uunsorted ,reverse sorted etc. and plotted graphs of how different sorting algorithms behave on them.

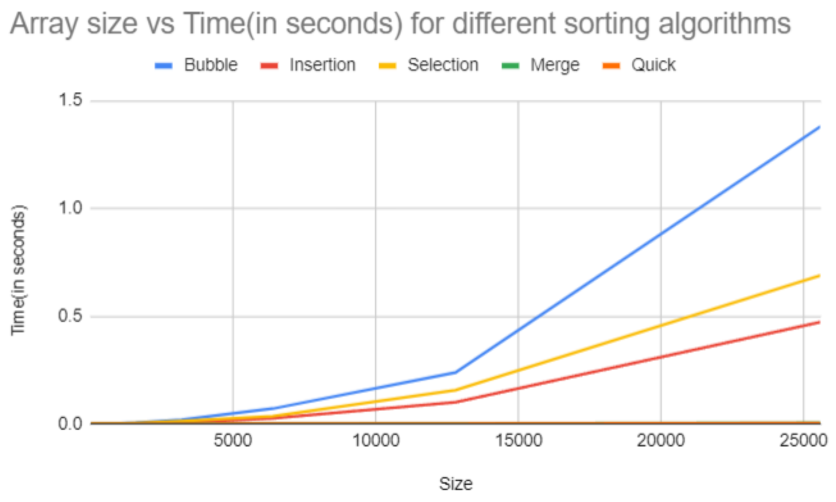
2.0.1 How sorting algorithms behave to diffrent array sizes?

Data in table format -

Size	Bubble Sort	Insertion Sort	Selection Sort	Merge Sort	Quick Sort
10	0	0	0	0	0
100	0.00003	0.00001	0.00002	0.00001	0.00001
200	0.00009	0.00003	0.00006	0.00002	0.00002
400	0.00036	0.00012	0.00022	0.00005	0.00004
800	0.00127	0.00045	0.00085	0.0001	0.00007
1600	0.0048	0.00172	0.00327	0.00021	0.00015
3200	0.01883	0.00687	0.01282	0.00046	0.00034
6400	0.07035	0.02548	0.0349	0.00075	0.00053
12800	0.2383	0.09995	0.15588	0.00161	0.00109
25600	1.38234	0.47288	0.68966	0.00434	0.00309

Table 2.1: Sorting Algorithms on different size arrays

Data in graph format -



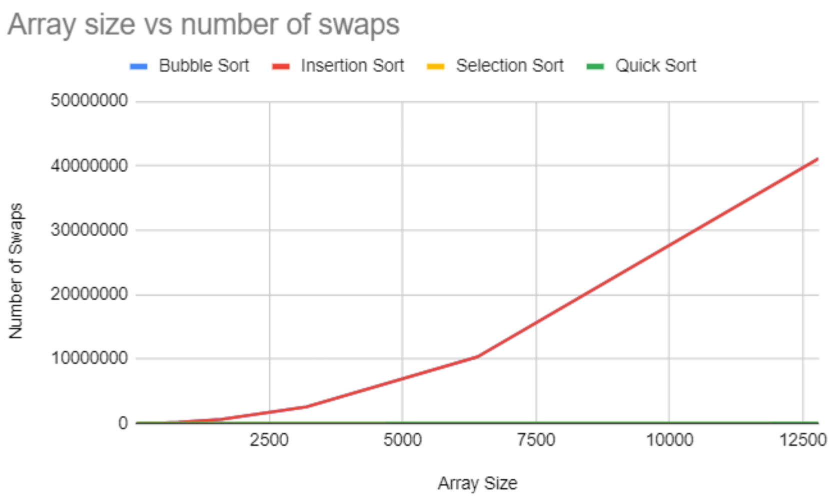
2.0.2 Number of swaps for different array size

Data in table format -

Array Size	Bubble Sort	Insertion Sort	Selection Sort	Quick Sort
10	37	27	7	10
100	2675	2575	93	158
200	10035	9834	195	377
400	41999	41590	393	857
800	164822	163991	793	1839
1600	640571	638857	1597	4017
3200	2558263	2554546	3192	8878
6400	10336066	10327571	6389	19338
12800	41184565	41163493	12789	41337
25600	161983948	161925779	25592	89208

Table 2.2: Number of swaps for different sorting algorithms on different sized arrays

Data in graph format -

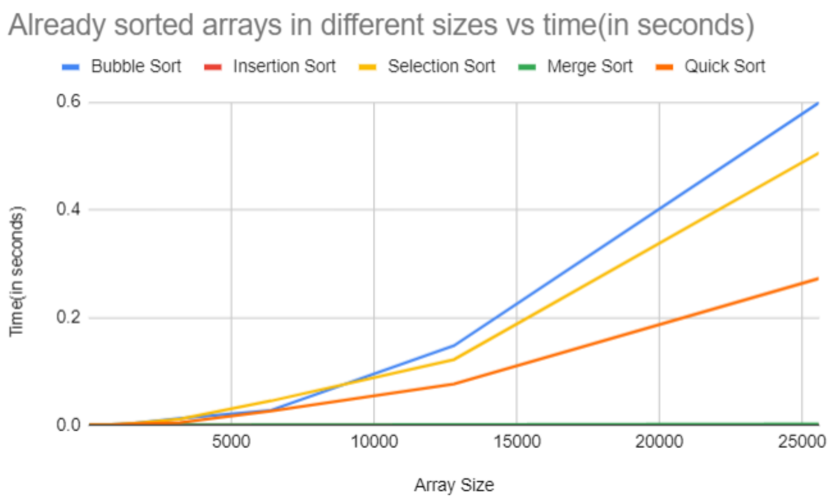


2.0.3 Testing sorting algorithms on already sorted arrays

Array Size	Bubble Sort	Insertion Sort	Selection Sort	Merge Sort	Quick Sort
10	0	0	0	0	0
100	0.00001	0	0.00001	0.00001	0.00002
200	0.00005	0	0.00005	0.00001	0.00004
400	0.0002	0	0.0002	0.00003	0.00015
800	0.00079	0	0.00079	0.00005	0.00059
1600	0.00309	0.00001	0.00309	0.00012	0.00212
3200	0.0121	0.00001	0.01121	0.00019	0.00475
6400	0.02746	0.00002	0.04488	0.00054	0.02599
12800	0.14739	0.00006	0.12168	0.00087	0.07607
25600	0.59978	0.00007	0.50619	0.0018	0.27262

Table 2.3: Time taken by Sorting Algorithms on different sized sorted arrays

Data in graph format -

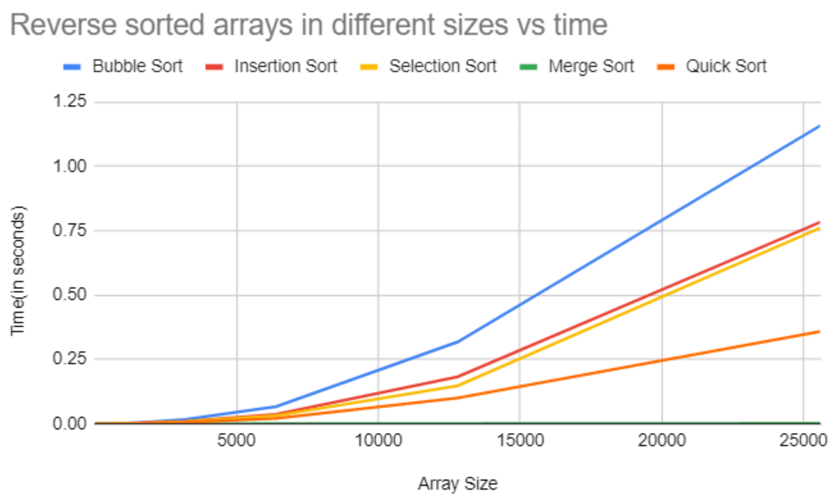


2.0.4 Testing sorting algorithms on arrays in reverse sorted order

Data in graph format -

Array Size	Bubble Sort	Insertion Sort	Selection Sort	Merge Sort	Quick Sort
10	0	0	0	0	0
100	0.00002	0.00001	0.00003	0.00001	0.00001
200	0.00007	0.00004	0.00004	0.00001	0.00003
400	0.00028	0.00015	0.00013	0.00002	0.0001
800	0.0011	0.00057	0.0005	0.00004	0.00034
1600	0.0042	0.00303	0.00199	0.00009	0.00137
3200	0.01695	0.00928	0.00769	0.00019	0.00518
6400	0.06711	0.037	0.03188	0.00041	0.02078
12800	0.31812	0.18226	0.14792	0.00091	0.10011
25600	1.15921	0.78429	0.76152	0.00179	0.35887

Table 2.4: Time taken by Sorting Algorithms on different sized reverse sorted arrays


Reverse sorted arrays in different sizes vs time
Bubble Sort
Insertion Sort
Selection Sort
Merge Sort
Quick Sort
1.25
1.00
0.75
0.50
0.25
0.00
Time(in seconds)
5000
10000
15000
20000
25000
Array Size