Yakupguly Malikov

Cmpe 223

Programming Homework 2

Sorting Algorithms Guessing

I created SortingAlgorithmsTester class in order to test the sorting algorithms. I used four types of arrays that are ascending ordered, descending ordered, randomly generated and array with the same elements. In order not to write same code again and again I created Test class which takes an array and calls a sorting algorithm to test them. In test class, it tests sorting algorithm 5 times and prints their average. In SortingAlgorithmsTester class I created four types of arrays to store ascending, descending, random and same. Using for loop assigned values to them.

Using these 2 classes I get data in the below table. Let’s guess which sorting algorithm they are.

It could be easily seen that sort 4 is always slower than others so it is selection sort because even the array is sorted it again goes through each element one by one. For this reason, time complexity of selection sort is O(n^2). So, we can conclude that **sort 4** is **selection sort**.

In the table, sort 1 and sort 2 is faster than others in ascending, descending and random cases. They could be quick sort and merge sort. In order to decide this, I used the array with same elements. Sorting array with all elements are same is worst case of quick sort. So, sort 1 took more time than sort 2 to sort array with the same elements. That’s why **sort 1** is **quick sort** and **sort 2** is **merge sort**.

There are sorting algorithm left that are sort 3 sort 5. For the best case, time complexity of insertion sort and bubble sort is O(n) because when the array is already sorted, they only iterate once through array. In insertion sort number of swaps reduced than bubble sort. Bubble sort takes more time than insertion sort for the larger inputs. In the table, sort 5 is faster than sort 3. As the result, **sort 5** is **insertion sort** and **sort 3** is **bubble sort**.

Sorting times of each algorithm according to ascending, descending, randomly generated and array with the same elements.

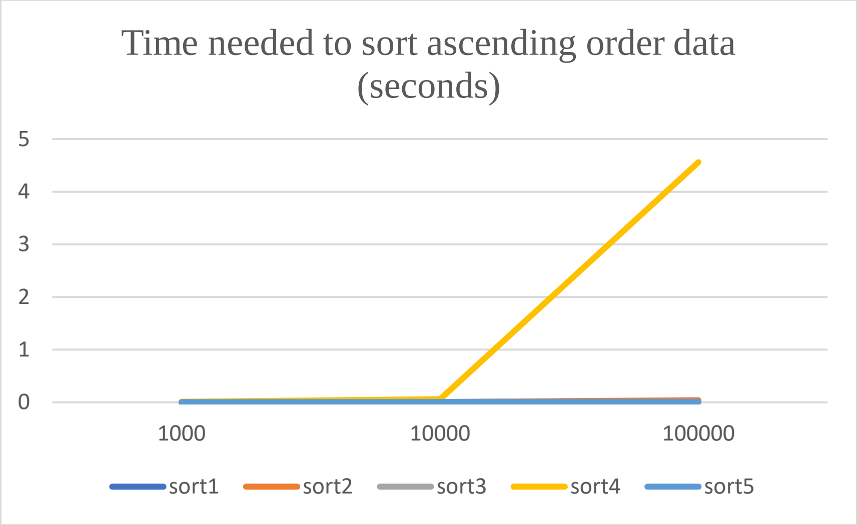
	Ascending		
Sorting Algorithms (seconds)	Size		
	1000	10000	100000
Sort 1	0.002	0.004	0.03
Sort 2	0.002	0.006	0.04
Sort 3	0.002	0.003	0.007
Sort 4	0.001	0.057	4.56
Sort 5	0.002	0.003	0.006

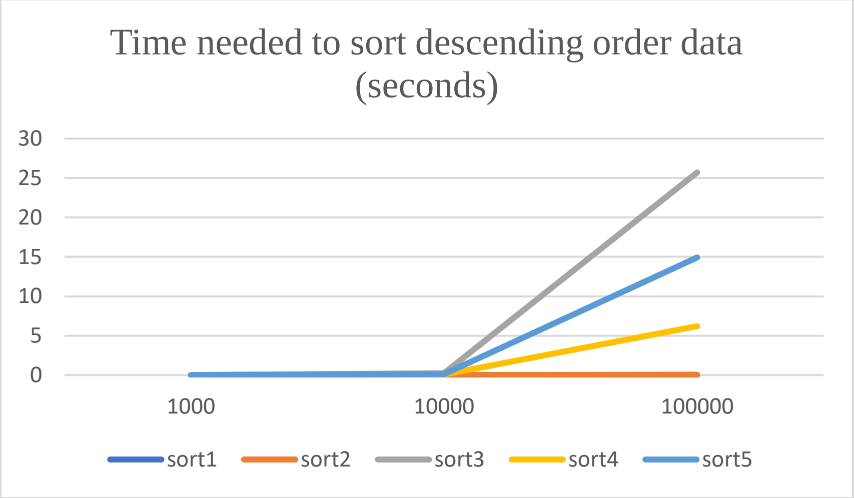
	Descending		
Sorting Algorithms (seconds)	Size		
	1000	10000	100000
Sort 1	0.002	0.004	0.03
Sort 2	0.002	0.006	0.04
Sort 3	0.008	0.25	25.7
Sort 4	0.01	0.07	6.21
Sort 5	0.01	0.14	14.92

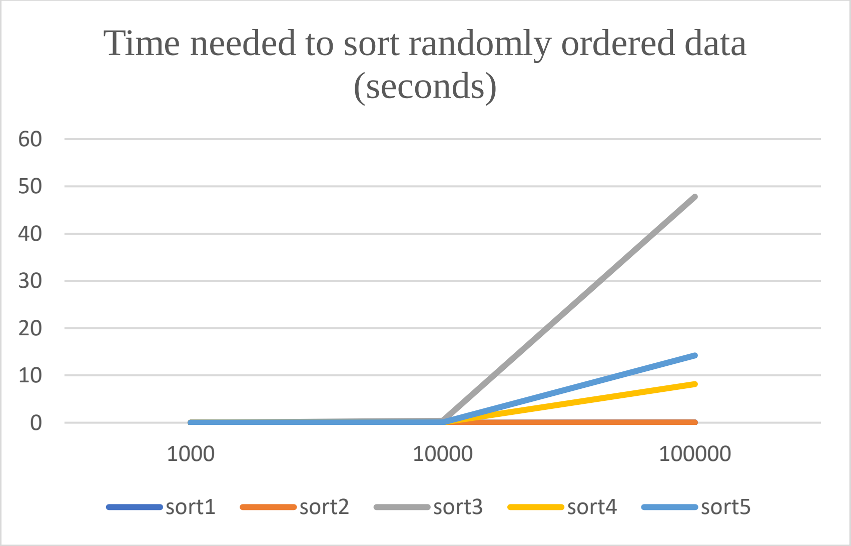
	Random		
Sorting Algorithms (seconds)	Size		
	1000	10000	100000
Sort 1	0.002	0.0062	0.06
Sort 2	0.002	0.0066	0.045
Sort 3	0.009	0.3818	47.815
Sort 4	0.01	0.063	8.16
Sort 5	0.007	0.09	14.2268

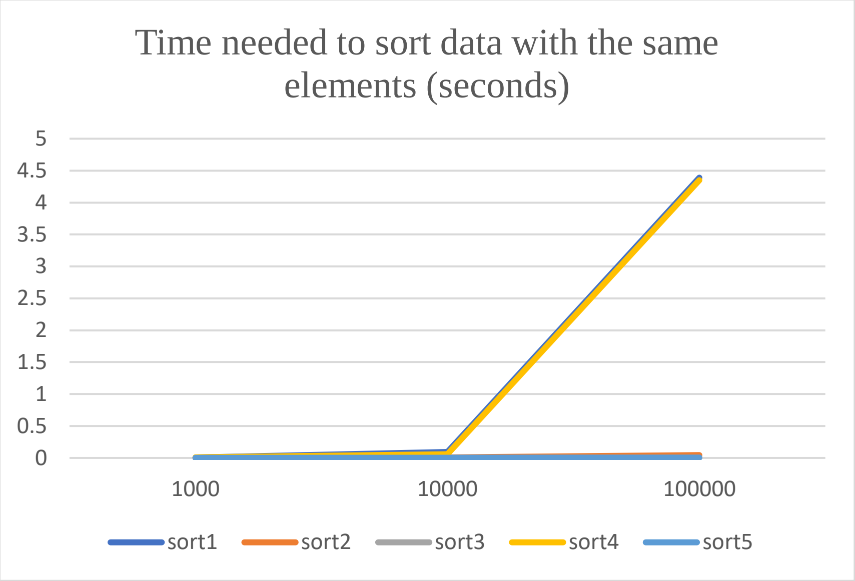
	Same		
Sorting Algorithms (seconds)	Size		
	1000	10000	100000
Sort 1	0.007	0.093	4.39
Sort 2	0.002	0.007	0.045
Sort 3	0.002	0.003	0.006
Sort 4	0.01	0.057	4.35
Sort 5	0.002	0.003	0.005

Linear graphs of table








Time needed to sort data with the same elements (seconds)
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1000
10000
100000
sort1
sort2
sort3
sort4
sort5