Time Measurements

As a exercise in the course 1DV507 - Programing and Data Structures. We are supposed to take a time measurements on algorithms that we have made earlier in the course. This is the report for the time measurements for Insertion sort where we both handle integer and string. The exercise wanted to see how many integers and string could be sorted in one second.

1. Exercises

The last assignment in the course 1DV507 - Programing and Data Structures we are required to write a report on sorting algorithms time measurements. The algorithms we have made is from an another assignment in this course. We were required to make a insertion sort that both handles integers and strings. Which this report will handle how many int and string can be sorted in one second.

2. Experimental Setup

All experiments are made with my xps 15 with an Intel Core i7 processor (2.21GHz) with 16GB of memory. The JVM was allowed a bigger heap space of 4GB using VM arguments: -Xms4096m -Xms4096m. All other application was closed so the performance was no affected by another program. This was made for all experiments. System.currentTimeMillis() was used to get the time that the sorting took. A garbage collection was run before the the sorting was run just so that there wouldn't be any memory problems. This approach was run for every experiment. Each time measurement is an average of 10 consecutive runs. All the array list was a random generated array. The code below is the code we ran for the insertion sort for integers.

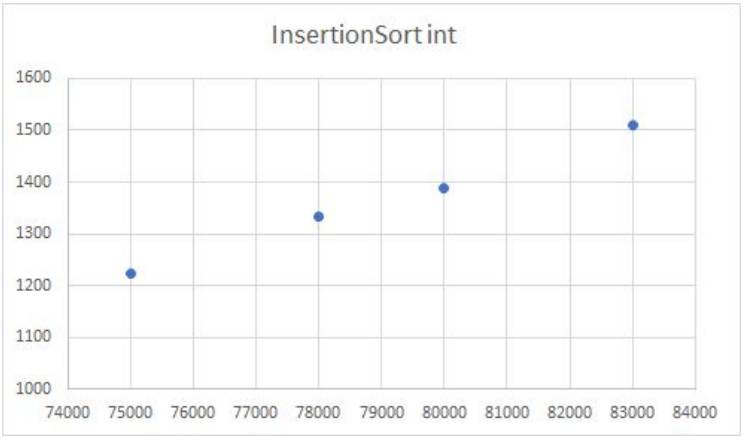
```
public static void main(String[] args) {
		SortingAlgorithms sort = new SortingAlgorithms();

		ArrayList<int[]> randIntArr = randomArr(size, runs);
		mem.gc();
		long start = System.currentTimeMillis();
		for (int i = 0; i < runs; i++) {
			sort.insertionSort(randIntArr.get(i));

		}
		long stop = (System.currentTimeMillis()- start)/runs;
		System.out.print( "time: " + stop);

	}
```

The randomArr method is used for generating a random array of integers. A warm up i required for each experiment to get the JVM fully optimised.



3. Sorting Integers array

Time	Size
1225	75000
1332	78000
1388	80000
1511	83000

Looking at the number tabel and the graph the numbers look OK. This graph don't show any outliers. We also didn't run in to any memory issues. The size is a little to big to be around one second but we can still trust the experiment. Because the numbers looks OK and without any memory problems. Using the data to esteemed the amount of integers to sort in one second using a data trendline.

4. Sorting String array

Time	Size
990	13000
1324	15000
1598	16000
1954	18000
2187	19000

The same as the integers sort the result are OK. without running in to memory problems. The graph also looks OK. And with the graph and the use of a trendline we can estimate the size that it takes one second for the string array.