

CMSC 451 Project 1

The first project involves benchmarking the behavior of Java implementations of one of the following sorting algorithms, bubble sort, selection sort, insertion sort, Shell sort, merge sort, quick sort or heap sort. You must post your selection in the "Ask the Professor" conference. No more than five students may select any one algorithm.

You must write the code to perform the benchmarking of the algorithm you selected. Your program must include both an iterative and recursive version of the algorithm. You do not have to write the sorting algorithms yourself, you may take them from some source, but you must reference your source.

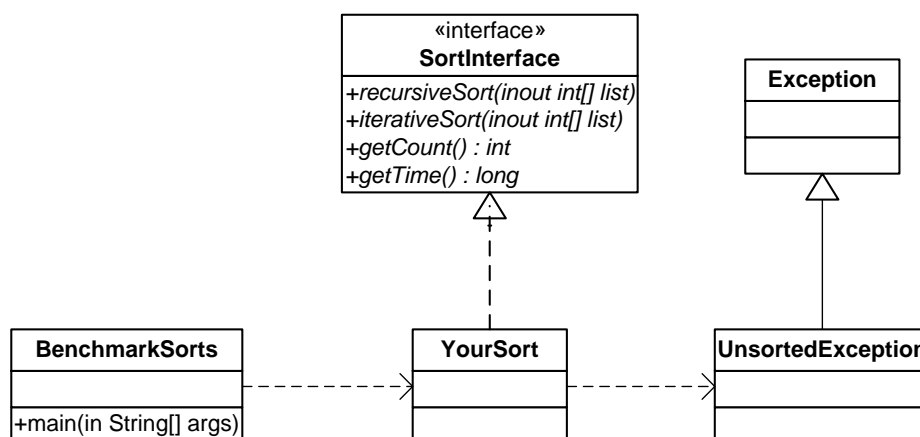
You must identify some critical operation to count that reflects the overall performance and modify each version so that it counts that operation. In addition to counting critical operations you must measure the actual run time in nanoseconds.

In addition, you should examine the result of each call to verify that the data has been properly sorted to verify the correctness of the algorithm. If the array is not sorted, an exception should be thrown.

It should also randomly generate data to pass to the sorting methods. It should produce 50 data sets for each value of n , the size of the data set and average the result of those 50 runs. The exact same data must be used for the iterative and the recursive algorithms. It should also create 10 different sizes of data sets. Choose sizes that will clearly demonstrate the trend as n becomes large. Be sure that the data set sizes are evenly spaced so this data can be used to generate graphs in project 2

This project should consist of two separate programs. The first of those programs should perform the benchmarking described above and generate two data files, one containing the results from the iterative algorithm and the one containing the results of the recursive algorithm.

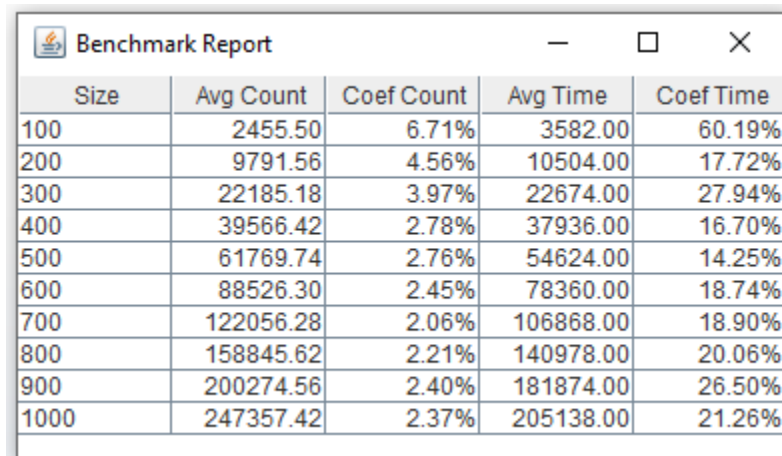
The benchmarking program must be written to conform to the following design:



The output files should contain 10 lines that correspond to the 10 data set sizes. The first value on each line should be the data set size followed by 50 pairs of values. Each pair represents the critical element count and the time in nanoseconds for each of the 50 runs of that data set size.

The second program should produce the report. It should allow the user to select the input file using `JFileChooser`. The report should contain one line for each data set size and five columns and should be displayed using a `JTable`. The first column should contain the data set size the second the average of the critical counts for the 50 runs and the third the coefficient of variance of those 50 values expressed as a percentage. The fourth and fifth column should contain similar data for the times. The coefficient of variance of the critical operation counts and time measurement for the 50 runs of each data set size provide a way to gauge the data sensitivity of the algorithm.

Shown below is an example of how the report should look:



Size	Avg Count	Coef Count	Avg Time	Coef Time
100	2455.50	6.71%	3582.00	60.19%
200	9791.56	4.56%	10504.00	17.72%
300	22185.18	3.97%	22674.00	27.94%
400	39566.42	2.78%	37936.00	16.70%
500	61769.74	2.76%	54624.00	14.25%
600	88526.30	2.45%	78360.00	18.74%
700	122056.28	2.06%	106868.00	18.90%
800	158845.62	2.21%	140978.00	20.06%
900	200274.56	2.40%	181874.00	26.50%
1000	247357.42	2.37%	205138.00	21.26%

On the due date for project 1, you are to submit a .zip file that includes the source code for both programs. All the classes should be in the default package.

You must research the issue of JVM warm-up necessary for properly benchmarking Java programs and ensure that your code performs the necessary warm-up so the time measurements are accurate.

Grading Rubric

Criteria	Meets	Does Not Meet
	100 points	0 points
Design	20 points	0 points
	Implemented the required design (20)	Did not implement the required design (0)
Input	20 points	0 points

	Created 10 different sizes of data sets (10)	Did not create 10 different sizes of data sets (0)
	Produced 50 data sets for each value of n (10)	Did not produce 50 data sets for each value of n (0)
Sorting Algorithm Benchmark Calculations	35 points	0 points
	Correctly averaged the count and time results of the 50 data sets (10)	Did not correctly average the count and time results of the 50 data sets
	Calculated the coefficient of variance of the critical operation counts and time measurement (5)	Did not calculate the coefficient of variance of the critical operation counts and time measurement (0)
	Included correct sorting algorithm and code to verify data was properly sorted (10)	Did not Include correct sorting algorithm and code to verify data was properly sorted (0)
	Performed the necessary warm-up so the time measurements were accurate (10)	Did not perform the necessary warm-up so the time measurements were accurate (0)
Output	25 points	0 points
	Output all the required data (15)	Did not output all the required data (0)
	Output displayed in the required format (10)	Output not displayed in the required format (0)