CSC 402 SP19   Homework 9

In this exercise you will do a timing analysis of several array sorting methods. The goal is to empirically determine the order of growth for an <u>average case</u> for each method. You will use the template form on page 2 of this document to record your data and provide your analysis & observations (make additional copies of the form as needed - copy/paste).

Suggested Procedure.

<u>Prepare Eclipse</u>

1. Start Eclipse using your usual workspace.
2. Download the jar file: *CSC402Sp19.jar* from the D2L submission folder. Drag this file onto the lib folder (Package Explorer view).
3. Right-click the *CSC402Sp19.jar* file, pull-right "Build-path" and choose "Add to Build Path". You should then see "*CSC402Sp19*.jar" listed under the Referenced Libraries folder
4. Download the starter file: CSC402HW9.java.
5. Drag this file to the homework folder
6. Open and Run the file.

<u>Validate the sorting routines</u>

The jar file you added to the library has five sorting methods: Sort0, Sort1, Sort2, Sort3, Sort4. Sort0 is just for demonstration purposes – see the assignment video; you do not need to do anything with Sort0 for this homework. At least one of Sort1 – Sort4 does not work; i.e. does not correctly sort the data. Write a function, isSorted() which will determine if an array is sorted (low to high). Test each of the sorts on an array of random values. Any sort that does not work is exempt from further testing in this assignment. You will indicate which sorts are valid by completing timing analyses for them. I.e. if you turn in completed forms for sorts 1,2,3 that will indicate that you found sort4 to be invalid.

<u>Collect data</u>

The forms provided indicate the array sizes for which you will need to collect data.
You are free to modify/use the provided code framework to carry out/automate data collection.

Requirements:

1. In order to approximate an 'average case', you will need to run each sorting method a number of times on randomized data. This will also improve the accuracy of your *order of growth* estimate AND allow for the slow Stopwatch clock to tick. You should experiment a bit to see what number M of repetitions gives reasonable timing accuracy. Making it too big will result in some sorts taking a very long (20 min?) time. Record the average times in the form along with the value of M used.
2. You may not use any other java classes or libraries.
3. Answer the 3 questions on page1 of the report template.

<u>Analyze the Data</u>

1. Compute the doubling ratio for consecutive array sizes.
2. If possible, determine the order of growth exponent.
3. Graph each set of data individually; your graphs should be similar to the sample one. I suggest you use a spreadsheet like Excel to generate the plots; however you can use pencil/graph paper and then copy/paste the plot into the document if you want. An excel spreadsheet 'template' is provided for your reference.
4. Determine if your order of growth function is consistent with the graph.
5. Note any observations
6. Create one additional chart showing all data sets together (for comparison sake).

Upload: java source file and completed forms (all in one file) to submission folder. Please delete these instructions in the document you submit.

CSC 402 Programming Assignment 9 Report 1.0   Name<u>   **Ximan Liu**   </u>

1. *Describe how you checked to see which sorts worked.*
   I made up a function named isSorted to make sure whether the sorts worked. If it is sorted successfully, then it comes with "true" and vice versa it comes with "false".

2. *Describe how you used your java program to gather the data.*

   My program was completely automated to run all the tests and print out all the results. I just copied the output to the form.

Example answers: (you can use one of these answers if they fit what you did, otherwise delete and replace with your answer).

- I manually ran the program 20 times and added up the times and divided by 20 to get the average for each problem size. I did this for all the working sorts.

- My program was completely automated to run all the tests and print out all the results. I just copied the output to the form.

3. *Based on the evidence, which sorting routine is the best?* _____**4**_____ *Worst?* _____**2**_____

For your reference: You should be able to answer the following questions.

What are the reasons that your program had to run multiple repetitions of each sorting test?

Keep the running time stable and effective. The more data collected, the accurate results of the reaction shows.

What is the purpose of the *control loop?*

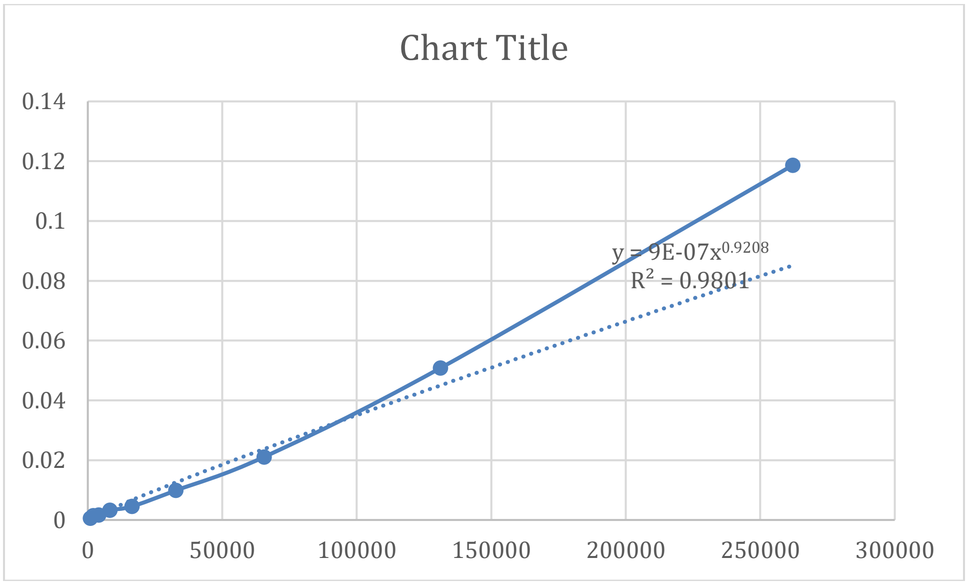
Compute the time for the experimental overhead.

Sort# ___**1**____. (fill in blank)

N	Time	#of Reps	Doubling Ratio
1024	0.000570	100	------------------
2048	0.001400	100	2.46E+00
4096	0.001680	50	1.20E+00
8192	0.003260	50	1.94E+00
16384	0.004520	50	1.39E+00
32768	0.009880	25	2.19E+00
65536	0.021080	25	2.13E+00
131072	0.050800	25	2.41E+00
262144	0.118680	25	2.34E+00

For the model T(N) = cN^b , what is the of order of growth exponent b: ____**1**_____

Plot of N vs Time (replace this sample with your plot)

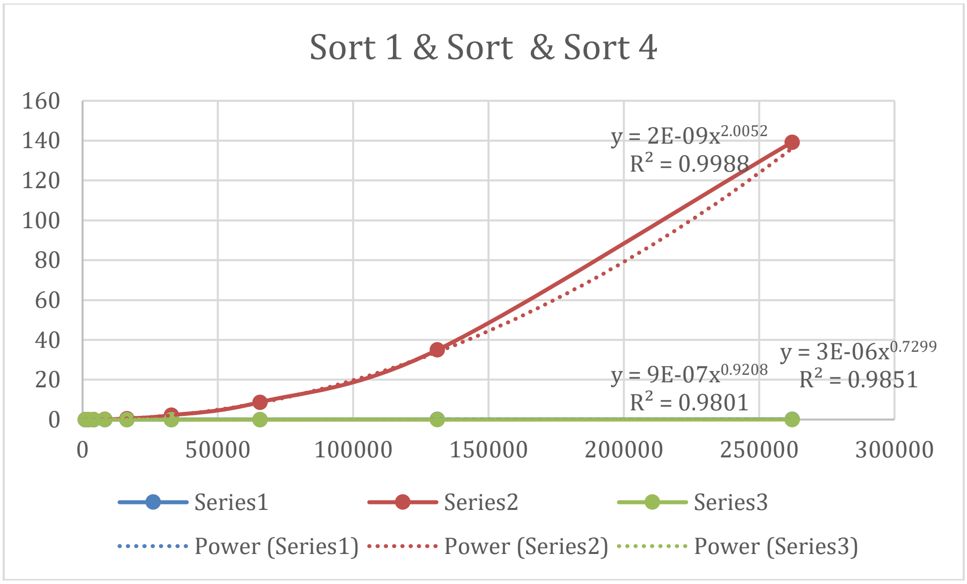


Is the plot consistent with your order of growth estimate? **Yes.**

Observations:

Comparison plot page.

Replace the plot below with your plot of all the sort data sets.



What conclusions can you draw about the sorting routines from your plot?

As N is going to be larger and larger, the differences in running time are getting drastically larger due to the different efficiency of the sorting method. The larger value b is, the more inefficient a sort method will be.
In that case, the worst sorting method is 2, the best one is 4.