CS 2401 Assignment #7

CS 2401 – Elementary data structures and algorithms
Fall 2022: Lab 7

Due Date: Friday, October 28, 2022 – end of the day.

Objective: The goal of this assignment is to learn about how much time it takes to search for a value in an array. We will use two different search algorithms (i.e., linear and binary search) and learn how to use empirical performance testing to compare the speed of these algorithms.

Assignment: We learned about the time complexity of algorithms. In this lab, we will calculate the execution times of different methods. You can find the runtime of a method using a system method that shows the current system time before and after executing the method. The difference between these two times can show us how much time a method took to run.

In Java, you can use the method call `System.currentTimeMillis()` to get the time in milliseconds (variable type: long). If the execution time is very small you can also use `System.nanoTime()`. This method is not as accurate, but you can use it to get greater precision if you consistently see millisecond times of 0.

You will write at least three methods: one that linearly searches an array, the other method that uses the binary search algorithm in iterative format to search an array, and the last method that uses recursive binary search to search an array. Your goal is to determine an empirical estimate of the efficiency of these three algorithms (through the methods).

Description: Implement two search algorithms, linear and binary (iterative and recursive), that work on an array of doubles.

> `Search.java` should contain at least three methods, each with one of the three following search techniques: (1) a linear search to search in an array of doubles, (2) a binary search method to search in an ordered array in iterative manner, and (3) a recursive binary search method to search in an ordered array. Each method must **return** the location of the element, if the element is found in the array. If the element is not in the array, return -1.
>
> All methods will have two arguments: an array of double and a number to search for. You will generate random numbers to populate the array. You will also generate a random index to search for in the array (more details next).
>
> **More details:** To get an accurate estimate of the time taken, you will need to

generate a number of test cases and average the resulting times. Suppose first that we aim to estimate the time taken for an array of 10000 numbers. Here are the steps to perform:

1. Generate a new double array with 10000 random numbers using `Math.random()`.
2. Select a random index using `Math.random()`: Select the value to search for by selecting a random index between 0 and 9999 (so you can be sure the value is in the array). Use the value of that random index as the number to search for.

   For example, a random index to search for maybe 712 and the value in that index may be 3992.7. So, we will search the array for 3992.7 using linear search and binary search (iterative and recursive).

3. Sort the array using `Arrays.sort()`. Remember that binary search works on ordered arrays only. Record the time it takes to run the sort algorithm.
4. Run linear and binary search (iterative and recursive) for the same element on the sorted array, and record the elapsed time for each algorithm.
5. Repeat the above steps at least 20 times (or, more if your computer is too fast to capture the runtime) using a for loop and calculate the average time taken by each algorithm. This will be more accurate than doing the test only once.

Now that you are able to test the performance of the linear and binary search algorithms on arrays of 10000 elements, the next step is to test problems of several different sizes (20000, 30000, 40000, 70000, 100000, etc.) to see how the runtime changes as the problem size increases. The input sizes may vary based on the performance of your computer)

Generate a plot in Excel showing the performance trend of the three methods, with the array size on the X-axis and the average runtime on the Y-axis. Draw three lines, one for the linear search, one for the iterative binary search, and the last one for the recursive binary search algorithm, **on the same plot**. You can output the results in the terminal and copy the values into Excel by hand. Your PDF report must contain the excel plot.

Generate another separate plot in Excel to show the performance of java's array sort method. For different array size, measure the time taken by the `Array.sort()` method to sort the array. The X-axis should show the size of the array, and Y-axis should show the average runtime of the sort method. This plot should have one line because we are measuring the time for one sort algorithm.

CS 2401 Assignment #7

(Prepared by: Dr. Monika Akbar. This document is not for public distribution.)

Report: In addition to your codes, submit a short report in a docx file (`Report.docx`) describing your experiments and observations. The MS Word document must contain the plots you have drawn using excel. The report should include following information:

1. A description of the running environment (computer, processor, memory, OS, Java version),
2. description of the experiments including the steps that you took to gather the data,
3. the results (i.e., the Excel plots you generated), and
4. a discussion of factors that could influence the accuracy of the results.
5. Provide big O notations for each of the algorithms (a) linear search, and (b) iterative binary search, and (c) recursive binary search.
6. Using the 2nd chart provide an estimation of the big O notation for the `Array.sort()` algorithm.
7. Explain how your runtime plots relate to the big O notations.

Lastly, convert the `Report.docx` file to a PDF file (`Report.pdf`). Submit the java file, docx file, and the pdf file.

Deliverables: You are expected to submit a total of three files: `Search.java`, `Report.docx`, and `Report.pdf`. All the files must be submitted using Blackboard.

Grading Criteria:

- [30 points] The Program **compiles and runs**.
- [10 points] The program is **indented** and **documented** properly.
- [10 points] The program uses the correct **variable types** and **names**.
- [50 points] The report includes items 1-7 under the **Report** section on this document.

- Late submission: [**-10**] points for every 24 hours after the deadline.

If you need any clarification, please ask your TA for further details.