Joanna Klukowska
joannakl@cs.nyu.edu

Programming Project 2: Benchmarking Sort Algorithms

Due date: October 25, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own. Any sharing or copying of assignments will be considered cheating. If you get significant help from anyone, you should acknowledge it in your submission.

In this assignment you will evaluate relative performance of different sorting algorithms. We discussed them all in class and several different versions of code are in the textbook, OpenDSA website and other resources that you are using. Your job is to design a benchmark program for performance evaluation.

Problem Description

We spent some time discussing performance of the algorithms, but in practice what one often wants to test is how a given implementation performs on a given machine. You will write code for several implementations of the sort algorithms and evaluate their performance.

Problem Solution

The outline of your program should be as follows:

- Generate an input array to be used for different sort implementations.
- For each of the implemented sort algorithms
  - make a copy of the input array
  - start the timer
  - run the sort algorithm on the array
  - stop the timer
  - display the running time **in milliseconds.**

Implementation Details

Generating Data

The program needs to generate an array (not an ArrayList) that you'll sort using different methods. You need to decide on the type of the array to use for testing. It is totally up to you. Your program needs to populate the array with random values corresponding to whichever type you choose.

Before running each of the sort methods on the array, you should make a copy of it and run your implementation of the algorithm on the copy. You have to make sure that each algorithm starts with the exact same array as the previous one. You can use `System.arraycopy()` if you wish.

Sorting Algorithms

You should implement three sorting algorithms: selection sort, merge sort and quick sort. You must implement them using generic methods. Your implementation should follow the pseudocode outlined in the lectures.

Joanna Klukowska
joannakl@cs.nyu.edu

Sorting and Timing

The program needs to measure the time it takes for each of your three methods to complete sorting for several different sizes of the array.

It is up to you to

- either design a program that generates an array of a particular size and then run it multiple times,
- or design a program that runs all different sizes on its own.

In either case, your program should not be interactive (no user input should be expected).

Use `System.nanoTime()` method to get the time before and after the call to the sort function. The difference between the two will tell you how many nanoseconds it takes for the algorithm to complete its task. You will need to covert this result to milliseconds (just divide by 1000).

Report

You need to write a short report based on multiple runs of your program using different sizes of the array. Combine the numerical results into a table and produce graphs that demonstrate how the time changes as a function of number of elements. The template at
https://docs.google.com/spreadsheets/d/14nEYcG5bto6C5eIXt6j96ekyJAR62Px1nxXTtwldi34/edit?usp=sharing
should be used to generate the tables and graphs (just fill in your timing results and the graphs will get created automatically).

Your report should include the tables and graphs from the above spreadsheet (you can just copy and paste, or export it as PDF and attach to your document). It should also discuss what you observed and if the results were as expected or not.

You should try to optimize your implementation and include the discussion of changes you made and how they affected the time performance. Changes may sometimes decrease the performance - that's ok. You should try to explain why a particular change affected the time performance in a particular way.

NOTE: The template for the tables and graphs contains the sizes of the arrays that you should use. If your computer cannot handle the largest arrays (or it takes extremely long time) you may stop before.

Working on This Assignment

You should start right away! The code itself is not hard to develop, but you may want to spend some time optimizing and fine-tuning you implementations. You also need to complete your code in time to prepare the report and add improvements.

Grading

20 points - completeness and correctness of the report

15 points - code documentation (using Javadoc and inline comments)

20 points - development and correctness of the application (setting up the arrays, running and timing of the different sort implementations)

45 points - development and correctness of sorting methods

How and What to Submit

Your should submit the report as a PDF document and all of the source code files (the ones with .java extensions only) in a single **zip** file to NYU Classes.

Make sure that all the code is properly documented (using Javadoc and inline comments). All files have to contain a preamble (or class documentation) that contains your name. All the code should be your own. This is not a collaborative project.