Programming Assignment 2 - Quick Sort

Re-submit Assignment

Due May 3 by 11:59pm **Points** 100 **Submitting** a file upload
File Types java and cs **Available** Apr 21 at 6pm - Jun 18 at 11:59pm about 2 months

Introduction

For this programming assignment you will implement Quick Sort and execute it with 4 separate data sets. Each data set is a 1,000,000 element integer array, organized as follows:

- random
- nearly sorted ascending (sorted ascending data set with small number of elements out of order)
- sorted ascending
- sorted descending

There are a number of different implementations of Quick Sort. Your implementation should start with the pseudo coded Quick Sort algorithm presented in figure 3.5.1 of the zybook used for this course, with the modifications that follow.

Add a call to System.nanoTime() immediately before, and immediately after the call to Quick Sort to calculate how much time elapsed during the sorts. Output these values along with a description of what's being measured.

Median-of-Three Partitioning

The optimal choice for a pivot would be the median of the array. However, this is difficult to calculate and would slow down Quick Sort significantly. One approach is to pick three elements randomly , and use the median of these three as the pivot. As it turns out, the randomness doesn't help much. So instead, just use median of the first, last, and center elements as the pivot. For example, if you have the following values for input:

```
        17,  3,  9,  19,  13,  7,  11,  5,  15,  1
```

the first value is 17, the last value is 1, and the center value is 13. So in this case the pivot is 13.

Modify your implementation of Quick Sort to choose a pivot using Median-of-Three.

When the Sub-Arrays Get Small

Quick Sort is a recursive divide and conquer algorithm. As the recursive descent progresses, sooner or later the sub-arrays will get small enough that Quick Sort doesn't perform as well as other sorting algorithms. One possible remedy for this phenomenon is to switch to a sorting algorithms that is efficient for small arrays.

Modify your implementation of Quick Sort to switch to Selection Sort when the length of a sub-array is less than or equal to 20 elements.

Modify TestData.java

The current version of TestData.java (provided in this weeks Canvas Module) provides sorted ascending data and random data. You will need to add two new methods to the class. One for generating sorted descending data, and one for nearly sorted ascending data.

What to Turn In

When you have finished the assignment, upload your .java files to Canvas.