Assignment 1

This assignment is designed to introduce you to sorting and searching algorithms, and to consider their growth rates in terms of Big-O notation.

Description

You will be provided a working bubble sort routine, and a working linear search routine. These have been "enhanced" to capture extra data, specifically the number of comparisons made, the number of element swaps made, and the processing time required to complete each task.

One part of your assignment will be to add a binary search routine, plus insertion and selection sorts, all similarly enhanced.

A second part of your assignment will be to run your code on various sizes of input (arrays), to see how they behave in terms of performance (details below).

Lastly, you should identify which growth algorithm best describes each of the various searches and sorts using "Big-O" notation. For example, we know that the growth rate associated with the aptly named linear search is linear, so it would be O(n).

Overview

The project zip file contains the following five files (in addition to the assignment document):

1. Sort.java
2. Search.java
3. SortReturn.java
4. SearchReturn.java
5. TestSearchAndSort.java

Files Overview and requirements

Details on the five files.

Sort.java

Sort.java contains an implementation of the bubble sort algorithm. It has been enhanced to track the number of comparisons made, the number of swaps made when numbers are out of order, and also to track the processing time in nanoseconds.

SortReturn.java

SortReturn.java is an object designed to return the performance information from the Sort.bubble() method. Remember that a method can only return a single data element, so we're taking advantage of an object's ability to be that one thing while containing multiple pieces of information. Therefore, Sort.bubble()'s return type is this SortReturn class. It also has a toString() method to dress up its display.

Search.java

Search.java contains an implementation of the linear search algorithm. Like the sort routine above, it has been enhanced to track the number of comparisons made, the number of swaps made when numbers are out of order, and the processing time in nanoseconds.

SearchReturn.java

Like SortReturn.java above, the purpose of this class is to pass the performance data back to the client after executing Search.linear().

TestSearchAndSort.java

TestSearchAndSort.java is a basic program designed to execute Sort.bubble() and Search.linear(). This can serve as the starting point for executing your binary search and insert and selection sort algorithms.

Requirements

1. Implement Sort.selection(), Sort.insert() and Search.binary(), all enhanced to collect the same performance data as included in the sample Sort.bubble() and Search.linear().
2. Modify TestSearchAndSort.java to
   - run each sort (bubble, selection, and insert) for each of three sized arrays (100, 1000, and 10000), for a total of nine combinations.
   - run each combination of search (binary and linear) and arrays of sizes 100, 1000, and 10000, and searching for elements in the following positions: 10%, 50%, and 90%. Also search for an element that you know will not exist in the array at all. So, 2 search algorithms, 3 array sizes, and 4 positional searches (including one not in the array), producing 24 result sets.
3. Identify the growth rate of each search and sort routine using Big-O notation in a program comment associated with each method.

Deliverables

You should submit a zip file containing all your java source code. Your TestSearchAndSort.java should compile, and executing it should clearly show all the combinations specified above.