22c:31 Project 2: Due 10/21/2013

For this assignment, you will modify the code `sorting.java` to fulfil the following requirements.

- Problem 1: Implement the insertion sort algorithm which can sort the subarray of the array `arr` (from `low` to `high`, inclusive):
  `private static void insertSort(int low, int high)`

- Problem 2: Implement the following method which checks if the subarray of the array `arr` (from `low` to `high`, inclusive) is already sorted:
  `private static boolean isSorted(int low, int high)`

- Problem 3: Create three versions of the mergesort algorithm such that (a) one version uses `insertSort` when the number of elements to be sorted is less than 100; (b) one version uses `isSorted` before recursive calls of mergesort: if the subarray is already sorted, then skip the call; (c) the last version uses both ideas of (a) and (b). Compare the performances of three new versions with the original version of mergesort on the same set of examples: 10 instances of 1,000,000 elements, 10 instances of 10,000,000 elements, and 10 instances of 50,000,000 elements (all numbers are in the arrange of 10,000). Give a summary on the average performances of these four versions of mergesort and your conclusions.

- Problem 4: Create two versions of the quicksort algorithm which uses different pivots: One uses the median of three elements and one uses the median of three medians (of three elements).

- Problem 5: Repeat Problem 3 for the six versions of the quicksort algorithm: (a) one uses `insertSort` for small subarrays; (b) one uses `isSorted` before recursive calls; (c) one uses the median of three for pivot; (d) one uses the median of three medians for pivot; (e) one uses (a), (b), and (c); (f) one uses (a), (b), and (d), plus the java built-in Arrays.sort.

Please submit your the modified code along with a transcript of its execution in the ICON dropbox for Project2 before or on 10/21/2013.

Thank you!
