Fuzzy Sorting

Mostly from Problem 7-6 from textbook, but includes an addition in part c.

Consider a sorting problem in which we do not know the numbers exactly. Instead, for each number, we know an interval on the real line to which it belongs. That is, we are given n closed intervals of the form $[a_i, b_i]$, where $a_i \leq b_i$.

We wish to fuzzy-sort these intervals, i.e., to produce a permutation $\langle i_1, i_2, \dots, i_n \rangle$ of the intervals such that for $j = 1,2, \dots, n$ there exist $c_j \in [a_{i_j}, b_{i_j}]$ satisfying $c_1 \leq c_2 \leq \cdots \leq c_n$.

a) Design a randomized algorithm for fuzzy-sorting n intervals. Your algorithm should have the general structure of an algorithm that quicksorts the left endpoints (the a_i values), but it should take advantage of overlapping intervals to improve the running time. (As the intervals overlap more and more, the problem of fuzzy-sorting the intervals becomes progressively easier. Your algorithm should take advantage of such overlapping, to the extent that it exists.)

b) Argue that your algorithm runs in expected time $\Theta(n\lg n)$ in general, but runs in expected time $\Theta(n)$ when all of the intervals overlap (i.e., when there exists a value x such that $x \in [a_i, b_i]$ for all i). Your algorithm should not be checking for this case explicitly; rather, its performance should naturally improve as the amount of overlap increases.

c) Use the two provided files with non-overlapping and overlapping intervals to time your algorithm in each case. Use different values of n (i.e. use the first n intervals in the files), time the execution time of the algorithm, and plot as a function of n. Can you observe $\Theta(n\lg n)$ and $\Theta(n)$ behavior in your plots?