Problem 1.  [20 points]
KT Chapter 5 - Problem 5. Recall the problem of finding the number of inversions. As in the text, we are given a sequence of \( n \) numbers \( a_1, \ldots, a_n \), which we assume are all distinct, and we define an inversion to be a pair \( i < j \) such that \( a_i > a_j \). We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let's call a pair a significant inversion if \( i < j \) and \( a_i > 2a_j \). Give an \( O(n \log n) \) algorithm to count the number of significant inversions between two orderings.

Problem 2.  [20 points]
KT Chapter 5 - Problem 3. Suppose you're consulting for a bank that's concerned about fraud detection, and they come to you with the following problem. They have a collection of \( n \) bank cards that they've confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we say that two bank cards are equivalent if they correspond to the same account.

It's very difficult to read the account number off a bank card directly, but the bank has a high-tech equivalence tester that takes two bank cards and, after performing some computations, determines whether they are equivalent. Their question is the following: among the collection of \( n \) cards, is there a set of more than \( n/2 \) of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only \( O(n \log n) \) invocations of the equivalence tester.

Problem 3.  [20 points]
KT Chapter 5 - Problem 6. Consider an \( n \)-node complete binary tree \( T \), where \( n = 2^d \) for some \( d \). Each node \( v \) of \( T \) is labeled with a real number \( x_v \). You may assume that the real numbers labeling the nodes are all distinct. A node \( v \) of \( T \) is a local minimum if the label \( x_v \) is less than the label \( x_w \) for all nodes \( w \) that are joined to \( v \) by an edge.

You are given such a complete binary tree \( T \), but the labeling is only specified in the following implicit way: for each node \( v \), you can determine the value \( x_v \) by probing the node \( v \). Show how to find a local minimum of \( T \) using only \( O(\log n) \) probes to the nodes of \( T \).

Problem 4.  [20 points] - No help problem
a) For the recursive algorithm defined to solve the weighted interval scheduling problem, shown in page 254. Explain, in your own words, why the algorithm correctly finds the optimal schedule.

b) Again, in your words, analyze the running time complexity of the iterative algorithm presented in page 258.