1. Create an ordering of the numbers 1 through 8 that will cause mergesort to do the worst-case number of comparisons of 17. *Hint:* work backwards through the sorting process. Show a recursive trace of merge-sorting your numbers and a count of the comparisons at each level of the trace.
2. 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 chip with some encrypted data. Each card corresponds to a single account in the bank, but an account may have many cards corresponding to it. So, in other words, a customer might have only a single account at the bank, but they could have many cards for that account (all such cards would have the same account number). Two or more bank cards are considered equivalent if they correspond to the same account. The bank has determined that if a majority of the confiscated cards have the same account number, then there is a high probability of fraud.

It’s very difficult to automate the reading of the account number from a bank card directly, so the bank has developed a high-tech “equivalence tester.” This tester takes two bank cards, and, after performing some computations, determines whether they are equivalent.

Their question to you is the following: Among a given collection of \( n \) cards, is there a set of more than \( n/2 \) of them are all equivalent to one another? Assume that the only feasible physical operations that can be performed with the cards are to pick two of them and insert them into the equivalence tester. Show how to decide the answer (high probability of fraud or not) using a divide-and-conquer approach, using only \( n \log n \) invocations of the equivalence tester. Can you find the answer in linear time?