

4. *Kleinberg, Jon. Algorithm Design (p. 246, q. 3).* You're consulting for a bank that's concerned about fraud detection. They have a collection of n bank cards that they've confiscated, suspecting them of being used in fraud.

It's difficult to read the account number off a bank card directly, but the bank has an "equivalence tester" that takes two bank cards and determines whether they correspond to the same account.

Their question is the following: among the collection of n cards, is there a set of more than $\frac{n}{2}$ of them that all correspond to the same account? 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.



5. Implement the optimal algorithm for inversion counting in either C, C++, C#, Java, or Python. Be efficient and implement it in $O(n \log n)$ time, where n is the number of elements in the ranking.

The input will start with a positive integer, giving the number of instances that follow. For each instance, there will be a positive integer, giving the number of elements in the ranking. A sample input is the following:

```
2
5
5 4 3 2 1
4
1 5 9 8
```

The sample input has two instances. The first instance has 5 elements and the second has 4. For each instance, your program should output the number of inversions on a separate line. Each output line should be terminated by a newline. The correct output to the sample input would be:

```
10
1
```