Divide and Conquer

A complete solution to a problem will include the following elements:

- a recursive algorithm to the problem
- an explanation *or* formal proof of why that formulation is correct
- pseudocode showing how to compute the solution in a recursive way
- an analysis of the running time.

For each problem you may assume that the size of the input to the problem is a power of 2.

1. A *fixed point* of an array $A[1..n]$ is an index i such that $A[i] = i$. Given a sorted array of *distinct* integers $A[1..n]$ as input, give a divide-and-conquer algorithm to determine if A has a fixed point that runs in time $O(\log n)$.

2. For a sequence of n numbers $a_1, .., a_n$, a *significant inversion* is a pair (a_i, a_j) such that $i < j$ and $a_i > 2a_j$. Assuming each of the numbers a_i is distinct, give an $O(n \log n)$ time algorithm to count the number of significant inversions in a sequence. (Hint: modify merge sort.)

3. You are given two sorted arrays of size m and n. Give an $O(\log m + \log n)$ time algorithm for computing the k-th smallest element in the union of the two arrays.

4. You are given an $n \times n$ matrix $A[1..n, 1..n]$ where all elements are distinct. We say that an element $A[x]$ is a *local minimum* if it is less than its (at most) four neighbors, i.e. its up, down, left and right neighbors. Give an $O(n)$ time algorithm to find a local minimum of A.