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. Given a sorted array of distinct integers $A[1..n]$, you want to find out whether there is an index $i$ for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(\log n)$.

2. For a sequence of $n$ numbers $a_1, \ldots, a_n$, which we assume are all distinct, we define a significant inversion to be a pair $(a_i, a_j)$ such that $i < j$ and $a_i > 2a_j$. Give an $O(n \log n)$ time algorithm to count the number of significant inversions in this 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 all its (at most four) neighbors, i.e. the up, down, left and right neighbors. Elements on the boundary can have less neighbors. Give an $O(n)$ time algorithm to find a local minimum of $A$. 