Problem 1: The law firm of Gennaro & Gennaro is looking for a new office in Manhattan. Because they often have to hand deliver important parcels to their clients, they are looking for a location that minimizes the sum of the distances between their new office and the clients’ locations. They have \( n \) clients \( C_1, \ldots, C_n \), where each client is represented by a point in the plane \( C_i = (x_i, y_i) \). Because this is Manhattan where we travel on a grid, the distance between the new office \( O = (x, y) \) and the client location is computed as \( d(O, C_i) = |x - x_i| + |y - y_i| \). So the firm is looking for a location \( O = (x, y) \) such that

\[
\delta(x, y) = \sum_{i=1}^{n} d(O, C_i) = \sum_{i=1}^{n} (|x - x_i| + |y - y_i|)
\]

is minimized.

- Give an algorithm that computes the above location in \( O(n) \) time.

- A very smart intern at the firm notices that some clients receive parcels more often than others. He suggests to the partners that the right quantity to minimize is 

\[
\delta_f(x, y) = \sum_{i=1}^{n} f_i \cdot d(O, C_i)
\]

where \( f_i \) is the frequency with which client \( C_i \) receives parcels. In other words the \( f_i \)'s are real numbers such that \( 0 \leq f_i \leq 1 \) and \( \sum_{i=1}^{n} f_i = 1 \). Modify your previous algorithm to determine a location that minimizes the above quantity. Your algorithm should still work in \( O(n) \) time.

*Hint: For the first part, use the median algorithm. To simplify things consider first the one-dimensional case in which points are on a line, not on the plane. Each client is a single coordinate \( C_i = x_i \) and the distance is \( |x - x_i| \). Does the median of the \( x_i \) minimize the distance? For the second part, modify the notion of median to take into account the frequencies. Assume without loss of generality that \( n \) is an odd number.*

Problem 2: As we will see later in the class, a binary search tree (BST) is a data structure organized as a binary tree where each node \( x \) holds a value \( key[x] \). The important property of a BST is that for any subtree of a BST rooted at \( x \), we have that \( key[x] \) is larger than the keys of all the nodes stored on the left subtree of \( x \), and smaller than the keys of all the nodes stored on the right subtree of \( x \). An interesting consequence of this is that by performing an ”inorder” walk of the tree it is possible to output the keys stored in the tree in sorted order. An inorder walk of the tree rooted at \( x \) is the following recursive procedure

**INORDER(x)**

1. If \( x \neq \text{NIL} \):
   - **INORDER(Left[x]);**
   - **PRINT key(x);**
   - **INORDER(Right[x])**

Notice that **INORDER** makes no comparisons since it only reads the elements in the tree in a specific order.
Given $n$ elements can you build a BST containing the elements in $O(n)$ time? If yes show your algorithm.
If no, explain why you think it is impossible.

**Problem 3:** On input an array $A$ of $n$ elements, each of which is an integer in $[0..n^4]$, describe a simple method for sorting $A$ in $O(n)$ time.

*Hint: think of alternative ways of viewing the elements.*