When asking for designing an algorithm, both English description and pseudo-code are required. When asking for an efficient algorithm, the score will depend on how efficient your algorithm is.

1. (20 points) Suppose the array A contains [1, 6, 4, 9, 7]. (a) Please display the contents of the array whenever its content changes during the execution of the heapsort algorithm. (b) Please do the same for the quicksort algorithm where the first element in the subarray is used as pivot.

2. (20 points) You are inserting the following elements 1, 6, 4, 9, 7, in the given order into a binary search tree. Please display the sequence of the trees whenever the tree changes if the tree is (a) AVL trees; (b) 2-3 trees.

3. (15 points) Given an $m$ by $m$ matrix $A$ and an integer $n$, provide an efficient algorithm to compute $A^n$ ($n$ multiplications of $A$) and analyze its complexity.

4. (15 points) Given two sorted lists of $n$ elements, the merge algorithm will merge them into one sorted list of $2n$ elements. Please provide a function $f(n)$ which computes the number of possible outcomes of the merge algorithm. For example, $f(1) = 2$ (i.e., either $ab$ or $ba$, assuming the first list contains $a$ and the second list contains $b$), and $f(2) = 6$ (i.e., if one list is $a_1a_2$ and the other list is $b_1b_2$, then there are six possible outputs: $a_1a_2b_1b_2$, $a_1b_1a_2b_2$, $a_1b_1b_2a_2$, $b_1a_1a_2b_2$, $b_1a_1b_2a_2$, $b_1b_2a_1a_2$).

5. (15 points) Two binary search trees are said to be equivalent if they store the same set of elements. Provide an efficient algorithm to check if two binary search trees are equivalent and analyze its complexity. For your pseudo code, you are allowed to use the following methods on a binary search tree: Given a tree node $x$, isEmpty($x$) returns true if $x$ is empty. If $x$ is not empty, key($x$) returns the key value of $x$; left($x$) returns the left child of $x$ if $x$ is not empty; right($x$) returns the right child of $x$. Key values can be compared using $<$, $==$, or $>$.

6. (15 points) Given a list of numbers, an inversion is a pair of numbers that are out of order. For example, there are three inversions in the list [1, 4, 3, 2], i.e., (4, 3), (4, 2), and (3, 2). Please provide an efficient algorithm to count the number of inversions of a given list and analyze its complexity.