Problem 1. Solve the following recurrence by using (a) substitution method, (b) recursion tree method. Simplify the final expression as much as possible for full credit.

\[
T(n) = \begin{cases} 
2/7 & \text{if } n = 1 \\
T(n) = 2T(n/3) + n^2 & \text{otherwise}
\end{cases}
\]

Problem 2. Use the Master Theorem to derive an asymptotic bound on the following recurrences.

(a) \(T(n) = 2T(n/4) + n\sqrt{n}\).
(b) \(T(n) = 2T(n/3) + 5^{\log_2{n}}\).

Problem 3. Suppose you are given an array \(A\) containing \(n - 1\) unique integers in the range \([0, n-1]\) in sorted order (smallest to largest). Give pseudocode for an \(O(\log n)\) algorithm for finding the integer in the range \([0, n-1]\) that is not in \(A\).

Problem 4. Let \(A[1..n]\) be an array of \(n\) distinct numbers. If \(i < j\) and \(A[i] > A[j]\), then the pair \((i, j)\) is called an inversion.

(a) List the five inversions of the array \([2,3,8,6,1]\).
(b) Give a worst-case \(\Theta(n \log n)\) algorithm that determines the number of inversions in \(A\). Explain your algorithm, give an example and derive its running time.