Problem Set 1

1. Prove the mentioned upper bounds (given in the brackets) for the following recurrence relations. The base case is $T(2) = 1$ for all the relations. Try both the techniques – (i) the range and domain transformations method and (ii) the recurrence tree method.

   (a) $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$ $(\mathcal{O}(n \log \log n))$

   (b) $T(n) = 4T(\frac{n}{2}) + n \log n$ $(\mathcal{O}(n^2))$

   (c) $T(n) = 2T(\frac{n}{2}) + n \log n$ $(\mathcal{O}(n(\log n)^2))$

   (d) $T(n) = 2T(\frac{n}{2}) + \frac{n}{\log n}$ $(\mathcal{O}(n \log \log n))$

2. An inversion in an array $A[1, \ldots, n]$ is a pair of indices (i, j) such that $i < j$ and $A[i] > A[j]$. The number of inversions in an array of length n is between 0 (if the array is sorted) and $\binom{n}{2}$ (if the array is sorted in the decreasing order and all the elements are distinct). Design an algorithm to count the number of inversions in an n-element array in $\mathcal{O}(n \log n)$ time. [Hint: modify merge sort]

3. You are given an array containing n integers (Notice that the array contains both positive and negative integers). Design an algorithm to find the sum of contiguous subarray of numbers which has the largest sum. prove its correctness and analyze its running time.

4. You are given an array $A[1, \ldots, n]$ that contains n distinct elements from $\{1, \ldots, n+1\}$ and the elements are in the increasing order in A. Design an $\mathcal{O}(\log n)$ time algorithm to find the missing element from $\{1, \ldots, n+1\}$ in A. That is, find the element in $\{1, \ldots, n+1\} \setminus \{A[1], \ldots, A[n]\}$.