Remarks: In all the algorithms, always explain their correctness and analyze their complexity. The complexity should be as small as possible. A correct algorithm with large complexity, may not get full credit.

Choose 5 out of the next 6 questions.

Question 1: You are given a sorted array A and a number x. We know that x belongs to the array, namely, A[k] = x for some k. Give an algorithm to find x in A that works “well” if k is “close” to n or “very small” (close to 1). The running time should depend on n and k.

Example: For the best solution, if k = n - constant the running time should be constant, etc.

Question 2: You are given a black-box algorithm \( \mathcal{A} \) that can sort up to \( p \) elements in \( O(p) \) time. However, \( p \) may be much smaller than \( n \) (say \( p = \log n \), or \( p = \sqrt{n} \) or \( p = O(1) \)). Write an algorithm that uses \( \mathcal{A} \) to sort. The running time should be a function of \( n, p \). If \( p = n \) the running time is \( O(n) \) and if \( p = 1 \), the running time is \( O(n \log n) \). Try to make the running time go down if \( p \) goes up.

Question 3: Answer true or false for all the following questions and explain your answer precisely.

1. Say that we can find the \( \lceil n/2 \rceil \) largest number (the median) in an array of \( n \) elements in \( O(n) \). Using that we can make Quicksort run in \( O(n \log n) \) time (worse case of course).

2. The \( i \) and the \( i+1 \) largest elements in an array must be compared in a (comparison based) sorting algorithm

3. There are cases so that two sorted arrays \( A \) and \( B \) each with \( n \) elements require \( 2 \cdot n - 1 \) comparisons to merge.

4. The worse cases and the best case of Mergesort perform the same number of comparison as a function of \( n \) and even with the same constant

Question 4: Solve the subset sum problem for items \( a_1, a_2, \ldots, a_n \) and target \( S \) if the following property holds. For every \( i, a_i > \sum_{j=1}^{i-1} a_j \)

Question 5: Let \( S = \{a_1, a_2, \ldots, a_n\} \) be a set of positive integers. A contiguous subset of \( S \) is a a subset \( \{a_i, a_{i+1}, \ldots, a_j\} \), \( i \leq j \) that contains all the elements \( a_p, \ i \leq p \leq j \) (including \( a_i, a_j \)). Give an algorithm that computes the number of contiguous subsets of \( S \) whose sum of elements is divisible by 3
Question 6: Give an algorithm to compute the number of (not necessarily contiguous) subsets of $S$ whose sum of elements is divisible by 3