COL100: Assignment 6

Note: Answer to each question should be accompanied with the correctness argument and analysis for time complexity

1. Given an array a[0, , n-1] of n nonzero integers ordered in the ascending order, develop an algorithm to remove all duplicates by replacing a duplicate with 0 value; the algorithm should also return the effective size of the output array. An instance of an ordered array before duplicate removal is: |2|2|4|8|8|23|37|37|42|. After the duplicate removal we have the following array: |2|4|8|23|37|42|0|0|0 with an effective size of this output array to be 6.

2. Given a randomly ordered array of n elements partition the elements into two subsets such that elements less than x are in one subset and elements greater than x are in separate subset. For instance, |28|26|25|11|16|12|24|29|6|10| and $x = 17$ lead to the array |10|6|12|11|16|25|24|29|26|28|. We provide here a high level algorithmic description of the problem.

   - Establish the array $a[0, ..., n-1]$ and the partitioning value x.
   - Move the partitions towards each other until the wrongly placed elements are encountered. Allow for special cases such as x being outside the range of array values.
   - While the two partitions have not crossed over
     - exchange the wrongly partitioned pair and extend both partitions inward by one element
     - extend left partition while elements less tha or equal to x
     - extend the right partition while elements are greater than x
     - Return the partitioning index p in the partitioned array.

3. Give an algorithm to find the k^{th} smallest element in a given randomly ordered array of n elements. Note that one could solve the problem by first sorting the array and subsequently returning the k^{th} element in the array as the solution. However, such sorting is unnecessary and costly. Use the partitioning strategy from Question(2) to develop a solution for this problem. We provide some hints to solving this problem:

   - Set l and u to be the bounds of the array
   - while $l < u$ do:
     - choose some random x about which to partition the array
     - partition the array in to two partitions A and B marked by indices i and j
     - Choose a partition to perform further search by using a test: if $j < k$ then $l := i$, if $i > k$ then $u := j$

4. Given a array of n distinct integers, find the length of the longest monotone increasing subsequence. Consider an array: $|1|2|9|4|7|3|11|8|14|6|$. In this example the longest increasing subsequence is of $1, 2, 4, 7, 11, 14$ and is of length 6. Note that in for a monotone increasing subsequence, the numbers are not required to be adjacent in the original array. A general algorithmic description is as follows:

   - Establish $a[0, ..., n-1]$ of n elements
   - set the initial condition of the longest subsequence terminating in the first position of the array
   - for the remaining $n-1$ positions in the array do:
     - if current element is less than the maximum in the longest previous set then locate the position and value of maximum among the predecessors and update the position and length of maximum, if required. Else, update the length, position of maximum and the maximum length so far.
   - return the length of longest monotone increasing sequence.

5. Give an algorithm to perform long division of two numbers which are represented as arrays. If possible, provide methods other than long division for computing mod and div