ESO207: Data Structures and Algorithms (Practice Problems – I)

Question 1: Finding k-majority Element

Given an array A with n elements and a number $1 \leq k \leq n$, an element of A is said to be k-majority of A if it appears more than n/k times in A. Design an $O(nk)$ time and $O(k)$ space algorithm which computes a k-majority element, if one exists, in an array A.

Question 2: Sub-array with sum closest to input

Given an array A storing n numbers and a number x, determine the subarray of A whose sum is closest to x in $O(n \log n)$ time.

Question 3: Row with most 0s

Given a $n \times n$ matrix of 0s and 1s such that in each row no 0 comes before a 1, design an $O(n)$ time algorithm to find the row with the most 0s.

Question 4: Finding maximum and minimum efficiently

Design an algorithm that takes an array A having n distinct numbers and outputs the maximum and minimum element of the array using at most $\lceil 3n/2 \rceil$ comparisons.

Question 5: Closest pair of points on a line

Given the x-coordinates of n points on the real line, design an $O(n \log n)$ time algorithm to find the closest pair of points.

Question 6: Finding the peak entry in an array

An array A having n distinct elements is said to be *unimodular* if for some index p between 0 and $n-1$, the values in the array entries increase up to position p in A (also known as the *peak*) and then decrease the remainder of the way until position $n-1$.

Design an $O(\log n)$ time algorithm that given a unimodular array A finds the the peak position p of A.

Question 7: Counting number of significant inversions

A pair (i, j) is called a *significant inversion* if $i < j$ and $A[i] > 2A[j]$. Design an $O(n \log n)$ time algorithm to count the number of significant inversions in an array.

Question 8: Check if red-black tree

Given a binary tree T on n nodes where each node is colored red or black, design an $O(n)$ time algorithm to determine if T is a red-black tree.

Question 9: Missing value in BST

Given a BST T having a node with a missing value, find the exact range from which you can select a value and get back a valid BST in $O(n)$ time.

Question 10: Problems on BSTs

Design an $O(h)$ time algorithm for the following problems on BSTs having n nodes and height h:

(a) Given a BST T and an index $1 \leq i \leq n$, output the i-th element in T.

(b) Compute the successor of an element in a BST.

(c) Compute the predecessor of an element in a BST.