Data Structures and Algorithms

Practice-sheet

Divide and Conquer paradigm.

1. There is an array A storing n distinct numbers. Design an algorithm to find the maximum and the minimum of an array in $< 2n$ number of comparisions. Your algorithm may change array A if required.

2. In class, we discussed an $O(n)$ time algorithm for finding a contiguous subarray in a 1-d array which has the maximum sum. Though we were able to solve the problem efficiently, there exists a divide and conquer approach for the same problem which takes $O(n \log n)$ time. Design the divide and conquer algorithm, stating clearly the divide and conquer steps.

3. Suppose you are given an array A with n entries, with each entry holding a distinct number. You are told that the sequence of values $A[0], A[2], ..., A[n-1]$ is *unimodular*: For some index p between 0 and $n-1$, the values in the array entries strictly increase up to position p in A and then strictly decrease the remainder of the way until position $n-1$. (So if you have to draw a plot with the array position j on the x-axis and the value of the entry $A[j]$ on the y-axis, the plotted points would rise until x-value p, where they would achieve their maximum, and then fall from there on.)

   You would like to find the peak entry p without having to read the entire array - in fact, by reading as few entries of A as possible. Show how to find the entry p by reading at most $O(\log n)$ entries of A.

4. Recall the problem of finding the number of inversions in an array. This problem was discussed in the class. Let us call a pair (i, j) 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.

5. Design an $O(n \log n)$ time algorithm for finding 2-majority element which is based on divide and conquer paradigm in way similar to Merge Sort or Counting Inversion.

6. There are n points on real line. A is an array that stores the distance of each of these points from the origin. We have to design an algorithm to find the closest pair of points.

   - Design an $O(n \log n)$ time algorithm based on divide and conquer.
   - Design an $O(n \log n)$ time algorithm based on some other common technique/procedure.

   In the next course CS345A, we shall solve this problem when points are in a plane.

7. There are 2 sorted arrays A and B, each storing n distinct numbers. Write an algorithm to find the median of $A \cup B$. The time complexity of your algorithm should be $O(\log n)$.