International Institute of Information Technology, Bangalore
CS 501 Data Structures and Algorithms.
Practice Problems 1: August , 2014

1. Solve the following recursions (in terms of Θ). $T(0) = T(1) = \Theta(1)$ in all of the following.

   (a) $T(n) = 2T(n/2) + n$.
   (b) $T(n) = T(n/2) + n$.
   (c) $T(n) = 3T(n/2) + n$.
   (d) $T(n) = 2T(n/2) + n^2$.
   (e) $T(n) = T(n/2) + n^2$.
   (f) $T(n) = 3T(n/2) + n^2$.
   (g) $T(n) = 2T(n/2) + 1$.
   (h) $T(n) = T(n/2) + 1$.
   (i) $T(n) = 3T(n/2) + 1$.
   (j) $T(n) = 2T(n/2) + n \log n$.
   (k) $T(n) = T(n/2) + T(n/4) + 1$.
   (l) $T(n) = T(n/2) + T(n/4) + n$.
   (m) $T(n) = T(n/2) + 2T(n/4) + 1$.

2. **Maximum product Subsequence problem**

   Given an array $a_1, a_2, \ldots a_n$ of integers (both positive and negative), design a linear time algorithm to find the contiguous subsequence with the maximum product.

3. You are given an array $a_1, a_2, \ldots a_n$ of integers (both positive and negative), and a interger $l, 1 \leq n$. The length of a subsequence is defined as the number of integers in the subsequence.

   (a) Design a linear time algorithm to find the a maximum sum subsequence of lenght exactly l.
   (b) Design a linear time algorithm to find the a maximum sum subsequence of lenght at least l.

(c) Design a linear time algorithm to find the a maximum sum subsequence of lenght at most l.

4. You are given an array $a_1, a_2, \ldots a_n$ of integers (both positive and negative), and a interger $l, 1 \leq n$. The density of a subsequence is defined as the sum of numbers in the subsequence devided by the length of the subsequence.

   (a) Design a linear time algorithm to find the a maximum density subsequence of lenght exactly l.

   (b) Design a linear time algorithm to find the a maximum density subsequence of lenght at most l.

   (c) Design an $O(n^2)$ algorithm to find the a maximum density subsequence of lenght at least l.

   (d) Design an $O(nl)$ algorithm to find the a maximum density subsequence of lenght at least l. **Hard**

   (e) Design a linear time algorithm to find the a maximum density subsequence of lenght at least l. **Very Hard**

5. Given two arrays A and B, containing m and n integers respectively , design an efficient algorithm to determine how many integers are in common between the two arrays.

6. The $addBlock()$ operation is used to add m integers to to an existing sorted n integers, where $m << n$ (m is very small compared to n). One of your senior student gave me the following algorithm for this problem: the algorithm simply creates an array of length $n+m$, copies over the old n values into the new array, copies over the m values to the end of the array, and finally insertion sort is used (from the nth location onwards) to bring everything into order.

   (a) What is the complexity of the above algorithm.

   (b) Design an efficient algorithm for this problem.

7. Let $a_1, a_2, \ldots a_n$ be a sequence of distinct numbers. The pair (i, j) is called a inversion, if $i < j$ and $a_i > a_j$. Give an $O(n \log n)$ to determine the number of inversions in the given array.

8. Given a sorted array of distinct integers $A[0] < A[1] < \ldots A[n-1]$, design an $O(\log n)$ algorithm for the following

   (a) Decide whether there is an index i such the $A[i] = i$.

   (b) Given x and y, find the number of integers in the given array which are strictly greater than x, but strictly smaller than y.

9. Give a $O(n \log k)$ time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists.

10. Let A, B and C be three sequence of n integers each. Design an $O(n^2)$ algorithm to determine if there are three integers $a \in A, b \in B, c \in C$ such that $c = a + b$.

11. Suppose you are given two sorted arrays of integers $A[1..m]$ and $B[1..n]$ and an integer k. Describe an efficient algorithm to find the kth smallest element in the union of A and B.

    For example, given the input $A[1..10] = [0, 1, 3, 6, 11, 13, 15, 22, 32, 45]$, $B[1..5] = [2, 5, 8, 17, 29]$, $k = 9$, your algorithm should return 13. You can assume that the arrays contain no duplicates.

12. An element in a given sequence $a_1, a_2, \ldots a_n$ is said to be a majority element, if it repeats at least $n/3 + 1$ times. Design a linear time algorithm to decided if a given sequence has a majority element.