Design and Analysis of Algorithms (CS345A)

Practice-sheet : **Divide and Conquer**

- This practice sheet has all the problems from the *Divide and Conquer Paradigm*. We shall devote 5 lectures to cover this paradigm.
- Make sincere attempts to solve the problems. Do not postpone the problems to the last day before Quiz or mid-semester exam.
- Feel free to contact the instructor if you face any difficulty in these problems.

Miscellaneous

1. **Counting Double-Inversions**
   You are given an array A storing n numbers. A pair (i, j) with $0 \leq i < j \leq n-1$ is said to be a double-inversion if $A[i] > 2A[j]$. Design and analyze an $O(n \log n)$ time algorithm based on divide and conquer paradigm to compute the total number of double-inversions in A.

2. **Local minima in a complete binary tree**

   Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labeled with a real number x_v. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a local minimum if the label of x_v is less than the label x_w for all nodes w that are joined to v by an edge.

   You are given such a complete binary tree T, but the labeling is only specified in the following implicit way: For each node v,you can determine the value x_v by probing the node v. Show how to find a local minimum of T using only $O(\log n)$ probes to the nodes of T.

Geometric Problems

1. **Closest Pair Distance**
   The following problems are based on the divide and conquer algorithm for computing closest pair distance discussed in the class.

   (a) Write a neat and complete pseudocode of the algorithm we discussed in the lecture (without referring to the slides).

   (b) The algorithm we discussed in the lecture assumed that the median line passes through only one point. What if the median line passes through multiple points ? In particular, how will you ensure that the left half as well as right half has $\lceil n/2 \rceil$ points given that the median line passes through multiple points ?

   (c) Is the following statement about the algorithm discussed in the class true ?

   A point belongs to left strip in only a constant number of recursive calls.
   Prove or give a counterexample.

2. **Non-dominated points**

   In the lectures, we discuss a divide and conquer based algorithm to compute non-dominated points. However, there is an alternate and equally simple algorithm for this problem that runs in $O(n \log n)$ time. Provide complete description and analysis of this algorithm.

   Hint: The algorithm makes use of sorting as the first step.

3. **Least perimeter triangle***

   Let P be a set of n points in a plane. Design an $O(n \log n)$ algorithm to find the least perimeter triangle out of all possible triangles defined by P. Assume, without loss of generality, that there are no three collinear points in P.

   Hint: If you fully internalized the algorithm for closest pair problem, then you just have to pursue the same direction and make minor changes.

4. **Convex Hull**

   The following 2 problems should be attempted after we have discussed the algorithm for Convex Hull in the class.

   (a) In a lecture, we discussed an $O(n \log^2 n)$ time algorithm to compute the convex hull of a given set of n points in a plane. If we can improve the time complexity of the Conquer step of this algorithm to linear, this will result in an $O(n \log n)$ time algorithm for convex hull. You have to modify the current Conquer step so that it takes at most linear time. You must provide a complete analysis of the modified Conquer step as well.

   (b) **An application in computer graphics**

   There is a set S of n lines: L_1, L_2,..., L_n. For any $1 \leq i \leq n$, Line L_i is described by two parameters m_i and c_i such that the equation of line L_i is $y = m_i x + c_i$. Each m_i is a positive number. As input, you are given $\{(m_i, c_i) | 1 \leq i \leq n\}$. You may assume that no three lines in S pass through the same point. A line from S is said to be visible from above if there is some real number a_i such that the point through which L_i passes at $x = a_i$ is the highest (has maximum y-coordinate) among all the points through which other lines of set S pass at $x = a_i$. Design an $O(n \log n)$ time algorithm to compute all the visible lines from the set S. *Hint:* This problem will test your ability to divide the problem. What seems to be the most natural parameter to be used for dividing the n lines ?

1 Application of Polynomial Multiplication

The following problems should be attempted after we have discussed $O(n \log n)$ time algorithm to multiply 2 polynomials.

1. **Finding Polynomial given all its zero's**

   Given a list of values $z_0, \cdots, z_{n-1}$ (possibly with repetitions), show how to find the coefficients of the polynomial $P(x)$ of degree less than n that has zeros only at $z_0 \ldots, z_{n-1}$ (possibly with repetitions). Your procedure should run in time $O(n \log^2 n)$. *Hint:* The polynomial $P(x)$ has a zero at z_j if and only if $P(x)$ is a multiple of $(x - z_j)$.

2. **Toeplitz Matrix**

A Toeplitz matrix is an $n \times n$ matrix $A = (a_{i,j})$ such that $a_{i,j} = a_{i-1,j-1}$ for $i = 2, 3, \ldots, n$ and $j = 2, 3, \ldots, n$.

(a) Is the sum of two Toeplitz matrix necessarily Toeplitz ? What about the product ?

Yes, it is.

(b) Describe how to represent a Toeplitz matrix so that two $n \times n$ matrices can be added in $O(n)$ time.

(c) Give an $O(n \log n)$ time algorithm for multiplying an $n \times n$ Toeplitz matrix by a vector of length n. Use your representation from part (b).

Hint: Using part (b), express this problem as a multiplication of two *suitably defined* polynomials.

3. **Cartesian sum**

Consider two sets A and B, each having n integers in the range from 0 to $10n$. We wish to compute the **Cartesian** sum of A and B, defined by

$$C = \{x + y \ : \ x \in A \ \text{ and } \ y \in B\}.$$

Note that the integers in C are in the range from 0 to $20n$. We want to find the elements of C and the number of times each element of C is realized as a sum of elements in A and B. Design an $O(n \log n)$ time algorithm to achieve this objective.

Hint: Express this problem as a multiplication of two *suitably defined* polynomials.

4. **A computational physics problem**

You have been working with some physicists who need to study, as part of their experimental design, the interactions among large numbers of very small charged particles. Basically, their setup works as follows. They have an inert lattice structure, and they use this for placing charged particles at regular spacing along a straight line. Thus we can model their structure as consisting of the points $\{1, 2, 3, ...n\}$ on the real line; and at each of these points j, they have a particle with charge q_j. (Each charge can be either positive or negative.)

They want to study the total force on each particle, by measuring it and then comparing it to a computational prediction. This computational part is where they need your help. The total net force on particle j, by Coulomb's Law, is equal to

$$F_j = \sum_{i<j} \frac{Cq_iq_j}{(j-i)^2} - \sum_{i>j} \frac{Cq_iq_j}{(j-i)^2}$$

They have written the following simple program to compute F_j for all j.

```
For j = 1,2,...,n
    Initialize F_j to 0
    For i = 1, 2, ..., n
        If i < j then
            Add C q_i q_j / (j−i)^2 to F_j
        Else  if i > j  then
            Add −C q_i q_j / (j−i)^2 to F_j
        Endif
    Endfor
    Output F_j
Endfor
```

It is not hard to observe that the running time of the above algorithm is $\Theta(n^2)$. The trouble is, for the large values of n they are working with, the program takes several minuted to run. On the other hand, their experimental setup is optimized so that they can throw down n particles, perform the measurements, and be ready to handle n more particles within a few seconds. So they would really like it if there were a way to compute all the forces F_j much more quickly, so as to keep up with the rate of the experiment. Help them out by designing an algorithm which is much faster than $O(n^2)$. *Hint:* Express this problem as a multiplication of a few *suitably defined* polynomials.