COL 351 : Analysis and Design of Algorithms
Semester I, 2022-23, CSE, IIT Delhi

Assignment - 3 (due on 25th October, 11:00 PM)

Important Guidelines:

- Each assignment must be done in a group of size at most two.
- Handwritten submissions will not be accepted. Solutions must be typed-up (in Latex, Microsoft Word, etc.), and submitted in pdf format. Each solution must start on a new page.
- **Your answer to each question must be formal and have a proper correctness proof.** No marks will be granted for vague answers with intuition or for algorithms without proof. You must be very rigorous in providing mathematical detail in support of your arguments.
- Cheating of any form will lead to strict penalty.

1 Particle Interaction

Some physicists are working on interactions among large numbers of very small charged particles. Basically, their set-up 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

$$F_j = \sum_{i<j} \frac{C\ q_i\ q_j}{(j-i)^2} - \sum_{i>j} \frac{C\ q_i\ q_j}{(j-i)^2}\ .$$

They have written a simple $O(n^2)$ time algorithm to compute F_j, for all j. See Algorithm 1.

The trouble is, for the large values of n they're working with, the program takes several minutes to run. On the other hand, their experimental set-up 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.

```
for j = 1, 2, ..., n do
    Initialize F_j to 0;
    for i = 1, 2, ..., n do
        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;
        end
    end
    Output F_j;
end
```

Algorithm 1: Compute F_j

So they had 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. Your task is to design an algorithm that computes all the forces F_j in $O(n \log n)$ time. [15 marks]

2 Non dominated Points

Let $P = \{p_i = (x_i, y_i) \mid 1 \leqslant i \leqslant n\}$ be a set of n points in x-y plane with $x_i, y_i > 0$, for $i \in [1, n]$. Assume that there are no two points with same x-coordinate or y-coordinate. We say that a point p_i dominates another point p_j if $x_i > x_j$ and $y_i > y_j$. A point is said to be non-dominated if there is no point in P which dominates it. See example of non-dominated points in Figure 1.

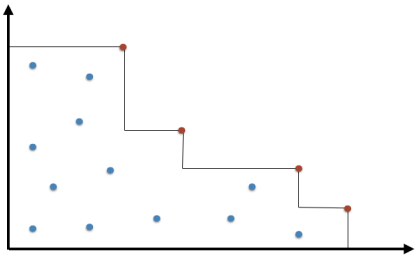

Figure 1: Depiction of a set comprising of 16 points and non-dominated points in it (marked in red color).

Design a divide and conquer strategy to compute the set of ALL non non-dominated points in set P in $O(n \log n)$ time. [23 marks]

3 Majority

An array A is said to have a majority element if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as voting answers, say.) However you can answer questions of the form: "is $A[i] = A[j]$?" in constant time.

(a) Show how to solve this problem in $O(n \log n)$ time [10 marks].
(Hint: Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A?)

(b) Next present a linear-time algorithm using the following approach [12 marks].

(i) Pair up the elements of A to get $n/2$ pairs.

(ii) Look at each pair: if the two elements are different, discard both; if they are the same, keep one of them. After this procedure there are at most $n/2$ elements left.

(iii) Prove that the resultant set has a majority element if and only if A does.