CS345 : Algorithms II
Semester II, 2016-17, CSE, IIT Kanpur

Assignment I

Deadline : 6:00 PM, 12 August 2016.

Important guidelines

- It is only through the assignments that one learns the most about the algorithms and data structures. You are advised to refrain from searching for a solution on the net or from a notebook or from other fellow students. Before cheating the instructor, you are cheating yourself. The onus of learning from a course lies first on you and then on the quality of teaching of the instructor. So act wisely while working on this assignment.
- This assignment is to be done in groups of 2 students. You have to form groups on your own. You are strongly advised not to work alone.
- There are two versions of this assignment: Difficult and Easy. The difficult one carries 60 marks and the easy one carries 45 marks. Attempt only one of them.
- Each group has to submit only one hard copy. The venue will be announced in the class or through email. You must upload its softcopy (pdf file) on the website. NO hand written assignment will be acceptable.

Difficult version

Maximum marks = 60

1. Non-dominated points

Recall the problem of non-dominated problem discussed in the class. Here is a brief summary. There is a set $P = \{p_i | 1 \leq i \leq n\}$ of n points in x-y plane. Let (x_i, y_i) denote the coordinates of point p_i. Assume that there are no pairs of points with the same x-coordinates or y-coordinates. We say that a point p_i *dominates* 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. The set of non-dominated points form a stair case like structure as shown in Figure 1.

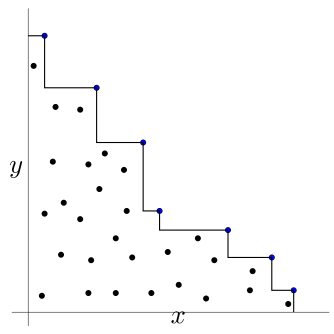

Figure 1: The non-dominated points form a staircase like structure

1. **Output sensitive algorithm** (*marks=20*)
   We discussed two algorithms in the class for computing non-dominated points - one trivial algorithm taking $O(nh)$ time, and another based on divide and conquer strategy taking $O(n \log n)$ time. You may use the insight from $O(nh)$ time algorithm to just *slightly* modify the second algorithm so that its running time is improved to $O(n \log h)$. You must provide the complete details of the analysis of the algorithm.

   Remark: Note that $O(n \log h)$ is superior to $O(n \log n)$ in those cases where the number of non-dominated points are very few. In fact, it can be shown that if n points are selected randomly uniformly from a unit square, then the expected(average) number of non-dominated points is just $O(\log n)$.

2. **Extension to 3 dimensions** (*marks=10,10*)
   You can extend the notion of non-dominated points to 3 dimensions as well. Spend some time to realize that it is not so simple to apply divide and conquer strategy to compute the non-dominated points in 3 dimensions. Some of you may be tempted to solve this problem by reducing an instance of this problem to three instances of 2-dimensional problem (by projecting the points on x-y, y-z, x-z plane). But that will be wrong (think over it to convince yourself). Interestingly, there is a very simple elegant algorithm using a simple data structure that computes non-dominated points in 3 dimensions. The purpose of this exercise is to make you realize this fact.

   (a) Design an algorithm that receives n points in x-y plane one by one and maintains the non-dominated points in an online fashion. Upon insertion of ith point, the algorithm should take $O(\log i)$ time to update the set of non-dominated points.
   Note: It is perfectly fine if your algorithm only guarantees a bound of $O(i \log i)$ on the total time for insertion of i points. It is not necessary that your algorithm achieves $O(\log i)$ bound on the processing carried out during insertion of ith point.

   (b) Design an $O(n \log n)$ time algorithm to compute non-dominated points of a set of n points in 3 dimensions. You must use part (a) above carefully.

2. A computational problem of an experimental physicist

(*marks = 20*)
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 $\frac{Cq_iq_j}{(j-i)^2}$ to $F_j$
        Else if $i > j$ then
            Add $-\frac{Cq_iq_j}{(j-i)^2}$ 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 $O(n^2)$.

Easy version

Maximum marks = 45

1. Non-diminated points in on-line fashion

(15 marks)
Design an algorithm that receives n points in x-y plane one by one and maintains the non-dominated points in an online fashion. Upon insertion of ith point, the algorithm should take $O(\log i)$ time to update the set of non-dominated points.
Note: It is perfectly fine if your algorithm only guarantees a bound of $O(i \log i)$ on the total time for insertion of i points. It is not necessary that your algorithm achieves $O(\log i)$ bound on the processing carried out during insertion of ith point.
Hint: make use of a simple data structure

2. Application of the algorithm for multiplication of polynomials

(15 marks)
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.

3. Augmented binary search tree

(15 marks)
Design a data structure that can maintain a set S of positive numbers that support the following operations in $O(\log n)$ time.

- Insert(S, x): Insert x into S
- Delete(S, x): Delete x from S
- Search(S, x): Search for number x in S
- Min-Gap(S, x): return the minimum difference between any pair of numbers in S. For example, if the set is $S = \{22, 1, 9, 15, 5, 18\}$, then the answer should be 3.