Due: 1 pm on Friday, March 8

Assignment 2

NOTE:

- The textbook exercises we select are, in most cases, much fewer than that for the regular track class, most of which are unable to determine the students' capabilities in the honor track. Instead, we manage to provide high-quality supplementary exercises (which also differ from that for the regular track) to let you have fun. However, we will also display the exercises for the regular track students only for your reference, at the end of each assignment.
- Unless explicitly stated in the question, you can omit the full proof for the correctness and time cost of your proposed algorithms **in this and all the incoming assignments**.
- Please solve all the problems in this assignment with divide-and-conquer methods. Using advanced data structures is NOT allowed.

1 Textbook Exercises

2.7, 2.27

2 Forces Between Particles

You guys have learned about how to compute the electrostatic forces between two charged particles in high school physics. Consider a case where we have particles at points $\{1, 2, \ldots, n\}$ on the real axis, and the particle at point j has charge q_j (the charge can be either positive or negative). By Coulomb's law, the total force on each particle j is:

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

where C is a given constant (the Coulomb constant). Design an algorithm that computes all the forces $F_j, j = 1, 2, \ldots, n$ in $O(n \log n)$ time.

3 Median of Medians

The Quickselect(A, k) algorithm for finding the k-th smallest element in an unsorted array A picks an arbitrary pivot, then partitions the array into three pieces: the elements less than the pivot, the elements equal to the pivot, and the elements that are greater than the pivot. It is then recursively called on the piece of the array that still contains the k-th smallest element.

(a) Consider the array A = [1,2,...,n] shuffled into some arbitrary order. What is the worst-case runtime of Quickselect(A, n/2) in terms of n? Construct a sequence of bad pivot choices that achieves this worst-case runtime.

(b) The above worst case has a chance of occurring even with randomly-chosen pivots, so the worst-case time for a random-pivot Quickselect is O(n^2).

Lets define a new algorithm Better-Quickselect that deterministically picks a better pivot. This pivot-selection strategy is called Median of Medians, so that the worst-case runtime of Better-Quickselect(A, k) is O(n).

Median of Medians

(a) Group the array into $\lfloor \frac{n}{5} \rfloor$ groups of 5 elements each (ignore any leftover elements)

(b) Find the median of each group of 5 elements (as each group has a constant 5 elements, finding each individual median is O(1))

(c) Create a new array with only the $\lfloor \frac{n}{5} \rfloor$ medians, and find the true median of this array using Better-Quickselect.

(d) Return this median as the chosen pivot

Let p be the chosen pivot. Show that for least $3n/10$ elements x we have that $p > x$, and that for at least $3n/10$ elements we have that $p \leq x$.

(c) Show that the worst-case runtime of Better-Quickselect(A, k) using the Median of Medians strategy is O(n).

Hint: Using the Master theorem will likely not work here. Find a recurrence relation for T(n), and try to use induction to show that $T(n) \leq cn$ for some $c > 0$.

4 Draw the Skyline!

You are given n non-vertical lines in the plane, labeled $L_1, \ldots, L_n$, with the i^{th} line specified by the equation $y = a_i x + b_i$. We assume that no three of the lines all meet at a single point. We say line L_i is *uppermost* at a given x-coordinate x_0 if its y-coordinate at x_0 is greater than the y-coordinates of all the other lines at x_0, i.e., $a_i x_0 + b_i > a_j x_0 + b_j$ for all $j \neq i$. We say line L_i is *visible* if there exists some x-coordinate at which it is uppermost. Give an algorithm that takes n lines as input, and in $O(n \log n)$ time returns all the visible ones.

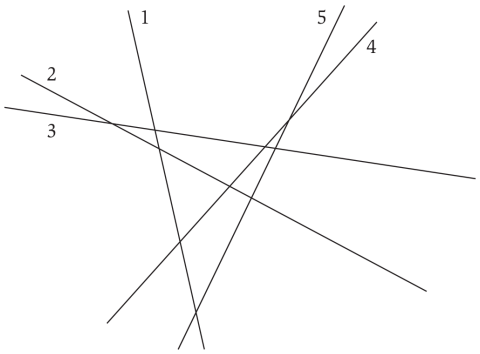

Figure 1: An example of finding all the visible lines. All the lines except for line 2 are visible.

5 Local Minima of A Chessboard

Given a $n \times n$ matrix M, where each element $M[i][j], 1 \leqslant i, j \leqslant n$ is labeled with a real number $m_{i,j}$; you may assume that all these values are distinct. For each element $M[i][j]$, you can determine the value $m_{i,j}$ by *probing* it. We define that $M[i][j]$ is a *local minimum* iff. its value is smaller than the values of all the adjacent (left, right, up, down) elements (except for those falling out of the range of the matrix). Show how to find a local minimum of M using only $O(n)$ probes to the elements of M.