CSOR W4231 (sec. 001, 002, H02): Analysis of Algorithms Sep 17

Homework 2: Due on Oct 3 by 12:01am

Instructors: *Alex Andoni, Cliff Stein*

Instructions. Please follow the homework policy and submission instructions in the Course Information handout. A few highlights to remember:

- follow the collaboration and academic honesty policies;
- write your name, uni, and collaborators on the top page;
- submission is via GradeScope (you are encouraged to use LaTeX, using the provided template);
- if you don't know how to solve a particular part of a problem, just write "*empty*", for which you will get 20% of the points of that part (in contrast, note that non-sensical text may get 0%).

Note that, for each bullet item of a problem, you can use the previous bullets as "black box", even if you didn't manage to solve them. Similarly, you can use anything from the lectures as black-box.

1 Problem 1 (20pts)

Consider elements $x_1, x_2, \ldots, x_n$ with positive weights $w_1, w_2, \ldots, w_n$ such that $\sum_{i=1}^n w_i = 1$. The *weighted (lower) median* is an element x_k satisfying

$$\sum_{x_i < x_k} w_i < \frac{1}{2}$$

and

$$\sum_{x_i > x_k} w_i \le \frac{1}{2} .$$

For example, if the elements are $0.1, 0.35, 0.05, 0.1, 0.15, 0.05, 0.2$ and each element equals its weight (that is, $w_i = x_i$ for $i = 1, 2, \ldots, 7$), then the median is 0.1, but the weighted median is 0.2.

a) Argue that the median of $x_1, x_2, \ldots, x_n$ is the weighted median of the x_i with weights $w_i = 1/n$ for $i = 1, 2, \ldots, n$.

b) Show how to compute the weighted median of n elements in $O(n \lg n)$ worst-case time using sorting.

c) Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm.

The *post-office location problem* is defined as follows. The input is n points $p_1, p_2, \ldots, p_n$ with associated weights $w_1, w_2, \ldots, w_n$. The goal is to find a point p (not necessarily one of the input points) that minimizes the sum $\sum_{i=1}^n w_i \, d(p, p_i)$, where $d(a, b)$ is the distance between points a and b.

d) Argue that the weighted median is a best solution for the 1-dimensional post-office location problem, in which points are simply real numbers and the distance between points a and b is $d(a, b) = |a - b|$.

e) Find the best solution for the 2-dimensional post-office location problem, in which the points are (x, y) coordinate pairs and the distance between points $a = (x_1, y_1)$ and $b = (x_2, y_2)$ is the *Manhattan distance* given by $d(a, b) = |x_1 - x_2| + |y_1 - y_2|$.

2 Problem 2 (20pts)

Consider the following game. There are n cards with the numbers $1, ..., n$ written on them, where each of the numbers $1, ..., n$ is written on exactly one of the cards. The cards are face down on the table, so we do not see their numbers. We turn over the cards one by one. Before turning each card, you guess the number written on the card; then you see the card, and you gain 1 point if you guess correctly and 0 if incorrectly. A random strategy is to guess in each step uniformly at random a number that has *not appeared so far* in the previous cards.

a) What is the probability that the guess in the i^{th} step is correct?

b) Show that the expected total number of points that you gain using the random strategy is $\Theta(\log n)$.

3 Problem 3 (20pts)

Suppose we are given n points in the Euclidean space, described by coordinates (x_i, y_i), where x_i, y_i are integers for $i = 1, 2, \ldots n$. The goal of this problem is to find a pair (x_i, y_i) and (x_j, y_j), with $i \neq j$, that are at the smallest distance from each other (i.e., $\|x - y\|_2 = ((x_i - x_j)^2 + (y_i - y_j)^2)^{1/2}$ is minimized). Below assume that $r > 0$ is a positive real.

a) Call a set of points P to be *r-spread*, for some $r > 0$, if any pair of points from P are at distance at least r.

Fix integers ℓ_x and ℓ_y. Consider a set of r-spread points P that lies inside the box $[\ell_x, \ell_x + 10r] \times [\ell_y, \ell_y + 10r]$ (i.e., such that for any $(x, y) \in P$, we have that $x \in [\ell_x, \ell_x + 10r]$ and $y \in [\ell_y, \ell_y + 10r]$). Prove an upper bound on the number of points in P, i.e., an upper bound on $|P|$ (note: it is enough to get an asymptotic bound only, no need to optimize constants).

b) Now consider a vertical band of width $2r$ with the following properties. Fix integer m. Let L be a set of points (x, y) such that $m - r \leq x \leq m$ (and any y). Let R be a set of points (x, y) such that $m < x \leq m + r$. Furthermore, suppose L and R are r-spread.

Design an algorithm that computes whether there exists a pair $(x, y) \in L$ and $(x', y') \in R$ that are at distance less than r; and, if such a pair exists, finds such a pairs of points at minimal distance. Your algorithm should have runtime $O(n \log n)$.

c) Based on the observations from above, design a Divide-And-Conquer algorithm for the main problem, i.e., given a set of points P, find the pair of points in P that minimizes the distance between the two points.

Your algorithm should have runtime $O(n \log n)$ (runtimes of $O(n \log^2 n)$ will get most of the credit as well).

4 Problem 4 (20pts)

Given a min-heap A, in the form of an array, and another key x (which may or may not be in A), we want to find all the elements of A that are smaller than x, and print them; the elements can be printed in any order (not necessarily in sorted order). For example, if $A = [4, 7, 5, 9, 8, 5]$ and $x = 8$, then the output should consist of $4, 7, 5, 5$ (printed in any order). Give an algorithm for this problem that runs in $O(k + 1)$ time, where k is the number of elements of A that are smaller than x. (*Hint:* Consider the tree representation of the heap, and show that if an element is in the output, then its parent must also be in the output.)

5 Problem 5 (20pts)

We are given a table A of size n by n such that each row and each column is sorted. In other words, for each $i, j \in \{1, \ldots n\}$, and any $i' > i, j' > j$, we have that $A[i, j] < A[i', j]$ and $A[i, j] < A[i, j']$. Furthermore, assume for simplicity that all the entries of A are distinct and are from the range $[1, n^4]$.

Design an algorithm for finding the median (element of rank $n^2/2$) of A in $O(n \log n)$ time (partial credit will be given for solutions achieving $O(n \log^2 n)$ time). In particular, you can consider the following two steps:

a) Design an algorithm that, given $x \in [1, n^4]$, outputs the number of entries in A that are smaller than or equal to x. Your algorithm should run in $O(n)$ time ($O(n \log n)$ time will get partial credit).

b) Now design an algorithm for the overall problem running in $O(n \log n)$ time.