Homework Exam 2 2018

Deadline: 16 January 2019, 15:00

This homework exam has 6 questions for a total of 100 points. You can earn an additional 10 points by a careful preparation of your hand-in: using a good layout, good spelling, good figures, no sloppy notation, no statements like "The algorithm runs in $n \log n$." (forgetting the $O(..)$ and forgetting to say that it concerns time), etc. Use lemmas, theorems, and figures where appropriate. Your final grade will be the number of points divided by 10 (with a maximum of a 10). Note that solving only a subset of the problems is sufficient to get a 10.

In many of the questions below we ask for algorithms that have subquadratic running time. This means that for n input the running time should be $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$. In many cases $O(n \log^c n)$, for some c are actually achievable!

Question 1

Let P be a set of n points in $\mathbb{R}^2$, and let $NN(p)$ denote the (Euclidean) *nearest-neighbor* of p. That is, $NN(p) = \operatorname{argmin}_{q \in P} \|pq\|$.

(a) *(10 points)*
Prove that the Voronoi regions of p and $NN(p)$ are adjacent in the Voronoi diagram $VD(P)$ of P.

(b) *(10 points)*
Describe an $O(n \log n)$ time algorithm to compute, for every point $p \in P$, its nearest neighbor $NN(p)$.

Question 2 *(10 points)*

Let P be a set of n points in $\mathbb{R}^2$, and let $w : P \to \mathbb{R}$ be a function that assigns a weight to each point in P. You can assume that all coordinates and weights are unique.

We would like to store P so that we can efficiently report the maximum weight point in a three-sided query range $Q = [x_\ell, x_u] \times [y_\ell, \infty)$. Describe a data structure that can answer such queries efficiently (i.e. as fast as possible). Analyze the query time and the space use of your data structure.

Question 3 *(15 points)*

Suppose that we have a trace $\mathcal{T}$ that records the n operations that an algorithm executes on a (dynamic) dictionary storing some dynamic set X of real numbers. That is, every entry in $\mathcal{T}$ is a triple consisting of a time stamp t, an operation, which is either INSERT or DELETE, and a value $v \in \mathbb{R}$ –the number that is inserted or deleted. Describe a data structure that allows us to efficiently "replay" the queries of the algorithm. That is; given an arbitrary query pair (t, q) consisting of a time $t \in \mathbb{R}$ and a value $q \in \mathbb{R}$ it allows us to efficiently report the value v that was the successor of q at time t. Analyze the space, preprocessing time, and query time of your solution.

The number of points awarded will depend on the space, preprocessing, and query time of your solution.

Question 4

Let $\mathcal{R}$ be a set of n axis-aligned rectangles in $\mathbb{R}^2$. You can assume that the coordinates of all sides are unique.

(a) *(10 points)*
The union of n axis-aligned squares has complexity $O(n)$. Describe an $O(n \log n)$ time algorithm to compute the union of $\mathcal{R}$ for the case that all rectangles in $\mathcal{R}$ are actually axis-parallel squares.

(b) *(10 points)*
Give a construction that shows that in general the union of n axis-aligned rectangles may have complexity $\Omega(n^2)$.

(c) *(10 points)*
Design an algorithm to compute the *area* of the union of $\mathcal{R}$. Your algorithm should run in subquadratic time. Argue that it is correct and analyze its running time.

Hint: Note that to get a subquadratic running time you cannot afford to construct the union explicitly. So think of a way of computing *just* the area.

Question 5 *(15 points)*
Let R be an axis-parallel rectangle in $\mathbb{R}^2$ and let P be a set of n points in R. Once again, all coordinates are unique. Design an algorithm that decides if there exists an empty axis parallel unit square Q that lies completely inside R, that is, Q has width one, does not contain any points from P in its interior, and is itself contained in R. Your algorithm may use at most $O(n)$ *working storage* (space) and should run in subquadratic time.

Any algorithm that runs in subquadratic time is sufficient to get full points.

Question 6 *(10 points)*
Let R be a set of n "red" points in $\mathbb{R}^2$, let B be a set of n "blue" points in $\mathbb{R}^2$, let $age : R \cup B \to \mathbb{R}$ be a function that assigns an age to every point, and let $\Delta > 0$ be some real number. You can again assume that all coordinates and ages are unique, and that there are no three colinear or four corcicular points. Design an algorithm to compute, for every red point $r \in R$, the closest (in terms of the Euclidean distance) blue point b among B for which $age(b) \in [age(r), age(r) + \Delta]$. Your algorithm should run in subquadratic time.

Hint: Try to identify subproblems/parts of the problem that you know how to solve efficiently!