CS 573: Algorithms, Fall 2012

Homework 4, due Monday, Nov. 12, 23:59:59, 2012

Version 0.22

Name:	
Net ID:	

Neatly print your name(s), NetID(s). If you are off campus, please submit the homework on moodle, otherwise submit the homework in SC 3306 (or sliding it under the door). Please solve each problem on a separate page.

"Is there anything in the Geneva Convention about the rules of war in peacetime?" Stanko wanted to know, crawling back toward the truck.
"Absolutely nothing," Caulec assured him. "The rules of war apply only in wartime. In peacetime, anything goes."
– Gasp, Romain Gary

Required Problems

1. Some preparations.
[30 Points]

(A) **[10 Points]** Show how to modify a treap, such that one can, in $O(\log n)$ time (with high probability), insert an element, delete an element, find an element, and report the rank of a given element in the treap. As a reminder, the ***rank*** of an element x in a set X of n values is the number of elements of X strictly smaller than x.

(B) **[10 Points]** Consider two permutations π, σ of $\{1, \ldots, n\}$. The ***edit*** distance between π and σ is the minimum number of times one has to flip consecutive pair of elements in π to get σ. For example, if $\pi = \langle 4, 3, 2, 1 \rangle$ and $\pi = \langle 1, 2, 3, 4 \rangle$ then the edit distance is 6. Indeed:

$$\left\langle \overbrace{4, 3}^{*}, 2, 1 \right\rangle \Longrightarrow \left\langle 3, \overbrace{4, 2}^{*}, 1 \right\rangle \Longrightarrow \left\langle 3, 2, \overbrace{4, 1}^{*} \right\rangle \Longrightarrow \left\langle \overbrace{3, 2}^{*}, 1, 4 \right\rangle$$
$$\Longrightarrow \left\langle 2, \overbrace{3, 1}^{*}, 4 \right\rangle \Longrightarrow \left\langle \overbrace{2, 1}^{*}, 3, 4 \right\rangle \Longrightarrow \langle 1, 2, 3, 4 \rangle .$$

Describe an $O(n \log n)$ time algorithm, using (A), that computes the edit distance between permutations. Hint: Consider the target permutation to always be $\langle 1, 2, \ldots, n \rangle$, then show to remove this assumption (hint: the numbers here are just labels).

(C) **[10 Points]** Let $L = \{\ell_1, \ldots, \ell_n\}$ be a set of n linear functions, where $\ell_i(x) = \alpha_i x + \beta_i$. You can safely assume that all the α_i and β_i are non-zero and distinct. One can interpret a set of such functions as a set of lines in the plane, where the ith line is $y = \alpha_i x + \beta_i$. A ***vertex*** is an intersection point of two lines. Assume that all the pairs of lines of L have distinct vertices, and furthermore, they all have x coordinate in the range $[0, 1]$. Let V be this set of $\binom{n}{2}$ vertices.

Given two numbers x_0 and x_1, provide an $O(n \log n)$ time algorithm (using (B), that outputs the number of vertices in V that have x coordinate in the interval $[x_0, x_1]$. To keep things simple, you can safely assume that no vertex in V has an x-coordinate that is equal to either x_0 or x_1. (Hint: Consider the order in which the lines of L intersect the vertical lines $x = x_0$ and $x = x_1$. What does a vertex corresponds to?)

2. Parametric search.

[40 Points]

Given a set of lines L as above, and (implicitly) its set of vertices V (we assume all the vertices have distinct x value), let the ***rank*** of a vertex $v \in V$ be the number of vertices in V that have smaller x coordinate than v. (Again, you can assume all the vertices in V have x coordinate in the range $[0, 1]$.) Given k, we are interested in computing the vertex of rank k in V. Let this unknown vertex be $v^* = (x^*, y^*)$.

(A) **[5 Points]** Describe an $O(n^2)$ time algorithm for computing v^*.

(B) **[10 Points]** Quadratic time is pathetic. But doing better requires quite a bit of cleverness. To this end, given a set X of $O(n)$ vertices in V, show how to partition X, in $O(n \log^2 n)$ time, into the set of all the vertices of X left of v^*, and all the vertices of X right of v^*. (If one of these vertices is v^* then bingo - we are done.) Of course, you are not given v^*, but you can (and must) use the algorithms from the first question to solve this part.

(C) **[5 Points]** Consider a pair of lines $\ell_i, \ell_j \in L$, and let v be their intersection. Consider the vertical line $\tau \equiv (x = x^*)$. Given that v^* is (say) to the left of v, show how to decide (in constant time) if ℓ_i is above ℓ_j along the line τ.

(D) **[10 Points]** We are going to do the most bizarre thing ever – get ready. We want to sort the lines of L in their order along the vertical line $\tau \equiv (x = x^*)$ (without knowing v^* no less!).

To this end, assume that you are given an explicit construction of sorting network for n numbers. Specifically, we are given the layers $G_1, \ldots, G_m$ ($m = O(\log^2 n)$), where G_k is the set of gates in the kth layer of the sorting network. Each gate is a pair of numbers (i, j) that corresponds to comparing the number on the ith wire to the number on the jth wire. It is not hard to compute the sets $G_1, \ldots, G_m$ from the construction shown in class, but lets just assume this is given.

We are going to sort the lines along τ using this sorting network. Show how to resolve all the comparisons of the gates of G_i, in $O(n \log^2 n)$ time, using (B) and (C).

(E) **[10 Points]** Describe how using the above, one can output v^* in $O(n \log^4 n)$ time.

3. Computing Polynomials Quickly

[30 Points]

In the following, assume that given two polynomials $p(x), q(x)$ of degree at most n, one can compute the polynomial remainder of $p(x) \bmod q(x)$ in $O(n \log n)$ time. The ***remainder*** of $r(x) = p(x) \bmod q(x)$ is the unique polynomial of degree smaller than this of $q(x)$, such that $p(x) = q(x) * d(x) + r(x)$, where $d(x)$ is a polynomial.

Let $p(x) = \sum_{i=0}^{n-1} a_i x^i$ be a given polynomial.

(A) **[8 Points]** Prove that $p(x) \bmod (x - z) = p(z)$, for all z.

(B) **[8 Points]** We want to evaluate $p(\cdot)$ on the points $x_0, x_1, \ldots, x_{n-1}$. Let

$$P_{ij}(x) = \prod_{k=i}^{j} (x - x_k)$$

and

$$Q_{ij}(x) = p(x) \bmod P_{ij}(x).$$

Observe that the degree of Q_{ij} is at most $j - i$.

Prove that, for all x, $Q_{kk}(x) = p(x_k)$ and $Q_{0,n-1}(x) = p(x)$.

(C) **[6 Points]** Prove that for $i \le k \le j$, we have

$$\forall x \quad Q_{ik}(x) = Q_{ij}(x) \bmod P_{ik}(x)$$

and

$$\forall x \quad Q_{kj}(x) = Q_{ij}(x) \bmod P_{kj}(x).$$

(D) **[8 Points]** Given an $O(n \log^2 n)$ time algorithm to evaluate $p(x_0), \ldots, p(x_{n-1})$. Here $x_0, \ldots, x_{n-1}$ are n given real numbers.