CS 6301.008 Homework 2

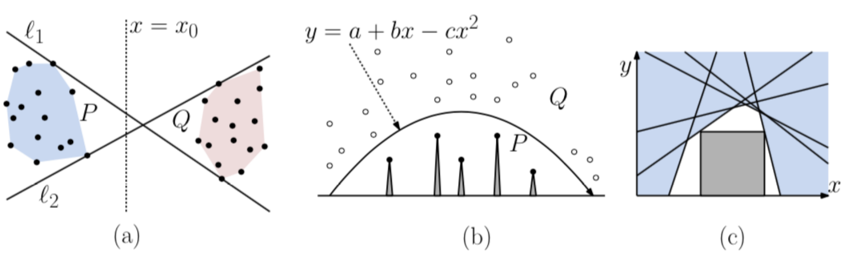
Due ~~Tuesday February 20th~~ Thursday February 22nd at 11:30am, on eLearning

February 6, 2018

Please answer all three questions. You may form groups of up to three students. Each group should write a single set of solutions with each group member's name and Net ID on the front page. Each group member should then submit a copy through eLearning.

1. **(From Mount)** Explain how to solve each of the following problems in linear (expected) time. Each can be modeled as a linear programming (LP) problem, perhaps with some additional pre- and/or post-processing. In each case, explain how the problem is converted into an LP instance and how the answer to the LP instance is used/interpreted to solve the stated problem.

   (a) You are given two point sets $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_m\}$ in the plane, and you are told they are separated by a vertical line $x = x_0$, with P to the left and Q to the right. Compute the line equations of the two "crossing tangents,", that is, the lines ℓ_1 and ℓ_2 that are both supporting lines for the convex hulls of each of P and Q such that P lies below ℓ_1 and above ℓ_2 and the reverse holds for Q. (Note that you are <u>not</u> given the hulls, just the point sets.) Your algorithm should run in $O(n+m)$ time.



(From Mount). Examples for each of the scenarios in Problem 1.

   (b) You have a cannon in $\mathbb{R}^2$. It has three controls labeled "a", "b", and "c". A projectile shot from this cannon travels along the parabolic arc $y = a + bx - cx^2$. You are asked to determine whether it is possible to adjust the controls so that the projectile travels above a set of n building tops, represented by a point set $P = \{p_1, \ldots, p_n\}$ and beneath a set of m floating balloons, represented by a point set $Q = \{q_1, \ldots, q_m\}$. Your algorithm should run in time $O(n+m)$. (Do not worry about the cannon's location. Just determine if there is <u>some</u> parabola along which the projectile can travel.)

   (c) You are given a set of n halfplanes $H = \{h_1, \ldots, h_n\}$, where h_i is given as a pair (a_i, b_i), and it consists of all the points of the plane that lie on or beneath the line $y = a_i x + b_i$. Compute the axis-parallel square of the largest side length lying in $h_1 \cap \cdots \cap h_n$ whose lower edge lies on the x-axis. In no such square exists, your algorithm should indicate this.

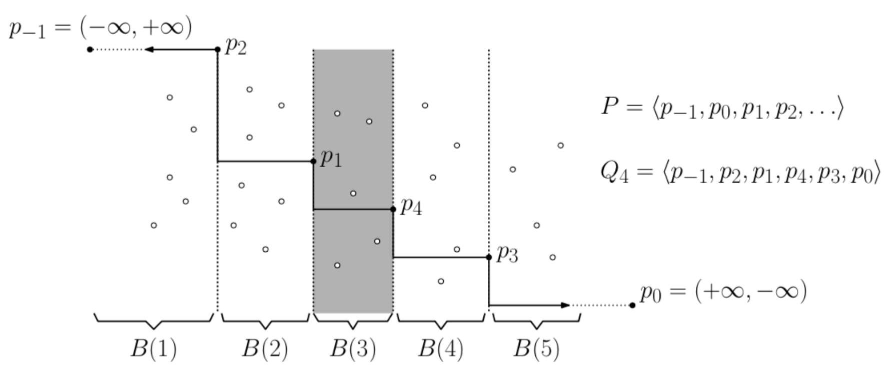
2. **(From Mount)** The objective of this problem is to get some practice working with backwards analysis. Consider the following randomized incremental algorithm for the Pareto-set problem. Recall that the input is a set of n points $P = \{p_1, \ldots, p_n\}$ in the plane, where $p_i = (x_i, y_i)$, and the objective is to compute the subset of points p_i such that there is no $p_j \in P, (j \neq i)$ such that $x_j \geq x_i$ and $y_j \geq y_i$. Our approach will be to add the points one-by-one in random order. We make the usual general-position assumption that there are no duplicate x- or y-coordinates.

The algorithm begins by randomly permuting the point set. Let $P = \langle p_1, \ldots, p_n \rangle$ denote the permuted sequence. Let's add two sentinel points $p_{-1} = (-\infty, +\infty)$ and $p_0 = (+\infty, -\infty)$. The initial Pareto set Q_0 consists of the pair $\langle p_{-1}, p_0 \rangle$. Observe that as we add points to the Pareto set, p_{-1} will always be the leftmost point, and p_0 will always be the rightmost. For $1 \leq i \leq n$, let Q_i denote the sequence of points of the Pareto set after inserting $\langle p_{-1}, \ldots, p_i \rangle$, sorted from left to right. The final output will be Q_n. Also, let $R_i = \langle p_{i+1}, \ldots, p_n \rangle$ denote the sequence of points that remain to be inserted.

Let us assume inductively that we have already inserted $i-1$ points, and we are about to insert p_i. In order to make it easy to determine where to insert the new point within the existing Pareto sequence, we will store all the remaining points of R_{i-1} in a collection of buckets, one for each point in the current Pareto set. In particular, let us suppose that we have already inserted $i-1$ points into the Pareto set. Let $Q_{i-1} = \langle p_{j_0}, p_{j_1}, \ldots, p_{j_m} \rangle$. (By our earlier observation, $j_0 = -1$ and $j_m = 0$.) For $1 \leq k \leq m$, define $B(k)$ to be the subset of remaining points that lie within the vertical strip between the $(k-1)$st and kth points of the current Pareto sequence, that is,

$$B(k) = \{p_\ell \in R_{i-1} : x_{j_{k-1}} < x_\ell < x_{j_k}\}.$$

(For example, in the figure below, the white points in the shaded region belong to bucket $B(3)$, because they lie between $x_{j_2} = x_1$ and $x_{j_3} = x_4$.)



(From Mount). The example scenario for Problem 2.

(a) Using the above structure, explain how to insert the ith point p_i and update the Pareto sequence Q_{i-1} to form Q_i. Also, explain how to update the associated buckets. (If points are removed from the Pareto sequence, then their associated bucket sets must be redistributed among the points that are currently in the Pareto sequence.) Don't worry about data structures, just tell me which sequences/sets change and what points enter or leave them.

(b) Analyze the running time of your algorithm. Your analysis should include two elements (i) the time needed to update the Pareto sequence, and (ii) the time needed to update the associated bucket sequences. Element (i) should be significantly simpler to analyze. You may assume changes to the Pareto sequence or bucket sets occur in constant time per point inserted or deleted from the sequence/set. It may also help to recall the n harmonic number $H_n = \sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n)$.

3. (a) Recall we can use fractional cascading to reduce the range reporting query time for a 2-dimensional range tree from $O(\log^2 n + k)$ to $O(\log n + k)$ where k is the number of points lying in the query range. Briefly discuss how to modify the technique to perform weighted range counting queries in $O(\log n)$ time. Your answer should describe the necessary changes to the range tree itself (if any) and how to compute the total weight of points lying within a query range and a particular auxiliary list.

   (b) Let S be a set of n axis-parallel rectangles in the plane. We want to be able to report all rectangles in S that are completely contained in a query rectangle $Q = [x_{lo}, x_{hi}] \times [y_{lo}, y_{hi}]$. Describe a data structure for this problem that uses $O(n \log^3 n)$ space and has $O(\log^3 n + k)$ query time, where k is the number of reported rectangles. [Hint: Transform the problem into some orthogonal range searching problem in a higher dimensional space. You may assume orthongal range trees can support both points and ranges that include $-\infty$ or $+\infty$ in some components (so 1D range $[-\infty, 5]$ would include all real numbers less than or equal to 5 and point $(2, +\infty)$ would lie higher than any bounded rectangular range in the plane).]

   (c) Let P be a set of n points in the plane. We want to be able to report all points in P that are completely contained in a query triangle. However, the triangle is guaranteed to have one horizontal edge, one vertical edge, and one edge of slope -1 or $+1$. Describe a data structure for this problem that uses $O(n \log^3 n)$ space and has $O(\log^3 n + k)$ query time, where k is the number of reported points.