15-451 Recitation 11

Computational Geometry

1 Setup and Sweep

You are given a set S of n points with integer coordinates in the plane. You are also given m axis-alligned rectangles, the ith one specified by a pair of points $((x_i, y_i), (x'_i, y'_i))$. (x_i, y_i) is the lower left corner and (x'_i, y'_i) is the upper right corner.

The goal is, to compute for each rectangle the number of points of S that are in it. Give an algorithm whose running time is $O((n+m)\log m)$. For simplicity, assume that all coordinates are bounded by m.

Solution: Maintain a segment tree on m leaves where initially all leaves are set to 0 and the function being computed is $+$. Sort points and edges of rectangles by x coordinate and put them in an array. Now, iterate through this sorted array. When you encounter a point, increment the leaf corresponding to the y-coordinate of the point by 1. When you encounter the left edge of a rectangle R with y-coordinates y_1 and y_2, use the segment tree to obtain the sum in range $[y_1, y_2]$ and store it as a variable T_R. When you encounter the right edge of a rectangle R, use the segment tree to obtain the sum in range $[y_1, y_2]$ and store it as a variable T'_R. The value $T'_R - T_R$ gives the number of points of S in R.

2 Circle with Most Points

Given a set of points $S = \{p_1, \ldots, p_n\}$, and a radius $r > 0$, the goal is to find a circle of radius r that contains the maximum number of points from S.

(a) Give an $O(n^3)$ algorithm for this problem.

Solution: You can find the closest pair in time $O(n)$. If their distance is more than $2r$, you know the answer is 1. Hence we can assume from now on that the optimal answer is at least 2.
Now, we can move the optimal circle so that it touches two of the points from S. So we can focus only on circles that touch two points in S. For any $p, q \in S$, there are two circles of radius r that touch both. And we can count how many points each one contains in $O(n)$ time. Iterating over all $\binom{n}{2}$ pairs p, q, this gives an $O(n^3)$ algorithm.

(b) Give an $O(n^2 \log n)$ algorithm. (Hint: sweep-*angle*.)

Solution: Fix point $p \in S$, and consider all radius-r circles that touch it. Their centers lie on the radius r ball centered at p. As you sweep the radius-r circle around, for each other point $q \in S$, you can figure out the first and last angle at which q lies within the circle. Say these angles are f_q, ℓ_q. Now take all the $2n-2$ angle values $\{f_q, \ell_q \mid q \in S, q \neq p\}$ and sort these in $O(n \log n)$ time. Now in linear more time we can figure out the angle for which the maximum number of points lie within the circle.
Doing this for each of the n points p will take $n \times O(n \log n) = O(n^2 \log n)$.

3 The Width of a Set of Points

You're given a set $S = \{p_1, \ldots, p_n\}$ of n points in the plane. A strip of width w is the region between two parallel lines, where the distance between the two lines is w. The goals is to find the strip of minimum width that contains all the points. Given an $O(n \log n)$ algorithm for this problem.

Here's a bit of useful background. The equation $Ax + By = C$ defines a line. Any line (including vertical or horizontal) can be represented this way, where at least one of A or B is non-zero. We can normalize it by dividing the whole equation by $\sqrt{A^2 + B^2}$.

So let's assume that the line has been normalized so that $A^2 + B^2 = 1$. The result of this is a normalized vector (A, B) that is perpendicular to the line $Ax + By = C$.

Consider the function of x and y given by $d(x, y) = Ax + By - C$. Now this function is 0 on the line. In fact, its value at any point (x, y) in the plane is just the (signed) distance between the point and the line.

So the problem of finding the minimum width of a strip containing the set S becomes that of finding a unit vector (A, B) such that the following quantity is minimized:

$$(\max_{(x,y)\in S} Ax + By) - (\min_{(x,y)\in S} Ax + By)$$

If we think of $Ax + By$ as the objective function, we want the difference between the maximum value of it over the set of points minus the minimum of it.

Solution: The first observation is that we only need to consider values of (A, B) which arise from the lines that define the sides of the convex hull. This is because any strip that touched precisely two points of the convex hull of S can be rotated to make a narrower strip that also contains S. The rotation continues to shrink the width of the strip until a boundary line of the convex hull coincides with the boundary of the strip.
This gives rise to the following algorithm: Compute the convex hull of S. So for each side of the convex hull find the other point in S that is farthest away from that side. Take the minimum of this quantity over all sides. This is our answer. But this seems to be an $O(n^2)$ algorithm.
The solution is to to use the "two pointer" approach. Let the sides of the convex hull be named $a_1, \ldots, a_k$, and the points of it be $q_1, \ldots, q_k$ both in clockwise order. Start with side a_1, and find the farthest point away from it, q_j. This takes $O(n)$. Now advance to side a_2. To find its farthest anti-neighbor we start from q_j and go to q_{j+1} etc, each time testing if its getting farther away or closer. When it gets farther away, we stop, because we know we've gone too far. This follows from the fact that we're working with a convex set.
Therefore this two-finger scan finds the furthest point from each side of the convex hull in $O(n)$ time. Combining this with the convex hull running time gives an $O(n \log n)$ algorithm for this problem.