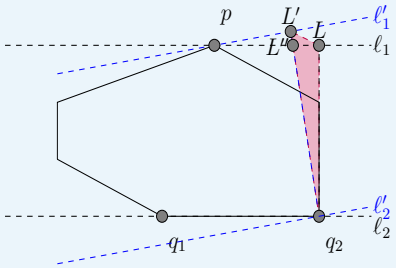
Student ID: 1155189917. (Feb 24, 2024)

Problem 1. Let P be a set of n points in $\mathbb{R}^2$. A slab is the region between two parallel lines (inclusive of the two lines). The *perpendicular* width of a slab is the (perpendicular) distance between its boundary lines. Suppose that parallel lines ℓ_1 and ℓ_2 define a width the smallest perpendicular width among all the slabs enclosing all the points of P. Prove: either ℓ_1 or ℓ_2 passes two points of P.

Proof. The perpendicular distance of two parallel lines $\ell_1 : y = ax - b_1$, $\ell_2 : y = ax - b_2$ $(b_1 < b_2)$ is $d = \frac{b_2 - b_1}{\sqrt{1+a^2}}$.

ℓ_1 and ℓ_2 must both at least pass one point of P. Otherwise, we can let $\ell_1' : y = ax - b_1', \ell_2 : y = ax - b_2'$ $(b_1' < b_2'$, also $b_1 < b_1'$ or $b_2' < b_2)$ such that these two lines both touch at least one point on P to decrease the perpendicular width between lines.



WLOG, consider the case where ℓ_1 and ℓ_2 touches one points only (pictorially on the graph depicted by ℓ_1' and ℓ_2'). Then, we can always rotate these lines simultaneously by the smallest angle ϕ so that at least one point touches the another point.

Suppose ℓ_2 is such a line touching q_1 and q_2, and ℓ_1 is touching p. Then, let $q_2L \perp \ell_1$ where L is a point on ℓ_1, and $q_2L' \perp \ell_1'$ where L' is a point on ℓ_1' and crosses ℓ_1 at L''.

Consider the triangle $LL''q_2$. Since q_2L is the height of the right triangle, it must be that $q_2L > q_2L''$ by triangle inequality. Thus, $q_2L < q_2L'' < q_2L'$, since ℓ_1 is pivoted at p which causes L' to be out of the segment $\overline{q_2L''}$. □

Problem 2. Let P be a set of n points in $\mathbb{R}^2$. Describe an algorithm to find a slab with the minimum perpendicular width that encloses all the points of P. Your algorithm should run in $O(n \log n)$ time.

Solution. Rotating Calipers. The convex hull of P with vertices in clockwise order $p_1, p_2, \cdots, p_n$ can be computed in $O(n \log n)$ time. Let ℓ_1 be a line passing an edge of $CH(P)$. Let ℓ_2 be a line passing this farthest point. Find in $O(n)$ time a point maximizing the area of the triangle formed by this point and the points passed by the line ℓ_1.

Since we have proven that either ℓ_1 or ℓ_2 touches at least two points on $CH(P)$. We can enumerate all possible ℓ_1s by traversing through the edge in clockwise order, while advancing ℓ_2. Let ℓ_1 pass through p_i, p_{i+1}, and ℓ_2 pass through p_j. We advance ℓ_2 while the area of the triangle $p_{j+1}p_ip_{i+1}$ is no smaller than that of $p_jp_ip_{i+1}$. Then, we just need to find the compare all the slab distance formed by the calipers. The rotation of lines can thus be done in $O(n)$ time.

The running time of this algorithm is $O(n \log n)$ in total.

Problem 3. Let L be a set of n non-vertical lines in $\mathbb{R}^2$ where no two lines are parallel. Explain how to compute in $O(n \log n)$ time an axis-parallel rectangle that contains all the $\binom{n}{2}$ intersect points of those lines.

Solution. Consider the left boundary of the rectangle $\ell : x = -M$. Then, imagine scanning a line from $x = -\infty$ to the right. All lines must be in the order of decreasing slope from top to bottom, where the line with the smallest slope has the largest intercept.

If there were to be an intersection between two lines when sweeping the line, it can only be introduced by two neighbouring lines. Therefore, we can sort all lines with respect to their slope in $O(n \log n)$ time. Then, we iterate through every two neighboring lines in $O(n)$ time and pick the intersection point with the smallest x-coordinate.

Hence, the leftmost vertex can be computed in $O(n \log n)$ time.

Apply the similar steps for right boundary, top boundary and bottom boundary of the rectangle. This can be completed in $O(n \log n)$ time as required.

Problem 4. Let P be a set of points in $\mathbb{R}^2$, and $k \leq n$ be an integer. Describe an algorithm to find a slab with the minimum perpendicular width that encloses precisely k points of P. Your algorithm should run in $O(n^2 \log n)$ time.
Hint: Think in the direction of Problem 1.

Solution. If $k = 1$, then the answer is zero.

We have proven that a slab with at least one of the lines ℓ passing two points is the optimal way of enclosing points. Therefore, we can consider the dual space of the set of points, which are mapped to a set of lines.

Now, sweep a line from $x = -\infty$ to $x = +\infty$. To reach the optimality, it suffices to consider only the points at intersection of these dual lines. Maintain an array in the order of lines from the top to the bottom of the sweeping line which passes through. Additionally, on every intersecting points in the dual space, supposing that it is the i-th, $(i+1)$-th line in the array which intersects, we calculate the distance between the current point and the line $A[i-k+2]$, and the distance the distance between the current point and the line $A[i+k-1]$, and multiply it by a factor of $1/\sqrt{1+x^2}$, where x is the current posiiton of the sweeping line. Minimizing the result of all calculated distances will result in the minimum width of the enclosing slab with k points.

This can be solved by the line intersection algorithm in $O(n^2 \log n)$ time, where n^2 comes from the number of intersecting points of the lines, and the $\log n$ factor maintains the position of these lines.

Problem 5. Let P be a set of n points in $\mathbb{R}^2$. Describe an algorithm to find the smallest-area triangle whose vertices are from P. Your algorithm should finish in $O(n^2 \log n)$ time.

Solution.

In the primal space, the area of a triangle pqr is $dist(p, q) \cdot$ distance to $\overline{pq}$ from r.

In the dual space, we need to find a point p lying on the intersecting lines and another line ℓ which minimizes the distance of p to ℓ. Therefore, we need an algorithm for any vertex $\overline{pq}^*$ of the dual arrangements of $\mathcal{A}(P^*)$, the closest line above/below ℓ^*.

Compute first the arrangements of $\mathcal{A}(P^*)$. Then, sweep a line from $x = -\infty$ to $x = +\infty$. Apply

the line intersection algorithm to discover all possible intersection points in $O(n^2 \log n)$ time. Now, we can determine the closest line to ℓ^* in $O(\log n)$ time by querying the sweepline status, when we encounter an event. The line corresponds to a point in the primal space and ℓ^* corresponds to a point.