Homework 2

Vu Tuan Phong Pham

September 2019

1. (*Placeholder, question answered in the attached python file.*)

2. **Note:** I am assuming that the input points are in general position (no 3 points are colinear, and no 2 points share the same x coordinate).

   (a) Let's consider a point P. I draw 2 lines through P that is parallel to the $x-$axis and the $y-$axis, which will divide the plane into 4 quadrants. Observe that whatever point that, when combine with P, will make a line with positive slopes will lie in the North-East and South-West quadrants. These points, hence, will have the property that their x and y coordinates will be either both more than x and y of P, or both less than x and y of P.

   Algorithm:
   We sort the list of points by x coordinate, and in case of tie, resolve by lexicographic order. Start by iterate from the points with lowest x coordinate. We maintain a binary search tree (maybe a red black tree) of y coordinates. Each time we get to a point P, we know that previous points already have lower x coordinate than P, so we just have to report the number of points that has y coordinate strictly less than y coordinate of P. This query can be done efficiently in $\mathrm{O}(\log n)$ with the binary search tree. After we are done with P, we then push $P.y$ to the binary search tree.

   There are n points, and for each point, it takes $\mathrm{O}(\log n)$ per query. The total time complexity is $\mathrm{O}(n \log n)$

   (b) Consider the duality of the plane: $\Phi(P(a, b)) \to y = ax - b$ (a point $P(a, b)$ is mapped to the line $y = ax - b$). The question of whether a line created by joining 2 points P, Q has a slope between $[m', m'']$ in the primary plane can be translated to the dual plane as whether the intersection of 2 lines P^*, Q^* corresponding to P, Q has the x coordinate in between $[m', m'']$. We already solve this intersections problem in class, the only modification that needs to be done is instead of counting all the intersection, we will only count the number of intersections that have x coordinate in between $[m', m'']$.

3. The general idea is to perform a rotation line sweep (360 degree line sweep), with the base at point p. We store endpoints of each segments by the polar coordinate (r, θ). We start the sweep at the endpoint that has the smallest r coordinate, and we sort the endpoint list by θ coordinate. We store a binary search tree of the distance to p for the status.

   Starting at the endpoints with smallest r coordinates, we go either CCW or CW depends on the other endpoint of that segment (we just want to make sure that the other endpoint gets sweep to right after, not at the end of the list). There are 2 cases:

   - **Case 1:** If the endpoint is a closing endpoint (ie. we already see this segment in the status), then we remove this segment from the status.

- **Case 2:** If this is an opening endpoint (ie. first time we see the segment), we push the segment into the status, using r coordinate of the current endpoint. Note that, using the θ coordinate of the current endpoint, we can re-compute the distance from p to segments already in the tree on the way of insertion, this way, the tree will stay correct in regard of distance to p. If the closest segment to p changed, we add 1 to the result.

The preprocess takes $O(n)$ time. For each query, in either case, the insertion and deletion in the binary search tree takes $O(\log n)$ time, and there are $2n$ such queries, so the total time complexity is $O(n \log n)$.

4. I'm assuming general position: no 2 points have the same x coordinates.

   The problem requires us to find 2 parallel lines such that there is at least 1 point on each line, and the band between these 2 lines does not contain any other points, and the distance between 2 lines has to be maximized. If we map this primal plane to a dual plane, all points will become lines, and 2 lines will become a segment such that they have the same x coordinate (same slopes), and largest difference in y coordinate (we can think of the distance between 2 lines to be a line shift).

   The problem is now, translated to the dual plane, given a list of lines, find the longest vertical segments such that its endpoints touch given lines and there is no line that cross the segment (excluding endpoints). Let this segment has 2 endpoints $P(x_p, y_p), Q(x_q, y_q)$, it has to satisfy the following property:

   - P, Q is on some lines
   - $x_p = x_q$
   - $|y_p - y_q|$ is maximized
   - No lines intersect PQ except at P, Q

   Also, we observe that for each intersections of line in the dual plane, it form a wedge, and we can find a potential answer segment by starting at the intersection of line p, q, increase x coordinate while keeping P, Q on the 2 lines until we meet the first intersection of a third line to p, q. This gives us a sweeping algorithm:

   First spawn the n^2 intersections in the dual plane by calculating n^2 support line of segments in the primal plane. We sort n^2 intersections by x-coordinate. At each intersection K, we push the wedge to the active list, and if we see an intersection H that shares a line with K, we calculate the longest segment satisfy the properties created by wedge H, and then we can delete H. The answer is the maximum segment we can find (note that we have to translate this back to the primal plane, and into the line we need to find).

   There are n^2 iteration (each iteration is an intersection). Each intersection is push and pop from the status once, so sweeping takes $O(n^2)$. Sorting at first takes $O(n^2 \log n)$. So total time complexity is $O(n^2 \log n)$.