CSL 852, Computational Geometry: Practice Problems

Voronoi diagram and delaunay triangulation

1. Prove the following about a set of points (no three are collinear and no four points are cocircular).
   (i) The circumcircle of a delaunay triangle doesn't contain any other input point.
   (ii) A pair of points is an edge of Delaunay triangulation iff there is a circle passing through these points that is empty.

2. Develop an $O(n \log n)$ algorithm for directly computing the Delaunay triangulation using divide and conquer.

3. Show that given the Voronoi diagram, one can find the nearest neighbour

4. The *Relative Neighbourhood Graph* (RNG) of a set of points is one where there is an edge between points p_i and p_j iff the $lune(i, j)$ does not contain any other points inside it. $Lune(i, j)$ is a region that is common to the two circles with radius $d(i, j)$ and centers p_i and p_j. Show that for any given set of points S

   $$\text{Euclidean MST (S)} \subseteq \text{RNG (S)} \subseteq \text{DT (S)}$$

5. **Motion planning** Given a scene with rectangular obstacles, and a disk of radius r, we would like to find out a feasible path from an initial position to a final position (avoiding the obstacles).

   Design and implement an efficient algorithm for this. You can use existing code for Voronoi Diagrams and integrate with your algorithm.

   Possible approaches:
   1. You can grow the obstacles using Minkowski's sum and compute the maximal connected regions. Solution: While growing objects, make sure that the corners of rectangle are rounded. This can be achieved by approximating the rounded corners with a large number of line segments. Then after objects are grown, join all corner points of objects with each other and retain those which do not pass through any object. Now using any shortest path algorithm, find a feasible path between the initial and final point using the graph obtained by vertices and edges above.

   2. Alternatively, you can approximate the obstacles by points on the boundary that are less than $2r$ distance - denote this by S'. You can then compute the Voronoi Diagram $Vor(S')$ and only retain those edges in $Vor(S')$ that at least r from the closest obstacle point. Now work with this sub-graph. This has the disadvantage that the input size may blow up significantly but you can make reasonable assumptions.

6. Given a set S of n points in the plane, design an efficient algorithm to find the *largest empty circle*, that doesn't contain any point of S and the center of the disk is within the $CH(S)$.

   Solution: It can be proved that the centre of the largest empty circle (where the centre has to be internal to $CH(S)$) is either a Voronoi vertex or the intersection of a Voronoi edge with a convex hull edge.
   Once this is proved, the $O(n \log n)$ algorithm follows:
   First compute $Vor(S)$ and put all Voronoi vertices in a list of candidates. Also, with every vertex, maintain one of its closest points in S by choosing any incident face of an incident edge on it.
   Since every Voronoi vertex need not lie inside the convex hull of S, do a planar point location for each Voronoi vertex to determine if it is inside $CH(S)$ or outside. For this compute $CH(S)$ as upper hull and lower hull. Then for each Voronoi vertex, determine, if it is inside or outside $CH(S)$ by using

a binary search on the slabs obtained by drawing vertical lines through all vertices. You could first do a binary search on the slabs formed by the upper hull vertices and locate in which slab a Voronoi vertex lies and then check if it is above the upper hull or below. Then you could check for the lower hull slabs. So this takes $O(\log n)$ time per Voronoi vertex and so $O(n \log n)$ in total.
Each Voronoi edge can intersect at most one edge of the upper hull and at most one edge of the lower hull - so the total number of points of intersection between edges of $Vor(S)$ and $CH(S)$ is $O(n)$. Use the line segment intersection algorithm to compute all the points of intersection - this runs in $O(n \log n)$ time by the above justification. Whenever we compute the intersection of a Voronoi edge with a convex hull edge, we not only store the point of intersection but also the closest point corresponding to the incident face of the Voronoi edge involved.
Finally, for each Voronoi vertex and for each point of intersection computed above, compute the distance to its closest point - this would be the radius of the largest empty circle centered at that point. There are $O(n)$ candidates and we have to choose the maximum among them - this takes $O(n)$ time. Hence, all the steps together take $O(n \log n)$ time.

7. Given a set S of n points in the plane, design an efficient algorithm to find the *smallest enclosing disk*, that contains all point of S.
Hint: Use Randomized incremental construction and analyse carefully.

   Solution: It can be observed that a smallest enclosing circle has at least three points on its boundary, or only two in which case they are diametrically opposite. Choose a random order to add the points and maintain the solution so far. If the new point p_i lies inside the solution circle obtained so far, then new solution is same as previous solution. Else, compute the smallest enclosing circle of points so far, with p_i on its boundary. This problem of finding the smallest enclosing circle passing through a given point is further solved by RIC similar to above. But now if the new point considered lies outside the solution circle obtained so far, then we have to compute the smallest enclosing circle passing through 2 given points. This problem can be solved as follows : Assume w.lo.g. that both points, say p and q lie on a vertical line. Define a first circle to be the smallest passing through p and q. Let l be the line through p and q. For all points left of l, find the one that, together with p and q, defines a circle whose center is leftmost. For all points right of l, find the one that, together with p and q, defines a circle whose center is rightmost. Then decide which of these 3 circles is the smallest enclosing circle.
   Smallest enclosing circle for n points with two points already known takes O(n) time. Using this, do the expected running time analysis of smallest enclosing circle for n points with one point known. Use the fact that the probability that the $i-th$ point addition is expensive is $2/i$. This gives an expected running time of O(n). Now using similar argument, show that the expected running time for smallest enclosing circle with none of the points on boundary known, is O(n)

8. Given a set S of n points in the plane, design an efficient algorithm to find a disk D that encloses k points (any of the k out of n points) and whose radius is no more than twice that of $D_o(n, k)$, the smallest disk that contains k points. When k is $\Omega(n)$, show that your algorithm runs in $O(n)$ time.
Hint: Prove the property that the smallest disk centered at $q \in D_o(n, k)$ that contains k points, will be no more than twice the radius of D_o.

   Solution: Divide the plane into horizontal and vertical lines such that the number of points between any two horizontal lines is $\frac{k}{4}$. Same is the case with vertical lines. This can be done in $O(n \log \frac{n}{k})$ time using median finding together with recursion. Now for all the intersection points of the grid induced by these lines, find the smallest circle centered at the point and containing exactly k of the n given points. Return the smallest of these circles. It can be easily argued that the $D_o(n, k)$ must contain one of the intersection points of grid, otherwise it cannot have k points inside it as it will lie

in the union of area of atmost one horizontal and one vertical strip, limiting the points inside it to $\frac{k}{4}$. Thus using the property given in hint, it is easy to see that the smallest circle returned above will be less than twice the size of $D_o(n,k)$.
Since the intersection points of grid are $O((\frac{n}{k})^2)$ and finding each circle takes $O(n)$ time, so total time taken is $O(n(\frac{n}{k})^2)$, which when k is $\Omega(n)$ gives $O(n)$.

9. Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in the plane so that no three of them lie on a line. The farthest-point Voronoi diagram of S is planar decomposition of the plane into maximal cells so that the same point of S is the farthest neighbor of all points within each cell. That is, it is the decomposition induced by the cells

$$\text{Vor}_f(p_i) = \{x \in R^2 \mid \|p_i x\| \geq \|p_j x\| \forall j\}.$$

   (i) Show that $\text{Vor}_f(p_i)$ is convex.

   (ii) Show that $\text{Vor}_f(p_i)$ is nonempty if and only if p_i is a vertex of the convex hull of S.

   (iii) Show that if $\text{Vor}_f(p_i)$ is nonempty then it is unbounded.

   Solution:

   (i) Fix a point $p_i \in S$, for any other point $p_j \in S$, let h_j^- denote the half-space defined by the perpendicular bisector of p_i, p_j, not containing p_i. The Voronoi cell of p_i, is the intersection of half-spaces h_j^-, i.e., $\text{Vor}_f(p_i) = \bigcap_{j \neq i} h_j^-$, and therefore it is convex.

   (ii) Part A: If p_i is a vertex of the convex hull of S, then $\text{Vor}_f(p_i)$ is non-empty.
   Since no three points of S are collinear, we can choose a tangent τ to the convex hull at p_i that is not parallel to any side of the boundary of the convex hull. Let ρ be the perpendicular to τ at p_i. As we move along ρ such that the distance to p_i increases, we can find a point q on ρ such that the disk of radius $||qp_i||$ contains every point of the convex hull, other than p_i, in its interior.
   So $||qp_i|| \geq ||qp_j||$, for all $i \neq j$, so $q \in \text{Vor}_f(p_i)$.
   Part B: If p_i is not a vertex of convex hull of S, then $\text{Vor}_f(p_i)$ is empty.
   Suppose for some p_i that is not a vertex of convex hull of S, there is an $x \in \text{Vor}_f(p_i)$. Let y be the point of intersection of the ray $\overrightarrow{xp_i}$ with the boundary of the convex hull of S such that y does not lie on the segment $\overline{xp_i}$. Using triangular inequality, it can be shown that one of the two vertices v of the convex hull that is adjacent to y is farther than p_i from x. Contradiction.

   (iii) Following the same argument as in (ii), Part A, for any point p_i on the convex hull we can find points q on the perpendicular ρ such that $q \in \text{Vor}_f(p_i)$, as we move away from p_i along ρ. So the Voronoi region is unbounded.