

COMPUTATIONAL GEOMETRY Practice Final – Solution Notes

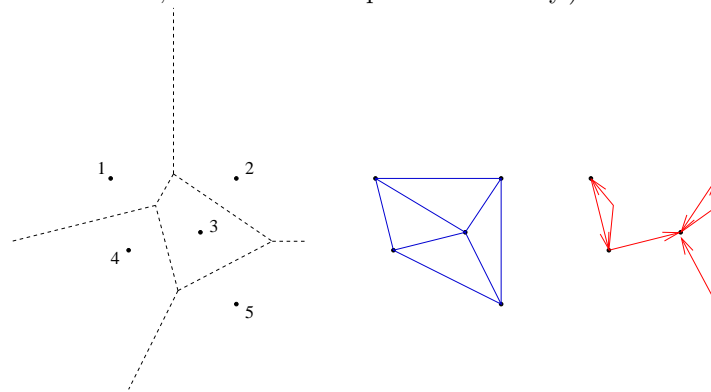
(1). The convex hull of n points in 3D can be computed in time $O(n \log n)$ (and this is best possible, in the worst-case, since $\Omega(n \log n)$ is a lower bound even in 2D). This can be done using divide-and-conquer, since the convex hull of two disjoint convex polytopes can be computed in linear time, allowing for linear-time merge: $T(n) = 2T(n/2) + O(n)$, solving to $T(n) = O(n \log n)$.

(2). (a). The Voronoi diagram is below, left. There are 4 Voronoi vertices, 8 Voronoi edges, 5 Voronoi regions. (If we count the Voronoi vertex “at infinity,” we may say there are 5 Voronoi vertices; however, this is not the convention followed by our text.)

(b). The Delaunay diagram is below, middle. There are 8 Delaunay edges, 4 Delaunay faces. (If we count the face at infinity (dual to the Voronoi vertex at infinity), there are 5 Delaunay faces; some authors count it this way, but our text does not.)

(c). The directed nearest neighbor graph is below, right. Note that there are two arcs out of point 4, since it has two nearest neighbors (at exactly the same distance from it).

(d). We would usually store the Voronoi diagram (and the Delaunay diagram) in a winged-edge, twin-edge, quad-edge or similar data structure. (For point location queries, we would augment the data structure with a point location data structure, such as the Kirkpatrick hierarchy.)



(3). (a). We can determine whether or not a set S of n line segments in the plane has *any* point of intersection among the segments in time $O(n \log n)$, using plane sweep (essentially using Bentley-Ottmann sweep, but stopping the first time we find an intersection).

(b). We can compute the Euclidean minimum spanning tree of n points in the plane in time $O(n \log n)$, by first computing the Delaunay diagram, and then applying Kruskal’s or Prim’s algorithm to compute the MST in the Delaunay diagram (which we know to contain the MST of the points).

(c). Given a winged edge data structure for a Delaunay diagram of n points in the plane, we can compute the convex hull of the points in time $O(n)$, by traversing the boundary of the face at infinity.

(d). We can preprocess a simple polygon P in time $O(n)$, by triangulating it and then building the Kirkpatrick hierarchy (or other point location data structure). Then, we can answer queries of the form *Is point q inside P ?* in time $O(\log n)$.

(e). We can preprocess a set S of n points in the plane to support efficient queries of the form: *Does the unit circle centered at point q contain any points of S ?* as follows. Build the Voronoi diagram (time $O(n \log n)$), then preprocess it for point location (additional $O(n)$ time). To answer a query, we locate q (in time $O(\log n)$), which then lets us know which point of S is closest to q – say it is p_i . We compute the distance from q to p_i , and if this distance is larger than 1, then the unit circle centered at q contains no points of S ; otherwise it does.

(4). (a). The nearest neighbor graph of n points in the plane is connected: **SOMETIMES TRUE**. (It can be false: consider points $(0,0)$, $(1,0)$, $(10,0)$, $(11,0)$. It can be true: see problem 1.)

(b). The Delaunay diagram of n points in the plane is connected: ALWAYS TRUE. We know that the Delaunay diagram contains the MST (which is by definition a connected graph).

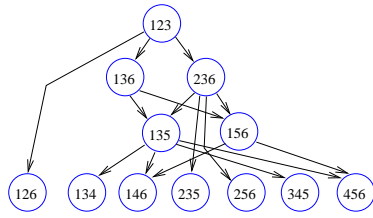
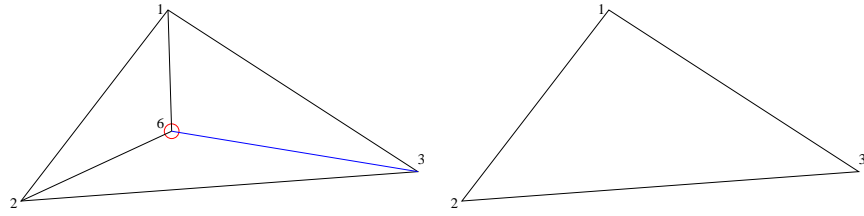
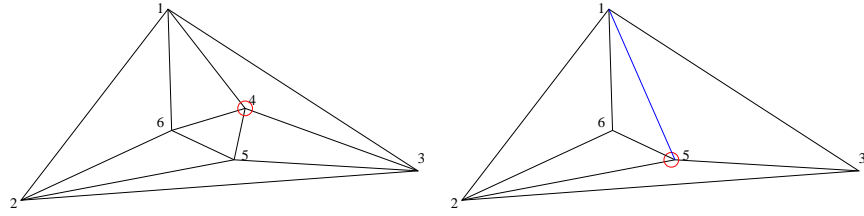
(c). An optimal travelling salesperson tour on n points in the plane is the boundary of a simple polygon: ALWAYS TRUE. If a tour intersects itself, it can be shortened by doing an exchange (by the triangle inequality).

(d). A polygonal subdivision of the plane into n convex polygonal cells is the Voronoi diagram for some appropriately chosen set of sites, S , with one point per cell: SOMETIMES TRUE. It can be false (you can readily draw examples – do it!)

(5). Below, I give the full schedule of events for the sweep algorithm. Note that there is a degeneracy: events b_2 and x_{45} occur simultaneously; thus, we break the tie by ordering the events according to their x -coordinates.

Event	Event Queue, Q	Sweep Status, \mathcal{L}
-	$(a_1 a_3 a_4 a_2 a_5 b_1 b_2 b_3 b_4 b_5)$	$()$
a_1	$(a_3 a_4 a_2 a_5 b_1 b_2 b_3 b_4 b_5)$	(s_1)
a_3	$(a_4 a_2 a_5 b_1 b_2 b_3 b_4 b_5)$	(s_1, s_3)
a_4	$(a_2 a_5 b_1 x_{34} b_2 b_3 b_4 b_5)$	(s_1, s_4, s_3)
a_2	$(x_{24} a_5 b_1 x_{34} b_2 b_3 b_4 b_5)$	(s_1, s_4, s_2, s_3)
x_{24}	$(a_5 x_{12} b_1 x_{34} b_2 b_3 b_4 b_5)$	(s_1, s_2, s_4, s_3)
a_5	$(x_{12} b_1 x_{34} b_2 b_3 b_4 b_5)$	$(s_1, s_2, s_4, s_3, s_5)$
x_{12}	$(b_1 x_{34} b_2 b_3 b_4 b_5)$	$(s_2, s_1, s_4, s_3, s_5)$
b_1	$(x_{34} b_2 b_3 b_4 b_5)$	(s_2, s_4, s_3, s_5)
x_{34}	$(b_2 x_{45} b_3 b_4 b_5)$	(s_2, s_3, s_4, s_5)
b_2	$(x_{45} b_3 b_4 b_5)$	(s_3, s_4, s_5)
x_{45}	$(b_3 b_4 b_5)$	(s_3, s_5, s_4)
b_3	$(b_4 b_5)$	(s_5, s_4)
b_4	(b_5)	(s_5)
b_5	$()$	$()$

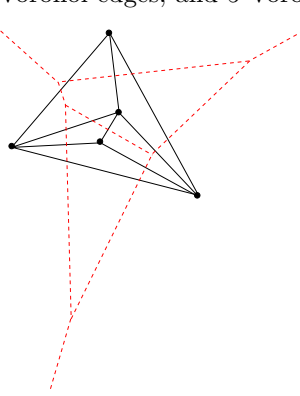
(6).



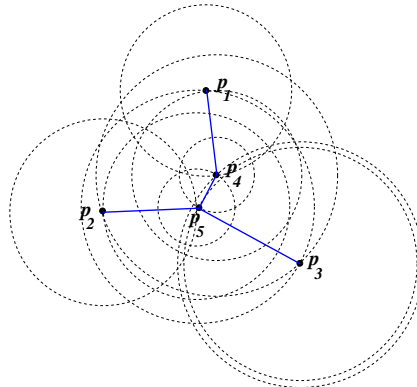
COMPUTATIONAL GEOMETRY

Another Practice Final – Solution Notes

(1). The Delaunay diagram is drawn in solid black; the Voronoi in dashed red. There are 9 Delaunay edges, 5 Delaunay faces, 5 Voronoi vertices, 9 Voronoi edges, and 5 Voronoi regions.



The Relative Neighborhood Graph is drawn in solid blue below. An edge $p_i p_j$ is drawn if and only if the Lune(p_i, p_j) is empty. I draw several circles determined by pairs of points, so that it is easy to tell which lunes are empty.



(2). (a). There are $3n$ segments, so we can compute all k intersections between pairs of segments in time $O(k + n \log n)$, using an optimal line segment intersection algorithm (e.g., Chazelle-Edelsbrunner). This gives us the answer we need. (A less than optimal answer is to use the Bentley-Ottmann sweep, which takes more time: $O(k \log n + n \log n)$.)

(b). The NNG is a subgraph of the Delaunay diagram. The Delaunay diagram can be constructed in time $O(n \log n)$ and is of size $O(n)$. We just walk through the diagram and for each vertex v check, in time $O(\deg(v))$, which is the closest neighbor, given by one of its Delaunay neighbors. Since $\sum_v \deg(v) = 2e = O(n)$, the total time is $O(n \log n) + O(n) = O(n \log n)$. (Note that the very naive answer of building the NNG in time $O(n^2)$ received only 1 point out of 5 points.)

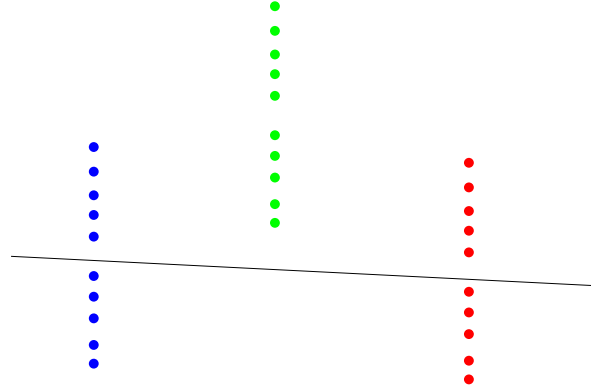
(c). One can walk through the winged edge data structure of the Voronoi diagram in time $O(n)$ to obtain the winged edge data structure of the Delaunay diagram (and vice versa).

(d). One can build the Voronoi diagram of the $3n$ points in time $O(n \log n)$, preprocess it in time $O(n)$ for point location, using a data structure of size $O(n)$ (e.g., Kirkpatrick hierarchy), and then a point location query is answered in time $O(\log n)$. Once we know the point closest to q , we report its color.

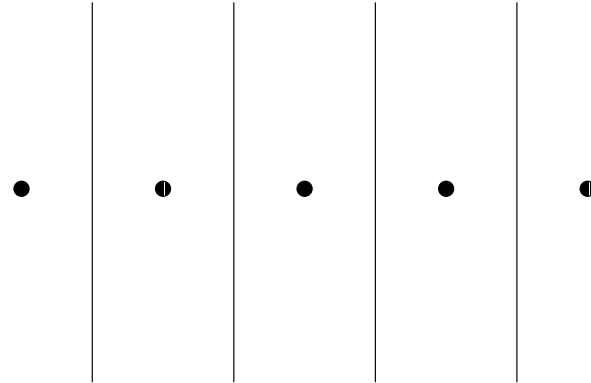
(e). Given a set \mathcal{L} of n lines, we can build a winged-edge data structure for their arrangement, $A(\mathcal{L})$, in time $O(n^2)$, by the incremental insertion algorithm. The Zone Theorem tells us that each insertion takes time $O(n)$.

(f). Given n points in 4D, their convex hull can be computed in time $O(n^2)$. In general, for $d \geq 4$, the convex hull of n points in d dimensions can be constructed in time $O(n^{\lfloor d/2 \rfloor})$, and this is worst-case optimal.

(3). (a). SOMETIMES TRUE (and sometimes FALSE, as in the figure below): Given n red points, n blue points, and n green points in the plane, there exists a line L that simultaneously splits the red, blue, and green points in half ($n/2$ on each side of L). (You may assume that n is even.)

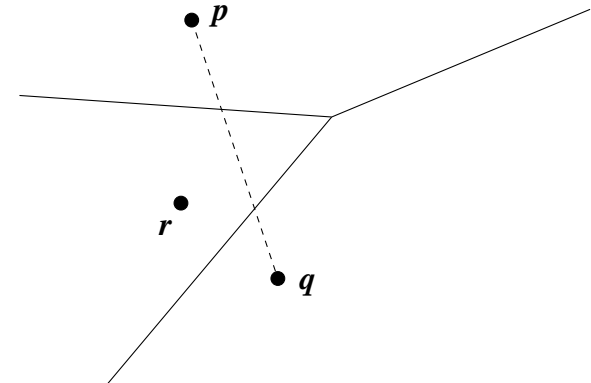


(b). SOMETIMES TRUE (and sometimes FALSE, as in the figure below): For any set of n distinct points in the plane, the Voronoi edges and Voronoi vertices form a connected graph.



(If we consider the “Voronoi vertex at infinity” to be a Voronoi vertex (which we usually do not), then the Voronoi graph is connected.)

(c). SOMETIMES TRUE (and sometimes FALSE, as in the figure below): If (p, q) is a Delaunay edge, then the line segment pq crosses the Voronoi edge that is shared by the cells $V(p)$ and $V(q)$.



(d). SOMETIMES TRUE (e.g., when the p_i 's are all collinear along a **vertical** line), and sometimes FALSE (if the points are collinear along a non-vertical line ℓ , in which case the duals are a bundle of lines all passing through a common point (the dual of ℓ)).

Event	Event Queue, Q	Sweep Status, \mathcal{L}
-	$(a_1 a_4 a_5 a_3 a_2 b_3 b_4 b_2 b_5 b_1)$	$()$
a_1	$(a_4 a_5 a_3 a_2 b_3 b_4 b_2 b_5 b_1)$	(s_1)
a_4	$(a_5 a_3 a_2 x_{14} b_3 b_4 b_2 b_5 b_1)$	$(s_1 s_4)$
a_5	$(a_3 x_{45} a_2 [x_{14}] b_3 b_4 b_2 b_5 b_1)$	$(s_1 s_5 s_4)$
a_3	$(x_{45} a_2 [x_{14}] b_3 b_4 b_2 b_5 b_1)$	$(s_1 s_5 s_4 s_3)$
x_{45}	$(a_2 x_{14} b_3 b_4 b_2 b_5 b_1)$	$(s_1 s_4 s_5 s_3)$
a_2	$(x_{25} x_{14} b_3 b_4 b_2 b_5 b_1)$	$(s_1 s_4 s_5 s_2 s_3)$

(4).

(5). The independent sets chosen are, in order, $\{4\}$, then $\{5\}$. Below I draw the triangulations (left to right), and the DAG below.

