1. Suppose you are given a set of n circular disks in the plane, all with the same radius. Describe and analyze an efficient algorithm to determine whether the union of these disks is connected. For full credit your algorithm should run in $O(n \log n)$ time.

Solution: Without loss of generality, suppose all input disks have radius 1; let P be the set of all disk centers. At a high level, we construct the Delaunay triangulation of P, discard all edges of length greater than 2, and then report whether the remaining subgraph is connected. The entire algorithm runs in $O(n \log n)$ time.

But why does it work? For each point $p \in P$, let $D(p)$ denote the unit disk centered at p, and let U denote the union of these disks. Finally, let G be the subgraph of the Delaunay triangulation of P containing all edges of length at most 2. (G is sometimes called the *unit-disk graph* or *proximity graph* of P.)

Lemma. *U is connected if and only if G is connected.*

Proof: First, suppose U is connected; this means for any two points $s, t \in U$, there is a path in U from s to t. Choose any two points $s, t \in P$, and let π be a path in U from s to t. Let $Vor(p_1), Vor(p_2), \ldots, Vor(p_L)$ be the finite[a] sequence of Voronoi regions that π intersects, where $p_1 = s$ and $p_L = t$. By definition, the portion of π within any Voronoi region $Vor(p_i)$ is closer to p_i than to any other disk center, and therefore lies entirely within the disk $D(p_i)$. For each index i, the centers p_i and p_{i+1} are Delaunay neighbors, because their Voronoi regions $Vor(p_i)$ and $Vor(p_{i+1})$ intersect at the point $\pi \cap Vor(p_i) \cap Vor(p_{i+1})$. Moreover, the distance from p_i to p_{i+1} is at most 2, because the intersection point $\pi \cap Vor(p_i) \cap Vor(p_{i+1})$ lies within both disks $D(p_i)$ and $D(p_{i+1})$. We conclude that $s = p_1$ and $t = p_L$ are connected in G.

On the other hand, suppose G is connected. Let s and t be any two points in U. Let s' and t' be the nearest neighbors of s and t in P. Because G is connected, there is a path π' in G from s' to t'. Each edge $p_i p_{i+1}$ of this path has length at most 2, and therefore lies in the union $D(p_i) \cup D(p_{i+1}) \subseteq U$. Thus, the entire path π' lies in U. Adding segments ss' and tt' (which lies in $D(s')$ and $D(t')$, respectively) gives us a path π in U from s to t. We conclude that U is connected. □

[a]Formally, assuming that this sequence is finite requires some subtle topological arguments.

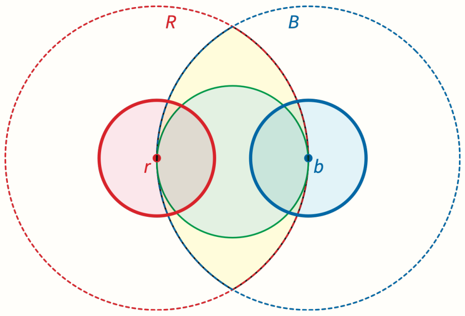
Solution (via duality): Without loss of generality, assume all disks have radius 1; let P be the set of disk centers. At a high level, we construct the *Voronoi diagram* of P, discard all edges whose corresponding Delaunay edges have length *at most* 2, and then report whether the remaining subgraph *has any cycles*. The entire algorithm runs in $O(n \log n)$ expected time.

But why does this work?? For each point $p \in P$, let $D(p)$ denote the unit disk centered at p, and let U denote the union of these disks. Call an edge of the Voronoi diagram of P a *fence* if the corresponding Delaunay edge has length greater than 2. Equivalently, a Voronoi edge is a fence if the Voronoi regions it separates belong to the centers of two disjoint balls. Let F be the graph whose nodes are the Voronoi vertices (including the sentinel vertex at infinity) and whose edges are Fences. We can construct F in $O(n \log n)$ expected time by building the Voronoi diagram (or equivalently, the Delaunay triangulation) and then checking each of the $O(n)$ Voronoi edges by brute force.

Lemma. *U is connected if and only if the graph F is a forest.*

Proof: First, suppose the graph F contains a cycle C.[a] The Jordan Curve Theorem implies that C separates the plane into two components, each containing at least one Voronoi cell, and therefore each containing at least one point in P. Color the points on one side of C red, and color the points on the other side of C blue.

Let r and b be the closest red/blue pair of points in P. The open disk R centered at r with b on its boundary contains no blue points. Similarly, the open disk B centered at b and with r on its boundary contains no red points. Thus, the lens $R \cap B$ contains no points in P. This lens contains the circular disk centered at the midpoint of rb with both r and b on its boundary. Thus, r and b are neighbors in the Delaunay triangulation of P; their Voronoi regions share an edge. That Voronoi edge must be a fence—otherwise, r and b would have the same color—so the distance between r and b is greater than 2. It follows that *every* red/blue pair has distance greater than 2; in other words, no red disk intersects any blue disk.[b] We conclude that the union of disks is disconnected.



Now suppose on the other hand that the union of disks is disconnected. Color the disks in one component of the union red and all other disks blue. Color each Voronoi regions to match the disk that defines it, and let R be the union of the red Voronoi regions. Because no red disk intersects a blue disk, any Voronoi edge incident to both a red region and a blue region is a fence. Thus, the subgraph F contains the entire boundary of R. If R is bounded, its boundary is a cycle in F that avoids the sentinel vertex at infinity (geometrically, a simple polygon). If R is unbounded, then its boundary is a cycle in F that includes the sentinel vertex at infinity (geometrically, and infinite polygonal curve). In both cases, F contains a cycle and thus is not a forest. □

[a] If C contains the sentinel vertex, then it is geometrically an infinite polygonal path. Otherwise, C is a simple polygon.

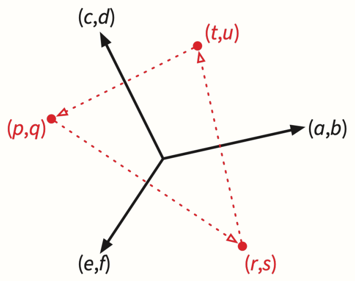
[b] More strongly, no disk intersects the cycle C.

Rubric: 10 points = 6 for algorithm + 4 for justification. This is more detail than necessary for full credit. These are not the only correct solutions.

2. (a) Describe and analyze an algorithm to determine, given a planar straight-line graph G, whether G is a Voronoi diagram. Your algorithm should either return a finite point set P such that G is the Voronoi diagram of P, or report correctly that no such point set exists. *[Hint: Consider the case of four sites.]*

Solution: I'll assume that every vertex in input graph G has degree 3 (general position) and that every face of G is strictly convex; we can verify both of these conditions in $O(n)$ time.

First, let's consider the case $n = 3$. Suppose we are given a "Voronoi diagram" of three sites, consisting of three infinite rays from a single Voronoi vertex; without loss of generality, suppose the vertex is the origin $(0, 0)$. Let (a, b), (c, d), and (e, f) be nonzero vectors pointing outward along the Voronoi edges in counterclockwise order, and let (p, q), (r, s), and (t, u) be the unknown sites, as shown below.



The unknown site coordinates satisfy the following set of linear equations. The three equations on the left state that corresponding Delaunay and Voronoi edges are perpendicular; the three equations on the right place the midpoints of the Delaunay edges on the lines through their dual Voronoi edges.

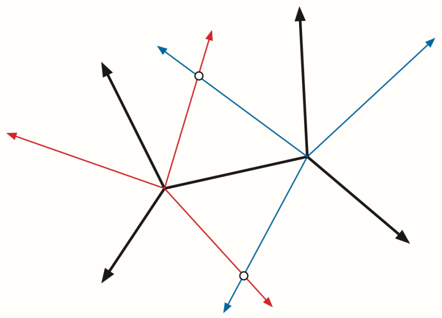
$$\begin{aligned} ar + bs &= at + bu & \qquad au + as &= br + bt \\ ct + du &= cp + dq & \qquad cq + cu &= dp + dt \\ ep + fq &= er + fs & \qquad eq + es &= fp + fr \end{aligned}$$

Rewriting in matrix notation:

$$\begin{bmatrix} 0 & 0 & a & b & -a & -b \\ 0 & 0 & b & -a & b & -a \\ -c & -d & 0 & 0 & c & d \\ d & -c & 0 & 0 & d & -c \\ e & f & -e & -f & 0 & 0 \\ f & -e & f & -e & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \\ s \\ t \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This linear system has rank 5, so it always has a one-parameter family of solutions $\lambda(p, q, r, s, t, u)$ for any $\lambda \in \mathbb{R}$.

The Voronoi diagram of any set of $n > 3$ points in general position contains at least one finite Voronoi edge e. Solving the three-point problem for each of these Voronoi vertices yields a triple of rays out of each once. In particular, for each of the Voronoi regions on either side of e, we obtain two rays that *must* intersect at the site for that region.



Once we have the location of one site, we can compute the locations of the remaining sites by reflecting across the Voronoi edges, using a whatever-first search of the dual graph. Finally, we verify that each Voronoi edge is in fact the perpendicular bisector of the corresponding pair of sites. (If the input is *not* a Voronoi diagram, different paths through the dual graph will assign different coordinates to the same site; this final check ensures that the coordinate assignments are consistent.)

The entire algorithm runs in ***$O(n)$ time***. ■

Rubric: 5 points. This is more detail than necessary for full credit.

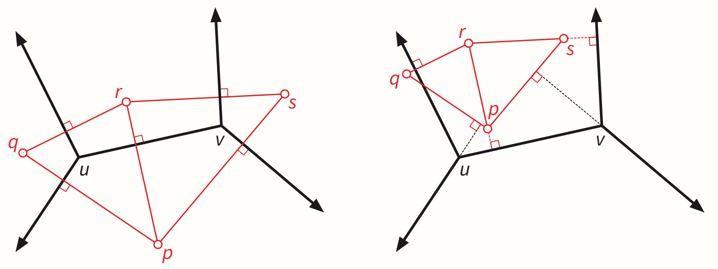
(b) Generalize your algorithm from part (a) to *weighted* Voronoi diagrams (also known as *power diagrams*). Your new algorithm should either return a finite *weighted* point set P such that G is the weighted Voronoi diagram of P, or report correctly that no such weighted point set exists. *[Hint: Consider the case of five sites.]*

Solution: As in part (a), I'll assume that every vertex in input graph G has degree 3 and that every face of G is strictly convex. I'll derive a linear-time algorithm based on the following claim:

Lemma. *A 3-regular planar straight-line graph G with strictly convex faces is a weighted Voronoi diagram if and only if there is an embedding of the dual graph G^* such that every edge in G is orthogonal to its dual edge in G^*.*

Proof: One direction of this claim is straightforward: If G is a weighted Voronoi diagram, the corresponding weighted Delaunay triangulation G^* is dual to G and corresponding edges are orthogonal. So let G^* be ***ANY*** planar straight-line triangulation that is both dual and orthogonal to G, and let P denote the set of vertices of G^*. I'll derive weights $w\colon P \to \mathbb{R}$ so that G is the weighted Voronoi diagram of (P, w) and G^* is the weighted Delaunay triangulation of (P, w).

The figure on the next page shows two examples of perpendicular dual triangulations G^* for the same four-region diagram G. Yes, these are both weighted Delaunay triangulations of two different weighted points sets with exactly the same weighted Voronoi diagram. More generally, scaling and translating any perpendicular dual of G yields another perpendicular dual of G, and therefore another set of points for which G is a weighted Voronoi diagram.



Let uv be any finite edge of G, and let pr be the corresponding dual edge in G^*, as in the figures above. The weights of p and r must satisfy the identities

$$\begin{aligned} \|p-u\|^2 - w(p) &= \|r-u\|^2 - w(r) \\ \|p-v\|^2 - w(p) &= \|r-v\|^2 - w(r) \end{aligned}$$

By assumption, the segments pr and uv are orthogonal, so $(p-r)\cdot(u-v) = 0$. Let $\|v\|^2 = v \cdot v$ denote the squared length of vector v. Then we have

$$\begin{aligned} \|p-u\|^2 - \|r-u\|^2 &= \|p\|^2 - \|r\|^2 - 2u\cdot(p-r) \\ &= \|p\|^2 - \|r\|^2 - 2v\cdot(p-r) \\ &= \|p-v\|^2 - \|r-v\|^2 \end{aligned}$$

Thus, we can assign weights to P as follows. Fix an arbitrary spanning tree T of G^*, rooted at an arbitrary point $o \in P$. Then set $w(o) := 0$ (or to any other value) and for each directed edge $p{\rightarrow}r$ in T (via postorder traversal), set

$$w(r) := w(p) - \|p-v\|^2 + \|r-v\|^2,$$

where v is either endpoint of the edge of G dual to pr. The previous analysis implies that any spanning tree T will yield the same weights. This weight assignment guarantees that each vertex v of G has the correct power-distance to the vertices of the corresponding triangle in G^*.

Now consider any two neighboring counterclockwise triangles pqr and qrs in G^*, dual to neighboring vertices u and v in G, respectively. Because qr and uv are perpendicular, we have $p\cdot(v-u) < s\cdot(v-u)$. These inequalities imply that

$$\begin{aligned} &\big(\|s-u\|^2 - w(s)\big) - \big(\|p-u\|^2 - w(p)\big) \\ &\qquad = \big(\|s-u\|^2 - \|s-v\|^2\big) - \big(\|p-u\|^2 - \|p-v\|^2\big) \\ &\qquad = -2s\cdot u + 2s\cdot v + 2p\cdot u - 2p\cdot v \\ &\qquad = 2(s-p)\cdot(v-u) > 0, \end{aligned}$$

and therefore

$$\|p-u\|^2 - w(p) \;<\; \|s-u\|^2 - w(s)$$

A symmetric (or equivalent) calculation implies $\|p-v\|^2 - w(p) > \|s-v\|^2 - w(s)$. It follows that the edge qr is locally weighted Delaunay! Applying the same analysis to all internal edges of G^* (dual to the finite edges of G), we conclude that G^* is the weighted Delaunay triangulation of (P, w), and thus G is the weighted Voronoi diagram of (P, w). □

With this lemma in hand, we can decide whether G is a weighted Voronoi diagram as follows. Choose an arbitrary finite edge uv of G, and fix an ***arbitrary*** lines segment pr perpendicular to uv, and declare the points p and r are dual to the regions on either side of uv. Then for any vertex of G, if two of its neighboring regions have been assigned dual points, we can define the site for the third region by extending perpendiculars from the other two sites. After placing all sites, we verify that all edges of the resulting triangulation are in fact perpendicular to the corresponding edges in G. (Again, if G is not a weighted Voronoi diagram, different traversals through G^* will assign different coordinates to the same site, so we need to check consistency.) Finally, we can assign weights as described in the previous proof. The entire algorithm runs in ***$O(n)$ time***. ■

Rubric: 5 points. This is *far* more detail than required for full credit.

⋆(c) **Extra credit:** Describe an efficient algorithm to determine, given a triangulation T of a set of n points *with integer coordinates*, whether T is a weighted Delaunay triangulation. Your algorithm should either report an appropriate weight assignment, or report correctly that no such assignment exists.

Solution: This is a *linear programming* problem. If T is a weighted Delaunay triangulation, then the weights satisfy the following inequality for each pair of adjacent triangles $p_i p_j p_k$ and $p_i p_k p_l$ in T:

$$\det \begin{bmatrix} 1 & x_i & y_i & x_i^2 + y_i^2 - w_i^2 \\ 1 & x_j & y_j & x_j^2 + y_j^2 - w_j^2 \\ 1 & x_k & y_k & x_k^2 + y_k^2 - w_k^2 \\ 1 & x_l & y_l & x_l^2 + y_l^2 - w_l^2 \end{bmatrix} > 0$$

Recall that the coordinates x_i and y_i are fixed, as part of the input; only the weights w_i are unknown. Cofactor expansion around the last column reveals that this is a *linear* inequality in the *squared* weights w_i^2. Thus, we need to find any weight vector $(w_1^2, w_2^2, \ldots, w_n^2)$ that satisfies all $O(n)$ strict inequalities, or report that no such weights exists.

To transform this system of strict inequalities into a standard linear program, we can introduce a new variable ε, replace every strict inequality $\det > 0$ with the loose inequality $\det \geq \varepsilon$, and then maximize ε. The triangulation is weighted Delaunay if and only if the maximum feasible value of ε is positive.

If all coordinates x_i and y_i are integers, we can solve the resulting linear program in polynomial time (on the integer RAM) using the ellipsoid method (for example). *It is a long-standing open question whether high-dimensional linear programming problems can be solved in polynomial time on the real RAM.* ■

3. Let P be a convex polygon in the plane with n vertices, represented as a circular doubly-linked list of vertices in counterclockwise order. The following randomized incremental algorithm constructs the Delaunay triangulation of the vertices of P.

```
ConvexDelaunay($P$):
  if $P$ is a triangle
      return $P$
  else
      $q \leftarrow$ **random** vertex of $P$
      $p \leftarrow pred(q)$;  $r \leftarrow succ(q)$
      $P' \leftarrow P - pq - qr + pr$
      $T' \leftarrow$ ConvexDelaunay($P'$)
      $T \leftarrow T' + pq + qr$
      repair $T$ by flipping non-Delaunay edges opposite $q$
      return $T$
```

(a) Prove that the expected number of flips performed by this algorithm is $O(n)$.

Solution: Imagine running the algorithm backward: Choose a random point q, flip away each diagonal incident to q, delete q and its two incident convex hull edges, and recursively delete the remaining triangulation. The Delaunay triangulation of P has exactly $n-3$ diagonals, each incident to two vertices, so the expected number of diagonals incident to a random vertex is $(2n-6)/n < 2$. Thus, the expected number of flips to delete q is less than 2.

We conclude that the expected number of flips over the entire algorithm is less than $2n = O(n)$. (The *exact* expected number of flips is $2n - 6H_n + 5$.) ■

(b) We already know from Lawson's algorithm that we can perform the necessary flips in $O(1)$ time each. But how do we choose the random vertex q in $O(1)$ time?

Solution: In a preprocessing phase, we build an array *Shuffled*$[1..n]$ of pointers, where *Shuffled*$[i]$ points to the ith vertex of P, and then randomly shuffle this array. Building and shuffling this array takes $O(n)$ worst-case time.

Finally, we modify ConvexDelaunay to insert points in the (random!) order specified by the array *Shuffled*. ■

4. Describe and analyze algorithms for each of the following natural variants of problem 3. The input to each problem is a convex polygon P, represented as a doubly-linked list of n vertices in counterclockwise order. For full credit, each of your algorithms should run in $O(n)$ expected time. Describe only the necessary changes to the algorithm from problem 3 and its analysis.

   (a) Describe and analyze an algorithm to construct the *Voronoi diagram* of the vertices of P.

   Solution: Run the algorithm in problem 3. Type-cast the resulting Delaunay data structure (or "flip the dual bit") to interpret it as a Voronoi diagram.

   (Depending on the precise form of the desired output, we may need to add methods to the Delaunay DCEL data structure that return the coordinates of Voronoi vertices (circumcenters of Delaunay triangles) and equations of lines supporting Voronoi edges (perpendicular bisectors of Delaunay edges. These methods can be implemented to run in $O(1)$ time each, so they behave exactly as if they were explicit fields.) ■

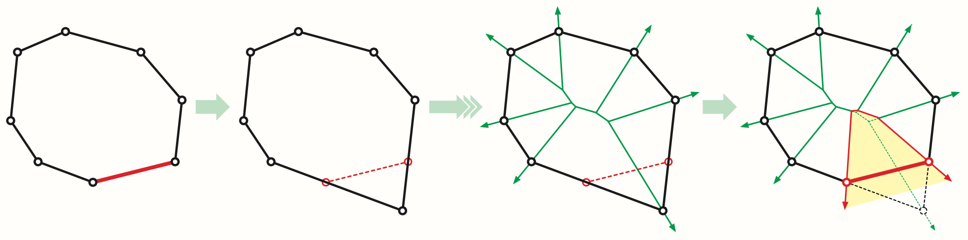
   (b) Describe and analyze an algorithm to construct the ***anti-Voronoi*** diagram of the vertices of P.

   Solution: Compute the anti-Delaunay triangulation of P using the algorithm from problem 3, but flipping away *Delaunay* edges incident to q instead of flipping away non-Delaunay edges.[a] Because the anti-Delaunay triangulation also has $n-3$ diagonals, precisely the same analysis implies expected running time $O(n)$. Type-cast the resulting data structure to interpret it as the dual anti-Voronoi diagram. triangulation. ■

   [a]Equivalently, after lifting the points to the unit paraboloid, we flip away any edge of the lifted triangulation that is convex (down) instead of concave (up). To see that this algorithm correctly computes the upper envelope of the lifted points, stand on your head.

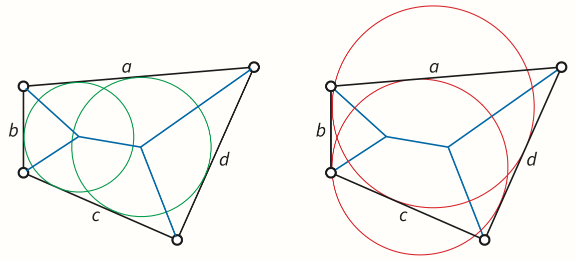
   (c) Describe and analyze an algorithm to construct the ***medial axis*** of P.

   Solution: We follow a randomized incremental strategy similar to problem 3. We regard the input polygon P as the intersection of halfspaces, one for each edge. First we delete a random *edge* e of P, extending the neighboring edges until they intersect (or to infinity if they don't intersect) to obtain a simpler convex polygon P'. Then we recursively compute the medial axis of P'. Finally, we reinsert e, cutting off a corner of P', and then compute the medial axis region of e.



The dual graph of the medial axis is a triangulation, whose vertices correspond to the *edges* of P. Equivalently, let P^* be the convex polygon that has one vertex for each edge of P; the dual of the medial axis of P is a triangulation of P^*. We can construct this dual triangulation in $O(n)$ expected time using exactly the same delete-recurse-insert-flip strategy as problem 3; the only question is how to decide which edges to flip.

Just like the local Delaunay condition, the "local median" condition asks whether a triangulation of four vertices of P^* is consistent with the medial axis of the four corresponding edges of P (or more accurately, the halfspaces that determine these four edges). We can characterize this condition geometrically as follows: The *incircle* of a (possibly unbounded) convex polygon with three sides is the unique circle tangent to all three edges. Let Q be a convex quadrilateral bounded by four edges a, b, c, d in cyclic order. The medial axis of Q includes a bisector segment of a and c if and only if the incircle of a, b, c does not intersect d, or equivalently, the incircle of b, c, d does intersect a.



We can obviously check this condition in $O(1)$ time, because it only depends on the coefficients of four lines. ***This is enough for full credit.***

We can express this condition *algebraically* by developing an appropriate lifting transformation. Without loss of generality, let's assume the origin lies strictly in the interior of P. Then we can express each halfspace bounding P as the inequality $a_i x + b_i y \le 1$ for unique coefficients a_i and b_i. Let $d_i(x, y)$ denote the signed distance from (x, y) to the line $a_i x + b_i y = 1$, signed so that $z_i(0,0) < 0$. The gradient of d_i is the unit vector parallel to (a_i, b_i), and therefore

$$d_i(x,y) = \frac{a_i}{\sqrt{a_i^2 + b_i^2}}x + \frac{b_i}{\sqrt{a_i^2 + b_i^2}}y - \frac{1}{\sqrt{a_i^2 + b_i^2}}$$

If we treat this formula as the equation of a plane $\hat{p}_i^*$, the medial axis is the projection of the upper envelope of these planes (the graph of the *maximum* of the n distance functions d_i). In particular, the we can find both the coordinates (x, y) of a medial-axis vertex and the distance r from that vertex to its three defining edges by solving the linear system

$$d_i(x,y) = d_j(x,y) = d_k(x,y) = -r.$$

Under the projective duality that maps points (α, β, γ) to planes $z = \alpha x + \beta y - \gamma$

and vice versa, each plane $\hat{p}_i^*$ is dual to the point

$$\hat{p}_i = \left(\frac{a_i}{\sqrt{a_i^2 + b_i^2}}, \frac{b_i}{\sqrt{a_i^2 + b_i^2}}, \frac{1}{\sqrt{a_i^2 + b_i^2}} \right),$$

and the dual triangulation is the projection of the *lower* convex hull of these n points. Equivalently, the dual medial triangulation is a weighted Delaunay triangulation of points p_i with weights w_i^2, where

$$p_i = \left(\frac{a_i}{\sqrt{a_i^2 + b_i^2}}, \frac{b_i}{\sqrt{a_i^2 + b_i^2}} \right) \quad \text{and} \quad w_i^2 = 1 - \frac{2}{\sqrt{a_i^2 + b_i^2}}.$$

(The points p_i all lie on the unit circle!) Thus, the "medial incircle" test is algebraically equivalent to the orthocircle test that defines local-weighted-Delaunay-ness.

This test is more directly equivalent to the orientation test for a quadruple of points $\hat{p}_i, \hat{p}_j, \hat{p}_k, \hat{p}_l$:

$$\det \begin{bmatrix} 1 & \frac{a_i}{\sqrt{a_i^2+b_i^2}} & \frac{b_i}{\sqrt{a_i^2+b_i^2}} & \frac{1}{\sqrt{a_i^2+b_i^2}} \\ 1 & \frac{a_j}{\sqrt{a_j^2+b_j^2}} & \frac{b_j}{\sqrt{a_j^2+b_j^2}} & \frac{1}{\sqrt{a_j^2+b_j^2}} \\ 1 & \frac{a_k}{\sqrt{a_k^2+b_k^2}} & \frac{b_k}{\sqrt{a_k^2+b_k^2}} & \frac{1}{\sqrt{a_k^2+b_k^2}} \\ 1 & \frac{a_l}{\sqrt{a_l^2+b_l^2}} & \frac{b_l}{\sqrt{a_l^2+b_l^2}} & \frac{1}{\sqrt{a_l^2+b_l^2}} \end{bmatrix} > 0?$$

We can simplify the matrix, without changing the sign of its determinant, by multiplying each row by its common denominator:

$$\det \begin{bmatrix} \sqrt{a_i^2 + b_i^2} & a_i & b_i & 1 \\ \sqrt{a_j^2 + b_j^2} & a_j & b_j & 1 \\ \sqrt{a_k^2 + b_k^2} & a_k & b_k & 1 \\ \sqrt{a_l^2 + b_l^2} & a_l & b_l & 1 \end{bmatrix} > 0?$$

This is a standard 3D orientation test with the columns permuted! Moreover, the only difference from the circumcircle test for standard Delaunay triangulations is the square root.[a] Geometrically, we are "lifting" points to the inverted unit cone $z = -(x^2 + y^2)^{1/2}$ instead of the unit paraboloid $z = \frac{1}{2}(x^2 + y^2)$. ■

[a]Evaluating the incircle inequality directly in $O(1)$ time requires our underlying real RAM model to support exact $O(1)$-time square roots. With more effort, we can eliminate square roots from the inequality by repeatedly gathering terms and squaring, leaving us with a multivariate polynomial inequality of degree 24 in the 8 line coefficients. Can, meet worms.