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

   (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 $2n - 6H_n + 5$.) ■

   Rubric: 5 points.

   (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*. ■

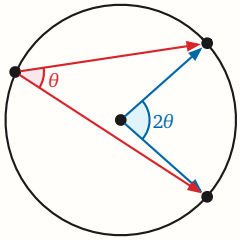
   Rubric: 5 points.

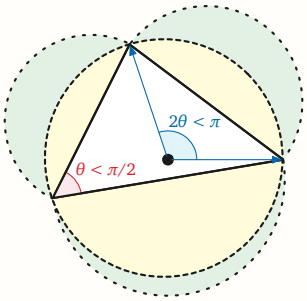
2. Let P be any set of points in the plane. A triangulation of P is a planar straight-line graph whose vertices are precisely the points P, and whose bounded faces are triangles whose union is the convex hull of P.

   (a) Let T be any triangulation of P. Prove that if every interior angle of T is acute (that is, strictly smaller than a right angle), then T is the Delaunay triangulation of P.

Solution: The *diametral circle* of a line segment is the circle centered at the midpoint of the segment that passes through both endpoints of the segment—more concicely, the smallest circle containing the segment.

We need the following observation, which goes back to Euclid. For any three points p, q, r on a circle with center c, the angle $\angle pcr$ is exactly twice the angle $\angle pqr$. See the figure on the left below.





This observation implies two useful facts about acute triangles:

- For any acute triangle $\triangle$, each vertex of $\triangle$ is outside the diametral circle of the opposite edge of $\triangle$.
- The circumcircle of any acute triangle $\triangle$ lies inside the union of $\triangle$ and the interiors of the diametral circles of the edges of $\triangle$. (See the right figure above.)

These two facts imply that if T is an acute triangulation of P, the diametral circle of every edge of T is empty, and thus the circumcircle of every triangle in T is empty. ■

Rubric: 5 points

(b) For any triangle $\triangle$ with vertices pqr, define

$$Vol(\triangle) = area(\triangle) \cdot (\|p\|^2 + \|q\|^2 + \|r\|^2)$$

where $\|(a,b)\|^2 = a^2 + b^2$ is the squared Euclidean norm. For any triangulation T, let $Vol(T) = \sum_{\triangle \in T} Vol(\triangle)$. Prove that among all triangulations of P, the Delaunay triangulation of P minimizes $Vol(T)$. *[Hint: Why is the function called "Vol"?]*

Solution: For any point p in the plane, let $\hat{p} = (a, b, a^2 + b^2)$ denote its lifting to the unit paraboloid; note that $\|p\|^2 = a^2 + b^2$ is the z-coordinate of $\hat{p}$. For any triangle $\triangle$ with vertices pqr, let $\hat{\triangle}$ denote the planar triangle with vertices $\hat{p}, \hat{q}, \hat{r}$. Finally, for any triangulation $T = \{\triangle_1, \triangle_2, \ldots\}$ of P, let $\hat{T}$ denote the lifted triangulation $\hat{T} = \{\hat{\triangle}_1, \hat{\triangle}_2, \ldots\}$.

Now consider the vertical triangular prism with bottom facet $\triangle$ and top facet $\hat{\triangle}$; the volume of this prism is equal to

$$area(\triangle) \cdot \frac{\hat{p}.z + \hat{q}.z + \hat{r}.z}{3} = \frac{Vol(\triangle)}{3}.$$

Thus, $Vol(T)$ is exactly three times the volume between the lifted triangulation $\hat{T}$ and the plane $z = 0$.

Now let T be a non-Delaunay triangulation of P; we need to argue that there is another triangulation T' wuch that $Vol(T') < Vol(T)$. Because T is not globally Delaunay, there must be four points $p, q, r, s \in P$, such that pqr and prs are triangles in T whose common edge pr is not locally Delaunay. Let T' be the triangulation obtained by flipping pr, replacing it with qs. Because pr is not locally Delaunay, this flip lowers the lifted triangulation; specifically, we have

$$\frac{Vol(T')}{3} = \frac{Vol(T)}{3} - volume(\hat{p}\hat{q}\hat{r}\hat{s}) < \frac{Vol(T)}{3}.$$

■

Rubric: 5 points

*(c) For any triangle $\triangle$ with side lengths a, b, and c, define

$$\Phi(\triangle) = area(\triangle) \cdot (a^2 + b^2 + c^2).$$

For any triangulation T, let $\Phi(T) = \sum_{\triangle \in T} \Phi(\triangle)$. Prove that among all triangulations of P, the Delaunay triangulation of P minimizes $\Phi(T)$.

Solution (Rajan[a]): Let $\Psi(\triangle)$ denote the volume below the lifted triangle $\hat{\triangle}$ *and above the paraboloid* $z = x^2 + y^2$. To solve the problem, it suffices to prove two claims:

- Among all triangulations of P, the Delaunay triangulation minimizes the function $\Psi(T) = \sum_{\triangle \in T} \Psi(\triangle)$.
- $\Psi(\triangle) = \Phi(\triangle)/12$

The first claim follows from the same argument as part (b). Fix any non-Delaunay triangulation T, and let T' be the triangulation obtained from T by flipping one locally non-Delaunay edge. Then we have

$$\Psi(T') \;=\; \Psi(T) - volume(\hat{p}\hat{q}\hat{r}\hat{s}) \;<\; \Psi(T),$$

where p, q, r, s are the vertices of the flipped quadrilateral.

The second claim requires a bit of careful integral calculus.[b] For any point $p = (x, y) \in \triangle$, let $\psi(x, y)$ denote the vertical distance from the corresponding point on $\hat{\triangle}$ to the point $\hat{p} = (x, y, x^2 + y^2)$. Then $\Psi(\triangle) = \iint_{\triangle} \psi(x, y)\, dx\, dy$.

The function ψ is easier to integrate over $\triangle$ if we express it in terms of *barycentric coordinates*. Suppose p, q, r are the vertices of $\triangle$ in counterclockwise order. Then each point in $\triangle$ can be written as a weighted average $\alpha p + \beta q + \gamma r$ for unique real numbers $\alpha, \beta, \gamma > 0$ such that $\alpha + \beta + \gamma = 1$. Now we can define

$$\psi(\alpha p + \beta q + \gamma r) \;:=\; \underbrace{\alpha\|p\|^2 + \beta\|q\|^2 + \gamma\|r\|^2}_{\text{height of point on lifted triangle}} \;-\; \underbrace{\|\alpha p + \beta q + \gamma r\|^2}_{\text{height of point on paraboloid}}$$

Using the identity $\|v\|^2 = (v \cdot v)$, we can expand this definition into a weighted sum of dot products as follows:

$$\begin{aligned}
&\psi(\alpha p + \beta q + \gamma r) \\
&\quad = \alpha(p \cdot p) + \beta(q \cdot q) + \gamma(r \cdot r) - (\alpha p + \beta q + \gamma r) \cdot (\alpha p + \beta q + \gamma r) \\
&\quad = (\alpha - \alpha^2)(p \cdot p) + (\beta - \beta^2)(q \cdot q) + (\gamma - \gamma^2)(r \cdot r) \\
&\qquad - \; 2\alpha\beta(p \cdot q) - 2\alpha\gamma(p \cdot r) - 2\beta\gamma(q \cdot r)
\end{aligned}$$

We can integrate the function ψ over $\triangle$ by computing

$$\iint_{\triangle} \psi(x, y)\, dx\, dy \;=\; |\det(J)| \int_{\alpha=0}^{1} \int_{\beta=0}^{1-\alpha} \psi(\alpha p + \beta q + (1 - \alpha - \beta) r)\, d\beta\, d\alpha,$$

where J is the Jacobian of the affine transformation from barycentric coordinates (α, β) to native coordinates (x, y). (Chain rule FTW!) The Jacobian determinant

is equal to the 3×3 orientation determinant for the triple p, q, r, which implies $|\det(J)| = 2 \cdot area(\triangle)$. We can similarly integrate over $\triangle$ using any other ordered pair of barycentric coordinates.

We compute the latter integral by separately integrating the coefficients of six dot products, which helpfully fall into two equivalence classes.

$$\iint_{\triangle} (\alpha - \alpha^2) = \int_{\alpha=0}^{1} \left(\int_{\beta=0}^{1-\alpha} \alpha(1-\alpha)\, d\beta \right) d\alpha = \int_{0}^{1} \alpha(1-\alpha)^2\, d\alpha = \frac{1}{12}$$

$$\iint_{\triangle} 2\alpha\beta = \int_{\alpha=0}^{1} \left(\int_{\beta=0}^{1-\alpha} 2\alpha\beta\, d\beta \right) d\alpha \quad = \int_{0}^{1} \alpha(1-\alpha)^2\, d\alpha = \frac{1}{12}$$

Identical calculations imply $\iint_{\triangle} (\beta - \beta^2) = \iint_{\triangle} (\gamma - \gamma^2) = \iint_{\triangle} 2\alpha\gamma = \iint_{\triangle} 2\beta\gamma = \frac{1}{12}$. Finally, putting all the pieces together, we conclude

$$\begin{aligned}
\iint_{\triangle} \psi &= 2\,area(\triangle) \cdot \left(\frac{1}{12}(p \cdot p) + \frac{1}{12}(q \cdot q) + \frac{1}{12}(r \cdot r) \right. \\
&\qquad \left. - \frac{1}{12}(p \cdot q) - \frac{1}{12}(p \cdot r) - \frac{1}{12}(q \cdot r) \right) \\
&= \frac{area(\triangle)}{12} \cdot (2(p \cdot p) + 2(q \cdot q) + 2(r \cdot r) - 2(p \cdot q) - 2(p \cdot r) - 2(q \cdot r)) \\
&= \frac{area(\triangle)}{12} \cdot ((p-q) \cdot (p-q) \;+\; (p-r) \cdot (p-r) \;+\; (q-r) \cdot (q-r)) \\
&= \frac{1}{12} \cdot area(\triangle) \cdot \left(a^2 + b^2 + c^2 \right)
\end{aligned}$$

■

[a] V. T. Rajan. Optimality of the Delaunay triangulation in $\mathbb{R}^d$. *Discrete Comput. Geom.* 12:189–202:1994.

[b] What? You thought integrals were only for *real* engineers? Barycentric coordinates were first proposed by August Möbius in 1827; integration via barycentric coordinates has been a standard component of finite-element methods since at least the 1960s.

Rubric: 5 points (extra credit).

I would dearly love to see a more intuitive geometric explanation that avoids integration. Unfortunately, the only proof I've been able to find anywhere is in Rajan's original paper, where he actually proves a more general optimality theorem for Delaunay triangulations *in arbitrary dimensions*. Every other source I found—papers, surveys, lecture notes, textbooks—just cites Rajan.

3. Suppose you want to get a balloon out of a forest. The forest is modeled as a set P of point sin the plane, each representing the location of a tree; the balloon is modeled as a circle, specified by its center c and radius r. Your goal is to move the balloon arbitrarily far away from the trees by continuous translation, without touching any of the trees.

   (a) Describe an algorithm that either computes an *escape path* for the balloon or correctly reports that no such path exists. An escape path is a path π from the initial center c to some point far outside the convex hull of P, such that every point on π has distance greater than r to every point in P.

Solution: Let me define an escape path to be ***canonical*** if it consists of a line segment from c to some point on the Voronoi diagram of P, followed by a path through the Voronoi diagram. Let $\square$ be a large box containing P; we assume that every escape path ends on the boundary of this bounding box.

For any point $q \in \mathbb{R}^2 \setminus P$, let $\boldsymbol{nnd(q)}$ denote the distance from q to its nearest neighbor(s) in P, and let $\boldsymbol{nn(q)}$ denote any nearest neighbor of q in P. Consider a ray starting at q and pointing directly away from $nn(q)$; let $\boldsymbol{out(q)}$ denote the first point along this ray that is either on the Voronoi diagram of P or the bounding box $\square$. In particular, if q is already on the Voronoi diagram, then $out(q) = q$.

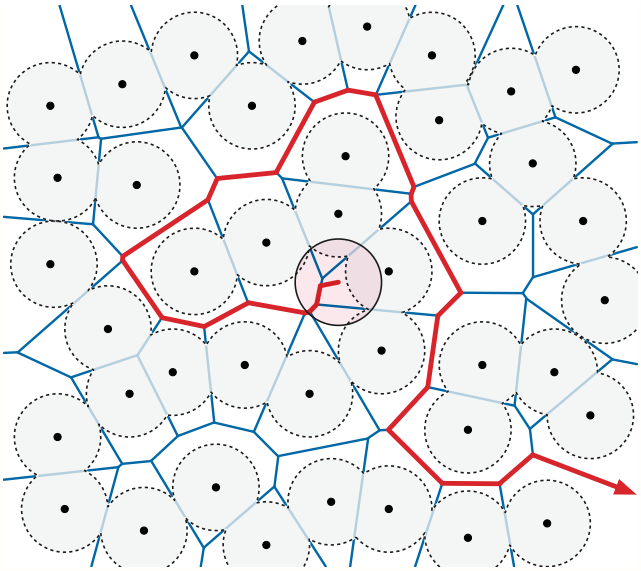
Lemma. *There is an escape path if and only if there is a canonical escape path.*

Proof: Suppose there is an escape path π. Let $p = nn(c)$; because there is an escape path, we must have $|cp| > r$. Let $c' = out(c)$, and observe that $|c'p| \geq |cp| > r$. The path $\pi' : [0,1] \to \mathbb{R}^2$ consisting of the segment $c'c$ followed by the escape path π is an escape path starting at c'.

Now let $\overline{\pi} : [0,1] \to \mathbb{R}^2$ be the path defined by pushing every point in π' away from its nearest neighbor until it hits the Voronoi diagram; more formally, we have $\overline{\pi}(t) := out(\pi'(t))$ for all t. (The path $\overline{\pi}$ is well-defined, because π' never passes through a point in P.) For all t, we have $nnd(\overline{\pi}(t)) \geq nnd(\pi'(t))$, so $\overline{\pi}$ is an escape path starting at c'. Moreover, because $\overline{\pi}$ lies entirely on the Voronoi diagram, $\overline{\pi}$ is a *canonical* escape path. Finally, the segment cc' followed by $\overline{\pi}$ is a canonical escape path starting at c. $\square$

To find a canonical escape path, we start by constructing the Voronoi diagram of P in $O(n \log n)$ time. For each Voronoi edge e, let $e_{\leq r} = \{q \in e \mid nnd(q) \leq r\}$; this subset is either empty or a single line segment. Let G be the union of subsets $e \setminus e_{\leq r}$ over all Voronoi edges e, plus the edges of the bounding box $\square$. G is a planar straight-line graph with at most $O(n)$ vertices and edges.

Finally, we compute the point $c' = out(c)$, verify that $nnd(c') > r$, and then find a path in G from c' to any point on $\square$ via whatever-first search. The entire algorithm runs in $\boldsymbol{O(n \log n)}$ ***time***.



Rubric: 5 points.

(b) Suppose you don't know the size or the initial location of the balloon. Describe an algorithm to preprocess the point P into a data structure that supports queries of the following form: Given a center point c, what is the maximum radius r of a balloon centered at c that can escape the forest?

Solution: Let $nn(q)$, $nnd(q)$, and $out(q)$ be defined as in part (a). Also as in part (a), we surround the points P with a large bounding box $\square$.

We start by constructing the Voronoi diagram of P, clipped to the bounding box $\square$, in in $O(\log n)$ time. We partition each Voronoi region into triangles by connecting each point in P to the vertices of its Voronoi region. Then we build a point-location data structure that allows us to identify the "Voronoi triangle" containing an arbitrary point q in $O(\log n)$ expected time. Building this point-location data structure also takes $O(n \log n)$ expected time. (For example, if we use a randomized incremental construction to build a trapezoidal decomposition of the triangulated Voronoi diagram, we can use the resulting history dag as the point-location structure.)

Next we assign a *capacity* $c(e) = \min_{q \in e} nnd(q)$ to each each Voronoi edge e, equal to the largest radius of a balloon whose center can be safely moved along the entire edge e. For each edge e, we can compute $c(e)$ in $O(1)$ time.

Then we compute the maximum spanning tree T of the Voronoi diagram with respect to these weights, in $O(n \log n)$ time, using any textbook algorithm.[a] For any two Voronoi vertices u and v, the unique path from u to v in T has the largest possible minimum capacity.

Finally, let $maxr(v)$ denote the minimum edge capacity along the unique path from v to a point on the bounding box $\square$. We can compute $maxr(v)$ for every Voronoi vertex v in $O(n)$ time by traversing T. Our lemma in part (a)

implies that a balloon of radius r centered at Voronoi vertex v can escape if and only if $r < maxr(v)$.

Altogether, preprocessing the points takes $O(n \log n)$ expected time.

It remains to describe how to quickly determine the maximum escape radius of a given point using our data structure. Suppose we are given a point c. We can identify the points $p = nn(c)$ and $c' = out(c)$ in $O(\log n)$ expected time using our point-location data structure. If $c = p$, we can immediately return $-\infty$; no balloon centered at a tree can escape the forest. Otherwise, suppose c lies in the "Voronoi triangle" pqr. Suppose the distance to p increases monotonically along the ray $c'q$ (otherwise swap the variable names q and r). Then as our final escape radius, we return $\min\{|cp|, maxr(q)\}$.

Altogether, computing the escape radius of c takes $O(\log n)$ expected time. ■

[a] A careful implementation of Borůvka's algorithm runs in $O(n)$ time, because the Voronoi diagram is a planar graph, but we've already spent $O(n \log n)$ time building the Voronoi diagram, so why bother?

Rubric: 5 points for $O(n \log n)$ preprocessing and $O(n)$ query time.
10 points for $O(n \log n)$ preprocessing and $O(\log n)$ query time.

4. Suppose you are given a set of n halfplanes that define a convex polygon P in the plane. Show that each of the following objects inside P can be found in $O(n)$ expected time using linear programming. ("Axis-aligned" means all are edges either horizontal or vertical.)

   (a) The largest axis-aligned square inside P.

   Solution: Suppose the given halfspaces have the form $a_i x + b_i y \le c_i$. We can compute the largest square inside P in $O(n)$ expected time by solving the following linear program using Seidel's algorithm. The LP has three variables: the coordinates (x, y) of the bottom left corner of the square, and the side length w of the square.

$$\begin{array}{lllll} \text{maximize} & & w & & \\ \text{subject to} & a_i x & + b_i y & \le c_i & \text{for all } i \\ & a_i(x+w) & + b_i y & \le c_i & \text{for all } i \\ & a_i x & + b_i(y+w) & \le c_i & \text{for all } i \\ & a_i(x+w) & + b_i(y+w) & \le c_i & \text{for all } i \end{array}$$

   The four groups of inequalities respectively state that the bottom-left corner, the bottom-right corner, the top-left corner, and the top-right corner lie inside P. ■

   Rubric: 2 points.

   (b) The axis-aligned rectangle inside P with maximum perimeter.

   Solution: Again, suppose the given halfspaces have the form $a_i x + b_i y \le c_i$. We can solve the following linear program in $O(n)$ expected time using Seidel's algorithm. The four variables are the coordinates (x, y) of the bottom left corner, the (horizontal) width w, and the (vertical) height h.

$$\begin{array}{lllll} \text{maximize} & & 2(w+h) & & \\ \text{subject to} & a_i x & + b_i y & \le c_i & \text{for all } i \\ & a_i(x+w) & + b_i y & \le c_i & \text{for all } i \\ & a_i x & + b_i(y+h) & \le c_i & \text{for all } i \\ & a_i(x+w) & + b_i(y+h) & \le c_i & \text{for all } i \end{array}$$

   Again, the four groups of inequalities respectively state that the four corners of the rectangle lie inside P. ■

   Rubric: 2 points.

(c) Two interior-disjoint axis-aligned squares inside P with maximum total perimeter.

Solution: Again, suppose the given halfspaces have the form $a_i x + b_i y \le c_i$. Any pair of interior-disjoint squares can either be separated by a horizontal line or by a vertical line. We consider each of these cases separately; for each case, we can find the optimal squares using Seidel's algorithm in $O(n)$ expected time.

The following linear program describes the minimum-perimeter pair of squares in P separated by a vertical line. The LP has six variables: the bottom-left coordinates (x, y) and width w of the left square, and the bottom-left coordinates (x', y') and width w' of the right square.

$$\begin{array}{lllll}
\text{maximize} & & 4(w+w') & & \\
\text{subject to} & & x+w & \le x' & \\
& a_i x & + b_i y & \le c_i & \text{for all } i \\
& a_i(x+w) & + b_i y & \le c_i & \text{for all } i \\
& a_i x & + b_i(y+h) & \le c_i & \text{for all } i \\
& a_i(x+w) & + b_i(y+h) & \le c_i & \text{for all } i \\
& a_i x' & + b_i y' & \le c_i & \text{for all } i \\
& a_i(x'+w') & + b_i y' & \le c_i & \text{for all } i \\
& a_i x' & + b_i(y'+h) & \le c_i & \text{for all } i \\
& a_i(x'+w') & + b_i(y'+h) & \le c_i & \text{for all } i
\end{array}$$

The horizontal case is similar; in fact the only difference is the first constraint, which becomes $y + w \le y'$. ■

Rubric: 3 points.

(d) The largest circle inside P.

Solution: Again, suppose the given halfplanes have the form $a_i x + b_i y \le c_i$. The Euclidean distance from any point (p, q) inside the halfplane $ax + by \le c$ to the boundary line $ax + by = c$ is exactly

$$\frac{ap + bq - c}{\sqrt{a^2 + b^2}}.$$

For each index i, precompute the following values:

$$\hat{a}_i = \frac{a_i}{\sqrt{a_i^2 + b_i^2}} \qquad \hat{b}_i = \frac{b_i}{\sqrt{a_i^2 + b_i^2}} \qquad \hat{c}_i = \frac{c_i}{\sqrt{a_i^2 + b_i^2}}$$

Then the lines $\hat{a}_i x + \hat{b}_i y = \hat{c}_i$ and $a_i x + b_i y = c_i$ are identical, and for any point (p, q), we have $\hat{a}_i p + \hat{b}_i q \le \hat{c}_i$ if and only if $a_i p + b_i q \le c_i$.

The following linear program has three variables: the coordinates (p, q) of

the center of the circle, and the radius r of the circle.

$$\begin{array}{l}\text{maximize} \quad r \\ \text{subject to} \quad \hat{a}_i p + \hat{b}_i q \le \hat{c}_i - r \quad \text{for all } i\end{array}$$

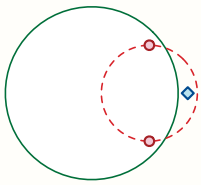
Each inequality specifies that the center point (p, q) not only satisfies the constraint $a_i p + b_i q \le c_i$ but also has Euclidean distance at least r from the line $a_i x + b_i y = c$. Thus, for any feasible solution (p, q, r), the entire circle of radius r centered at (p, q) lies inside each halfplane $a_i x + b_i y \le c_i$, and therefore inside the polygon P. ■

Rubric: 3 points.

5. Suppose you are given a set of n red points and n blue points in the plane.

   (a) Give an example where the smallest circle containing the red points also contains a blue point, but some other circle has all red points inside and all blue points outside.

Solution:



■

   (b) Describe and analyze an algorithm to determine if there is a circle with all red points inside and all blue points outside.

Solution (the hard way): Let R and B respectively denote the given red and blue points. We find the smallest circular disk containing R and excluding B using a natural variant of Welzl's minidisk algorithm.

Let $MinD(R,B,X)$ denote the minimum-radius disk with R in its interior, B in its exterior, and X on its boundary; points in X could be either red or blue. Variants of Welzl's correctness proof (omitted here) imply the following, assuming $MinD(R,B,X)$ exists:

- $MinD(R,B,X)$ is unique.
- $MinD(R,B,X) = MinD(\emptyset,\emptyset,Y)$ for some superset $Y \supseteq X$. Ignoring boundary cases and degeneracies, Y contains either two red points, three red points, or two red points and one blue point.
- For any point $r \in R$:
  - If $r \in MinD(R-r,B,X)$, then $MinD(R,B,X) = MinD(R-r,B,X)$.
  - If $r \notin MinD(R-r,B,X)$, then $MinD(R,B,X) = MinD(R-r,B,X+r)$.
- For any point $b \in B$:
  - If $b \in MinD(R,B-b,X)$, then $MinD(R,B,X) = MinD(R,B-b,X+b)$.
  - If $b \notin MinD(R,B-b,X)$, then $MinD(R,B,X) = MinD(B,B-b,X)$.

Together these observations imply that the following recursive algorithm correctly computes $MinD(R,B,X)$.

```
MinD(R, B, X):
  if |X| > 3
      abort and return Infeasible
  else if R = ∅ and B = ∅
      compute MinD(∅, ∅, X) by brute force
  else
      p ← random point in R ∪ B
      if p ∈ R
          D ← MinD(R − p, B, X)
          if p ∈ D
              return D
          else
              return MinD(R − p, B, X + p)
      else ⟨⟨p ∈ B⟩⟩
          D ← MinD(R, B − p, X)
          if p ∉ D
              return D
          else
              return MinD(R, B − p, X + p)
```

The same backward analysis used for Seidel's LP algorithm and Welzl's minidisk algorithm implies that the algorithm runs in ***O(n) expected time***. ■

Solution (the easy way): This is a linear programming feasibility problem. Suppose we lift the points to the paraboloid $(x, y) \mapsto (x, y, \frac{1}{2}(x^2 + y^2))$. Then our problem is equivalent to asking whether there is a plane in $\mathbb{R}^3$ that lies above all the red points and below all the blue points. The following linear program is feasible if and only if there is such a plane $z = ax + by - c$.

maximize	*whatever*	
subject to	$ax_i + by_i - c \geq \frac{1}{2}(x_i^2 + y_i^2)$	for all red points (x_i, y_i)
	$ax_i + by_i - c \leq \frac{1}{2}(x_i^2 + y_i^2)$	for all blue points (x_i, y_i)

We can determine whether this linear program is feasible in ***O(n) time*** using Seidel's algorithm.

Any feasible point (a, b, c) for this linear program corresponds to the circle $(a - x)^2 + (b - y)^2 = a^2 + b^2 - 2c$. Thus, minimizing c in this linear program is not the same as minimizing the radius, but rather equivalent to *maximizing* the power distance from the origin to the circle. ■

(c) Describe and analyze an algorithm to determine if there are two *disjoint* circles, one containing all the red points but no blue points, and the other containing all the blue points but no red points. *[Hint: This is easier than it looks.]*

Solution: The red and blue points can be separated by two disjoint circles if and only if they can be separated by a straight line! The following LP describes the set of all lines $y = ax - b$ that lie above the red points and below the blue points.

$$\begin{array}{lll} \text{maximize} & \textit{whatever} & \\ \text{subject to} & ax_i - b \geq y_i & \text{for all red points } (x_i, y_i) \\ & ax_i - b \leq y_i & \text{for all blue points } (x_i, y_i) \end{array}$$

A symmetric LP describes the set of all lines above the blue points and below the red points. We return TRUE if either of these LPs is feasible; otherwise, we return FALSE. If we use Seidel's algorithm to test feasibility, the entire algorithm runs in $O(n)$ ***expected time***.

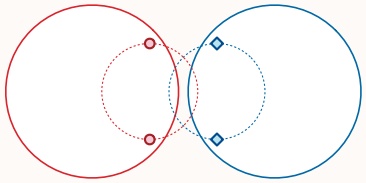
- Suppose there are disjoint circles R and B that respectively contain the red and blue points. Then any line that separates R and B (for example, their power-bisector, or an inner common tangent line) also separates the red and blue points.
- Suppose there is a line ℓ that separates the red and blue points. General position implies that ℓ does not contain any of the points. Let R be the smallest circle containing the red points and tangent to ℓ; symmetrically, let B the be smallest circle containing the blue points an tangent to ℓ. These two circles are separated by ℓ and therefore (by general position) disjoint.

■

Solution: Run the algorithm from part (a) twice—first to find a circle enclosing the red points, and second to find a circle enclosing the blue points—and return TRUE if and only if both calls are successful. The algorithm runs in $O(n)$ ***expected time***.

The algorithm is trivially correct whenever it returns FALSE. Suppose some circle R contains all the red points but no blue points, and some circle B contains all the blue points but no red points. If R and B are disjoint, we're immediately done, so suppose otherwise. Let p and q be the intersection points of the boundary circles ∂R and ∂B, and let ε be the minimum distance from the segment pq to any input point. Increase the radii of R and B while keeping p and q on their boundary, until their intersection has width less than $\varepsilon/10$, and then translate the centers the disks directly away from each other by $\varepsilon/10$. The resulting circles R' and B' are disjoint; R' contains the red points; and B' contains the blue points. Thus, algorithm is also correct whenever it returns TRUE. ■

Non-solution: Compute the smallest disk R containing the red points and the smallest disk R containing the blue points, using Welxl's algorithm. If these disks intersect, return FALSE; otherwise, return TRUE. The entire algorithm runs in ***$O(n)$ expected time***.



The smallest disk containing the red points intersects the smallest disk containing the blue points.

♣