CS623 Winter 2012 \ Assignment #4

Ariel Stolerman

1)

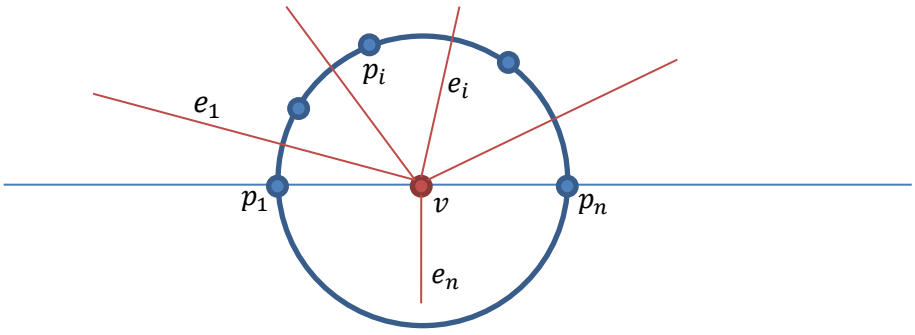
Following is a reduction from the sorting problem to the problem of computing Voroni diagrams, proving a $\Omega(n \lg n)$ bound for the latter:

For a given set $X = \{x_1, \dots, x_n\}$ of n real numbers (assuming all distinct, otherwise we can reduce it to a distinct set in linear time, remembering how many copies we have for each value), do the following:

- Find $x_{min} = \min X$ and $x_{max} = \max X$.
- Construct a set P of points that are the projections of the values in X on the top (or bottom, w.l.o.g) arc of the circle that has $(x_{min}, 0) \rightarrow (x_{max}, 0)$ as diameter. For any point $x_i \in X$ this can be computed by finding the intersections of the line $x = x_i$ (the perpendicular to the x-axis that goes through x_i) with the circle equation (for the circle centered in $\left(\frac{x_{max}-x_{min}}{2}, 0\right)$ with radius $\frac{x_{max}-x_{min}}{2}$), and taking the top one (for the upper arc intersection). Denote these points $P = \{p_1, \dots, p_n\}$. Note that p_1 and p_n are the only ones known to be mapped to their sorted order - x_{min} and x_{max} respectively.
- Find the Voroni diagram of P, which has exactly one vertex and n cells ($V(p_i)$, for all i). Denote that vertex v (which equals to $\left(\frac{x_{max}-x_{min}}{2}, 0\right)$).
- For each $p \in P$, $V(p)$ has 2 edges. Look at the vector $\overline{vp}$, and denote the edge to the left of it (in the direction from v to p) as the left edge of $V(p)$, and the edge to the right similarly.
- Find e_1 the right edge of $V(p_1)$, and continue in clockwise cyclic order over all the edges of v, each phase mapping e_i, the right edge of $V(p_i)$ to the original value from X, namely $p_i.x$.
- Return the resulted sequence of values, which is the sorted sequence of the points in X.

Correctness:

We know that a point q is a vertex of $V(P)$ iff its largest empty circle $C_P(q)$ (a circle centered at q) contains 3 or more sites on its boundary. The construction above makes sure all subsets of 3 or more sites in P have the same unique circle, centered at $v = \left(\frac{x_{max}-x_{min}}{2}, 0\right)$, therefore v is the only vertex in the Voroni diagram of P, and it looks as follows (in red):



Since we can trail on the incident edges of v in cyclic order around it, we can find the order of the corresponding sites (defined as those whose cell is bounded "on the right" by the edge) which is the order of the points P on the boundary of the circle, which corresponds to the sorted sequence of the values in X.

Running time:

Every phase but the Voroni diagram calculation is linear – mapping the values to the points P and trailing over the incident edges of v, returning the corresponding x_i values in sorted order.

Therefore we have mapped the sorting problem to the Voroni diagram computation problem, so the latter is bounded by $\Omega(n \lg n)$, otherwise we would have a sorting algorithm with $o(n \lg n)$ running-time, which cannot exist.

2)

Let P be a set of n points in the plane. Following is a $O(n \lg n)$ algorithm that finds 2 points in P that are closest together:

- Compute the Voroni diagram of P, maintaining a pointer of each half-edge to its corresponding site in P.
- For each edge e in the resulted diagram (excluding any bounding box edges):
  - Get the sites p_i, p_j corresponding to both half edges.
  - Compute the distance $\overline{p_i p_j}$
  - Store the minimum $\overline{p_i p_j}$ encountered thus far, with the corresponding sites p_i, p_j.
- Return the sites p_i, p_j left with the minimal distance between them $\overline{p_i p_j}$.

Correctness:

Using Fortune's algorithm it is easy to maintain the sites corresponding to half-edges in the Voroni diagram. After obtaining the Voroni diagram, it is sufficient to check only distances between two points p_i, p_j that their Voroni cells $V(p_i), V(p_j)$ share an edge in the Voroni diagram.

Suppose for any two sites p_i, p_j, $V(p_i)$ and $V(p_j)$ don't share an edge. In this case there would be a site p_k and a point $q \in \overline{p_i p_j}$ such that $q \in V(p_k)$. By definition $|\overline{qp_k}| \leq |\overline{qp_j}|$, and so:

$$|\overline{p_i p_k}| \underset{\substack{triangle \\ inequality}}{\leq} |\overline{p_i q}| + |\overline{qp_k}| \underset{\substack{minimality \\ of\ |\overline{qp_k}|}}{\leq} |\overline{p_i q}| + |\overline{qp_j}| = |\overline{p_i p_j}|$$

Therefore p_k is a closer site to p_i than p_j is, and $V(p_i), V(p_k)$ share an edge.

After checking all possible pairs p_i, p_j that their Voroni cells share an edge, it follows that we have found the minimal distance pair in the entire set P.

Running time:

Computing the Voroni diagram can be done in $O(n \lg n)$ time using Fortune's algorithm. There are at most $3n - 6$ edges in the Voroni diagram, and for each one we do a constant time check (obtaining the points corresponding to the edge, checking the distance between them and compare to the minimal distance found thus far), so the total time of this phase is linear. The total running time is therefore $O(n \lg n)$, as required.

3)

Suppose there are $O(n)$ intersections between the edges of the Voroni diagram and the farthest site Voroni diagram. Following is a proof that the smallest width annulus can be computed in $O(n \lg n)$ expected time.

We follow the algorithm in the book for computing the smallest width annulus, with specific details on how to compute the intersecting segments of the Voroni diagram (denoted VD) and the farthest site Voroni diagram (denoted FSVD). Given a set of n points P:

- Compute the VD of P and the FSVD of P.
- Collect sets of size 4, whose points define the candidate annuli:
  - For the first and second case (the outer circle contains at least 3 points or the inner circle contains at least 3 points): for each vertex in FSVD, we determine the point $p \in P$ that is closest; for each vertex in VD, we determine the point $p \in P$ that is farthest. The total number of such sets is $O(n)$.
  - For the third case (2 points on each of the inner and outer circle): find all pairs of edges, one from each of the diagrams, which intersect (in the way described later), and consider them for additional candidate sets of points.
- For all candidates choose the one that gives the smallest-width annulus as the solution.

For calculating the intersection of edges in VD with edges in FSVD: run a modified plain sweep algorithm, where all coinciding endpoints of edges from the same diagram (VD or FSVD) are not considered as intersections (this can be done by infinitesimally distant them from one another).

Correctness:

The correctness derives from the correctness of the algorithm described in the book. In addition, the method of determining intersections is correct, since we are only interested in intersections between edges from different diagrams, and any two edges in the same diagram may only intersect in their endpoints – so it is sufficient to ignore those cases only.

Running time:

Computing the VD takes $O(n \lg n)$ by Fortune's algorithm, and computing the FSVD takes $O(n \lg n)$ **expected** time by the algorithm given in the book. Collecting the sets for the first and second cases is $O(n)$. Calculating the intersections for the third case is $O(n \lg n + k)$, where it is given that $k = O(n)$. Therefore the third case takes a total of $O(n \lg n + n) = O(n \lg n)$ time. The last phase of finding the smallest-width annulus of all $O(n)$ candidates is linear (an $O(1)$ check per each set). The total concludes to $O(n \lg n)$ expected time, as required.

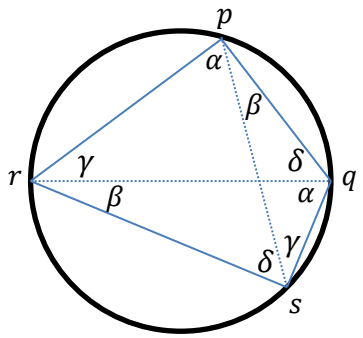
4)

Following is a proof that the smallest angle of any triangulation of a convex polygon whose vertices lie on a circle is the same:

Let P be the set of vertices of the given convex polygon. According to theorem 9.7 (page 198), any triangulation $\mathcal{T}$ of P is a Delaunay triangulation if and only if the circumcircle of any triangle of $\mathcal{T}$ does not contain a point of P in its interior. Since all points of P reside on the same circle, then for any triangle $\Delta p_i p_j p_k \in \mathcal{T}$, none of the other points $p \in P \backslash \{p_i, p_j, p_k\}$ lies in the interior of the circumcircle of $\Delta p_i p_j p_k$. Therefore, any triangulation of such convex polygon is a Delaunay

triangulation. By theorem 9.9 (page 199), any Delaunay triangulation of P maximizes the minimum angle over all triangulations of P. Since all triangulations are Delaunay triangulations, then all have the same unique maximum value of the minimal angle, i.e. the smallest angle of any triangulation of a convex hull whose vertices lie on a circle is the same, as required.

If this statement does not suffice, it can be shown that the minimum angle in any triangulation of a set of cocircular points is the same using Thales's theorem. It is sufficient to show that for any triangulation of a quadrilateral whose vertices are cocircular, the minimum is the same. It is easy since we have only 2 possible triangulations (dotted lines):



The equalities of the angles denoted above is due to Thales's theorem. For instance, $\angle rps = \angle rqs = \alpha$ since they both reside on the circumference of the circle, and on the same side of the line passing through r, s. Denote the minimum angle of the two triangulations:

- $\mu_1 = \min\{\gamma + \beta, \alpha, \delta, \gamma, \beta, \alpha + \delta\} = \min\{\alpha, \beta, \gamma, \delta\}$
- $\mu_2 = \min\{\alpha + \beta, \gamma, \delta, \alpha, \beta, \gamma + \delta\} = \min\{\alpha, \beta, \gamma, \delta\}$

Therefore $\mu_1 = \mu_2$, and so for any triangulation of the quadrilateral $pqrs$, the value of the minimum angle is the same. Therefore the minimum angle in any triangulation of a set of cocircular points is the same, as required.

5)

a.

Following is a proof that the set of edges of a Delaunay triangulation of P contains an EMST for P:

Assume by contradiction that some EMST T for P has an edge (u, v) not in the set of edges of some Delaunay triangulation $\mathcal{T}$. By theorem 9.6 (page 198), two points $p, q \in P$ form an edge of the Delaunay graph of P iff there is a closed disc C that contains p, q on its boundary and does not contain any other point of P. By the assumption it follows that for any disc C with u, v on its boundary there is some $w \in P \backslash \{u, v\}$ such that w is contained in C, specifically the circle where (u, v) is its diameter, in which case it is certain that $|(u, v)| > |(u, w)|, |(w, v)|$. But then we can construct a tree T' that is identical to T but instead of (u, v) it has either (u, w) or (w, v) (depending in which subtree w is of the two subtrees created by removing (u, v) from T). But then T' is smaller than T, in contradiction of T being an EMST. Therefore the set of edges of a Delaunay triangulation of P must contain an EMST for P.

b.

Following is an $O(n \lg n)$ (worst-case) algorithm to compute an EMST for P:

- Compute the Delaunay triangulation (graph) of P from the Voroni diagram of P, calculated using Fortune's algorithm. Do that by having all sites as vertices and crossing an edge between any two sites u, v that their cells $V(u), V(v)$ are adjacent (have a shared edge) in the Voroni diagram.
- Use Prim's algorithm to compute an MST out of the resulted Delaunay graph.

Correctness:

Immediately derives from the correctness Fortune's algorithm, correspondence between the Voroni diagram and the Delaunay triangulation of P, Prim's algorithm for finding MSTs and what is proven in section (a): since the set of edges of a Delaunay triangulation contains an EMST for P, and from the uniqueness of the weight of an MST, it must be that any MST constructed from those edges is an EMST.

Running time:

Calculating the Voroni diagram of P using Fortune's algorithm is $O(n \lg n)$, and calculating the Delaunay triangulation from it is linear (since the set of edges in the Voroni diagram is linear). Prim's MST algorithm is $O\big((n+m) \lg n\big)$, and in this case $m = O(n)$ (linearity of number of edges in a triangulation of the convex hull of P, theorem 9.1, page 193), which means this phase takes $O(n \lg n)$. The total running time is therefore $O(n \lg n)$ expected time.

6)

a.

Following is a proof that $\mathcal{DG}(P)$ contains the Gabriel graph of P:

Let (u, v) be an edge in Gabriel graph of P, then by definition the disc with diameter (u, v) does not contain any other point in P, therefore u, v are on a boundary of some disc that does not contain any other point in P, which means by theorem 9.6 (page 198) that (u, v) is an edge of $\mathcal{DG}(P)$. The Delaunay graph property is simply more general than the Gabriel graph property. Therefore any edge of the Gabriel graph of P is contained in $\mathcal{DG}(P)$, so the later contains the former, as required.

b.

Following is a proof that p and q are adjacent in the Gabriel graph of P iff the Delaunay edge between p and q intersects its dual Voroni edge.

First assume p and q are adjacent in the Gabriel graph, then p, q are the two ends of a diameter of a disc C that contains no other points in P, and C is centered at the midpoint between p, q, denoted o. Since no other point is contained in C, then the closest sites to o are p and q and no other point in P is closer. Therefore an edge of the Voroni diagram passes between the cells $V(p)$ and $V(q)$. Moreover, since o is equidistant from p and q, o resides on that Voroni diagram edge. Therefore the Delaunay edge between p and q intersects its dual Voroni edge – the Voroni edge between the cells $V(p)$ and $V(q)$.

Now assume the Delaunay edge between p and q intersects its dual Voroni edge, then $V(p)$ and $V(q)$ must be adjacent in the Voroni diagram. Moreover, the point of intersection o lies on that Voroni edge and is equidistant from p and q. Now observe the disc C centered at o that has p, q on its boundary: suppose some point v lies inside C, then o would have been closer to v than to p or q, and would not lie on the Voroni edge between $V(p)$ and $V(q)$ – in contradiction. Therefore the edge (p, q) is the diameter of that disc C that has no other points of P in its interior, thus p, q are adjacent in the Gabriel graph of P.

c.

We use what is proven in the previous sections to come up with an $O(n \lg n)$ algorithm for computing the Gabriel graph of a set P of n points:

- Construct the Voroni diagram of P.
- For any edge e of the Voroni diagram between two cells $V(p)$ and $V(q)$ ($p, q \in P$), connect the edge (p, q) and check if it intersects e. If so, add it to the result set.
- Return the graph corresponding to the constructed result set.

Correctness:

The correctness derives immediately from section c.

Running time:

Computing the Voroni diagram takes $O(n \lg n)$ time using Fortune's algorithm. The number of edges in that diagram is $O(n)$, and for each edge we do a constant amount of work – finding p and q from the faces adjacent to the edge e we are checking, connecting segment $\overline{pq}$ and check if it intersects e – the total running time of this phase is then linear. The total running time is therefore $O(n \lg n)$.