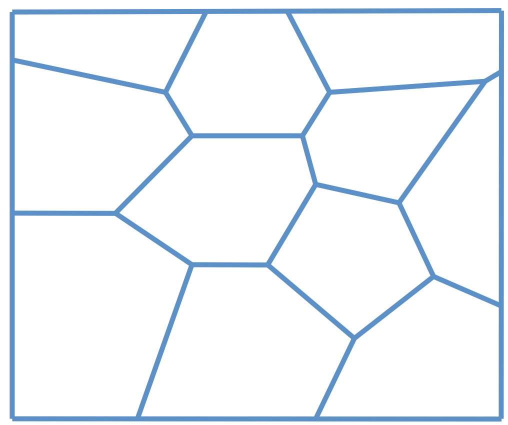
Voronoi Applications

Nearest Neighbour Queries

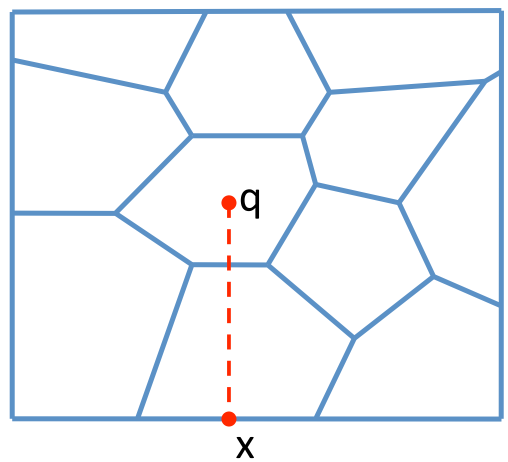
- Given a fixed set of points P in the plane, construct the Voronoi diagram in O(nlogn) time.
- Now for a query point q, finding the nearest neighbour of q reduces to finding in which Voronoi region it falls
- The problem of locating a point inside a partition is called point location

Point Location in a Planar Subdivision



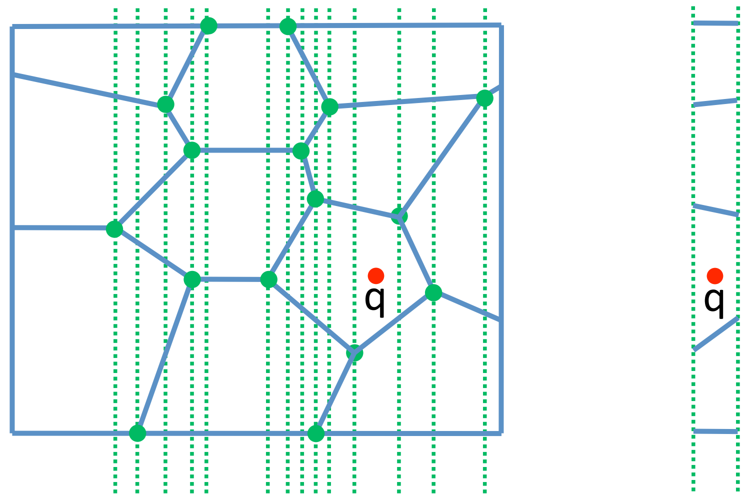
- **Input:** A planar subdivision S with n edges
- **Aim:** Preprocess S such that point location queries can be answered quickly
- **Query:** Given a query point, report the face of S that contains q

Point Location in a Planar Subdivision



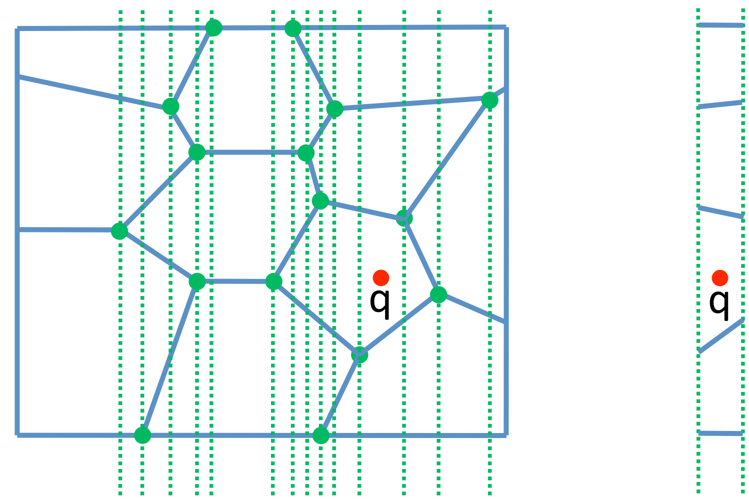
- **Brute Approach:**
  - If S is stored in DCEL, the query can be answered in O(n) time. Just visit each face and determine if q is contained in it. (How?)
    - Take any ray from q. Find x, the point of intersection of the ray with the boundary
    - Walk the faces intersected by qx

Point Location in a Planar Subdivision



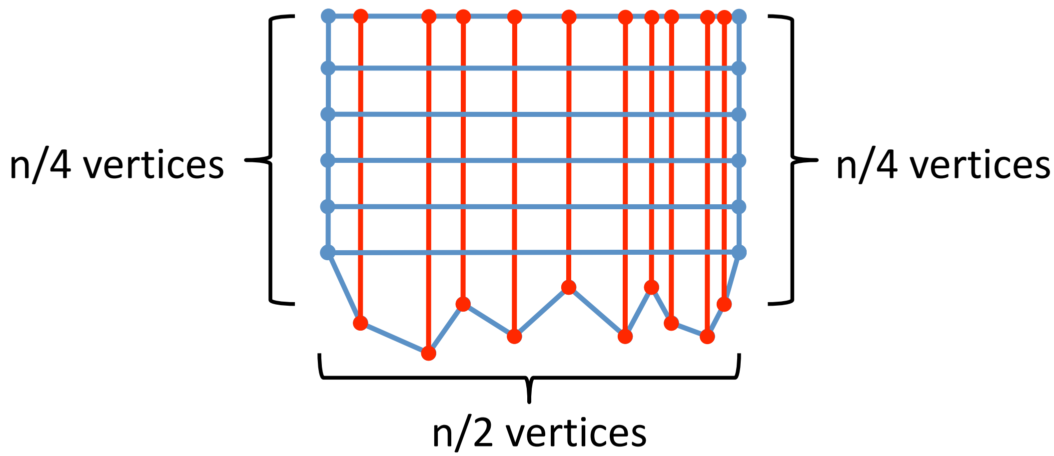
- **Slab Method:**
  - Idea: Draw a vertical line through every vertex
  - This partitions the plane into slabs
  - Finding the right vertical slab can be done in O(logn) time
  - How do we answer a query within a slab?
    - Binary search: O(logn) time
    - We can y-order the edges crossing the slab so that binary search is possible

Point Location in a Planar Subdivision



- **<u>Slab Method:</u> Query**
  - Search for the vertical slab that contains q
    - O(logn) time
  - Search for trapezoid in the vertical slab
    - O(logn) time
  - Total time : O(logn)

Point Location in a Planar Subdivision



- **<u>Slab Method:</u> Storage**
  - O(n) slabs
  - O(n) Trapezoids per slab
  - Number of slabs : $\Omega\ (n^2)$

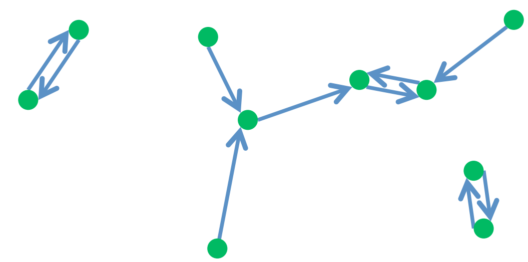
Point Location in a Planar Subdivision

	Query Time	Space
1^{st} try	O(n)	O(n)
2^{nd} try	O(logn)	O(n^2)
Best	O(logn)	O(n) Expected case already discussed; Similar worst-case bound is possible.

All Nearest Neighbours

- Given a set P of n points, determine the nearest neighbour of each point of P

- p -> q means the nearest neighbour of p in P is q

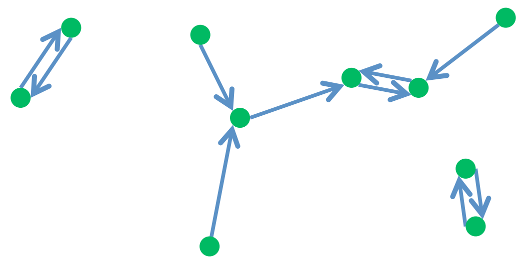


- **<u>Nearest Neighbour Graph of P:</u> NNG(P)**
  - A node is associated with each point
  - There is an edge between two nodes if one of the corresponding points is nearest to the other corresponding poin

All Nearest Neighbours

- **<u>Lemma:</u>** NNG(P) $\subseteq$ DT(P)

- **<u>Proof:</u>** Clearly if q is nearest to p, p and q are directly connected in DT(P)
  - Here, p and q are called Delaunay neighbours



Minimum Spanning Tree

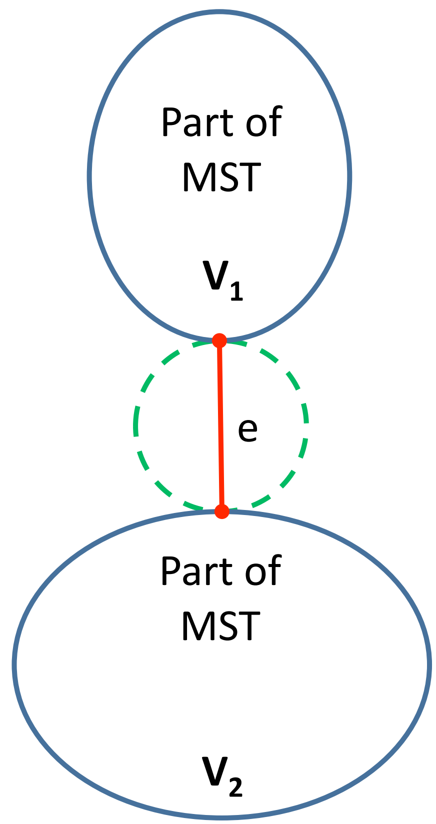
- **<u>Definition</u>**: a <u>spanning tree</u> of a point set P is a tree that connects all the points of P
- **<u>Definition</u>**: <u>Minimum spanning tree (MST)</u> is a spanning tree of P with minimum cost
- **<u>Kruskal's Algorithm of Computing MST of a Graph</u>**

  G = (V,E)
  - Sort all edges of G by length $e_1, \ldots, e_n$
  - Initialize T to the empty set
  - **While** T is not a spanning tree of P **do**
    - **If** $T + e_i$ is a tree **then**

      $T \leftarrow T + e_i$
    - $i \leftarrow i+1$
- Complexity: $O(\,|v|^2 \log|v|\,)$

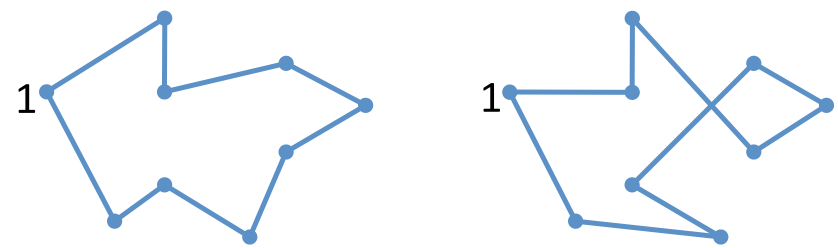
Minimum Spanning Tree

- **<u>Lemma:</u>** MST(P) $\subseteq$ DT(P)
  - The MST edges of P are contained in DT(P)
- **<u>Proof:</u>** Consider an MST edge e
- **<u>Observation:</u>**
  - e is the shortest edge between V_1 and V_2
  - The circle with e as diameter is empty
  - e is a Delaunay edge
- For the MST of points in the plane there are $\binom{n}{2}$ edges. However we can only use DT(P) edges (O(n) in number). Therefore Kruskal's algorithm costs O (nlogn) time



Traveling Salesman Tour

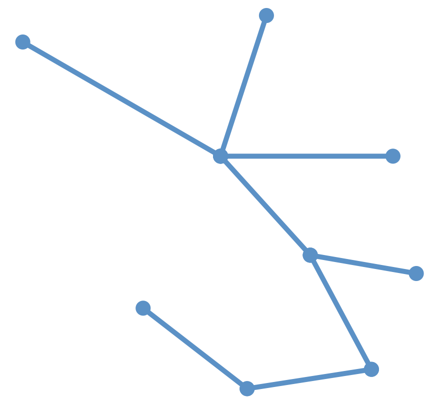
- Find the shortest closed path that visits every point of the set. Such a closed path is called a traveling salesman tour.

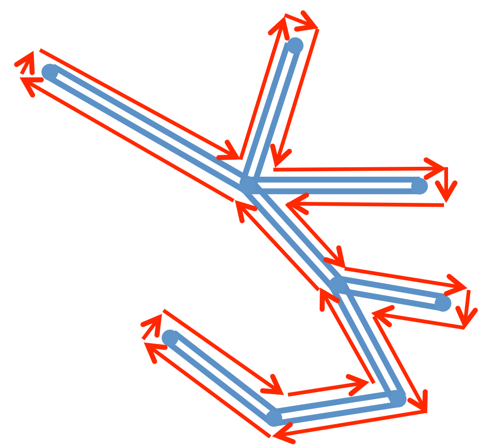


- There are (n-1)! Different tours (starting from 1)
- Hard problem (**NP-hard**)
  - No polynomial time solution is known
- **Approximation algorithm:**
  - An algorithm which computes a solution which is close to optimal

Traveling Salesman Tour

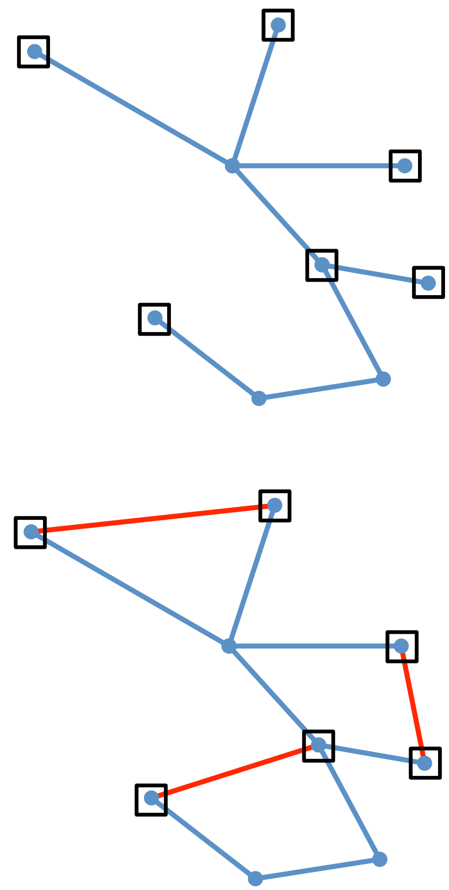
- Consider an MST of P
- Double the edges
  - Each node has degree even
- Construct an Euler tour
- Eulerian tour cost is 2 * MST cost **(1)**
- TSP tour cost is no more than Eulerian tour cost
- TSP tour cost is more than MST cost **(2)**
- Therefore:
  - (1) Euler tour cost = (2*MST)
  - (2) Euler tour cost < (2*TSP)
- Euler tour is 2-approximations of TSP tour





Traveling Salesman Tour

- **3/2-Approximation Tour**
  - Start from MST
  - Identify the odd-degree vertices
    - There will always be an even number of such vertices
  - Pair up the odd degree vertices with minimum cost
  - Every node has even degree now
  - Construct an Euler tour
  - The cost of Euler tour
    - = Cost of MST + Cost of edges between paired up vertices
- Euler tour cost < (TSP + ½ TSP) = 3/2 TSP



Medial Axis

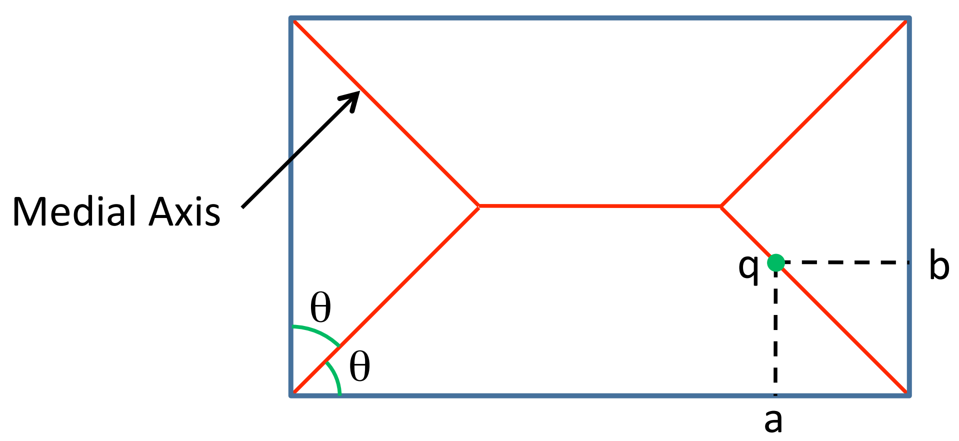
- A generalization of Voronoi diagram. The set of sites is an infinite set of points, in particular the continuous boundary of a polygon

- **Voronoi diagram:** set of points whose nearest neighbour is not unique

- **Medial Axis of a Polygon P:**
  - Set of points inside P that have more than one closest point among the points of ∂P, the boundary of P

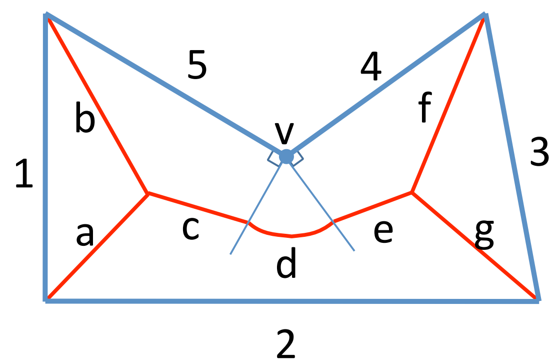
Medial Axis

- **Medial Axis of a Rectangle**
  - The nearest neighbour of any point on the medial axis is not unique.
  - For point q, there are two boundary edges



Medial Axis

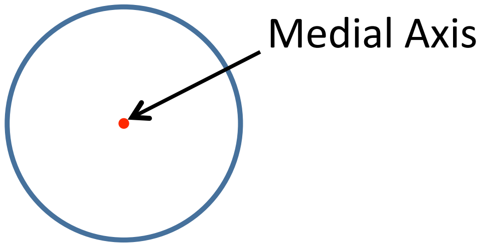
- **Other examples:**

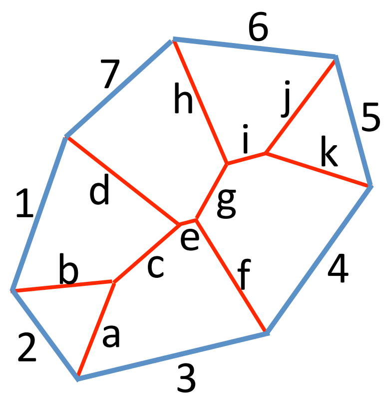


- a: angle bisector of 1 and 2
- b: angle bisector of 1 and 5
- c: angle bisector of 2 and 5
- d: equidistant to vertex v and edge 2
- e: angle bisector of 2 and 4
- f: angle bisector of 3 and 4
- g: angle bisector of 2 and 3

Medial Axis

- **Other examples:**

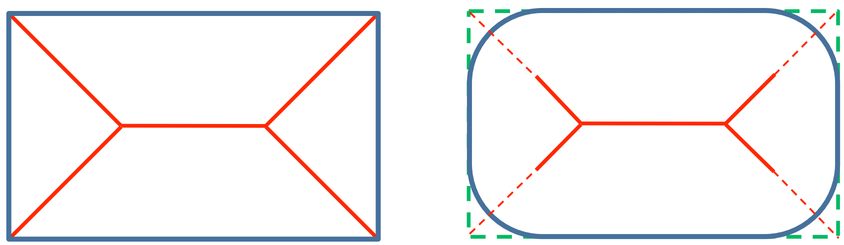




- c: nearest to 1 and 3
- e: nearest to 3 and 7
- g: nearest to 4 and 7
- i: nearest to 4 and 6
- …

Medial Axis

- Blum 1967: Introduced medial axis
- Medial axis is used to represent an object
- An object can be smoothed by smoothing the medial axis



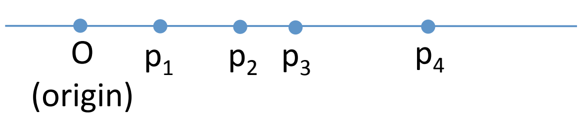
- Medial axis of a polygon can be computed in O(nlogn) time
  - Lee 1982
- For a convex polygon, O(n) time algorithm is known
  - Aggarwal, Guibas, Saxe, and Shor 1989

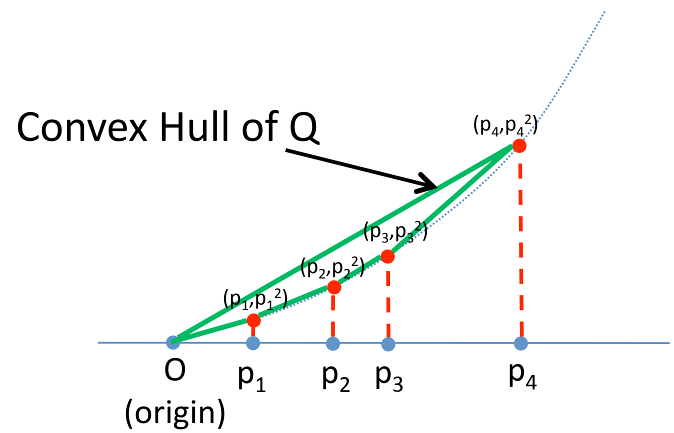
Voronoi Diagram's Connection to Convex Hulls

- Initial idea came from Brown (1979)
- The Voronoi diagram of a point set in the plane can be computed by computing the convex hull of a transformed point set in 3-dimensions
- Voronoi diagrams in d-dimensions are equivalent to convex hulls in d+1 dimensions

One-Dimensional Delaunay Triangulation

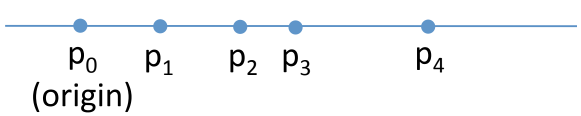
- Let $P = \{p_1, p_2, \ldots, p_n\}$
- We transform pi on a line to a point $q_i = (p_i, p_i^2)$
- Compute the convex hull of the points $Q = \{q_1, q_2, \ldots, q_n\}$
- The projection of the lower convex hull of Q (that is visible from y= -∞) on to the x-axis realizes the Delaunay triangulation of P

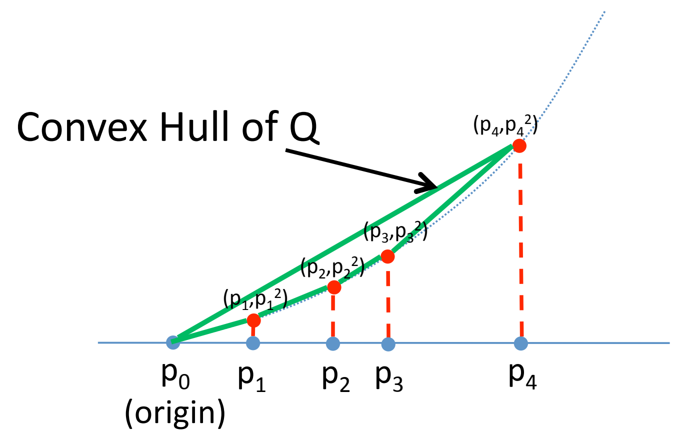




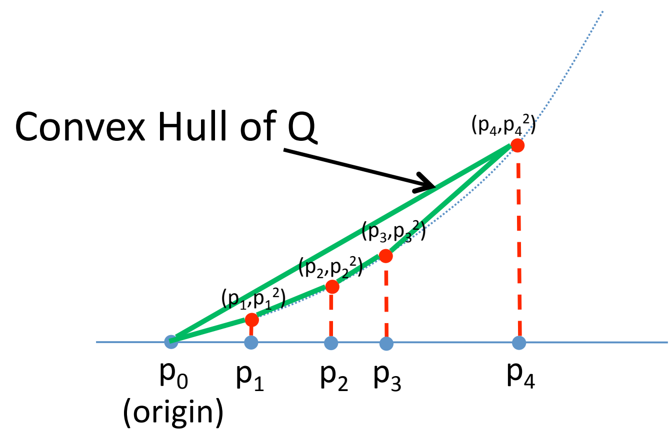
One-Dimensional Delaunay Triangulation

- **Observation:**
  - Each $q_i = (p_i, p_i^2)$ is an extreme point of Q
  - The projection of the lower convex hull of Q onto x is the Delaunay triangulation of P
  - **Note:** The edge q_0q_4 is an edge of the convex hull of Q and its projection onto x contains all the points of P



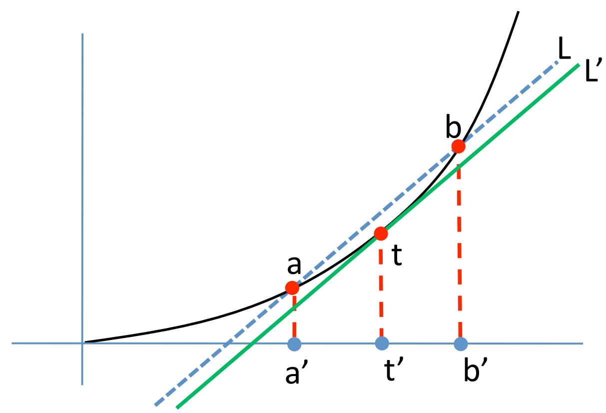


One-Dimensional Delaunay Triangulation



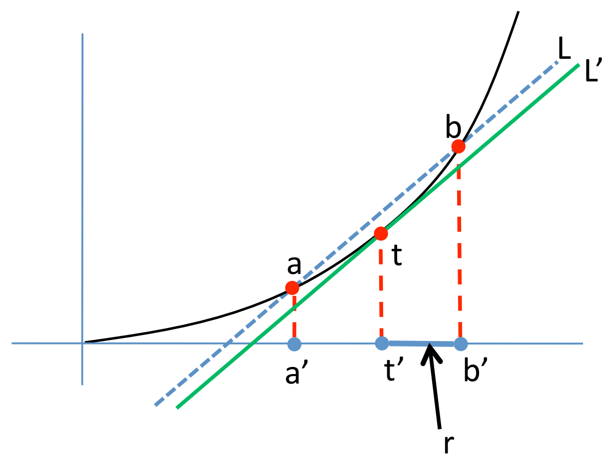
- **<u>Observation:</u>**
  - Each $q_i = (p_i, p_i^2)$ is an extreme point of Q
  - The projection of the lower convex hull of Q onto x is the Delaunay triangulation of P
  - **Note:** The edge $q_0 q_4$ is an edge of the convex hull of Q and its projection onto x contains all the points of P

One-Dimensional Delaunay Triangulation



- **<u>Observation:</u>**
  - Suppose a and b lie on $y = x^2$
  - Let a’ and b’ be the projections of a and b onto the x axis
  - Let t be the tangent point of the parabola where the line through a and b is translated (staying parallel)
  - Can show that the projection of t onto x-axis is the middle point of ab

One-Dimensional Delaunay Triangulation



- **Observation:**
  - Equation of L': $y - t'^2 = 2t'(x-t')$ i.e. $y = 2t'x - t'^2$
  - Translate L' vertically by r^2 to obtain L
  - Equation of L : y = 2t'x – t'2+r2
  - Intersection of L with y = x2 (a and b) can be obtained by solveing
    - $Y = x^2 = 2t'x - t'^2 + r^2$
    - ➔ $x = t' \pm r$

Two-Dimensional Delaunay Triangulation

- Given $P = \{p_1, p_2, \ldots, p_n\}$; $p_i = (x_i, y_i)$
- Paraboloid $z = x^2+y^2$
  - $p_i = (x_i, y_i) \rightarrow (x_i, y_i, x_i^2+y_i^2) = q_i$
- Let $Q = \{q_1, \ldots, q_n\}$ be the transformed points in 3-dimension
- Construct the convex hull of Q
- Project the lower convex hull of Q (visible from $z = -\infty$) onto the xy-plane
- The projected diagram is the Delaunay triangulation

Two-Dimensional Delaunay Triangulation

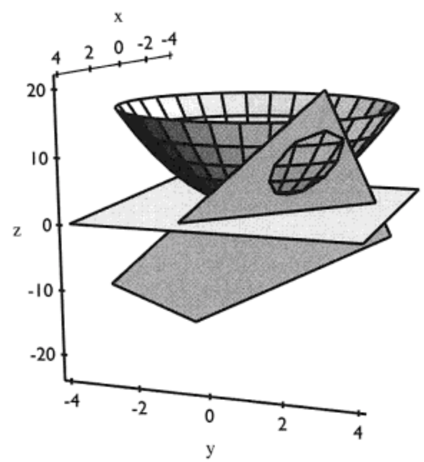


FIGURE 5.27 Plane for $(a, b) = (2, 2)$ and $r = 1$ cutting the paraboloid.

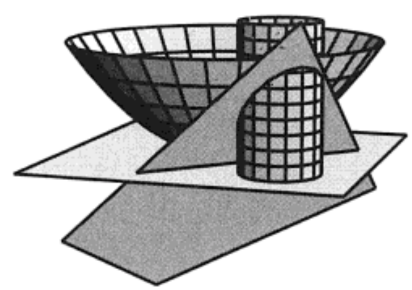


FIGURE 5.28 The curve of intersection in Figure 5.27 projects to a circle of radius 1 in the xy-plane.

Two-Dimensional Delaunay Triangulation

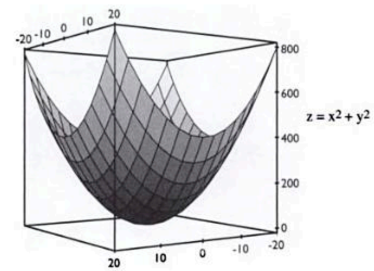


FIGURE 5.24 The paraboloid up to which the sites are projected.

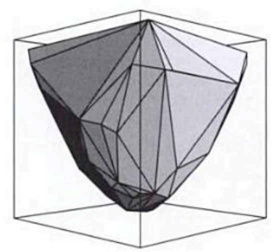


FIGURE 5.25 The convex hull of 65 points projected up to the paraboloid.

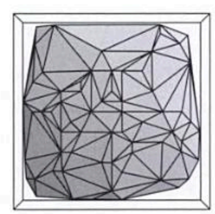


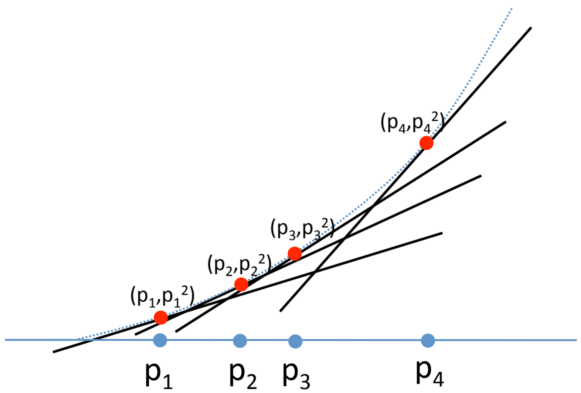
FIGURE 5.26 The paraboloid hull viewed from $z \approx -\infty$.

Three-Dimensional Delaunay Triangulation

- Consider 3-dimensional convex hull facet f.
- Three points a, b, and c of Q determine the facet f
- All the remaining points of Q lie on one side of the facet f.
- Let L be the plane that contains f
- **Observation:**
  - L intersects $z = x^2+y^2$ into an ellipse
    - Equation of L : $\alpha x + \beta y + \gamma z = 1$
    - Equation of paraboloid : $z = x^2+y^2$
  - The projection of an ellipse onto xy-plane is a circle
  - The projection of a, b, and c (say a', b', and c') lie on the circle
  - The circle is empty
  - The triangle $\Delta a'b'c'$ is a Delaunay triangle

Connection to Arrangements

- **One Dimensional Voronoi Diagram**
  - $P = \{p_1, \ldots, p_n\}$, $p_i = x_i$
  - Suppose $q_i = (x_i, x_i^2)$, p_i's projection onto $y = x^2$
  - Consider the tangent line L_i through pi and tangent to $y=x^2$
  - Projecting the visible part of the arrangment from $y = +\infty$ to the x-axis realizes the Voronoi diagram of P



Connection to Arrangements

- **Two Dimensional Voronoi Diagram**
  - $P = \{p_1, \ldots, p_n\}$, $p_i = (x_i, y_i)$
  - $Q = \{q_1, \ldots, q_n\}$, $q_i = (x_i, y_i, x_i^2+y_i^2)$
  - Let l_i be the plane tangent to the paraboloid at q_i
  - Compute the arrangements of L_i, i=1, 2, …, n that are visible from $z = +\infty$
  - The projections of A onto the xy-plane determines the Voronoi Diagram of P