Voronoi Diagrams

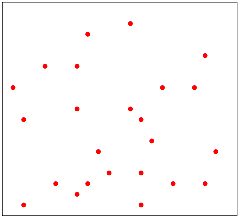
Aritra Banik*
National Institute of Science Education and Research

Summer School on Graph Theory and Graph Algorithm at NIT Calicut

*Slide ideas borrowed from Marc van Kreveld and Swami Sarvottamananda

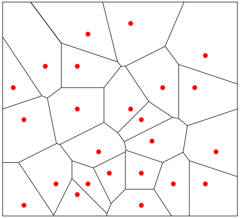
Outline

Motivation



- Given is a set of post offices modeled as a set of points in the plane.
- Each point in the plane has a post office which is closest to the point.
- Each post office has a service region consisting of the points in the plane from which it is closest.
- We would like to demarcate the service regions of the post-offices

Motivation



- Given is a set of post offices modeled as a set of points in the plane.
- Each point in the plane has a post office which is closest to the point.
- Each post office has a service region consisting of the points in the plane from which it is closest.
- We would like to demarcate the service regions of the post-offices

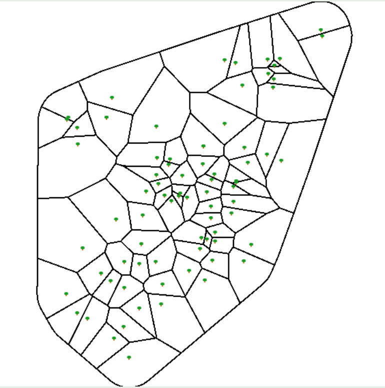
Applications in Cross-disciplines

This concept has independently emerged, and proven useful, in various fields of sciences

- *Medial axis transform* in biology and physiology,
- *Wigner-Seitz zones* in chemistry and physics,
- *Domains of action* in crystallography,
- and *Thiessen polygons* in meteorology and geography.

Other Direct Applications—I

Example (Jurisdiction of Schools in Boston)



A relevant modern day use.

Other Direct Applications—II

Example (1854 Cholera Epidemic in London)

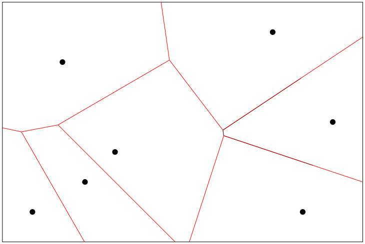
A particularly notable use of a Voronoi diagram was the analysis of the 1854 cholera epidemic in London, in which physician John Snow determined a strong correlation of deaths with proximity to a particular (and infected) water pump on Broad Street.

What are Voronoi Diagrams (very formally)

Definition

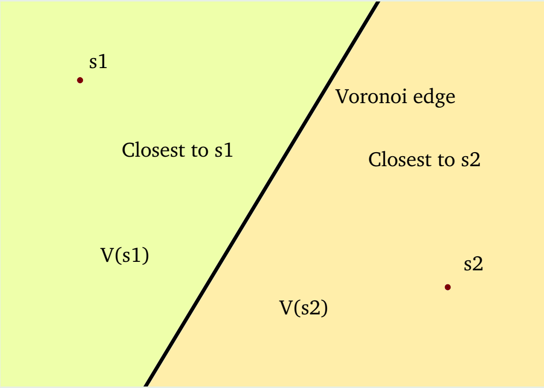
- Let S be a set of n distinct points, s_i, $\forall i \in n$, called *sites* in the plane
- The Voronoi diagram of S is the subdivision of the plane into n cells, $V(s_i)$, one for each site s_i,
- A point q lies in $V(s_i)$ iff $||q - s_i|| < ||q - s_j||$, for each $s_j \in S, i \neq j$

Simply put, $V(s_i)$ IS THE SET OF POINTS WHOSE NEAREST POINT IS s_i.



Computing Voronoi Diagram

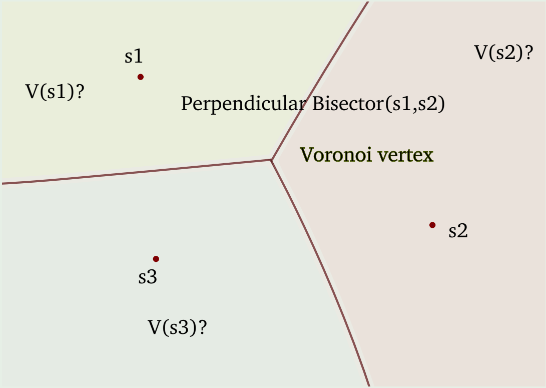
Example (Voronoi Diagram of 2 points)



It's perpendicular bisector

Computing Voronoi Diagram of 3 points

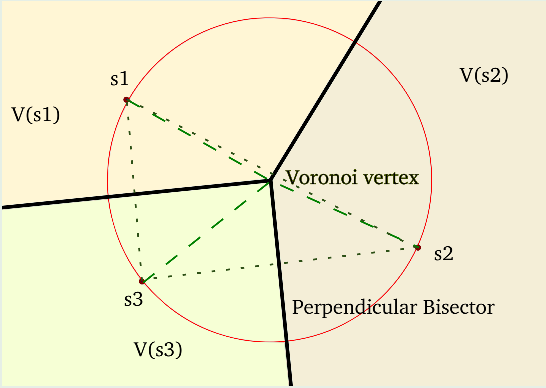
Example (What is Voronoi Diagram of 3 points)



Partition the plane in three sets. Points nearer to s_1, s_2 and s_3

Properties of Voronoi Diagram

Example (Voronoi Diagram of 3 points)



Bisectors intersect at circumcentre

Properties

- Voronoi edges are in part perpendicular bisectors.
- Voronoi vertices are equidistant to three points (circumcentre).
- Not immediately obvious, the circumcircle must be empty (Let us see this again).

Properties

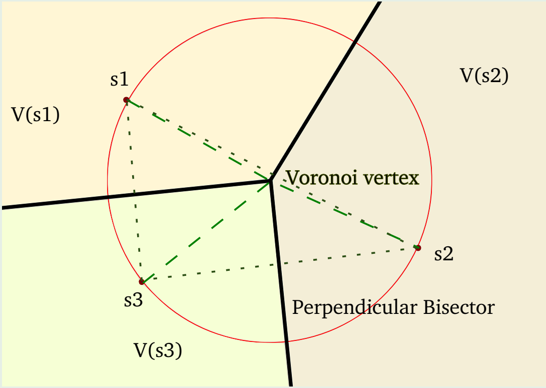
- Voronoi edges are in part perpendicular bisectors.
- Voronoi vertices are equidistant to three points (circumcentre).
- Not immediately obvious, the circumcircle must be empty (Let us see this again).

Properties

- Voronoi edges are in part perpendicular bisectors.
- Voronoi vertices are equidistant to three points (circumcentre).
- Not immediately obvious, the circumcircle must be empty (Let us see this again).

Properties of Voronoi Diagram

Example (There can not be a point nearer than the three points)



Circumcircle is empty and voronoi vertex in centre of circumcircle.
This motivates an algorithm.

Outline

Straight-forward Algorithm

Frontal Attack to the Problem

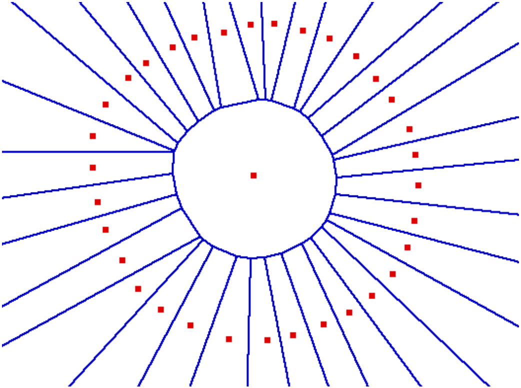
- Compute $\binom{n}{3}$ circumcentres
- Remove non-empty circumcircles
- Intelligently join circumcentres by bisectors (whenever two points are common)
- Voila, you get the Voronoi Diagram

Bad algorithm, $O(n^4)$ time complexity

Space Complexity of Voronoi Diagrams

Before giving good algorithms, we ask what is the size of output. Why?
Because if output is big we do not hope to improve.

Bad Case for Space Complexity



A cell so big that it has $n - 1$ edges.

Space Complexity of Voronoi Diagrams

Theorem

The Voronoi diagram on n sites in the plane has at most $2n - 5$ Voronoi vertices and at most $3n - 6$ Voronoi edges (including lines and half-lines).

PROOF: If the sites are colinear, then it is trivial, otherwise, we will use Euler's formula for planar graphs

Theorem

A connected planar graph with n_v vertices, n_e edges, and n_f faces satisfies: $n_v - n_e + n_f = 2$

However, a Voronoi diagram is not a proper graph

Space Complexity of Voronoi Diagrams

Theorem

The Voronoi diagram on n sites in the plane has at most $2n - 5$ Voronoi vertices and at most $3n - 6$ Voronoi edges (including lines and half-lines).

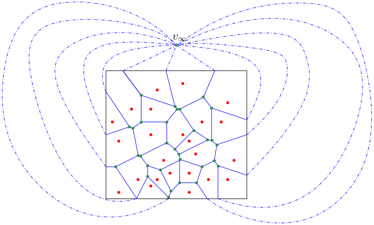
PROOF: If the sites are colinear, then it is trivial, otherwise, we will use Euler's formula for planar graphs

Theorem

A connected planar graph with n_v vertices, n_e edges, and n_f faces satisfies: $n_v - n_e + n_f = 2$

However, a Voronoi diagram is not a proper graph

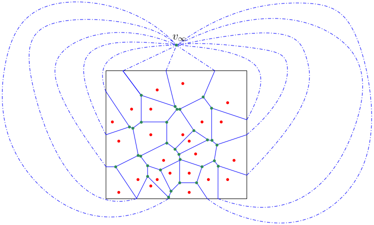
Space Complexity of Voronoi Diagrams



- We make it proper by connecting all half-infinite edges to a new vertex v_∞
- Every edge is incident to exactly 2 vertices, and every vertex is incident to at least 3 edges.
- Sum-of-degree-of-all-vertices $= 2n_e$
- Sum-of-degree-of-all-vertices $\geq 3n_v$
- $2n_e \geq 3n_v$

We have $n_v - 1 + n_e - n = 2$, Thus $n_v \leq 2n - 5$ and $n_e \leq 3n - 6$

Space Complexity of Voronoi Diagrams

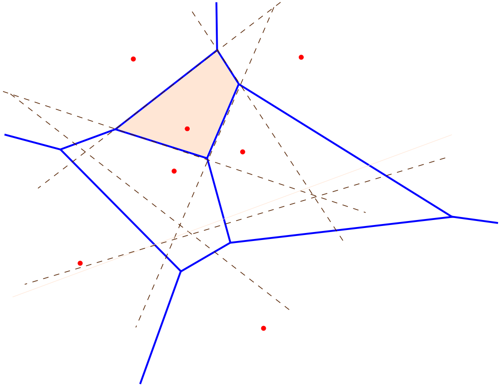


- We make it proper by connecting all half-infinite edges to a new vertex v_∞
- Every edge is incident to exactly 2 vertices, and every vertex is incident to at least 3 edges.
- Sum-of-degree-of-all-vertices $= 2n_e$
- Sum-of-degree-of-all-vertices $\geq 3n_v$
- $2n_e \geq 3n_v$

We have $n_v - 1 + n_e - n = 2$, Thus $n_v \leq 2n - 5$ and $n_e \leq 3n - 6$

Another Algorithm

Each Voronoi cell is intersection of $n-1$ half planes



We need to compute intersections efficiently

Another Algorithm—Analysis

- Each $V(s_i)$ is intersection of $n-1$ half planes
- Intersection of n half-planes can be computed in $O(n \log n)$ time
- Voronoi diagram of S can be computed in $O(n^2 \log n)$ time by computing the Voronoi cells one by one.
- A large improvement over previous method

Another Algorithm—Analysis

- Each $V(s_i)$ is intersection of $n - 1$ half planes
- Intersection of n half-planes can be computed in $O(n \log n)$ time
- Voronoi diagram of S can be computed in $O(n^2 \log n)$ time by computing the Voronoi cells one by one.
- A large improvement over previous method

Another Algorithm—Analysis

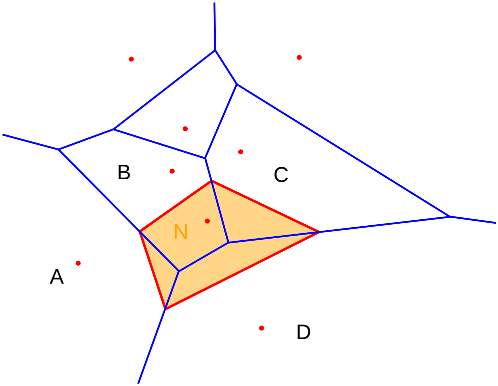
- Each $V(s_i)$ is intersection of $n-1$ half planes
- Intersection of n half-planes can be computed in $O(n \log n)$ time
- Voronoi diagram of S can be computed in $O(n^2 \log n)$ time by computing the Voronoi cells one by one.
- A large improvement over previous method

Another Algorithm—Analysis

- Each $V(s_i)$ is intersection of $n-1$ half planes
- Intersection of n half-planes can be computed in $O(n \log n)$ time
- Voronoi diagram of S can be computed in $O(n^2 \log n)$ time by computing the Voronoi cells one by one.
- A large improvement over previous method

Yet Another Algorithm

Each point is inserted in Voronoi Diagram one by one



Yet Another Algorithm

- Site s_{i+1} is located in Voronoi Diagram of i sites
- Its boundary is calculated
- Voronoi diagram of S can be computed in $O(n^2)$ as $O(i)$ edges might be added in each step
- Further improvement over previous method

Yet Another Algorithm

- Site s_{i+1} is located in Voronoi Diagram of i sites
- Its boundary is calculated
- Voronoi diagram of S can be computed in $O(n^2)$ as $O(i)$ edges might be added in each step
- Further improvement over previous method

Yet Another Algorithm

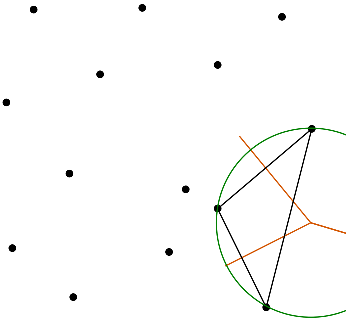
- Site s_{i+1} is located in Voronoi Diagram of i sites
- Its boundary is calculated
- Voronoi diagram of S can be computed in $O(n^2)$ as $O(i)$ edges might be added in each step
- Further improvement over previous method

Yet Another Algorithm

- Site s_{i+1} is located in Voronoi Diagram of i sites
- Its boundary is calculated
- Voronoi diagram of S can be computed in $O(n^2)$ as $O(i)$ edges might be added in each step
- Further improvement over previous method

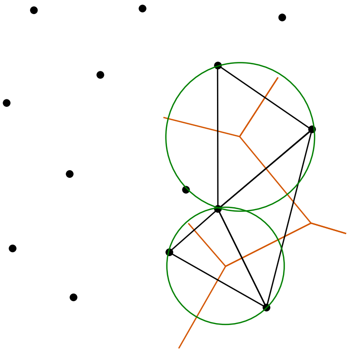
One more Incremental Algorithm-I

Each Voronoi vertex is inserted in Voronoi Diagram one by one



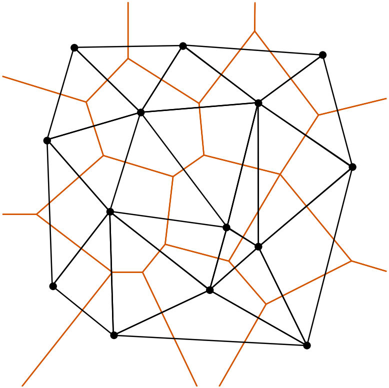
We start with any hull edge and compute *the* empty circle

One more Incremental Algorithm-II



Compute adjacent empty circles

One more Incremental Algorithm-III



Final Voronoi Diagram

Complexity Analysis of Incremental Algorithm

This algorithm also computes Voronoi Diagram in $O(n^2)$

Even though the discussed algorithms are non-aptimal, they are useful for Voronoi Diagram generalisations where the structures are complex and cannot be calculated using usual methods

Can we Compute Voronoi Diagram Faster?

- Clearly no method is optimal.
- Can we do better?

Three Popular Algorithms

We know of three good algorithms.

- Fortune's Sweep Line (Beach Line) Method
- Divide and Conquer
- Reduction to Convex Hull in $\Re^3$

Outline

Reduction to Convex Hull in $\Re^3$

Transform points in plane to space

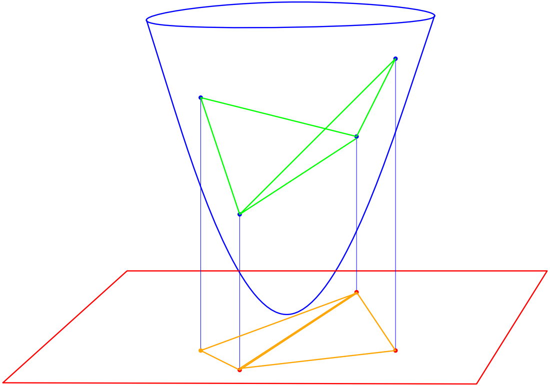
$(x, y) \rightarrow (x, y, x^2 + y^2)$

Basically we project the plane to the surface of a paraboloid

The lower convex hull is Delaunay Triangulation, the dual of Voronoi Diagram

Projection to Paraboloid

Reduction to Convex Hull



Lower Convex Hull is Delaunay Triangulation.

Outline

Fortune's Algorithm

Assumptions

General position assumption: No four sites are co-circular.

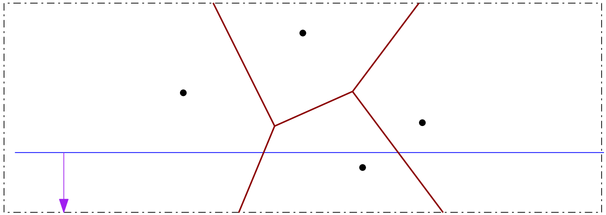
Figures sources: http://www.ams.org/featurecolumn/archive/voronoi.html

Plane sweep for Voronoi diagrams

PLANE SWEEP: Note that the Voronoi diagram above the sweep line may be affected by sites below the sweep line

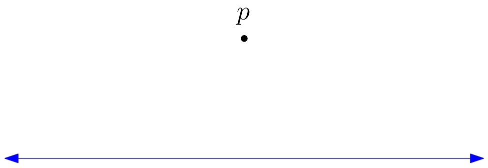
MAINTAIN AND GROW THE PORTION OF VORONOI DIAGRAM ABOVE THE SWEEP LINE THAT IS KNOWN FOR SURE

Plane sweep for Voronoi diagrams



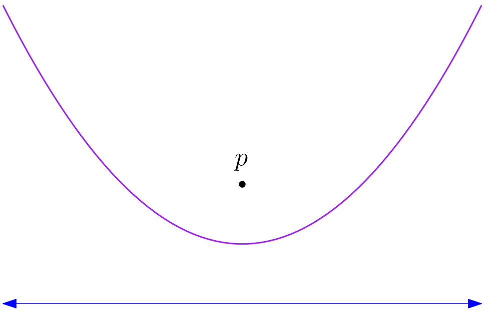
- What are the set of points closer to p than any other point below line the line?
- The BEACH LINE separates the known and unknown part of the Voronoi diagram, it is the minimum of the parabolas defined by sites above the sweep-line and the sweep-line itself

Plane sweep for Voronoi diagrams



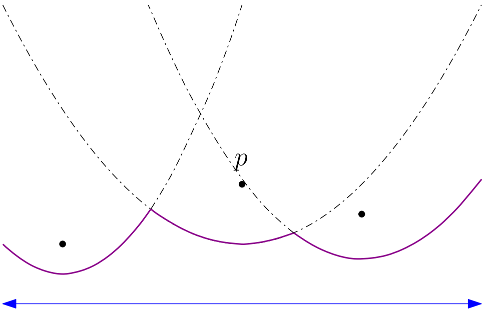
- What are the set of points closer to p than any other point below line the line?
- The BEACH LINE separates the known and unknown part of the Voronoi diagram, it is the minimum of the parabolas defined by sites above the sweep-line and the sweep-line itself

Plane sweep for Voronoi diagrams



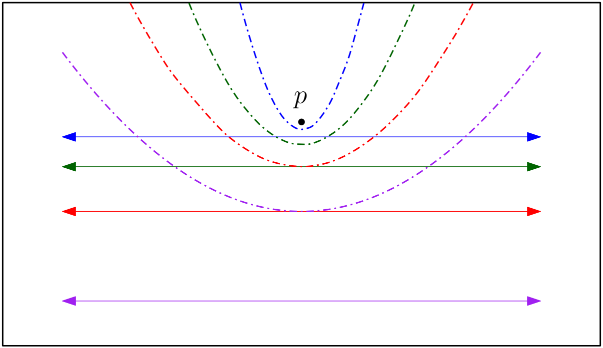
- What are the set of points closer to p than any other point below line the line?
- The BEACH LINE separates the known and unknown part of the Voronoi diagram, it is the minimum of the parabolas defined by sites above the sweep-line and the sweep-line itself

Plane sweep for Voronoi diagrams



- What are the set of points closer to p than any other point below line the line?
- The BEACH LINE separates the known and unknown part of the Voronoi diagram, it is the minimum of the parabolas defined by sites above the sweep-line and the sweep-line itself

Beach line

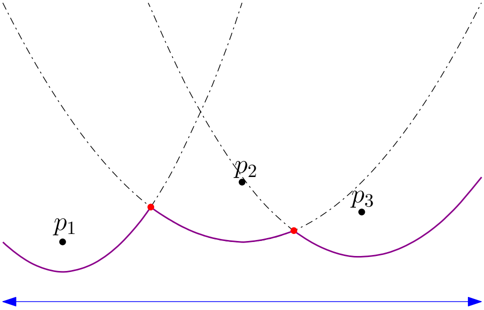


THE BEACH LINE CHANGES CONTINUOUSLY, EVEN ONE PARABOLA DOES

QUESTION: The beach line has break points, what do they represent?

The break points move and trace out the Voronoi diagram edges

Beach line

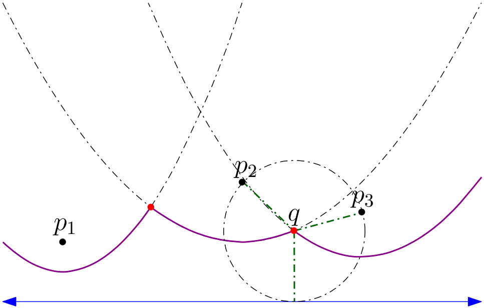


THE BEACH LINE CHANGES CONTINUOUSLY, EVEN ONE PARABOLA DOES

QUESTION: The beach line has break points, what do they represent?

The break points move and trace out the Voronoi diagram edges

Beach line

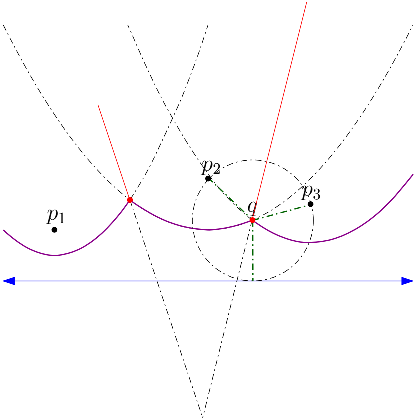


THE BEACH LINE CHANGES CONTINUOUSLY, EVEN ONE PARABOLA DOES

QUESTION: The beach line has break points, what do they represent?

The break points move and trace out the Voronoi diagram edges

Beach line

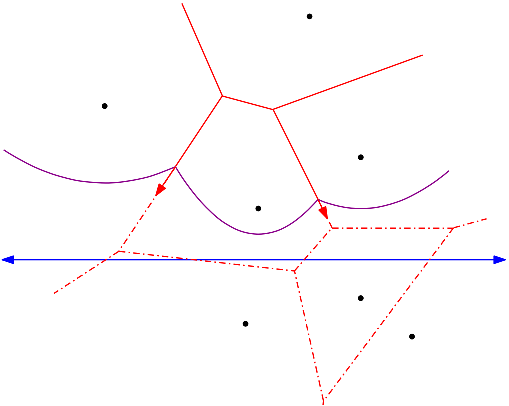


THE BEACH LINE CHANGES CONTINUOUSLY, EVEN ONE PARABOLA DOES

QUESTION: The beach line has break points, what do they represent?

The break points move and trace out the Voronoi diagram edges

Beach line



THE BEACH LINE CHANGES CONTINUOUSLY, EVEN ONE PARABOLA DOES

QUESTION: The beach line has break points, what do they represent?

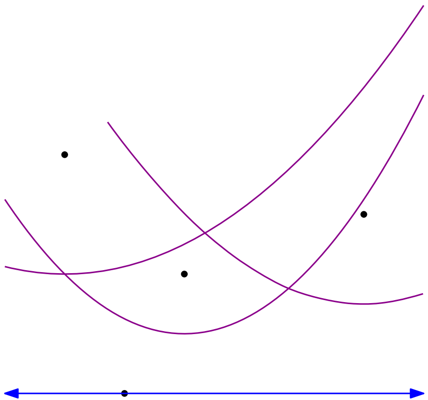
The break points move and trace out the Voronoi diagram edges

Events

The events are where the status changes = where the beach line changes

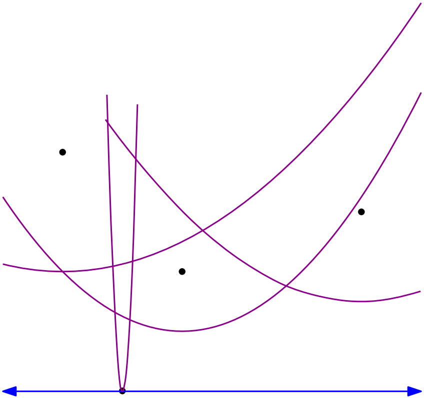
- When the sweep-line reaches a new site
- When a break point reaches the end of the edge it traces

Events



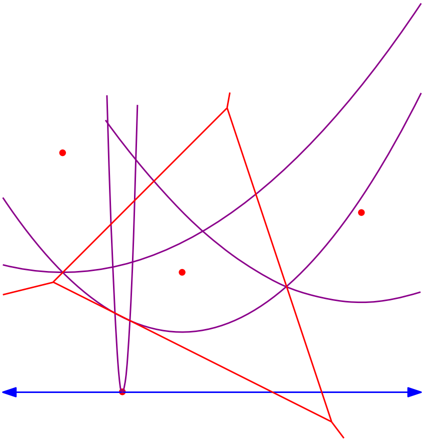
- The sweep-line reaches a new site, a SITE EVENT a new parabola starts

Events



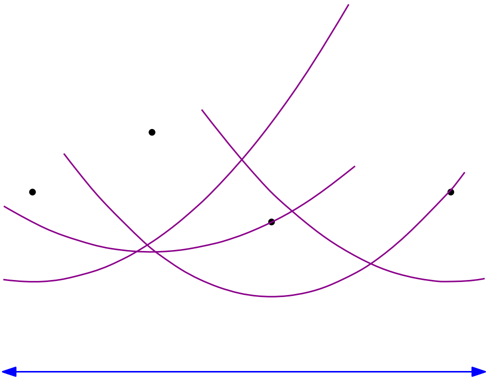
- The sweep-line reaches a new site, a SITE EVENT a new parabola starts

Events

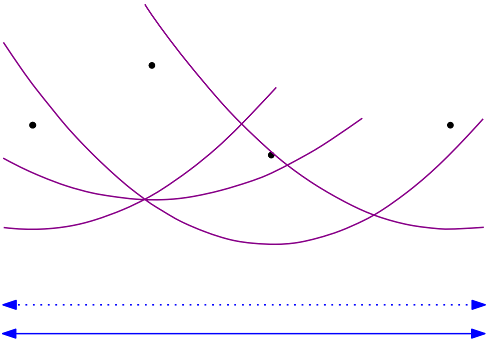


- Two new break points appear on the beach line
- A new Voronoi edge is discovered

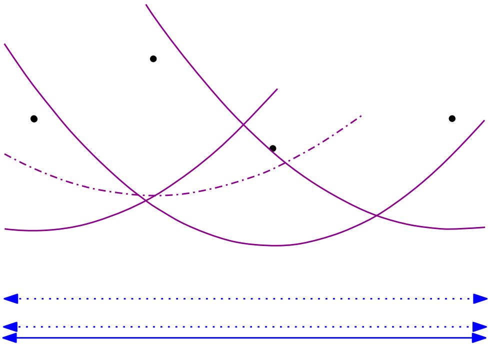
The other events



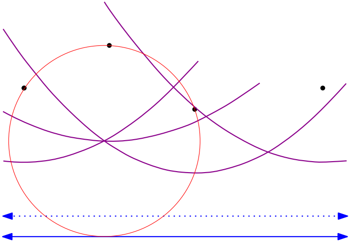
The other events



The other events



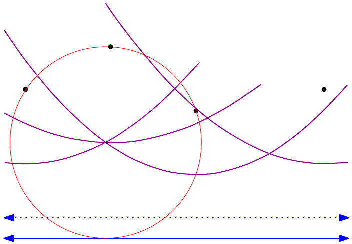
The other events



At a CIRCLE EVENT

- A parabolic arc disappears from the beach line
- Two adjacent break points come together
- A Voronoi vertex is discovered as the vertex incident to two known Voronoi edges
- A new break point starts to be traced
- The sweep line reached the bottom of an empty circle through 3 sites

The other events

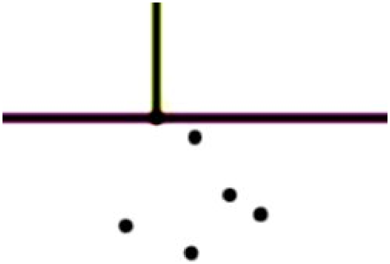


- Circle events can only happen for three sites that have adjacent parabolic arcs on the beach line
- The only way for a new parabolic arc to appear on the beach line is through a site event
- The only way for a parabolic arc to disappear from the beach line is through a circle event
- There are no other events

Execution of the Fortune's Algorithm

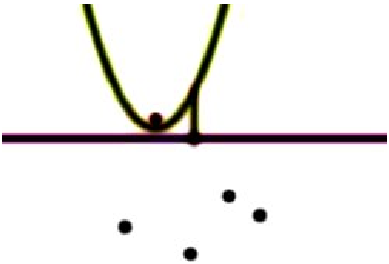
Let us see how the Fortune's algorithm calculates the Voronoi Diagram step by step

Algorithm—Step 1



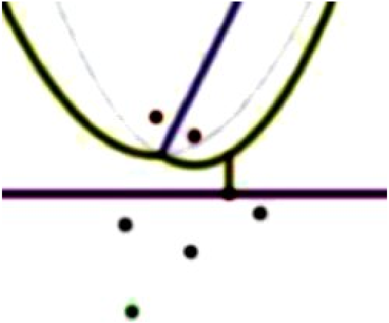
First site and a strange beach line

Algorithm—Step 2



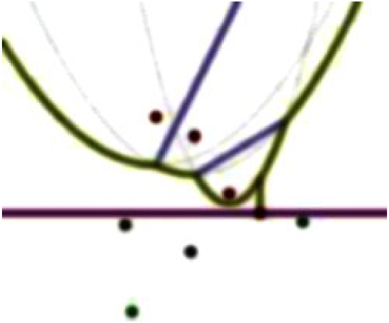
Second site

Algorithm—Step 3



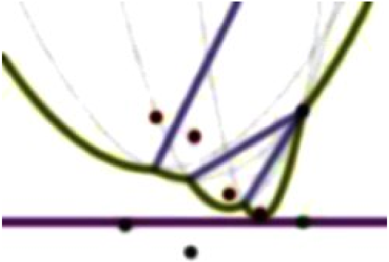
Third site

Algorithm—Step 4



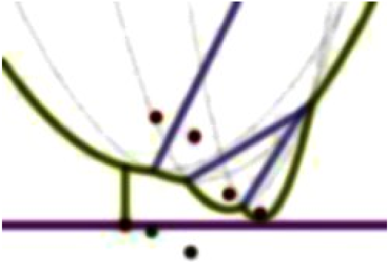
Fourth site and a circle event

Algorithm—Step 5



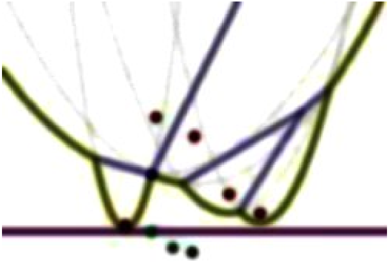
Circle event

Algorithm—Step 6



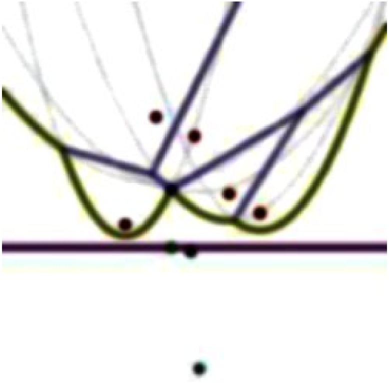
Fifth site and a circle event

Algorithm—Step 7



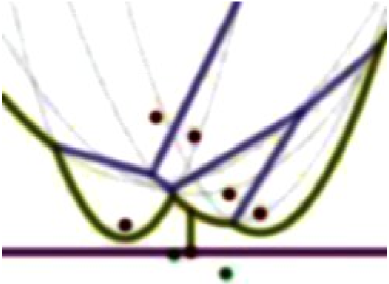
A circle event and another circle event

Algorithm—Step 8



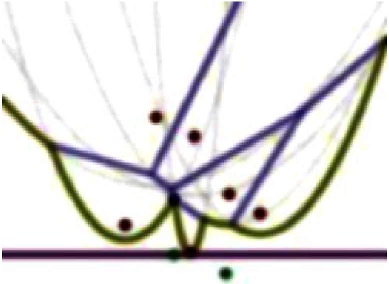
Another of the same

Algorithm—Step 9



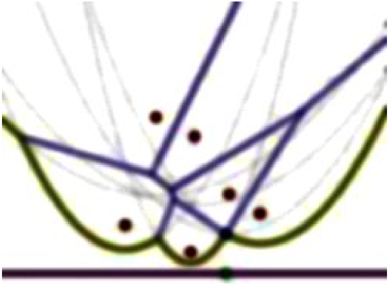
Last site and two circle event

Algorithm—Step 10



Last but one circle event

Algorithm—Step 11



Last circle event

Algorithm—Step 12

Final Output

Analysis of the Fortune's Algorithm

Fortune's algorithm being an example of typical sweep line technique is $O(n \log n)$

Optimal because sorting problem can be reduced to construction of Voronoi Diagrams

Reduction of Sorting Problem to Voronoi Diagram

To sort $x_1, x_2, \ldots, x_n$
We find Voronoi Diagram of points
$(x_1, 0), (x_2, 0), \ldots (x_n, 0)$ and $(0, \infty) = s_\infty$
Either neighbourhood relation or the cell $V(s_\infty)$ gives the required sorted order