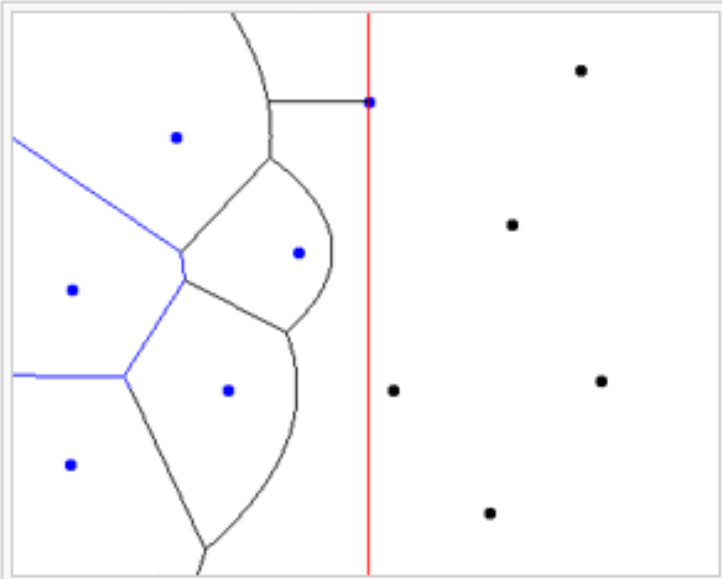
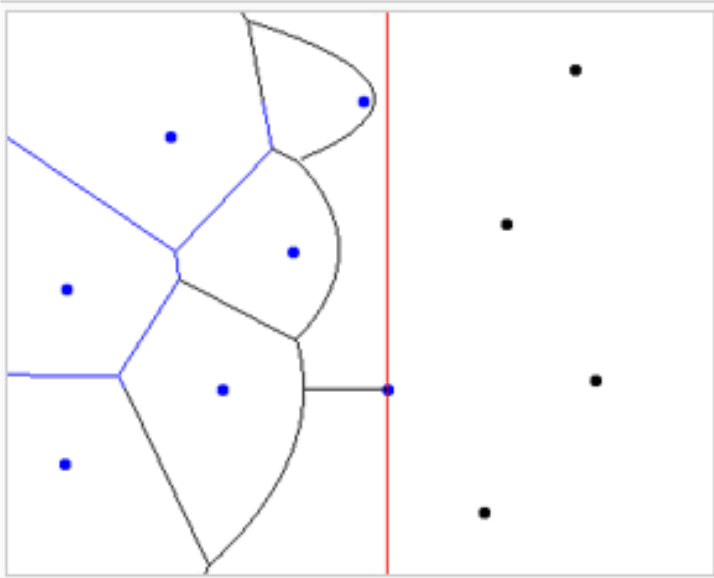
Voronoi Cells and Fortune's algorithm

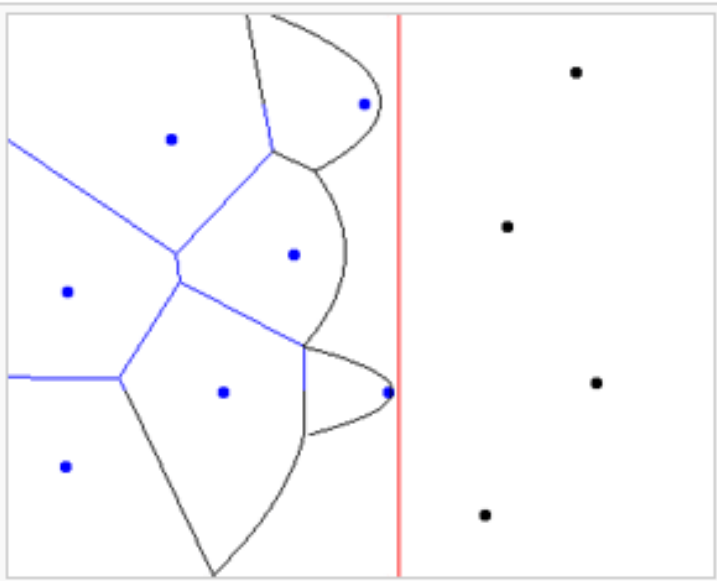
Math/Stats 377:
Operations Research



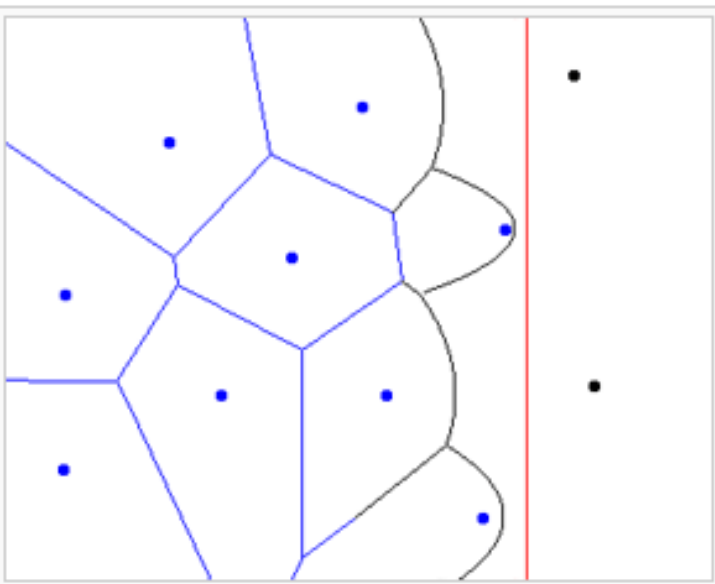
Fortune's algorithm animation



Fortune's algorithm animation

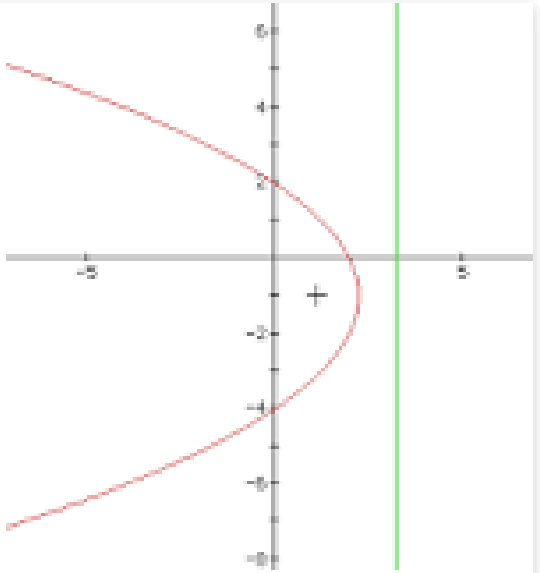


Fortune's algorithm animation



Fortune's algorithm animation

The standard form is $(x - h)^2 = 4p (y - k)$, where the focus is (h, k + p) and the directrix is y = k - p. If the **parabola** is rotated so that its vertex is (h,k) and its axis of symmetry is parallel to the x-axis, it has an **equation** of $(y - k)^2 = 4p (x - h)$, where the focus is (h + p, k) and the directrix is x = h - p.



Parabolas

jwilson.coe.uga.edu/emt725/class/sarfaty/emt669/.../**parabolas**/**parabolas**.html

If points are (a_1, b_1), (a_2, b_2), and the directrix is x = d, then equations of the parabolas are

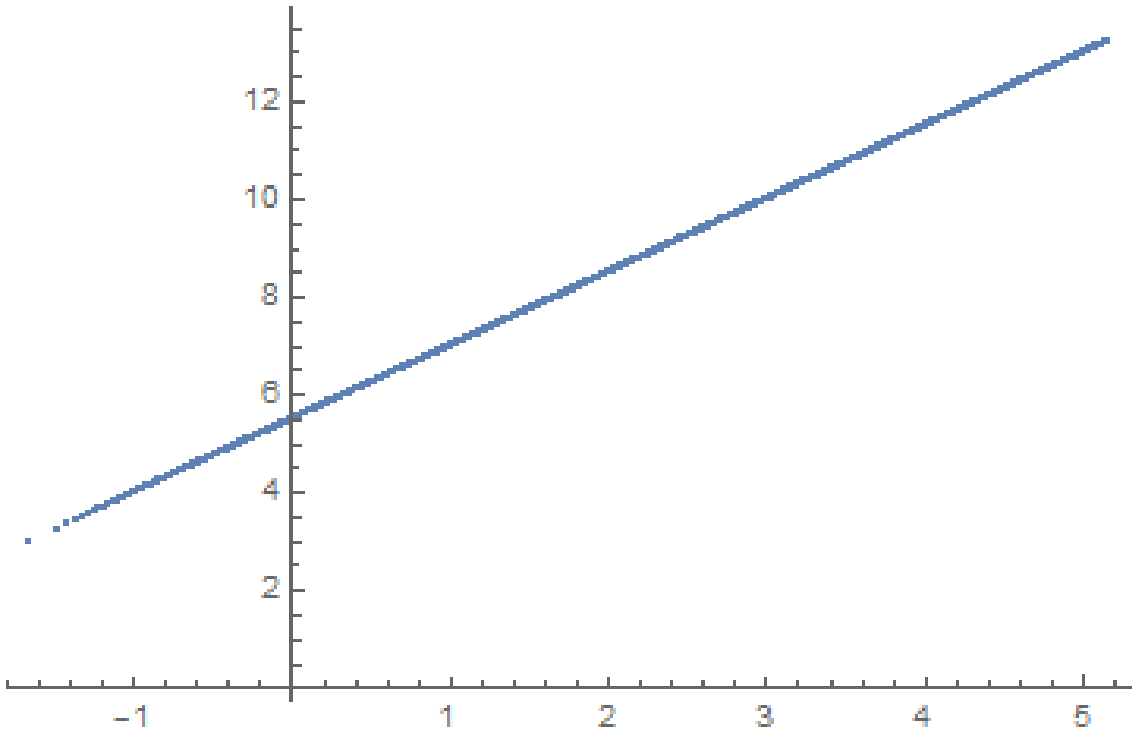
- $2(a_1 - d)(x - (a_1 + d)) = (y - b_1)^2$
- $2(a_2 - d)(x - (a_2 + d)) = (y - b_2)^2$

In[3]:=
```
Simplify[Solve[{2 (a1 - d) (x - (a1 + d)) == (y - b1)^2, 2 (a2 - d) (x - (a2 + d)) == (y - b2)^2}, {x, y}]]
```

Out[3]=
$$\left\{\left\{x \to \frac{1}{2\,(a1-a2)^2}\left(2\,a1^3 - 2\,a1^2\,a2 + 2\,a2^3 + a1\left(-2\,a2^2 + (b1-b2)^2\right) + a2\,(b1-b2)^2 - 2\,(b1-b2)\left(\sqrt{\left(2\,a1^2 - 4\,a1\,a2 + 2\,a2^2 + (b1-b2)^2\right)(a1-d)(a2-d)} + b1\,d - b2\,d\right)\right),\right.$$
$$\left. y \to \frac{-a2\,b1 + a1\,b2 + \sqrt{\left(2\,a1^2 - 4\,a1\,a2 + 2\,a2^2 + (b1-b2)^2\right)(a1-d)(a2-d)} + b1\,d - b2\,d}{a1-a2}\right\},$$
$$\left\{x \to \frac{1}{2\,(a1-a2)^2}\left(2\,a1^3 - 2\,a1^2\,a2 + 2\,a2^3 + a1\left(-2\,a2^2 + (b1-b2)^2\right) + a2\,(b1-b2)^2 + 2\,(b1-b2)\left(\sqrt{\left(2\,a1^2 - 4\,a1\,a2 + 2\,a2^2 + (b1-b2)^2\right)(a1-d)(a2-d)} - b1\,d + b2\,d\right)\right),\right.$$
$$\left. y \to -\frac{a2\,b1 - a1\,b2 + \sqrt{\left(2\,a1^2 - 4\,a1\,a2 + 2\,a2^2 + (b1-b2)^2\right)(a1-d)(a2-d)} - b1\,d + b2\,d}{a1-a2}\right\}\right\}$$

In[16]:=
```
x[a1_, a2_, b1_, b2_, d_] :=
  1/(2 (a1 - a2)^2) (2 a1^3 - 2 a1^2 a2 + 2 a2^3 + a1 (-2 a2^2 + (b1 - b2)^2) + a2 (b1 - b2)^2 -
    2 (b1 - b2) (Sqrt[(2 a1^2 - 4 a1 a2 + 2 a2^2 + (b1 - b2)^2) (a1 - d) (a2 - d)] + b1 d - b2 d));
y[a1_, a2_, b1_, b2_, d_] := (-a2 b1 + a1 b2 + Sqrt[(2 a1^2 - 4 a1 a2 + 2 a2^2 + (b1 - b2)^2) (a1 - d) (a2 - d)] + b1 d - b2 d)/(a1 - a2);
```

```
list = {};
For[t = 0, t ≤ 1000, t++, list = AppendTo[list, {x[1, -2, 3, 5, t / 100], y[1, -2, 3, 5, t / 100]}]];
ListPlot[list]
```



Algorithm description[edit] https://en.wikipedia.org/wiki/Fortune%27s_algorithm

The algorithm maintains both a sweep line and a *beach line*, which both move through the plane as the algorithm progresses. The sweep line is a straight line, which we may by convention assume to be vertical and moving left to right across the plane. At any time during the algorithm, the input points left of the sweep line will have been incorporated into the Voronoi diagram, while the points right of the sweep line will not have been considered yet. The beach line is not a line, but a complicated, piecewise curve to the left of the sweep line, composed of pieces of parabolas; it divides the portion of the plane within which the Voronoi diagram can be known, regardless of what other points might be right of the sweep line, from the rest of the plane. For each point left of the sweep line, one can define a parabola of points equidistant from that point and from the sweep line; the beach line is the boundary of the union of these parabolas. As the sweep line progresses, the vertices of the beach line, at which two parabolas cross, trace out the edges of the Voronoi diagram. The beach line progresses by keeping each parabola base exactly half way between the points initially swept over with the sweep line, and the new position of the sweep line. Mathematically, this means each parabola is formed by using the sweep line as the directrix and the input point as the focus.

The algorithm maintains as data structures a binary search tree describing the combinatorial structure of the beach line, and a priority queue listing potential future events that could change the beach line structure. These events include the addition of another parabola to the beach line (when the sweep line crosses another input point) and the removal of a curve from the beach line (when the sweep line becomes tangent to a circle through some three input points whose parabolas form consecutive segments of the beach line). Each such event may be prioritized by the x-coordinate of the sweep line at the point the event occurs. The algorithm itself then consists of repeatedly removing the next event from the priority queue, finding the changes the event causes in the beach line, and updating the data structures.

As there are $O(n)$ events to process (each being associated with some feature of the Voronoi diagram) and $O(\log n)$ time to process an event (each consisting of a constant number of binary search tree and priority queue operations) the total time is $O(n \log n)$.