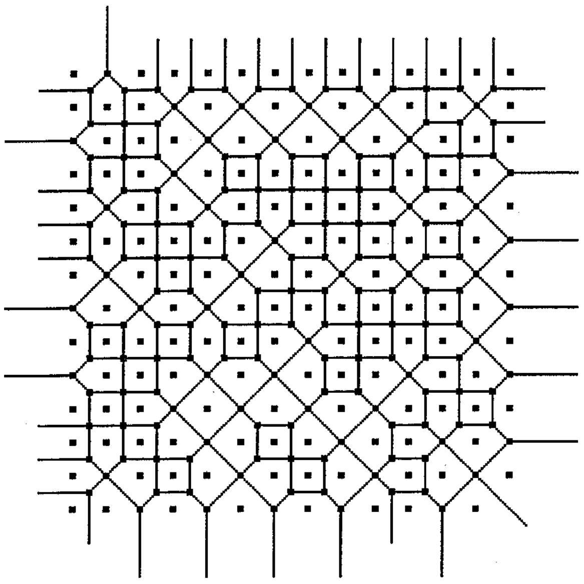


computationele meetkunde

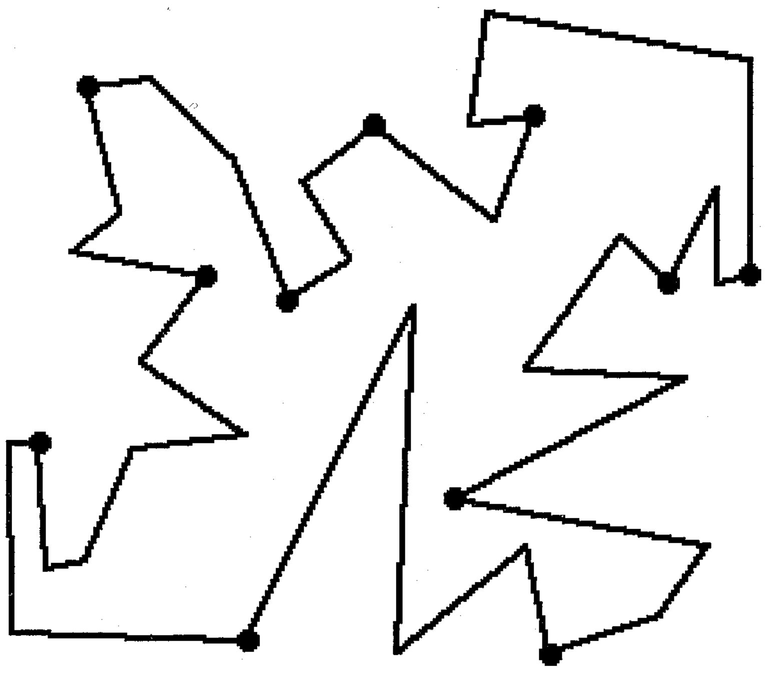
lieven le bruyn, uia 1999-2000

Week 1

ArtGallery.

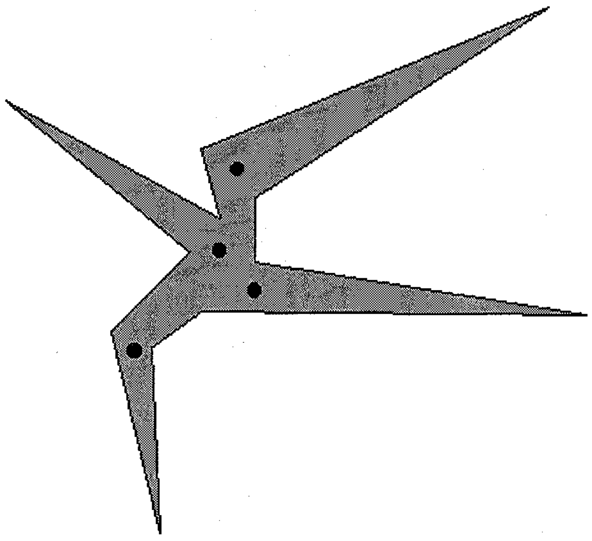
Imagine an art gallery whose floor plan can be modeled by a polygon of n vertices. How many stationary guards are needed to guard the room ? Each guard is considered a fixed point that can see in every direction (that is have a 2π range of visibility). Of course a guard cannot see through a wall of the gallery.

Below a solution for a gallery having 38 vertices needing at most 11 guards placed at the marked vertices.



The 'ArtGallery theorem' asserts that this can always be done with $\leq \lfloor \frac{n}{3} \rfloor$ guards. The theorem was first proved by V. Chatával in 1975 and we will give the computational proof found by S. Fisk.

We start by fixing some terminology. A *polygon* is a region P of the plane $\mathbb{R}^2$ bounded by a finite collection of line segments forming a simple closed curve, that is homeomorphic to the circle. The *Jordan curve theorem* asserts that every simple plane curve divides the plane into two parts : the *interior* and *exterior* of the curve. To make the notion of *visibility* precise, we say that a point x can *see* point y iff the closed segment xy is nowhere exterior to the polygon P. A set of guards is said to *cover* a polygon if every point in the polygon is visible to some guard. To begin, let us give an example of a gallery where the bound is attained : a gallery with 12 vertices requiring 4 guards :

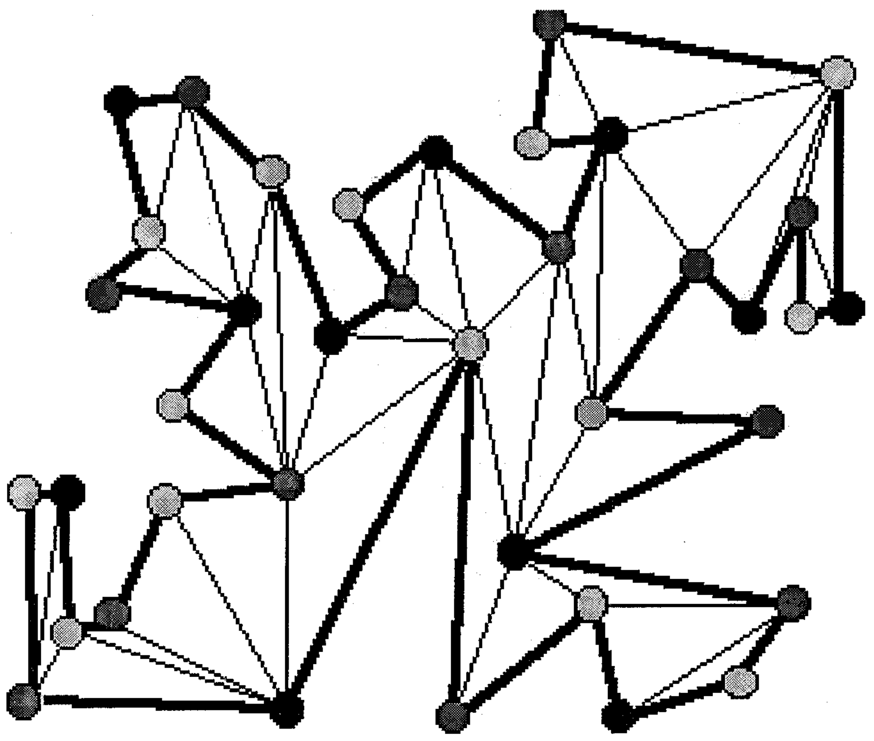


A *diagonal* of a polygon P is a line segment between two of its vertices a and b that are visible to each other, that is, $ab \cap \partial P = \{a, b\}$ where ∂P denotes the *boundary* of P. We call two diagonals *noncrossing* if their intersection is a subset of their endpoints (that is, they share no interior point). A *triangulation* of a polygon P is a covering of the interior by triangles all of whose sides are either sides of the polygon or noncrossing diagonals.
A *k-coloring* of a graph G is an assignment of $\leq k$ colors to the vertices of G such that no two vertices connected by an edge are assigned the same color. If G has n vertices and if G can be k-colored than at least one of the k colors colors at most $\lfloor \frac{n}{k} \rfloor$ vertices. This is a variant of the *pigeon-hole principle* : if n objects are placed into k pigeon holes, then at least one hole must contain no more than $\frac{n}{k}$ objects.

Fisk's proof of the ArtGallery theorem is based on the following facts (which we will prove later)

- Every polygon can be triangulated.
- The graph of the sides of the triangulation can be 3-colored.

For example, in the example above we have the following triangulation and 3-coloring



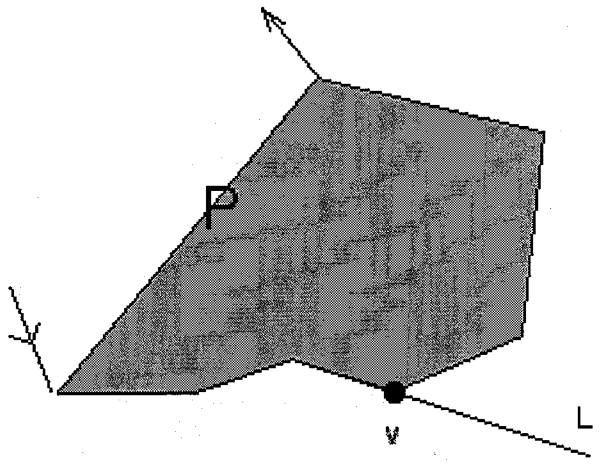
Let black, grey and white be the three colors needed with black the one coloring the least vertices (in the example, black colors 11 vertices, grey 13 and white 14). Now, place guards in all the black vertices. Every triangle in he triangulation of P must have vertices of each of the three colors, so contains one black vertex. Clearly, a guard posted in that vertex can see every point in the triangle. As the triangles cover the whole of the interior this finishes the proof of the ArtGallery theorem.

1a Triangulation and Three-coloring.

We will now prove the claims needed in the proof of the ArtGallery theorem. A vertex of a polygon is called *reflex* if its internal angle is $> \pi$ and is called *convex* if its internal angle is $< \pi$.

Lemma 1.1. *Every polygon has at least one convex vertex.*

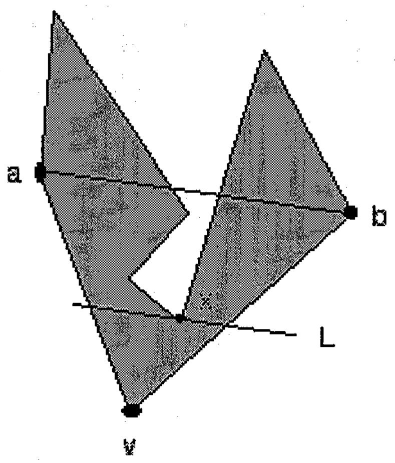
Proof. Walk around the edge of the polygon in counter-clockwise direction, then the interior is always on your left side. Let v be the vertex with minimal y-value and if there are more such vertices, take the one most to the right.



Consider the line L extending the edge coming into v, then the interior of P must be above L and also the edge following v must lie above L. Therefore, one makes a left-turn at v which happens precisely at convex vertices. □

Lemma 1.2. *Every polygon on $n \geq 4$ vertices has a diagonal.*

Proof. Let v be a convex vertex and a and b the vertices adjacent to v. Either ab is a diagonal or the triangle $\triangle avb$ contains at least one other vertex of P. Now, move a line L parallel towards the line ab and let x be the first vertex hitting L



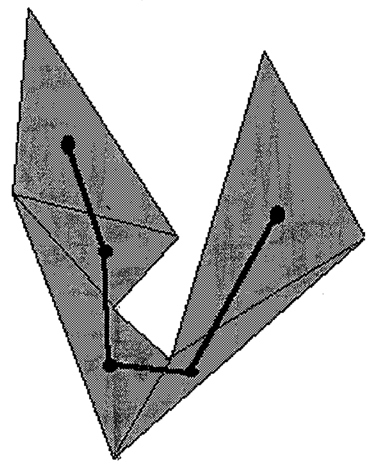
The region of P strictly below L cannot contain points of ∂P, so vx intersects ∂P only at the endpoints so is a diagonal. □

Theorem 1.3. (Triangulation) *Every polygon P on n vertices can be partitioned into triangles by the addition of (zero or more) diagonals. Such a triangulation uses $n-3$ diagonals and consists of $n-2$ triangles.*

Proof. By induction. If $n = 3$, then P is a triangle and the result is obvious. If $n \geq 4$, let $d = ab$ be a diagonal of P, then d partitions P into two polygons each having fewer than n vertices say n_1, n_2 with $n_1 + n_2 = n + 2$. Applying induction to the two subpolygons we get a triangulation. Again by induction, the subpolygons are triangulated using $n_1 - 3$ and $n_2 - 3$ diagonals, so we need $n_1 - 3 + n_2 - 3 + 1 = n - 3$ diagonals and have $n_1 - 2 + n_2 - 2 = n - 2$ triangles. □

The *dual* T of a triangulation of a polygon is a graph with a vertex associated to each triangle and an edge between two vertices iff their triangles share a diagonal.

fout



Lemma 1.4. *The dual T of a triangulation is a tree with the degree of each vertex at most* 3.

Proof. If T is not a tree, it must contain a cycle C. We can draw this cycle as a path C' in the interior of P by straight lines connecting the dots with the midpoints of diagonals shared by the triangles whose vertices make up C. This path must enclose some polygon vertices (one endpoint for each diagonal crossed by C'. But then C' also encloses points exterior to P as all vertices lie on ∂P. But by the Jordan curve theorem this contradicts the simplicity of the polygon. Clearly the degree of each vertex can be at most 3 as a triangle has at most 3 sides to share. □

We call three consecutive vertices a, b, c of a polygon P an *ear* of the polygon if ac is a diagonal, b is then called the *ear tip*.

Theorem 1.5. (Meister's TwoEars theorem) *Every polygon on $n \geq 4$ vertices has at least two non-overlapping ears.*

Proof. Consider the dual T for a triangulation of P. It is a tree with $n-2$ vertices. Now, any tree with at least 2 vertices has at least 2 vertices of degree 1 (the *leaves* of the tree). A leaf node in T corresponds to an ear of P. □

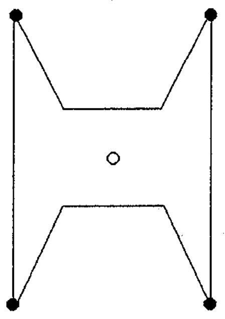
Theorem 1.6. (Three-coloring) *The triangulation graph of a polygon P can be three-colored.*

Proof. Induction on the number of vertices n. If $n = 3$ then we are done. If $n \geq 4$, then P has an ear $\triangle abc$ with ear tip b. Consider the polygon P' obtained by cutting off the ear (that is, $ac \subset \partial P'$). P' has $n-1$ vertices so by induction the triangulation graph of P' (which is a subgraph of that of P) can be 3-colored. Now, put the ear back and color b with the color not used by a and c. □

Solutions 1

1a Exercises

1. An example is given by the situation



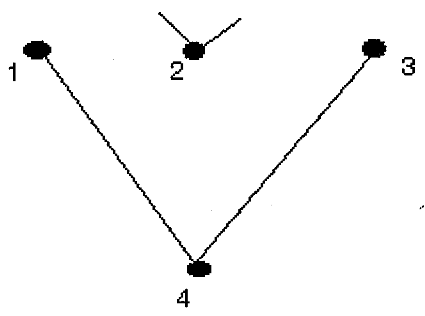
2. In a triangulation of P there are $n - 2$ triangles each contributing π to the internal angles.

3. By induction : suppose that the binary tree with two connected leaves dropped is the dual to a triangulation of a polygon P. The vertex connecting the dropped leaves corresponds to a triangle in the triangulation having 2 external edges. One can then glue appropriately small triangles to these two edges.

1b Homework

1. (Stijn S. + Wim M.) Som van binnen- en buitenhoeken is $2\pi n$. Som van de binnenhoeken is $(n-2)\pi$, dus is de som van de buitenhoeken $(n+2)\pi$.

2. (Sam R. + Tom H.) Het antwoord is ja, neem gewoon hetzelfde bewijs. Het enige probleem dat zich kan voordoen is een situatie als



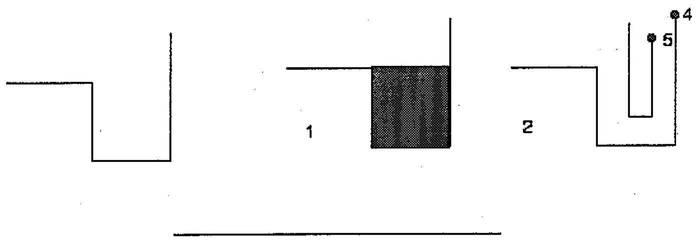
want in dit geval heeft een bewaker in 1 geen clear visibility over]2, 3]. Dit is echter onmogelijk want [1, 3] is een diagonaal en dus kan 2 er niet op liggen (anders namen we in ons bewijs de diagonaal [1, 2] om te trianguleren).

3. (Tom H.) 1. P heeft twee opeenvolgende vertices waar we twee keer achter elkaar naar links (naar rechts) draaien.

bewijs : indien niet dan is de som van de binnenhoeken

$$S \geq \lfloor\frac{n}{2}\rfloor(\frac{3\pi}{2}+\frac{\pi}{2}) = \lfloor\frac{n}{2}\rfloor 2\pi > (n-2)\pi$$

2. Dus komt er ergens een situatie als beneden links voor en dan zijn er twee megelijkheden : 1 of 2

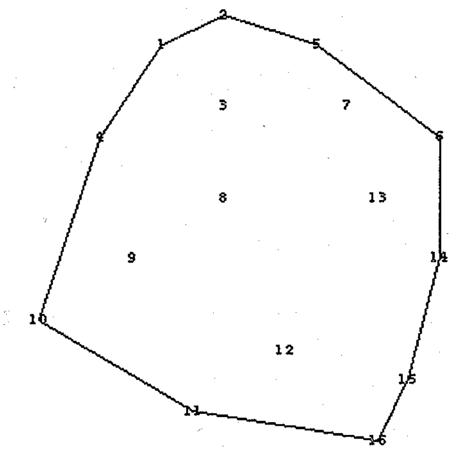


In het geval 1 zijn we klaar, in geval 2 moeten we om van 4 naar 5 te gaan twee keer naar links draaien en krijgen we dus weer ergens een situatie als voorheen (boven links). Dit proces blijven we herhalen en aangezien we telkens minder vertices beschouwen zal er ergens een boxed ear voorkomen en zal he proces stoppen.

Week 2

ConvexHull.

```
In[1]:=<<DiscreteMath`ComputationalGeometry`

In[2]:=data2D={{4,14},{6,15},{6,12},{2,11},{9,14},
{13,11},{10,12},{6,9},{3,7},{0,5},{5,2},
{8,4},{11,9},{13,7},{12,3},{11,1}};

In[3]:=convexhull=ConvexHull[data2D]

Out[3]:={14,6,5,2,1,4,10,11,16,15}

In[4]:=PlanarGraphPlot[data2D,convexhull]
```



```
ConvexHull[{{x1,y1},{x2,y2},...}] computes the convex
hull of a set of points in the plane.
```

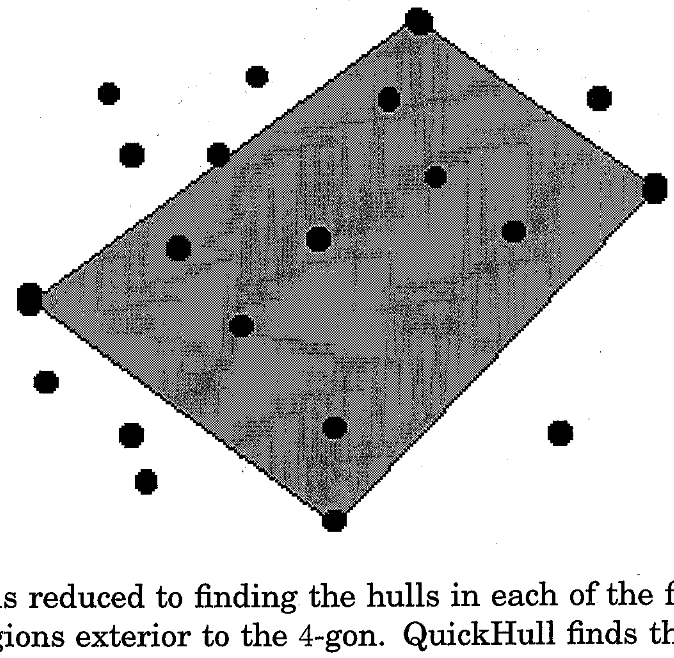
2a QuickHull.

The most ubiquitous structure in computational geometry is the convex hull. Recall that a set S is said to be *convex* if $x \in S$ and $y \in S$ implies that the closed segment $xy \subset S$. A *convex combination* of points $p_1, \ldots, p_k$ is a sum of the form

$$\alpha_1 p_1 + \ldots \alpha_k p_k \quad \text{with} \quad \alpha_i \geq 0 \quad \text{and} \quad \alpha_1 + \ldots + \alpha_k = 1$$

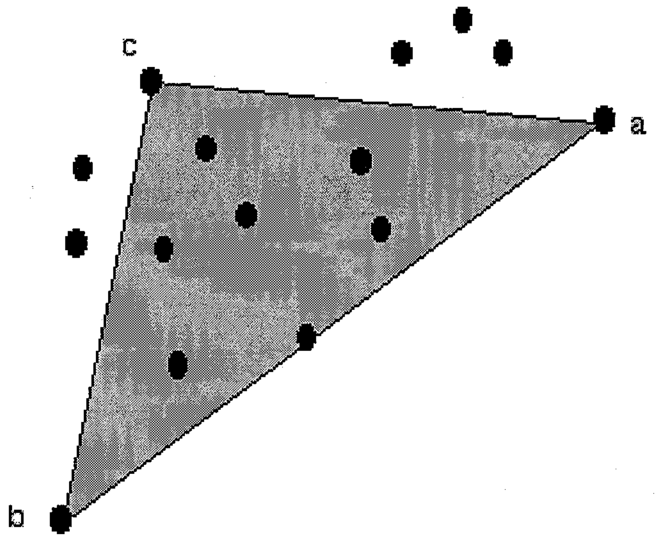
Here, points p_i can live in any affine space $\mathbb{R}^d$. The *convex hull* of a set of points S is the set of all convex combinations of S. If $S \subset \mathbb{R}^d$, then it is the set of all convex combinations of $d+1$ (or fewer) points of S.
If S is a set of points in the plane, then the convex hull of S is the smallest convex polygon P that encloses S. We will give a few algorithms to compute the ConvexHull of a set of plane points.

The basic intuition of the 'QuickHull' algorithm proposed in the late 1970s is : for 'most' sets of points, it is easy to discard many points as definitely interior to the hull and then concentrate on those closer to the hull boundary. The first step is to find the points extreme in the four compass directions : highest, lowest, leftmost and rightmost points. Connect these extreme points with a polygon (usually a 4-gon) then all the points of S on the boundary or inside this polygon may be discarded from further consideration.



The problem is reduced to finding the hulls in each of the four remaining triangular regions exterior to the 4-gon. QuickHull finds these by finding an extreme point in the triangle, discarding points and recursing to two smaller sets of points.

At any stage we know two points a and b on the hull and a set of points S strictly to the right of ab. We then find a point $c \in S$ extreme in the direction orthogonal to ab and form the triangle $\triangle acb$. We can then discard all points on the edges or inside this triangle (apart from the vertices) and repeat the procedure on the points A right of ac and the set of points B right of cb.



The essence of the algorithm is a recursive function that takes as input a,b and S

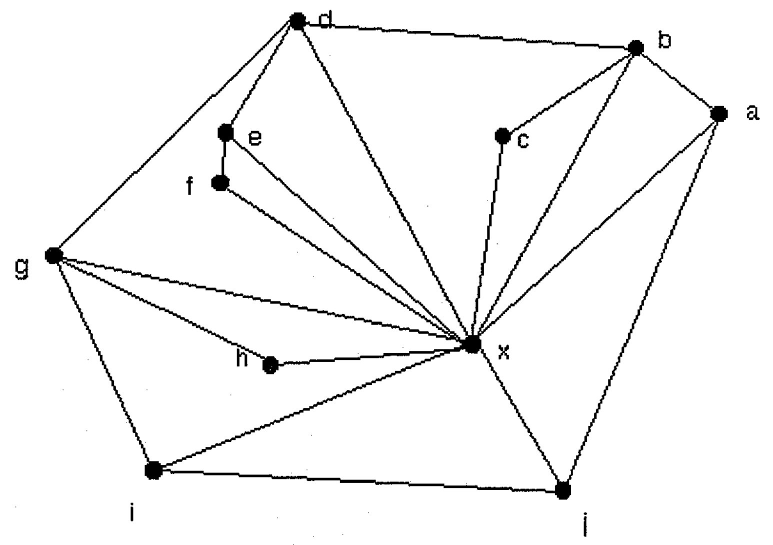
```
function QuickHull(a,b,S)
   if S={a,b} then return (a,b)
   else
      c <- point of max distance from ab
      A <- points right of (a,c)
      B <- points right of (c,b)
      return QuickHull(a,c,A) concatenated with
             QuickHull(c,b,B)
```

The complexity of the QuickHull algorithm is $O(n^2)$, that is quadratic.

2b Graham's algorithm.

In 1972 R. Graham found an algorithm to construct the convex hull with complexity $O(n \log n)$. Assume we are given a point x interior to the hull and assume that no three points of the set are collinear. Sort the points by angle, counterclockwise about x. The points are processed in their sorted order and the hull grows incrementally.

The hull-so-far is maintained in a stack S of points. Initially, the stack contains the first two points $S = (a, b)$ with b on top.

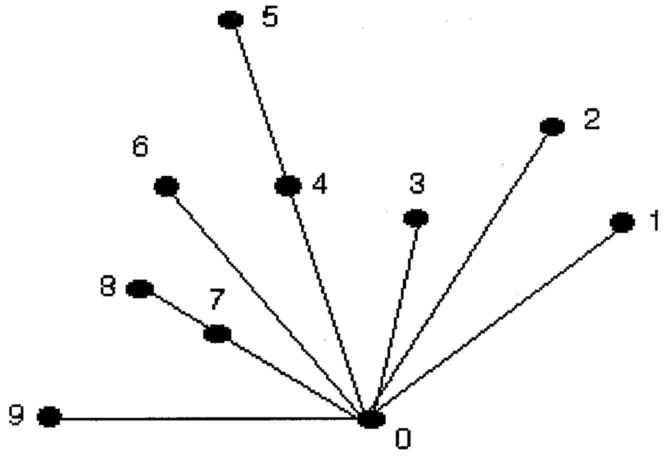


We continue according to the following rules :

- the next point is added on top whenever we make a left turn at the last point,
- when we make a right turn we drop points from the stack until we make a left turn.

In the example we get (a, b, c) but then d would be a right turn at c so we drop c and as we make a left turn in b we keep b and have (a, b, d).
If we are so fortunate as in the example and our first point a is on the hull, the convex chain will close naturally. We will see below how this and other problems discarded in the above simplifications can be avoided.
We started with a point x lying in the interior of the hull. We can simplify by starting with the lowest point which must be on the hull. If there are several points with minimal y-coordinate we take the rightmost one of them : call this p_0 and sort the other points around p_0 calling them $p_1, \ldots, p_{n-1}$ as in the picture below. There are no problems to end the procedure as both p_0 and p_1 must belong to the hull as does p_{n-1} (the last point in the sorting). There is also no possible start-up problem (which might arise otherwise when the stack is (a, b) and we make a right turn to reach c, then we must pop b from the stack but there do not remain at least two points to declare left from right turns). We now initialize the stack to be $S = (p_{n-1}, p_0)$, then the stack will always contain two points.
The other problem of having collinearities when sorting points around p_0 can be solved by enumerating two points with the same angle according to

their distance from p_0.



The only remaining problem is that of collinearities along the hull. This is solved by requiring a *strict* left turn (p_{t-1}, p_t, p_i) to push p_i onto the stack having p_t and p_{t-1} as its top two points and if p_t is collinear with p_{t-1} and p_i it will be deleted.

```
Algorithm Graham
find rightmost lowest point : p(0)
sort other point angularly about p(0)
     break ties in favor of closeness to p(0)
     label p(1),p(2),...,p(n-1)
stack S=(p(n-1),p(0))=(p(t-1),p(t) ; t indexes top
i=1
while i<n do
   if p(i) is strictly left of (p(t-1),p(t))
           then push(S,i) and set i <- i+1
           else pop(S)
```

Exercises 2.

1. Give a method to determine whether a point c lies to the left or right of $\vec{ab}$.
2. Give algorithm to determine the point c of maximal distance to $\vec{ab}$.
3. What will be the output of the Graham algorithm when all points are collinear ?
4. (*Convex deficiency tree*) . Let P be a polygon and $H(P)$ its convex hull. Define the convex deficiency $D(P)$ of P to be the set of points $H(P)-P$. In general there will be several disconnected components called *bays*. The closure of each of these bays is again a polygon.

   Define the deficience tree $T(P)$ for a polygon as follows. The root of $T(P)$ is a node associated to P. The children of the root are nodes associated with the distinct bays of $D(P)$. In general if P' is the polygon corresponding to a node of $T(P)$, the children of this node correspond to the bays of $D(P')$.

   - Prove that $T(P)$ is a finite tree.
   - For a polygon on n vertices, what is the largest possible degree of a node in $T(P)$? Give a worst case example.

Homework 2.

1. ● ● : Design an algorithm to find a line L such that

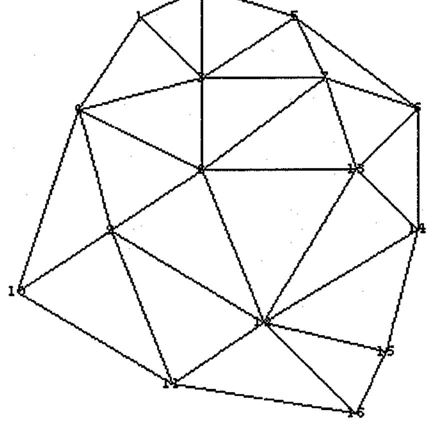
   - L has all points of a given set to one side.
   - L minimizes the sum of the perpendicular distances of the points of the set to L.

   You may assume existence of a `ConvexHull` algorithm.
2. ● ● : For each n construct a set of points such that the Graham algorithm needs the largest number of iterations to finish.
3. ● ● ● : Can every tree be relized as a convex deficiency tree $T(P)$ of some polygon P ? Why (not).

Week 3

DelaunayTriangulation.

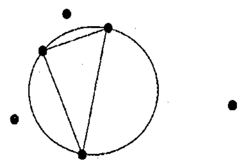
```
In[1]:=<<DiscreteMath`ComputationalGeometry`

In[2]:=data2D={{4,14},{6,15},{6,12},{2,11},{9,14},
{13,11},{10,12},{6,9},{3,7},{0,5},{5,2},
{8,4},{11,9},{13,7},{12,3},{11,1}};

In[3]:=delaynay=DelaunayTriangulation[data2D]

Out[3]:={{1,{4,3,2}},{2,{1,3,5}},{3,{2,1,4,8,7,5}},
   {4,{10,9,8,3,1}},{5,{2,3,7,6}},{6,{5,7,13,14}},{7,{5,3,8,13,6}},
   {7,{5,3,8,13,6}},{8,{3,4,9,12,13,7}},{9,{4,10,11,12,8}},
   {10,{11,9,4}},{11,{16,12,9,10}},{12,{8,9,11,16,15,14,13}},
   {13,{7,8,12,14,6}},{14,{6,13,12,15}},{15,{14,12,16}},
   {16,{15,12,11}}}

In[4]:=PlanarGraphPlot[data2D,delaynay]
```



```
DelaunayTriangulation[{{x1,y1},{x2,y2},...}] computes the
Delaunay triangulation of a set of points in the plane.
```

The importance of the Delaunay triangulation will become clear when we will encounter the *Voronoi diagrams* next weeks. For the moment we formalize

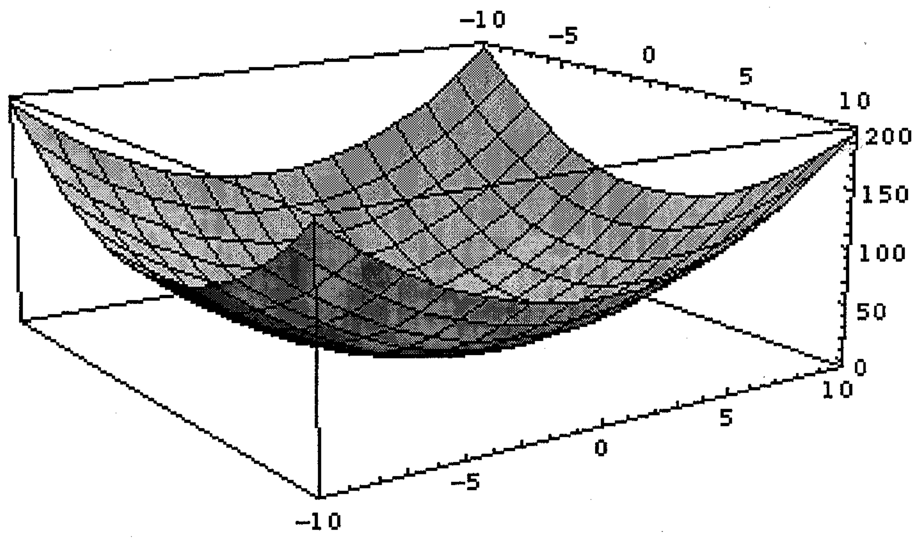
Definition 3.1. Let P be a set of points in the plane. The Delaunay triangulation $\mathcal{D}(P)$ is a triangulation of the points P having the following property : for every triangle in $\mathcal{D}(P)$ the circumscribed circle



does not contain any other points of P either on its boundary or in its interior.

3a EdelsbrunnerSeidel.

In 1986, H. Edelsbrunner and R. Seidel discovered a beautiful connection between Delaunay triangulations and convex hulls of points in $\mathbb{R}^3$. As these convex hulls can be calculated in $O(n\ log\ n)$ time, this gives a very efficient algorithm to compute these triangulations.



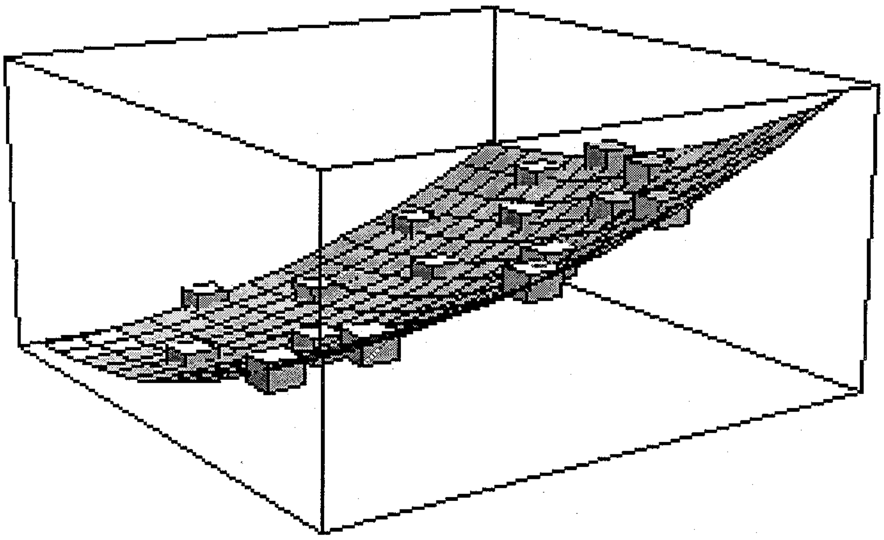
Consider the *paraboloid* in $\mathbb{R}^3$ with equation

$$z = x^2 + y^2$$

Given points $P = \{p_1, \ldots, p_z\}$ in the plane we project them upwards until they hit the paraboloid, that is, we map every point as follows

$$(x_i, y_i) \longrightarrow (x_i, y_i, x_i^2 + y_i^2)$$

In the example above, we obtain the following points (each represented by a cuboid) on the first quadrant of the paraboloid.



Take the convex hull in $\mathbb{R}^3$ of this set of 3-dimensional points $P' = \{p'_1, \ldots, p'_z\}$ and discard the 'top' faces of this convex hull (that is, all those faces whose outward pointing normal vector points upward, in the sense of having a positive scalar product with the $(0,0,1)$ vector). What we get is the 'bottom shell' of the triangulation.
The claim is that the projection to the xy-plane of this bottom shell is the Delaunay triangulation ! We will now prove this stunning connection.

Consider a point $(a, b) \in \mathbb{R}^2$, then the equation of the tangent plane T to the paraboloid Q in the point $(a, b, a^2 + b^2)$ (the point lying over (a, b)) is

$$T \quad : \quad z = 2ax + 2by - (a^2 + b^2)$$

Shift this tangent plane upwards by r^2 for some $r \in \mathbb{R}_+$, then this plane T_r has the equation

$$T_r \quad : \quad z = 2ax + 2by - (a^2 + b^2) + r^2$$

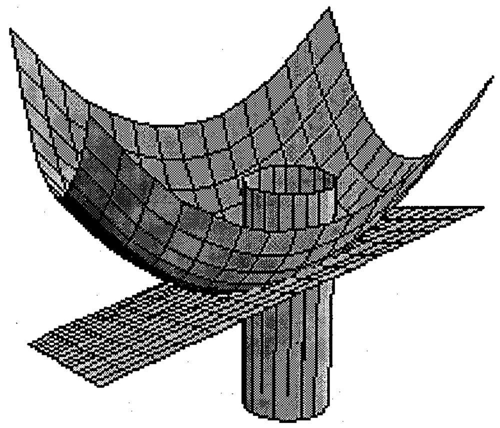
In order to see the intersection $T_r \cap Q$ we have to solve the system

$$\begin{cases} z = x^2 + y^2 \\ z = 2ax + 2by - (a^2 + b^2) + r^2 \end{cases}$$

which reduces to the equality

$$(x-a)^2 + (y-b)^2 = r^2 \quad \text{and} \quad z = 2ax + 2by - (a^2 + b^2) + r^2$$

Hence, the shifted tangent plane T_r intersects the paraboloid Q in a curve (actually an ellipse) that projects to the xy-plane to a circle with center (a, b) and radius r. All of this can be visualized in the following picture



Now, consider a plane T' through 3 points on the paraboloid that form a face $\triangle = \triangle(p'_i, p'_j, p'_k)$ of the convex hull in 3 dimensions. Now, translate T' parallel downwards then at some point it will cease to intersect the paraboloid Q.
Let us call last point it touches $(a, b, a^2 + b^2)$, then it is the tangent plane T in this point and $T' = T_r$ for some shift amount r^2.
Since $\triangle$ is a lower face of the convex hull in 3-dimensions, all the other points p'_n of P' lie above T' and hence they are more than r^2 above T. Therefore, these points project to the xy-plane outside of the circle of radius r with center (a, b) in the xy-plane.
On the other hand, the three points $\{p'_i, p'_j, p'_k\}$ lie on $Q \cap T'$ and so they are projected down to points on this circle. Therefore, the triangle $\triangle(p_i, p_j, p_k)$ in the xy-plane is circumscribed by the circle with center (a, b) and radius r which contains no other points from P either in its interior or on its boundary. Therefore, the triangle $\triangle(p_i, p_j, p_k)$ is part of the Delaunay triangulation. Repeating this procedure for the other faces of the 3-dimensional complex hull we have proved :

Theorem 3.2. (Edelsbrunner-Seidel) *The Delaunay triangulation of a set of points $P = \{p_1, \ldots, p_z\}$ in the plane is precisely the projection to the xy-plane of the bottom shell of the convex hull in $\mathbb{R}^3$ of the transformed points $P' = \{p'_1, \ldots, p'_z\}$, transformed by mapping upwards to the paraboloid Q with defining equation $z = x^2 + y^2$.*

Exercises 3.

1. Design a set of n points, no four cocircular, such that one vertex of the Delaunay triangulation has degree $n-1$.
2. A triangulation of a set of plane points P is called a *Pitteway triangulation* if, for each triangle $T = \triangle(a,b,c)$ every point in T has one of a, b or c as its nearest neighbor among the points of P.

   Show by example that not every Delaunay triangulation is a Pitteway triangulation.

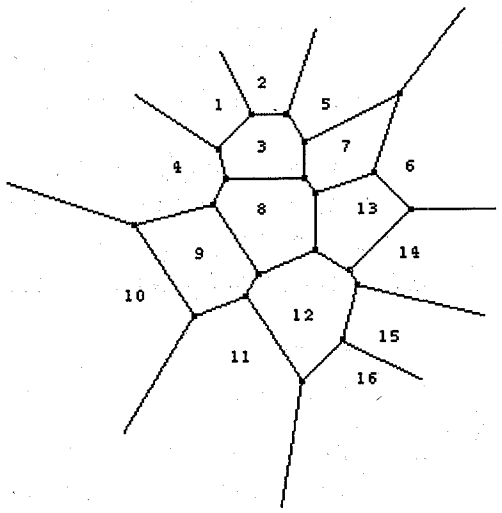
Homework 3.

1. ● : If p_i is a nearest neighbor of p_j among the set of plane points P, then the edge p_ip_j is an edge of the Delaunay triangulation.
2. ● ● ● : Characterize those Delaunay triangulations that are also Pitteway triangulations (see above for definitions).
3. ● ● ● ● : Given two sets of plane points A and B. Design an algorithm for finding (if it exists) a closed disc that encloses every point of A but excludes every point of B.

Week 4

VoronoiDiagram.

```
In[1]:=<<DiscreteMath`ComputationalGeometry`

In[2]:=data2D={{4,14},{6,15},{6,12},{2,11},{9,14},
{13,11},{10,12},{6,9},{3,7},{0,5},{5,2},
{8,4},{11,9},{13,7},{12,3},{11,1}};

In[3]:=voronoi=VoronoiDiagram[data2D]

Out[3]:= \!\({{{\(-0.071428571428571428571428`13.17\),
        8.357142857142857142857142`14.8165}, {
        2.815789473684210526315787`14.8051,
        4.026315789473684210526314`14.6786}, {
        3.642857142857142857142857`14.4056,
        9.285714285714285714285714`14.4457}, {
        3.899999999999999999999973`14.3786,
        11.899999999999999999999999`14.4113}, {4.25`14.2289, 10.5`14.2233}, {
        5.184210526315789473684209`14.606,
        4.973684210526315789473680`14.5937}, {5.5`14.1964, 13.5`14.1879}, {
        5.815789473684210526315787`14.6115,
        6.026315789473684210526315`14.5202}, {
        7.166666666666666666666667`14.0831, 13.5`14.096}, {7.9`14.8458,
        0.899999999999999999999999`14.0979}, {
        7.999999999999999999999996`14.4743,
        10.499999999999999999999994`14.4809}, {8.`14.2892, 12.25`14.2648}, {
        8.499999999999999999999978`14.4483,
        9.833333333333333333333332`14.4232}, {
        8.499999999999999999999992`14.6348,
        7.099999999999999999999987`14.6394}, {
        9.833333333333333333333333`14.3591,
        2.833333333333333333333332`14.2424}, {10.125`14.3744,
        6.125000000000000000000011`14.3245}, {
        10.499999999999999999999992`14.6066,
        5.499999999999999999999993`14.5054}, {11.25`14.1809,
        10.75`14.1748}, {12.49999999999999999966569`14.0531,
        14.49999999999999999984638`14.0272}, {13.`14.1935, 9.`14.1805},
      Ray[{5.5`14.1964, 13.5`14.1879}, {4.`12.6894, 16.5`13.001}],
      Ray[{3.899999999999999999999973`14.3786,
          11.899999999999999999999999`14.4113}, {\(-4.`-10.5544*^-24\),
          14.500000000000000000000028`15.5458}],
      Ray[{7.166666666666666666666667`14.0831, 13.5`14.096}, {
          8.49999999999999992598513`12.7843,
          17.49999999999999997779554`12.6192}],
      Ray[{\(-0.071428571428571428571428`13.17\),
          8.357142857142857142857142`14.8165}, {
          \(-6.071428571428571428571425`14.7575\),
          10.357142857142857142857141`15.2811}],
      Ray[{12.49999999999999999966569`14.0531,
          14.49999999999999999984638`14.0272}, {
          15.49999999999999999996075`13.0641,
          18.49999999999999999987943`13.0131}],
      Ray[{13.`14.1935, 9.`14.1805}, {17.`13.3668, 9.`15.3648}],
      Ray[{2.815789473684210526315787`14.8051,
          4.026315789473684210526314`14.6786}, {
          \(-0.500000000000000000041624`13.2817\),
          \(-1.500000000000000000004161`13.5109\)}],
      Ray[{7.9`14.8458, 0.899999999999999999999999`14.0979}, {
          6.899999999999999999999998`13.3781,
          \(-5.100000000000000000000001`12.4662\)}],
      Ray[{10.499999999999999999999992`14.6066,
          5.499999999999999999999993`14.5054}, {
          16.499999999999999999999996`13.9418,
          4.000000000000000000000002`13.8935}],
    Ray[{9.8333333333333333333333333`14.3591,
       2.8333333333333333333333332`14.2424}, {
       13.4999999999999940788105`13.5078,
       1.0000000000000002960594 75`12.671}]}, {{1, {4, 7, 21, 22}}, {
      2, {7, 9, 23, 21}}, {3, {7, 4, 5, 11, 12, 9}}, {
      4, {1, 3, 5, 4, 22, 24}}, {5, {9, 12, 19, 25, 23}}, {
      6, {19, 18, 20, 26, 25}}, {7, {12, 11, 13, 18, 19}}, {
      8, {5, 3, 8, 14, 13, 11}}, {9, {1, 2, 6, 8, 3}}, {
      10, {2, 1, 24, 27}}, {11, {10, 6, 2, 27, 28}}, {
      12, {8, 6, 10, 15, 17, 16, 14}}, {13, {13, 14, 16, 20, 18}}, {
      14, {20, 16, 17, 29, 26}}, {15, {17, 15, 30, 29}}, {
      16, {15, 10, 28, 30}}}}\)

In[4]:=DiagramPlot[data2D]
```



```
VoronoiDiagram[{{x1,y1},{x2,y2},...}] computes the
Voronoi diagram of a set of points in the plane.
```

The Voronoi diagram of a collection $P = \{p_1, \ldots, p_n\}$ in the plane is the collection of 'nearest neighborhoods' for each of the points in P. That is we partition the plane by assigning each point in the plane to the nearest point in P. All the points assigned to p_i form the *Voronoi region* $V(p_i)$. Its formal definition is

$$V(p_i) = \{p \in \mathbb{R}^2 : d(p_i, p) \leq d(p_j, x) \quad \forall j \neq i\}$$

Each of these regions is closed. Some points in the plane do not have a unique nearest *neighbor*. The set of all points that have more than one nearest neighbor form the *Voronoi diagram* $\mathcal{V}(P)$ of the set of points P. While DelaynayTriangulate need only specify the connections between points, VoronoiDiagram must specify both a set of diagram vertices and the connections between those vertices. Another difference between the

two functions is that while a triangulation consists of segments, a Voronoi diagram consists of both segments and rays.
These considerations make the output of VoronoiDiagram more complex than that of DelaunayTriangulation. The diagram is given as a list of diagram vertices followed by a diagram vertex adjacency list. The finite vertices of the diagram are listed first in the vertex list. The vertices lying at infinity have head Ray and are listed last.
Consider two points p_i and p_j from P and let B_{ij} be the perpendicular bisector of the segment p_ip_j. Let $H(p_i,p_j)$ be the closed halfplane with boundary B_{ij} and containing p_i. Elementary geometry then tells us that

$$V(p_i) = \bigcap i \neq jH(p_i,p_j)$$

where the intersection is taken over all $1 \leq j \leq n$ such that $i \neq j$. In particular this shows :

Proposition 4.1. *The Voronoi regions $V(p_i)$ of a finite set of points P all have polygonal boundaries.*

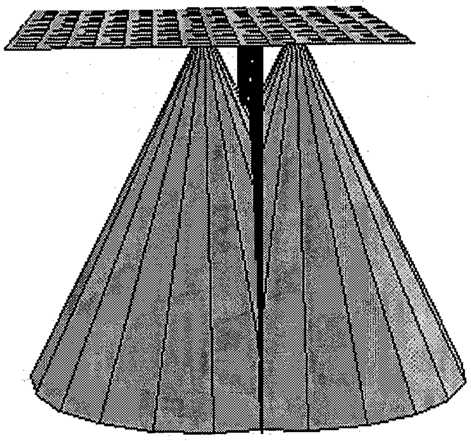
This gives already an algorithm to construct the Voronoi diagram by constructing each Voronoi region separately by intersecting $n-1$ halfplanes. This algorithm has a complexity of $O(n^2\ log\ n)$.

4a Fortune.

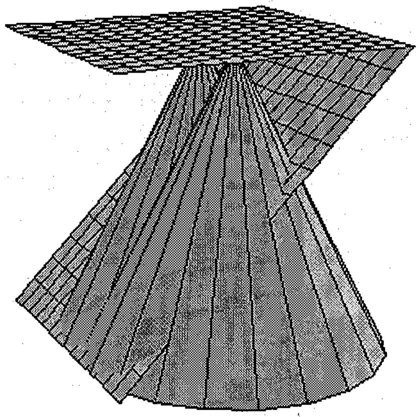
In 1985, S. Fortune invented a clever plane-sweep algorithm to construct VoronoiDiagram with complexity $O(n\ log\ n)$. Plane-sweep algorithms pass a sweep line over the plane, leaving at any time the problem solved for the portion of the plane already swept, and unsolved for the portion not yet reached. At first this seems impossible as Voronoi edges of a region $V(p)$ would be encountered by the sweep line L *before* L encounters the point p responsible for the region. S. Fortune surmounted this difficulty by the following clever idea.

Imagine the points lying on the xy-plane in $\mathbb{R}^3$ and construct for each point $p_i \in P$ a cone with apex p_i and whose side slopes are at 45^o. If $-z$ is viewed as time, then the cone under p represents a circle around p expanding about p at unit velocity : after t units of time, its radius is t.
Consider two nearby cones under p_1 and p_2. They intersect in a curve in space which lies entirely in a vertical plane which is the vertical plane on the bisector of p_1p_2. Thus, although the intersection of the cones is curved

in $\mathbb{R}^3$, it projects to a straight line in the xy-plane

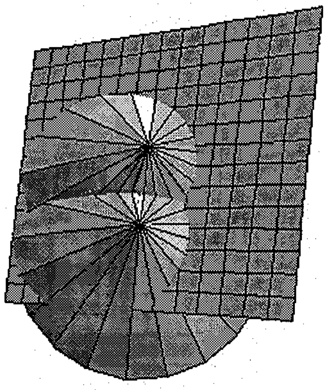


Now we come to the main idea of Fortune's algorithm. His algorithm sweeps the cones with a slanted plane π, slanted at 45^o to the xy-plane. the sweep line L is then the intersection of π with the xy-plane.



Let us assume that L is parallel to the y-axis and that the x-coordinate of L is l. Imagine that π as well as all the cones are opaque and view the configuration from $z = +\infty$.
To the $x > l$ side of L only π will be visible from above as it cuts above the xy-plane. To the $x < l$ side of L, the Voronoi diagram is visible up to the intersection of π with the 'frontier' of cones. The intersection of π with any of the cones is a parabola and so the intersection of π with the right frontier projects to the xy-plane as a 'parabolic front' a curve composed of pieces of parabolas.
Two parabolas join at a spot where π meets two cones and from the discussion above on the intersection of two cones, this must be a Voronoi edge. So the sweeping algorithm does not precisely construct the Voronoi diagram

at all times to the left of L, but it is at all times constructed *above* π, which means that it is constructed to the left of L up to the parabolic front which lags L a bit.



What is maintained at all times by the algorithm is the parabolic front, whose joints trace out the Voronoi diagram over time, since these kinks all lie on Voronoi edges. The algorithm only needs to store the parabolic front which is of size $O(n)$ and often of size $O(\sqrt{n}$. This is a significant advantage of Fortune's algorithm when n is large : the storage needed at one time is often much smaller than the size of the diagram.

Exercises 4.

1. Describe the Voronoi diagram for the vertices of a regular polygon.
2. Describe the number of combinatorial different Voronoi diagrams of n vertices for $n \leq 4$.
3. Prove that the number of edges in a Voronoi polygon, averaged over all polygons in any set of n points, does not exceed 6.

Homework 4.

1. ● : A one-dimensional Voronoi diagram for a set of points $P = \{p_1, \ldots, p_n\}$ on a line is a set of points $V(P) = \{x_1, \ldots, x_{n-1}\}$ such that x_i is the midpoint of $p_i p_{i+1}$.

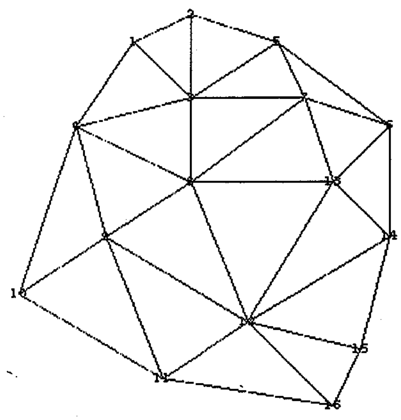
   Given a set $X = \{x_1, \ldots, x_{n-1}\}$, design criteria to decide whether or not X is a one-dimensional Voronoi diagram of a set of points P, and if so determine P.
2. ● ● : Prove that $V(p_i)$ is unbounded if and only if p_i is on the convex hull of the set of points P.
3. ● ● ● ● : Imagine a set of n points moving in the plane, each with a fixed velocity and direction. Let $\mathcal{V}(t)$ be the Voronoi diagram of the points at time t. Find a set of n moving points such that $\mathcal{V}(t)$ changes its graphical structure n^2 times.
4. ● ● ● ● ● : (open) As in the previous exercise, find a set of n moving points that have more than n^2 changes in the graphical structure of $\mathcal{V}(t)$.

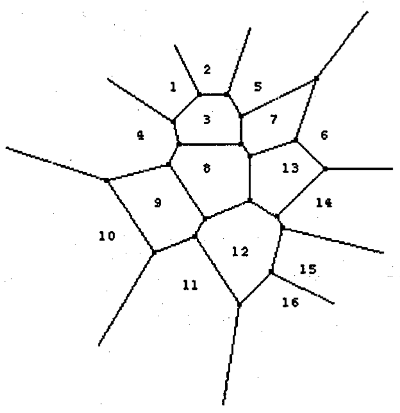
Week 5

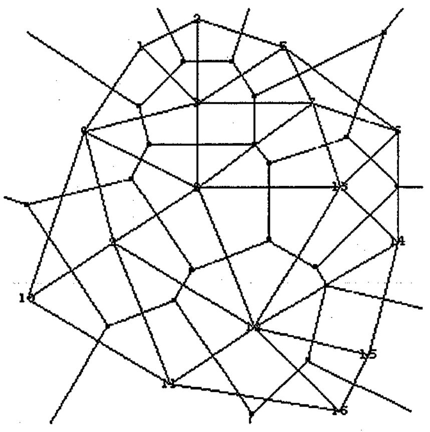
Delaunay vs. Voronoi.

Consider a finite set $P = \{p_1, \ldots, p_n\}$ of points in the plane. In 1934, Delaunay proved that when the dual graph of the VoronoiDiagram $\mathcal{V}(P)$ is drawn with straight lines, it produces a planar triangulation of P which is now called the DelaunayTriangulation $\mathcal{D}(P)$.

```
In[1]:=<<DiscreteMath`ComputationalGeometry`

In[2]:=data2D={{4,14},{6,15},{6,12},{2,11},{9,14},
{13,11},{10,12},{6,9},{3,7},{0,5},{5,2},
{8,4},{11,9},{13,7},{12,3},{11,1}};

In[3]:=delaunay=DelaunayTriangulation[data2D];

In[4]:=Show[PlanarGraphPlot[data2D,delaunay],DiagramPlot[data2D]]
```







5a Properties of DelaunayTriangulation.

As the DelaunayTriangulation and the VoronoiDiagram are dual structures, each contains the same information, but represented in a different format. We will now list the more properties of the DelaunayTriangulation for a fixed set $P = \{p_1, \ldots, p_n\}$.

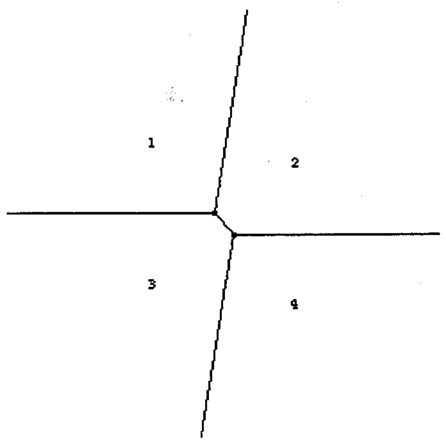
1. $\mathcal{D}(P)$ is the straight-line dual of $\mathcal{V}(P)$.
2. $\mathcal{D}(P)$ is a triangulation if no four points of P are cocircular (Delaunay's theorem). In this case every face is a triangle and they are called *Delaunay triangles*.
3. Each face of $\mathcal{D}(P)$ corresponds to a vertex of $\mathcal{V}(P)$.
4. Each edge of $\mathcal{D}(P)$ corresponds to an edge of $\mathcal{V}(P)$.
5. Each vertex of $\mathcal{D}(P)$ correspond to a region in $\mathcal{V}(P)$.
6. The boundary of $\mathcal{D}(P)$ is the convex hull of P.
7. The interior of each face of $\mathcal{D}(P)$ contains no point of P.
8. If p_j is the nearest neighbor of p_i in P, then p_ip_j is an edge of $\mathcal{D}(P)$.
9. The edge p_ip_j is an edge of $\mathcal{D}(P)$ iff there exists a circle through p_i and p_j such that the closed disc bounded by the circle contains no points of P other than p_i and p_j.

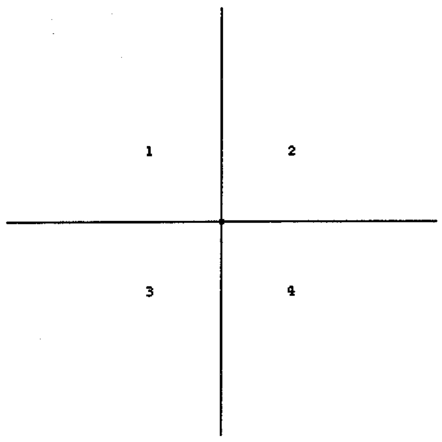
5b Properties of VoronoiDiagram.

Similarly, we can give a formal list of the main properties of the VoronoiDiagram of a fixed set $P = \{p_1, \ldots, p_n\}$ of points in the plane.

1. Each Voronoi cell is a convex polygon.
2. The region $V(p_i)$ is unbounded iff p_i is on the convex hull of P.
3. If v is a *Voronoi vertex* at the junction of $V(p_i)$, $V(p_j)$ and $V(p_k)$, then v is the center of the circle $C(v)$ determined by $p_i.p_j$ and p_k.
4. The *Voronoi circle* $C(v)$ is the circumscribed circle for the Delaunay triangle corresponding to v.
5. The interior of the Voronoi circle $C(v)$ contains no points of P.
6. If p_j is the nearest neighbor of p_i in P, then p_ip_j is an edge of $\mathcal{D}(P)$.

We will now prove the more important of these properties. Throughout we make the assumption that $P = \{p_1, \ldots, p_n\}$ contains no fourtuple of cocircular points. We glanced over this subtlety earlier. In fact, the formal definition of DelaunayTriangulation we gave before does only make sense in this case. On the level of VoronoiDiagrams this condition asserts that there are no 4 Voronoi regions touching each other in a point. That is, we only consider the situations on the left and not those on the right.





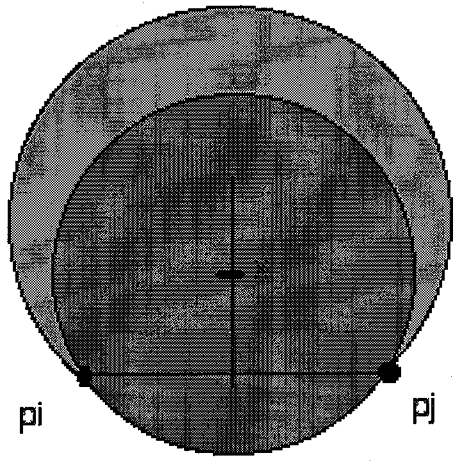
The attentive reader will have noticed that this condition was also needed in the EidelsbrunnerSeidel algorithm. Indeed, $p_i = (x_i, y_i)$ for $i = 1, 2, 3, 4$ will be cocircular if and only if the modified points $p_i' = (x_i, y_i, x_i^2 + y_i^2)$ all lie on the same plane (and of course also on the paraboloid).
Still, we can generalize the definition of a DelaunayTriangulation to the general case.

Definition 5.1. A DelaunayCovering $\mathcal{D}(P)$ of a set of points P in the plane is the straight-line dual of the VoronoiDiagram $\mathcal{V}(P)$ of P.

Theorem 5.2. *The edge $p_ip_j \in \mathcal{D}(P)$ iff there is a circle through p_i and p_j such that the closed disk bounded by this circle contains no other points than p_i and p_j.*

Proof. If p_ip_j is an edge of $\mathcal{D}(P)$ then $V(p_i)$ and $V(p_j)$ share an edge $e \in \mathcal{V}(P)$ of positive length (the definition of the dual graph). Let x be an interior point of this edge e and let $C(x)$ be a circle with center x and radius equal to the distance $d(x, p_i)$ (which is equal to the distance $d(x, p_j)$ because $x \in V(p_i) \cap V(p_j)$).
There can be no other point $p \in P$ lying in or on $C(x)$ for otherwise $x \in V(p)$ as well and by taking x an interior point of e we made sure that x belongs only to the Voronoi regions $V(p_i)$ and $V(p_j)$.

Conversely, assume that there is a circle $C(x)$ with center x going through p_i and p_j and empty of other points from P. We have to prove that $p_ip_j \in \mathcal{D}(P)$. Because x is equidistant from p_i and p_j we have that $x \in V(p_i) \cap V(p_i)$ for p_i cannot belong to any other $V(p)$ for the disc closed by $C(x)$ does not contain p. Because there are no other points on the boundary of $C(x)$ (which amounts to the fact that x is not the crossing of two Voronoi edges) we have some freedom to wiggle x a bit along the bisector B_{ij} while maintaining emptiness of the circles.



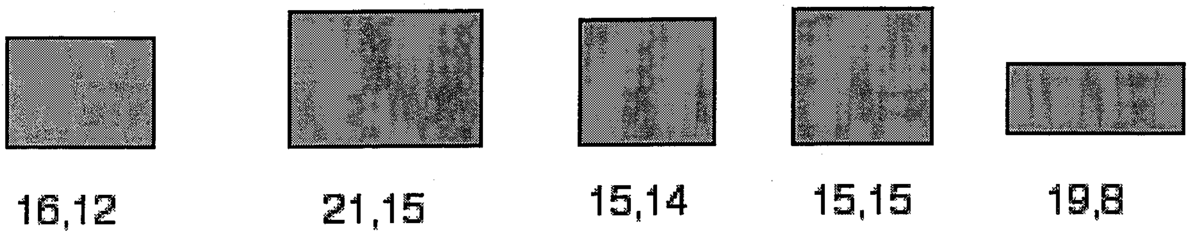
Therefore, x lies on a positive length Voronoi edge (a certain subset of the bisector B_{ij}) shared between $V(p_i)$ and $V(p_j)$ and therefore p_ip_j is a straight-line dual edge in $\mathcal{D}(P)$. □

Using similar arguments, the other properties of DelaunayTriangulation and of VoronoiDiagram can be proved as exercises. Do it!

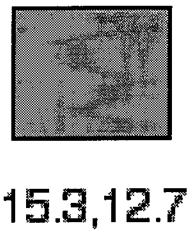
5c PatternRecognition.

We give here an immediate application (more will be presented over the next two weeks). A technique frequently employed in the field of pattern recognition is to map a set of target objects into a feature space by reducing the objects to points whose coordinates are feature measurements. The identity of an object of unknown affiliation then can be assigned the nearest target object in a feature space.

Let us give a concrete example. Assume we have a fixed set of rectangular objects of five different flavors



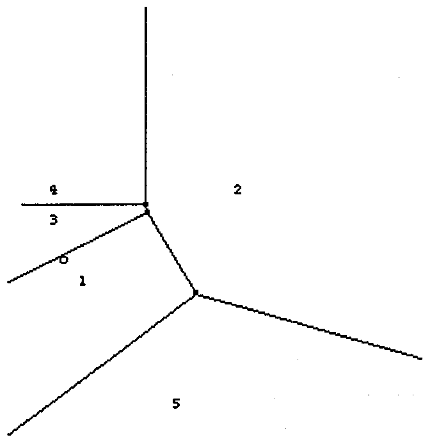
where we indicated the length of the longest and of the shortest side. Suppose that a vision system focuses on a particular object and measures the sizes of the two sides to be 15.3 and 12.7.



Knowing that there are always measurement inaccuracies, how would we make the best guess of the flavor of the object ? We can represent all flavors by points in the plane, namely

$$\{(16,12),(21,15),(15,14),(15,15),(19,8)\}$$

and determine the VoronoiDiagram of this set of points



where we indicated the location of the measurement by *o*. Therefore, it is most likely an object of type $(16,12)$.

Week 6

Applications.

This week we present some applications of DelaunayTriangulation and VoronoiDiagram.

6a LargestEmptyCircle

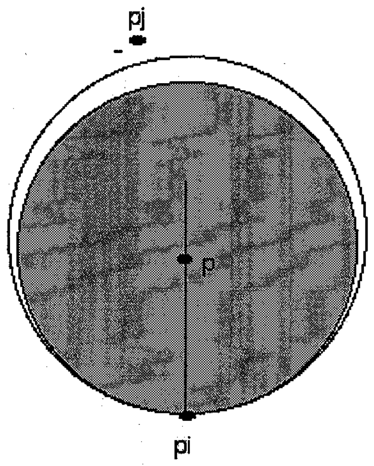
Suppose you would like to locate a new grocery store in an area with several existing, competing grocery stores. assuming uniform population density, where should the new store be located to optimize its sales ?
One natural method would be to choose a location whose distance to the nearest store is as large as possible. This is equivalent to locating the new store at the center of the largest empty circle, the largest circle whose interior contains no other stores. The distance to the nearest store is then the radius of this circle.

Another application is to locate a nuclear reactor as far away as possible from a collection of cities. We now examine the largest empty circle problem in some detail. First, we need to put restrictions on the center of this largest empty circle as there are always arbitrarily large circles outside a finite set of points. We rephrase the problem as :

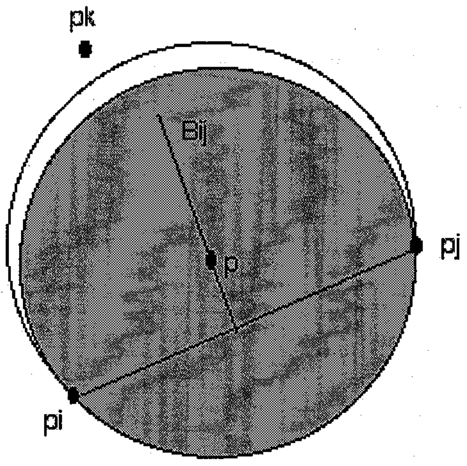
`LargestEmptyCircle`
Find a largest empty circle whose center is in the closed convex hull of a set P of n points in the plane. Empty in the sense that it contains no points from P and largest in the sense that there is no other such circle with strictly larger radius.

Let p be a point in the plane and imagine inflating a circle from p until it first bumps into one of the finite list of points $P = \{p_1, \dots, p_n\}$. Call $f(p)$ the

radius of this circle which we denote by $C(p)$.
If there is just one p_i on the boundary of $C(p)$, then $f(p)$ cannot be a maximum because if p is moved along the ray p_ip away from p_i a bit we get a strictly larger $f(p')$.



If there are precisely two points p_i and p_j from P on the boundary of $C(p)$, then $f(p)$ can also not be a maximum. For if p is moved along the bisector B_{ij} a bit away from p_ip_j we get a strictly larger circle.



It is only when there are three points p_i, p_j and p_k on the boundary of $C(p)$ that the radius $f(p)$ could be at a maximum. This proves

Lemma 6.1. *If the center p of a LargestEmptyCircle is strictly interior in the convex hull of P, then p must be a Voronoi vertex (that is, a crossing point of three of more boundaries of Voronoi regions).*

The problem with points p lying on the convex hull of p is that when we move to the neighboring point p' in the foregoing argument, p' may be outside the convex hull. Still, if p lies on the convex hull of P it must lie in the interior of one of the edges of $ConvexHull(P)$. Assume the boundary of $C(p)$ contains just one p_i then $f(p)$ cannot be maximal as we could move p a little bit on this boundary edge to increase the distance to p_i without hitting new points from P. But, if $C(p)$ contains two points p_i and p_j it may very well be that in order to increase the diameter we have to move outside of the convex hull. That is, we have :

Lemma 6.2. *If the center* p *of a LargestEmptyCircle lies on the convex hull of* P*, then* p *must lie on a Voronoi edge.*

We have now found a finite set of points that are potential centers of LargestEmptyCircles,

- the Voronoi vertices,
- intersections of Voronoi edges with the ConvexHull(P).

and we have the following algorithm.

```
LargestEmptyCircle

Compute the Voronoi diagram of P.
Compute the ConvexHull(P) = H.
for each Voronoi vertex v do
    if v is inside H then
       compute radius of C(v) and update max.
for each Voronoi edge e do
    compute intersection p of e with H.
    compute radius of C(p) and update max.
return max.
```

6b MinimumSpanningTree.

MinimumSpanningTree
Find the *minimum spanning tree* of a set of points $P = \{p_1, \ldots, p_n\}$ in the plane. That is, a minimum length tree that spans all the points of P : a shortest tree whose nodes are precisely those in P.

For example, many local area networks take the form of a tree spanning the host nodes. The MinimalSpanningTree is the network topology that minimizes total wire length, which usually minimizes both cost and time delays.

First construct the *complete graph* on P, that is we connect each couple of points p_i, p_j from P with a straight line in the plane. The algorithm is based on the intuition that a minimal spanning tree should be composed of the shortest edges. This suggests that such a tree can be built up incrementally by adding the shortest edge not yet explored subject to the condition that is maintains tree-ness (acyclicity of the graph).
Let T be the tree incrementally constructed and let the notation $T+e$ mean the tree T union the edge e. In 1956 Kruskal gave the following algorithm to solve the MinimalSpanningTree problem.

```
Kruskal

sort all edges of G by length : e1,e2,...
initialize T to be empty, i <- 1.
while T is not spanning do
   if T+ei is acyclic
      then T <- T+ei
   i <- i+1
```

As there are $\binom{n}{2}$ edges in the complete graph on n points, the complexity of the sorting step is $O(n^2 \, log \, n)$. The following important result reduces this complexity to $O(n \, log \, n)$.

Theorem 6.3. *A minimal spanning tree on a set of points P in the plane is a subset of the edges of the Delaunay triangulation of P.*

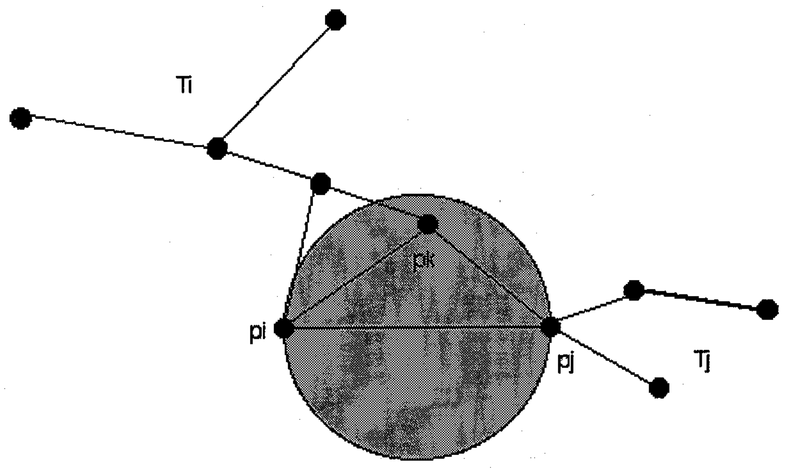
Proof. Assume that the edge p_ip_j is in the MinimalSpanningTree but is *not* an edge of the DelaunayTriangulation. We have seen before that an edge belongs to the Delaunay triangulation if and only if there is an empty circle through p_i and p_j. Therefore, there is no empty circle through p_i and p_j.
In particular, the circle with diameter p_ip_j must contain a point p_k in it or on its boundary. But then, we have

$$d(p_i,p_k) < d(p_i,p_j) \quad \text{and} \quad d(p_j,p_k) < d(p_i,p_j).$$

If we remove the edge p_ip_j from the MinimalSpanningTree T, then T disconnects in two trees T_i containing p_i and T_j containing p_j.
Assume without loss of generality that the point p_k lies in tree T_i. Now, form the new tree

$$T' = T_i + p_kp_j + T_j$$

that is, we have the following situation :



Now, T' is again a connected tree spanning all points but as $d(p_k, p_j) < d(p_i, p_j)$ is has strictly smaller length. That is, we have a contradiction and therefore the edge $p_i p_j$ must be an edge in the DelaunayTriangulation. □

Therefore, computing first DelaunayTriangulation of P before applying the sorting part of the Kruskal algorithm will drastically decrease the time needed for large n.

Week 7

GameOfLife.

The `GameOfLife` is played on an infinite squared board. At any time some of the cells will be *live* and others *dead*. The next cycle the status of the cells is determined by the following rules :

- `birth :` a cell that is dead at time t becomes live at $t+1$ only if *exactly three* of its neighbors were live at t,
- `death by overcrowding :` a cell that is live at t and has *four or more* of its eight neighbors live at t will be dead by time $t+1$,
- `death by exposure :` a live cell that has *one or less* live neighbors at time t will be dead by time $t+1$.

The `GameOfLife` is an important example of a class of mathematical objects called *cellular automata*. A cellular automaton consists of a number of things

1. A positive integer n which is the *dimension* of the cellular automaton. $n = 2$ in `GameOfLife`.
2. A finite set S of *states* having at least two elements. $S = \{\texttt{live}, \texttt{dead}\}$ in `GameOfLife`.
3. A *state* for the whole cellular automaton is obtained by assigning an element of S to each point of the n-dimensional *lattice* $\mathbb{Z}^n$ (where $\mathbb{Z}$ are the integers). The points of $\mathbb{Z}^n$ are usually called *cells*.
4. A definition of *neighborhood*. The neighborhood N of the origin is some finite non-empty subset of $\mathbb{Z}^n$. the neighborhood of any other cell is obtained in the obvious way by translating that of the origin. In `GameOfLife` the neighborhood of a cell (denoted with o) consists of the 8 cells denoted x together with o

```
.....
.xxx.
.xox.
.xxx.
.....
```

5. A *transition rule* which is a specified function

$$S^N \longrightarrow S$$

that is, for each possible state of the neighborhood N the transition rule specifies the state of the cell. In `GameOfLife` the transition function is given by

$$\{\texttt{live}, \texttt{dead}\}^9 \longrightarrow \{\texttt{live}, \texttt{dead}\}$$

specified by sending $(s_0; s_1, \ldots, s_8)$ where s_0 is the state of the middle cell to

$$\begin{cases} \texttt{live} & \text{iff} \quad s_0 = \texttt{live} \text{ and } \#\{s_i = \texttt{live}\} = 2 \text{ or } 3 \text{ or} \\ & \phantom{\text{iff}} \quad s_0 = \texttt{dead} \text{ and } \#\{s_i = \texttt{live}\} = 3. \\ \texttt{dead} & \text{iff} \quad s_0 = \texttt{live} \text{ and } \#\{s_i = \texttt{live}\} \leq 1 \text{ or } \geq 4 \text{ or} \\ & \phantom{\text{iff}} \quad s_0 = \texttt{dead} \text{ and } \#\{s_i = \texttt{live}\} \neq 3 \end{cases}$$

The state of the cellular automaton evolves in discrete time, with the state of each cell at time $t+1$ being determined by the state of its neighborhood at time t in accordance with the transition rule.

In order to appreciate the richness of `GameOfLife` one has to experiment with it. Fortunately, there are several excellent Java-applets and PC- or Mac-shareware available on the net. A good starting point is

`http://www.cs.jhu.edu/~callahan/lifepage.html`

Another excellent source of information on `GameOfLife` is the 100p+ "Life Lexicon" compiled by Stephen A. Silver which can be found at

`http://www.cs.jhu.edu/~callahan/lexiconf.html`

We will briefly indicate some of the more mathematically interesting notions and results. We expect that you experiment with `GameOfLife`.

7a Oscillators.

The *blinker* is the simplest example of a configuration whose life history repeats itself with period > 1

```
...      .o.      ...      .o.
ooo      .o.      ooo      .o.
...      .o.      ...      .o.
```

Such positions are called *oscillators*. A cellular automaton is said to be *omniperiodic* if it has oscillators of all periods. At this moment it is not known that `GameOfLife` is omniperiodic although there is good evidence that it must be. If you want to do some Life-research, at this moment the only periods for which no oscillator is known are

`19,23,27,31,37,38,41,43,49 and 53`

For example, Dave Buckingham discovered the *burloaferimeter* which has period ~~5~~ 7

```
.....o.....
....o.o....
...o.o.o...
...o.o.o...
oo.o...o.oo
oo.o....o.o
....oooo...
...........
....oo.....
....oo.....
```

[illegible] and a simple period 100 position called *centinal* was found by Bill Gosper

```
oo................................................oo
.o................................................o.
```

7b SpaceShips.

A *SpaceShip* is a finite pattern that reappears (without additions or losses) after a finite number of generations and displaced by a non-zero amount. One of the most common spaceships that occurs during random experiments is the *glider* which was found by Conway's group in 1970

```
ooo
o..
.o.
```

It is known that there exists spaceships travelling in all rational directions and at arbitrarily slow speeds. They come under various pretty names such as : brain, dart, ecologist, flotilla, fly, snail, swan ... Here a picture of the *spider* which was found by David Bell in 1997

```
......o...ooo.....ooo...o......
...oo.ooooo.oo...oo.ooooo.oo...
.o.oo.o.....o.o.o.o.....o.oo.o.
o...o.o...ooooo.ooooo...o.o...o
....ooo.....oo...oo.....ooo....
.o..o.ooo.............ooo.o..o.
...o.......................o...
```

The theoretical importance of SpaceShips is that they can be used to make GameofLife into a universal Turing machine. This was proved by Bill Gosper and John Conway and the proof is outlined in

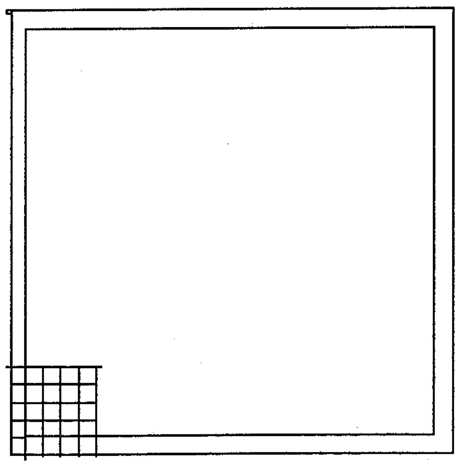
```
E.R. Berlekamp, J.H. Conway and R. Guy
"Winning ways for your mathematical plays, volume 2"
```

7c GardenOfEden.

There are GameOfLife configurations that can arise only as the initial state, because they have no ancestors. Such configurations are called *garden of Eden configurations*.

Theorem 7.1. *Garden of Eden configurations must exist in* `GameOfLife`.

Proof. Consider a $5n \times 5n$ square B and divide it into n five by five squares as below



Consider the inscribed $5n-2 \times 5n-2$ square S. On S there are exactly

$$2^{(5n-2)^2} = 2^{25n^2-20n+4}$$

different GameOfLife configurations. If such a generation c has an ancestor p, then the rules imply that p must be a configuration constrained to B. If one of the n^2 little 5×5 squares is empty we can replace it by

```
.....         .....
.....         .....
.....   ->    ..o..
.....         .....
.....         .....
```

as this does not affect the next generation. Therefore, there are at most $2^{25}-1$ relevant positions in the 5×5 squares to consider. This gives a maximum bound of

$$(2^{25}-1)^{n^2} = 2^{24.9999999570043366436125280757383320943...n^2}$$

possible ancestor-configurations in B. Now, we have to find an n such that the number of ancestors is strictly smaller than that of the configurations in S.

$$24.9999999570043366436125280757383320943...n^2 < 25n^2 - 20n + 4$$

This quadratic equation has two approximate roots

$$\alpha_1 \sim 0.20000000008599 \quad \text{and } \alpha_2 \sim 4.65163191590336296\ 10^8$$

therefore we can take $n = 465163192$ and there is a garden of Eden configurations lying in

$$2325815958 \times 2325815958$$

square ! $\square$

More careful counting reduces the size down to a square of side 1400. In 1971 a first explicit example was constructed by Roger Banks et al. at MIT. Their example of a garden of Eden configuration lives on a 9×33 square. At this moment, the smallest known garden of Eden configuration (in terms of the number of live cells) is the following having 143 lives

```
OO.O.O.O.OO.O.
O.OOO.OOO.OO.O
OOOO.OOO.OO.O.
OOO.O.O.O.OOOO
.OOO.O.OOO.OO.
OOOOOOO.OOOO.O
.O.O.OOOOOOOO.
O.OOO.OO.O.O.O
OOOOOO.OOOOOO.
O.OO.OOOOO.O.O
OOO.OOOOOOOOO.
.OOO.O.O.O.OOO
OOO.O.O.O.OO.O
O.OOOOOOOOOOOO
```

Week 8

StillLife.

If you experiment with GameOfLife you find that the terminal positions consist of a few oscillators (usually of period 2) and some stable configurations (or oscillators of period one). It is therefore important to classify/study such stable configurations.

In `GameOfLife` a configurations is called `StillLife` is it is left unchanged under the transition rules. Specifically, a `StillLife` is a subset S of $\mathbb{Z}^2$ satisfying the following three conditions :

- No element of $\mathbb{Z}^2 - S$ has exactly three neighbors in S,
- Every element of S has at least two neighbors in S,
- Every element of S has at most three neighbors in S.

8a Enumerating StillLife.

For the purposes of enumerating StillLifes the definition is unsatisfactory because any pair of smaller still lifes separated by some space is a StillLife. As the *block*

```
....
.oo.
.oo.
....
```

is a 4-cell StillLife there would therefore be an infinite number of 8-cell StillLifes. For this reason a stricter definition is often used, counting a stable pattern as a StillLife only if its *islands* cannot be divided into two nonempty sets both of which are stable in their own right. A *polyplet* is a finite collection of orthogonally or diagonally connected cells. An *island* is

a connected component of a StillLife in the polyplet-topology, that is, those parts of a StillLife that a chess-king can cover without having to pass via a dead cell.
The requirement that a StillLife cannot be decomposed into two separate stable parts is a bit arbitrary as it does not rule out the possibility that it might be decomposable into more than two such patterns. For example, in 1998 Matthew Cook found the 33-cell StillLife

```
..oo.oo....
.o.o.oo....
.o.........
oo..oo.o...
o...oo.ooo.
.o........o
..o......oo
...o.oo.o..
....oo.oo..
```

that can be broken into three stable pieces but not into two.
StillLifes have been enumerated by John Conway (4 to 7 cells), Robert Wainwright (8 to 10 cells), Dave Buckingham (11 to 13 cells), Peter Raynham (14 cells) and Mark Niemiec (15 to 24 cells). The resulting numbers are

cells	StillLifes	cells	StillLifes
4	2	13	240
5	1	14	619
6	5	15	1353
7	4	16	3286
8	9	17	7773
9	10	18	19044
10	25	19	45759
11	46	20	112243
12	121	21	273188
		22	672172
		23	1646147
		24	4051711

For a listing of these StillLifes one should consult the web-page of Mark Niemiec

`http://home.interserv.com/~mniemiec/lifepage.htm`

If you want to construct new StillLifes you should experiment with the Java-applet written by Calahan (see last week).

8b StillLife Conjecture.

After these excursions in GameOfLife it is about time to backtrack to computational geometry. We aim to present Noam Elkies' proof of the StillLife Conjecture.

The density of any infinite StillLife configuration is at most $\frac{1}{2}$.

First, it is easy to make an infinite StillLife out of any finite StillLife configuration. If the finite StillLife F is contained in a $k \times l$ square, then we can tile the plane with $k+2 \times l+2$ squares were the extra rows and columns are filled with dead cells.
There are infinite StillLifes which have *density* $\frac{1}{2}$. Let us define *density*. Let B_r be the square

$$B_r = \{(x,y) \in \mathbb{Z}^2 : | x |< r, | y |< r\}.$$

The density of a GameOfLife configuration S is defined to be the number

$$\delta(S) = \lim_{r \longrightarrow \infty} \frac{\#(B_r \cap S)}{\#B_r}.$$

Here are two examples (in fact the second theme gives an infinite number of examples) of density $\frac{1}{2}$ StillLifes

```
oooooooooo       ..oo..ooo.....oo...
..........       oo..oo...ooooo..ooo
oooooooooo       ..oo..ooo.....oo...
..........       oo..oo...ooooo..ooo
oooooooooo       ..oo..ooo.....oo...
..........       oo..oo...ooooo..ooo
```

Verify on a few examples that an infinite StillLife obtained by tiling a finite StillLife have densities $< \frac{1}{2}$.

The connection between such a density problem and the theory of Voronoi diagrams relies on the following observation.

Lemma 8.1. *Let $S \subset \mathbb{Z}^2$ be such that every $s \in S$ has a Voronoi region of finite area. If the average area of a Voronoi region is al least k, then the density of the configuration*

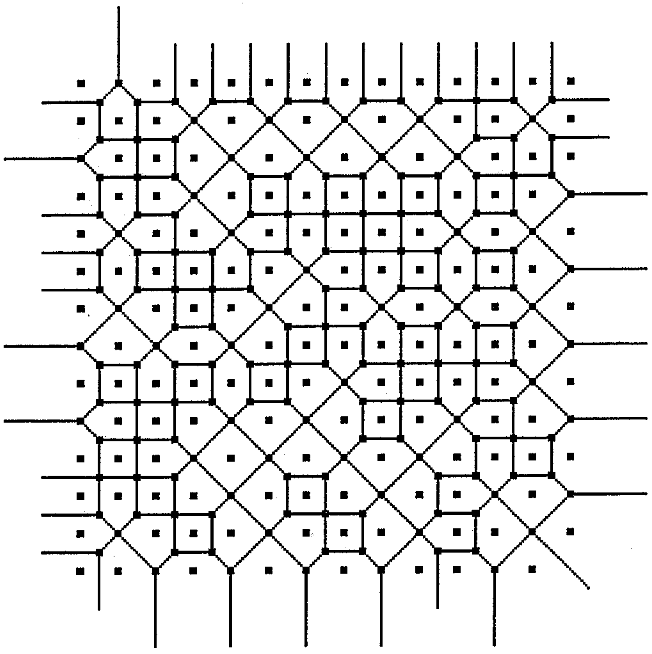
$$\delta(S) \leq \frac{1}{k}$$

Proof. Let us denote with $a(S)$ the average area of a Voronoi region of the points $s \in S$. The square B_r has area $4r^2 = \#B_r$. Further, in the limit we can also compute the area of B_r by summing over the areas of the Voronoi region $V(s)$ of the points $s \in S$ contained in B_r, that is,

$$\#B_r = 4r^2 = Area(B_r) = \sum_{s \in S \cap B_r} Area\, V(s) \approx \#(B_r \cap S)a(S)$$

from which the claim follows. $\square$

Because all the points of S lie on the square lattice $\mathbb{Z}^2$, the Voronoi cells of GameOfLife configuration have usually a very distinctive shape. For example, the Voronoi cells of the garden of Eden configuration given last week are



To avoid some rare anomalies we will slightly change the definition of the VoronoiDiagram when working with GameOfLife configurations. Remember that we defined Voronoi regions using the usual Euclidian metric in $\mathbb{R}^2$. If all our points lie on a square lattice it might be better to use *square norm* where

$$\mid (x,y) \mid_\infty = max(\mid x \mid, \mid y \mid)$$

the problem with this norm is that the Voronoi cells can have thick intersections because if two points have the same x-coordinate (or the same

y-coordinate) then their Voronoi cells have an intersection of infinite area. In order to avoid this problem we replace in this case the new halfplanes by the usual halfplanes on the Euclidian bisector. Mathematically a more elegant way is to define an *ϵ-norm* by

$$| (x,y) |_\epsilon = max(| x |, | y |) + \epsilon\, min(| x |, | y |)$$

where ϵ is a positive *infinitesimal* and then to define the Voronoi region of a point $s \in S$ as the closure of

$$V(s) = \{p \in \mathbb{R}^2 : | p - s' |_\epsilon \geq | p - s |_\epsilon, \quad \forall s' \in S\}$$

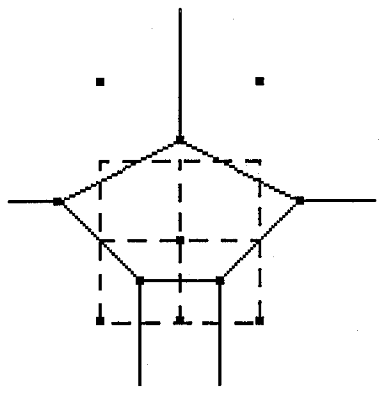
The relevance of these changes is that we will work with the *small Voronoi regions* which is the intersection of the (new) Voronoi region of a point $s = (x,y)$ with the 2×2 square bounded by the 8 neighbors of s, that is

$$V^0(s) = V(s) \cap [x-1, x+1] \times [y-1, y+1].$$

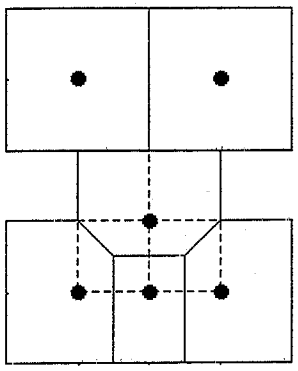
What we want is that the shape and size of these small Voronoi regions depend only on the neighbors of s and is not influenced by other points of S. A typical example where things are different is a position like

```
o.o
...
.x.
ooo
```

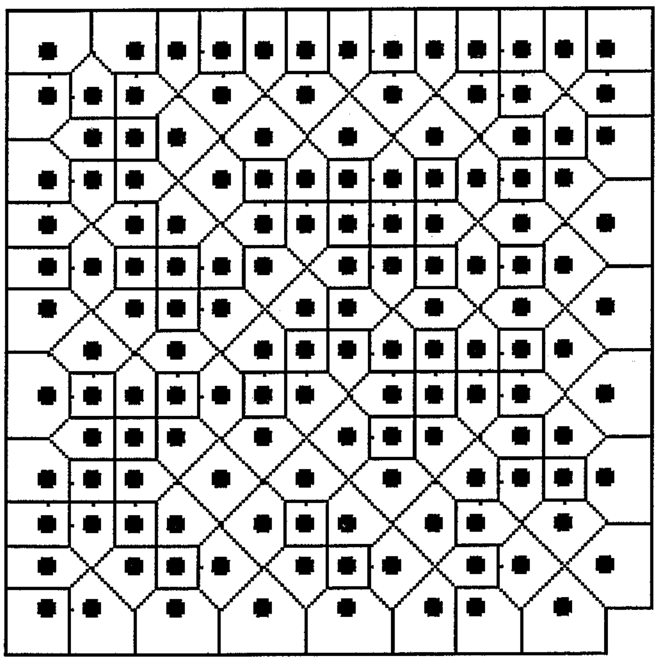
then the small Voronoi region of the point labeled x is in the usual Euclidian metric



(observe the two small cuts in the upper regions of the neighborhood due to the two cells not belonging to the neighborhood itself. In the new metric, the small Voronoi cells of the same configuration are



and we no longer have this problem. Using these small Voronoi cells we will prove a strong version of the StillLife conjecture. In dense Game-OfLife configurations, there is almost no distinction between old and new Voronoi regions or between small and big Voronoi regions. Here, the small Voronoi regions of the garden of Eden configuration. Compare with the usual Voronoi picture.



Week 9

DensityConjecture.

This week we begin the proof of a strong form of the StillLife Conjecture following the strategy with the modified small Voronoi diagrams outlined last time.

DensityConjecture.

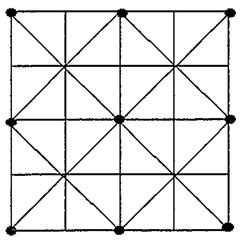
Let S be a subset of $\mathbb{Z}^2$ with the property that every element of S has *at most* three neighbors in S. Then S cannot have density greater than $\frac{1}{2}$.

9a VoronoiNeighborhoods.

First we make a list of all possible small Voronoi neighborhoods that can appear under this assumption. As mentioned before this neighborhood $V^0(s)$ depends only on the neighbors of s in S. Also, using the ϵ-norm we see that all the boundary lines of the Voronoi regions $V(s)$ are all of the form

$$2x = c \quad \text{or} \quad 2y = c \quad \text{or} \quad x \pm y = c$$

for some $c \in \mathbb{Z}$. Hence, the $V(s)$ are unions of isosceles triangles from a fixed tiling of the plane of which the fundamental neighborhood looks like



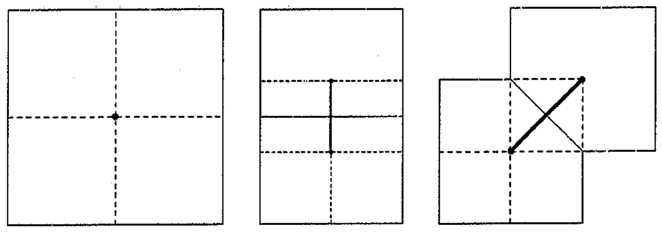
Each of these triangles has area $\frac{1}{8}$ so the areas of $V(s)$ and of $V^0(s)$ must be multiples of $\frac{1}{8}$. As we aim to prove that the average area of $V(s)$ or even of

$V^0(s)$ is ≥ 2 it makes sense to introduce the following integer to measure the deviation from our goal

$$\alpha(s) = 8.(Area\ V(s) - 2) \quad \text{and} \quad \alpha^0(s) = 8.(Area\ V^0(s) - 2).$$

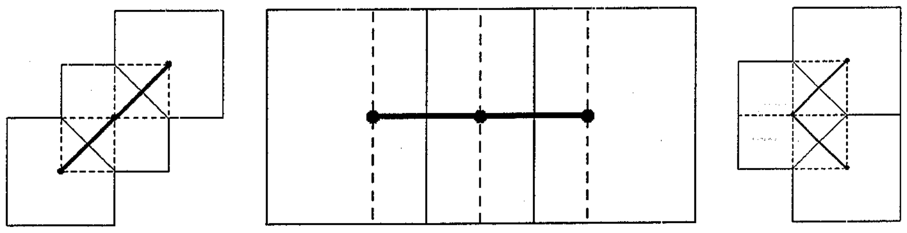
9a.1 At most one neighbor.

In the following table we list the possible small Voronoi diagrams $V^0(s)$ which is the intersection of the fundamental neighborhood (dashed lines where visible) with the region around the central point s of the fundamental neighborhood. Moreover, we indicate the neighborhood diagram (thick lines joining the center s to its neighbours) which explains the mnemotechnic name for this neighborhood and we give the value of $\alpha^0(s)$.

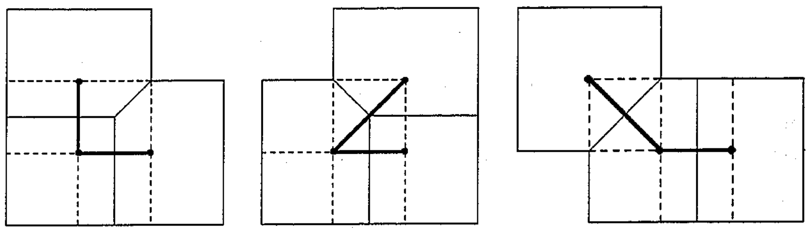


mnemo	*zero*	*side*	*halfdiagonal*
area	4	3	$3\frac{1}{2}$
$\alpha^0(s)$	16	8	12

9a.2 Two neighbors.

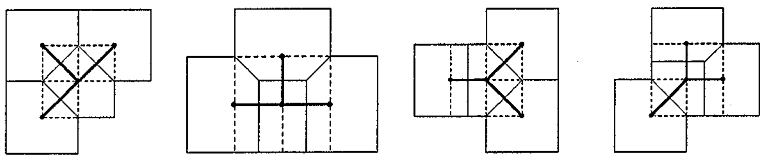


mnemo	*diagonal*	*line*	*quarter*
area	3	2	3
$\alpha^0(s)$	8	0	8

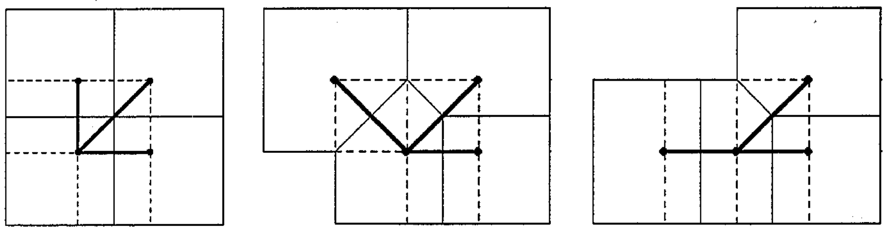


mnemo	rectangular	acute	obtuse
area	$2\frac{1}{4}$	$2\frac{7}{8}$	$2\frac{1}{2}$
$\alpha^0(s)$	2	7	4

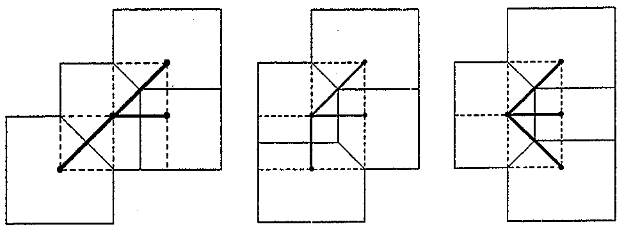
9a.3 Three neighbors.



mnemo	Diagonal	Line	Quarter	Rectangular
area	$2\frac{1}{2}$	$1\frac{1}{2}$	2	$1\frac{3}{4}$
$\alpha^0(s)$	4	−4	0	−2



mnemo	A_1cute	A_2cute	A_3cute
area	$2\frac{1}{4}$	$2\frac{3}{8}$	$1\frac{7}{8}$
$\alpha^0(s)$	2	3	−1



mnemo	$\underline{A_4}cute$	$\underline{A_5}cute$	$\underline{A_6}cute$
area	$2\frac{3}{8}$	$2\frac{1}{8}$	$2\frac{3}{4}$
$\alpha^0(s)$	3	1	6

9b VoronoiBlocks.

We see that most of the small Voronoi region have already area ≥ 2. Only those of type

$\underline{L}ine$ $\underline{R}ectangle$ and $\underline{A_3}cute$

may cause problems. The strategy in those cases is to prove that neighboring vertices of s must have types such that when averaging the total area over these local patches P we obtain ≥ 2, or equivalently,

$$\sum_{t \in P} \alpha(t) \geq 0.$$

An example will illustrate this idea. Assume we have a vertex $x \in S$ of type L, then the vertex y not lying on the big line must be of type A_6 (as no $s \in S$ can have more than three neighbors. If we then take P to be the union of the small Voronoi regions of x and y we obtain

$$\alpha(P) \geq \alpha^0(x) + \alpha^0(y) = -4 + 6 = 2 > 0.$$

As any L-vertex has a *unique* neighboring A_6-vertex (and conversely), we can pair the L's and A_3's in the local patches P and remove the L-problem.

Remains to consider the $\underline{R}ectangle$ and $\underline{A_3}cute$ vertices. Again, we will put them in good local patches but we will have more cases to consider. We will give here the easier of cases and refer to the original paper

```
Noam D. Elkies :
"The still-Life density problem and its generalizations"
ftp://ftp.math.harvard.edu/elkies/MISC/still.ps.Z
```

for full details. We expect you to work through at least one of the remaining cases.

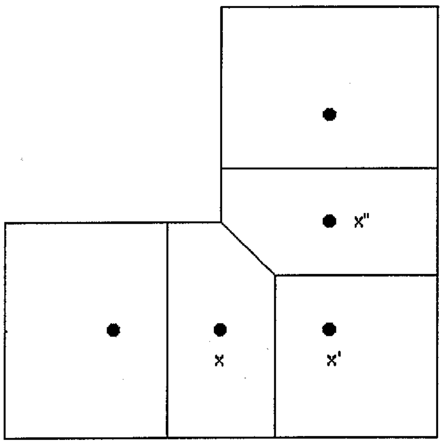
Let x be a vertex of type $\underline{A_3}cute$, then x has a unique pair $\{x', x"\}$ of neighbors in S adjacent to each other. Let x' be the vertex lying on the big line. Assume that x' has two neighbors, then it must be of type $\underline{r}ectangle$. Then, the vertex $x"$ must be one of the following types

$$\underline{a}cute \quad \underline{A_2}cute \quad \underline{A_3}cute \quad \text{or} \quad \underline{A_4}cute.$$

If we then make a patch P consisting of the Voronoi cells of $\{x, x', x"\}$ we obtain that $\alpha(P)$ is at least equal to

$$\alpha^0(A_3) + \alpha^0(r) + \begin{cases} \alpha^0(a) \\ \alpha^0(A_2) \\ \alpha^0(A_3) \\ \alpha^0(A_4) \end{cases} = -1 + 2 + \begin{cases} 7 \\ 3 \\ -1 \\ 3 \end{cases} \geq 0$$

with the worst possible case corresponding to the situation



When x' has three neighbors a lot more cases must be considered and bigger patches used.

Another use of these considerations is to classify the StillLife configurations having exactly density $\frac{1}{2}$. Then we must group the patches P everywhere such that $\alpha(P) = \alpha^0(P) = 0$ which allows us to exclude several types immediately such as

$$d \quad q \quad a \quad o \quad \text{and} \quad A_4$$

and that a vertex of type D must have one or two neighbors of type R. The case of two R neighbors leads to the density $\frac{1}{2}$ 'onions'

```
...o....o....o....o...
ooo.oooo.oooo.oooo.ooo
..o.o..o.o..o.o..o.o..
..o.o..o.o..o.o..o.o..
ooo.oooo.oooo.oooo.ooo
...o....o....o....o...
ooo..ooo..ooo..ooo..oo
o..ooo..ooo..ooo..ooo.
..o....o....o....o....
oo.oooo.oooo.oooo.oooo
.o.o..o.o..o.o..o.o..o
.o.o..o.o..o.o..o.o..o
oo.oooo.oooo.oooo.oooo
..o....o....o....o....
oo..ooo..ooo..ooo..ooo
..ooo..ooo..ooo..ooo..
o....o....o....o....o.
.oooo.oooo.oooo.oooo.o
.o..o.o..o.o..o.o..o.o
```

and a probably complete list of density $\frac{1}{2}$ StillLifes is compiled with this local patch method. For more details we again refer to Elkies' paper.

Week 10

CombinatorialGames.

Voronoi regions also play an important part in some games where there is a struggle for space, the most classical example being *go*. This week we introduce another such game where the methods described before can be used. We will also give a very quick introduction to the theory of CombinatorialGames.

10a ColeringGame.

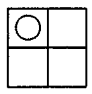
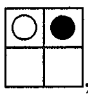
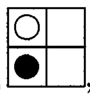
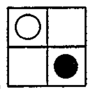
Colering

There are two players : Left (playing bLack) and Right (playing white). They take turns in filling one square of some rectangular board with a go-stone subject to the rule that rules of the game two neighboring squares (in the King-move topology of GameOfLife) cannot be colored with the same color. The first player unable to move is declared the looser.

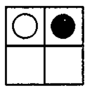
As each move simplifies the game, we can compute the value of a position G recursively. A position will be denoted by

$$G = \langle G^{L_1}, ..., G^{L_k} \mid G^{R_1}, \ldots, G^{R_l} \rangle$$

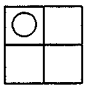
where G^{L_i} (resp. G^{R_j}) are the optional position for Left (resp. Right). For example

 $= \langle$ , ,  $\mid \emptyset \rangle$

Now any of the three Left options is a very simple position, as for instance

 $= \langle \emptyset \mid \emptyset \rangle \overset{def}{=} 0$

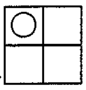
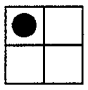
(and the others have the same value). In computing the value of a position we certainly can skip repeating positions of equal value, so

 $= \langle\, 0 \mid \emptyset \,\rangle \overset{def}{=} 1.$

Every position has a negative which is defined recursively by

$$-G = \langle -G^{R_1}, \dots, -G^{R_l} \mid -G^{L_1}, \dots, -G^{L_k} \rangle$$

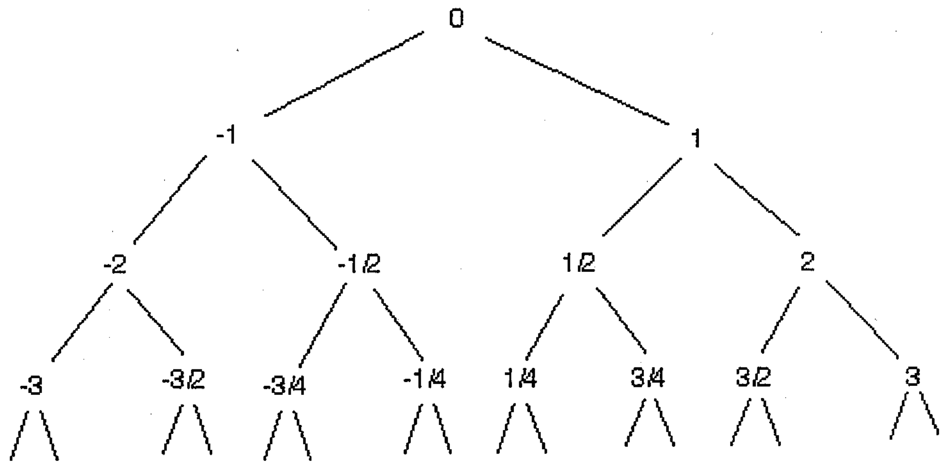
where of course $-\emptyset = \emptyset$ and $-0 = 0$. In `Colering` the negative of a position is the position obtained by changing the colors. Therefore,

$-$  $=$  $= \langle \emptyset \mid 0 \rangle \overset{def}{=} -1.$

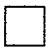
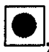
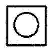
In general, a position $G = \langle G^{L_i} \mid G^{R_j} \rangle$ is a *number* provided

- All G^{L_i} and G^{R_j} are numbers or $\emptyset$,
- No G^{L_i} is larger or equal than some G^{R_j} and
- No G^{R_j} is smaller or equal than some G^{L_i}.

The *value* of a number-position is then the simplest value in the binary tree



strictly greater than any of the G^{L_i} and strictly smaller than any of the G^{R_j}. Many positions in `Colering are such dyadic numbers` but not all. For example,

 $= \langle$ ,  $\rangle = \langle 0 \mid 0 \rangle \overset{def}{=} *$

which is an example of a *fuzzy position*, that is a position G such that neither $0 \le G$ nor $G \le 0$. The relevance of the value of a position G is the following

- If $G = 0$ then the 2nd player wins.
- If $G > 0$ then L wins no matter who starts.
- If $G < 0$ then R wins no matter who starts.
- If G is fuzzy then the 1st player wins.

If you want a challenge : try to prove (by recursion) the following

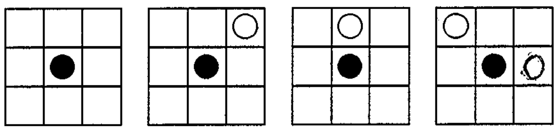
Theorem 10.1. (Conway,Guy) *Any position G in* `Colering` *has the value*

$$x \quad \text{or} \quad x* = \langle x \mid x \rangle$$

for x some dyadic integer.

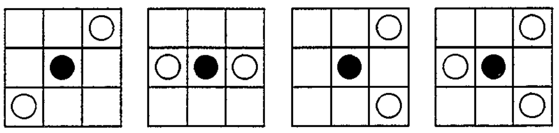
10b ColeringDensity.

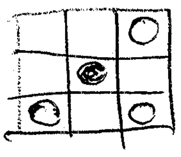
Let us apply what we learned to prove the maximal density of a legal `Colering` position on an infinite board and to classify those positions that attain this maximal density. The strategy will be similar. First for each *go*-stone we classify the possible neighborhood-stones and compute the α^0-value which is $8.(Area\ V^0(s) - 2)$ where $V^0(s)$ is the small Voronoi region of the stone s. One encounters only the following situations with ≤ 3 neighbors



m

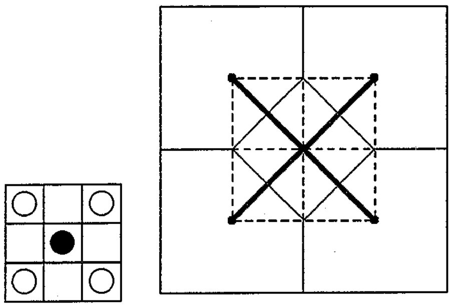
mnemo	*zero*	*halfdia*	*side*	*obtuse*
$\alpha^0(s)$	16	12	8	4





mnemo	*diag*	*line*	*quart*	*Quart*
$\alpha^0(s)$	8	0	8	0

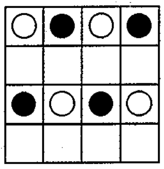
the only new neighborhood possibility has 4 neighbors, we remember it as *Cross* and its small Voronoi diagram has the following shape



Therefore, the area of $V^0(s) = 2$ and consequently $\alpha^0 = 0$. Repeating the argument for the StillLife configurations these computations prove

Theorem 10.2. *Every legal* `Coloring` *position has density* $\leq \frac{1}{2}$.

Moreover this density can be reached by an infinite tiling of the following pattern



We can also classify the positions of density $\frac{1}{2}$ as in them every stone must have a neighborhood of type *line*, *Quart* or *Cross*. Apart from the infinite pattern given above (where each line can be colored in 2 different ways) we obtain a variety of *Chicken wire* positions. The stones are placed in the following positions

```
o.o.o.o.
.o.o.o.o
.o.o.o.o
.o.o.o.o
o.o.o.o.
.o.o.o.o
.o.o.o.o
o.o.o.o.
.o.o.o.o
o.o.o.o.
```

where in each section the size of the hole is arbitrary (in this case they are of size ...,3,2,1,... . Observe that we also encounter these configurations in StillLife but then the sizes of all holes must be > 1. Also observe that coloring one stone dictates the colors of the other stones. Further, each of these positions has value 0 as no player has a remaining move. In short,

Theorem 10.3. *All legal* `Coloring` *positions of density* $\frac{1}{2}$ *are ChickenWire configurations with the size of the holes* $1, 2, \ldots, \infty$. *each of these configurations gives rise to exactly two legal positions (except when some holes are* ∞*). All maximal density positions have value* 0.