Top Ten Algorithms Class 12

John Burkardt
Department of Scientific Computing
Florida State University

.....

http://people.sc.fsu.edu/~jburkardt/classes/tta_2015/class12.pdf

16 November 2015

Our Current Algorithm List

1. Back Propagation algorithm
2. Bank routing number checksum for error detection
3. Bernoulli number calculation
4. Bootstrap algorithm
5. Data stream: most common item
6. Discrete Cosine Transform
7. Discrete Fourier Transform
8. Euclid's greatest common factor algorithm
9. Gram-Schmidt vector orthogonalization algorithm
10. Hamming error correcting codes

Our Current Algorithm List

1. ISBN (International Standard Book Number) checksum
2. k-means clustering algorithm
3. Luhn/IBM checksum for error detection
4. Monte Carlo Sampling
5. PageRank algorithm for ranking web pages
6. Pancake flipping algorithm for genome relations
7. Path counting with the adjacency matrix
8. Power method for eigenvector problems
9. Probability evolution with the transition matrix
10. Prototein model of protein folding

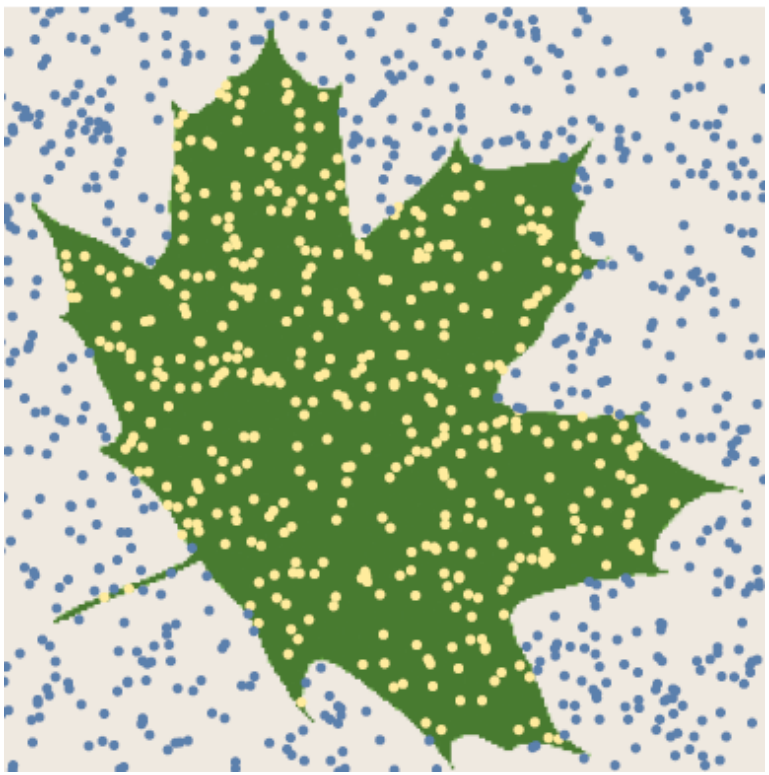
Our Current Algorithm List

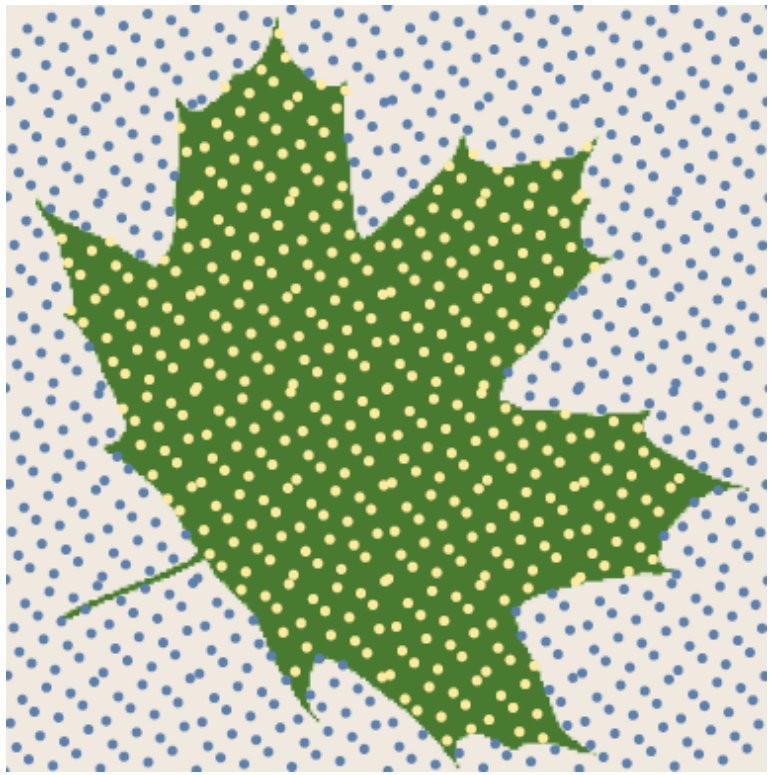
1. QR (Quick Response) images and error correction
2. QR matrix factorization
3. QR iteration for eigenvalues
4. Reed-Solomon error correcting codes
5. Ripple Carry algorithm
6. Search engine indexing
7. Signal detection in noisy data
8. Trees for computational biology
9. UPC (Universal Product Code) checksum for error detection
10. Zero Knowledge Proofs

Thomas Britton, *"Quasi-Random Numbers"*

Quasi-random number algorithms generate sequences of points that do a much better job of evenly sampling a line, a square, a cube or an arbitrary region.

Reference: Brian Hayes, "Quasirandom Ramblings", merican Scientist, July/August 2011.





Computational Geometry

Computational geometry identifies cases in which a computation must include information about the geometric arrangement of objects. Often, these objects are points, lines or curves, squares, circles or polygons, cubes, spheres or other solid shapes, which are to be studied for themselves, or because they are a natural way to represent part of some larger problem.

Geometric properties of interest include length, area and volume, distance, perpendicularity, inclusion, orientation, angles, convexity.

We may want to plan the motion of one object through a field of other objects.

Computer-aided design involves the construction and analysis of complex 3D shapes.

One of the advantages of the finite element method is that it can represent complex geometries and solve advanced partial differential equations on them.

Computational Geometry

- Is the ball in the box?
- Is the hat to the left of the line?
- Can I compute or estimate the area of a shape?
- Can I compute or estimate the centroid of a shape?
- Can I pick random points from a circle, triangle, the surface of a sphere?
- Define states by pointing to the nearest state capital.
- What is the best network of roads to connect 10 cities?
- If we have to have 50 states, what would be a "better" arrangement?
- If a rabbit randomly chose 1 of 50 boxes to hide in, how many boxes do I have to look in to find it? (obviously, I'm not telling you everything about this problem yet!)

Algorithm 1 for the Area of a Triangle

Points A, B and C determine a triangle. How can we compute its area?

Since the area of a triangle is one half the base times the height, we can find the length of the line segment [A,B], the **base**, then figure out the distance from C to the line [A,B], the **height**.

Algorithm 2 for the Area of a Triangle

Another way to compute the area of a triangle uses ideas from vector algebra.

The cross-product of two vectors is the area of the parallelogram between them.

The triangle can be regarded as two vectors (B-A) and (C-A).

The area is 1/2 the determinant of this 2x2 matrix:

$$\begin{aligned}\text{area}(A,B,C) &= 1/2 \begin{vmatrix} B_x - A_x & B_y - A_y \\ C_x - A_x & C_y - A_y \end{vmatrix} \\ &= 1/2((B_x - A_x)(C_y - A_y) - (B_y - A_y)(C_x - A_x))\end{aligned}$$

Code for Triangle Area

```
1  function area = triangle_area ( ax, ay, bx, by, cx, cy )
2
3    area = 0.5 * ...
4      ( ( bx - ax ) * ( cy - ay ) ...
5      - ( cx - ax ) * ( by - ay ) );
6
7    return
8  end
```

Listing 1: Area of triangle

Algorithm for the Signed Area of a Triangle

With the second approach, notice that it is possible to get a **negative area**. Indeed, if A = (0,0), B = (1,4), C = (3,2), we get area (A,B,C) = -5. Interestingly enough, area (A,C,B) = +5.

The minus sign is telling us something very useful: the triangle (A,B,C) has its vertices listed in clockwise order, but (A,C,B) lists them in counterclockwise order, The sign of the area is a warning about the orientation of the triangle.

As long as we promise to list triangle vertices in counterclockwise order, we will have no problems with the area formula. But it turns out that this bit of knowledge can be used to determine other information.

Algorithm for Testing Collinearity

Start with two points A and B, which define a line.

Now consider test points C, D, and E.

We'd like to know if one of these points lies on the line [A,B].

What do we know if:

- area(A,B,C) = -2?
- area(A,B,D) = 3?
- area(A,B,E) = 0?

Does a Triangle Contain a Point?

Given triangle A, B, C and a point P, it is natural to ask whether P is contained inside the triangle.

If P is contained inside the triangle, then the following three triangles will all have positive area:

- (P,B,C)
- (A,P,C)
- (A,B,P)

If any of the these values is negative, then P lies outside the triangle!

Algorithm for Barycentric Coordinates

For triangle A, B, C, define barycentric coordinates of point D:

(α, β, γ) = (area(D,B,C),area(A,D,C),area(A,B,C))/area(A,B,C).

The barycentric coordinates have some useful properties:

- $\alpha + \beta + \gamma = 1$.
- $(x, y) = \alpha * A + \beta * B + \gamma * C$.
- (1/3,1/3,1/3) is the centroid of the triangle.
- D strictly inside (A,B,C) if all 3 coordinates greater than 0.
- D on perimeter if one coordinate is 0, others positive.
- D is a vertex if two coordinates are zero.
- D is outside (A,B,C) if any coordinate is negative.

Code for Barycentric Coordinates

```
1  function [ alpha, beta, gamma ] = bary ( ax, ay, bx, by, cx,
       cy, px, py )
2
3    abc = triangle_area ( ax, ay, bx, by, cx, cy );
4    pbc = triangle_area ( px, py, bx, by, cx, cy );
5    apc = triangle_area ( ax, ay, px, py, cx, cy );
6    abp = triangle_area ( ax, ay, bx, by, px, py );
7
8    alpha = pbc / abc;
9    beta  = apc / abc;
10   gamma = abp / abc;
11
12   return
13 end
```

Listing 2: Barycentric Coordinates of XY

Algorithm for Barycentric Mapping

Bartycentric coordinates provide a universal coordinate system for relating any two triangles.

Given triangles (A,B,C) and (D,E,F), then points with the same barycentric coordinates will correspond.

To find the point Q in (D,E,F) that corresponds to P in (A,B,C):

- $P \rightarrow (\alpha, \beta, \gamma)$ in (A,B,C).
- $(\alpha, \beta, \gamma) \rightarrow Q$ in (D,E,F)

Algorithm for Barycentric Mapping

```
1   function [ qx, qy ] = triangle_to_triangle ( ax, ay, bx,
        ...
2     by, cx, cy, px, py, dx, dy, ex, ey, fx, fy )
3
4     [ alpha, beta, gamma ] = bary ( ax, ay, bx, by, cx, cy,
          px, py );
5
6     qx = alpha * dx + beta * ex + gamma * fx;
7     qy = alpha * dy + beta * ey + gamma * fy;
8
9     return
10  end
```

Listing 3: Triangle to Triangle Mapping

Quadrature Rules for Triangles

A quadrature rule for a triangle is a set of nodes and weights, which allow us to approximate an integral over the triangle.

$$\int_T f(x,y)\,dx\,dy \approx area(T) * \sum_{i=1}^{n} w_i f(x_i, y_i)$$

Except for some special simple cases (1 point centroid rule, the 3 point vertex rule), quadrature rules for the triangle are hard to work out, and depend on the shape of the triangle. So people have computed them for the reference triangle:
$T_{ref} = [(0,0),(1,0),(0,1)]$.

In order to use the rule to approximate an integral over a general triangle T, we have to map the reference rule to our triangle.

Algorithm for Transforming a Quadrature Rule

The triangle-to-triangle rule shows how to map the quadrature points.

Quadrature rule defined on triangle (A,B,C) = ([0,0], [1,0], [0,1]):

```
x = [ 4/6, 1/6, 1/6 ];
y = [ 1/6, 4/6, 1/6 ];
w = [ 1/3, 1/3, 1/3 ];
```

Transform rule to (D,E,F) = ([5,5], [8,2], [9,4]):

```
x' = [ 6.16, 7.66, 8.16 ];
y' = [ 4.33, 2.83, 3.83 ];
w' = [ 1/3, 1/3, 1/3 ];
```

Algorithm for Finite Element Basis Functions

The barycentric coordinates (α, β, γ) may also be regarded as the values of the three linear basis functions over the triangle, used by the finite element method.

This is because, if we regard them as linear functions of position P within the triangle (A,B,C), we have:

$$\begin{array}{lll} \alpha(A) = 1 & \alpha(B) = 0 & \alpha(C) = 0 \\ \beta(A) = 0 & \beta(B) = 1 & \beta(C) = 0 \\ \gamma(A) = 0 & \gamma(B) = 0 & \gamma(C) = 1 \end{array}$$

and thus any linear function $f(x, y)$ defined over triangle T can be regarded as a linear combination of these basis functions. The coefficients are simply the function value at the vertices:

$$f(P) = f(A) * \alpha(P) + f(B) * \beta(P) + f(C) * \gamma(P)$$

Triangles to Polygons

We have spent a long time concentrating on triangles, and yet the really interesting geometric shapes are complicated.

The thing to see is that even a map of the United States can be regarded as a very complicated polygon.

But if there is a way to turn a polygon into a collection of triangles, then all our work so far will apply.

The area of a polygon will be the sum of the areas of the triangles that make it up; polygon contains a point if one of its triangles contains the point, and so on.

So our question is, *given a polygon, is there a computational procedure for decomposing it into a collection of triangles?*

Polygon Triangulation Algorithm

A simple polygon is one whose perimeter consists of a single curve which does not cross itself.

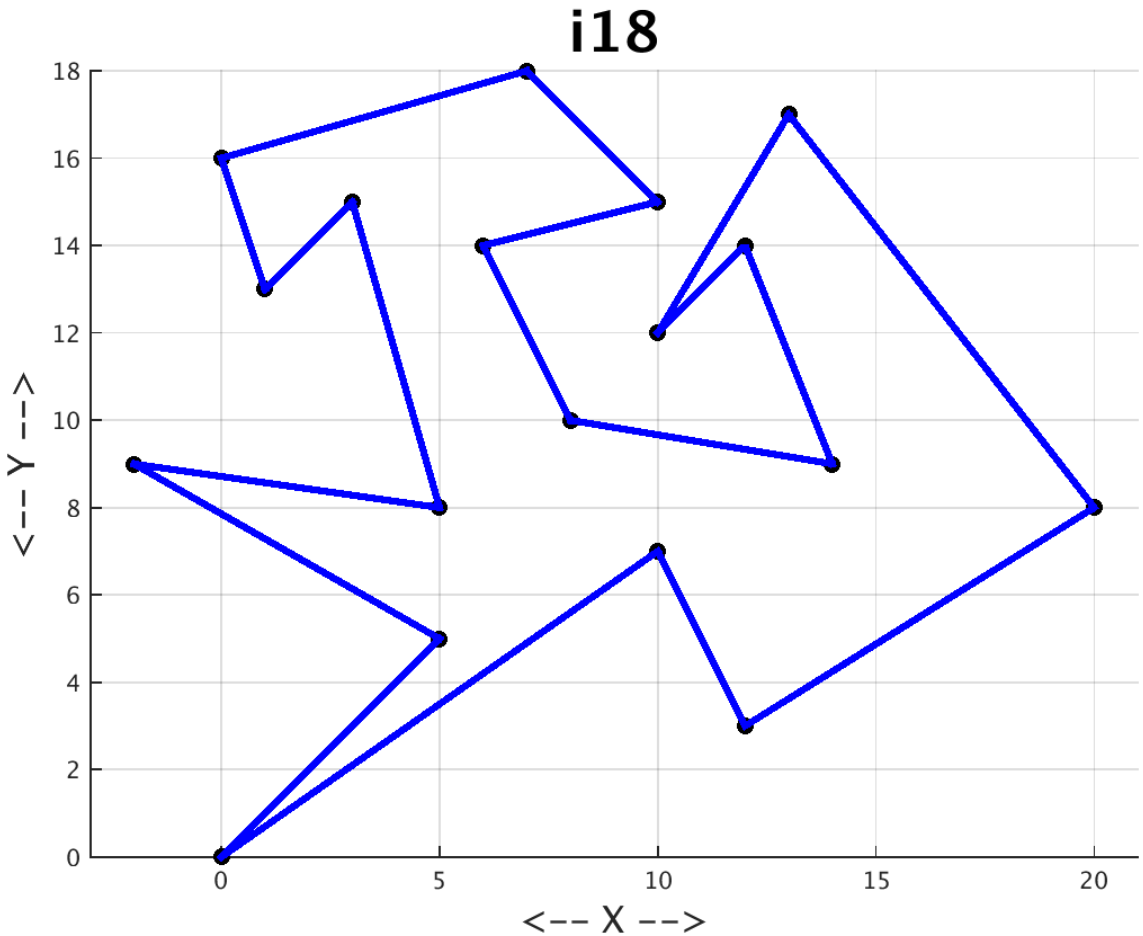
An "ear" of a polygon is a sequence of three vertices such that the diagonal connecting the first and last vertices is contained in the polygon.

If a simple polygon has at least 4 vertices then it has at least one ear.

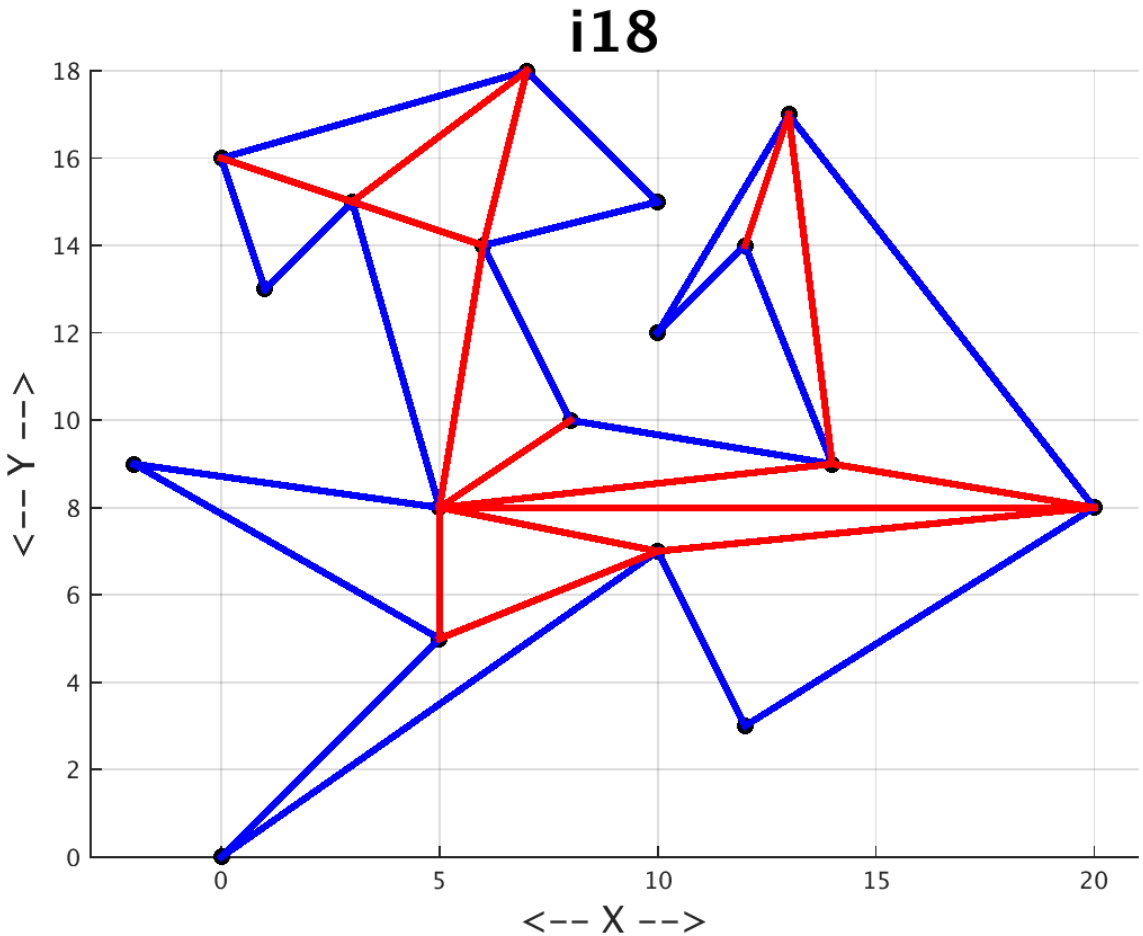
We can dissect a polygon into triangles by slicing off one ear at a time.

Our one computational concern is to ensure that the diagonal is inside!

A Polygon To Triangulate



A Triangulation



Unfinished Business

Computational geometry is a huge area of study by itself.

I have only been able to suggest how a study of triangles gives you a beginning insight into this field.

If you are interested, some accessible topics include the convex hull, the Delaunay triangulation, and the Voronoi diagram, all of which are useful geometric tools that are easy to understand, although trick to implement.