Computational Geometry

Aim:
(i) the study of algorithms for solving geometric problems on a computer.
(ii) to construct basic programs for computer graphics applications.

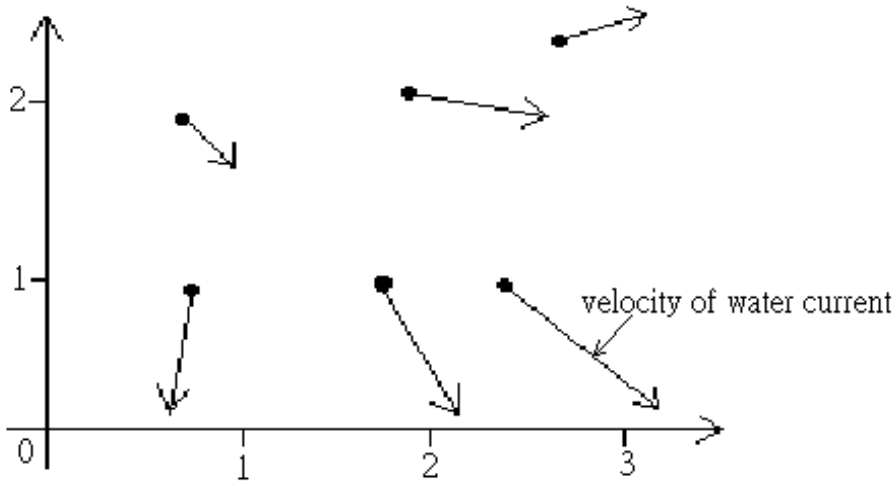
Topics

1. Polygon Triangulation
2. Convex Hulls in 2D
3. Voronoi Diagrams

Text Book

Computational Geometry in C
by J. O' Rourke
Cambridge University Press, 1994

Triangulation



Find the velocity of water current at (2.5, 2).

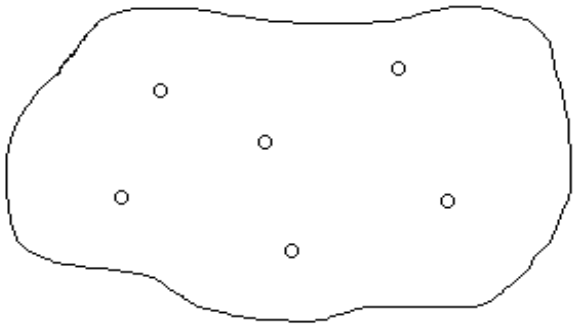
Convex Hulls

If a polyhedron is not convex, then the smallest convex set which contains it is called the convex hull of the polyhedron.

Question: Find the convex hull of the following points:
(3, -2), (5, 1), (7, 4), (6, 5), (4, 2),
(3, 3), (3, 5), (2, 5), (0, 5), (0, 1).

Question on facility location (Voronoi Diagram)

Suppose you would like to locate a new grocery store in an area with several existing, competing grocery stores, how would you locate it so that it is as far away from the old ones as possible?

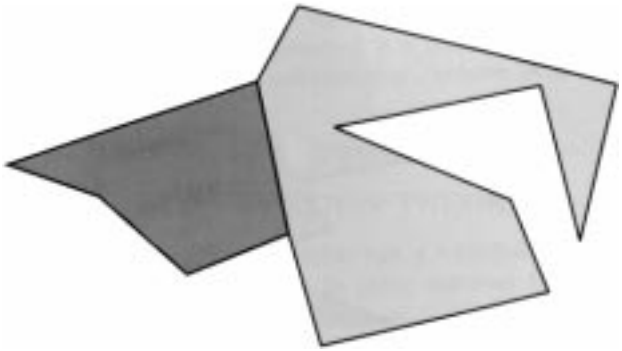


o Existing Grocery Store

Theorem 1 (Triangulation)

Every polygon P of n vertices may be partitioned into triangles by the addition of diagonals.

Proof by induction:
If n = 3, the polygon is a triangle.
Let $n \geq 4$. Add a diagonal to P and it partitions P into two polygons with vertices < n.
Applying the induction hypothesis to the two subpolygons completes the proof.



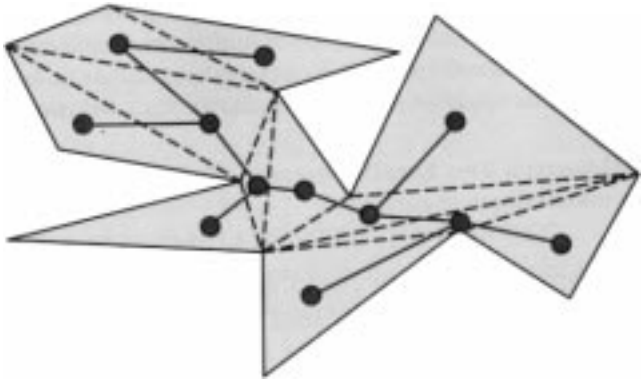
Properties of Triangulation

Let P be a polygon of n vertices.

1. No. of diagonals = n – 3.
2. No. of triangles = n – 2.
3. Sum of internal angles = (n – 2) π.

Dual of a triangulation:

A graph with a vertex associated with each triangle, and an edge between two vertices iff their triangles share a diagonal.



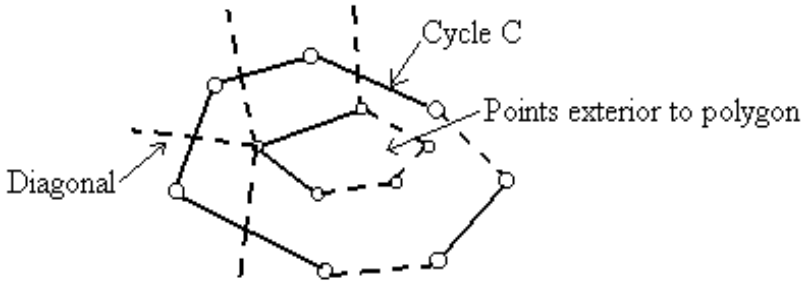
Lemma 2

The dual T of a triangulation is a tree, with each vertex of degree at most three.

Proof:

A triangle has at most 3 sides to share $\Rightarrow$ degree of a vertex ≤ 3.

Suppose T is not a tree; then it must have a cycle C. C must enclose some polygon vertices and thus some points exterior to the polygon. This contradicts the simplicity of the polygon.



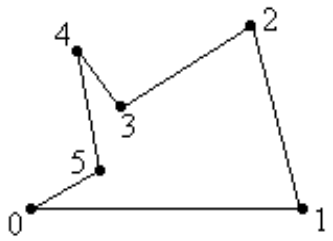
Ear of a polygon:

Three consecutive vertices of a polygon a, b, c form an ear of the polygon if ac is a diagonal; b is the ear tip.

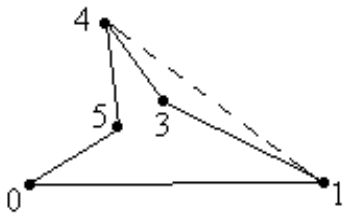
Simple Triangulation Algorithm

Input consecutive points in counter- clockwise direction.
n = number of vertices.
If n > 3 then
    for each potential ear diagonal (i, i+2) do
        if Diagonal (i , i+2) then
            Print diagonal .
            Remove ear at v_{i+1}.
            Recurse on remainder of polygon.

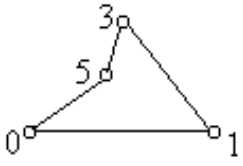
Example Triangulate the following polygon P using the simple algorithm.



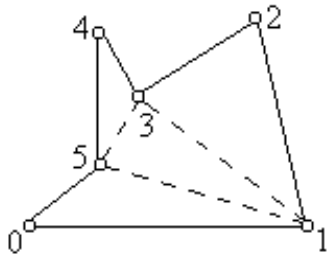
Solution: Consider vertices 0, 1, 2.
Since 02 intersects with 45, 02 is not a diagonal.
We then consider vertices 1, 2, 3.
13 is a diagonal. Δ123 is removed & the following new polygon is considered.



We now consider vertices 0, 1, 3.
03 is not a diagonal since it intersects with 45.
Next, vertices 1, 3, 4 are considered.
14 is an external diagonal which we do not want.
Then, vertices 3, 4, 5 are considered.
35 is a diagonal & Δ345 is removed. The reduced polygon is shown below:



Finally, 15 is another diagonal and the triangulation is as follows:



In the above example, for a potential diagonal, we have to determine whether
(i) it intersects with other edges of the (reduced) polygon; and
(ii) it is internal or external.

Program Implementation

(I) Segment intersection

Given two line segments. A common inclination is to calculate the point of intersection between the lines containing these segments and check whether the point falls inside the segments. However, this code is messy and may lead to serious error.

We will use a simple idea based on the sign of area of a triangle.

Proposition 3 Given 3 vertices $v_1 = (x_1 , y_1)$, $v_2 = (x_2 , y_2)$ and $v_3 = (x_3 , y_3)$ labeled counter-clockwise. Twice the area of the triangle $T = (v_1 , v_2 , v_3)$ is given by

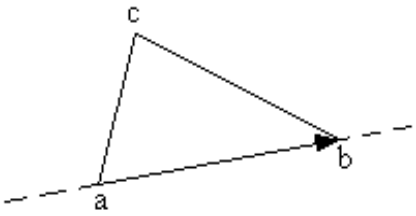
$$2A(T) = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

$$= x_1y_2 - x_2y_1 + x_2y_3 - x_3y_2 + x_3y_1 - x_1y_3$$

(See Code 1.3 Area 2)

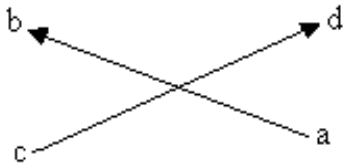
Corollary 4 For a directed line determined by two points given in a particular order (a, b) and another given point c,

(i) c is to the left of (a, b) iff the area of the counterclockwise triangle, A(a, b, c), is positive;
(See Code 1.4 Left)



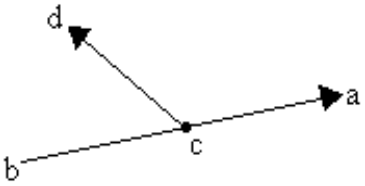
(ii) c is collinear with (a, b) iff *A*(a, b, c) = 0.
(See Code 1.5)

We now examine whether a potential diagonal intersects with any edge.
(Part (i) in P.21)
(i) Proper intersection – two segments intersect at a point interior to both.



Check : c and d on opposite sides of (a, b)
∧ a and b on opposite sides of (c, d).
This checking can be implemented by using the codes Left and Collinear.
(See Code 1.6 IntersectProp)

(ii) Improper intersection – an end-point of one segment lies somewhere on the other segment.

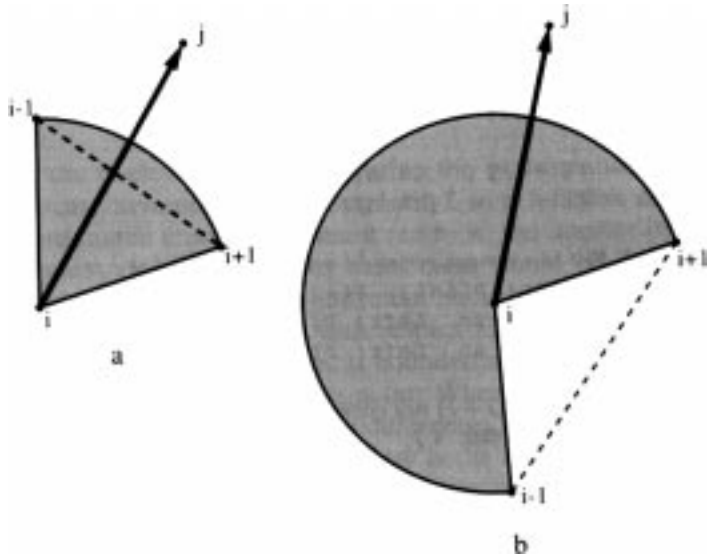


Check that c is between (a, b):
- a, b, c are collinear , and
- Coordinate of c is in between those of a and b.
(See Code 1.8 Between)

We have to check all possible combination of betweenness
- a, b, c; a, b, d; c, d, a; c, d, b.
(See Code 1.9 Intersect)

If an edge is a diagonal, we next check whether it is internal or not.

For a given diagonal $s = v_iv_j$, we have to find a way to check whether it is in the cone determined by v_{i-1}, v_i, v_{i+1} labeled counterclockwise.



(a) Convex
$s = v_iv_j$ is an internal diagonal
$\Leftrightarrow$ v_{i-1} is on the left of $s = v_iv_j$ and v_{i+1} is on the right.

(b) Reflex
$s = v_iv_j$ is internal
$\Leftrightarrow$ it is not external : it is not the case that both v_{i+1} is left or on v_iv_j, and v_{i-1} is left or on v_jv_i.
(See Code 1.11 InCone)

Note that this test simply builds on the codes Left and Collinear.

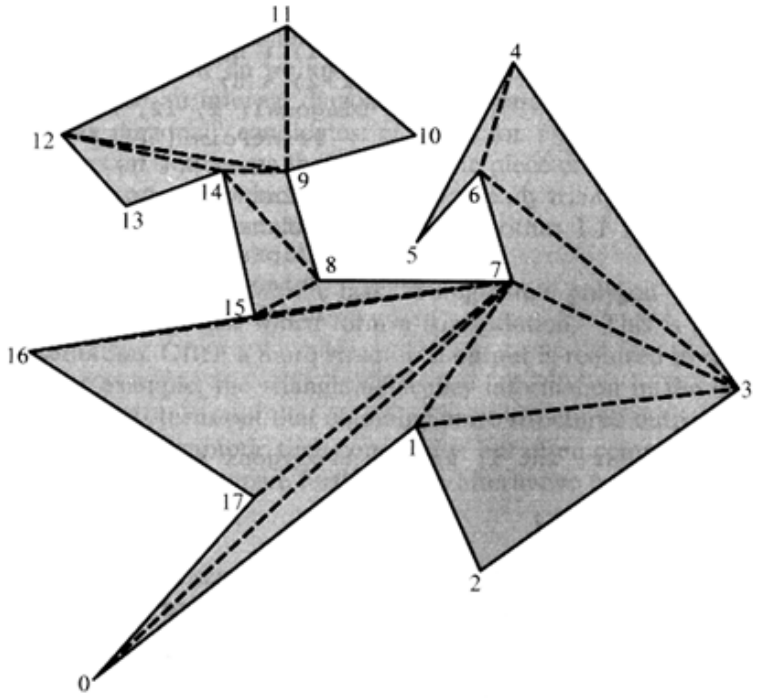
Data Structures

- The coordinates of the vertices of a polygon are inputed in the counter-clockwise order and assigned with a number.

- When an internal diagonal is found and the ear tip v_i is to be deleted the polygon array location [i+1, n-1] is copied into locations [i, n-2], thereby overwriting location i.
(See Code 1.17 ClipEar1)

- Output is a sequence of pairs of endpoint indices representing diagonals.

Polygon P



Input

i	x	y
0	0	0
1	10	7
2	12	3
3	20	8
4	13	17
5	10	12
6	12	14
7	13	11
8	7	11
9	6	14
10	10	15
11	6	18
12	–1	15
13	1	13
14	4	14
15	5	10
16	-2	9
17	5	5

n = 18 vertices

Triangulation Diagonals

1	3
4	6
3	6
3	7
1	7
0	7
9	11
9	12
12	14
9	14
8	14
8	15
7	15
7	16
7	17

Complexity

Triangulation	[T(n)]
if n > 3 then	
for each potential ear	
diagonal (i, i+2) do	[≤ n iteration]
if Diagonal (i, i+2) then	[O(n)]
Print diagonal	[O(1)]
Remove ear at v_i	[O(n)]
Recurse on remainder of polygon	[T(n-1)]
break	

$\therefore T(n) = O(n) \times O(n) + O(1) + O(n) + T(n-1)$
$\Rightarrow T(n) = O(n^2) + T(n-1)$
$\Rightarrow T(n) = O(n^3)$

History of Triangulation Algorithms

1978	O(n log n)
1988	O(n log log n)
1989	$O(n \log^* n)$
1991	O(n)

$\log^* n$ = no. of times the log must be iterated to reduce n to 1 or less

e.g. For $n = 2^{2^{16}} \approx 10^{19728}$,
$\log^* n = 5$.