CS60064 Spring 2022

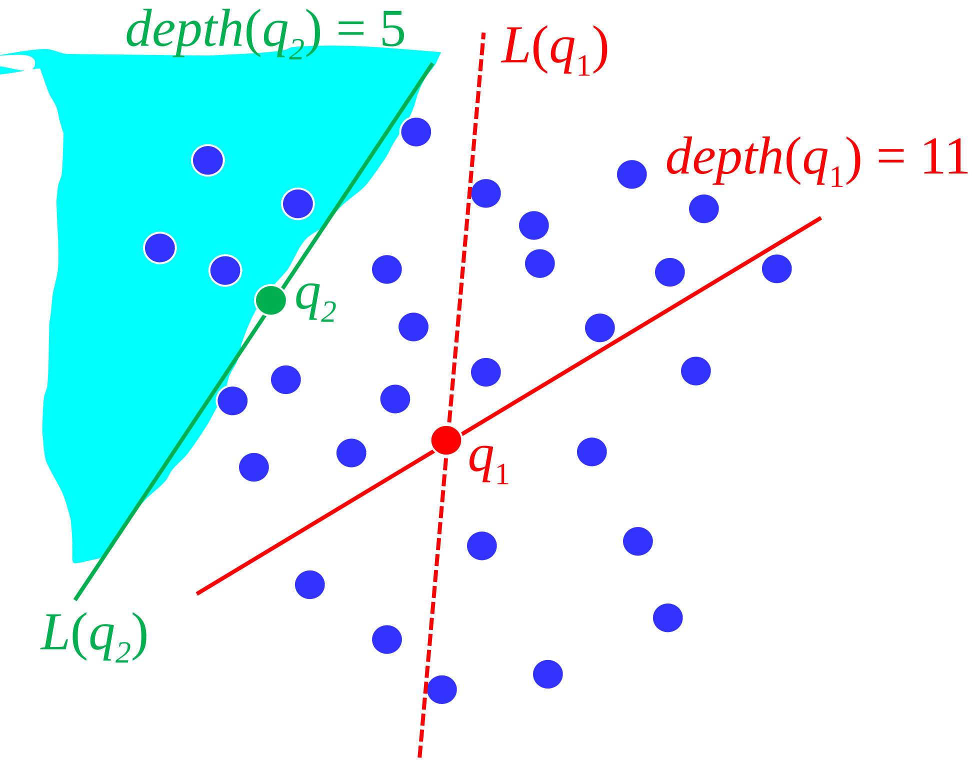
Computational Geometry

Instructors

Bhargab B. Bhattacharya (BBB)
Partha Bhowmick (PB)
Lecture 04
12 January 2022

Indian Institute of Technology Kharagpur
Computer Science and Engineering

Problem of the Day



Data depth – used in analytics, M/L

Given a cluster of data comprising *n* points, what is the relative location of a new query point?

depth of query point (*q*) in 2D:
- imagine a line *L* passing thru *q*;
- rotate *L* around *q* such that # points appearing on one side of *L* is *minimized* over all angles;
- Output the number including *q*;

i.e., the smallest number of points in any closed *half-plane* that contains *q* (Tukey depth)

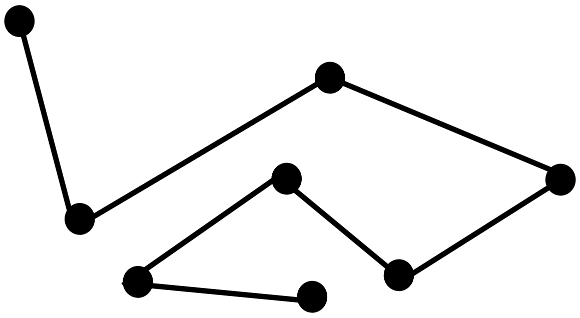
Problem: Design an efficient algorithm for computing the depth of a query point in a 2D cluster of *n* points

Other measures: use of convex hull; distance of *q* from the centroid (arithmetic mean) or from the Fermat point

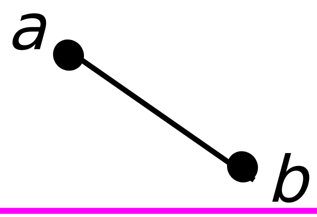
Introducing Polygons

Polygonal Curves

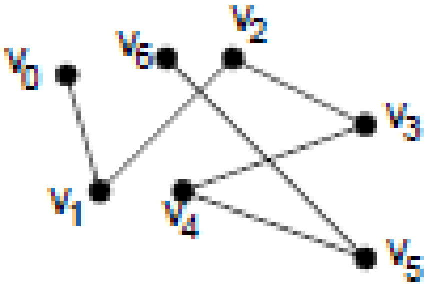
Ref: David Mount, Lecture Notes



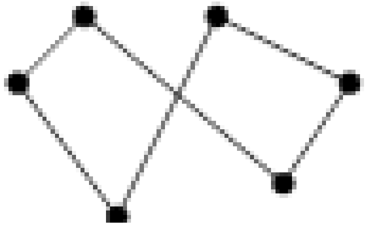
simple
polygonal curve
(open)



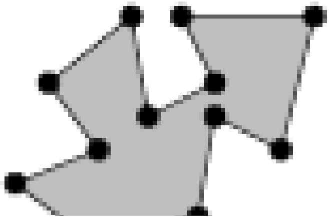
line segment: a subset of a straight-line contained between two end-points *a*, *b* (inclusive), denoted as *ab*



open polygonal curve but not simple



closed polygonal curve but not simple

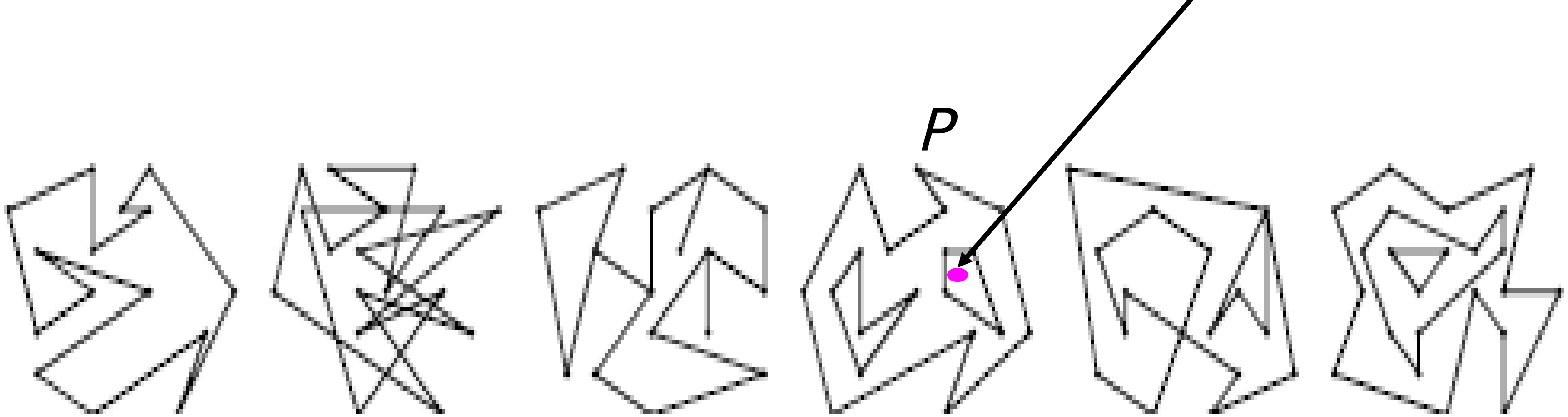


polygon: closed and simple polygonal curve

Simple Polygons

- *Definition:* A simple polygon *P* is the (closed) region bounded by a “simple closed polygonal curve”

q   *P*?



simple polygon

no

PSLG

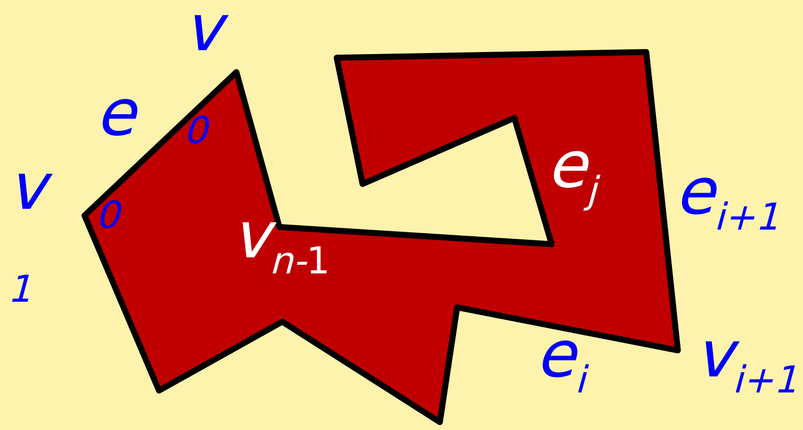
polygon with holes

no

nested polygons

Simple Polygon

Two non-consecutive edges are disjoint
Two consecutive edges have a single common end-point



Simple Polygons

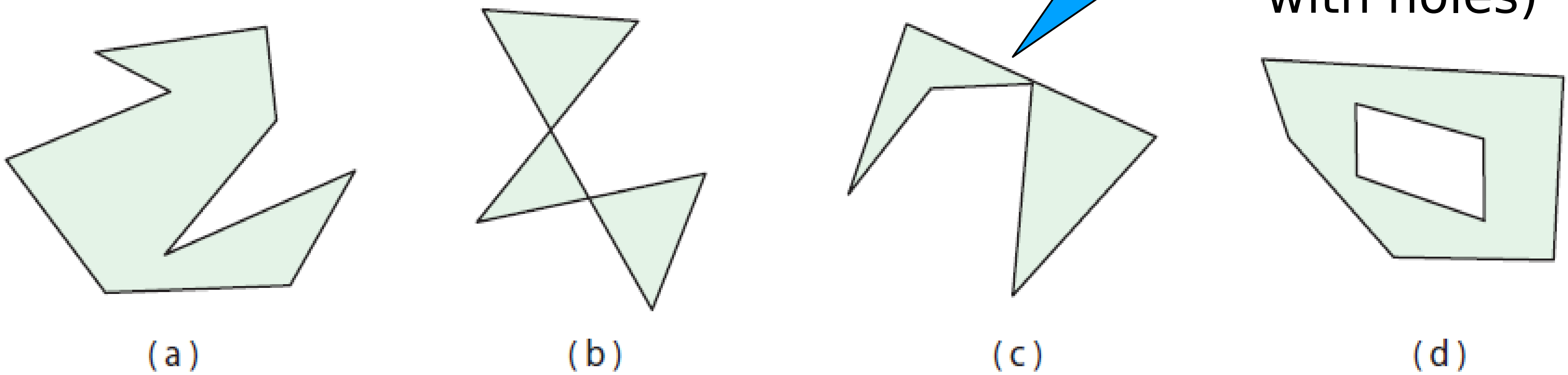
Some definitions would allow this as a “degenerate” simple polygon

yes

no

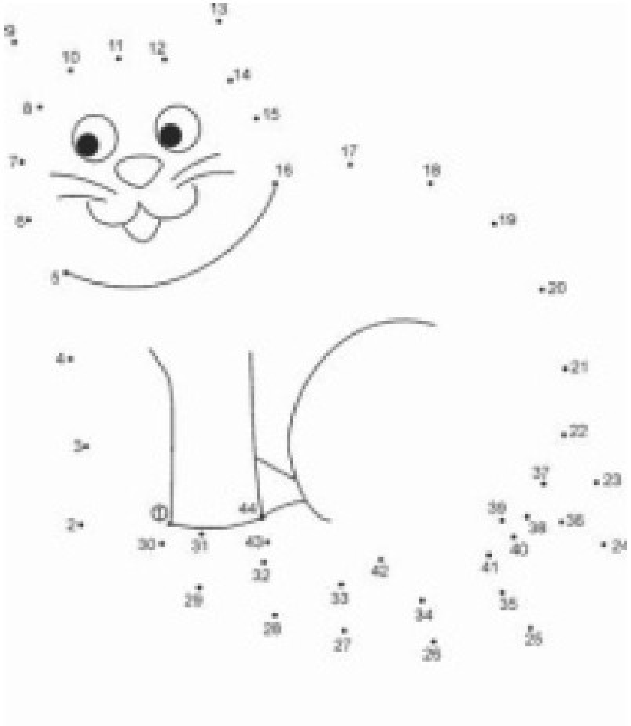
no

no (polygon with holes)

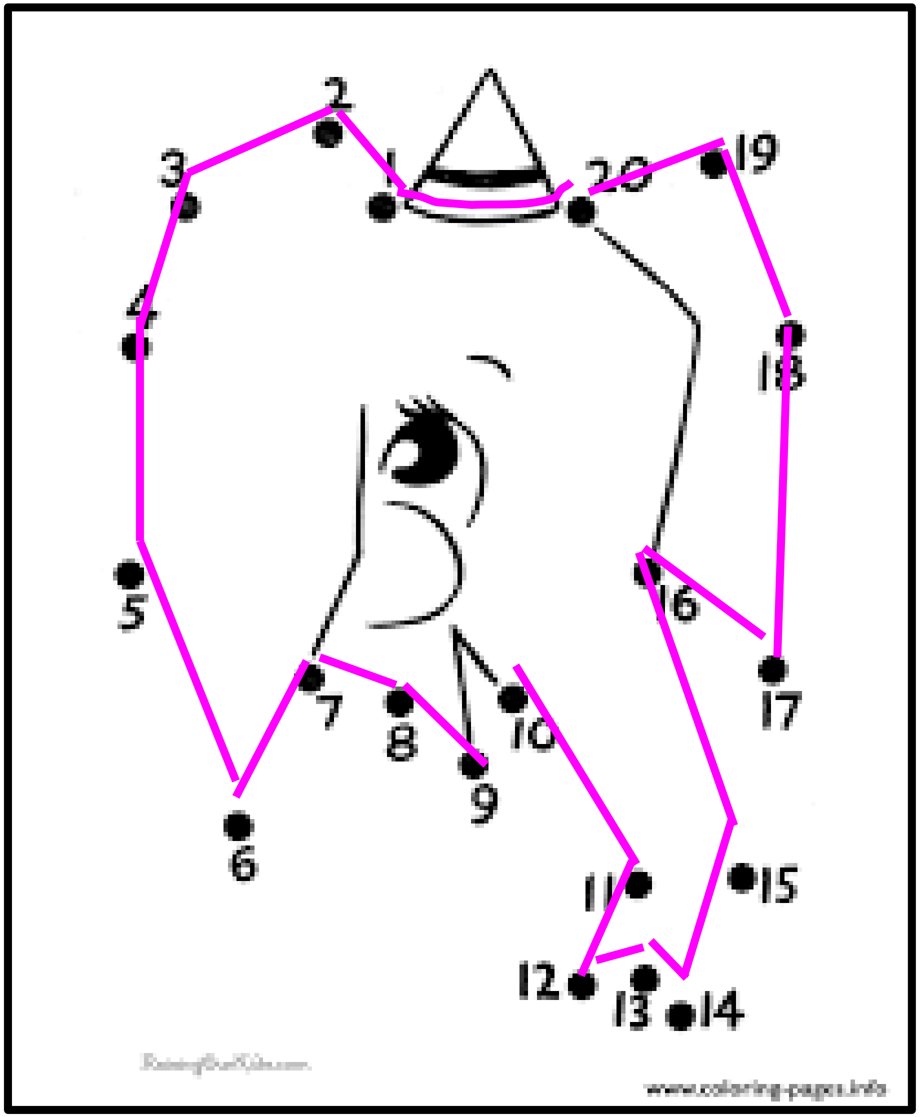


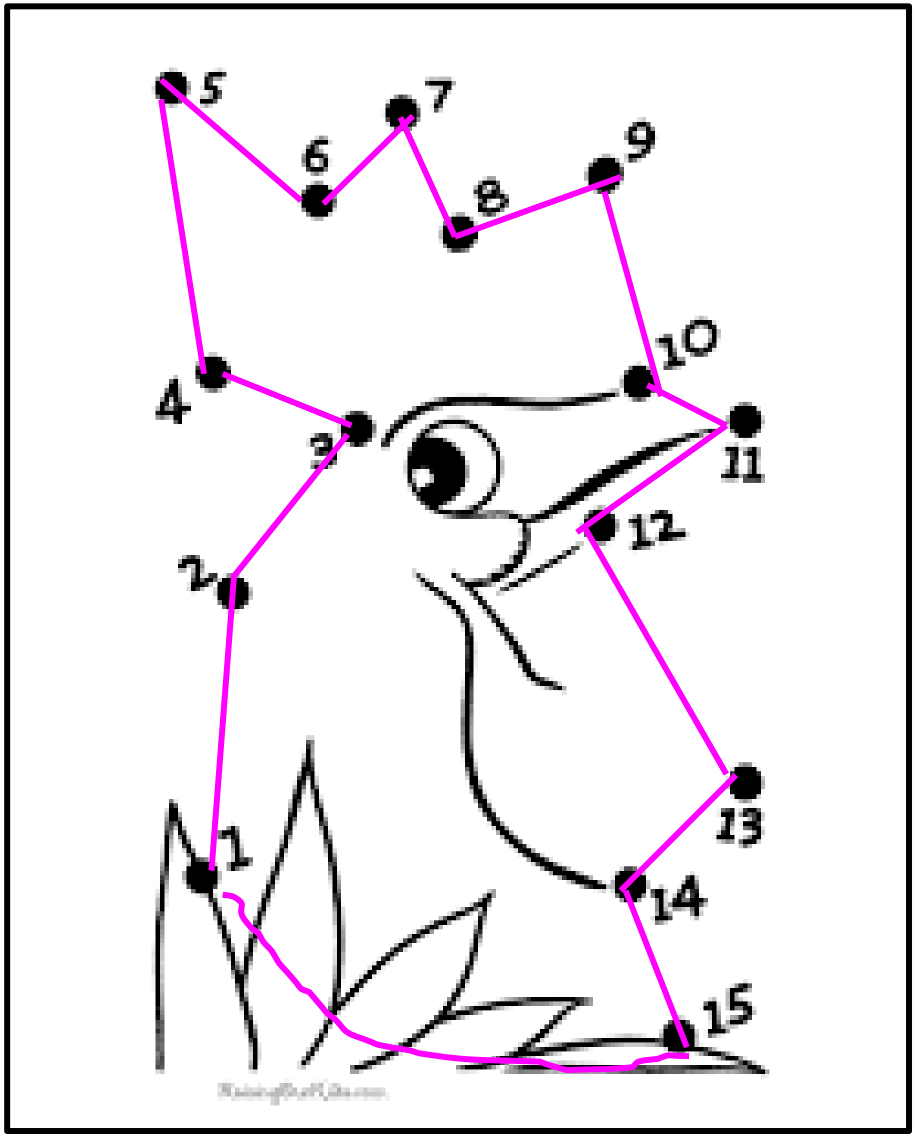
(Polygonal Jordan Curve). *The boundary ∂P of a polygon P partitions the plane into two parts. In particular, the two components of $\mathbb{R}^2 \setminus \partial P$ are the bounded interior and the unbounded exterior.*[2]

Connect-the-dots

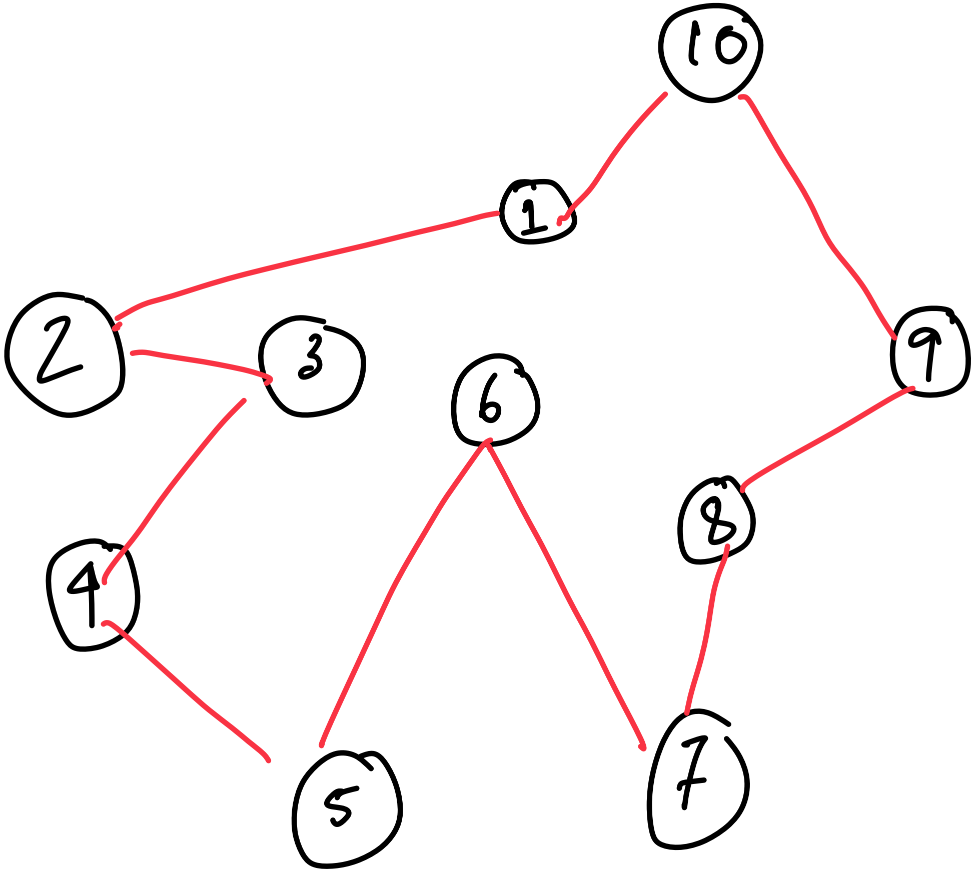


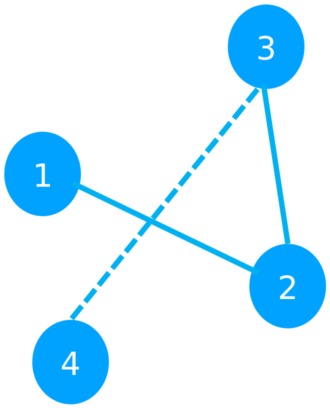
Labels of the vertices are given; draw the polygonal edges in sequence, to reveal ...





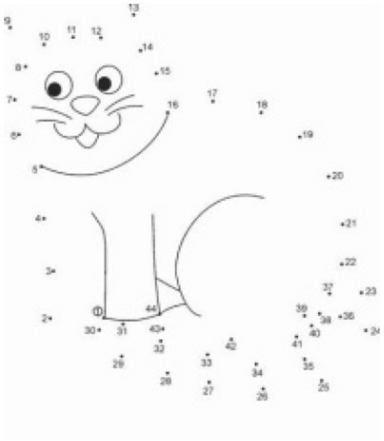
Polygon described as an ordered sequence of vertices

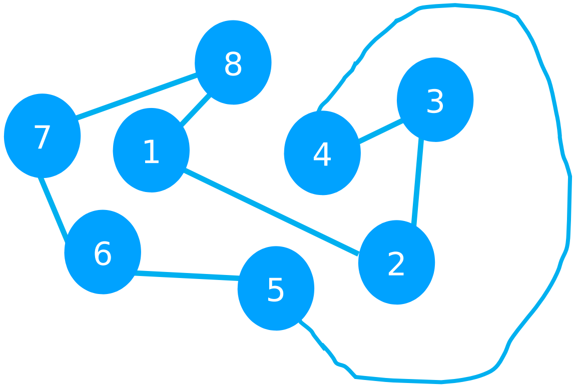




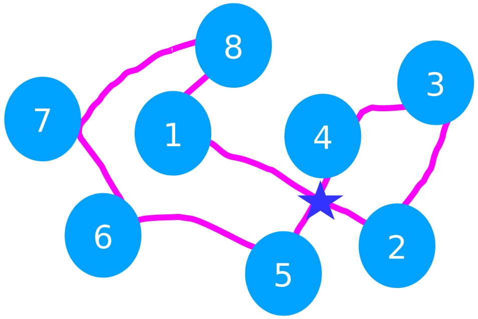
Given labelled points, is it always possible to construct the polygon?

Connect-the-dots





normal



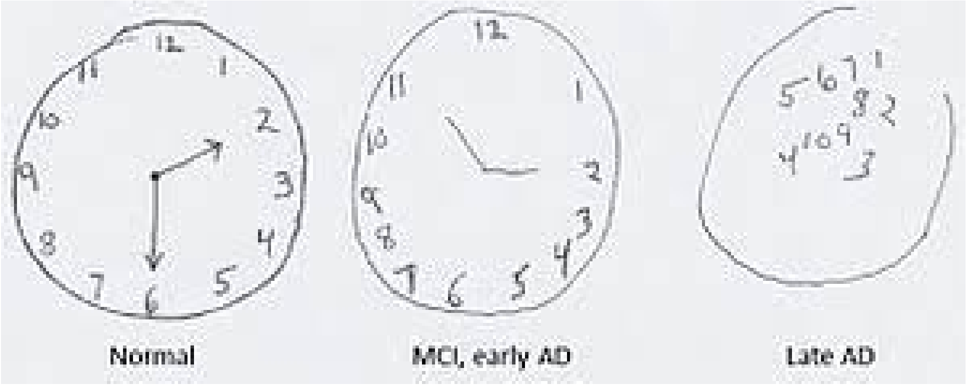
D, P,
AD

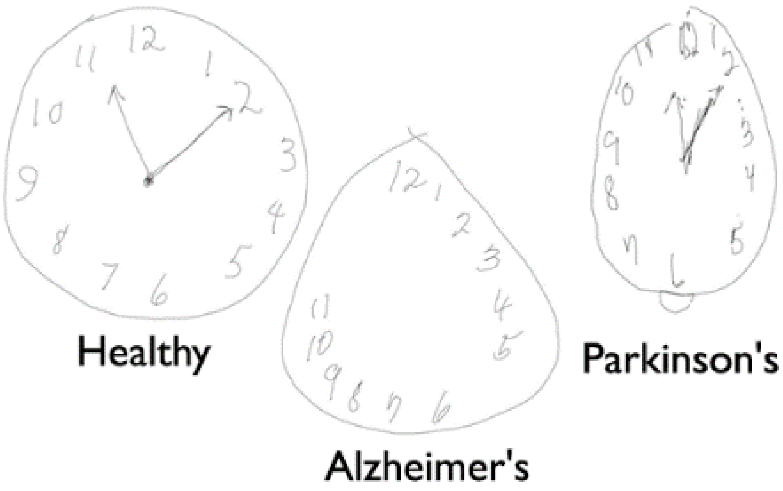
Trail Marking Test:

Often used in testing cognitive decline in dementia, Parkinson’s, Alzheimer's disease

Q. Given points 1, 2, …, *n* on 2D, can you sequentially connect them with a closed curve without crossing?

Geometric Cognition Test



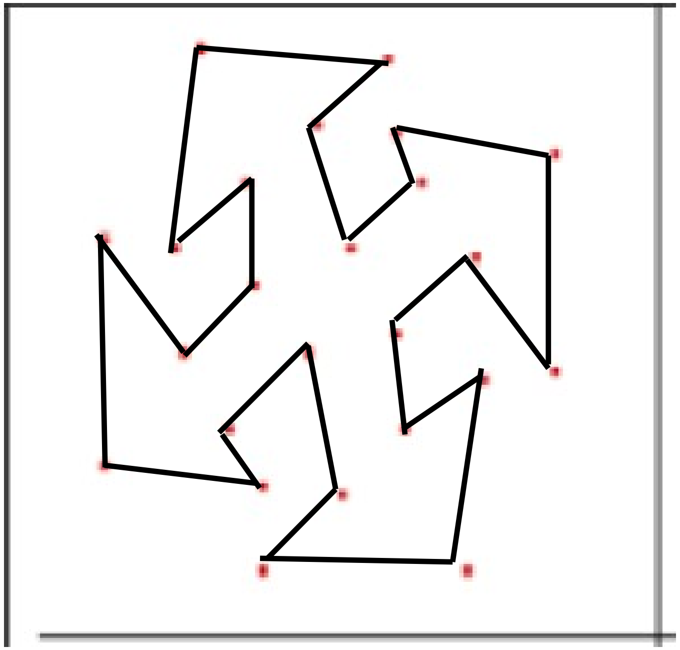


Clock Drawing Test: often used in testing cognitive decline in dementia, Parkinson’s, and Alzheimer's disease

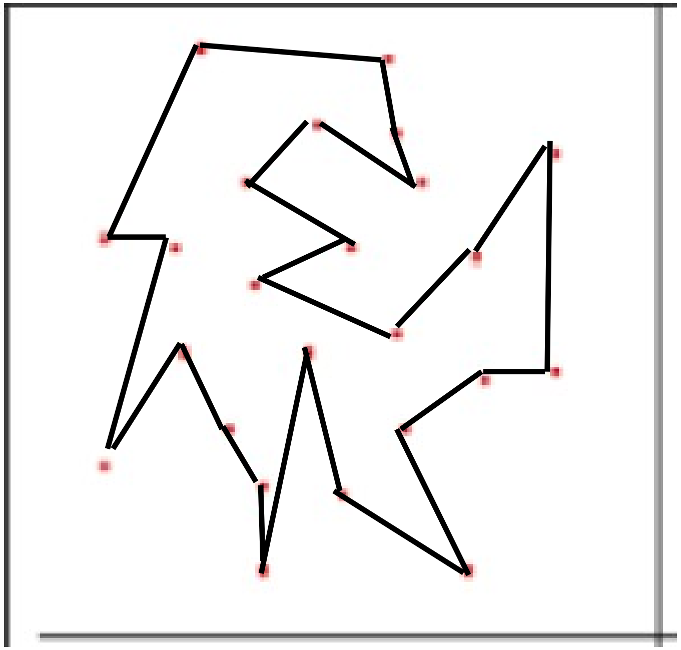
Q. Can you draw the clock with the current time?

Polygonization

Vertices are given but their ordering, i.e., labels are not given; the goal is to construct a simple polygon spanning all vertices



one way of polygonization



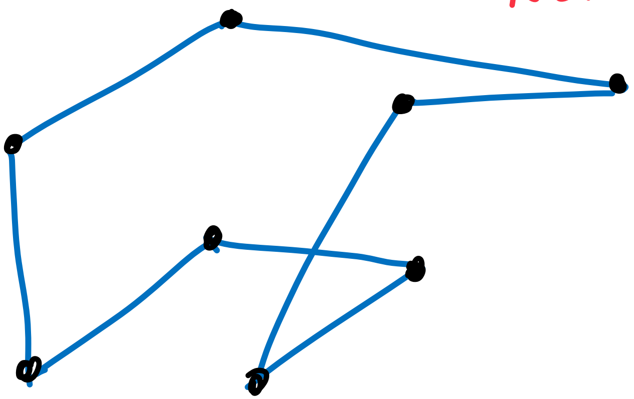
another way of polygonization

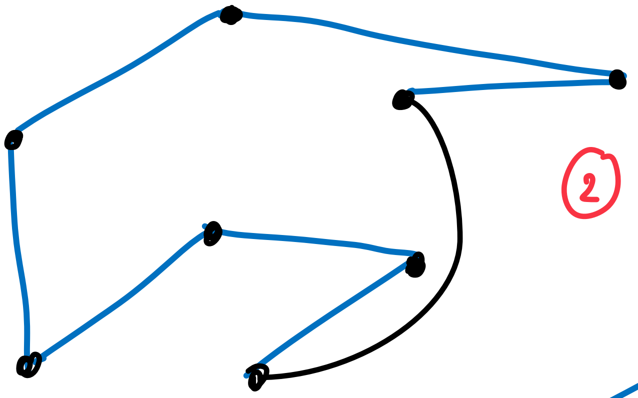
Polygonization

Given a set of points construct a simple polygon that spans all points

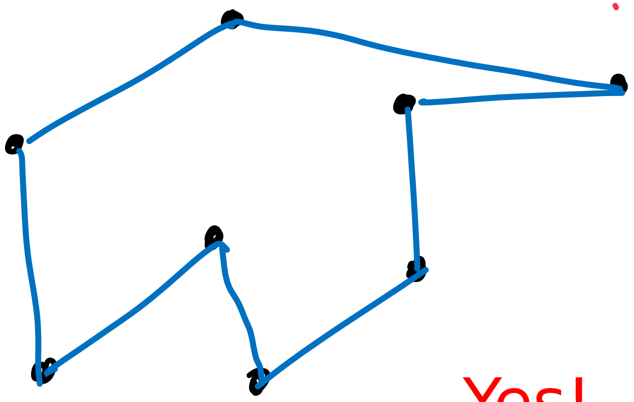
unordered

Not simple





② Does it always exist ?

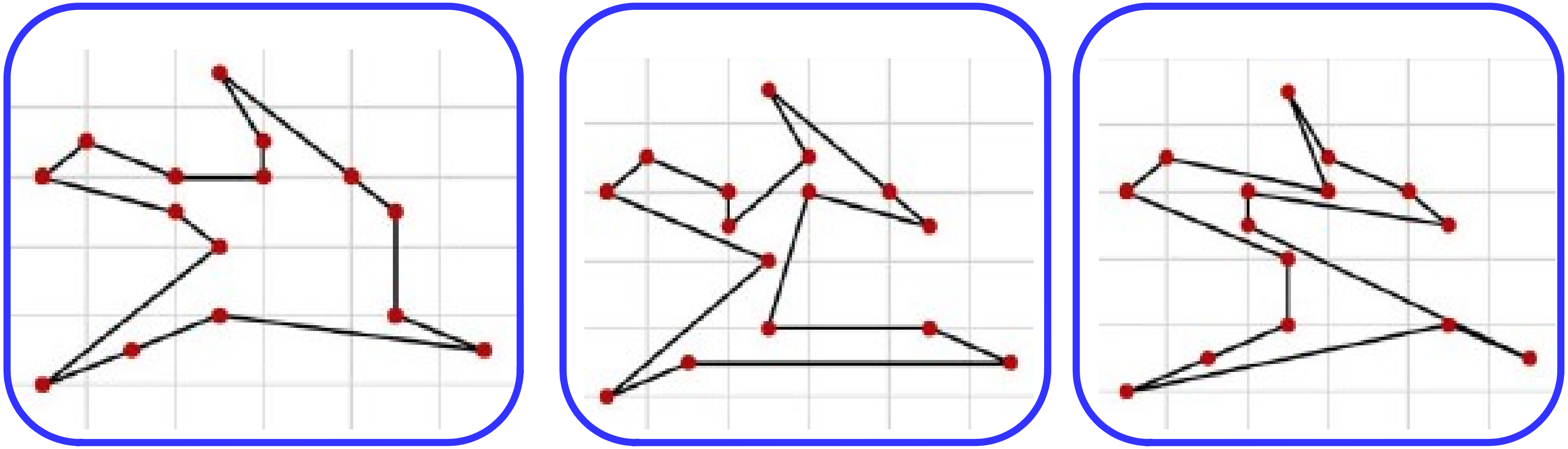


Yes!

① How to sequence points so as to guarantee polygonization

How?
HMK - 01

Polygonization



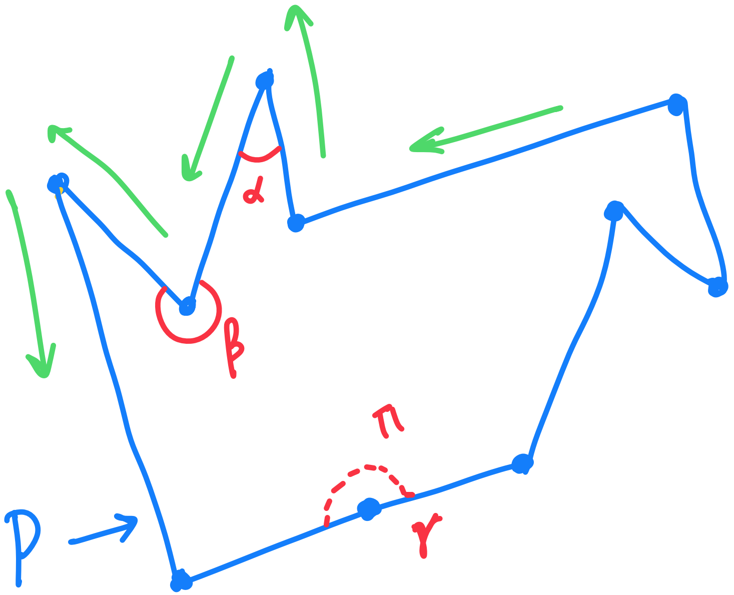
Different polygonizations of the *same set* of points

Q1. Can you find the one with minimum perimeter, area?

Q2. An unordered point set *P* and some edges *E* defined on a subset of *P*, are given. Can you always polygonise such that it includes all edges in *P*?

Q3. An unordered point set *P* and a hole *H* are given. Can you polygonise *P* such that *H* appears as a hole in *P*?

Simple polygons: Convex and reflex angles



internal angles of P

α: strictly convex $< \pi$

β: strictly concave $> \pi$
(also called reflex)

γ: assume convex $= \pi$

(We often consider collinearity as degenerate cases)

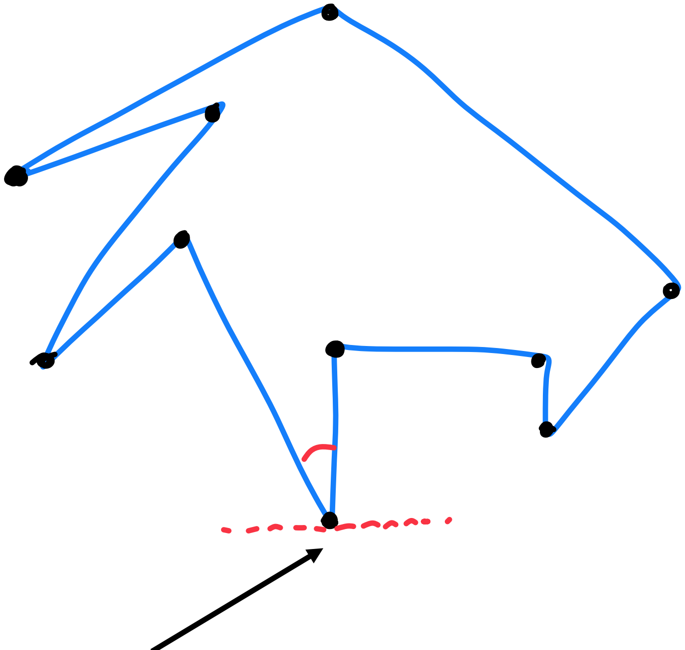
Walk along δP CCW

Left turn $\Rightarrow$ strictly convex

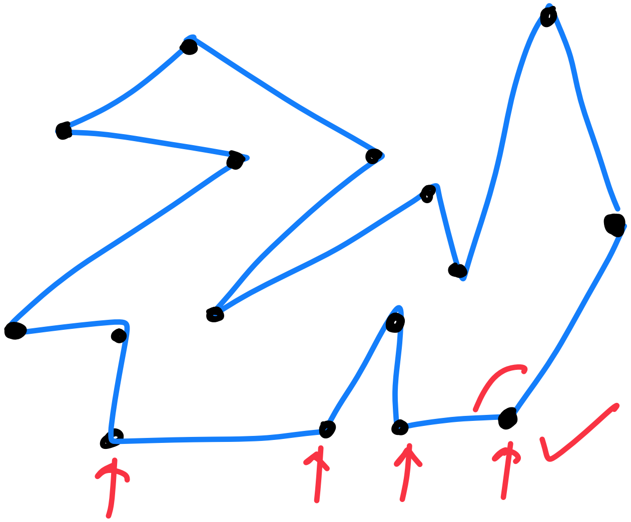
Right turn $\Rightarrow$ strictly reflex

No turn $\Rightarrow$ π

Lemma: Every polygon has at least one strictly convex vertex.

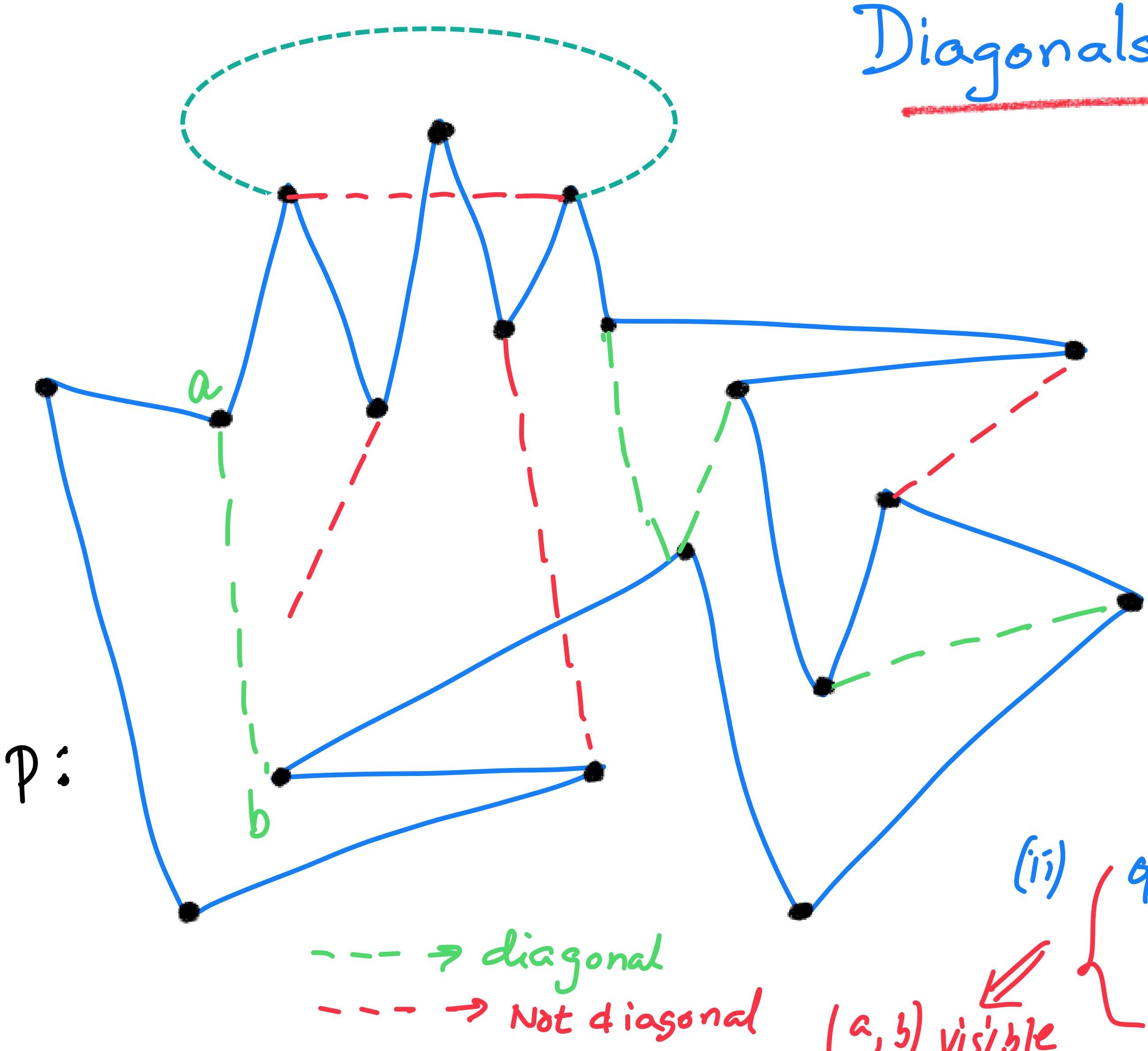


vertex with least *y*-coordinate



lowest, rightmost

Diagonals in a Polygon



straight-line

A line segment $\overline{ab}$ is a diagonal of P if

(i) a, b are two non-adjacent vertices of P

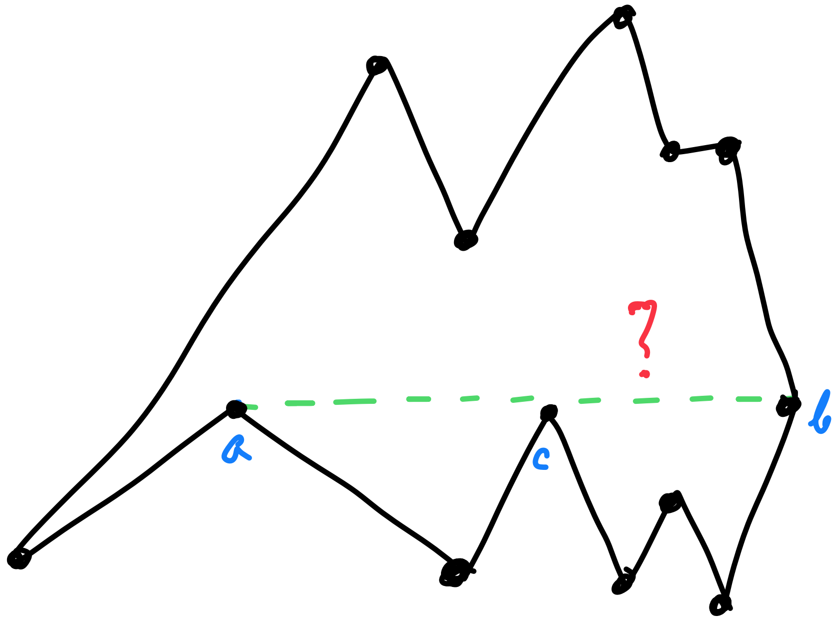
&

(ii) open $\overline{ab}$ lies completely within P, i.e.,

$\overline{ab} \in P$ &

Closed $\overline{ab} \cap \delta P = \{a, b\}$

(a, b) visible pair



Is $\overline{ab}$ a diagonal?

grazing

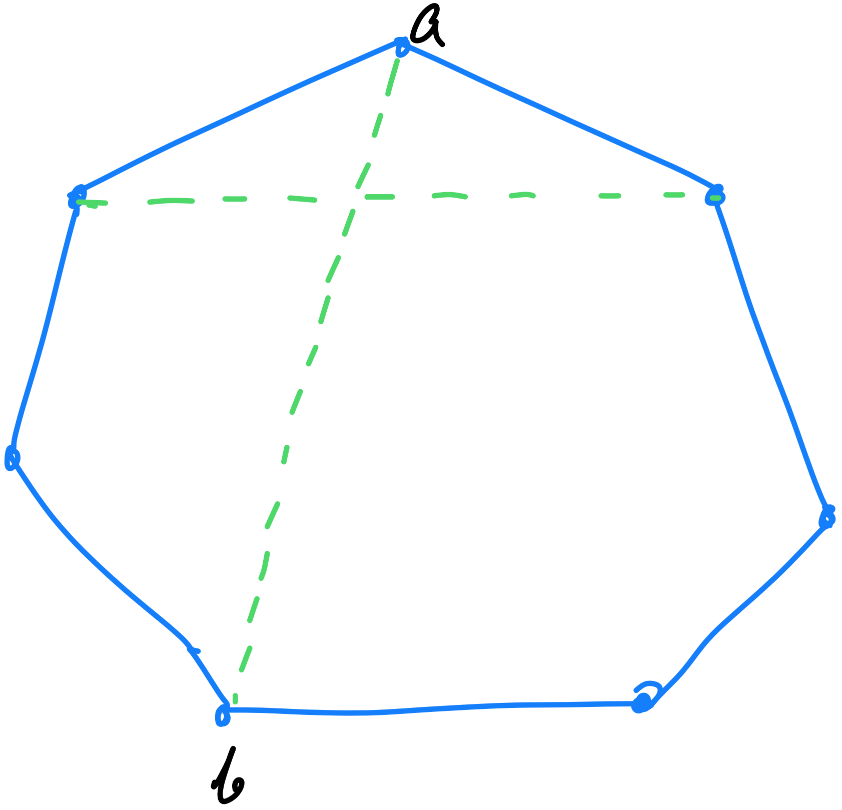
(degenerate case)

(ii) open $\overline{ab}$ lies completely within P, i,e,

$\overline{ab} \in P$ &

closed $\overline{ab} \cap \delta P = \{a, b\}$

Convex polygons

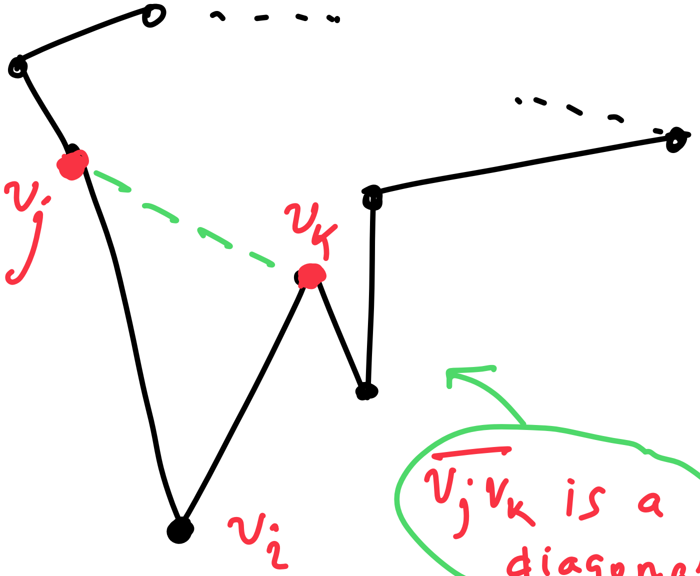


Any line segment $\overline{ab}$ joining two non-adjacent vertices of a convex polygon P is a valid diagnonal.

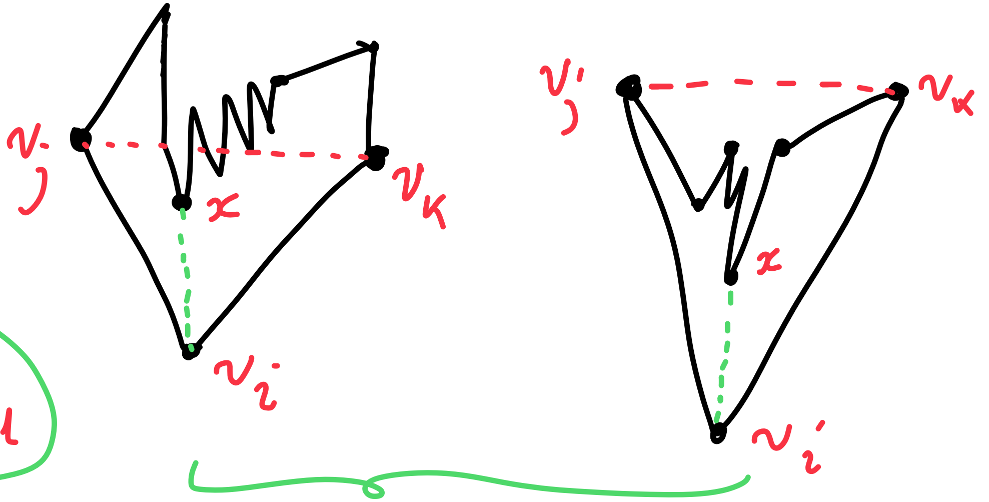
diagonals in a convex P with n vertices

$$\boxed{= \binom{n}{2} - n}$$

Lemma: Every polygon $P(n)$, $n \geq 4$ must have a diagonal



$\overline{v_j v_k}$ is a diagonal



Pick a strictly convex vertex v_i

$v_j, v_k \rightarrow$ immediate neighbor of v_i

$\triangle v_i v_j v_k$ must contain at least one vertex x that is closest to v_i

$\Rightarrow \overline{v_i x}$ is a diagonal

J. O'Rourke: *Computational Geometry in C*, Cambridge Univ. Press, 1998

CS60064 Spring 2022

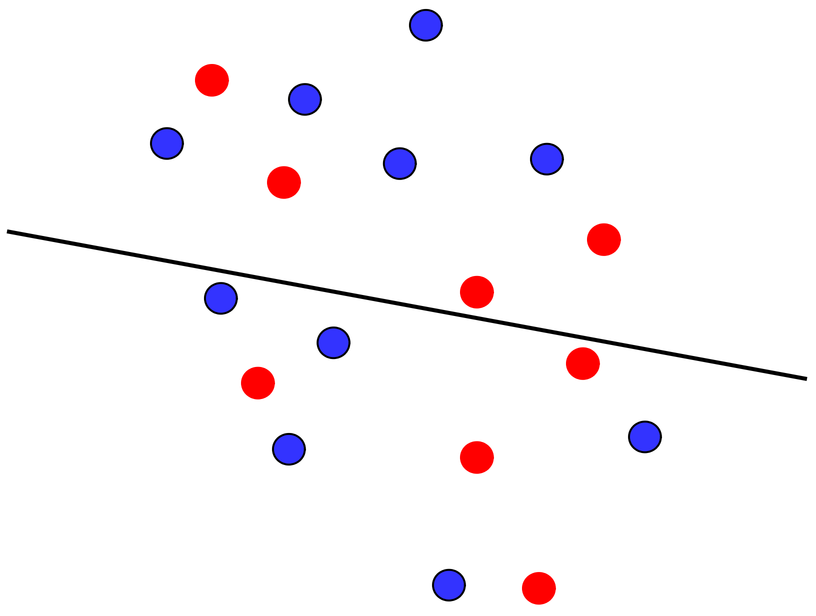
Computational Geometry

Instructors

Bhargab B. Bhattacharya (BBB)
Partha Bhowmick (PB)
Lecture 05 & Lecture 06
14 January 2022

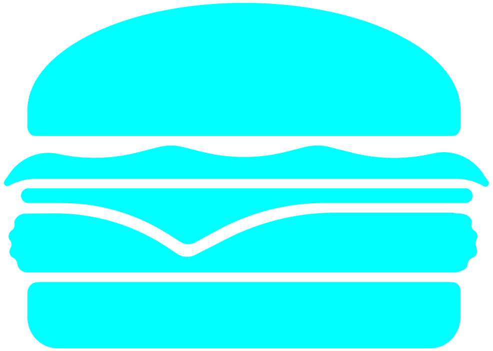
Indian Institute of Technology Kharagpur
Computer Science and Engineering

Problem of the Day: Magical Cut

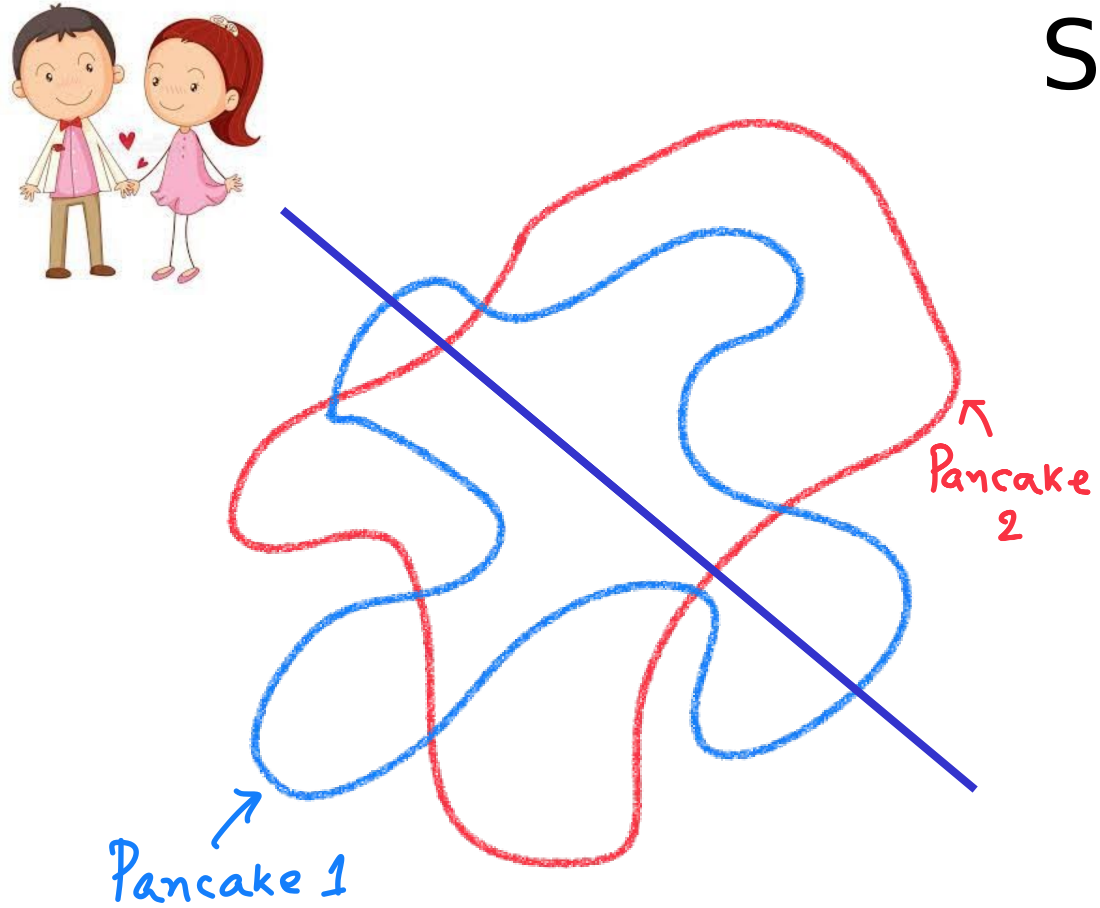


Question: Given $2m$ red points and $2n$ blue points in the plane in general positions, is it possible to divide them in half each, by a *single* straight-line cut?

Story of Pancake and Sandwich



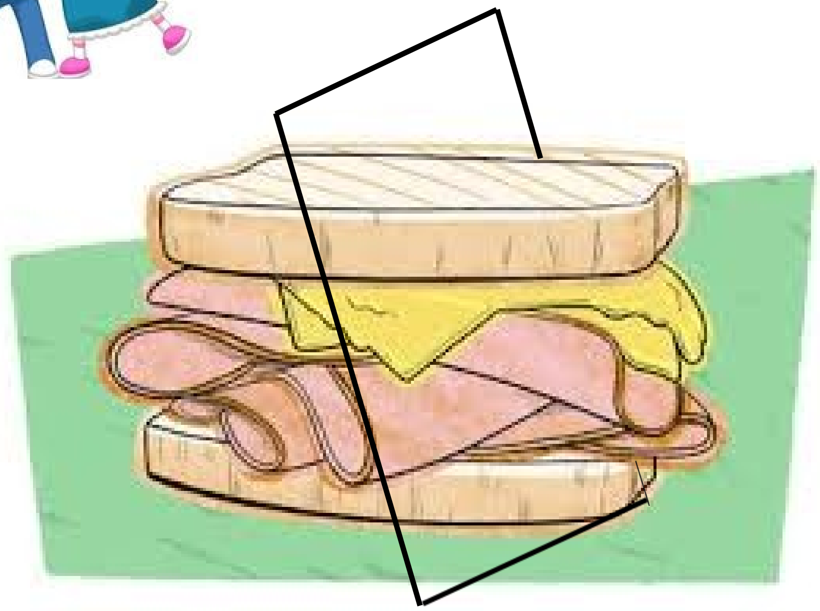
Story of Pancake

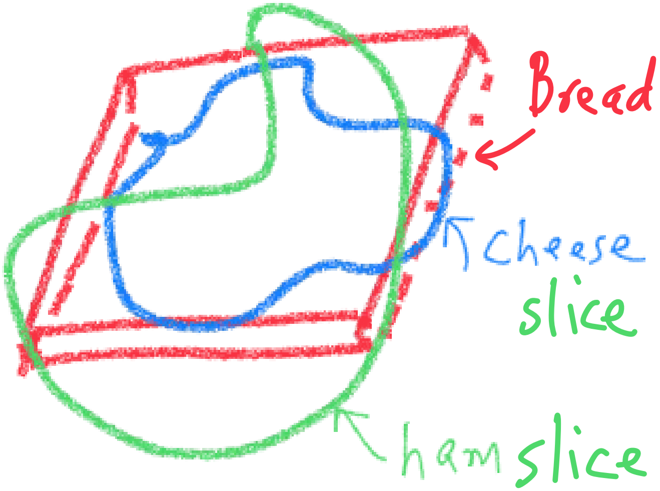


Cut by a straight knife (straight line cut) 2D

Pancake Theorem: It is *always possible* to cut the stack of two arbitrarily-shaped pancakes into two equal-size (area) portions each, by a single straight knife-cut, without moving them relative to each other

Story of Sandwich





cut by a 2D hyperplane (3D)

Ham-Sandwich Theorem: Given *n* measurable objects in *n*-dimensional Euclidean space, it is possible to divide them in half each, by a single (*n* -1)-dimensional hyperplane

Today’s Agenda

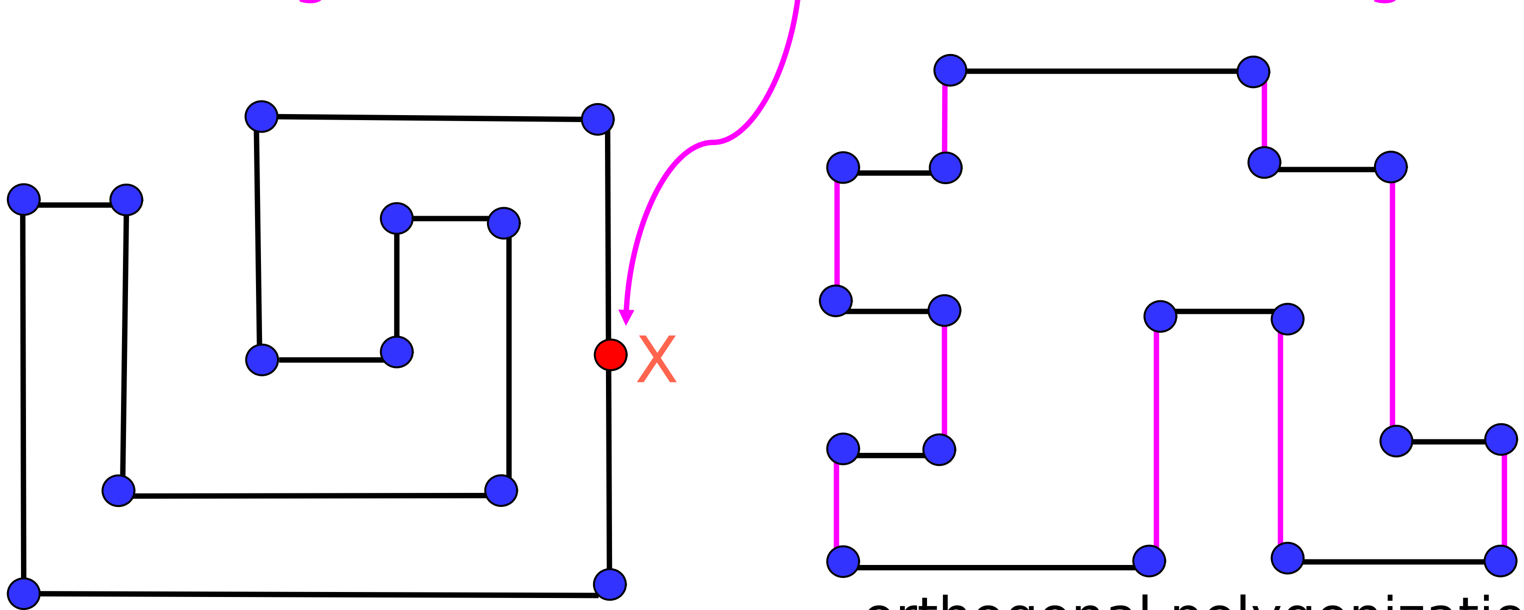
1. Orthogonal polygonization
2. Triangulation of simple polygons

Orthogonal Polygons

All edges are axis-parallel

In other words, internal turn angles are either π/2 or 3π/2

Avoid degenerate cases where internal angle is π



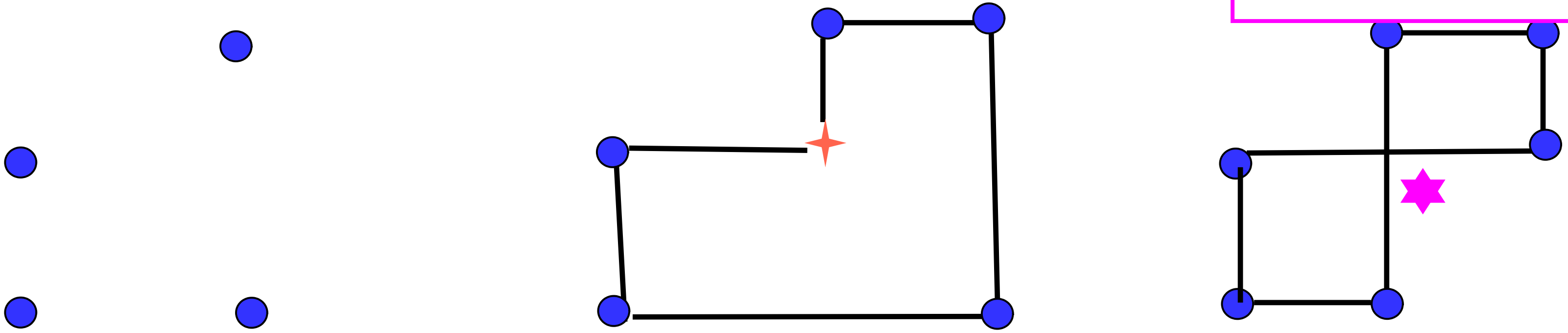
orthogonal polygonization (from unlabelled vertices to polygon)

Polygonization: Vertices are given but their ordering, i.e., labels are not given; the goal is to construct an orthogonal polygon spanning all vertices

Orthogonal Polygonization

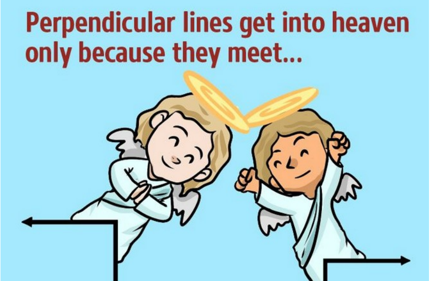
Every horizontal row or vertical column must have even number of vertices (assuming no degeneracy with internal angle ༂)

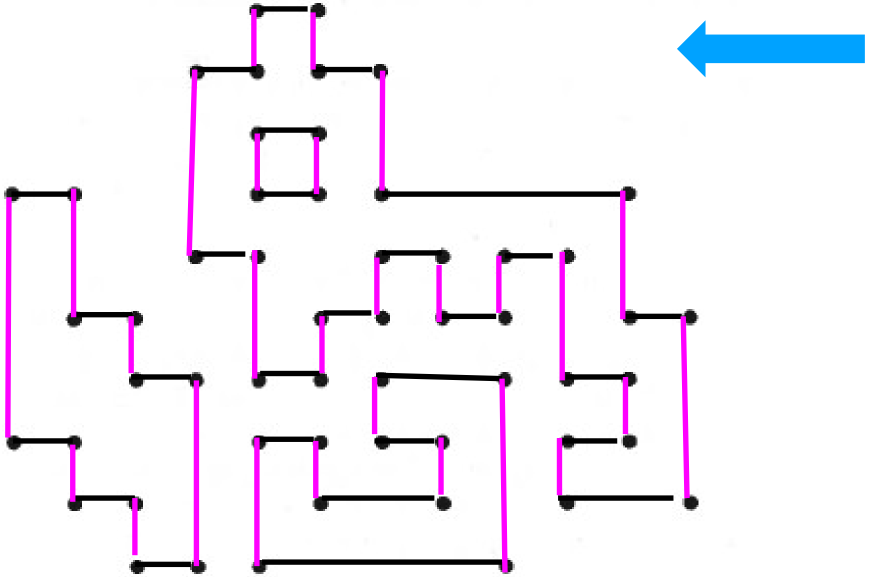
not sufficient

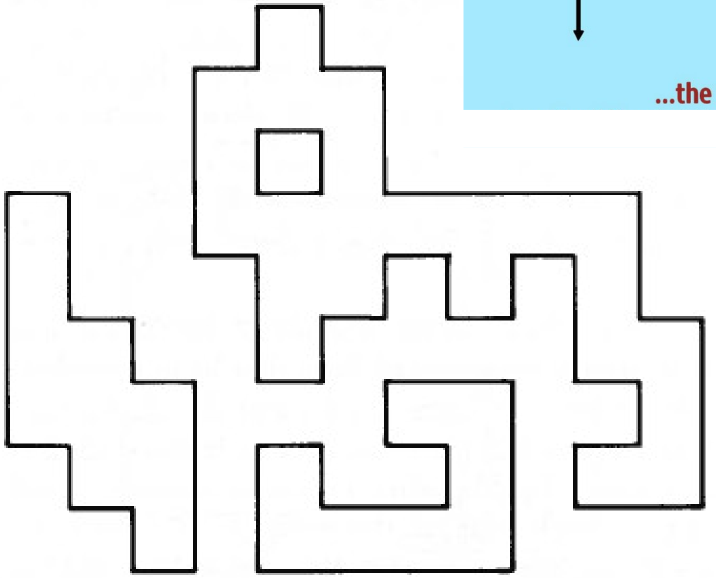


Instances which are not ortho-polygonizable

Orthogonal Polygonization


Perpendicular lines get into heaven only because they meet...
...the right angels!





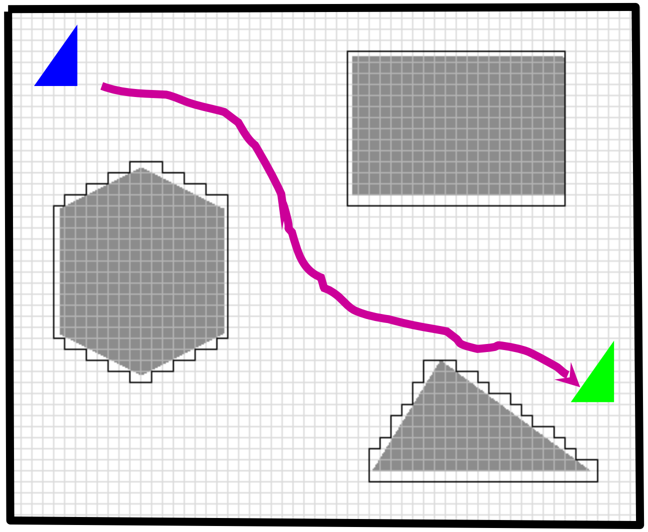
From point set to ortho-polygonization

orthogonal polygons with holes

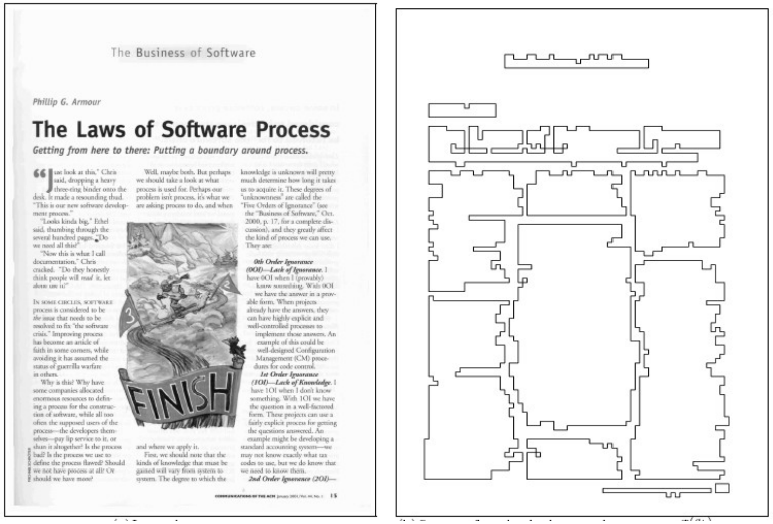
If polygonization is possible, then the solution is unique; may generate multiple polygons and with holes;

Can be accomplished in $O(n \log n)$ time, where n is the number of points

Why Orthogonal?



Robot-path configuration



The Business of Software

Phillip G. Armour

The Laws of Software Process

Getting from here to there: Putting a boundary around process.

"Just look at this," Chris said, dropping a heavy three-ring binder onto the desk. It made a resounding thud. "This is our new software development process."

"Looks kinda big," Ethel said, thumbing through the several hundred pages. "Do we need all this?"

"Now this is what I call documentation," Chris cracked. "Do they honestly think people will *read* it, let alone use it?"

In some circles, software process is considered to be *the* issue that needs to be resolved to fix "the software crisis." Improving process has become an article of faith in some corners, while avoiding it has assumed the status of guerrilla warfare in others.

Why is this? Why have some companies allocated enormous resources to defining a process for the construction of software, while all too often the supposed users of the process—the developers themselves—pay lip service to it, or shun it altogether? Is the process bad? Is the process we use to define the process flawed? Should we not have process at all? Or should we have more?

Well, maybe both. But perhaps we should take a look at what process is used for. Perhaps our problem isn't process, it's what we are asking process to do, and when and where we apply it.

First, we should note that the kinds of knowledge that must be gained will vary from system to system. The degree to which the knowledge is unknown will pretty much determine how long it takes us to acquire it. These degrees of "unknownness" are called the "Five Orders of Ignorance" (see the "Business of Software," Oct. 2000, p. 17, for a complete discussion), and they greatly affect the kind of process we can use. They are:

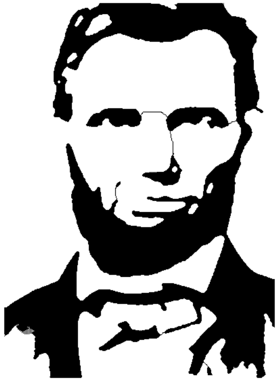
0th Order Ignorance (0OI)—Lack of Ignorance. I have 0OI when I (provably) know something. With 0OI we have the answer in a provable form. When projects already have the answers, they can have highly explicit and well-controlled processes to implement those answers. An example of this could be well-designed Configuration Management (CM) procedures for code control.

1st Order Ignorance (1OI)—Lack of Knowledge. I have 1OI when I don't know something. With 1OI we have the question in a well-factored form. These projects can use a fairly explicit process for getting the questions answered. An example might be developing a standard accounting system—we may not know exactly what tax codes to use, but we do know that we need to know them.

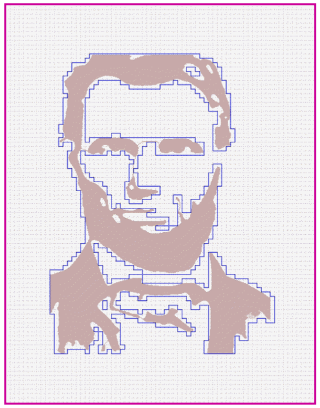
2nd Order Ignorance (2OI)—

COMMUNICATIONS OF THE ACM January 2001/Vol. 44, No. 1 15

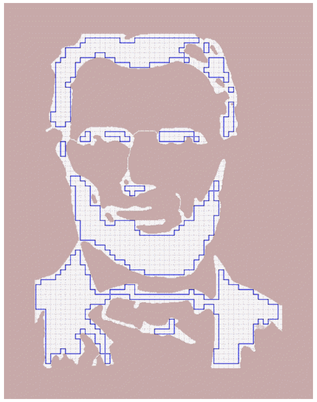
Document image segmentation



Original



Outer approximation



Inner approximation

huge space savings

Triangulation of Simple Polygons

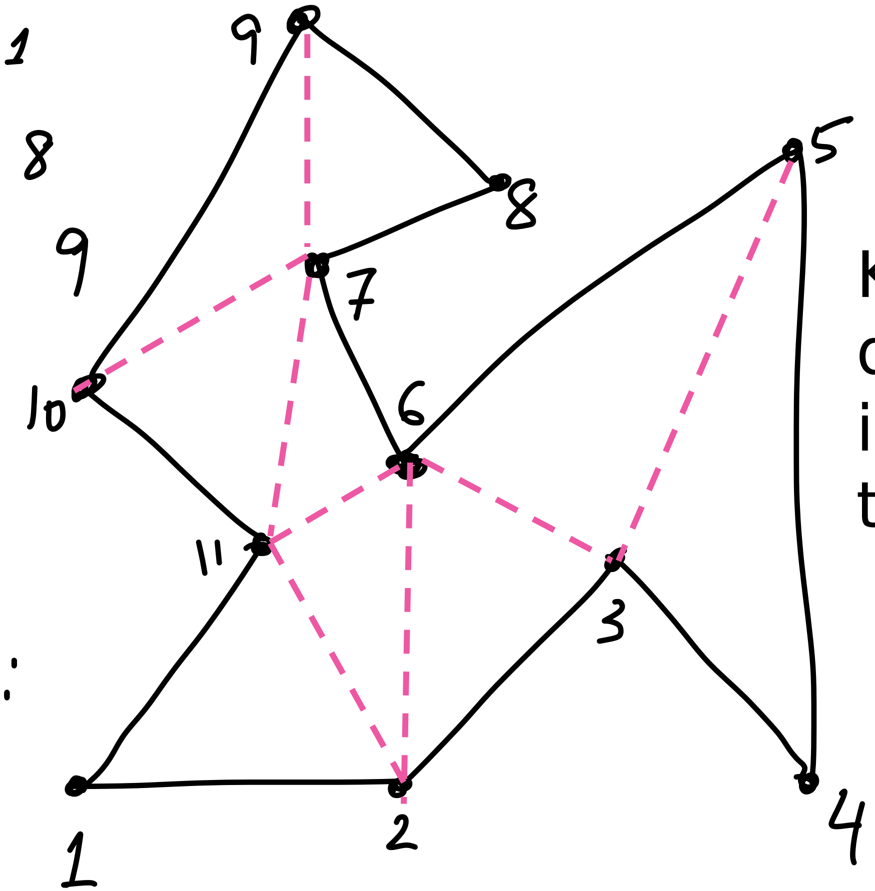
Triangulation of a simple polygon

$n = 11$

$\# D = 8$

$\# T = 9$

P:

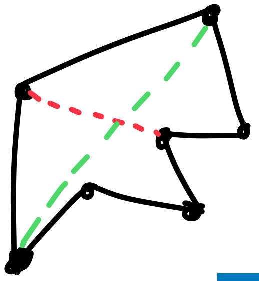


Keep on adding diagonals so that the P is partitioned into triangles

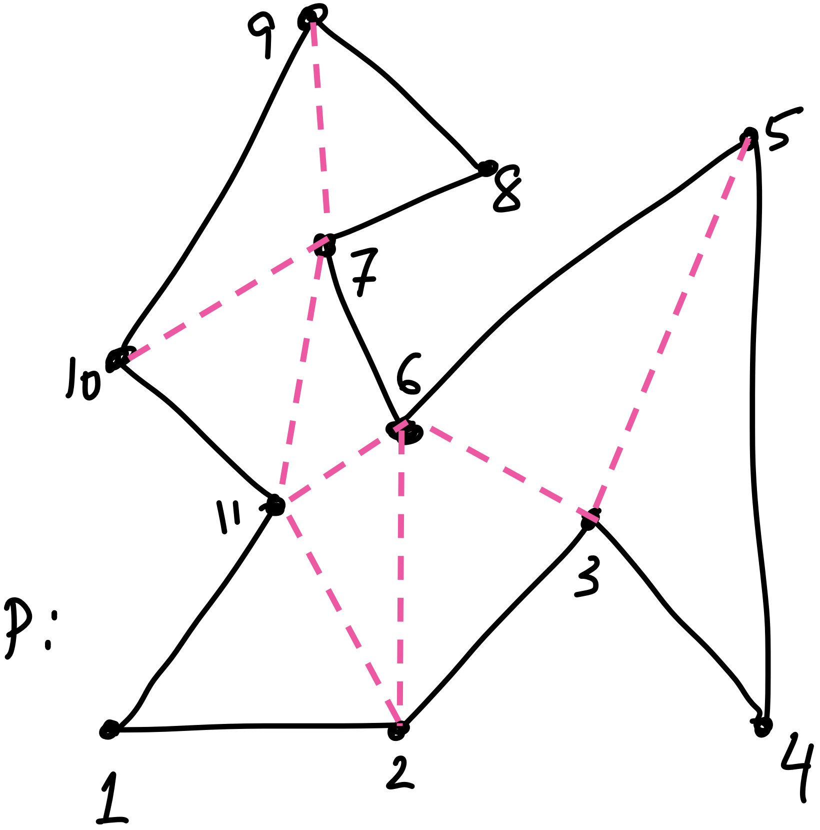
Claim: Triangulation is always possible

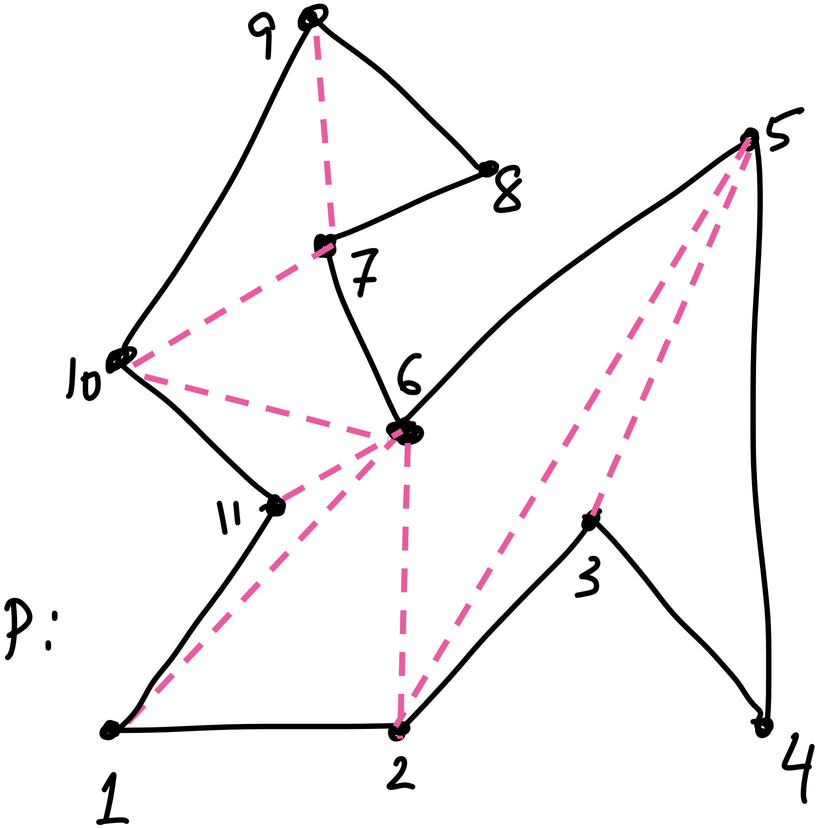
No crossing diagonals allowed

Vertices of all Δs must be the vertices of polygon



Triangulation: A partition of *P* into triangles by a *maximal* set of non-crossing diagonals.

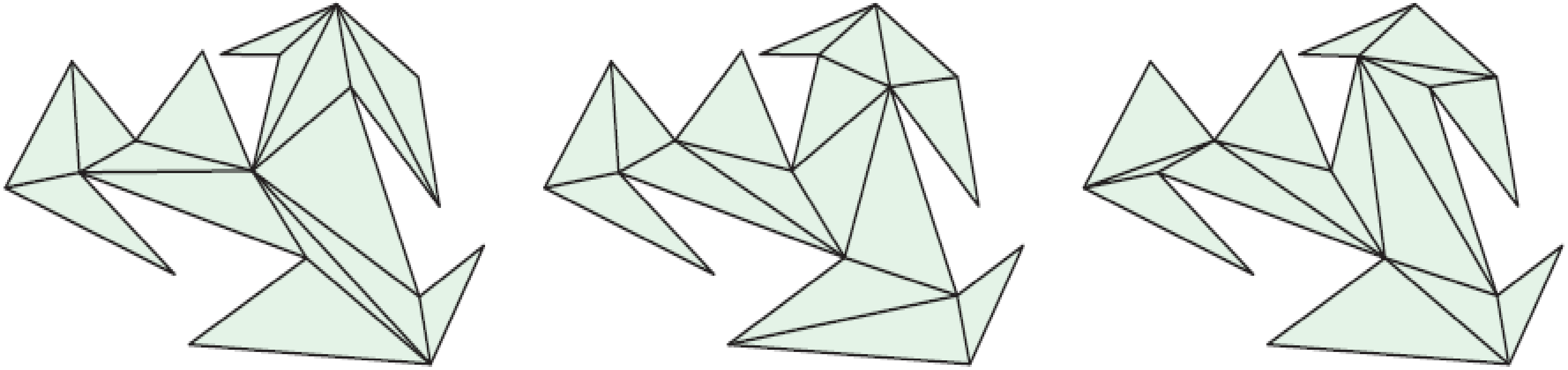




Triangulation is not unique

Triangulation of a simple polygon

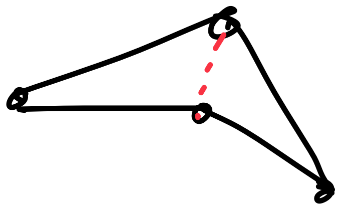
Triangulation is not unique



However, #D = *n*-3; #T = *n*-2, for all cases

Theorem: A simple polygon $P(n)$ $n \geq 4$ can always be triangulated using exactly $(n-3)$ diagonals that partition $P(n)$ into $(n-2)$ triangles.

Proof: Basis



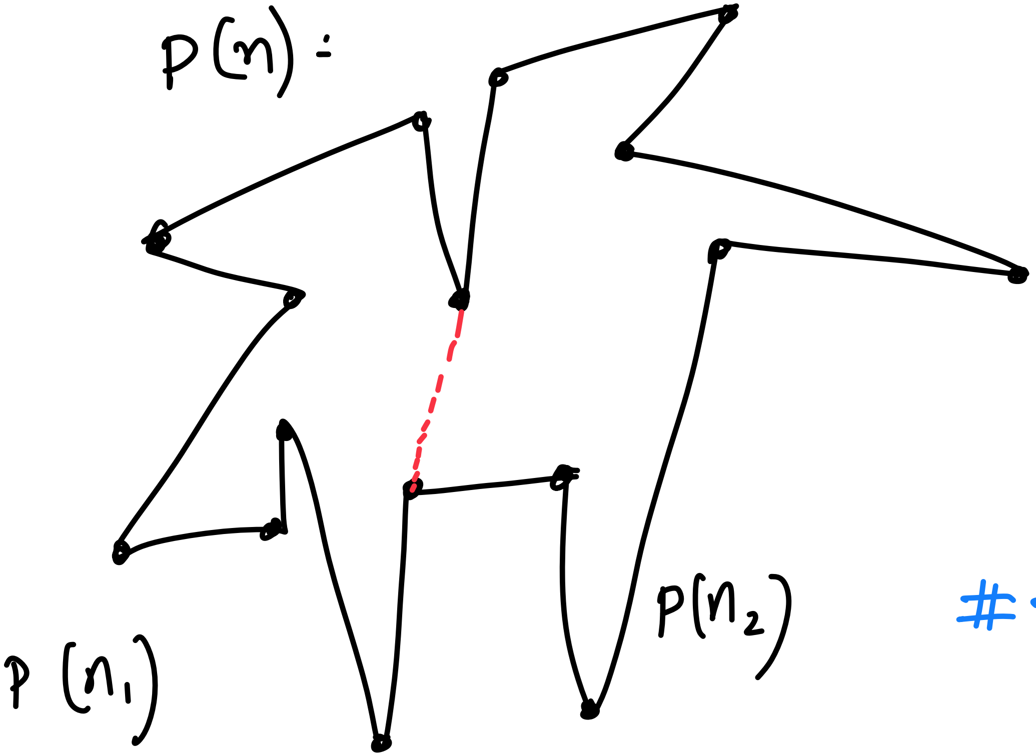
$n = 4$

$\# D = 1$

$\# T = 2$

Theorem holds for base case

Assume it holds for $k < n$



By induction hypothesis,

$\#$ diagonals

$= (n_1 - 3) + (n_2 - 3) + 1$

$= n_1 + n_2 - 5 = \boxed{n-3}$

$\#$ triangles

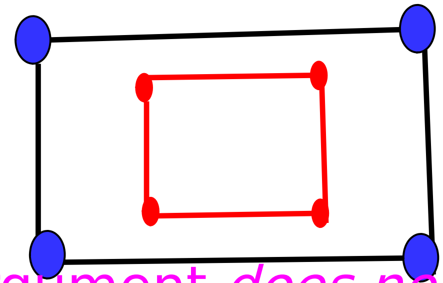
$= (n_1 - 2) + (n_2 - 2)$

$= n_1 + n_2 - 4 = \boxed{n-2}$

$n_1 + n_2 = n + 2$

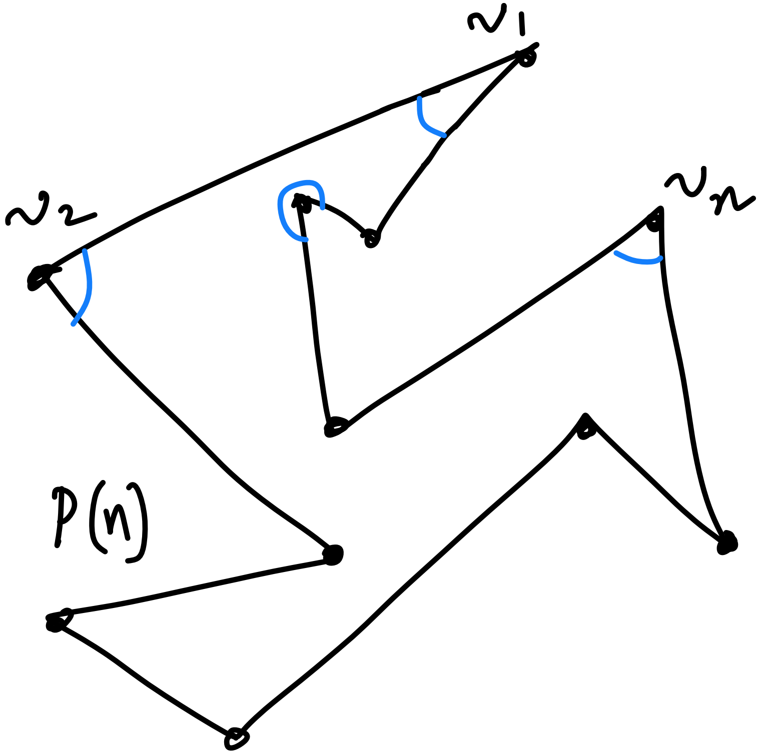
(*) A valid triangulation always exists

To prove: # D = *n* – 3; # T = *n* – 2



Same argument *does not* hold good for polygons with holes!

Corollary :

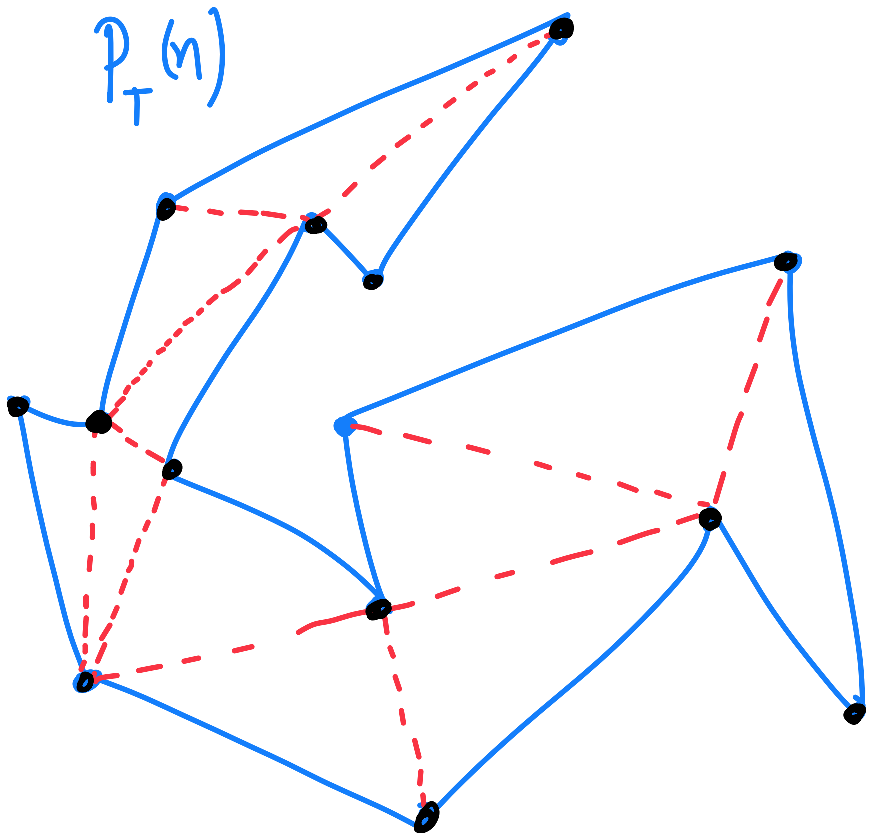


$$\sum_{1}^{n} \alpha_i =$$

$\alpha_i \in$ internal angles of P(n)

$$\text{Area of } P(n) = \sum_{i=1}^{n-2} A(T_i)$$

Triangulated polygon as a graph



$P_T(n) \Rightarrow G(V, E)$

$|V| = n, \quad |E| = n + (n-3)$

$= 2n-3$

$|F| = (n-2) + 1 = n-1$

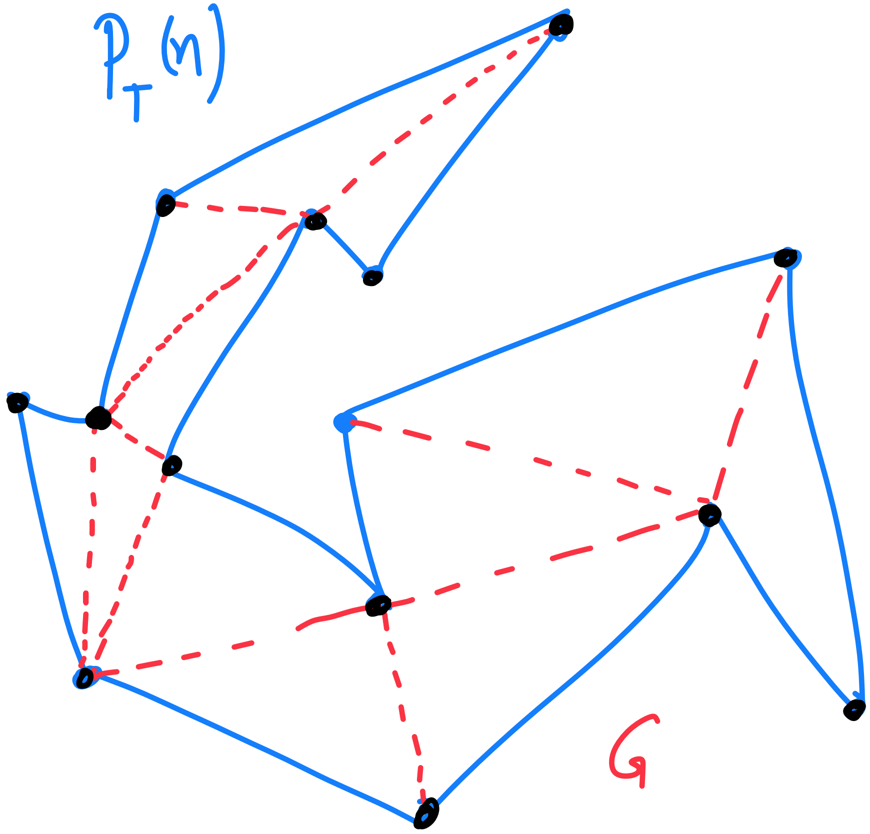
$\uparrow$ external face

G is planar

$|V| - |E| + |F| = 2$ ✓

# D = *n* – 3; # T = *n* – 2

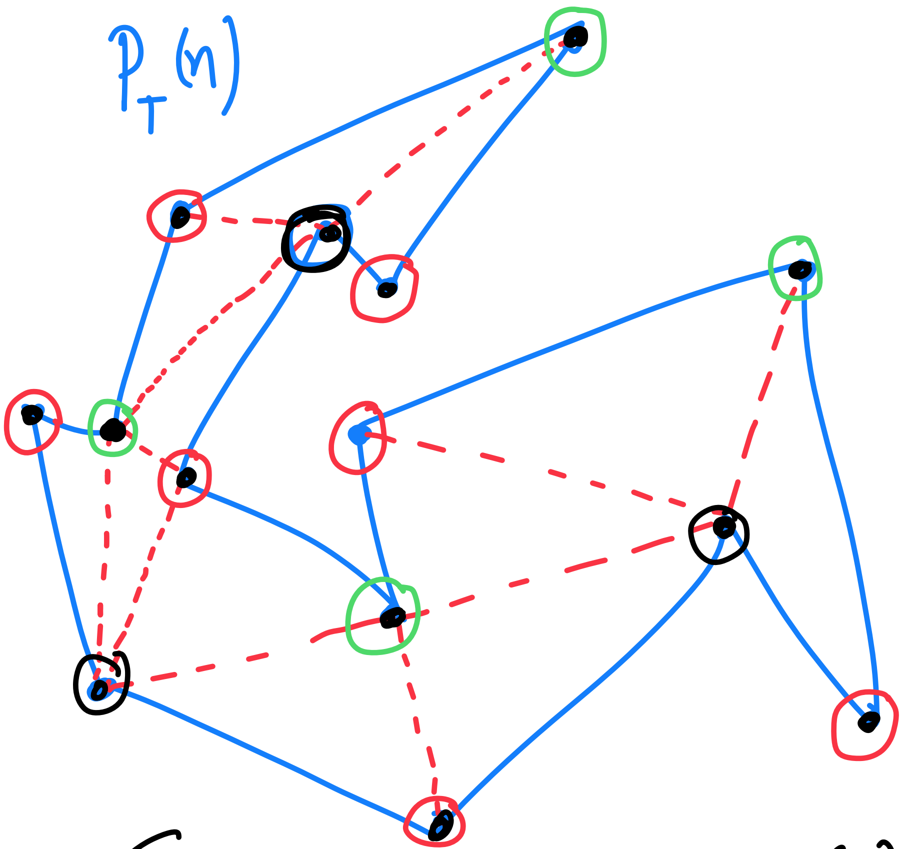
Triangulated polygon as a graph



G is a maximal outer planar graph

G is 3-colorable

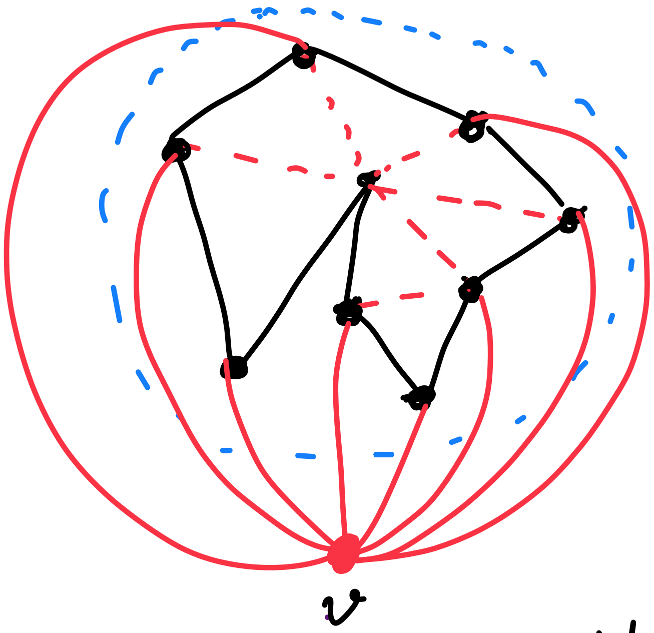
Triangulated polygon as a graph



G is 3-colorable, i.e., $\chi(G)=3$

Proof: Suppose not, i.e., $\chi(G)=4$

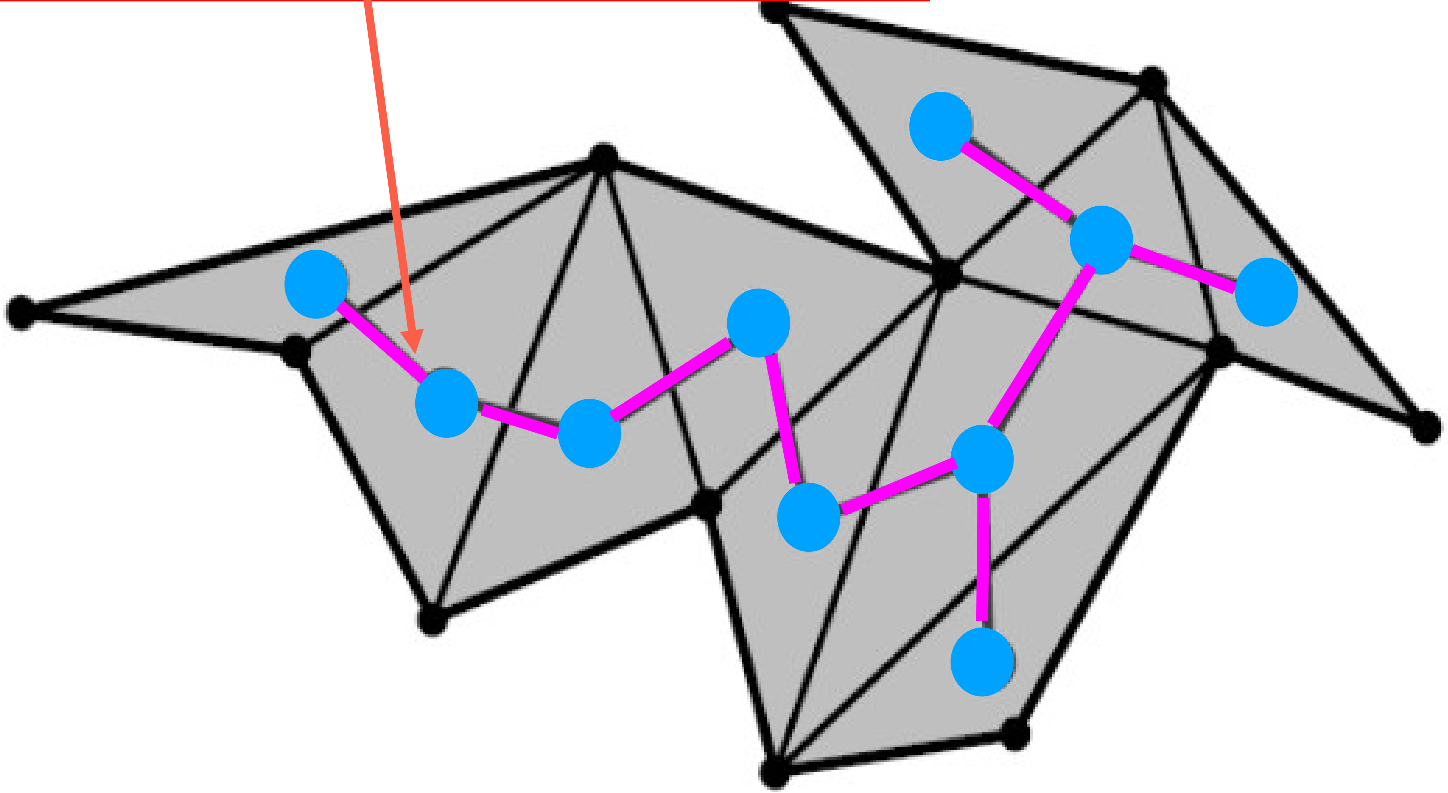
G' is also planar



Consider G' = G ∪ (v) as shown

$\chi(G')=5$
contradiction

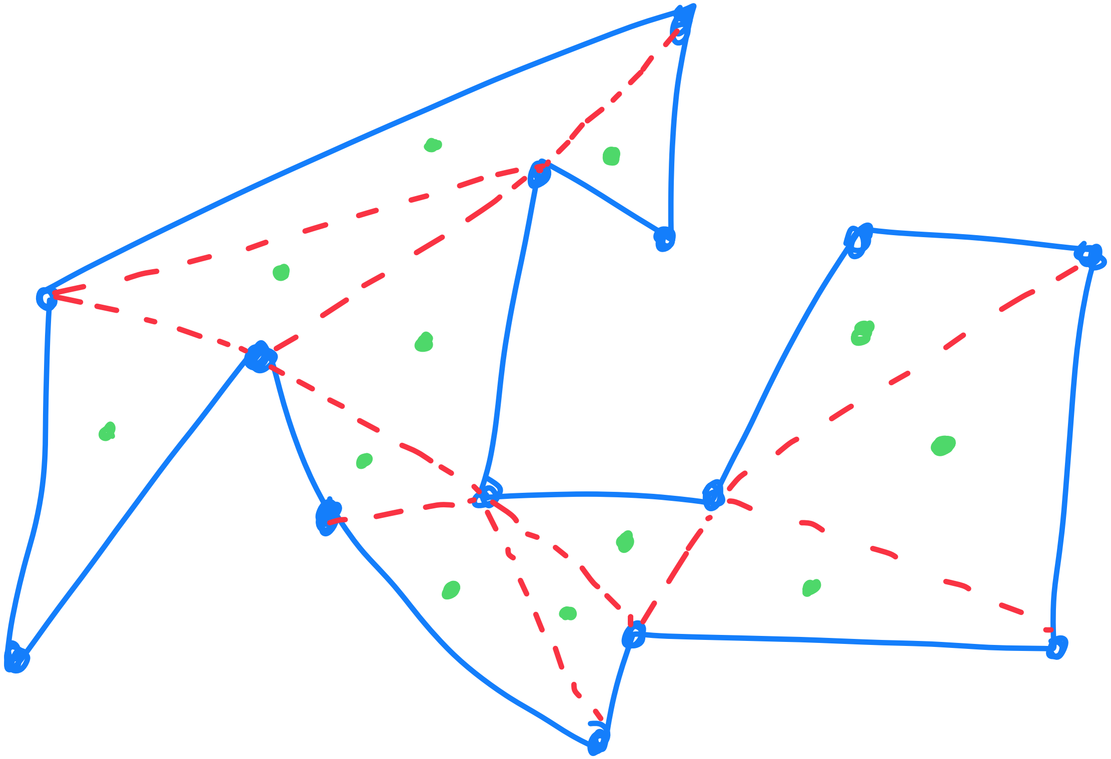
Dual of a triangulated polygon



each triangle (face) a vertex in dual graph

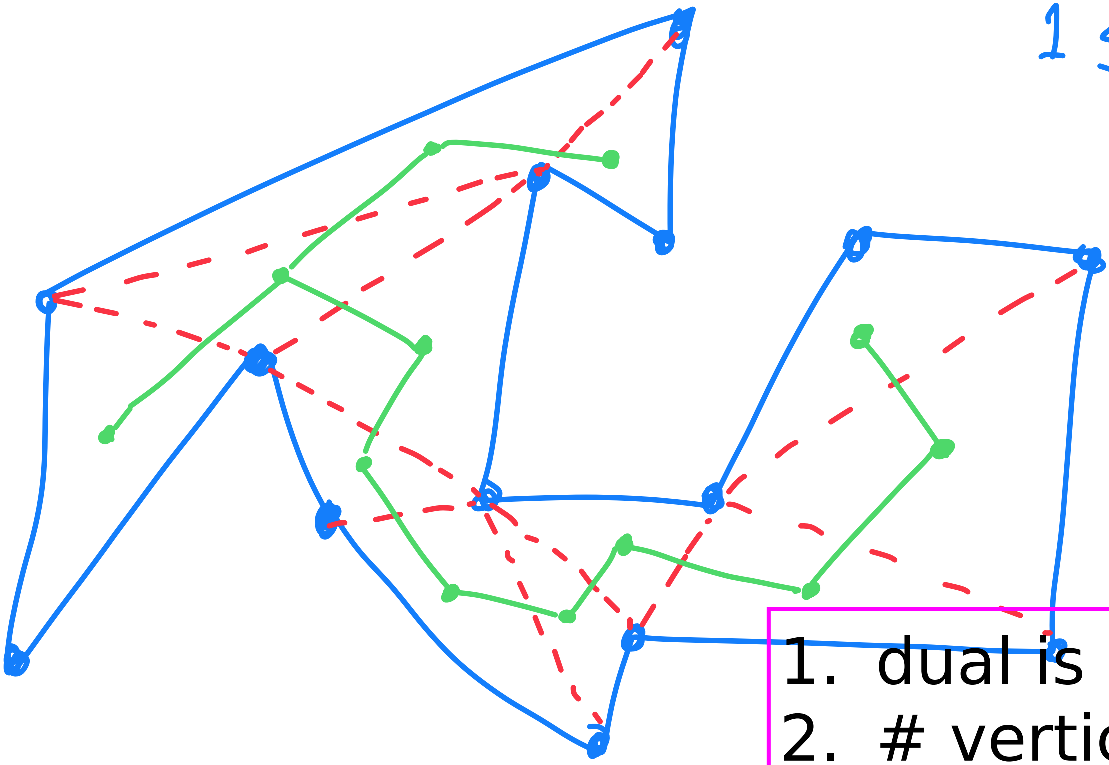
If two triangles share a diagonal, put an edge between the two corresponding vertices in the dual graph

The internal dual of a triangulated simple polygon
is a tree



The internal dual of a triangulated simple polygon is a tree T

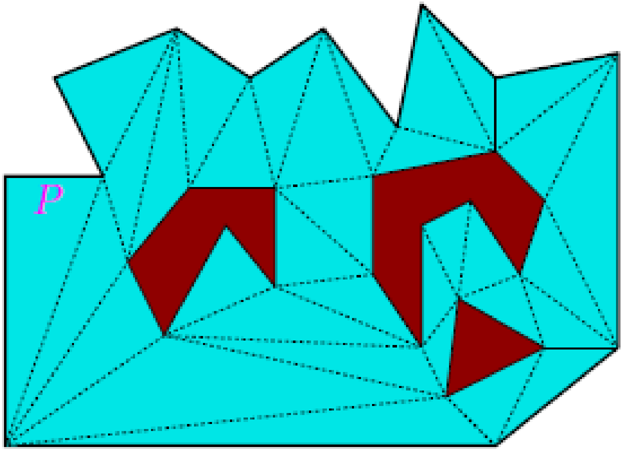
$1 \le d(v)\big|_T \le 3$ $\forall v$



# D = *n* – 3; # T = *n* – 2

1. dual is connected
2. # vertices(dual) = #T = *n* – 2;
3. # edges(dual) = #D = *n* – 3

Triangulations of a polygon with holes



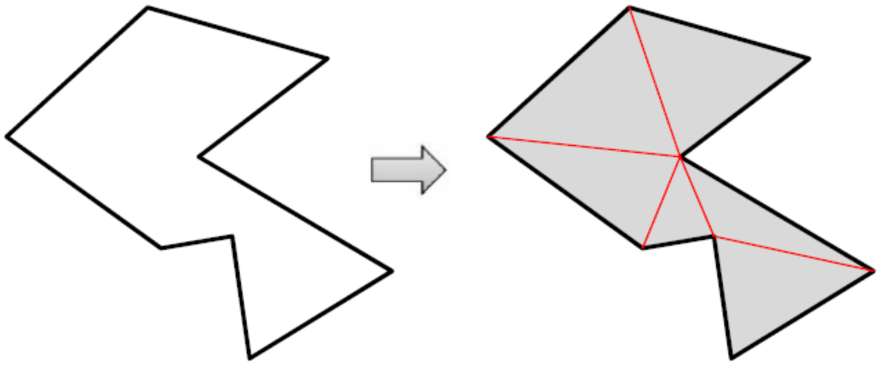
Every triangulation of a polygon with h holes with a total
of n vertices uses $n+3h-3$ diagonals and has
$n+2h-2$ triangles

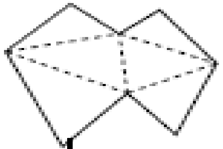
The dual graph of a triangulation of a polygon with holes

Self study

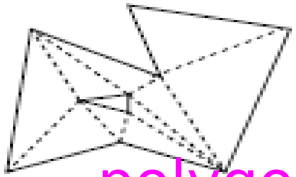
Summary and generalizations: Triangulation

Select a *maximal* set of non-intersecting diagonals or edges that subdivide the interior into triangles

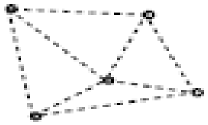




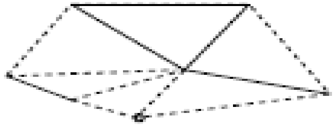
polygon



polygon with hole



point set



PSLG

Triangulation is a general concept applicable to many instances

Summary:

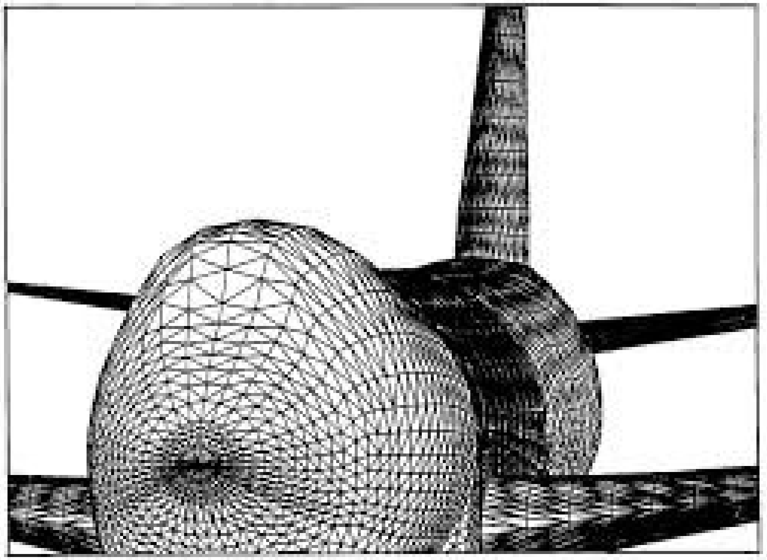
- A line segment l joining any two visible vertices of a polygon is called a *diagonal* of the polygon provided l lies completely within P

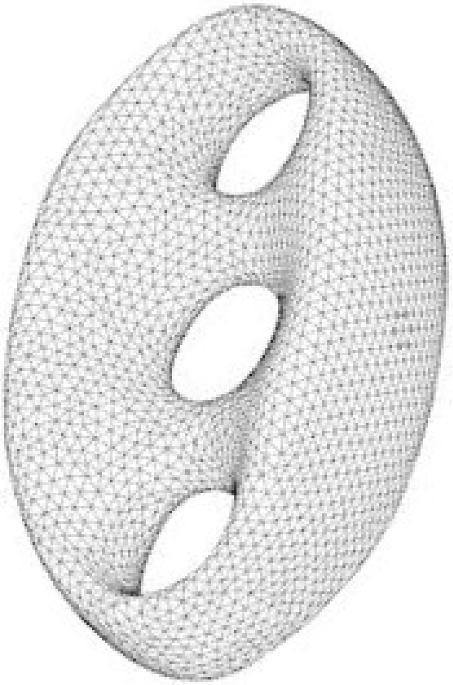
- Every triangulation of a simple polygon P of n vertices uses $n - 3$ diagonals and has $n - 2$ triangles

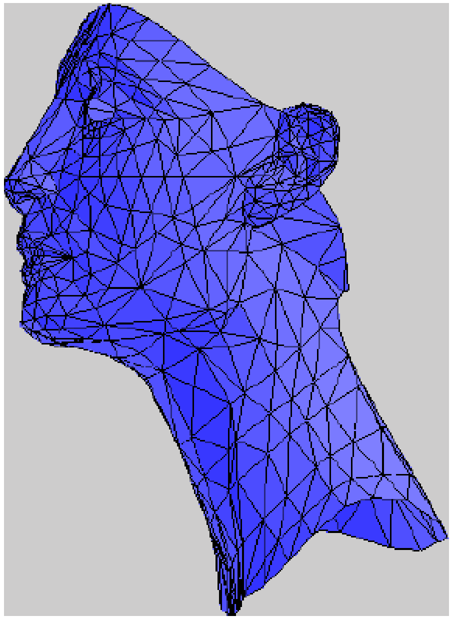
- The sum of the internal angles of a simple polygon of n vertices is $(n - 2)$ཨོཾཨཱཿཧཱུྃ

- The dual of a triangulation of a simple polygon P is a tree, while that of a

Surface Triangulation



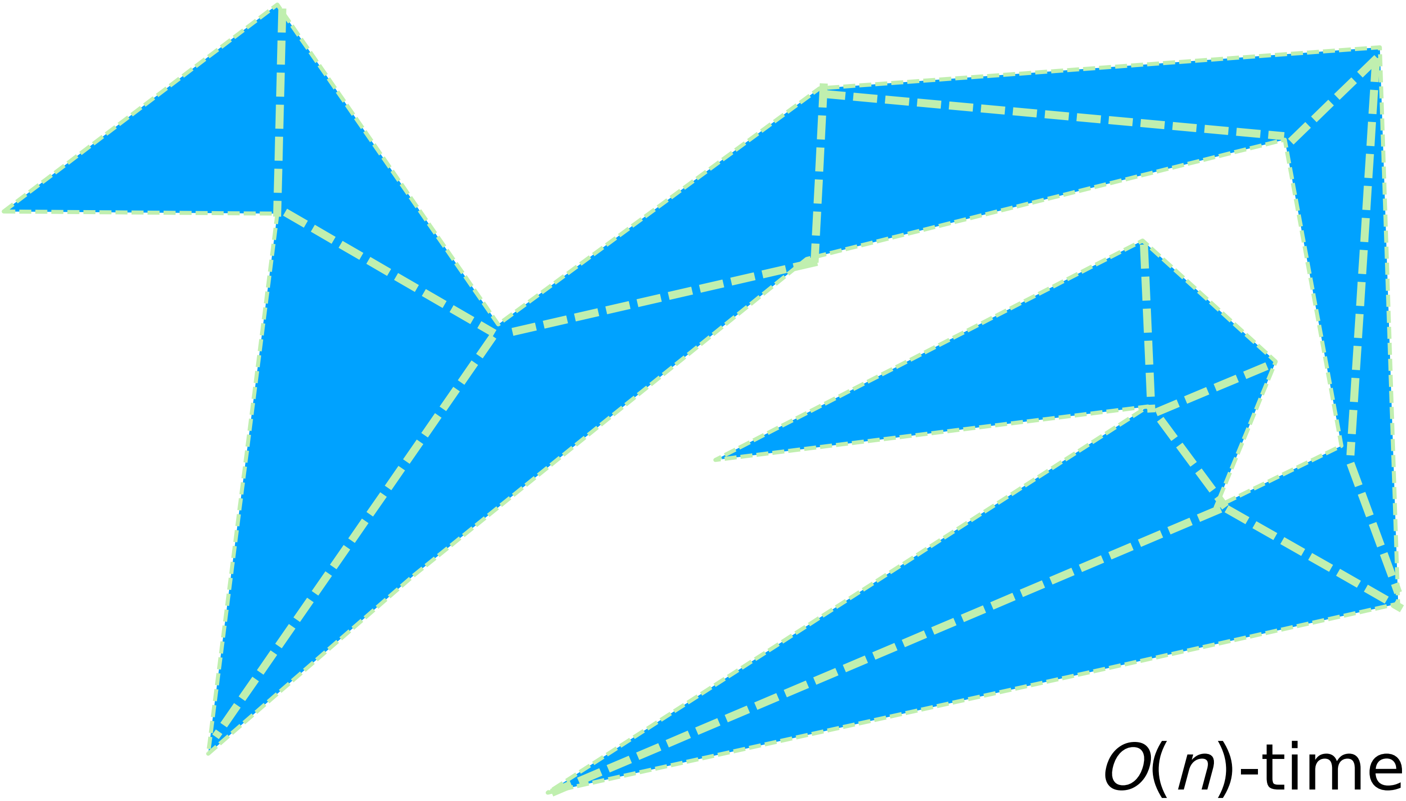


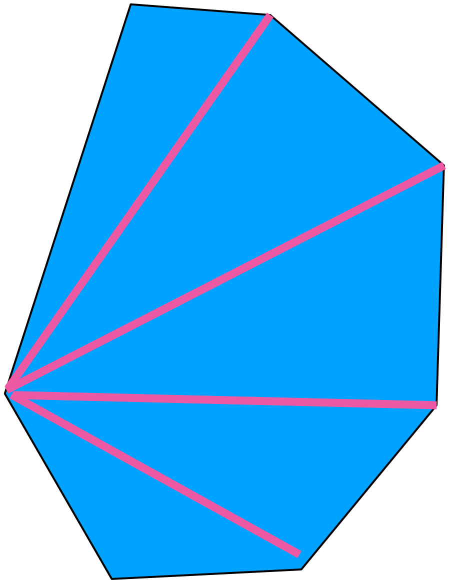


Surface triangulation of 3D objects, mesh generation, 3D modeling, visualization, computer graphics

Surface Triangulation

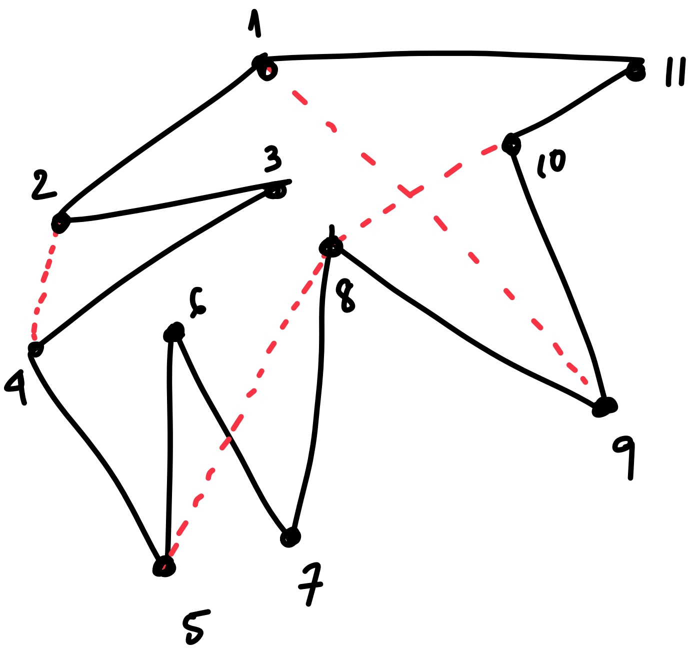
Algorithms for triangulation





$O(n)$-time fan algorithm for convex polygons

Challenges in triangulation: P_n



$D(2,4)$

$D(5,8)$

$D(1,9) \cup D(8,10)$

not allowed

1. Select $(n-3)$ sticks $\overline{ab}$ s.t. $a, b \in$ vertices of P_n
2. $\overline{ab} \in P_n$
3. $\overline{ab} \cap \delta P_n = \{a,b\}$
4. $\overline{ab}$ must not intersect with any other previously selected diagonals

i.e. $\overline{ab}$ is a diagonal

⇒ Result: $T(P_n)$, i.e., partition of P_n into $(n-2)$ Δs.

Every simple polygon P_n can be partitioned into triangles by adding $(n-3)$ diagonals

Triangulation Algorithm (Naive)

1. $\binom{n}{2} = O(n^2)$ diagonal candidates

2. Test for diagonal $\rightarrow O(n)$

Hence, to insert $(n-3)$ diagonals $\Rightarrow O(n^4)$

Interesting Fact: Triangulation algorithm time complexity

$O(n^4)$ $O(n^3)$ $O(n^2)$ $O(n\log n)$ $O(n)$

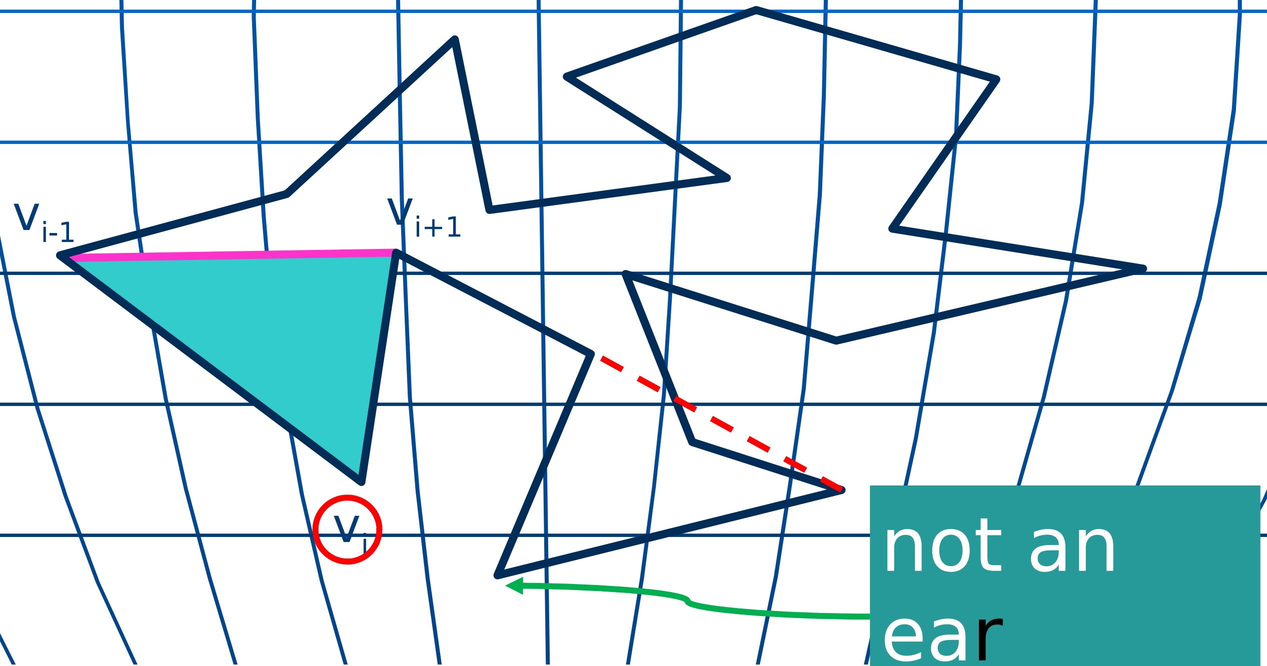
Self-Portrait by Vincent van Gogh (1853-1890)

Room where van Gogh chopped off his own ear (1888)

Polygon Triangulation by Ear-Clipping Algorithm

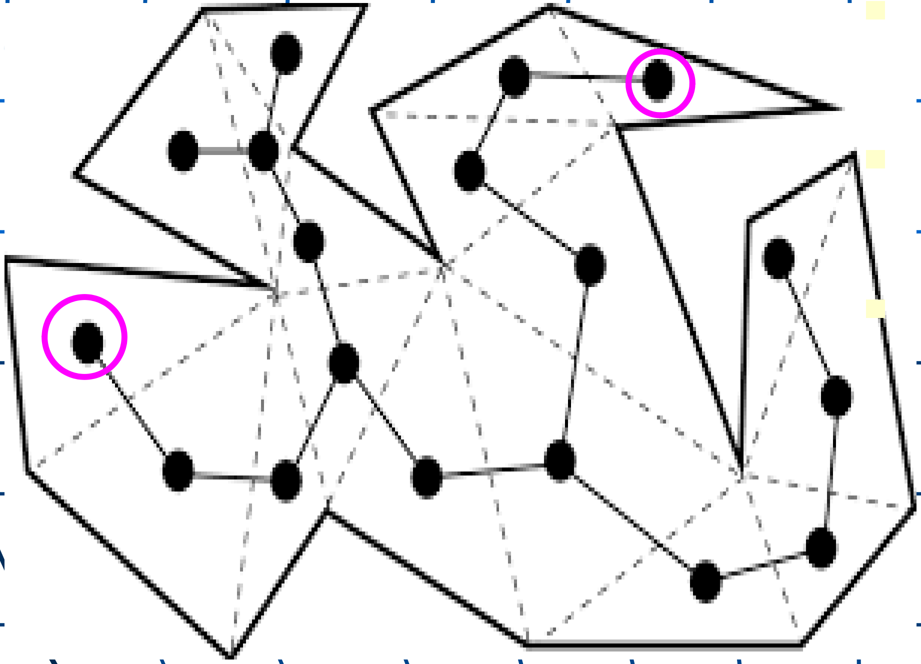
Ears in a Polygon

- Three consecutive vertices v_{i-1} v_i v_{i+1} of a polygon is an *ear* if $v_{i-1}v_{i+1}$ is a diagonal; and v_i is the ear-tip of the triangle
- There are at most *n* ears
- (a convex polygon has exactly *n* ears)



Meister’s Two-Ear Theorem

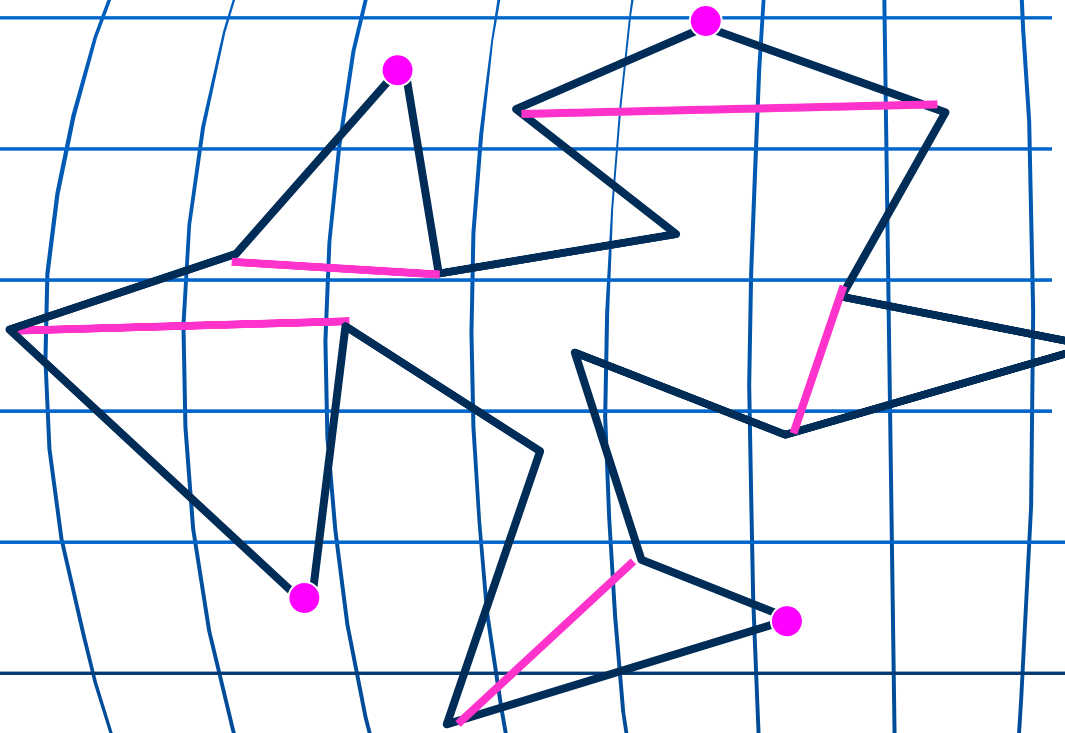
- Every polygon with n > 4 vertices has at least *two* non-overlapping ears

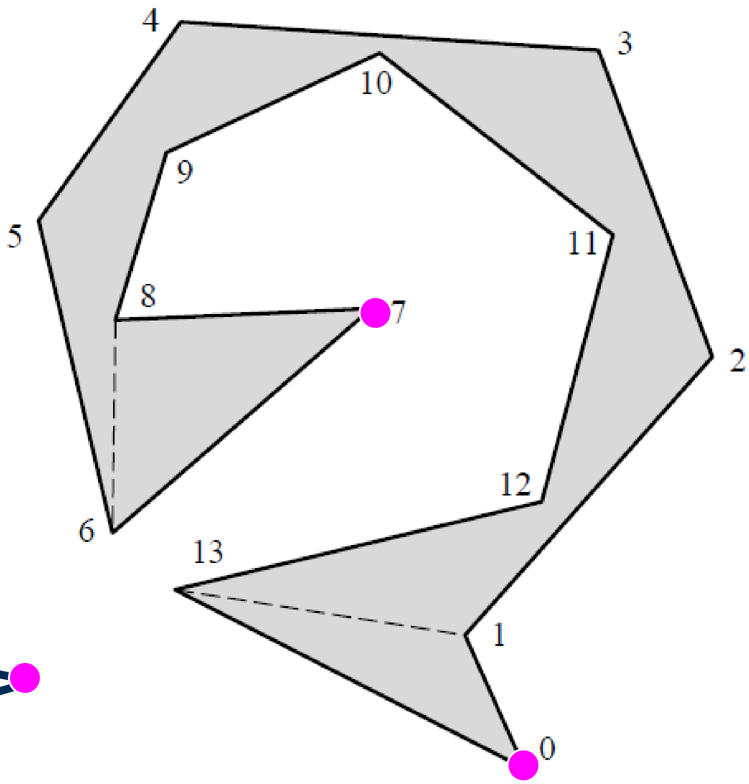


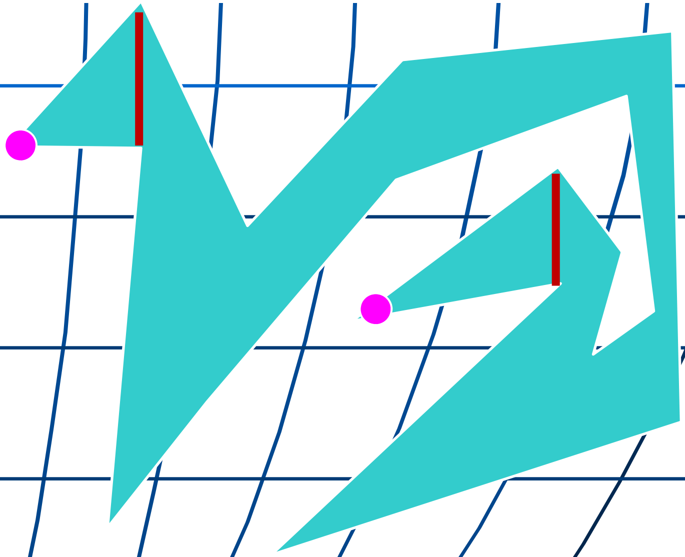
- The dual of a polygon is a tree
- A tree has at least two leaves
- The face (triangle) containing a leaf must be an ear

J. O'Rourke: *Computational Geometry in C*, Cambridge Univ. Press, 1998

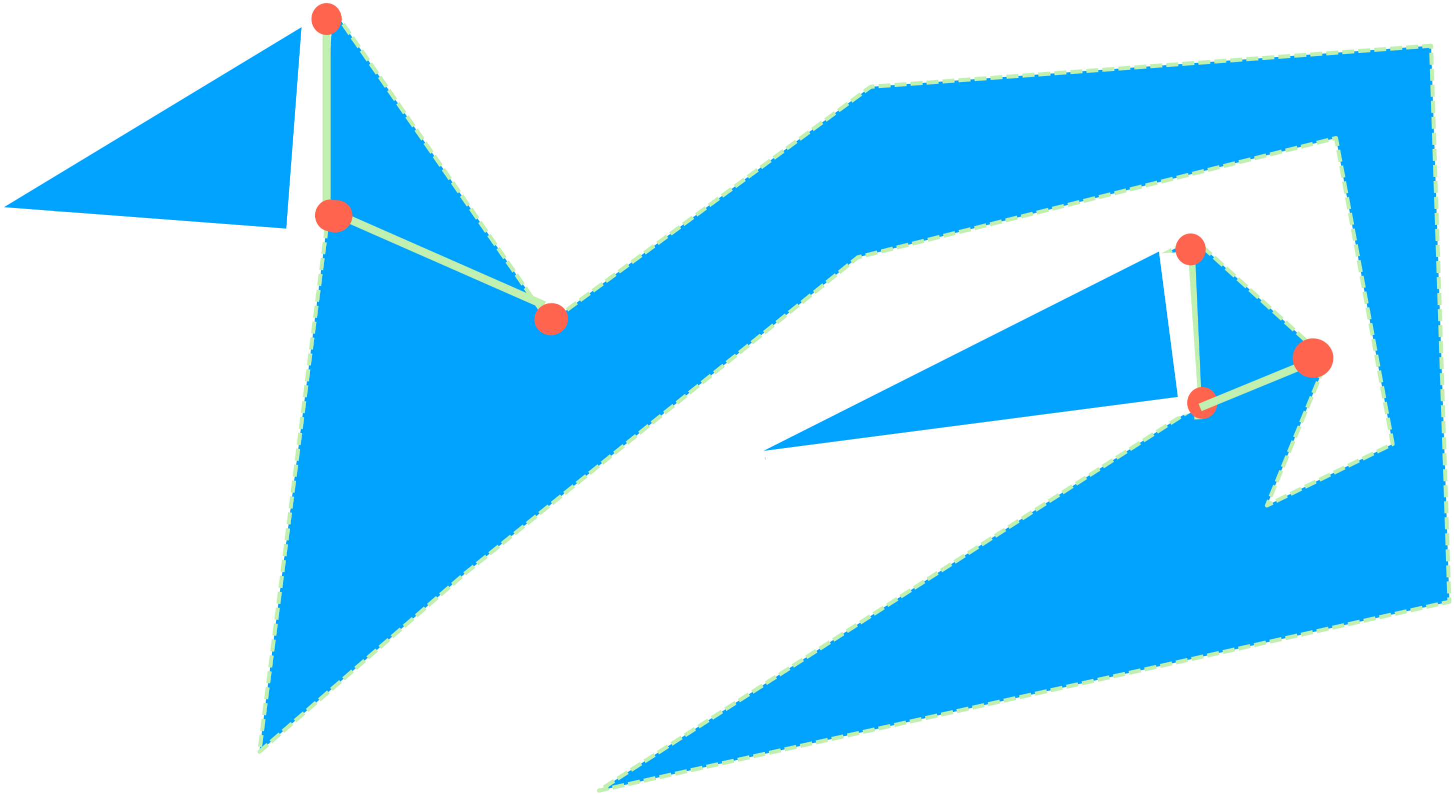
How many ears?







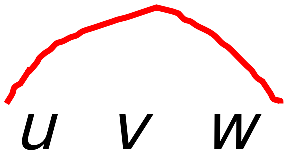
Triangulation: Ear-Clipping Algorithm



Every polygon has at least two ears!

Find an ear, fix a diagonal, chop the ear and iterate

Triangulation by ear-clipping

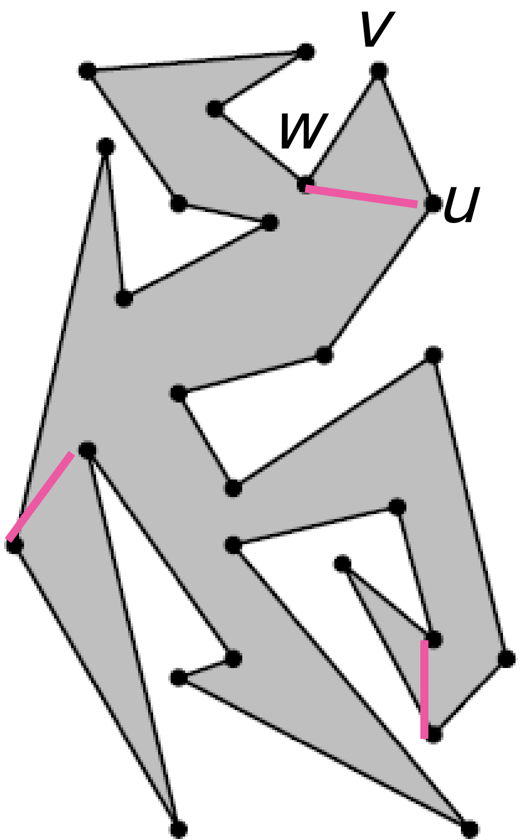


Using the two-ears theorem:
(an ear consists of three consecutive vertices u, v, w where $\overline{uw}$ is a diagonal)

Find an ear, cut it off with a diagonal, triangulate the rest iteratively

Question: Why does every simple polygon have an ear?

Question: How efficient is this algorithm?

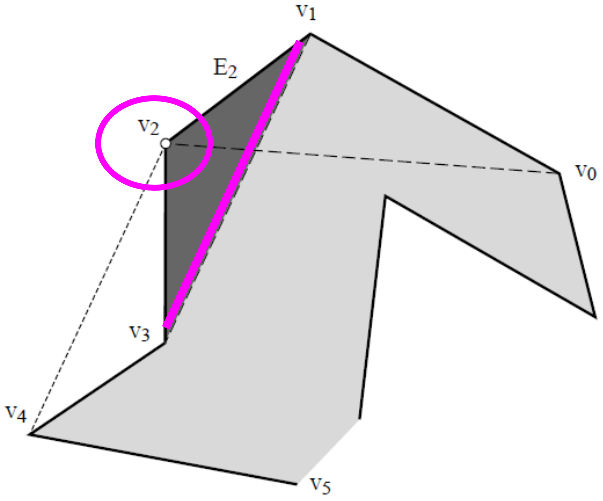


There are $O(n)$ ear candidates; checking each for ear $O(n)$; (n – 3) diagonals; Total time complexity $O(n^3)$

Can we improve it to $O(n^2)$

Smarter approach

- When clipping ear with tip v_i the only ear tip statuses that can change are at v_{i-1} and v_{i+1}



Both v_1 and v_2 are possible ear-tips; When v_2 is clipped and the diagonal is inserted, v_1 no longer remains an ear-tip. So, only the two neighbors of v_2 need to be checked for change of status. “Ear”-ities of other vertices are not affected. Thus, only $O(1)$ updates are needed after inserting one diagonal!

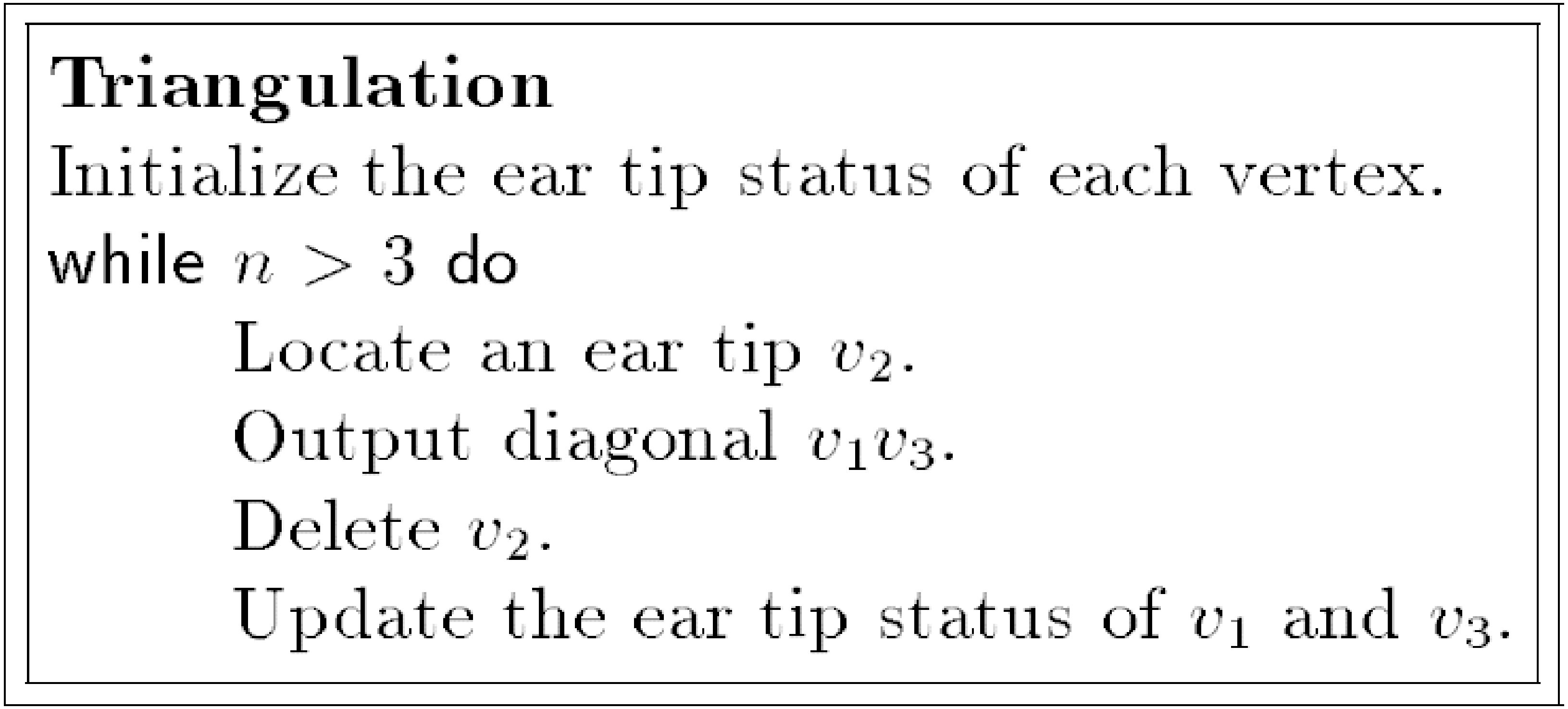
Naive: $O(n^3)$

There are $O(n)$ ear candidates; checking each for ear $O(n)$; Total $O(n^2)$

Update $O(1)$; There will be $O(n)$ iterations

Total: $O(n^2)$

Ear-clipping algorithm



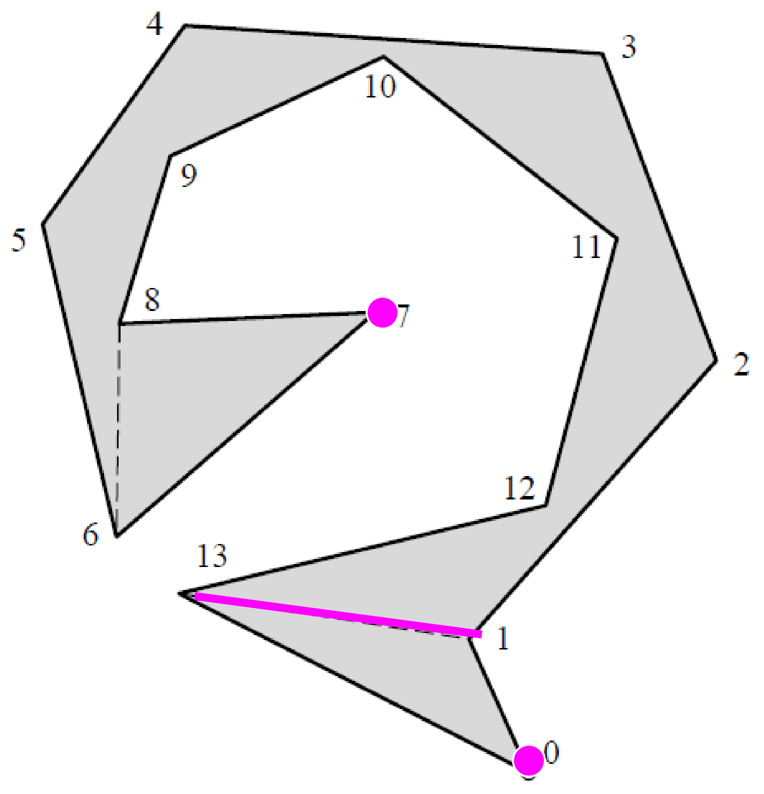
```
Triangulation
Initialize the ear tip status of each vertex.
while $n > 3$ do
    Locate an ear tip $v_2$.
    Output diagonal $v_1 v_3$.
    Delete $v_2$.
    Update the ear tip status of $v_1$ and $v_3$.
```

- Initially determine “ear-tip status” of each v_i, $O(n^2)$
- Update of each ear status requires $O(1)$; ear-tip tests @ $O(n)$ per test; n – 3 diagonals
- Total: $O(n^2)$

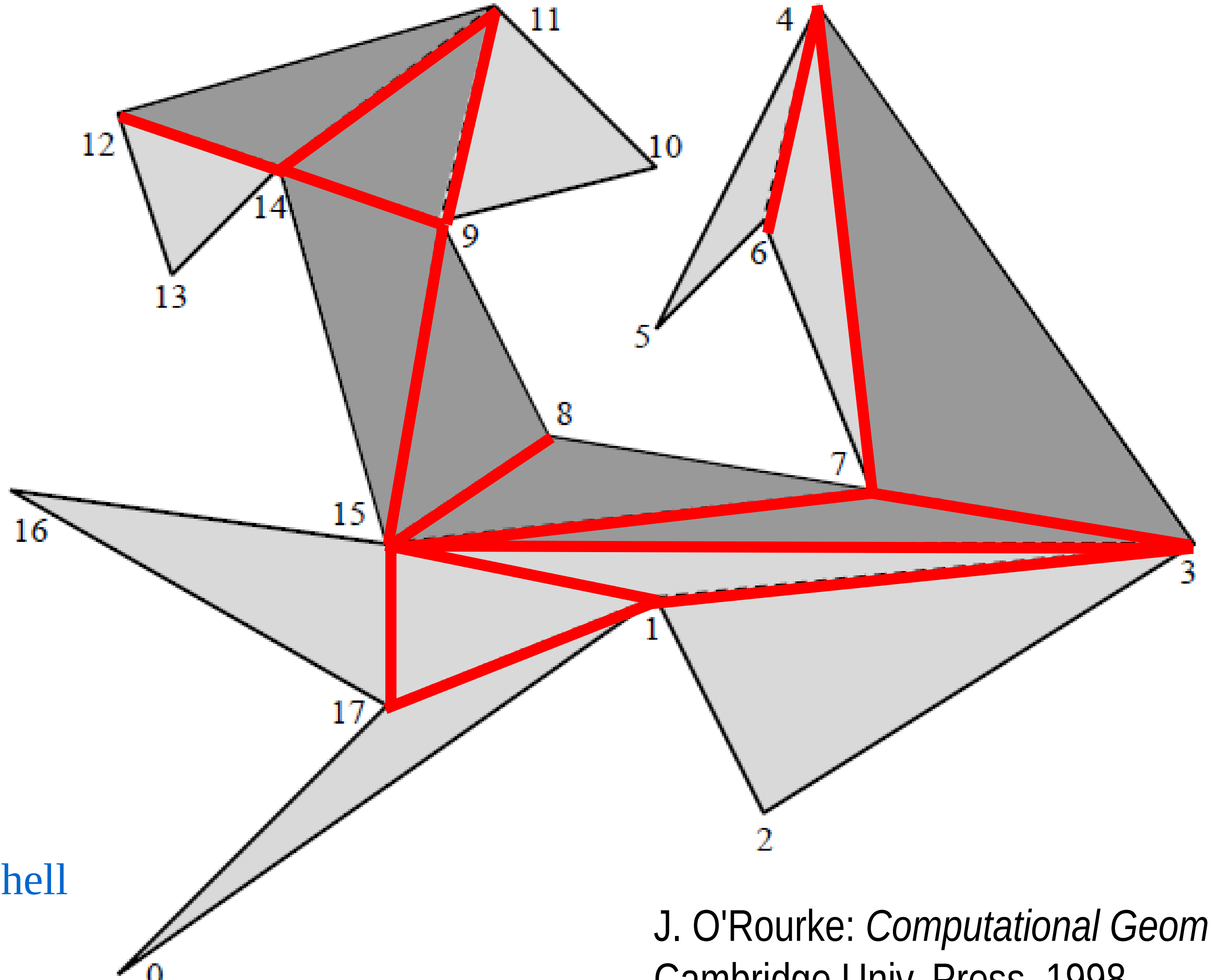
J. O'Rourke: *Computational Geometry in C*, Cambridge Univ. Press, 1998

After inserting one diagonal, the search for the next ear may take $O(n)$ time

Total: (n^2)



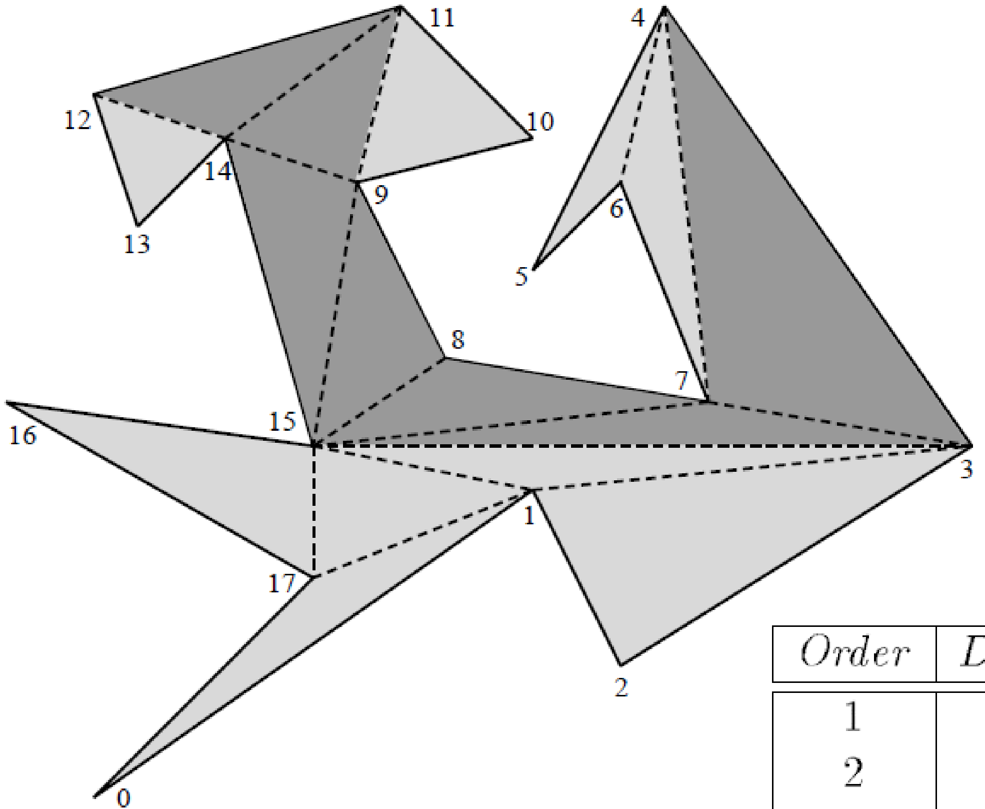
Example: Triangulate



Courtesy: Joe Mitchell

J. O'Rourke: *Computational Geometry in C*,
Cambridge Univ. Press, 1998

Example: Output



Order	*Diagonal indices*	*Order*	*Diagonal indices*
1	$(17, 1)$	10	$(3, 7)$
2	$(1, 3)$	11	$(11, 14)$
3	$(4, 6)$	12	$(15, 7)$
4	$(4, 7)$	13	$(15, 8)$
5	$(9, 11)$	14	$(15, 9)$
6	$(12, 14)$	15	$(9, 14)$
7	$(15, 17)$		
8	$(15, 1)$		
9	$(15, 3)$		

Courtesy: Joe Mitchell