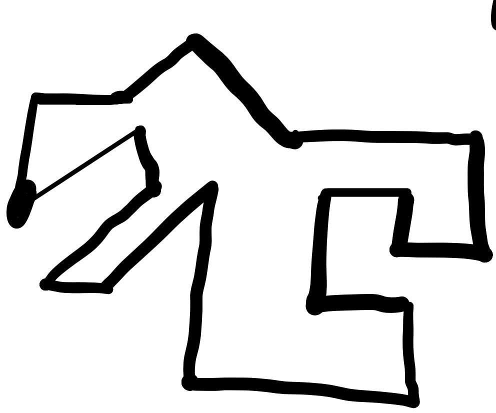
Lecture 4 - Polygon Triangulation

Motivation - Art Gallery Problem

Art Gallery



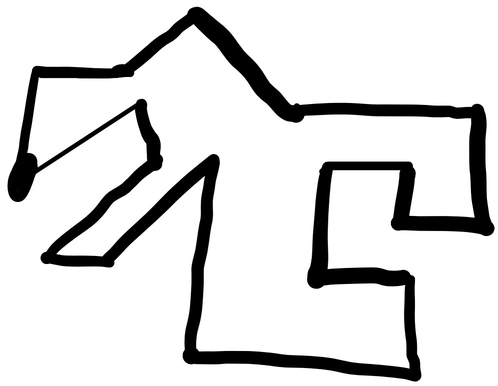
~ simple polygon

no holes

Lecture 4 - Polygon Triangulation

Motivation - Art Gallery Problem

Art Gallery



~ simple polygon
no holes

How many cameras does it take to guard the art gallery?

Lecture 4 - Polygon Triangulation

Motivation - Art Gallery Problem

Art Gallery

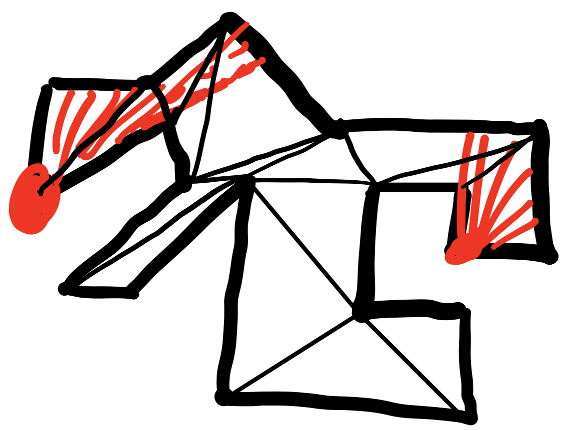
~ simple polygon
no holes

How many cameras does it take to guard the art gallery?

Lecture 4 - Polygon Triangulation

Motivation - Art Gallery Problem

Art Gallery



~ simple polygon

no holes

How many cameras does it take to guard the art gallery?

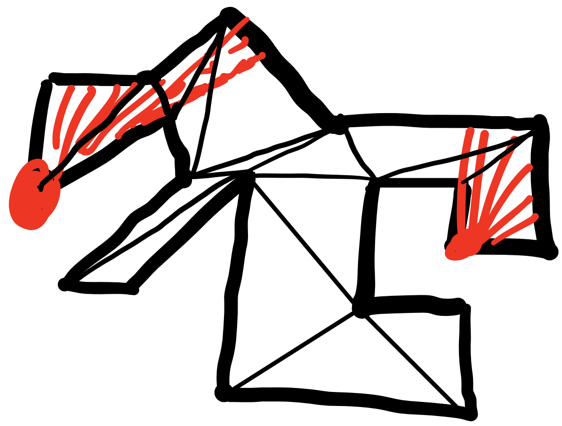
Upper bound: no of triangles in a triangulation of gallery:

division into triangles

Lecture 4 - Polygon Triangulation

Motivation - Art Gallery Problem

Art Gallery



~ simple polygon
no holes

How many cameras does it take to guard the art gallery?

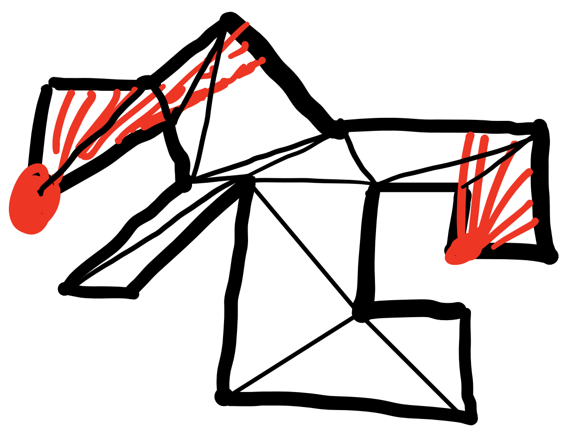
Upper bound: no of triangles in a triangulation of gallery:
division into triangles

Theorem: Any simple polygon with n vertices can be triangulated using n-2 triangles.

Lecture 4 - Polygon Triangulation

Motivation - Art Gallery Problem

Art Gallery



~ simple polygon
no holes

How many cameras does it take to guard the art gallery?

Upper bound: no of triangles in a triangulation of gallery:
division into triangles

Theorem: Any simple polygon with n vertices can be triangulated using n-2 triangles.

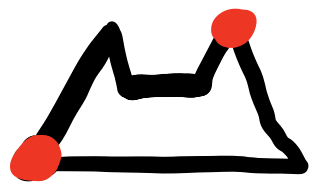
So an upper bound for the number of triangles is $n-2$.

Note

$\lfloor n/3 \rfloor$ cameras suffice

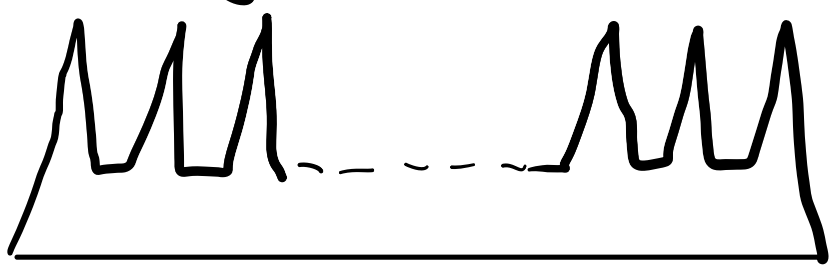
(proof using 3-colourings of graphs)

$\lfloor n/3 \rfloor$ sometimes needed



6 vertices
$\lfloor 6/3 \rfloor = 2$.

More generally



needs $\lfloor n/3 \rfloor$ cameras.

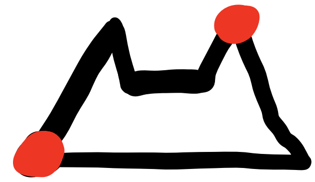
Today: <u>polygon triangulation algorithm</u>

Note

$\lfloor n/3 \rfloor$ cameras suffice

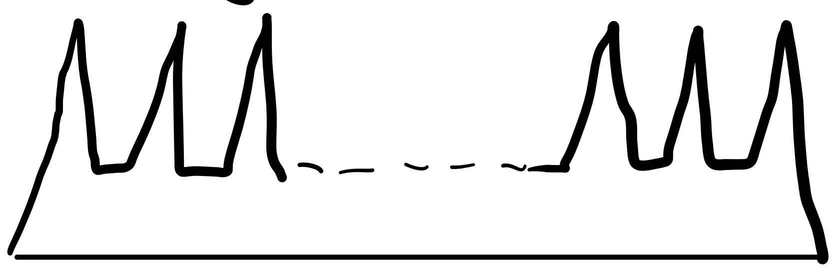
(proof using 3-colourings of graphs)

$\lfloor n/3 \rfloor$ sometimes needed



6 vertices
$\lfloor 6/3 \rfloor = 2$.

More generally



needs $\lfloor n/3 \rfloor$ cameras.

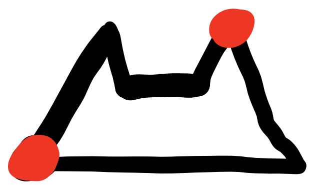
Today: polygon triangulation algorithm

Note

$\lfloor n/3 \rfloor$ cameras suffice

(proof using 3-colourings of graphs)

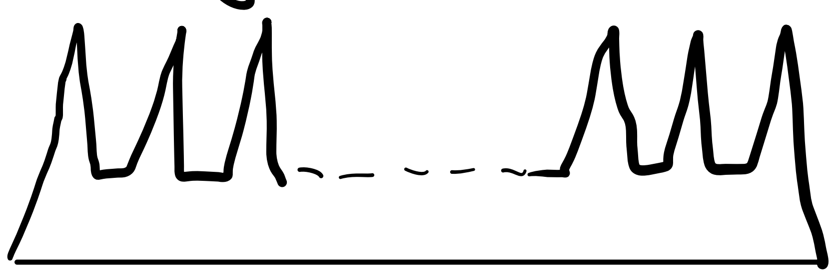
$\lfloor n/3 \rfloor$ sometimes needed



6 vertices

$\lfloor 6/3 \rfloor = 2$.

More generally



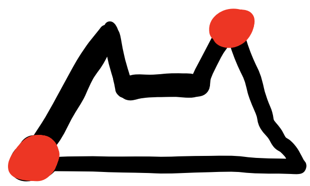
needs $\lfloor n/3 \rfloor$ cameras.

Note

$\lfloor n/3 \rfloor$ cameras suffice

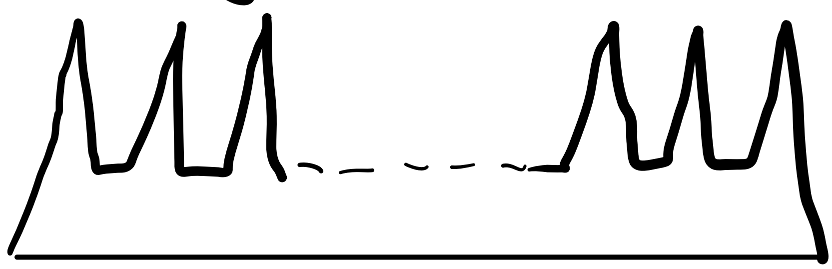
(proof using 3-colourings of graphs)

$\lfloor n/3 \rfloor$ sometimes needed



6 vertices
$\lfloor 6/3 \rfloor = 2$.

More generally

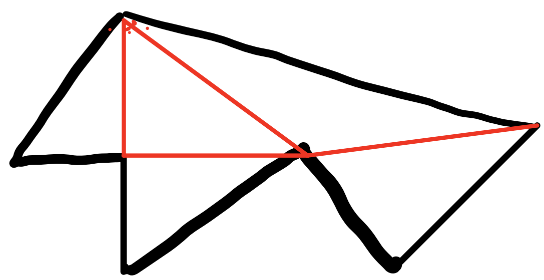


needs $\lfloor n/3 \rfloor$ cameras.

Today: polygon triangulation algorithm

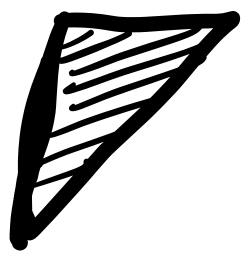
Theorem

Any simple polygon can be triangulated: if it has n vertices, we require n-2 triangles.

E.g. 

7 vertices
5 triangles

Proof

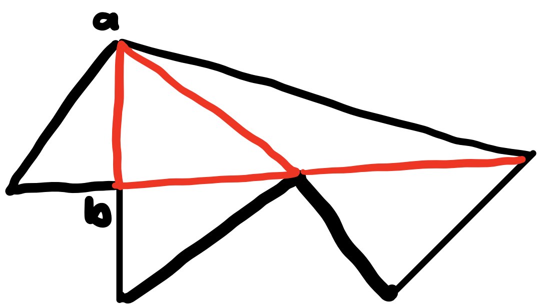
- $n=3$  is obvious.

A diagonal is a straight line segment $\vec{ab}$ whose endpoints are vertices & which otherwise belongs to interior of polygon.

Theorem

Any simple polygon can be triangulated: if it has n vertices, we require $n-2$ triangles.

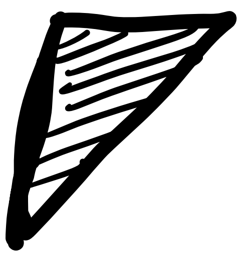
E.g.



7 vertices
5 triangles

Proof

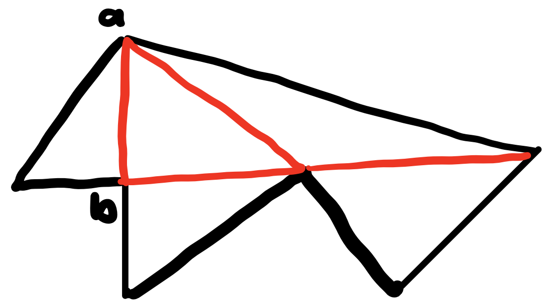
- $n=3$



is obvious.

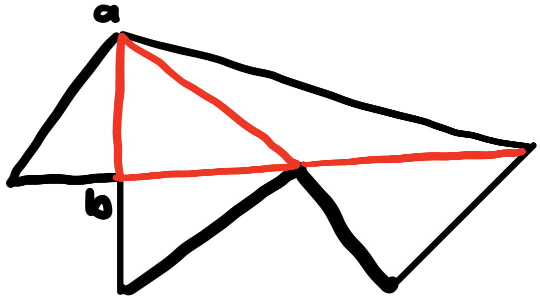
A diagonal is a straight line segment $\vec{ab}$ whose endpoints are vertices & which otherwise belongs to interior of polygon.

- Eg. $\vec{ab}$ above.



- A diagonal $\vec{ab}$ splits polygon into 2 parts $P = A \cup_{\vec{ab}} B$ with m, k vertices where $m+k=n+2$ & $m, k < n$.

- Triangulations of A&B can be combined - so result on existence of a Triangulation follows by induction on number of vertices if we can prove that a diagonal always exists.



- $P = A \underset{\overrightarrow{ab}}{\cup} B$ with m,k vertices where $m+k=n+2$ & $m,k<n$.

- Also formula for no. of triangles follows from existence of diagonal.

$$\underbrace{P}_{n \text{ verts}} = \underbrace{A}_{m} \underset{\overrightarrow{ab}}{\cup} \underbrace{B}_{k}$$

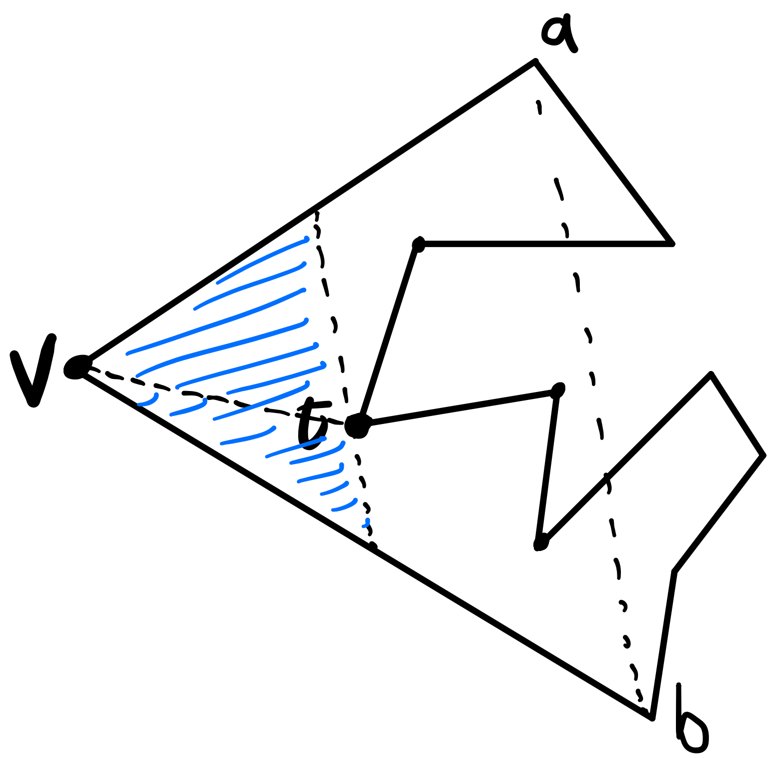
by ind. we have triang. of
A in $m-2$ triangles
B ... $k-2$ triangles

So P has triang. in

$$(m-2)+(k-2) \text{ triangles}$$
$$= m+k-4 = n-2 \text{ triangles.}$$

- So remains to prove existence of diagonal.

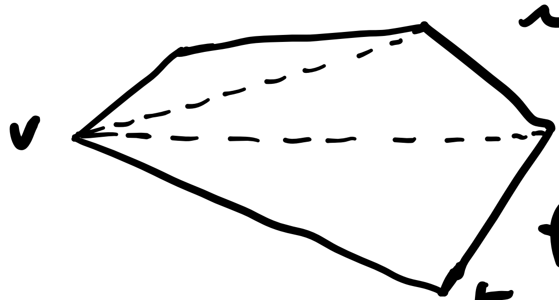
Example)



- let v be lexicographically smallest vertex on P (x coord, then y coord).
- Let a, b be vertices connected to v.
- If $\overrightarrow{ab}$ is a diagonal, job done!
- Else, an edge of P must have endpoint inside Δ_{vab} so $\exists$ vertices of P inside Δ_{vab}.
- Let t be furthest such from $\overrightarrow{ab}$.
- If $\overrightarrow{vt}$ did not lie in interior, as before, an endpoint of an edge would lie in blue region - contradiction.
- So $\overrightarrow{vt}$ is a diagonal. $\square$

Goal: Find algorithm with complexity $O(n \log n)$.

Convex polygon:



~ easy to triangulate: draw line from vertex v to all others.

Less restrictive notion (still easily triangulable) of monotone polygon:

monotone with respect to axis y:

any horizontal line $y=k$ intersects polygon in line segments, point or empty set.

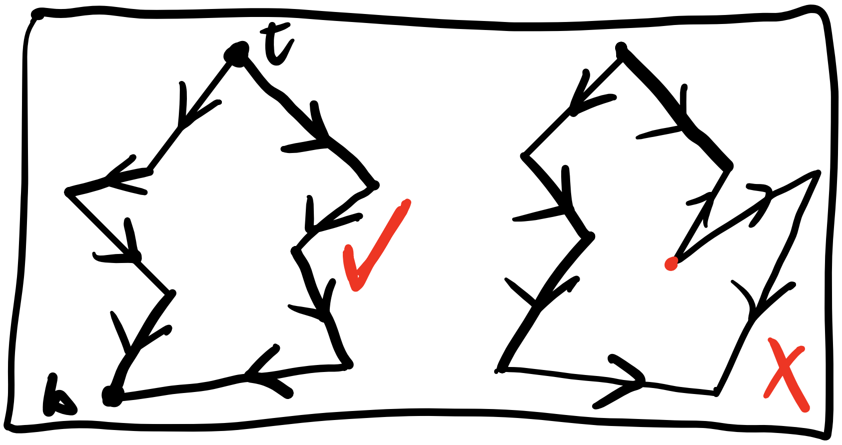
Today, we work with slightly stronger notion of monotone polygon

Monotone polygon:

Consider lex ordering:

$a > b \Leftrightarrow a_y > b_y$ or $(a_y = b_y \ \& \ a_x < b_x)$

- Determines two paths from top t to bottom b.
- Polygon P is monotone if both paths are decreasing (with respect to lex. ordering)



Algorithm:

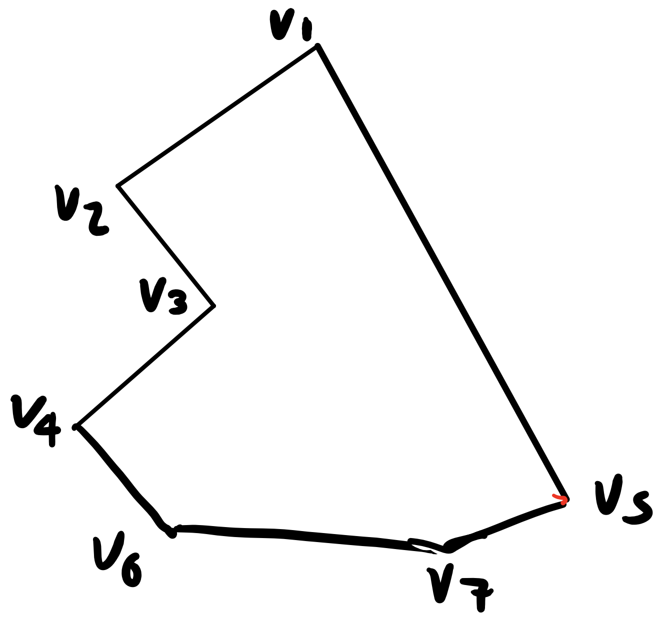
① Divide simple polygon into monotone pieces.

② Triangulate monotone polygons.

This week, we do 2.

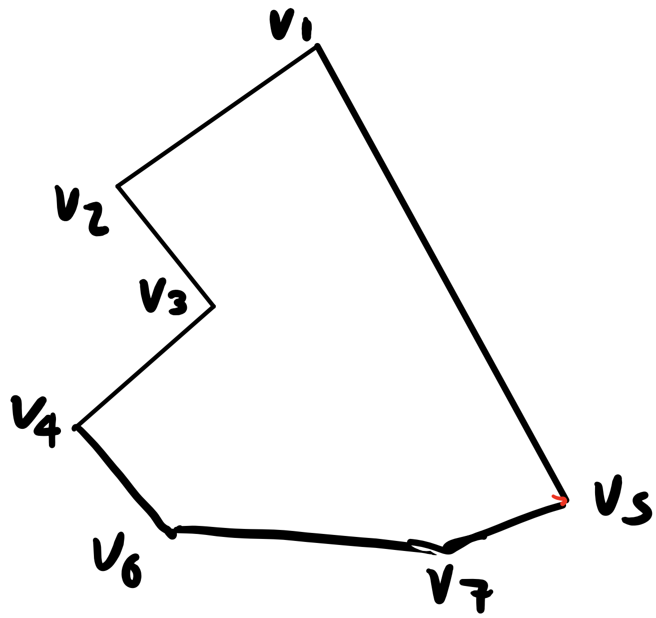
Next week, do 1.

Triangulate monotone polygon



Idea: draw diagonals to all possible preceding vertices (wrt lex ordering) where diagonals lie in polygon.

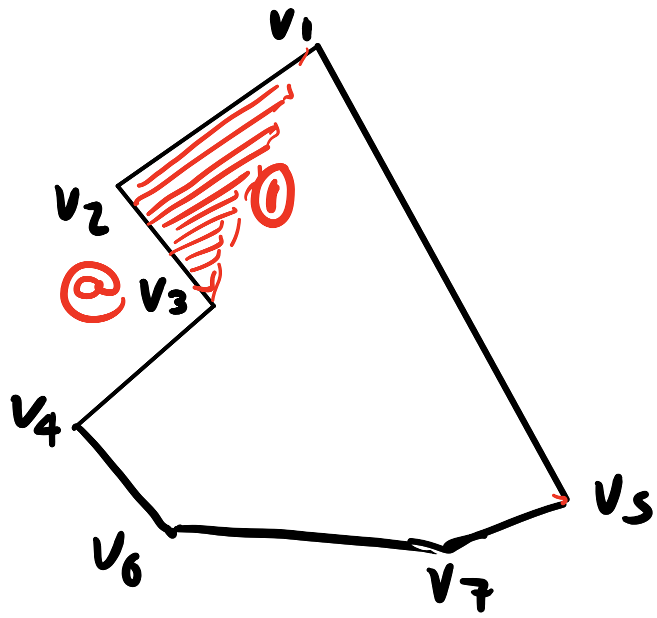
Triangulate monotone polygon



Idea: draw diagonals to all possible preceding vertices (wrt lex ordering) where diagonals lie in polygon.

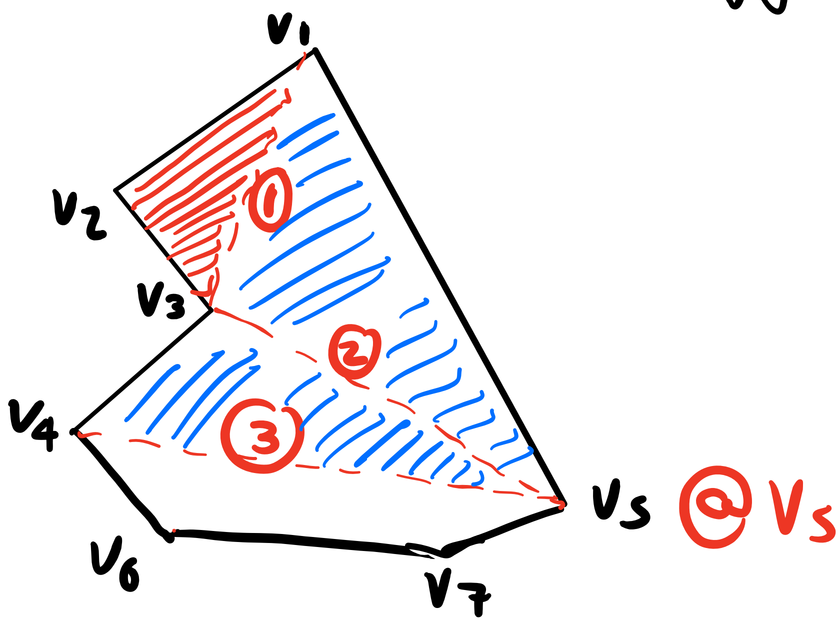
Break off triangles.

Triangulate monotone polygon



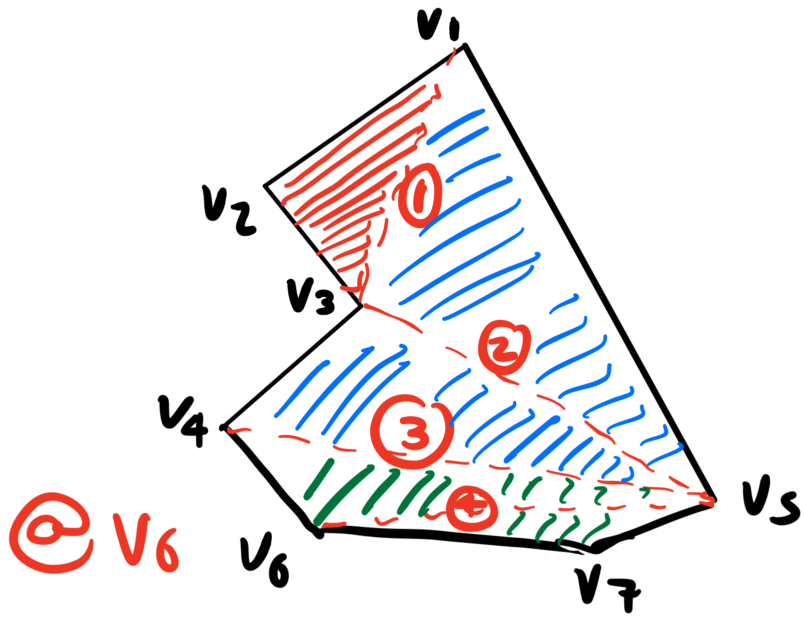
Idea: draw diagonals to all possible preceding vertices (wrt lex ordering) where diagonals lie in polygon, & break off triangles.

Triangulate monotone polygon



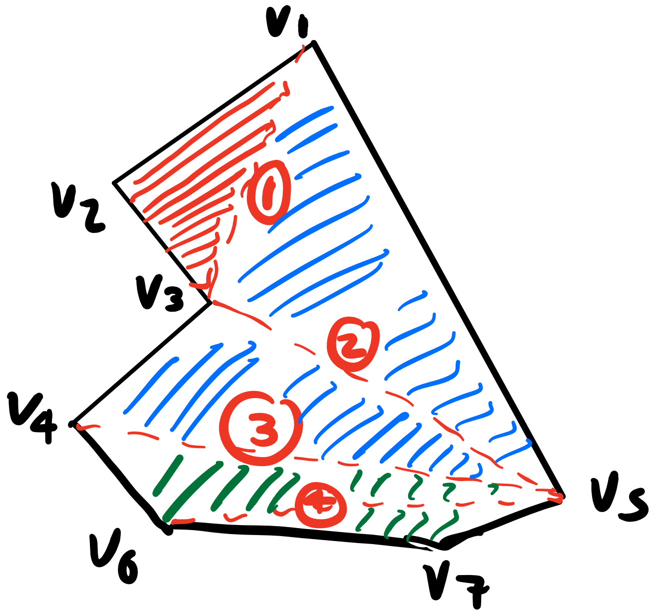
Idea: draw diagonals to all possible preceding vertices (wrt lex ordering) where diagonals lie in polygon, & break off triangles.

Triangulate monotone polygon



Idea: draw diagonals to all possible preceding vertices (wrt lex ordering) where diagonals lie in polygon, & break off triangles.

Triangulate monotone polygon



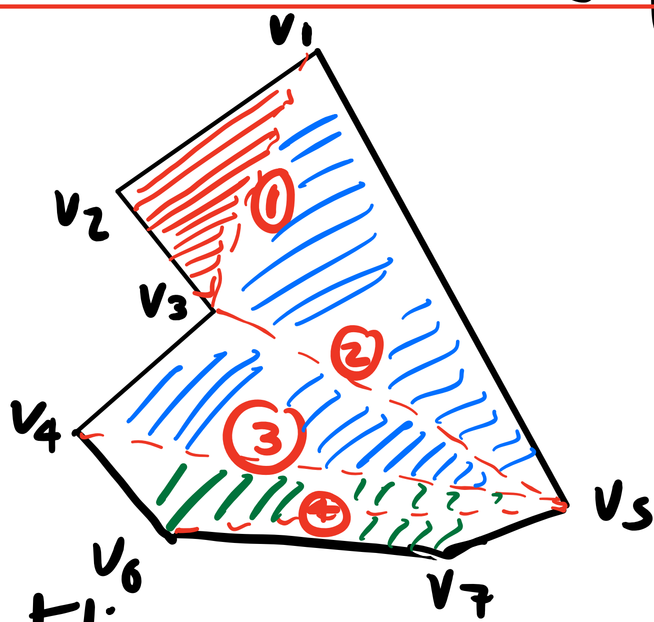
Formally : store monotone polygon in DCEL D.

Output : DCEL with diagonals added, capturing triangulated polygon.

- Another structure involved will be a stack S which, on reaching a vertex v, will contain those vertices above v which we have not yet broken off.
- These will always live on 1 path - left or right - of the leftover polygon.

- At v_5, S will contain (v_4, v_3, v_1)
- At v_6, - - - - (v_5, v_4)

Triangulate monotone polygon



Algorithm outline

- calculate left & right paths from top vertex to bottom.
- Merge two paths into a lex. ordered list $u_1, \ldots, u_n$.
- Initialise empty stack S.
  Push u_1 & u_2 onto it ~ so $S = (u_2, u_1)$
- For $j = 3, \ldots, n-1$,
  handlevertex(v_j):

- involves drawing diagonals from v to elements on stack, where possible, & updating stack.

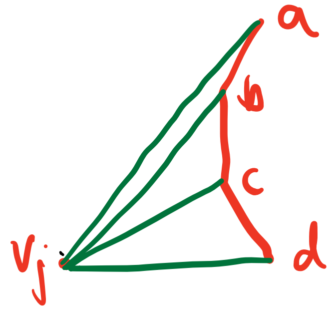
- At v_n, draw diagonals to all but first & last members of S.

Handlevertex(v_j):

For $j=3$ to $n-1$:

- if v_j & vertex on top of S are on diff. paths:
  - pop all vertices from S
  - add a diagonal (in D) to each popped vertex except the last one.
  - push v_{j-1} & v_j onto S.

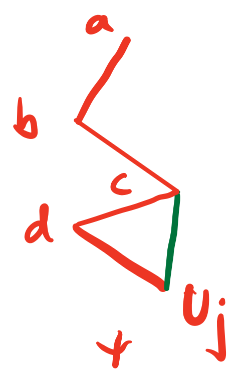
$S=(d,c,b,a)$



$+\ S=(v_j, d)$ where $d = v_{j-1}$

- Otherwise,
  - pop top vertex of S.
  - pop remaining vertices as long as diagonals from v_j to them lie inside P.
  - add the diagonals to D.
  - Push last popped vertex back onto S. Push v_j onto S.

$S=(d,c,b,a)$



$+$

$S=(v_j,c,b,a)$

- At v_n, add diagonals to all vertices in S except the first and last ones.

Complexity :

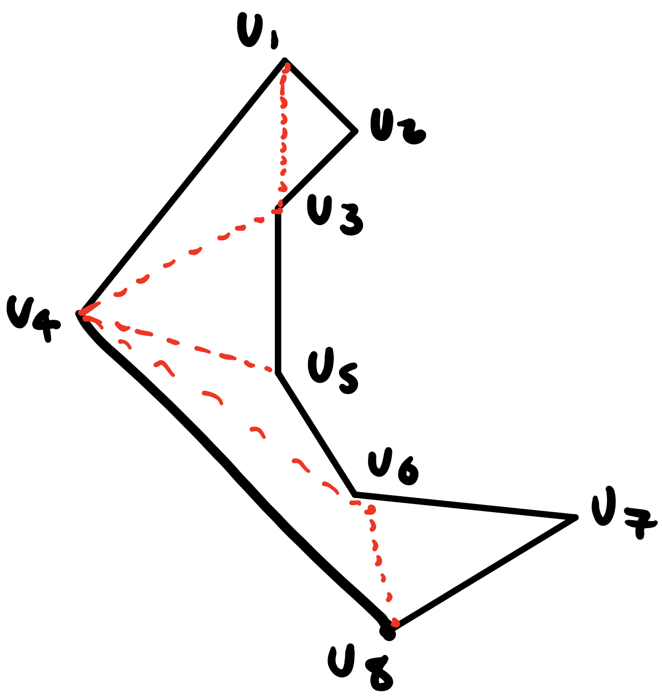
- calculate top, bottom $O(n)$
  - calc. paths top to bottom $O(n)$
    (use next)
- Merge two paths takes time $O(n)$.
- During running of loop, each vertex is added / removed at most 2 times
  - so $O(n)$.
- Total : $O(n)$.

Next week,

① Break simple polygon into monotone parts.

Combining with above, obtain alg. for triangulating a simple polygon.

Exerises in class



- $S = (v_2, v_1)$
- At v_3, pop v_2 so $S = (v_1)$

- pop v_1 & add v_1v_3 to D.
- Push v_1 & v_3 onto S so $S = (v_3, v_1)$.

• At v_4, pop all vertices from S. Add diag v_3v_4 to D. $S = (v_4, v_3)$.

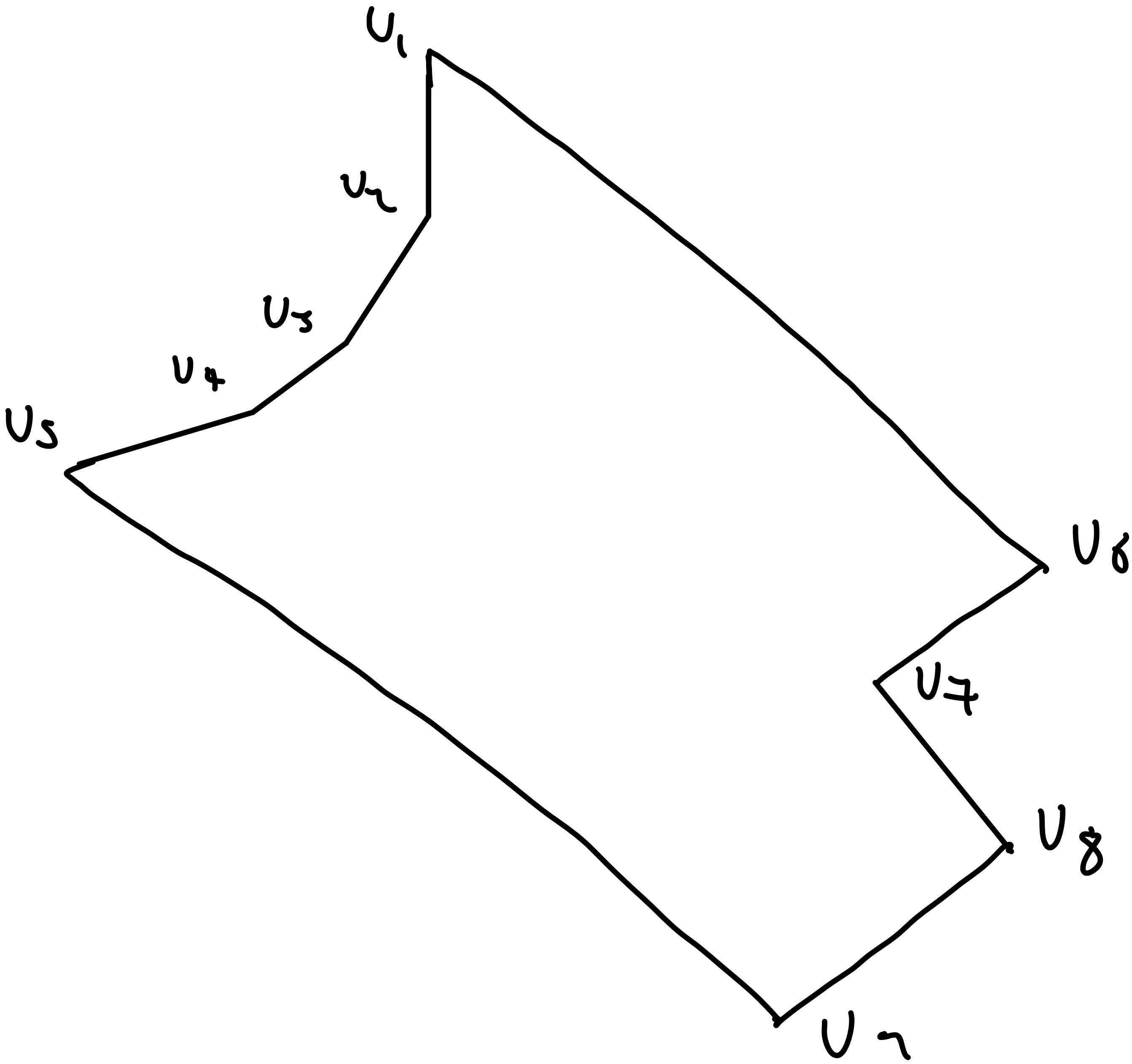
• At v_5, add diag v_4v_5 to D. $S = (v_5, v_4)$

• At v_6, pop v_5 $\mapsto$ $S = (v_4)$. Add v_4v_6 to D.

Pop, push $\to$ Then $S = (v_6, v_4)$.

• At v_7, pop v_6 so $S = (v_4)$. Diag. $v_4v_7 \notin P$. Push $\to$ (v_7, v_6, v_4)

• At v_8, add diag. v_8v_6 to D.


v1
v2
v3
v4
v5
v6
v7
v8
v9