polygons:

visibility in a cone
- enter on right, going →
- while angle goes →
- push edges on stack
  \[ e_1, e_2, e_3 \]

if this continues until \( e_k \) great.
Non-trivial case: somewhere angle backtracks.
Suppose "upward".

starts getting complicated
Non-trivial case: somewhere angle backtracks.

\[\text{Suppose "upward"}\]

Ignore until path reappears at same angle.

\[e_1 \quad e_2 \quad e_3 \quad e_m\]

partial edge
So far we can handle any upward backtrack.

Eventually the chain shows up again.

---

\[
e_1, e_2, w_1, e_x, w_2, e_y, e_{y+1}, w_3, e_z
\]

\[\uparrow\text{implicit}\]
So far we can handle any upward backtrack.

If downward backtrack...

\[ e_1, e_2, \omega_1, e_x, \omega_2, e_y, e_{y+1}, \omega_3, e_z \]
So far we can handle any upward backtrack.

If downward backtrack...

Just start popping from stack
So far we can handle any upward backtrack.

If downward backtrack...

Just start popping from stack
but check if you cross windows

→ enter upward backtrack mode
So far we can handle any upward backtrack.

If downward backtrack...

Just start popping from stack but check if you cross windows

→ enter upward backtrack mode
Why is $e_y$ not in stack?
Why is $e_y$ not in stack?

$\Rightarrow$ it will get covered eventually.

Notice all edges in stack point.
How can we continue?

1) Forward: push
2) Backtrack: pop
3) Invisible: ignore
How can we continue?

1) Forward: push
2) Backtrack: pop
3) Invisible: ignore

until emerge from window

if you go inside you will create a new sub-window and you will backtrack out anyway
every step is local
(Perhaps via sequence
of local pops)

all you need to check:
- Am I covering an edge when backtracking?
- Am I going through a window?
• Moving forward: easy
• Backtrack up: wait til you return, to move forward
• Backtrack down: pop while backtracking unless you're in your own pocket

$\Rightarrow$ could resume forward motion
$\Rightarrow$ could jump to "backtrack up"

(Going directly from $\bullet$ to $\bullet$ would have been the same)
Compute kernel of star
COMPUTE KERNEL OF STAR

every edge has a good side
COMPUTE KERNEL OF STAR
start with one reflex vertex

current kernel

out in
suppose you find 3 extensions that form a triangle
possible position of 4th line

conclude: not a star
To keep continuing means that we keep shrinking a convex polygon.
Compute intersection of convex n-gon & half-plane

- brute-force: $O(n)$
- binary search: $O(\log n)$

So the kernel of a polygon can be computed in $O(n + r \cdot \log r)$ time...

... or we conclude the polygon is not a star.

$r = \#\text{reflex vertices}$
we assumed that we started w/ a triangle

works just as well with unbounded convex regions
TRIANGULATION OF A STAR USING THE KERNEL
start with this fake triangulation
FLIP: still get a star
removed one ear
convex quad
Do this as long as you find convex quadrilaterals

this means you are looking for convex vertices only

flip
You will find another convex vertex not containing \( \bullet \) unless \( \geq 2 \)-ear thm.
The star has 2 ears. Find the one not containing • & cut

TIME?
The star has 2 ears. Find the one not containing • & cut

TIME?

• Looking for a convex quad
  ➔ scan : O(n)
The star has 2 ears. Find the one not containing • & cut

TIME?

- Looking for a convex quad
  \( \leftrightarrow \) scan: \( O(n) \)

In fact you get a list of ALL convex quads in \( O(n) \)
The star has 2 ears. Find the one not containing • & cut

TIME?
• Looking for a convex quad
  \( \rightarrow \text{scan} : O(n) \)
  In fact you get a list of ALL convex quads in \( O(n) \)
• When you flip, you only change things locally
  So you only update the list locally & can keep flipping in \( O(n) \) per flip.
Both failed.
They are convex but don't make a CQ with •.
Triangulation of a star w/o knowing kernel
Pick any vertex: 
Compute vis.polygon of o
SPECIAL WINDOW REGIONS

Geodesic path $x \rightarrow y$

sweep until you hit a vertex.

Repeat recursively

until you hit $\nu$
How do you compute one geodesic (in this setting)?
How do you compute one geodesic (in this setting)?

1) extend $\overrightarrow{v}$ & keep only parts of the pocket above

2) MELKMAN on modified pocket

$\text{time} = O(1_{\text{pocket}})$

total for all pockets: $O(n)$
How do you compute one geodesic (in this setting)?

Better: compute visibility region for \( \bullet \)

Then run Melkman.

1) extend \( \overrightarrow{ve} \)

& keep only parts of the pocket above \( O(n) \) is a little messy if you think about it
So far we have the extended vis. pol of $o$.

NEXT:

If $o$ sees $y$ then form diagonal $\overline{vy}$.
Then, form diagonal \( vu \).
Sub-polygon of still a vis.pol of star
- Sub-polygon of still a vis. pol. of star
- "fan" polygons visible from a vertex
- WEV: visible from an edge
Sub-polygon of still a vis.pol of \( \star \)

"Fan" polygons visible from a vertex

WEV: visible from an edge

good sub-polygons visible from \( \star \)

edge-visible from "lid"

every sub-polygon can be triangulated in linear time w/o actually using \( \bullet \) DONE