simple polygon: region enclosed by a simple closed polygon chain that does not intersect itself.

Camera position: a point in the polygon

Camera line of sight: camera sees all point in the polygon that it can connect to with an open segment

finding the minimum number of cameras is NP-hard.

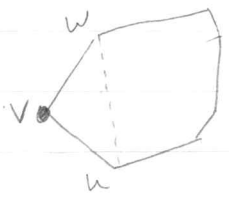
Note. A convex polygon can be guarded by one camera.

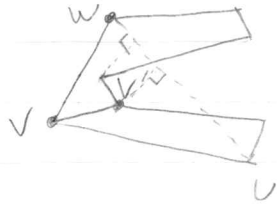
Thm 3.1 Every simple polygon admits a triangulation, and any triangulation of a simple polygon with n vertices consists of exactly n-2 triangles.

Proof. Induction

$n=3$, the polygon itself is a triangle.

Let $n>3$, assume the theorem is true for all $m<n$.





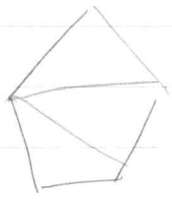
choose leftmost vertex v, u and w the neighboring vertices of v. If $\overline{wu}$ is interior, done. Else choose v' as the farthest from $\overline{wu}$.

3-coloring with the vertices of the triangle and choose one color to the camera positions! We only need $\lfloor n/3 \rfloor$ camera

Is this the best? Yes!

※ triangulation

easy for convex polygons.



① devide them into y-monotone polygons then

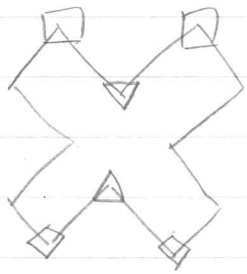
② triangulate them.

monotone to a line l: for any line l' perpendicular to l the intersection of l' and the polygon is connected

coord sys: points top to down relation: ↓ (priority) → (if ↓ is same)

①. Search for turn vertices and split from those points.

turn vertices: the points that has the same direction (upor down to neighbor vertices.



□ : start vertex rest is regular vertex

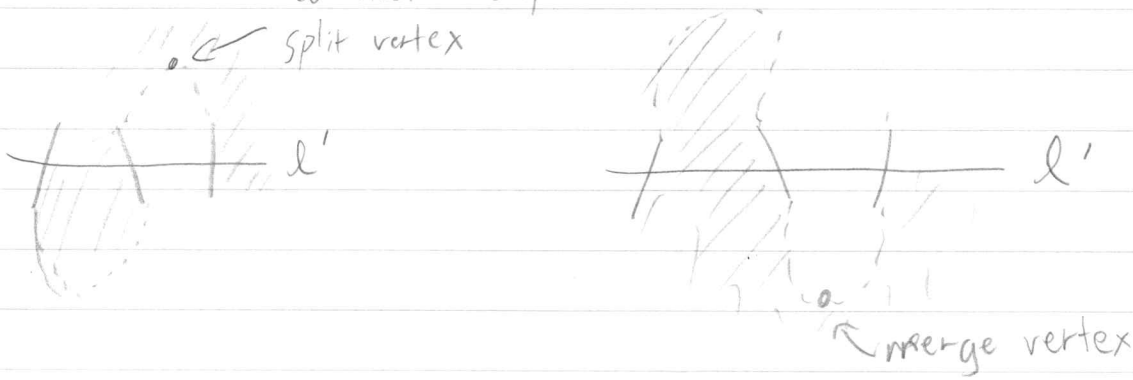
▽ : merge vertex

△ : split vertex

◇ : end vertex

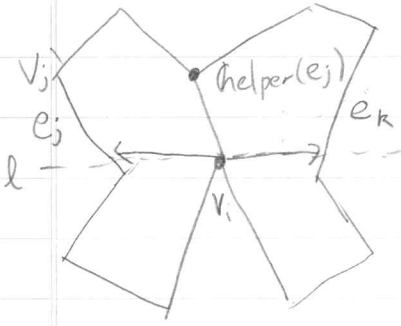
Lem 3.4: A polygon is y-monotone if it has no split vertices or merge vertices.

proof. If not monotone there is a line l' that intersect P with more than one connected component.

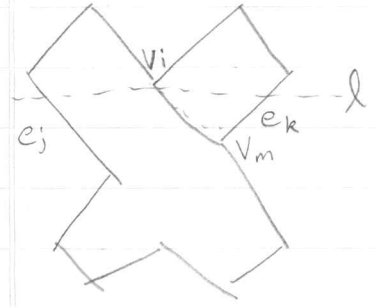


① continue

We sweep from top to bottom looking at event points
Event points are the n vertices.
For each split and merge vertices, we need to find a diagonal that split the polygon and remove the split or merge vertex.

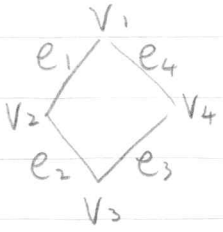


• split node
the lowest point between e_j and e_k
call it helper(e_j)



• merge node
make the v_i the helper node of e_j.
It will be later connected by v_m.
v_m: the highest point after v_i.

To maintain the edges in a dynamic binary search tree T.
with each edge we store the helper of that edge.

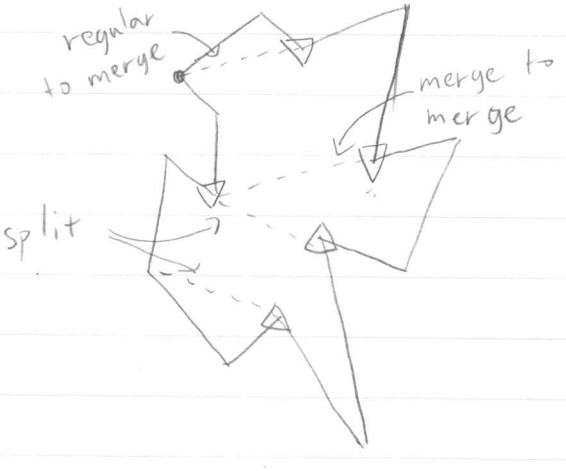


Algorithm
Going from top vertex to the last vertex (current vertex is v_i)
if {start vertex: add e_i to T and set helper(e_i) = v_i
if end vertex { if helper(e_{i-1}) is mergevertex { add diagonal between v_i and helper(e_{i-1})} Delete e_{i-1} from T}

if Split vertex:
get e_j left of v_i, and insert a diagonal from v_i to helper(e
set helper(e_j) to v_i
Insert e_i in T and set helper(e_j) to v_i

if Merge vertex:
if helper(e_{i-1}) is merge:
Insert the diagonal from v_i to helper(e_{i-1})
Delete e_{i-1}
get e_j left of v_i
if helper(e_j) is merge
Insert the diagonal from v_i to helper(e_j)
sethelper(e_j) to v_i

if Regular vertex:
if the interior is to the right of v_i:
if helper(e_{i-1}) is merge
Insert diagonal from v_i to helper(e_{i-1})
Delete e_{i-1}
Insert e_i and set helper(e_i) to v_i
else:
get e_j left of v_i
if helper(e_j) is merge
Insert diagonal from v_i to helper(e_j)
set helper(e_j) to v_i

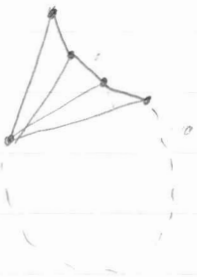


for each vertex event.
at most one query, one insertion,
one deletion on T
and at most two diagonal to D
$O(3\log n) \times n = O(n \log n)$

Split monotone polygons to triangles.

Consider the left and right bound of the monotone polygon.

From top to bottom



if the dot is the opposite chain, we can do the trianglazation by connecting all of them

else

if the interior angle is less than π we can trianglazation, otherwise, just add to stack and maintain

or

at most two push to the stack for each vertex, pops $\leq$ pushes

$O(2(n-3)+2) = O(n)$

A strictly y-monotone polygone can be triangltzed in linear time