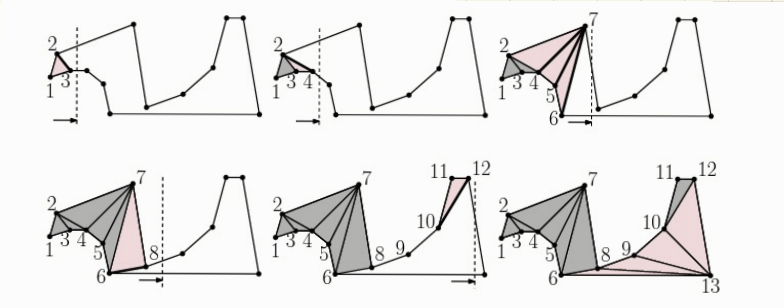
Polygon Δtion Pt 2

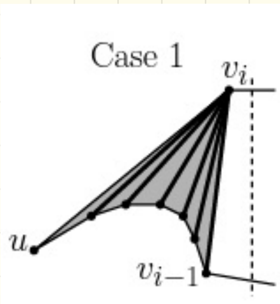
Δlate x-monotone polygon



Key idea:

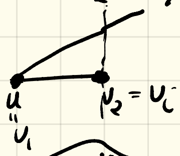
1. untriangulated region to left of the sweepline has "nice" structure allowing for fast Δtion
2. Polygon has n-3 diagonals ⇒ O(n) time

Lemma: (*Main Invariant*) For $i \geq 2$, let v_i be the vertex just processed by the triangulation algorithm. The untriangulated region lying to the left of v_i consists of two x-monotone chains, a lower chain and an upper chain each containing at least one edge. If the chain from v_i to u has two or more edges, then these edges form a reflex chain (that is, a sequence of vertices with interior angles all at least 180 degrees). The other chain consists of a single edge whose left endpoint is u and whose right endpoint lies to the right of v_i (see Fig. 32(a)).

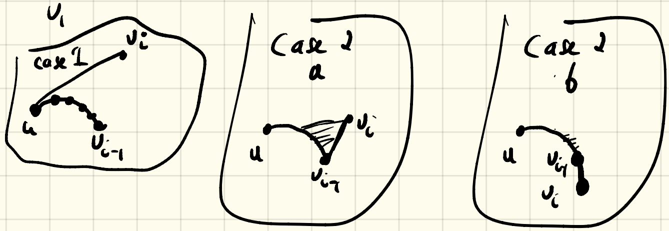


Pf (Main invariant) by induction

base case consider v_2 w/ $u = v_1$

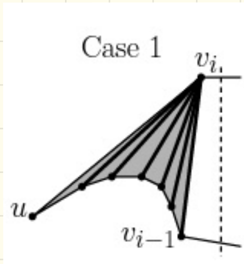


Induction step:



Case 1: v_i is on the opposite chain of v_{i-1}

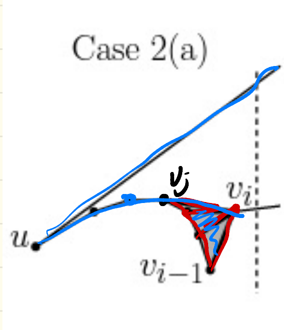
- add diagonals from v_i to $v_{i-1}, \dots,$ vertex before u



u is visible to v_i by assumption
lower chain is x-monotone
reflex and to the left of v_i
$\Rightarrow$ any vertex "after" v_i cannot sneak back to block visibility

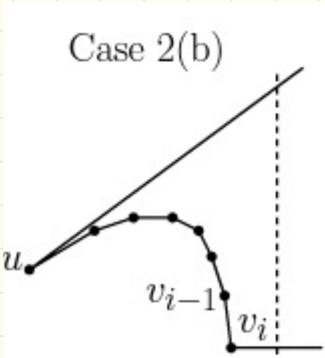
- set u to be v_{i-1} (restore invariant)
reflex chain is now $v_{i-1}\, v_i$

(2a) v_i is on the same chain as v_{i-1} and is non-reflex



- walk backwards on the reflex chain to the verts visible to v_i
- add diags from v_i to $v_{i-1}, \dots, v_j$
- remove $v_{i-1}, \dots, v_{j+1}$ from the reflex chain
- add v_i to the reflex chain (restored invariant)

(2b) v_i is on same chain as v_{i-1} and is reflex
- add v_i to reflex chain (restore invariant)



How to implement
- store the reflex chain on a stack
- store if the chain is lower or upper chain
- check visability w/ orientation tests

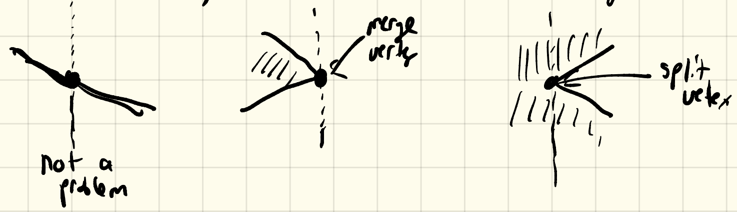
Analysis: claim $O(n)$ time
- montone polygon $\Rightarrow$ sorted list of verts in $O(n)$-time
- check for diag is $O(1)$ orientation tests $\Rightarrow$ $O(1)$ time
- adding daig take $O(1)$ time
- and $O(n)$ diags $\Rightarrow$ $O(n)$-time to Δ monotone polygon of n-verts

Monotone Subdivision

- turn a polygon into multiple monotone polygons

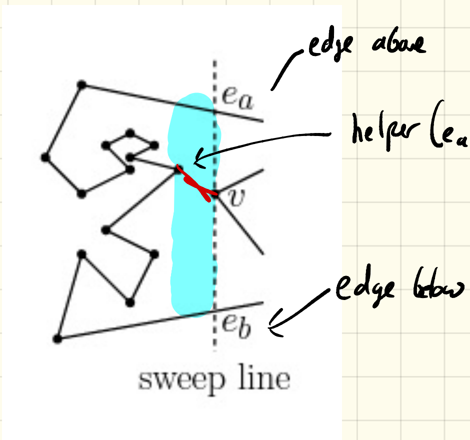
Observations

- x-monotonicty is violated when both edges are to left or right at vertex



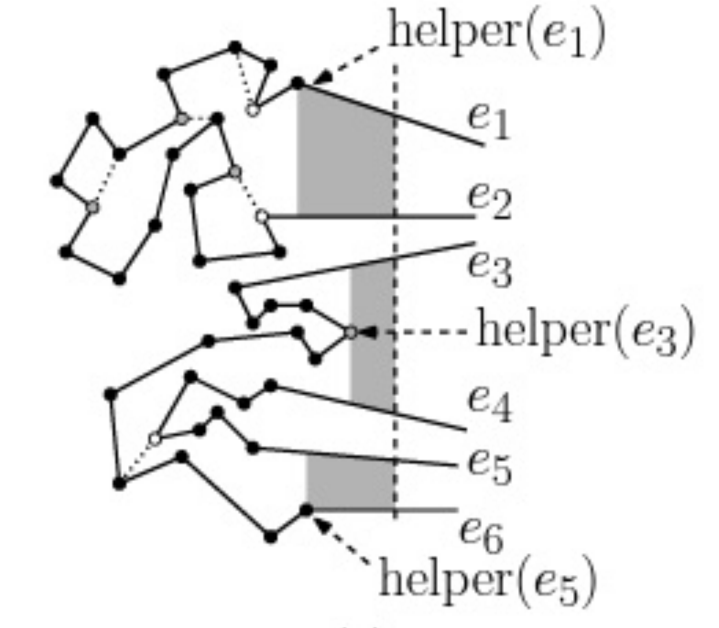
idea:

- plane sweep, add diagonals at split and merge verts
- store some info to add diag quickly



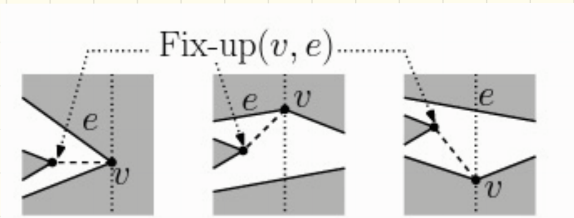
helper(e_a) - a vertex that is visible to any vertex between e_a, e_b

helper(e_a) : Let e_b be the edge of the polygon lying just below e_a on the sweep line. The helper is the rightmost vertically visible vertex on or below e_a on the polygonal chain between e_a and e_b. This vertex may either be on e_a, e_b, or it may lie between them.



Events: verts of the polygon
(no new events are created during algo)

Sweepline status:
- list of edges
  intersecting sweepline
  sorted top to bottom
- Fixup(v,e): if helper(e) is a merge vertex
  add a diag from v to the merge vertex



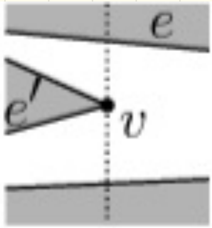
Processing events:

Event types for vertex v

- v is a split vertex
- v is a merge vertex
- v is a start vertex
- v is a end vertex
- v is part of an upper chain
- v is part of a lower chain

merge vertex:

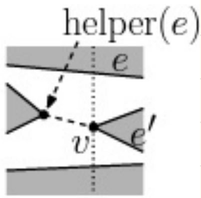
- find edges in the sweepline incident to v
- delete both edges
- e the edge above v
- fixup(v, e)
- fixup(v, e')



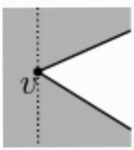
Merge

split vertex:

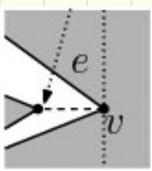
- search sweepline status for edge e above v
- add diag from v to helper(e)
- add 2 edges incident to v to the sweepline status
  label e' the lower of the 2 edges
- set v as the helper fo e and e'



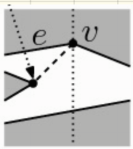
Split



Start



End



Upper

Analysis:

each event perfoms a const # of searches on the sweepline

$\Rightarrow$ each event takes $O(\log n)$ time

# of events is n

$\Rightarrow O(n \log n)$ for decomp

Δtion analysis

$O(n \log n)$ to decompose

k monotone polygons

of size $n_1, n_2, \dots, n_k$

each monotone polygon of size n_i takes $O(n_i)$ to Δlate

$\sum_{i=1}^{k} n_i = n \Rightarrow O(n)$ to Δlate all monotone polygons

$\Rightarrow O(n \log n)$ for Δtion