Polygon Triangulation: Given a simple polygon $P$ (that is, a simple, closed polygonal chain)...

- Simple polygon
- Not simple

Subdivide the interior of $P$ into triangles (vertices drawn from $P$'s vertices)

Notes:
- $P$ given as a cyclic seq. of pts
- Vertices $p_i$ and $p_j$ are visible if open segment $p_ip_j \subseteq \text{int}(P)$
- If $p_i$ and $p_j$ visible, segment $p_ip_j$ called a diagonal
Lemma: Given any $n$-vertex simple polygon ($n \geq 3$)
- A triangulation exists
- Any triangulation has $n-3$ diagonals
- Any triangulation has $n-2$ triangles

Dual Graph: A triangulation defines a graph:
Vertices $\leftarrow$ triangles
Edges $\leftarrow$ adjacent (share common edge)

The dual graph of a polygon triangulation is connected + acyclic $\Rightarrow$ tree

History of Polygon Triangulation:
$O(n^2)$ - Easy (find a diagonal + recurse)
$O(n \log n)$ - We'll present this
$O(n)$ - Chazelle 1991 (very complicated!)
Two steps:
1. Decompose the polygon into (simpler) polygons.
   - Monotone polygons - $O(n \log n)$
2. Triangulate each monotone polygon - $O(n)$

Output: Graph structure, called a doubly-connected edge list (DCEL)

Def: A polygon is $x$-monotone if any vertical intersects the polygon in a single segment (if at all)

Monotone Decomposition - Add (non-intersecting) diagonals so that connected components are all $x$-monotone
Triangulating a Monotone Polygon:

**General position:** No duplicate x-coords (no vertical edges)

**Reflex Vertex:** Internal angle ≥ \( \pi \)

**Reflex Chain:** Sequence of reflex vertices

**General approach:** Sweep from left to right and triangulate as much as we can behind us.

What's the loop invariant?
Lemma: For $i \geq 2$, let $v_i$ be the next vertex to process. The untriangulated region to left of $v_i$ consists of two $x$-monotone chains starting from a common vertex $u$. One chain is a single edge, and the other is a reflex chain (of one or more edges).

For concreteness, let's assume reflex chain is on lower side.

Case 1: ($v_i$ lies on upper chain)
- add diagonals between $v_i$ and all vertices of the chain

[By monotonicity, all are visible to $v_i$]
Now $u = v_{i-1}$. Reflex chain has just one edge.
Case 2: \((v_i \text{ lies on lower chain})\)

2a: \((v_{i-1} \text{ is non-reflex})\)
   - connect \(v_i\) to all visible vertices on chain until hitting point of tangency. (Similar to Graham’s scan)
   [May go all the way back to \(u\)]

2b: \((v_{i-1} \text{ is reflex})\)
   - Add \(v_i\) to the chain

Correctness: Invariant holds after each iteration

Running time: \(O(n)\) [As in Graham, once a vertex is removed from the chain, it never reappears]
Monotone Subdivision:
Recall: Add diagonals to create x-monotone
where? Scan reflex vertex: Reflex vertex
where both edges on same side of
vertical line.

Add a diagonal to right side of each merge
left
split

Plane-sweep Approach:
Need auxiliary info to help with diagonals
For each edge $e_a$ of sweep line with $\text{int}(L)$ below:

$\text{helper}(e_a) =$ rightmost vertically visible
vertex on or below $e_a$
to left of sweep line
Why is the helper helpful?

- When we see a split vertex, we add diagonal to helper of edge above

- When we see a merge vertex, it is the helper of edge above & we connect it to next vertex where helper(ea) changes

Events: Polygon vertices (sorted by x)

Sweep-line status: Edges intersecting the sweep line (ordered dictionary)

Event processing: There are many cases!

Utility:

\[
\text{fix-up}(v,e): \begin{cases} 
\text{if (helper(e) is a merge vertex)} & \text{add diagonal v to helper(e)} 
\end{cases}
\]
Split Vertex (v):
- \( e \leftarrow \text{edge above } v \text{ in sweep line} \)
- add diagonal \( v \) to helper(e)
- insert edges incident to \( v \) into sweep line
- letting \( e' \) be lower, set helper(e') \( \leftarrow v \)

Merge Vertex (v):
- Consider two edges incident to \( v \) + let \( e' \) be lower one
- Delete both from sweep line
- Let \( e \) be edge above \( v \)
- fix-up(v,e) + fix-up(v,e')

Start vertex (v):
- Insert \( v \)'s incident edges into sweep line
- Letting \( e \) be upper edge, helper(e) \( \leftarrow v \)

End vertex (v):
- Consider the two incident edges + let \( e \) be upper edge
- Delete both from sweep line
- fix-up(v,e)
**Upper-chain vertex (v):**
- Let e be edge to left, e' to right
- fix-up (v, e)
- Replace e with e' in sweep line
- helper(e') ← v

**Lower-chain vertex (v):**
- Let e be edge above
- fix-up (v, e)
- Let e' be edge to left, e'' to right
- Replace e' with e'' in sweep line