Triangulating a Polygon

Recall: PSLG = Planar Straightline graph.

**Def** (Simple) **Polygonal chain** is a PSLG consisting of a simple cycle $P$.

**Claim** A Polygonal chain has a unique interior.

**Def** **Polygon** is Polygonal chain + interior

**Triangulation**: Addition of seg so that

1) Still PSLG
2) Interior is decomposed into triangles
Theorem: Every simple polygon can be triangulated.

Proof: Induct on # of edges on points $n$.

Base case: $n = 3$.

Assume no $180^\circ$ angles. $n > 3$.

Let $V$ be the leftmost point with neighbors $W$ & $U$.


We get two poly $|P_1| = 3$, $|P_2| = n - 1$.

2) Case 2: 3 point $V'$ interior to $\text{Tri} = [V, V', U]$.

Let $V'$ be the leftmost such point.

$\text{Deg} = [V, V']$ is interior.

$|P_1| < n$ & $|P_2| < n$. 

Thm. Not every simple polygonal surface in 3D can be decomposed in tetrahedra.

By contrast, consider the tetrahedron with faces B.

Missing vertex is X or Y, not Z.

Not X since \( \text{seg}[X, Z] \) outside.

"Y" \( \overline{Y, C} \) too.

In general, problem is NP-hard.
Guarding A Polygon

Input: Polygon P

Output: locations $p_1, p_2, \ldots, p_k \in P$ (guards)

1) Guards cover $P$
2) $k$ small.

Thm A polygon $P$ with $n$ vertices

$\lceil \frac{n}{3} \rceil$ guards suffice and maybe necessary.

$P = \frac{n}{3}$ prongs

$|P| = n$ Needs a guard per prong.
\( 1/3 \)-guards Alg\((P)\)

1) Tri \( P \) \( \overline{P} \)
2) 3-color \( \overline{P} \)
   a) Construct geometric dual \( T \) (a tree)
   b) 3-Color \( \overline{P} \) by traversing trees in an inorder fashion.
3) Pick least used color.

Only non-linear time step is 1)
2D - Algorithm

Proof \implies O(n^2)

Known: O(n) Chazelle

Today: O(n \log n) (line/sweep line)

This Class: O(n \log^* n) Seidel (incremental randomized)

Def \log^* n = \min \{ \log \ldots \log n \mid n \leq 1 \}

\[\text{Prob: Give a } O(n) \text{ time alg to determine which side of an edge in interior/exterior of a simple poly}\]

\[\text{Prob: test } P \in \text{ Int}(P) \text{ in } O(n) \text{ time.}\]

3.14 \ O(n \log n) \ OK

\ O(n) \ ?

\text{Trap } \implies \text{ Try}
Step 1: Partition into Monotone Polygons

Def: Pro Y-monotone if
Every horizontal line l
\( l \cap P \) is connected or empty

Alg: Type: Line Sweep \( O(n \log n) \) time

Def:
\( \square \) = start vertex
\( {}\square {} \) = end
\( \bullet \) = ref
\( \triangle \) = split
\( \nabla \) = merge
Claim \( P \) is \( y \)-monotone iff no split or merge vertices.

\((\Rightarrow)\) (easy)

\((\Leftarrow)\) (not mono \( \Rightarrow \) \( \exists \) split or merge)

Assume not mono

Case 1

Case 2

merge

split
Algorithm: Sweep Line (top-to-bottom)

Events: endpoints

Dictionary: Intervals (sorted)

Interval: (left-segment, right-segment, helper vertex)

Helper vertex: 2 edges before higher

Define: helper(e, e') = the lowest vertex above line and between e and e'

No horizontal segment.

Procedure: \text{add}((p, q)) \equiv \text{add segment from } p \text{ to } q.
Make Monotone \( g \) event

Case (Start Vertex)
1) Add new interval
2) set helper \( \leq g \)

Case (End Vertex)
1) if helper is merge vertex then add \((g, \text{helper})\)
2) remove interval

Case (Regular)
\[
\begin{align*}
&c \\
\cdot &\geq \\
\cdot &e \\
\cdot &e'
\end{align*}
\]
1) add \((g, \text{helper})\)
2) replace \(e\) with \(e'\)
3) helper \(\leq g\)

Case (Split)
1) add \((g, \text{helper})\)
2) "split" interval say \(I_1, I_2\)
3) helper(I1) = helper(I2) \(\leq g\)
Case (Merge)

1) \text{add}(\text{help}_L, 8), \text{add}(\text{help}_R, 8)

2) "Merge" intervals

3) help \leftarrow 8
Another View

1) Make Trapezoidal Decom (sweep line)
2) For each trap add a diagonal if possible
   a) types of traps

![Diagram](image)