Triangulation

Efi Fogel

Tel Aviv University

Computational Geometry
Mar. 3rd, 2014

Outline

Polygon Terms and Definitions

Definition (Polygon)

A polygon is a region of the plane bounded by a finite collection of line segments forming a simple close curve.

Theorem (Jordan Curve Theorem)

If C is a simple closed curve in $\mathbb{R}^2$, then $\mathbb{R}^2 \setminus C$ has two components (an "inside" and "outside"), with C the boundary of each.

Definition (Simple Polygon)

A polygon is said to be simple (or Jordan) if it is enclosed by a single closed polygonal chain that does not cross itself. In particular, the polygon edges are pairwise disjoint in their interior and the degree of all vertices is two.

Polygon Terms & Definitions (Cont.)

The chain $v_1, v_2, \ldots, v_n$ defines a simple polygon iff

1. The segments $s_1 = v_1v_2, s_2 = v_2v_3 \ldots, s_{n-1} = v_{n-1}v_n, s_n = v_nv_1$ are disjoint in their interior.
2. Consecutive segments intersect only in their endpoints. Namely $s_i \cap s_{i+1} = v_{i+1}, i = 1, 2, \ldots, n-1$ and $s_n \cap s_1 = v_1$
3. Non adjacent segments do not intersect $s_i \cap s_j = \emptyset, j > i + 1$.

- P — a simple polygon.
- ∂P — the boundary of P.
- $\partial P \subseteq P$, P is closed and contains its boundary.
- By convention the vertices of a polygon are ordered counterclockwise around the interior of the polygon.
  - Interior of polygon is to the left of the boundary.

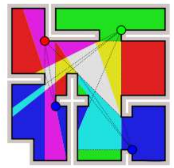
Outline

Application: Art Gallery

Application (Art Gallery)

Given the floor plan of an art gallery modeled as a simple polygon with n vertices. Find out how many (and where) guards are needed to see the entire gallery, where each guard is stationed at a fixed point, has 360° vision, and cannot see through the walls.

Problem posed to Vasek Chvatal by Victor Klee at a math conference in 1973. Chvatal solved it quickly with a complicated proof, which has since been simplified significantly using triangulation.



Art Gallery: Lower Bound

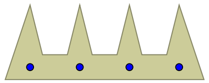
Definition (Seeing)

A (guard) point p sees points $q \in P$ if $pq \subseteq P$.

Definition (Covering)

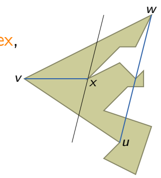
A set of guards G covers a polygon P if for any point $p \in P$ there is a guard $g \in G$ that sees p.

- $g(P)$ — minimum number of guards guarding P.
  - Cardinality of smallest set that covers P.
- $\mathcal{P}_n$ — set of all simple polygons with n vertices.
- $G(n) = \max_{P \in \mathcal{P}_n} g(P)$ — maximum number of guards needed to guard a simple polygon with n vertices.
- $G(n) \geq \lfloor n/3 \rfloor$



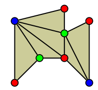
Art Gallery: Upper Bound

- A diagonal of a polygon P is a segment connecting two vertices of P that strictly see each other.
- A triangulation is a partition of P into triangles formed by repeatedly inserting diagonals into P.
- A vertex is strictly convex if its interior angle $\alpha < \pi$.
- The interior angle of a reflex convex is $\alpha > \pi$.
- Every polygon has at least one strictly convex vertex,
  - e.g., the lexicographically smallest vertex v.
- Every polygon with $n > 3$ vertices has a diagonal.
- Every polygon may be partitioned into triangles by the addition of (0 or more) diagonals.
  - Proof by induction.
- T — a triangulation of a polygon P of n vertices.
  - T uses $n-3$ diagonals and consists of $n-2$ triangles.



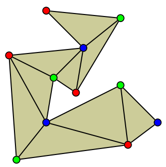
Art Gallery: Upper Bound (Cont.)

- The dual of a triangulation T is a graph $G(T)$ with a node associated with each triangle and an arc between two nodes iff their triangles share an edge.
- The dual graph $G(T)$ is a tree with a vertex degree at most 3.
- 3 consecutive vertices u, v, w, form an ear if uw is a diagonal
  - v is the ear tip.
- Every polygon of $n > 3$ has at least 2 non-overlapping ears.
- The graph of the triangulation $T(P)$ is three colorable.
- Every simple polygon P with n vertices can be guarded using $\leq \lfloor n/3 \rfloor$ guards; $G(n) \leq \lfloor n/3 \rfloor$.
- Compute the triangulation of P.
- Compute a 3 coloring for $T(P)$.
- Choose the smallest set of vertices with the same color.
  - Its cardinality must be $\leq \lfloor n/3 \rfloor$.



Art Gallery: Minimum Number of Guards

- A 3-coloring of the vertices yields 3 guards.
- However, the polygon can be guarded by only 2 guards.
- Finding the minimum number of guards is NP-hard.



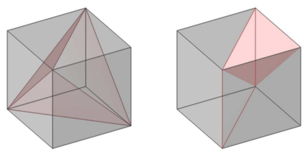
Problem (Art Gallery Decision)

Given both a polygon and a number k, determine whether the polygon can be guarded with k or fewer guards.

- Even the decision problem and all of its standard variations (such as restricting the guard locations to vertices or edges of the polygon) is NP-hard.

Art Gallery in $\mathbb{R}^3$

- Even n-vertex guards do not suffice!
- Different triangulations can have different number of tetrahedra.
- Determining whether a polyhedron requires Steiner vertices for triangulation is NP-Complete.
  - Smallest example of a polyhedron that cannot be triangulated without adding new vertices. (Schoenhardt [1928]).
- Every 3D polyhedron with n vertices can be triangulated with $O(n^2)$ tetrahedra. [Cha84]



5 tetrahedra 6 tetrahedra

Outline

Triangulation History

- Check all n^2 choices for a diagonal, each in $O(n)$ time. Repeat this $n-1$ times, $O(n^4)$.
- Find an ear in $O(n)$ time; then recurse, $O(n^2)$ time.
- First non-trivial algorithm: $O(n \log n)$. [GJP+78]
- A long series of papers and algorithms in 80s until Chazelle produced an optimal $O(n)$ algorithm. [Cha91]
- Linear time algorithm insanely complicated; there are randomized, expected linear time algorithm that are more accessible.

Regularization and Triangulation Algorithm Outline

Definition (Monotone Polygonal Chain)

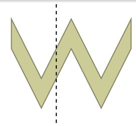
A polygonal chain C is monotone w.r.t. line L if any line orthogonal to L intersects C in at most one point.

Definition (Monotone Polygon)

A polygon is monotone w.r.t. L if it can be decomposed into two chains, each monotone w.r.t. L.

- Partition polygon into trapezoids.
- Use trapezoids to make a monotone subdivision.
- Triangulate each monotone piece.

[GJP+78]



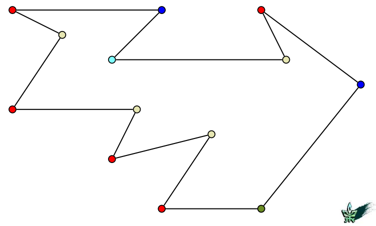
An x-monotone polygon

Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — *merge vertex*
2. ○ — split vertex
3. • — start vertex
4. • — end vertex
5. • — regular vertex

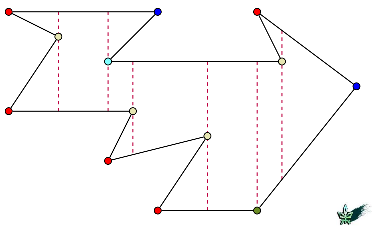


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — *merge vertex*
2. ○ — split vertex
3. • — start vertex
4. • — end vertex
5. • — regular vertex

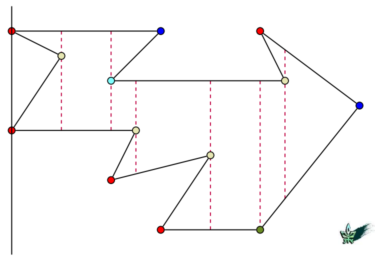


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — *merge vertex*
2. ○ — split vertex
3. • — start vertex
4. • — end vertex
5. • — regular vertex

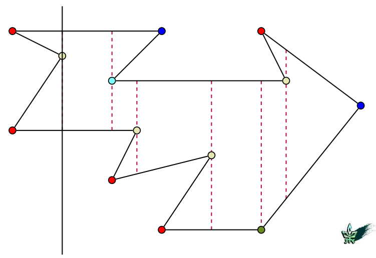


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — *merge vertex*
2. ○ — split vertex
3. • — start vertex
4. • — end vertex
5. • — regular vertex

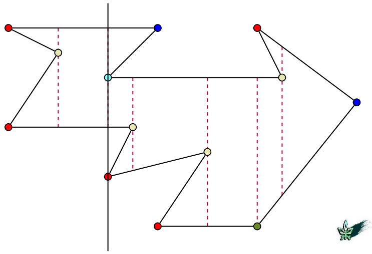


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

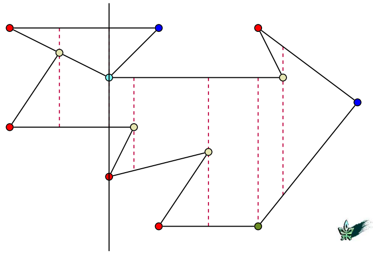


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

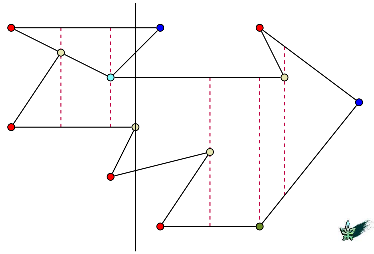


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

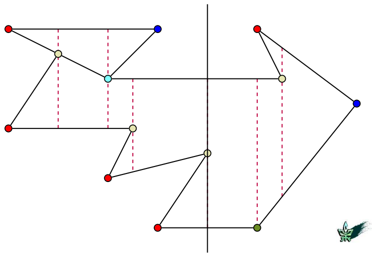


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

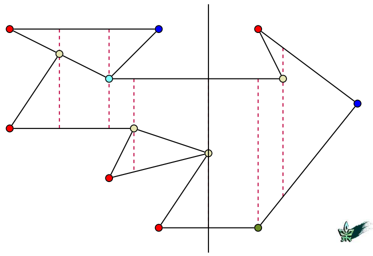


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

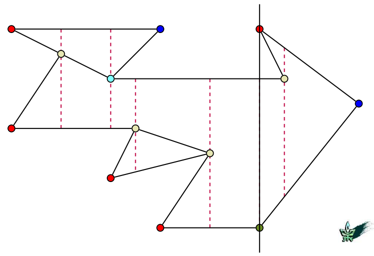


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

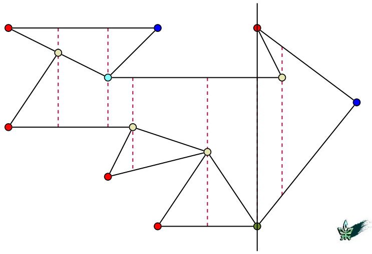


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

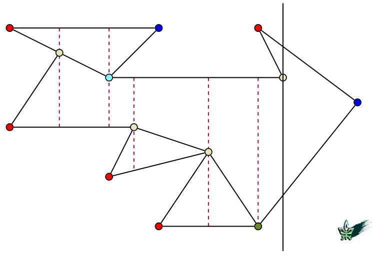


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

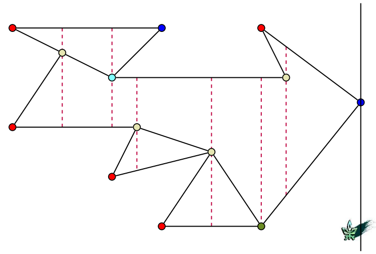


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex

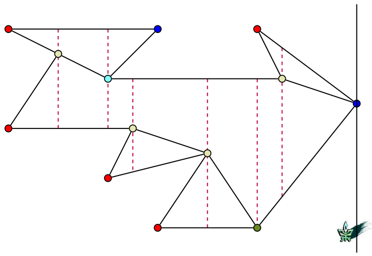


Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a psedo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex v, add a diagonal that connects v to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — split vertex
3. — start vertex
4. — end vertex
5. — regular vertex



Regularization and Triangulation Algorithm Proof

Lemma (x-monotone)

A polygon is x-monotone if it has no split vertices and no merge vertices.

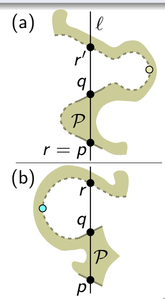
- Suppose P is a non x-monotone polygon.
- We have to prove that P contains a split or a merge vertex.

q — the top endpoint of $\ell \cap \mathcal{P}$.

p — the bottom endpoint of $\ell \cap \mathcal{P}$.

r — the first intersection of ℓ with $\partial\mathcal{P}$ starting at q going counterclockwise.

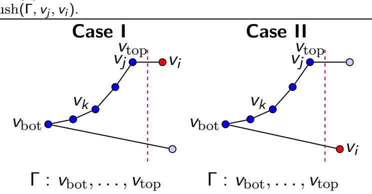
- There are two cases
  - (a) $p = r \Rightarrow \exists$ merge vertex
  - (b) $p \neq r \Rightarrow \exists$ split vertex



Triangulating Monotone Polygons

Triangulate a monotone polygon P on n vertices.

```
Sort the vertices in lexicographically increasing order to yield v_1, v_2, ..., v_n
Initialize a stack Γ, push(Γ, v_1, v_2).
for i = 3, ..., n do
    if v_i and v_j = top(Γ) on same chain
        I { Add diagonals v_i v_j, ..., v_i v_k, where v_k is last to admit legal diagonal.
          { v_{k+1}, ..., v_j = pop(Γ).
          { push(Γ, v_i).
    else
        II { Add diagonals from v_i to all vertices on stack.
           { v_j = pop(Γ).
           { clear(Γ).
           { push(Γ, v_j, v_i).
```

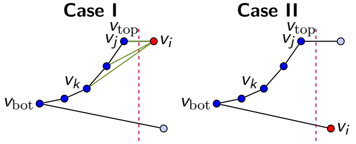


Triangulating Monotone Polygons

Triangulate a monotone polygon P on n vertices.

Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$
Initialize a stack Γ, $\mathrm{push}(\Gamma, v_1, v_2)$.
for $i = 3, \ldots, n$ **do**
  if v_i and $v_j = \mathrm{top}(\Gamma)$ on same chain
I   Add diagonals $v_i v_j, \ldots, v_i v_k$, where v_k is last to admit legal diagonal.
    $v_{k+1}, \ldots, v_j = \mathrm{pop}(\Gamma)$.
    $\mathrm{push}(\Gamma, v_i)$.
  else
II   Add diagonals from v_i to all vertices on stack.
    $v_j = \mathrm{pop}(\Gamma)$.
    $\mathrm{clear}(\Gamma)$.
    $\mathrm{push}(\Gamma, v_j, v_i)$.

Case I **Case II**



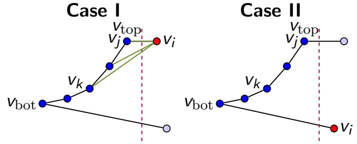
Γ : $v_{\mathrm{bot}}, \ldots, v_{\mathrm{top}}$ Γ : $v_{\mathrm{bot}}, \ldots, v_{\mathrm{top}}$

Triangulating Monotone Polygons

Triangulate a monotone polygon P on n vertices.

Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$
Initialize a stack Γ, $\mathrm{push}(\Gamma, v_1, v_2)$.
for $i = 3, \ldots, n$ **do**
  if v_i and $v_j = \mathrm{top}(\Gamma)$ on same chain
I   Add diagonals $v_i v_j, \ldots, v_i v_k$, where v_k is last to admit legal diagonal.
    $v_{k+1}, \ldots, v_j = \mathrm{pop}(\Gamma)$.
    $\mathrm{push}(\Gamma, v_i)$.
  else
II   Add diagonals from v_i to all vertices on stack.
    $v_j = \mathrm{pop}(\Gamma)$.
    $\mathrm{clear}(\Gamma)$.
    $\mathrm{push}(\Gamma, v_j, v_i)$.

Case I **Case II**



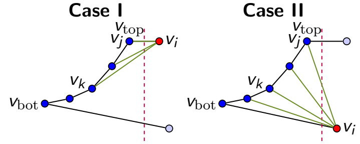
Γ : $v_{\mathrm{bot}}, \ldots, v_k, v_i$ Γ : $v_{\mathrm{bot}}, \ldots, v_{\mathrm{top}}$

Triangulating Monotone Polygons

Triangulate a monotone polygon P on n vertices.

Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$
Initialize a stack Γ, $\mathrm{push}(\Gamma, v_1, v_2)$.
for $i = 3, \ldots, n$ **do**
  if v_i and $v_j = \mathrm{top}(\Gamma)$ on same chain
I   Add diagonals $v_i v_j, \ldots, v_i v_k$, where v_k is last to admit legal diagonal.
    $v_{k+1}, \ldots, v_j = \mathrm{pop}(\Gamma)$.
    $\mathrm{push}(\Gamma, v_i)$.
  else
II   Add diagonals from v_i to all vertices on stack.
    $v_j = \mathrm{pop}(\Gamma)$.
    $\mathrm{clear}(\Gamma)$.
    $\mathrm{push}(\Gamma, v_j, v_i)$.

Case I **Case II**



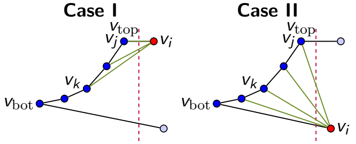
Γ : $v_{\mathrm{bot}}, \ldots, v_k, v_i$ Γ : $v_{\mathrm{bot}}, \ldots, v_{\mathrm{top}}$

Triangulating Monotone Polygons

Triangulate a monotone polygon P on n vertices.

Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$
Initialize a stack Γ, $\mathrm{push}(\Gamma, v_1, v_2)$.
for $i = 3, \ldots, n$ **do**
  if v_i and $v_j = \mathrm{top}(\Gamma)$ on same chain
I   Add diagonals $v_i v_j, \ldots, v_i v_k$, where v_k is last to admit legal diagonal.
    $v_{k+1}, \ldots, v_j = \mathrm{pop}(\Gamma)$.
    $\mathrm{push}(\Gamma, v_i)$.
  else
II   Add diagonals from v_i to all vertices on stack.
    $v_j = \mathrm{pop}(\Gamma)$.
    $\mathrm{clear}(\Gamma)$.
    $\mathrm{push}(\Gamma, v_j, v_i)$.

Case I **Case II**



Γ : $v_{\mathrm{bot}}, \ldots, v_k, v_i$ Γ : v_j, v_i

Regularization and Triangulation Algorithm Complexity

- Regularization via plane sweep takes $O(n \log n)$ time.
- Triangulation
  - Sorting by merging the two monotone chains of P takes $O(n)$ time.
  - A vertex is added to stack once. Once it's visited during a scan, it's removed from the stack.
  - In each step, at least one diagonal is added; or the reflex stack chain is extended by one vertex.
  - Triangulating a monotone polygon takes $O(n)$ time.
- Total time for polygon triangulation is therefore $O(n \log n)$.

Outline

Triangulation Bibliography I

B. Chazelle.
Triangulating a Simple Polygon in Linear Time.
Discrete and Computational Geometry, 6:485–524,1991.

M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan.
Triangulating a Simple Polygon.
Information Processing Letter, 7(4): 175-180, 1978.

R. E. Tarjan and C. J. van Wyk.
An $O(n \log \log(n))$ Time Algorithm for Triangulating a Simple Polygon.
SIAM Journal of Computing, 17(1):143-178, 1988.

Mark de Berg, Mark van Kreveld, Mark H. Overmars, and Otfried Cheong.
Computational Geometry: Algorithms and Applications.
Springer, 3rd edition, 2008. Chapter 3: Polygon Triangulation.

Alexey V. Skvortsov, Yuri L. Kostyuk
Efficient algorithms for Delaunay triangulation.
Geoinformatics. Theory and practice (Tomsk State University) 1: 22–47, 1998.

Bernard Chazelle.
Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm.
SIAM Journal on Computing, 13:488–507, 1984.

Michael J. Laszlo.
Computational Geometry and Computer Graphics in C++.
Prentice-Hall, 1996