Shortest Paths in polygon/polygonal region

- locally shortest means no local change can shorten the path.

- shortest path implies locally shortest.

Lemma: Any locally shortest path among polygonal objects is composed of line segments between vertices.

Proof: If a path curves, or bends except at a vertex, we can shorten it.

Shortest path in simple polygon

- For given start, there is only one locally shortest path.

- Can be found in $O(n)$ time after triangulating "funnel algorithm" (won't cover).
Shortest paths in plane with polygonal objects.

Use **Visibility Graph**
- nodes are vertices of polygons
- edge when two vertices "see" each other, i.e.
  line segment between them does not intersect
  interior of any object
- weight of edge = Euclidean distance
- see slide

Problem becomes: find shortest path in visibility graph

Use Dijkstra \(O(m + n \log n)\)

\(m = \# \text{edges} \) can be \(\Omega(n^2)\)

Note: need real RAM to compare sum of square roots.

---

Computing visibility graph - see slide for history

Welzl's \(O(n^3)\) algorithm
- for disjoint non-degenerate line segs.

Initialization:
Find, for every vertex, which segment it sees horizontally to the right

Use plane sweep: \(O(n \log n)\)
Then sweep direction vector cyclically

we find a visibility edge $ab$ when direction is parallel to it

let $vis(v) = \text{segment seen by } v$ (for current direction)

As the direction vector changes, visibility only changes when direction vector goes thru 2 vertices

Find all $\binom{n}{2}$ lines thru pairs of points and sort by slope $O(n^2 \log n)$

we will see a faster way.

---

How to update to slope through $a$ and $b$:

Let $s = vis(a)$

<table>
<thead>
<tr>
<th>Case 1</th>
<th>$ab$ is a segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>same visibilities</td>
<td></td>
</tr>
<tr>
<td>output $(a, b)$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 2</th>
<th>$s$ blocks $a$ from seeing $b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>same visibilities</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 3</th>
<th>$b$ an endpoint of $s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>output $ab$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 4</th>
<th>otherwise $a$ sees $b$, $b$ starts new segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>output $ab$</td>
<td></td>
</tr>
</tbody>
</table>

$vis(a) \leftarrow vis(b)$

$vis(a) \leftarrow \text{segment at } b$
Construct a directed acyclic graph.

Compute arrangement. $O(n^2)$

Direct edges left to right.

Use a "topological order" of this graph.

If $x \rightarrow y$, deal with $x$ before $y$. 
Shortest paths reach a segment on edge \( e \) via a cone.

How a cone expands to next triangle:

- May continue
- May split in two

Keep track of segments and the rays of cones reaching segment endpoints.

Build a segment tree — nodes are segments.

A node has 1 or 2 children.

**Lemma.** After depth \( n \) the tree captures all shortest paths.

**Proof.** A shortest path goes through \( \leq n \) faces (since never repeat a face).

Then, just compare all \( s \rightarrow t \) paths in tree for shortest.

OK, but this is exponentially big.
How to prune the segment tree.

Lemma ("one vertex one cut")

Suppose 2 segments $s_1$ and $s_2$ both split at vertex $v$ in tree

Then we can trim one of the 4 children...

Pf

$\sigma_1$ is shortest path $s_0 \to v$ in tree thru $s_1$

$\sigma_2 = s_0 \to v \cdots s_2$

Suppose $|\sigma_2| < |\sigma_1|$. Then prove $s_{12}$ never useful.

Consider a path through $s_{12}$

It crosses $\sigma_2$ at $x$.

Notation $\sigma_2(s_0, x) = \text{subpath of } \sigma_2$ from $s_0$ to $x$.

Claim $|\sigma(s_0, x)| > |\sigma_2(s_0, x)|$

Then $\sigma$ is not shortest.

Pf of claim

$|\sigma(s_0, x)| + |\sigma_2(x, v)| > |\sigma_1(s_0, v)| = |\sigma_1| > |\sigma_2|$

$= |\sigma_2(s_0, x)| + |\sigma_2(x, v)|$

So $|\sigma(s_0, x)| > |\sigma_2(s_0, x)|$. 
Size of segment tree:
- \( O(n) \) leaves because each vertex \( v \) in triangle \( T \) contributes one "branch" of height \( O(n) \) (no path goes through more than \( n \) triangles)

Thus \( O(n^2) \) size of tree. Time \( O(n^2) \) to construct level by level (each new level has \( O(n) \) children)

Further reduce space.

Compress chain of single vertices (= come through successive \( \Delta \)'s)

Time \( O(n^2) \) (still)

Dealing with non-convex vertices: actually negative curvature vertices.
Shortest paths may go through these vertices (think saddle points vs. mountain tops).

Vertex of negative curvature.

Range of locally shortest paths.

Each such vertex is treated as a "pseudo-source".
Applications to Unfolding — see slides