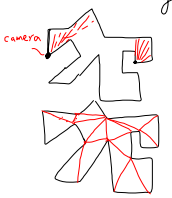
Polygon Triangulation

Motivation: Art Gallery Problem

Art Gallery ~ simple polygon ~ no holes



How many cameras does it take to guard an art gallery?

Upper bound: no of triangles it takes to divide up the polygon (Triangulation)

Theorem: Any simple polygon can be triangulated - if it has n vertices we require n-2 triangles.

říj 11-15:53

- So upper bound for no of cameras is n-2.
- In fact, can do better - $\lfloor n/3 \rfloor$ suffice
  uses triangulation (find 3-coloring - see book)

$\lfloor n/3 \rfloor$ Sometimes required

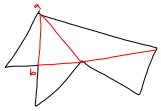
6 vertices
$\lfloor 6/3 \rfloor = 2$

Today: polygon triangulation algorithm

říj 11-16:12

Theorem: Each simple polygon can be triangulated - any triangulation of polygon with n vertices requires n-2 triangles.

Eg:



7 vertices
5 triangles

Proof n=3  is obvious.

Will prove by induction.

říj 11-16:19

- By a diagonal we mean a straight line segment $\overline{ab}$, whose endpoints are vertices & which otherwise belongs to interior of polygon. Eg: ($\overline{ab}$ in previous polygon)
- Any diagonal $\overline{ab}$ splits polygon $P = A \cup_{\overline{ab}} B$ with m,k vertices where $m,k < n$ & $m+k = n+2$
- Triangulations of A & B can be combined - so result on existence of triang. follows by induction if we can prove existence of diagonal.
- Also formula for no of triangles follows:

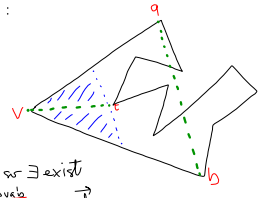
$P = A \cup_{\overline{ab}} B$ (n verts; m, k)

obtain triangulation of A & B in m-2 & k-2 triangles.

So P has triang in $(m-2)+(k-2) = m+k-4 = n-2$ triangles.

říj 11-16:23

Existence of diagonal:

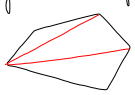


- Let v be smallest vertex on P in lexicographic order: x co-ord first, then y co-ord.
- let a,b be vertices connected to v.
- If $\overline{ab}$ is a diagonal, we are done.
- If not, an edge must cross $\overline{ab}$, or ∃ exist vertices of P lying inside $\triangle vab$
- Let t be the furthest such vertex from $\overline{ab}$.
- If vt did not lie in interior, an edge would have to cross it, & one of its endpoints would then lie inside blue region (ie closer to v than t). But this is impossible.

Thus vt a diagonal.

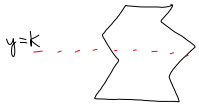
říj 11-16:32

- Goal: find alg with complexity O(nlogn).
- Convex polygon:



~ easy to Triangulate: draw line from vertex v to any other.

- Weaker notion (still easily Triangulable) of monotone polygon: monotone polygon with respect to axis y:



any horizontal line y=k intersects polygon in line segment, point or empty set.

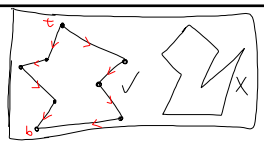
- We work with slightly stronger notion of monotone polygon (more computationally effective)

říj 11-16:41

Monotone polygon.

Consider lex ordering:

$a > b \Leftrightarrow a_y > b_y$ or $(a_y = b_y \ \& \ a_x < b_x)$

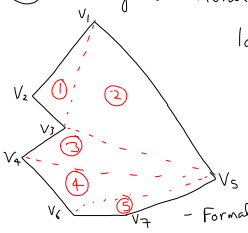


- Determines two paths from t (top) to b (bottom)
- The polygon is monotone if both paths are decreasing (with respect to lex order)

Algorithm: ① divide polygon into monotone pieces
② triangulate monotone polygon.

říj 11-16:52

② Triangulate monotone polygon



Idea: use lex ordering on vertices.

- Draw diagonals to all possible preceding vertices (type within) & break off triangles.
- Formally: start with monotone polygon stored in a DCEL D.
- Using this, calculate left & right path from top vertex t to bottom b (in lex order)
- Merge two paths into 1 lexicographically ordered list: $v_1, \ldots, v_n$.

říj 11-17:01

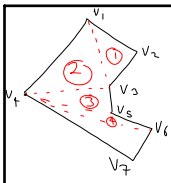
- Initialise empty stack S
  - push v_1 & v_2 onto it — so (v_2, v_1)

For $j = 3$ to $n-1$:

- if v_j & vertex on top of S are on diff. chains:
  - pop all vertices from S
  - Add a diagonal (in D) to each popped vertex except the last one removed.
  - Push v_{j-1} & v_j onto S.
- pop all vertices from S as long as diagonals from v_j to them lie inside P
  - Add those diagonals to D.
  - Push last popped vertex back onto S.
  - Push v_j onto S.

At v_n, add diagonals to all vertices of stack except first & last.

říj 11-17:14



(v_2, v_1) At v_3 - pop v_2 so (v_1)
- Pop v_1 so ( )
- Add ① to D.
- Then push v_1 & v_3 on - so $S = (v_3, v_1)$.

- At v_4: empty stack - so $S = ( )$.
  - Add ② (ie. v_4v_3) to D.
  - Push last 2 onto S - (v_4, v_3).
- At v_5 stack becomes (v_3, v_4) & add diag. v_5v_4 to D.
  At v_6 stack becomes (v_6, v_4) & add v_4v_6.
- At v_7 do nothing.

říj 11-17:23

Complexity:
- calc top, bottom verts $O(n)$
- calc paths from top to bottom $O(n)$
- Merge 2 paths takes time $O(n)$
- During running of loops each vertex added/removed at most twice - so $O(n)$
- Total time: $O(n)$

říj 11-17:33