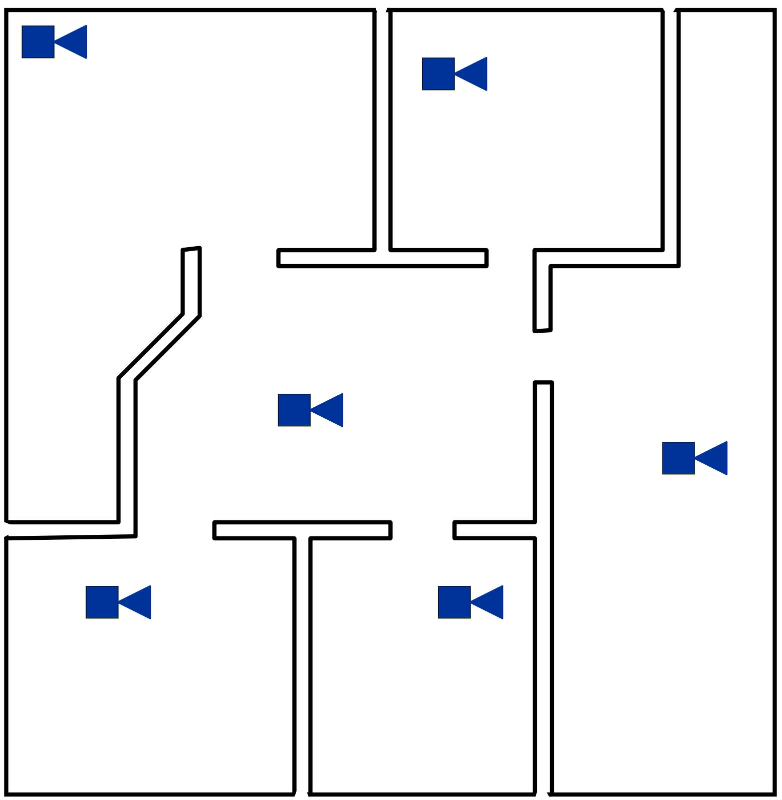
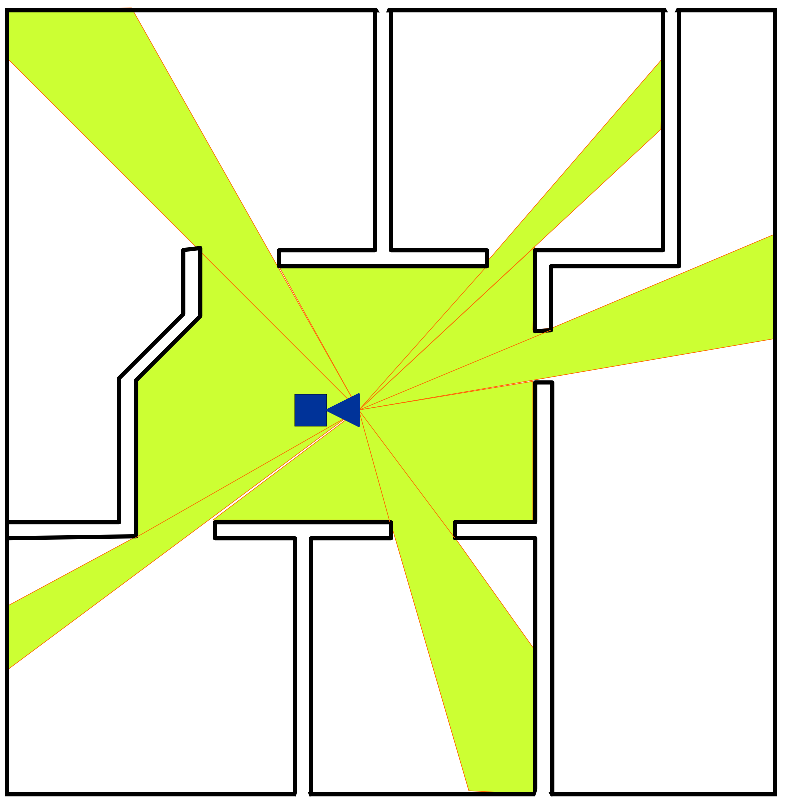
Polygon Triangulation

slides by Andy Mirzaian

(a subset of the original slides are used here)

Guarding an Art Gallery



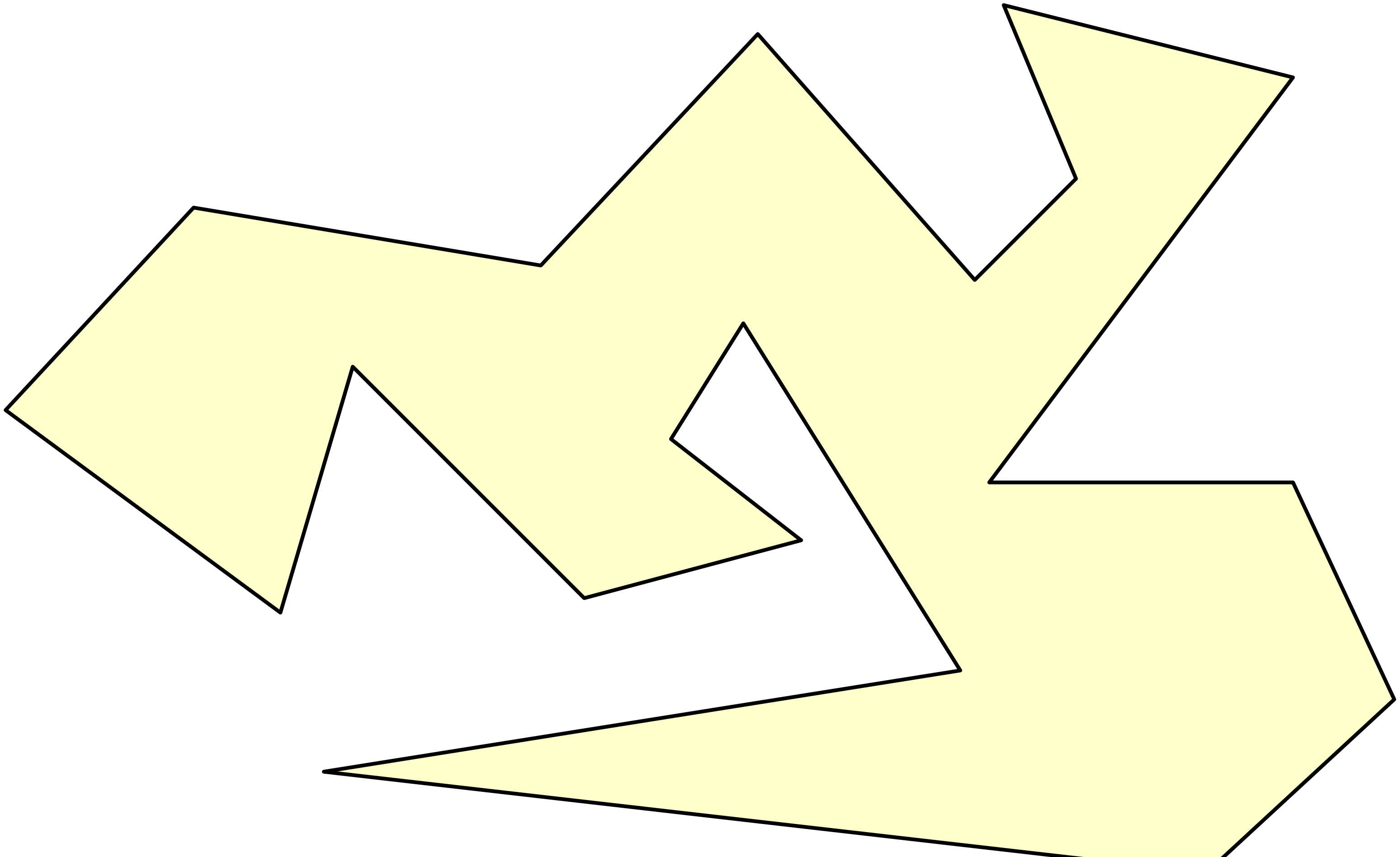


Art Gallery Problem [Victor Klee 1973]

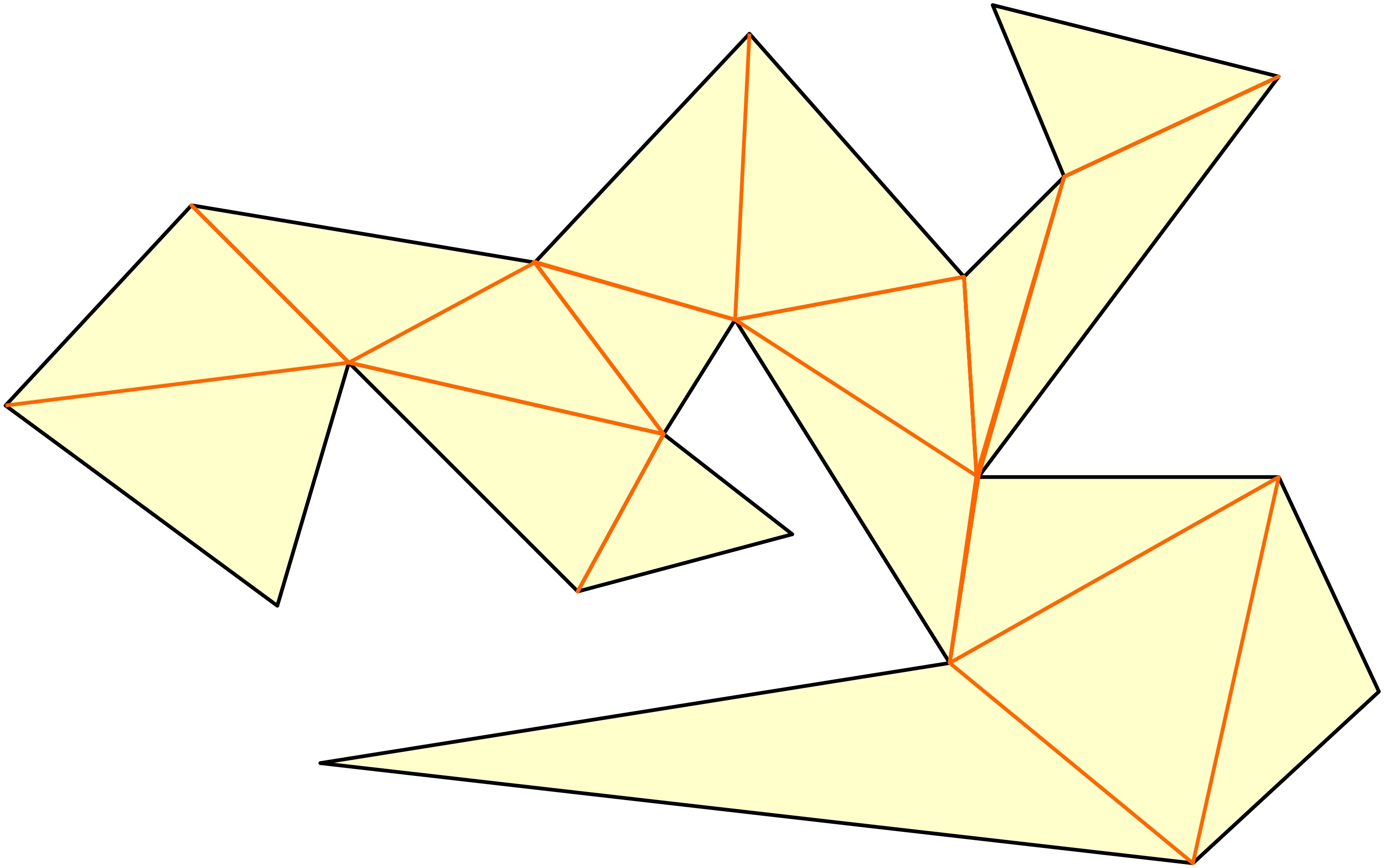
How many camera guards do we need to guard a given gallery and how do we decide where to place them?

- It's NP-hard to determine the MINIMUM number of camera guards for an arbitrary given simple polygon [Aggarwal 1984].
- Let P be an n-vertex simple polygon.
- If P is convex, then a single guard anywhere inside P is sufficient.
- n guards for P are always sufficient; one guard at each <u>vertex</u>.
  [This does not work for 3D polytopes!]
- Can we use less than n guards? Yes. Use Triangulation of P.

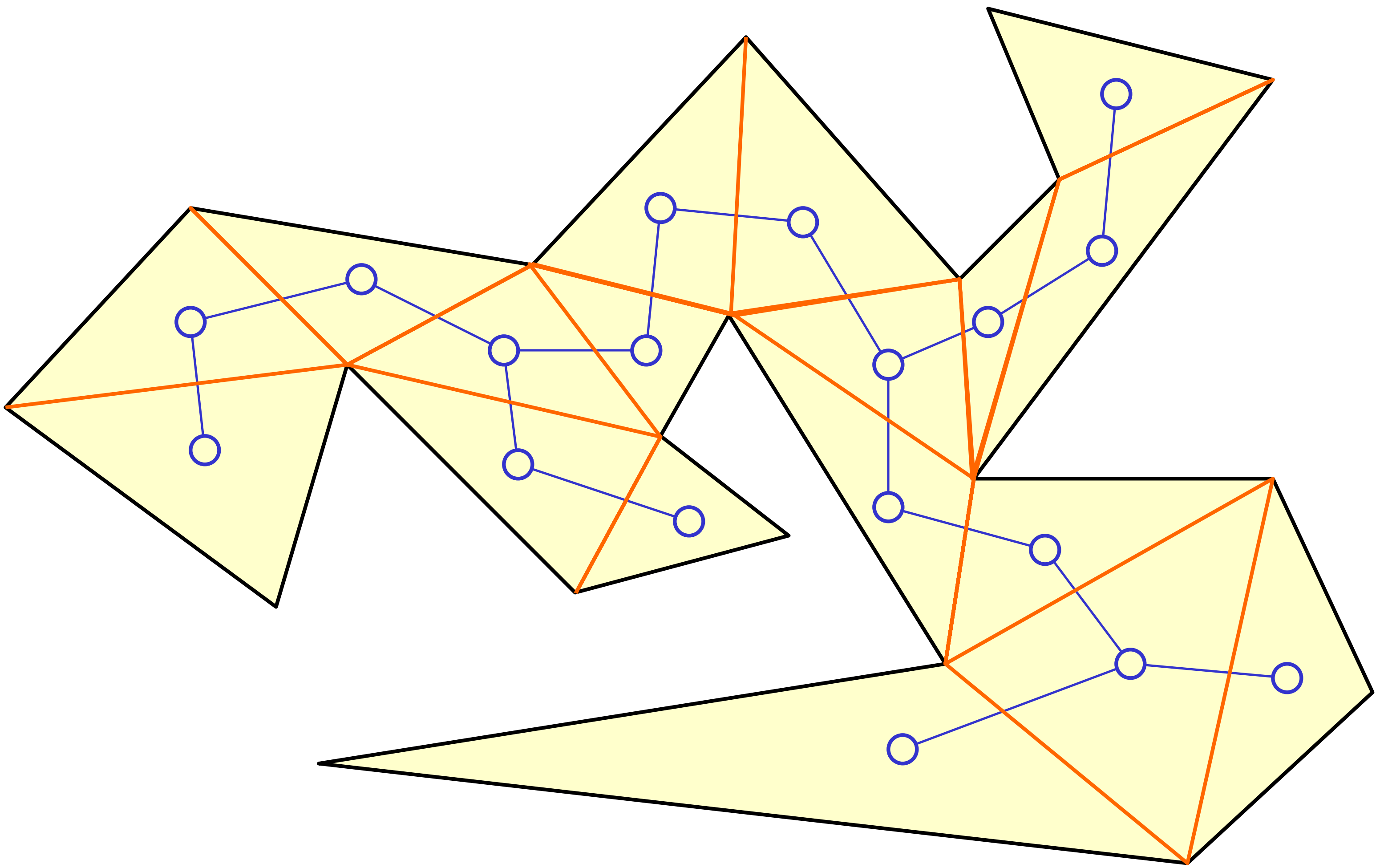
A simple polygon P



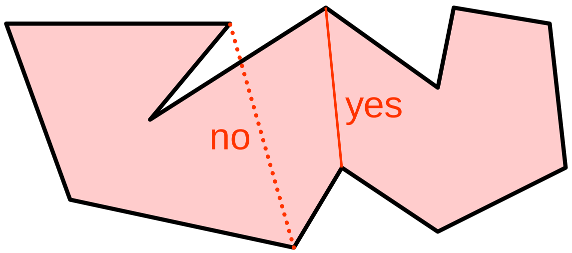
A triangulation of P



Dual Tree of the Triangulation



Diagonal of a simple polygon P: Any line-segment between two non-adjacent vertices of P that is completely inside P.

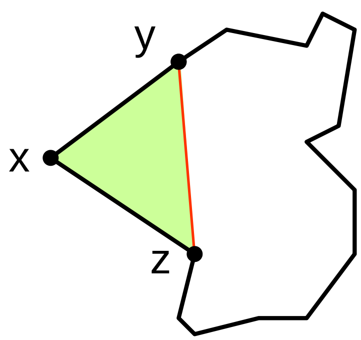


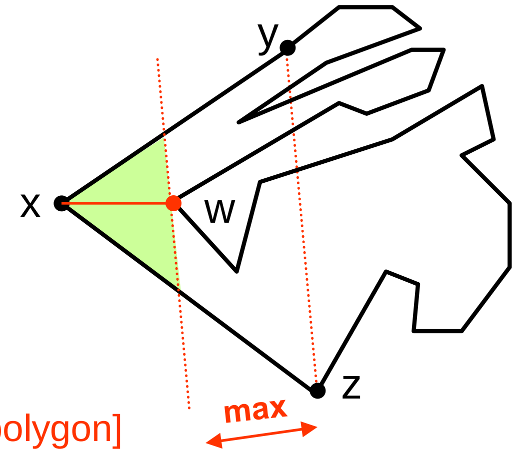
LEMMA 1 Any simple n-gon with n>3 admits at least one diagonal. Such a diagonal can be found in O(n) time.

Proof: Let x be any convex vertex of the polygon (e.g., an extreme vertex, say, the lowest-leftmost).

(case a) $\overline{yz}$ is a diagonal

(case b) $\overline{xw}$ is a diagonal





[Shaded triangle does not contain any vertex of the polygon]

THEOREM 2 Any simple n-gon P admits at least one Triangulation.
Such a triangulation T can be computed in $O(n^2)$ time.
Any such triangulation has n-3 diagonals, and n-2 triangles.

Proof: By induction on n.

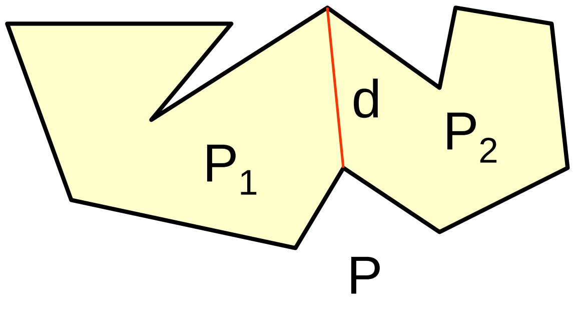
Basis (n=3): Obvious.

Ind. Step (n>3): By previous Lemma, a diagonal d of P exists and can be found in O(n) time, and divides P into simple polygons P_1 & P_2 with, say, n_1 & n_2 vertices, where d is an edge of both. Note, $n = n_1+n_2-2$.

Triangulations T_1 & T_2 of P_1 & P_2 can be obtained recursively.
Now set $T = T_1 \cup T_2$ with d as an extra diagonal.

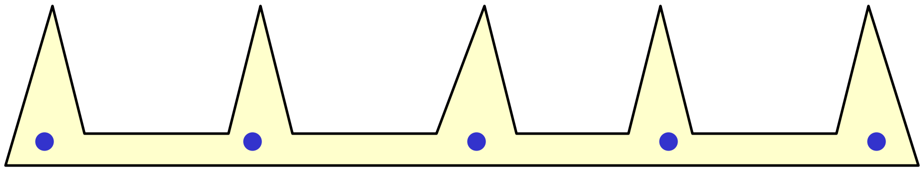
Total computation time: $\text{Time}(n) = \text{Time}(n_1) + \text{Time}(n_2) + O(n) = O(n^2)$.

By induction hypothesis:
T_1 has n_1-3 diagonals and n_1-2 triangles,
T_2 has n_2-3 diagonals and n_2-2 triangles,
These imply:
T has $n-3$ diagonals and $n-2$ triangles.



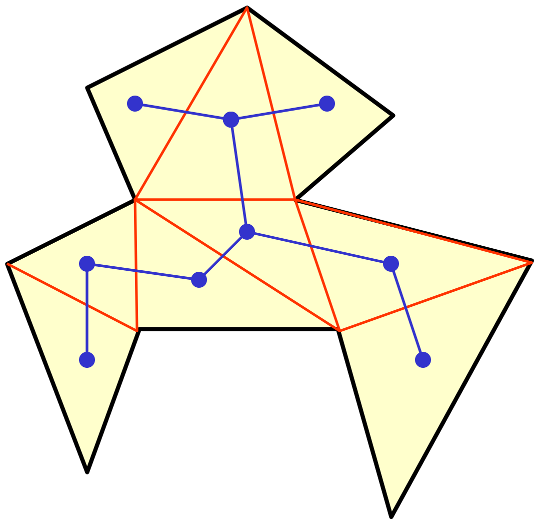
THEOREM 3 [Chvătal 1975, Fisk 1978]
$\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary to guard any simple n-gon.

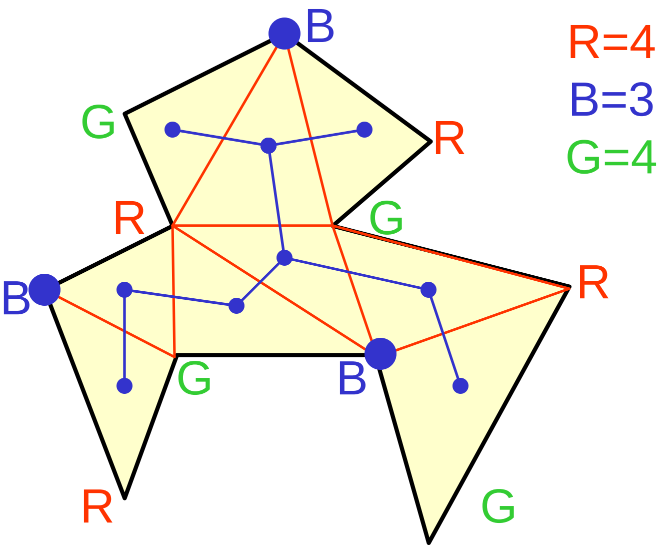
Proof: Necessity:



Sufficiency:

1. T = a triangulation of the n-gon.
2. 3-colour vertices of T (so that the vertices of each triangle get 3 different colours). This can be done (implicitly) by a DFS traversal on the dual tree of T.
3. Choose a colour least often used (break ties arbitrarily).
4. Place a guard at the vertices of the chosen colour. (Each triangle has a guard.)





Simple Polygon Triangulation Algorithms

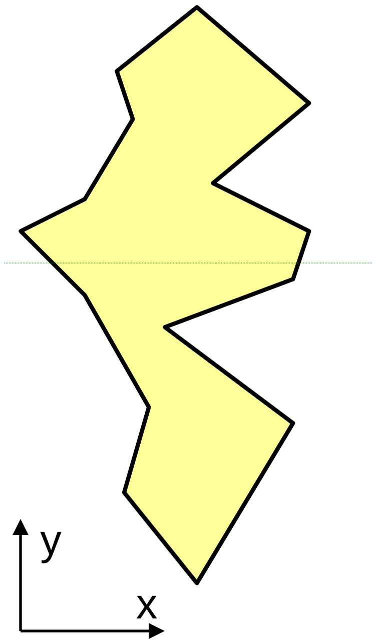
• $O(n^2)$ time	See Theorem 2. Also by “ear removal”, Lennes 1911.
• O(n log n)	Garey-Johnson-Preparata-Tarjan (plane sweep) 1978.
• O(n log log n)	Tarjan-van Wyk (balanced-cut & Jordan-sort) 1986-88.
• O(n log* n) randomized	Clarkson-Tarjan-van Wyk 1989.
• O(n)	Chazelle 1991. [Complicated. Can it be simplified?]
• O(n) randomized	Amato-Goodrich-Ramos 2000. [See LN15]

A possible generic candidate for simpler & efficient polygon triangulation algorithm:

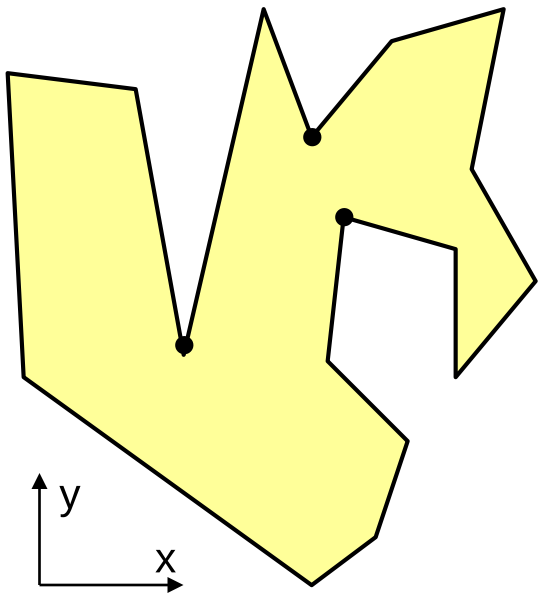
• via pseudo-triangulations	Mirzaian 1988 [See LN14]

Garey-Johnson-Preparata-Tarjan

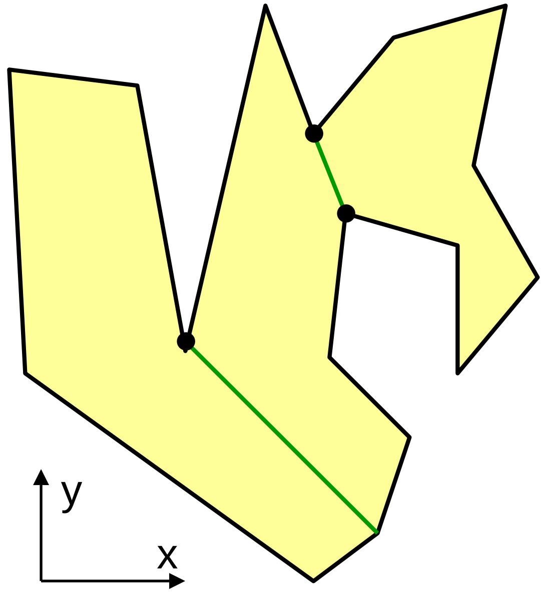
- **FACT:** P is y-monotone if and only if it does not have any cusps.
- A monotone polygon can easily be triangulated in linear time.
- Subdivide the simple polygon into monotone sub-polygons by adding diagonals to cusps.



y-monotone



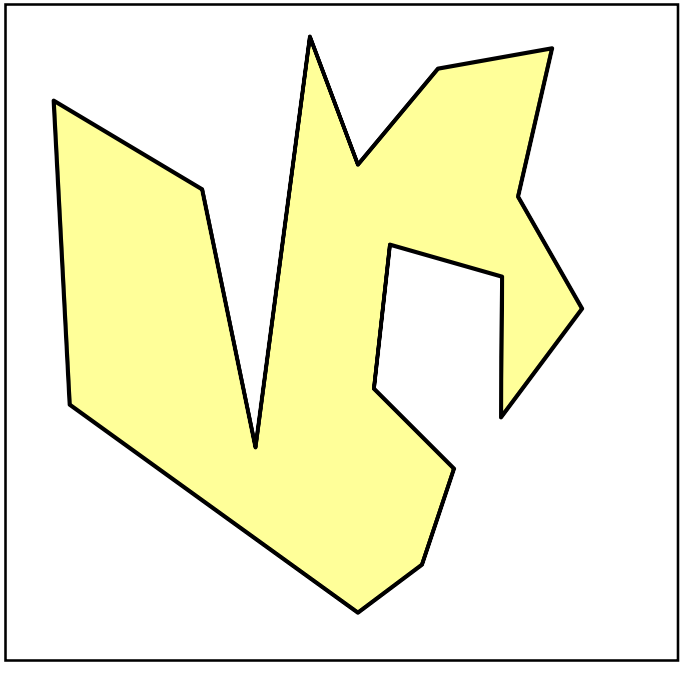
Cusp: concave local y-min or y-max vertex.



3 y-monotone sub-polygons.

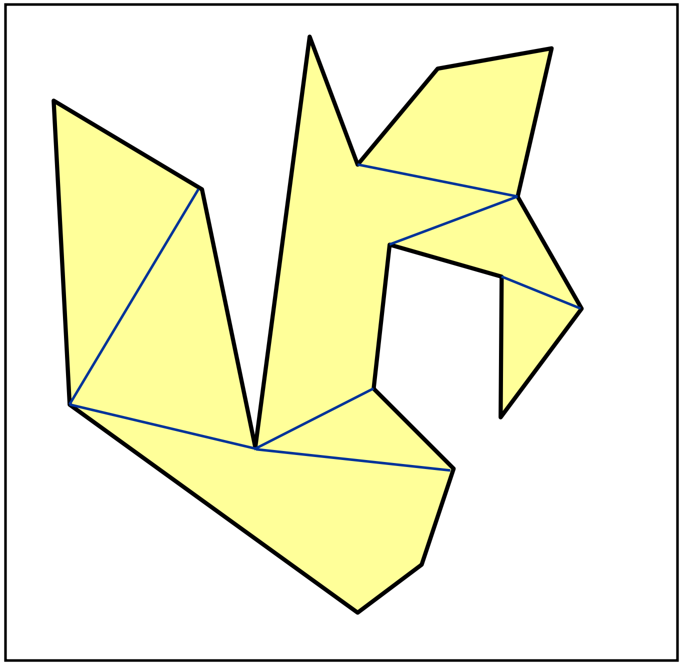
Garey-Johnson-Preparata-Tarjan

How to partition the polygon into monotone sub-polygons by adding suitable diagonals



Garey-Johnson-Preparata-Tarjan

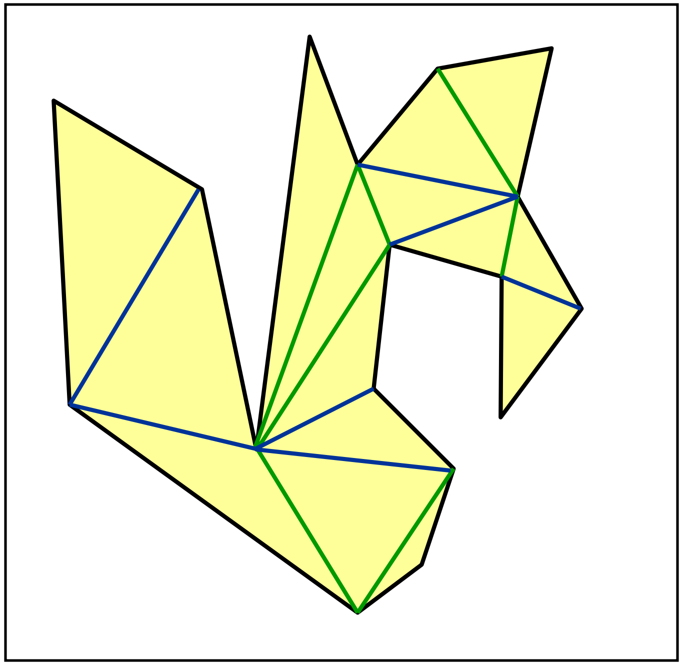
How to partition the polygon into monotone sub-polygons by adding suitable diagonals



1. Trapzoidize using plane sweep in O(n log n) time.
2. Remove visibility chords outside polygon.
3. Add one supporting diagonal (if any) per trapezoid. These diagonals eliminate cusps and subdivide polygon into y-monotone sub-polygons.
4. Ignore visibility chords.

Garey-Johnson-Preparata-Tarjan

How to partition the polygon into monotone sub-polygons by adding suitable diagonals



1. Trapzoidize using plane sweep in O(n log n) time.
2. Remove visibility chords outside polygon.
3. Add one supporting diagonal (if any) per trapezoid. These diagonals eliminate cusps and subdivide polygon into y-monotone sub-polygons.
4. Ignore visibility chords.
5. Triangulate each y-monotone sub-polygon in total O(n) time. [See next slides.]

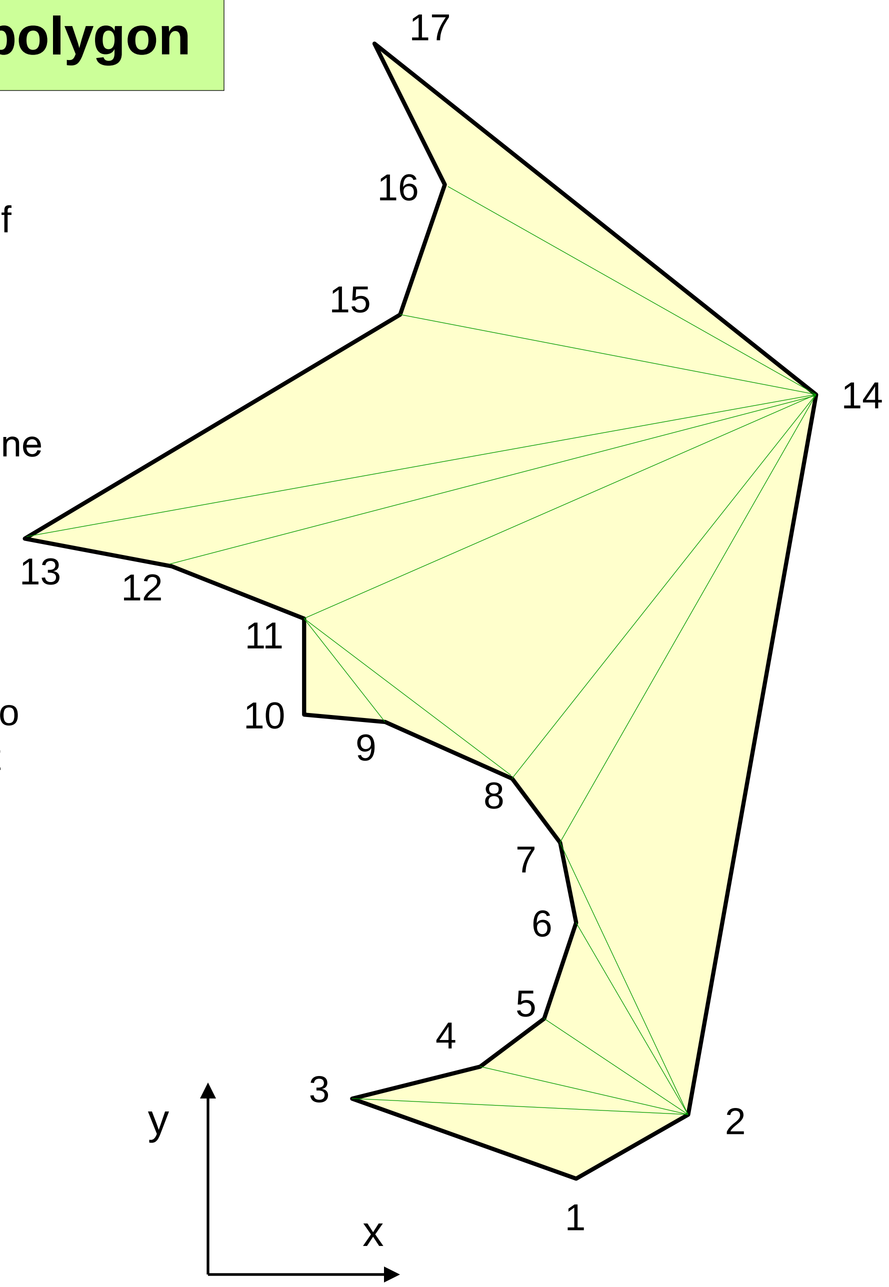
Triangulating a y-monotone polygon

• Merge y-sorted left & right boundary chains of the polygon to obtain the y-sorted vertex list.

• Advance along y-sorted vertex-list:

➢ An “uncapped” chord (edge or diagonal) is one that is considered but doesn’t yet have an incident triangle above. These chords form a concave chain.

➢ For each vertex v in y-sorted order, add downward visible chords and triangles from v to uncapped visible diagonals, starting from most recent & backwards. (Use a stack. See next slide.)



Algorithm Triangulate y-monotone polygon P

- merge the vertices of the left and right chains of P into y-sorted order, say, $u_1, u_2, \dots, u_n$.
- push u_1 and u_2 into an initially empty stack S.
- **for** $j \leftarrow 3 \,..\, n-1$ **do**

  if u_j & top(S) are on different chains

  then pop all vertices from S and add a diagonal between u_j and each popped vertex except the last.
  push u_{j-1} and u_j onto S.

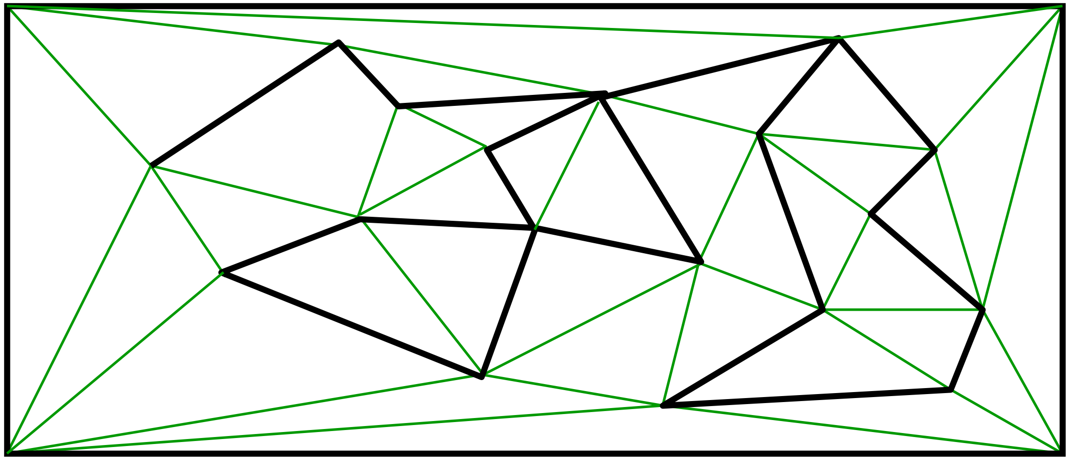
  else pop(S)
  pop the other vertices from S while they are visible from u_j, and add a diagonal between u_j and each popped vertex.
  push last popped vertex back onto S.
  push u_j onto S.

- add diagonals from the last vertex u_n to all stack vertices except first and last.

end

A straight-line planar subdivision with n vertices can be triangulated in O(n log n) time and O(n) space.

Use the same approach: plane-sweep; trapzoidize; monotonize; and triangulate the resulting monotone polygons.



This approach can also triangulate a polygon with polygonal obstacles inside, in O(n log n) time (applications in robotics).