# Graph Theory Problem: How to Guard A Museum

T. Truong

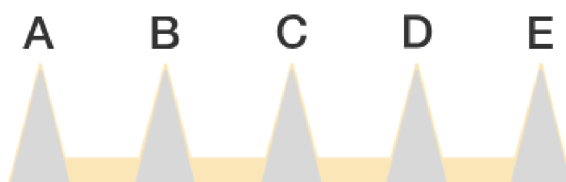
March 2024

From Proofs from THE BOOK by Martin Aigner and Günter M. Ziegler

In this activity, we will explore the art gallery problem, presented by Chv´atal in 1975, which aims to find the smallest number of guards required to cover an art gallery shaped as a polygon of n vertices.

**The Art Gallery Theorem: For surveilling a simple polygon with n vertices, $[\frac{n}{3}]$ guards are sometimes necessary and always sufficient.**

Consider this museum with 15 walls in the shape of a spiked comb: Since these triangles do not over-




lap, at least 5 guards are needed. But by the Art Gallery Theorem, $[15/3 = 5]$ guards are also sufficient. So in general, the comb museum layout gives an example of a museum with 3n walls that requires exactly $[3n/3] = n$ guards which shows that the bound in the theorem is best possible. This will also give us an idea of where we are headed in this proof.

Problem: In order to prevent thieves from stealing our museum's priceless masterpieces, we the managers will have to find the smallest number of guards required to cover every point of the museum at all times.

(1) You are each given a simple polygon (i.e., no holes, no self-interaction) to work with. This will be your museum.

(2) Place a guard anywhere in your museum by denoting it with a dot and naming it any letter you want.

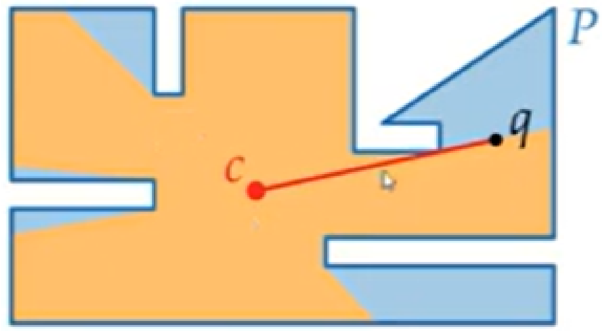
(3) From a bird's-eye-view of the museum, we can observe that the region your guard covers is a star-shaped region.

Definition: star-shape domain: a star-shaped domain is a subset U of an appropriate ambient space for which there exists a base-point that can be connected to every other point by a line segment in U.

(4) Before we start solving this problem, let's define visibility: At point $q \in P$ is visible from $c \in P$ if $\overline{qc} \subseteq P$

(5) Now, let's start filling up our museum with guards. We can start by partitioning our museum into convex polygons...then for each of them we can pick any point and place a guard there to see everything!

(6) Discussion: Having tried that, do you think that we can break this polygon up into even smaller pieces?

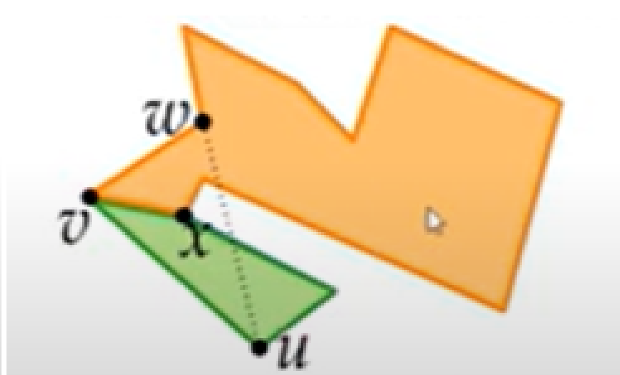



(7) Triangulation: decomposition of the polygon into triangles by drawing non-intersecting diagonals between pairs of vertices

**Theorem 1**:
(1)Every simple polygon can be triangulated.
(2) Any triangulation of a simple polygon with n vertices consists of (n-2) triangles.

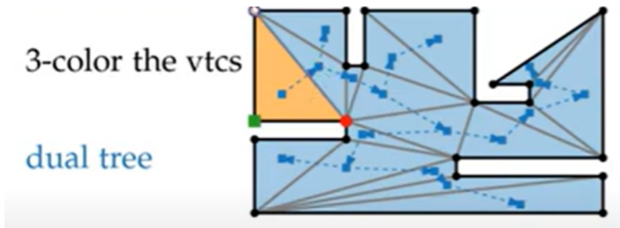
**Based on what we are trying to prove (as stated above), let's start with the base case. What if n=3? How many triangles do we have?**

We want to assume the theorem for all polygons with (n-1) vertices:
3,..,(n-1) $\rightarrow$ n
Since the sum of the interior angles of P is (n−2)*180, there is a vertex of P with interior angle less than 180. Let u,w be the neighboring vertices of u in the polygon. If the line segment u,w lies within the polygon, then this line segment is the desired diagonal. If the segment intersects the boundary, we pick the corner that is furthest from uw.


Via induction, what do we have now?
3 vtcs $\rightarrow$ 1 triangle
**(n-1) vtcs $\rightarrow$ (how many triangles?)**

**Theorem 2**: Any triangulated polygon is 3-colorable.
For $n = 3$, the polygon is a triangle and we can choose 3 different colors for the 3 vertices. Remember no 2 vertices that share an edge can have the same color.




To get the 3-coloring of the graph, we use the dual of the graph (basic idea: place a vertex in each face and connect the vertices if they share an edge). So, how do we get $[n/3]$ from this?...Let's do it!

**Draw the dual graph on your polygon and color the vertices on your polygon using the 3-Color Theorem.**

**Supplementary**: We have been told that every simple polygon can be triangulated but how would we physically do it? With an algorithm!

Using recursion running time algorithm that works something like this (just a general idea): Given n vertex polygon → partition into "nice" (convex) piecesn n' vtcs → n" triangles: 1) For each vertex in the polygon, compute the angle between the two linked edges
2) Sort vertices by decreasing angle relative to the interior of the polygon
3) If there is less than 3 vertices in the set, we're done
4) Take the last vertex in the set and output the triangle formed by it and its two neighbours
5) Remove the vertex from the set
6) Update the angle of the two neighbours
7) Jump to 2
We find these "nice" convex pieces with an order of nlog(n) times.

See link for further details: https://github.com/AbdShak/2D-Polygon-Triangulation