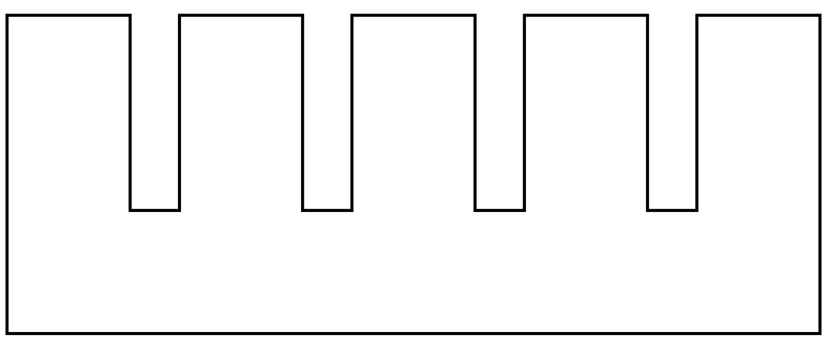
Problem 1

Prove that any polygon admits a triangulation, even if it has holes. Can you say anything about the number of triangles in the triangulation?

> Even if a given polygon has holes, the algorithm that splits a polygon apart into monotone polygons still works regardless. Therefore a polygon with holes can be decomposed into monotone parts and those are guaranteed to admit a triangulation, hence the original polygon with holes must have a triangulation.
>
> Now consider a polygon with k holes. These holes can be removed by introducing cuts/diagonals between holes and the outer boundary. Only 1 cut is needed per hole and each cut introduces 2 new "vertices" to the overall polygon. Therefore the final simple polygon made after all the cuts has $n + 2k$ vertices, meaning a triangulation of it must then have $n + 2k - 2$ triangles.

Problem 2

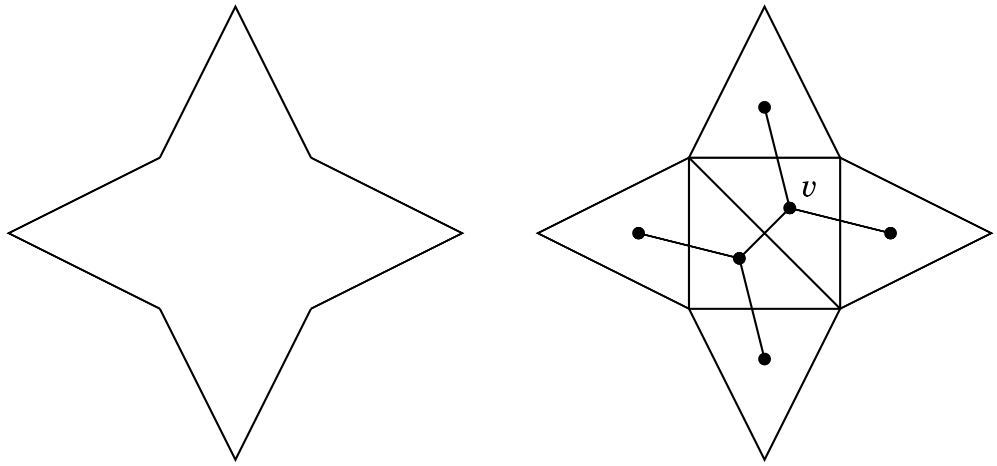
A *rectilinear polygon* is a simple polygon of which all edges are horizontal or vertical. Let $\mathcal{P}$ be a rectilinear polygon with n vertices. Give an example to show that $\lfloor n/4 \rfloor$ cameras are sometimes necessary to guard it.

> Similar to the pronged polygon to show that in general $\lfloor n/3 \rfloor$ cameras are needed sometimes, this can be extended to rectilinear polygons by using two vertices for the prongs. Pictorally,
>
> 
>
> There are $4p$ vertices where p is the number of prongs, and each one needs its own camera to guard it meaning $p = 4p/4 = n/4$ cameras are needed.

Problem 3

Prove or disprove: The dual graph of the triangulation of a monotone polygon is always a chain, that is, any node in this graph has degree at most two.

The statement is false. Consider the following x-monotone polygon and the dual graph of one of its triangulations



The marked node v has degree 3. ■

Problem 4

Give the pseudo-code of the algorithm to compute a 3-coloring of a triangulated simple polygon. The algorithm should run in linear time.

The proof that any triangulation of a simple polygon can be 3 colored provides the structure on how to algorithmically color it.

Algorithm 1 Triangulation 3-Coloring

1. Construct the dual graph G from the given triangulation
2. Pick some face f from the triangulation
3. Color each vertex of f with a different color
4. Perform a DFS starting at f over G
    - Each new face will be adjacent to a colored face, hence two vertices are already colored
    - Color the remaining vertex with the only possible color

As for time complexity:

1. Constructing the dual graph takes $O(n)$ time. Alternatively one can employ an existing DCEL to traverse the polygon, but the upper bound in either case for making the representation is $O(n)$.

2. Picking some face from the triangulation/dual graph can be done in $O(1)$.
3. The number of nodes in the dual graph is $O(n)$ meaning the DFS traversal is also $O(n)$.

Overall then the running time of the algorithm is $O(n)$.

Problem 5

Can the algorithm of this chapter also be used to triangulate a set of n points? If so, explain how to do this efficiently.

The key idea is that we can efficiently find the convex hull of the points and split it into 2 monotone polygons, of which can be triangulated in linear time. The convex hull can be found in $O(n \log n)$ time. Consider then the set of vertices in the convex hull but not on its boundary. These can be sorted such that they are increasing montonically with respect to their x-coordinate in $O(n \log n)$ time as well. Therefore the convex hull can be split into monotone halves by connecting these inner vertices where there are edges between adjacent x coordinate vertices. Each half then can be triangulated in $O(n)$ time. Hence in total the set of n point can be triangulated in $O(2n \log n + 2n) = O(n \log n)$ time.