COMP5045 – Computational Geometry

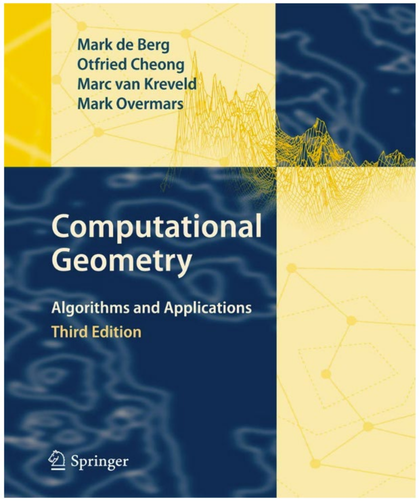
Course page: Ed

Lecturer: Joachim Gudmundsson
Head of School office
Level 2, School of CS (J12)
joachim.gudmundsson@sydney.edu.au
Ph. 9351 4494

Tutor: Lindsey Deryckere

Course book:

M. de Berg, O. Cheong, M. van Kreveld and M. Overmars
Computational Geometry: Algorithms and Application
Springer-Verlag (2nd or 3rd edition)



Outline:

13 lectures Thursdays 2-4pm
4 assignments (~4 questions/assignment)
Written exam [40% required to pass]

Tutorials:

Thursdays 4-5pm in Civil Eng Seminar Room 304
Fridays 4-5pm in Mechanical Eng Seminar Room 2 S226

Assessment:

Each assignment 17.5% (total 70%)
Written exam 30%

Handing in late is not accepted = 0 points
Assignments submitted via Canvas as pdf (no handwriting!)

Collaboration:

General ideas – Yes!
Formulation and writing – No!
Read Academic Dishonesty and Plagiarism.

Preliminary schedule

› Lecture 1:	The Art gallery problem
› Lecture 2:	Sweepline technique: triangulation & segment intersection
› Lecture 3:	Sweepline technique: Convex hulls
› Lecture 4:	Linear programming and probabilistic analysis
› Lecture 5:	Orthogonal range searching I: kd-trees and range trees
› Lecture 6:	Orthogonal range searching II: fractional cascading and interval trees
› Lecture 7:	Voronoi diagrams and Delaunay triangulation
› Lecture 8:	Arrangements and duality
› Lecture 9:	Planar point location
› Lecture 10:	Approximation Algorithms: Applications of the WSPD
› Lecture 11:	Curve similarity
› Lecture 12:	TBA
› Lecture 13:	Recap

Assumed Knowledge

- Proof techniques:
  - Proof of contradiction, Induction proof…
  - See also Sections 1.2-3 in "[Introduction to Theory of Computations]"
    [Maheshwari and Smid]
- Basic data structures
  - Binary search trees, lists, queues, stacks, graphs…
- Algorithms
  - Techniques: greedy and D&C
  - Sorting, searching, depth-first search, breadth-first search…
- big-O/big-Omega notations, basic analysis, recursion…

Brief History: Algorithms and CG

Algorithms:

300 BC: Euclid designed an algorithm for GCD.
780 AD: al-Khowarizmi – the word "Algorithm" is derived from his name.
…
Babbage, Cantor, Hilbert, Church, Gödel
…
1936: Turing developed a machine that provides a formal and rigorous definition of an algorithm

Computational Geometry:

1644: Descartes wrote about Voronoi diagrams
1759: Euler & Vandermonde discussed Euclidean TSP
1978: Shamos wrote his thesis which defines the start of modern CG
1985: Preparata & Shamos wrote the first CG textbook

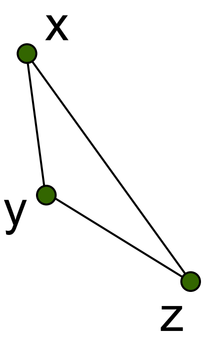
What is computational geometry?

The study of algorithms to solve problems stated in terms of geometry.

The problems we study are defined in a metric space!

For every two points x and y in the metric space, there is a function $g(x,y) \geq 0$ which gives the distance between them as a nonnegative real number. A metric space must also satisfy

1. $g(x,y) = 0$ iff $x = y$,
2. $g(x,y) = g(y,x)$, and
3. the triangle inequality must hold $g(x,y) + g(y,z) \geq g(x,z)$.

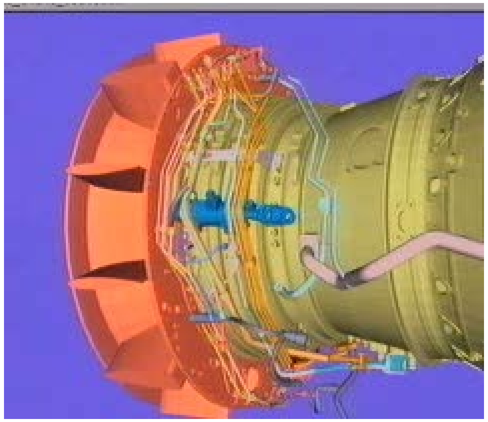


We will mainly consider the Euclidean metric (L_2-metric).

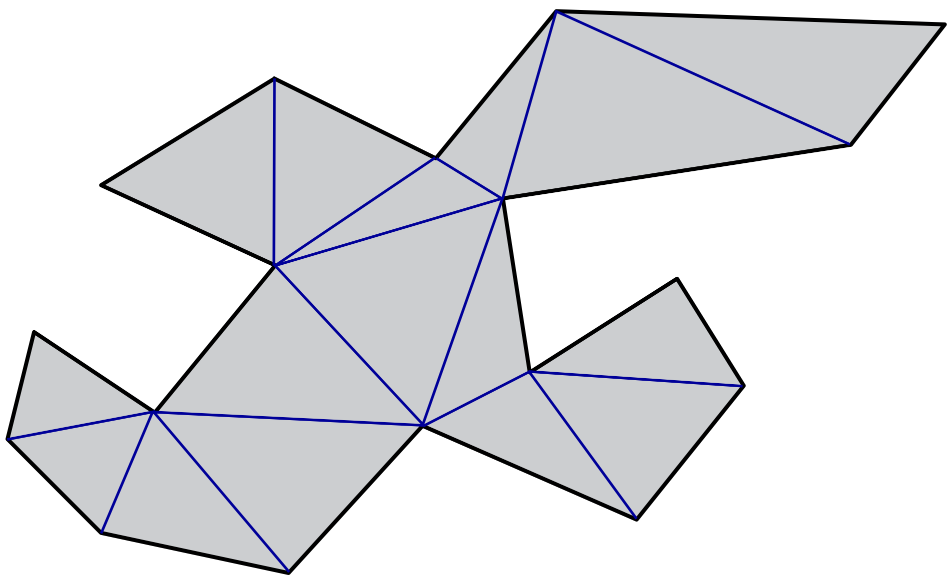
Why computational geometry?

Certain problems are inherently geometric:
- Point location
- Range searching
- Motion planning
- …

Other problems are just much easier to solve if we use the underlying metric:
- Travelling Salesman Problem
- Nearest neighbour
- …



Polygon Triangulation and The Art Gallery Problem



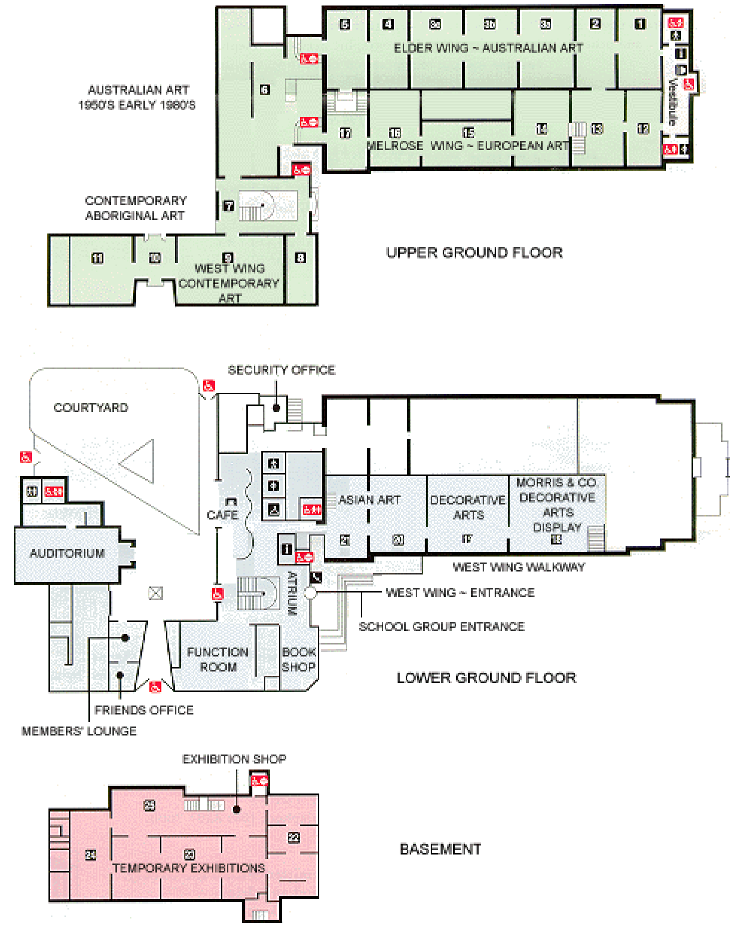
THE UNIVERSITY OF SYDNEY

The Art Gallery problem

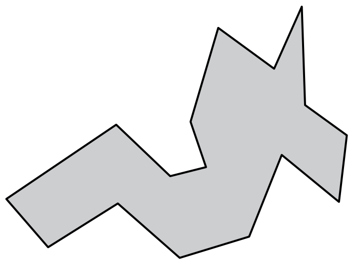
Question:

How many guards are needed to guard an art gallery?

Victor Klee posed this problem to Václav Chvátal in 1973.

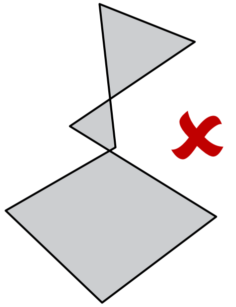


Art Gallery - definitions

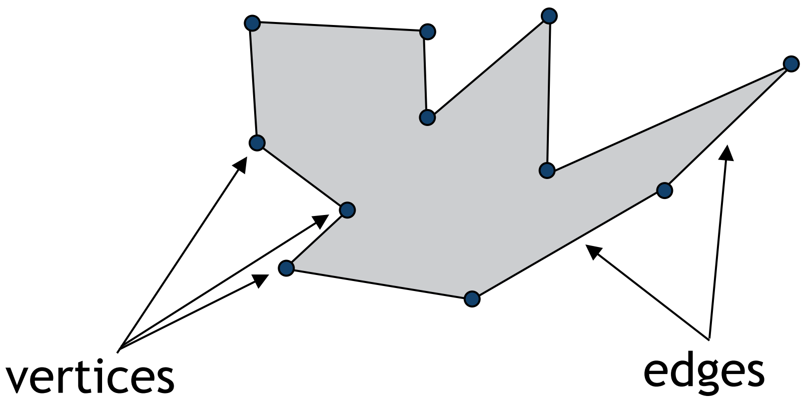


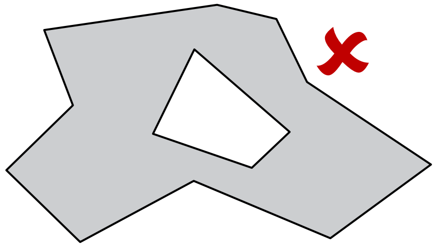
Input: An Art Gallery =
A simple polygon with n line segments

Polygon: A region of the plane bounded by a set of straight line segments forming a closed curve.



Simple: A polygon which does not self-intersect and doesn't have any holes.

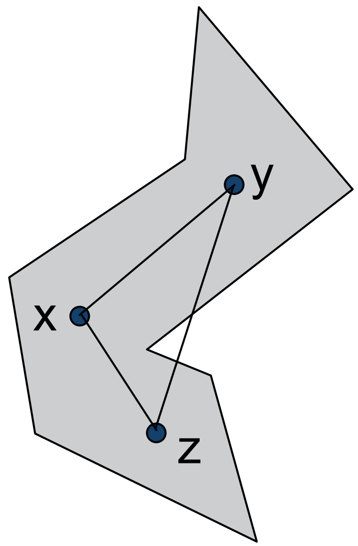




Art Gallery

Guard: (camera, motion sensors, fire detectors, …)
- 2π range visibility
- infinite distance
- cannot see through walls
- cannot move

x can see y iff – $(x,y) \subseteq P$



Art Gallery

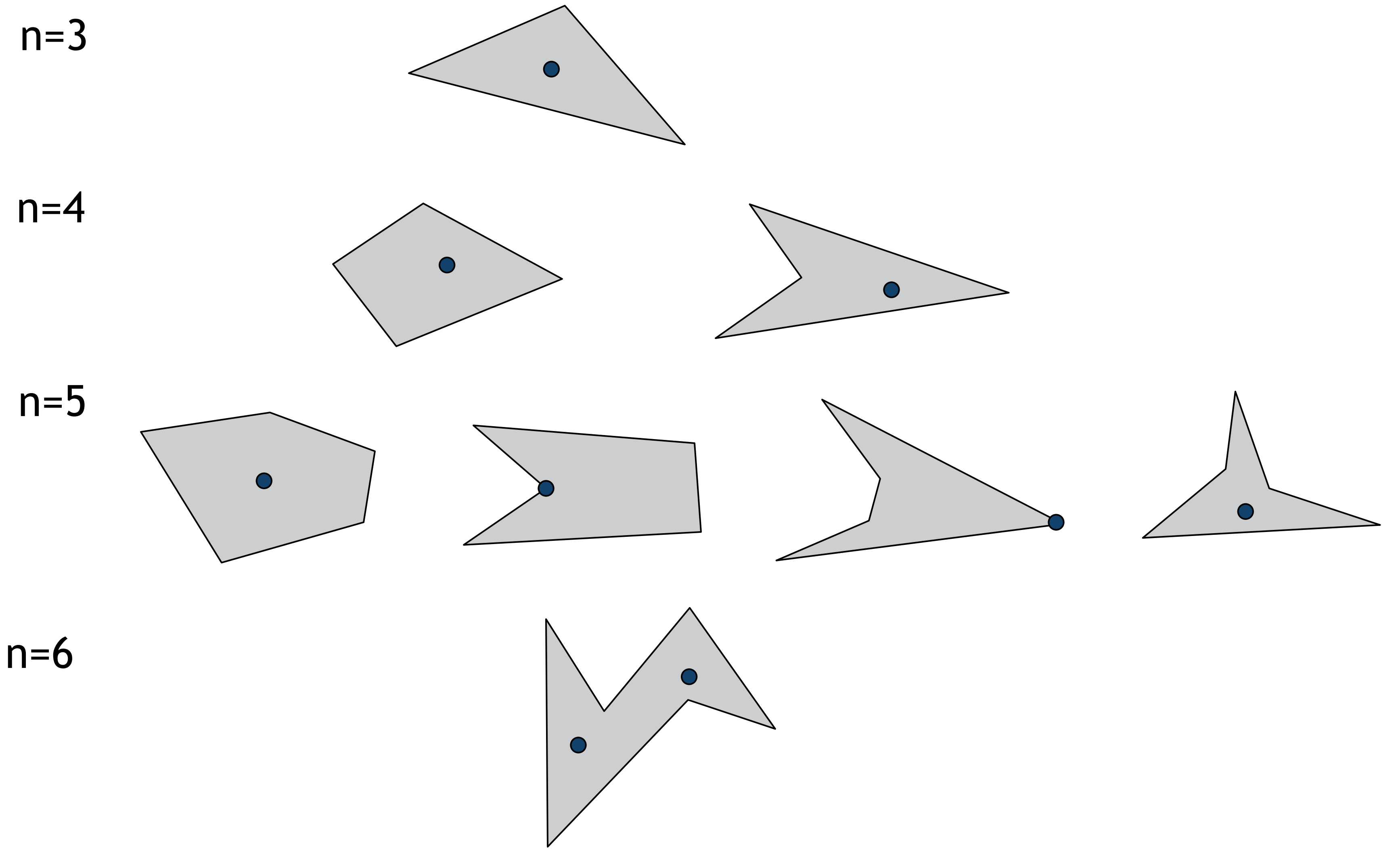
Question: How many guards are needed to guard an art gallery?

n=6

G(n) = the smallest number of guards that suffice to guard any polygon with n vertices.

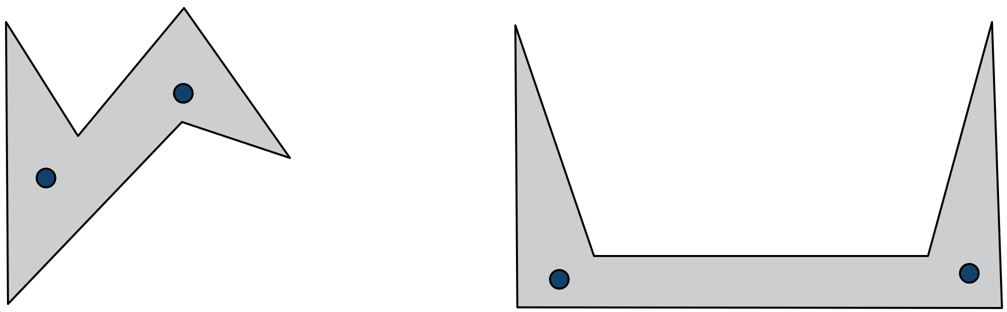
Is $G(n) \leq n$? If we place one guard on each vertex?

A lower bound

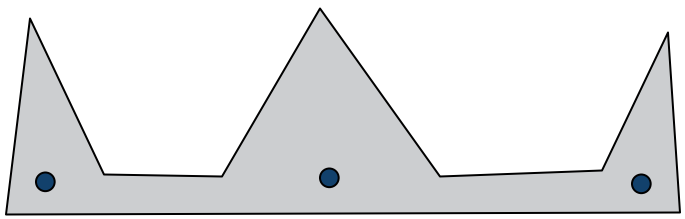


A lower bound

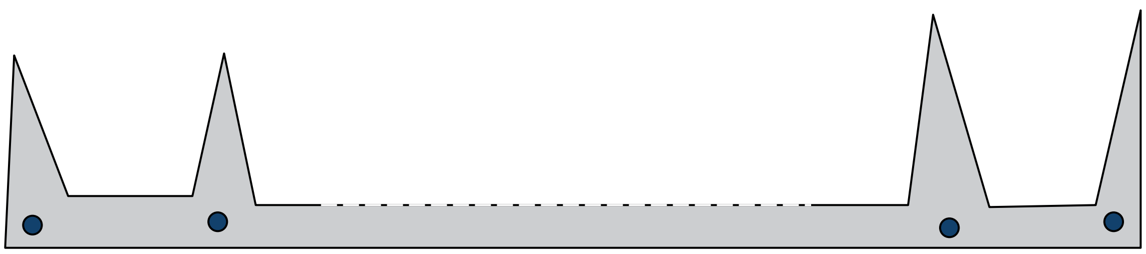
n=6



n=9



n=3k



A conjecture

We have shown: $G(n) \geq \lfloor n/3 \rfloor$

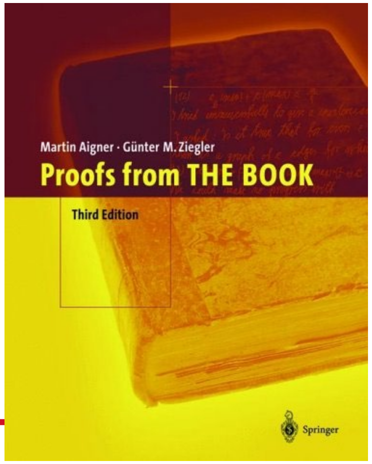
Conjecture: $G(n) = \lfloor n/3 \rfloor$

Proved by Chvátal in 1975.

“Steve Fisk learned of Klee's question from Chvátal's article, but found the proof unappealing. He continued thinking about the problem and came up with a solution while dozing off on a bus travelling somewhere in Afghanistan.”

An elegant proof was given by Fisk in 1978.

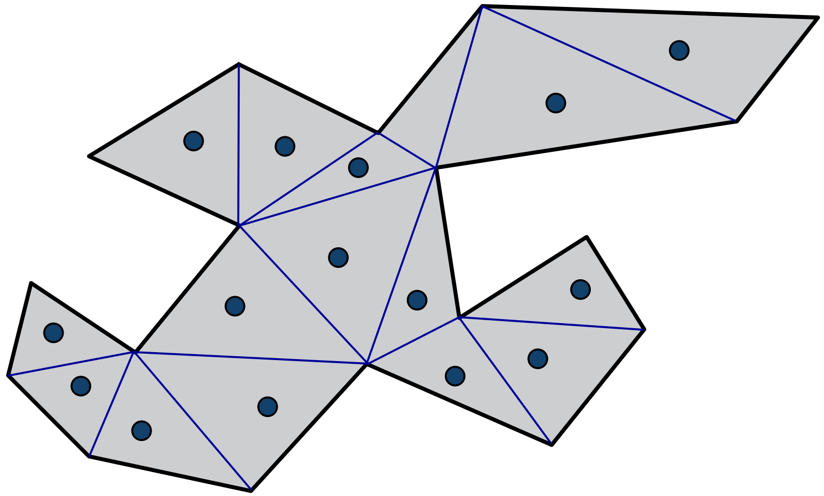
Included in “Proofs from the book”.



A first upper bound

Prove that $G(n) \leq n-2$.

- Decompose P into pieces that are easy to guard. For example triangles!
- How can we use a triangulation of P to place a set of guards?



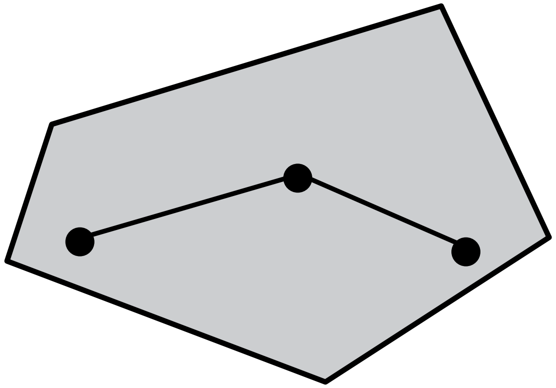
#guards = #triangles

Guard triangle

Why is a triangle easy to guard?

A triangle is convex.

Definition of convex set:
An object is convex if for every pair of points within the object, every point on the straight line segment that joins them is also within the object.



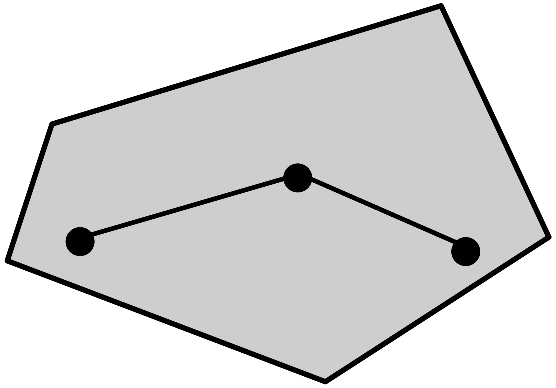
Direct proof

Why is a triangle easy to guard?

A triangle is convex.

Definition of convex set:
An object is convex if for every pair of points p and q within the object, p can see q.

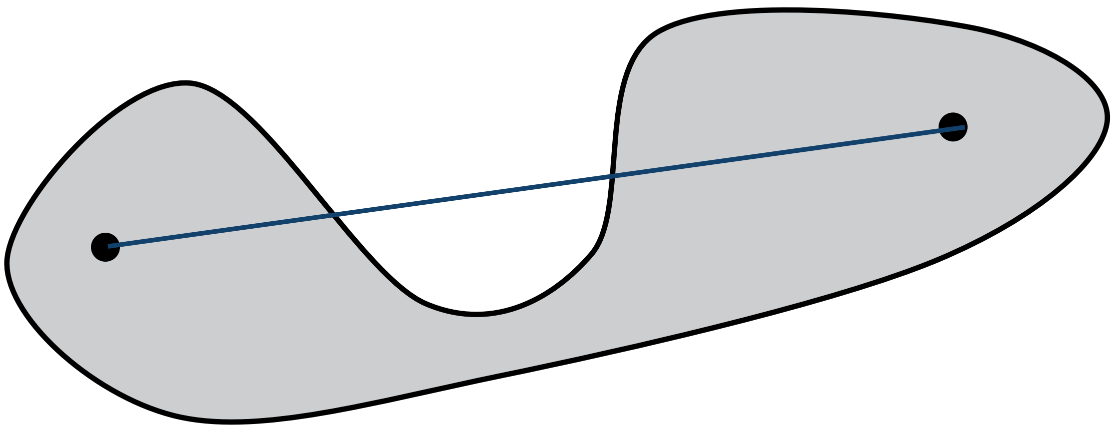
Every convex set can be guarded by one guard.



Proof by contradiction

Assume the opposite!

There exists two points within the triangle that cannot see each other.



But that contradicts the definition of a convex set
$\Rightarrow$ Every pair of points must see each other!

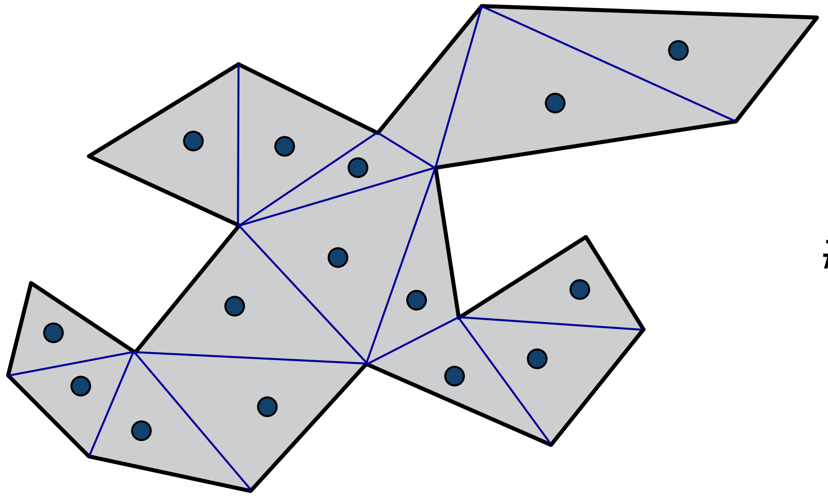
QED

A first upper bound

Prove that $G(n) \leq n-2$.

- Decompose P into pieces that are easy to guard.
  For example triangles!

- How can we use a triangulation of P to place a set of guards?

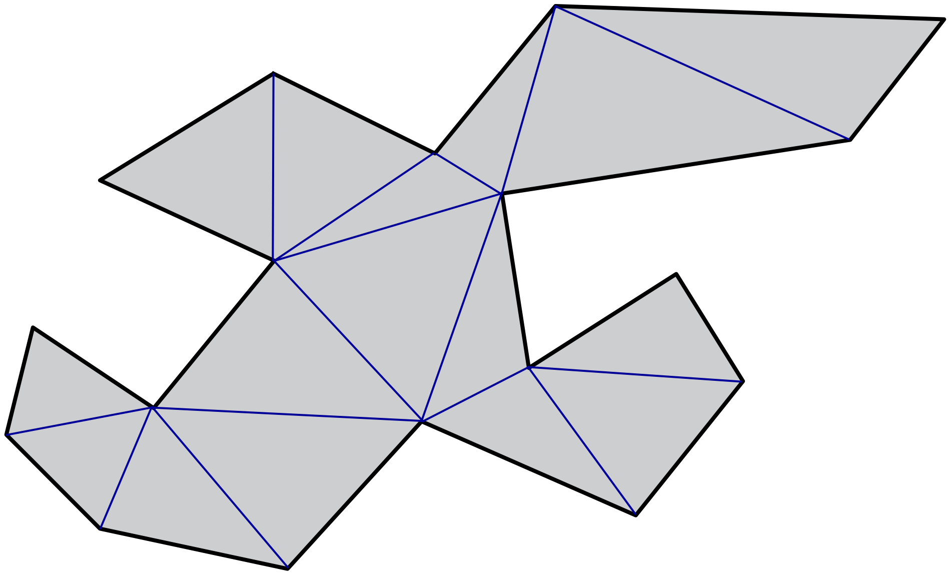


#guards = #triangles

Triangulation

A triangulation can be obtained by adding a maximal number of non-intersecting diagonals within P.

A diagonal of P is a line segment between two vertices of P that are visible to each other.



BUT!

1. Does a triangulation always exist?
2. What is the number of triangles?

Number of triangles

Does a triangulation always exist?

Is there always a diagonal?

Lemma: Every simple polygon with >3 vertices has a diagonal.

Constructive proof

Does a triangulation always exist?

Is there always a diagonal?

Lemma: Every simple polygon with >3 vertices has a diagonal.

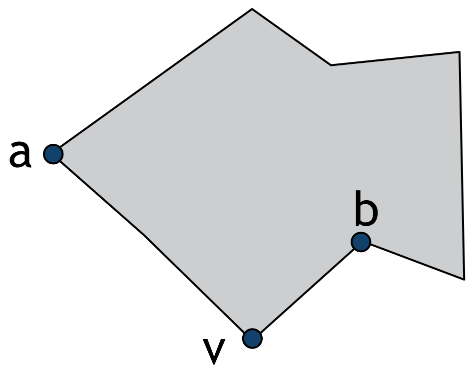
A **constructive proof** is a proof that demonstrates the existence of a mathematical object by creating such an object.

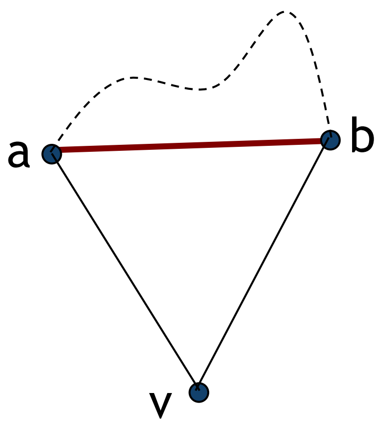
Constructive proof

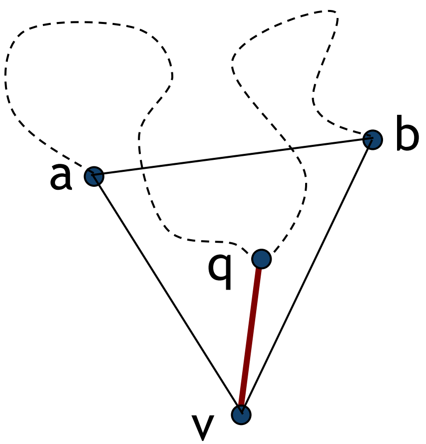
Does a triangulation always exist?

Is there always a diagonal?

Lemma: Every simple polygon with >3 vertices has a diagonal.







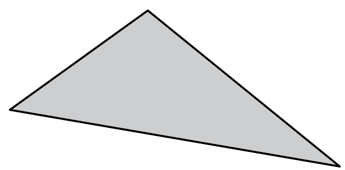
QED

Number of triangles

Theorem: Every simple polygon admits a triangulation.

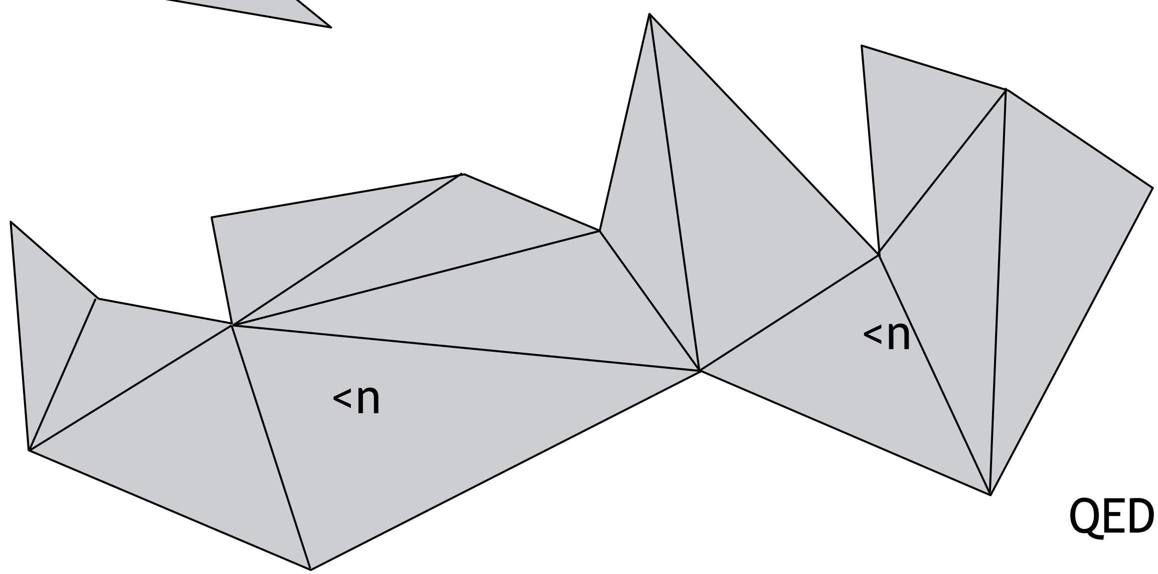
Proof by induction (over the number of vertices):

Base case:
$n=3$



Induction hyp.:
$n<m$

Induction step:
$n=m$

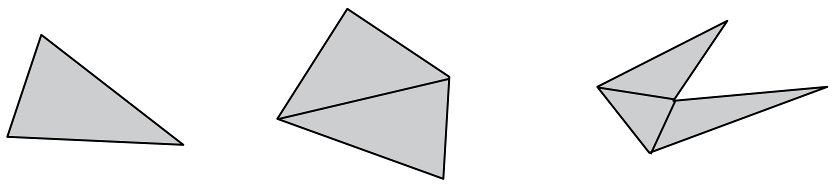


QED

Number of triangles

Theorem: Every triangulation of P of n vertices consists of x triangles.

What's x?

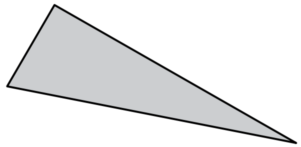


Conjecture: $x = n-2$

Number of triangles

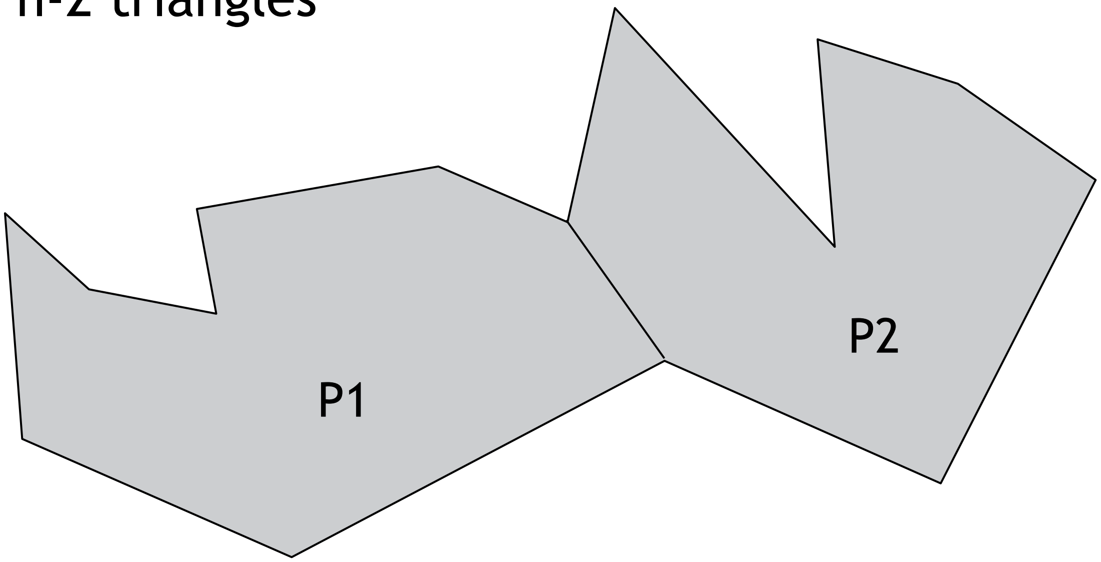
Proof by induction:

Base case (n = 3):

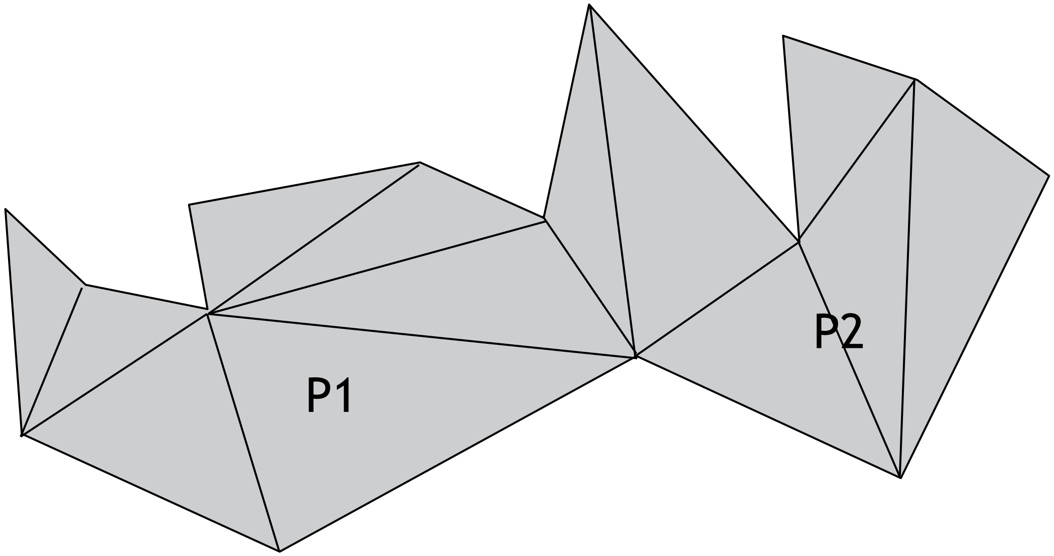


Ind. hyp. (n < m): n-2 triangles

Ind. step (n=m):



Number of triangles



|P1| = m1 < n
|P2| = m2 < n

m1+m2 = n + 2

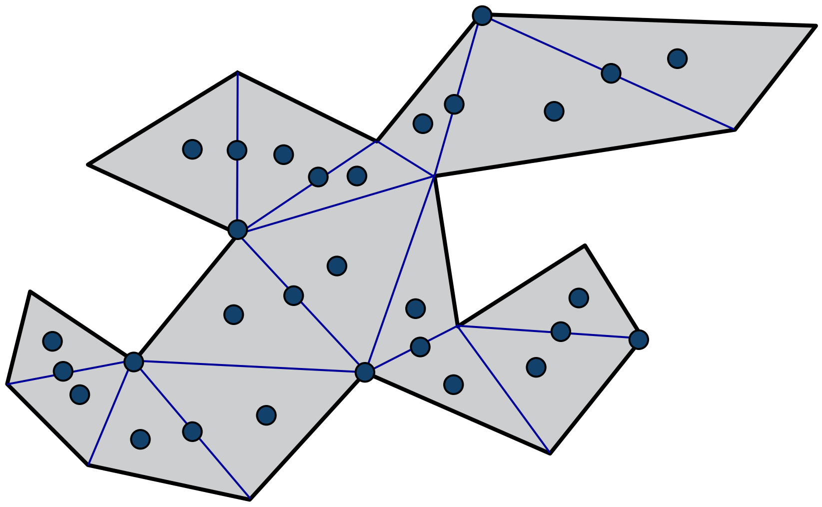
According to ind. hyp.
#triangles in P1 is m1-2
#triangles in P2 is m2-2

(m1-2) + (m2-2) = m1 + m2 - 4 = n – 2 QED

Theorem: Every triangulation of P consists of n-2 triangles.

THE UNIVERSITY OF SYDNEY

Back to the Art Gallery



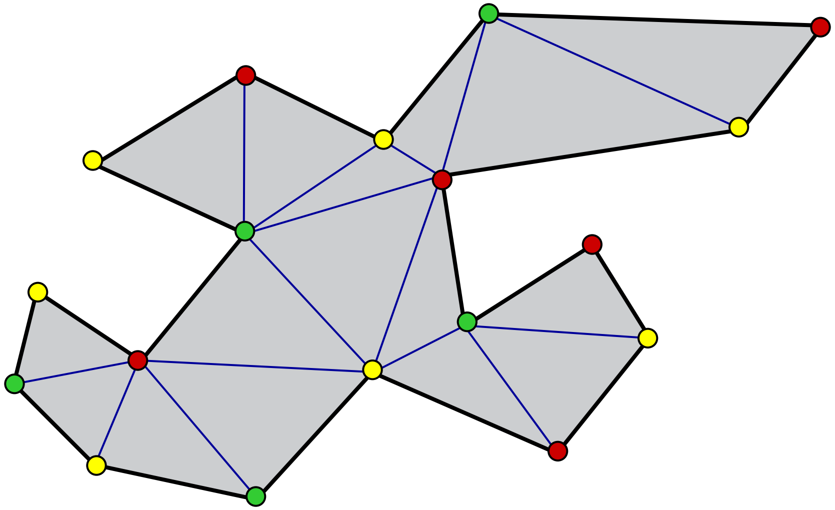
$G(n) \geq \lfloor n/3 \rfloor$

$G(n) \leq$ #guards = #triangles = n-2

How do we place the guards?

3-colouring

Idea: Assign a colour to each vertex such that no two adjacent vertices have the same colour.



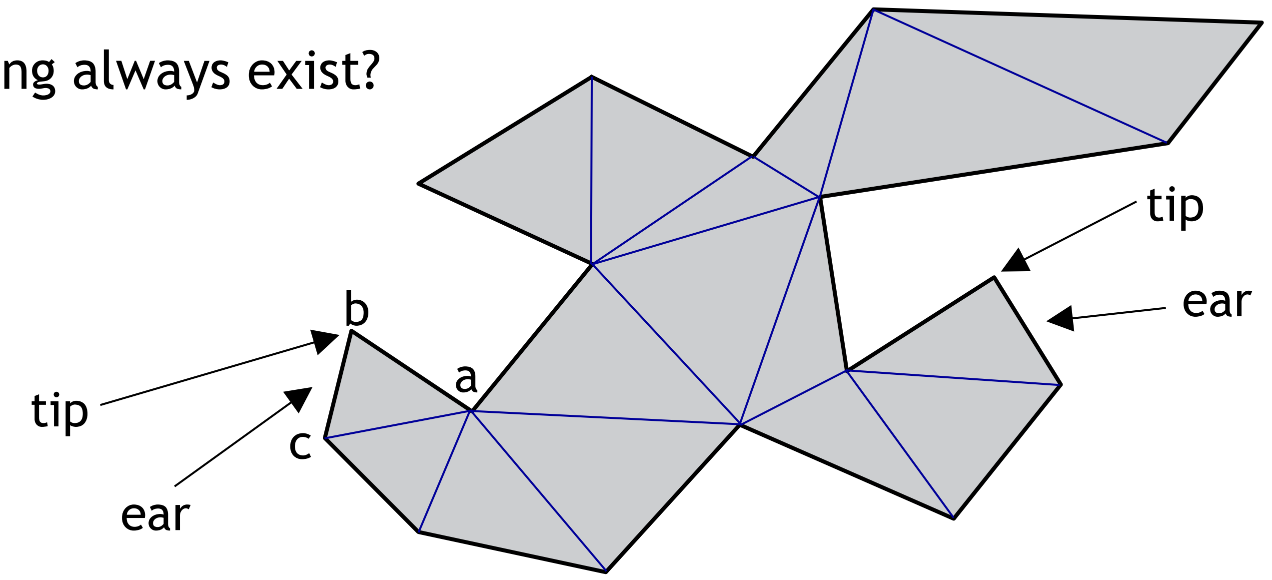
#yellow = 7
#green = 5
#red = 6

Place guards on the green vertices.

$\Rightarrow$ #guards $\leq \lfloor n/3 \rfloor$ Why?

3-colouring

Does a 3-colouring always exist?

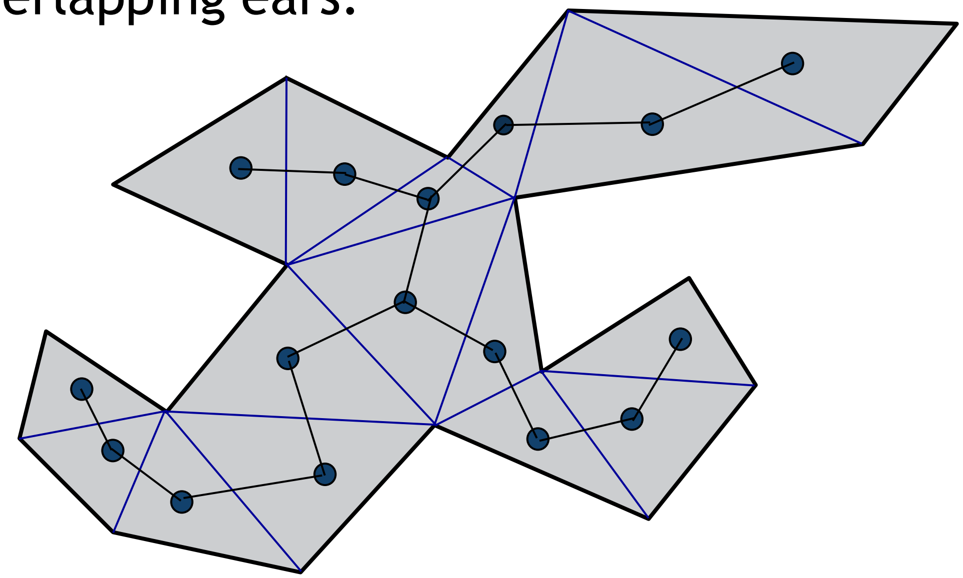


Definition: Three consecutive vertices a, b and c of P form an ear of P if ac is a diagonal of P, where b is the ear tip.

3-colouring

Theorem: Every polygon with n>3 vertices has at least two non-overlapping ears.

Consider the dual D(T) of the triangulation T.



D(T) is a (binary) tree. Why?

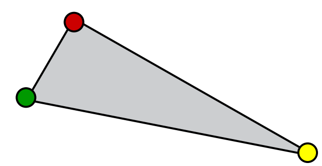
Every tree with at least two nodes has at least two vertices of degree 1 $\Rightarrow$ T has at least two ears.

3-colouring

Theorem: The triangulation of a simple polygon can always be 3-coloured.

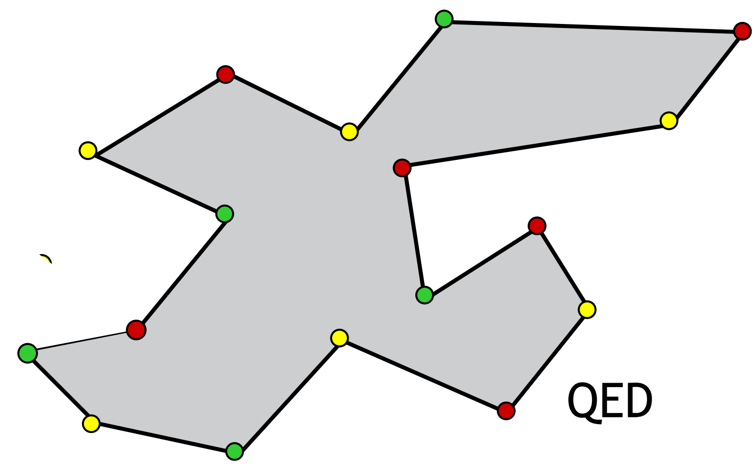
Proof by induction:

Base case (n=3):



Ind. hyp. (n<m):

Ind. step (n=m): Polygon has an ear.

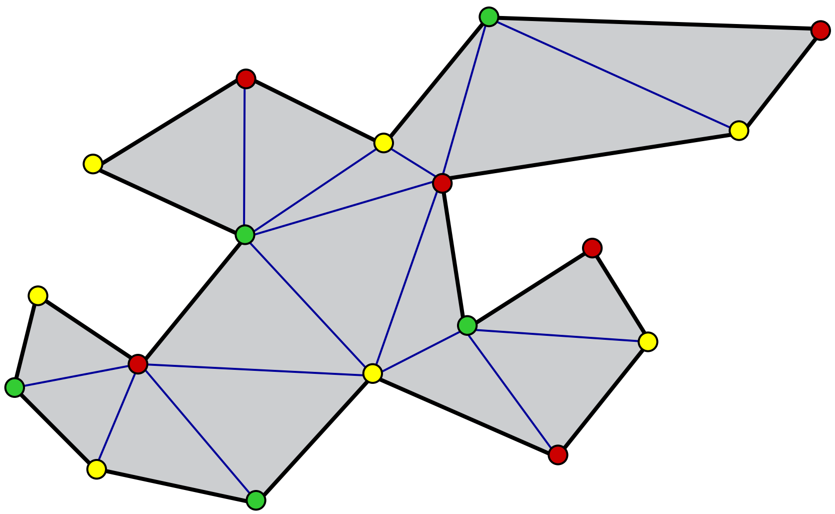


QED

Main results

Theorem:

1. Every simple polygon can be triangulated.
2. The triangulation of a simple polygon can be 3-coloured.
3. Every simple polygon with n vertices can be guarded with $\lfloor n/3 \rfloor$ guards.



A triangulation exists but how can we compute it?

Questions

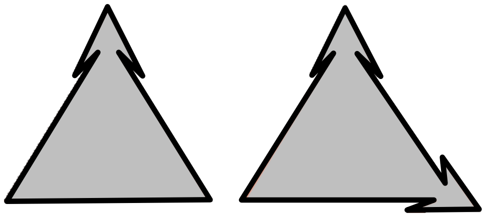
1. Construct a simple polygon P and a placement of guards such that the guards see every point of the perimeter of P, but there is at least one point interior to P not seen by any guard.

2. Construct a polygon P and a watchman route of a guard such that the guard sees the perimeter of P but there is at least one point interior to P not seen.

3. Open problem

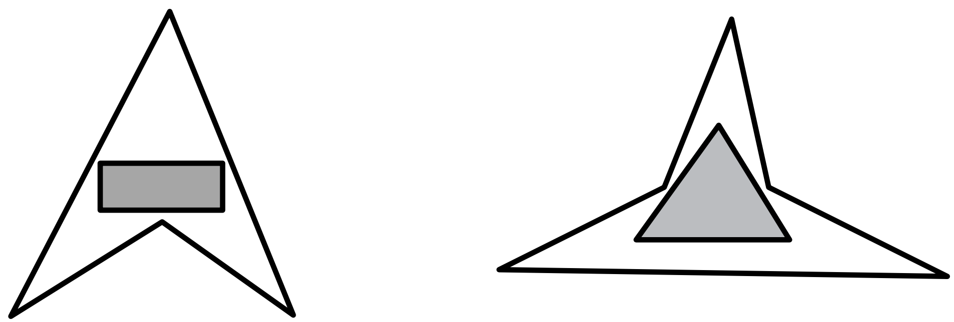
   Conjecture by Toussaint'81:

   Except for a few polygons, $\lfloor n/4 \rfloor$ edge guards are always sufficient to guard any polygon with n vertices.



Questions

4. What about a polygon with n vertices and h holes?
Shermer'82: $\lfloor (n+h)/3 \rfloor$ guards are sometimes necessary.
O'Rourke'82: $\lfloor (n+2h)/3 \rfloor$ guards are always sufficient.
Conjecture: $\lfloor (n+h)/3 \rfloor$ is a tight bound.



5. The problem of finding the smallest number of guards is NP-hard [Lee and Lin'86] and APX-hard [Eidenbenz'02]. There exists an O(log n)-approximation algorithm for vertex guards with running time $O(n^5)$. [Gosh'10].

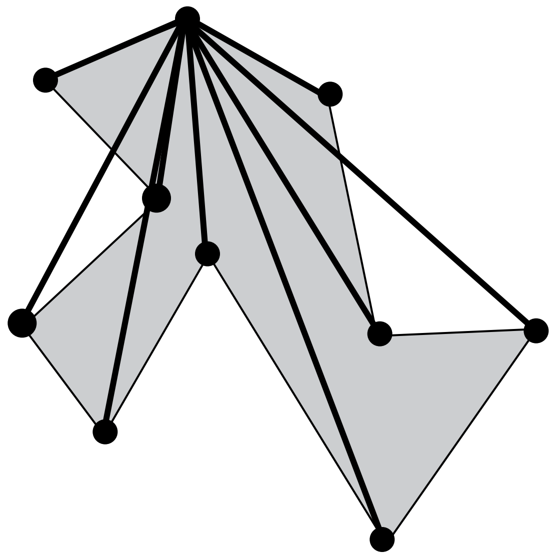
Triangulation algorithm 1

Theorem: Every polygon has a diagonal.

Testing a diagonal: $O(n)$ Why?

Algorithm 1:

while P not triangulated do
  (x,y) := find_valid_diagonal(P)
  output (x,y)



Time complexity:

#iterations = $O(n)$

#diagonals = $O(n^2)$

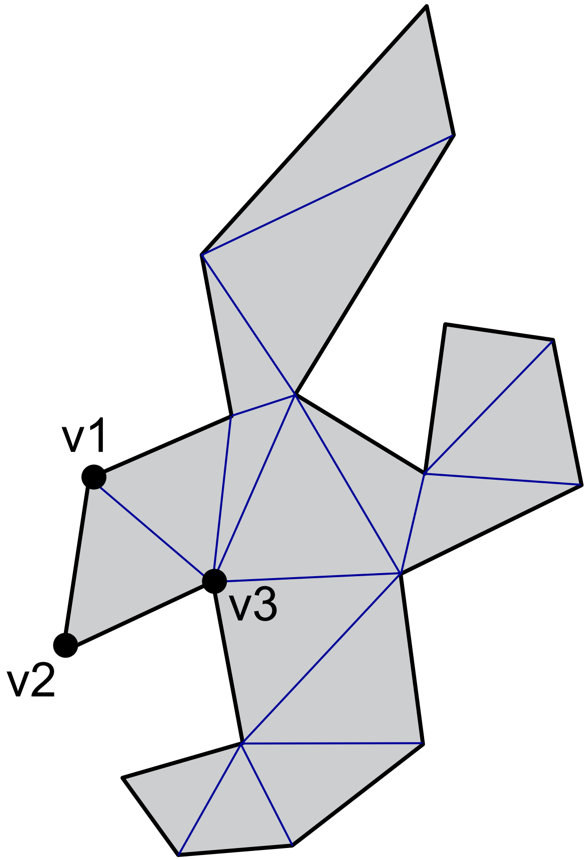
Test a diagonal = $O(n)$ $\Rightarrow$ $O(n^4)$

Triangulation algorithm 2

Theorem: Every polygon has at least two non-overlapping ears.

Algorithm 2:

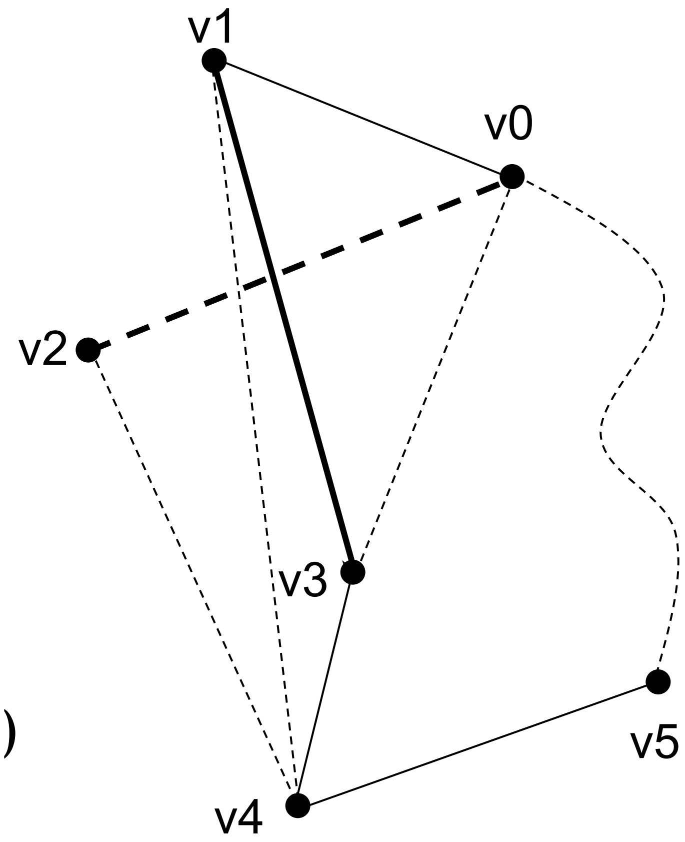
while n > 3 do	n-3
locate a valid ear tip v2	$O(n^2)$
output diagonal (v1,v3)	$O(1)$
delete v2 from P	$O(1)$
	Total: $O(n^3)$



Triangulation algorithm 3

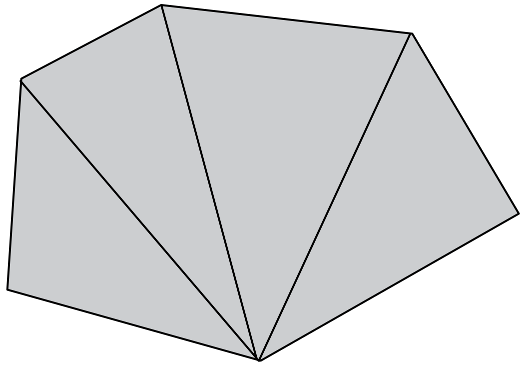
Algorithm 3:

compute all valid ears S	$O(n^2)$
while n > 3 do	n-3
locate a valid ear tip v2	O(1)
output diagonal (v1,v3)	O(1)
delete v2 from P	O(1)
delete (v0,v1,v2) from S	O(n)
delete (v2,v3,v4) from S	O(n)
check ear (v0,v1,v3)	O(n)
check ear (v1,v3,v4)	O(n)
	Total: $O(n^2)$

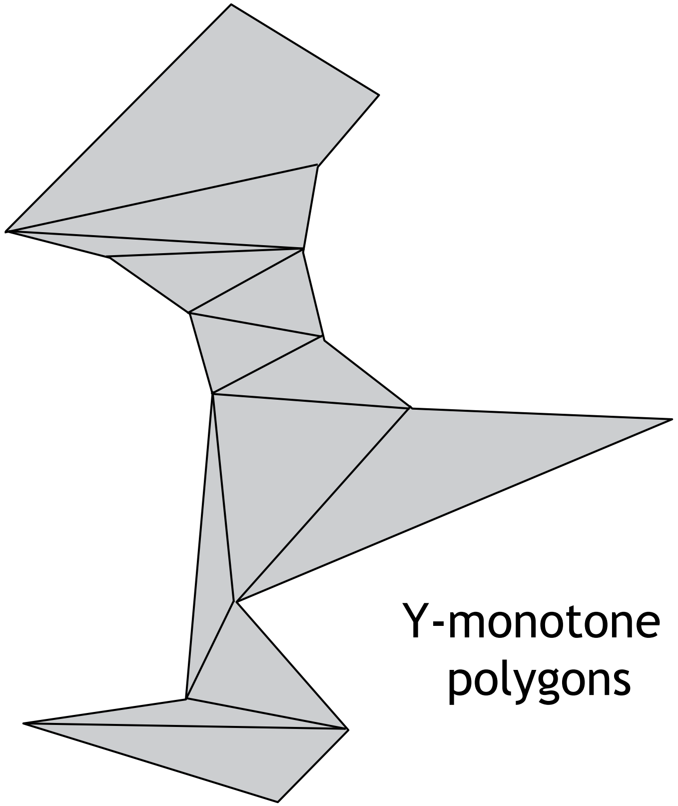


Triangulation algorithm 4

Observation: Some polygons are very easy to triangulate.



Convex polygons



Y-monotone polygons

Polygon triangulation 4

Algorithm 4: [Lecture 2]

Partition P into y-monotone pieces	O(n log n)
Triangulate every y-monotone polygon	O(n)

Theorem: Every simple polygon can be triangulated in O(n log n) time.

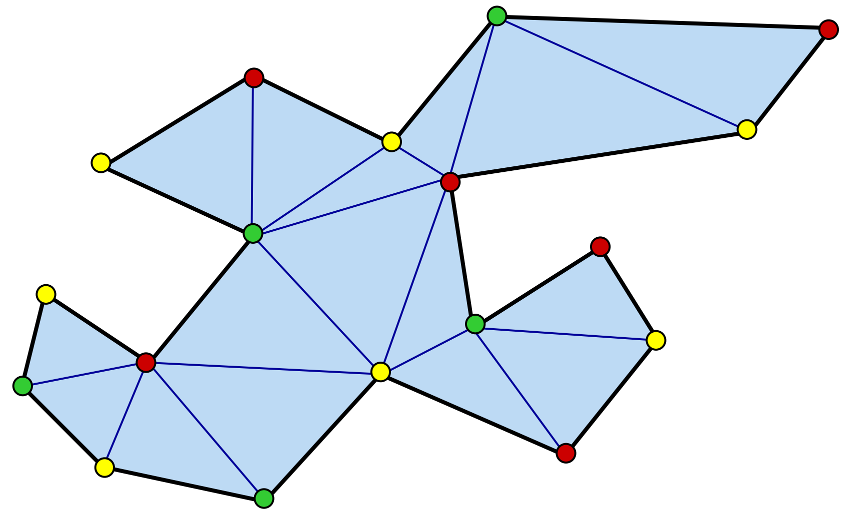
Polygon triangulation

- O(n log n) time [Garey, Johnson, Preparata & Tarjan'78]
- O(n loglog n) [Tarjan & van Wijk'88]
- O(n log* n) [Clarkson et al.'89]
- O(n) [Chazelle'91]
- O(n) randomised [Amato, Goodrich & Ramos'00]

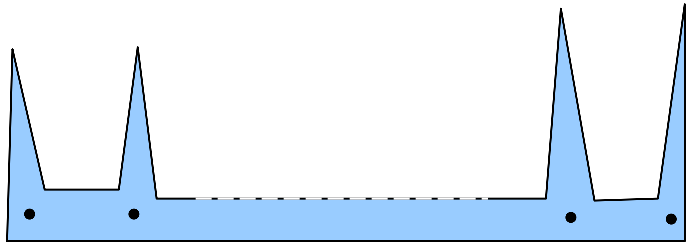
Open problem: Is there a simple O(n)-time algorithm?

Summary

- Every simple polygon with n vertices can be decomposed into n-2 triangles.



- Every triangulated simple polygon can be 3-colourable.
- Every simple polygon can be "guarded" by n/3 guards, and n/3 guards is sometimes necessary.



- To find a guard set our algorithm requires a triangulation.
  - $O(n^2)$ time algorithm
  - $O(n \log n)$ time algorithm [Lecture 2]