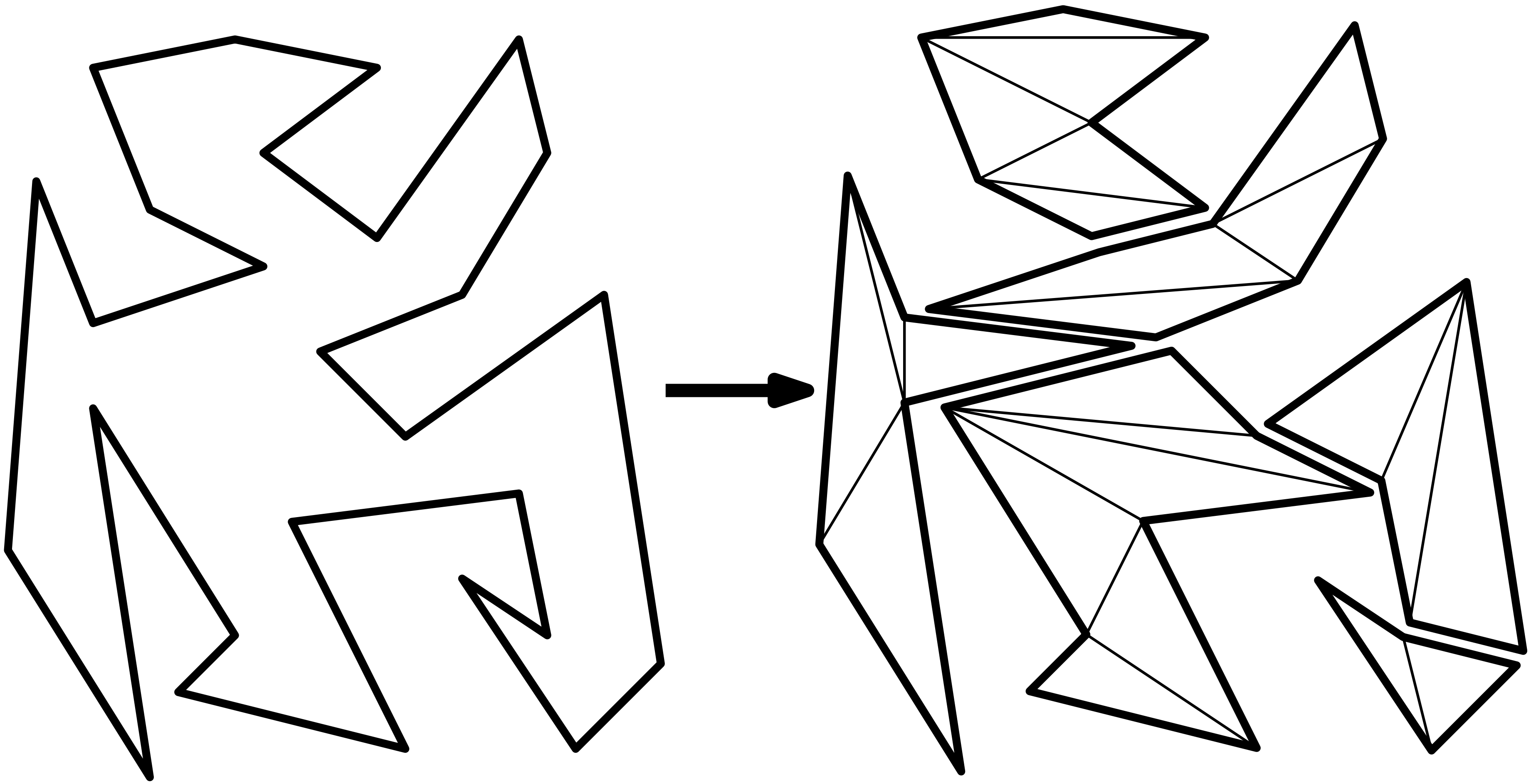
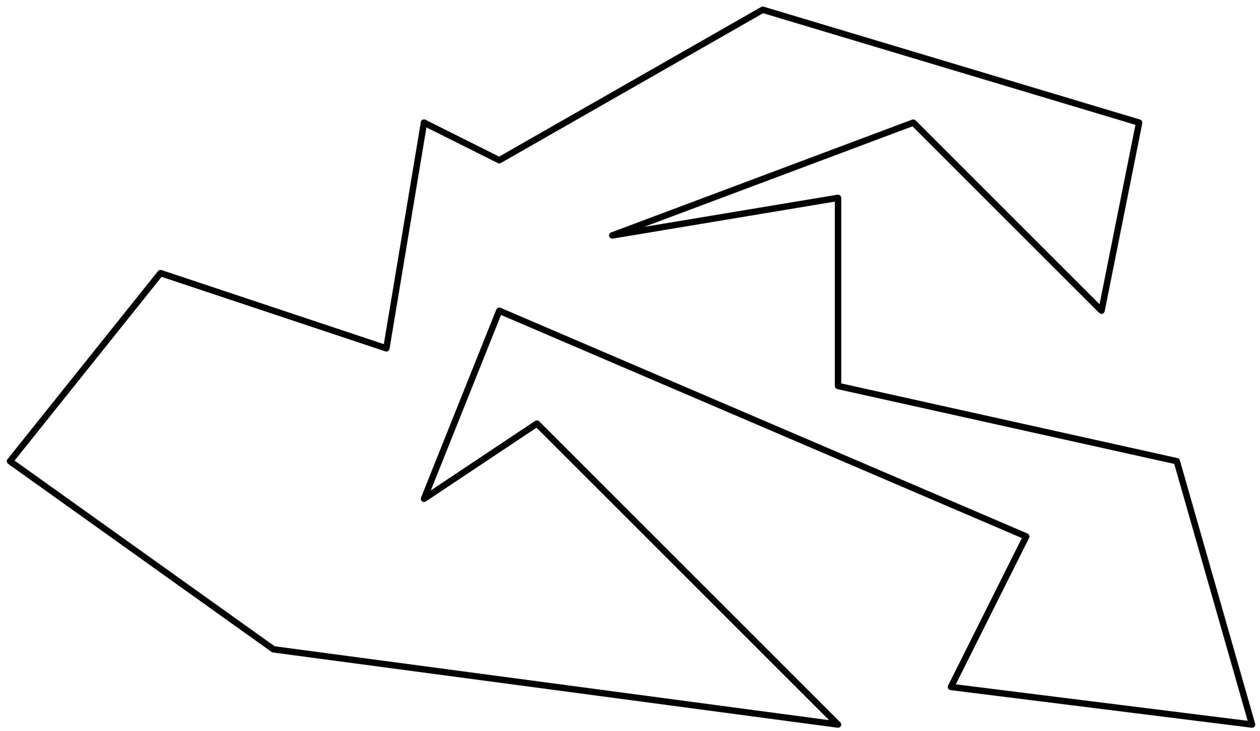
Polygon Triangulation



Mikkel Abrahamsen

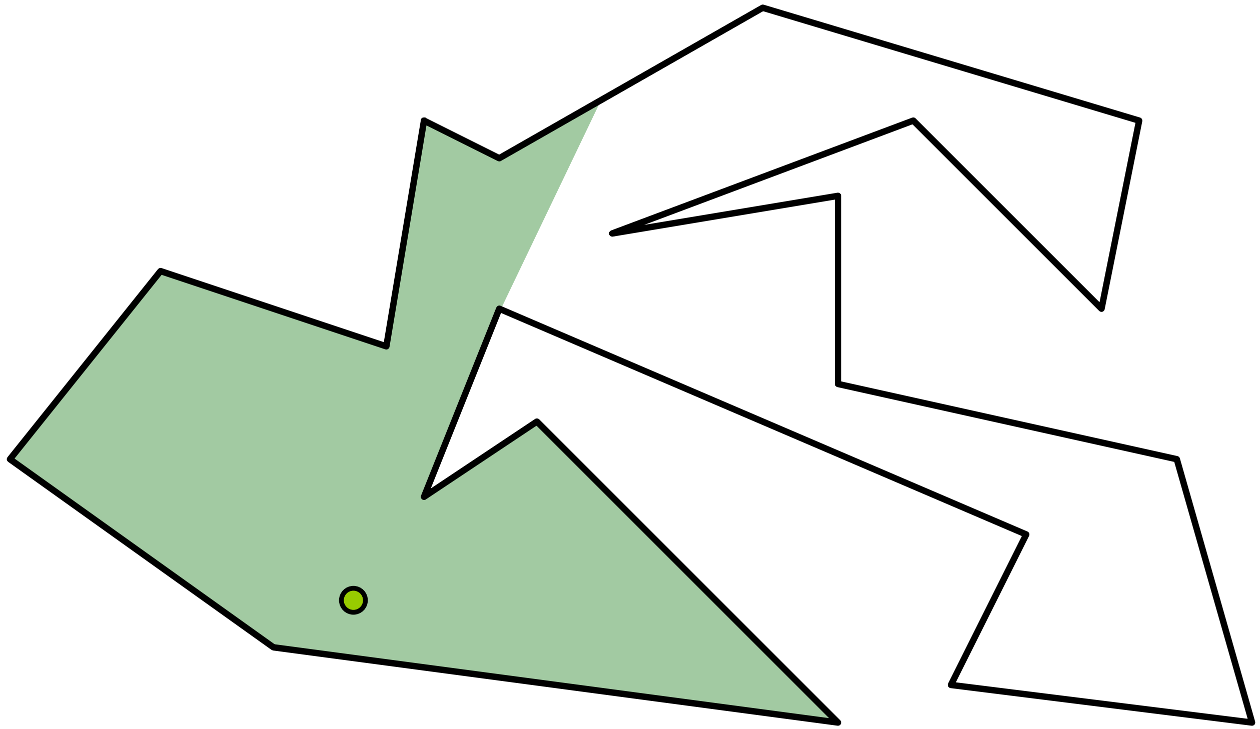
Motivation: The Art Gallery Problem

Viktor Klee 1973: How many guards are needed to cover a given art gallery (polygon) with n vertices?



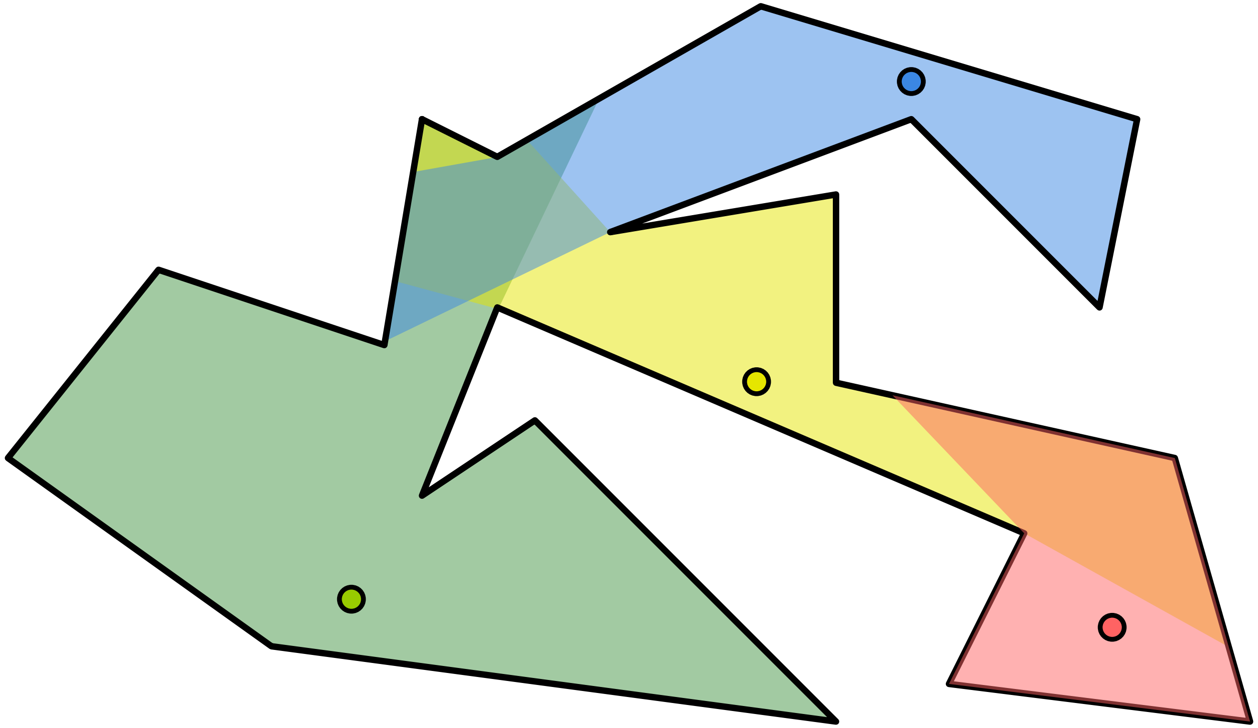
Motivation: The Art Gallery Problem

Viktor Klee 1973: How many guards are needed to cover a given art gallery (polygon) with n vertices?



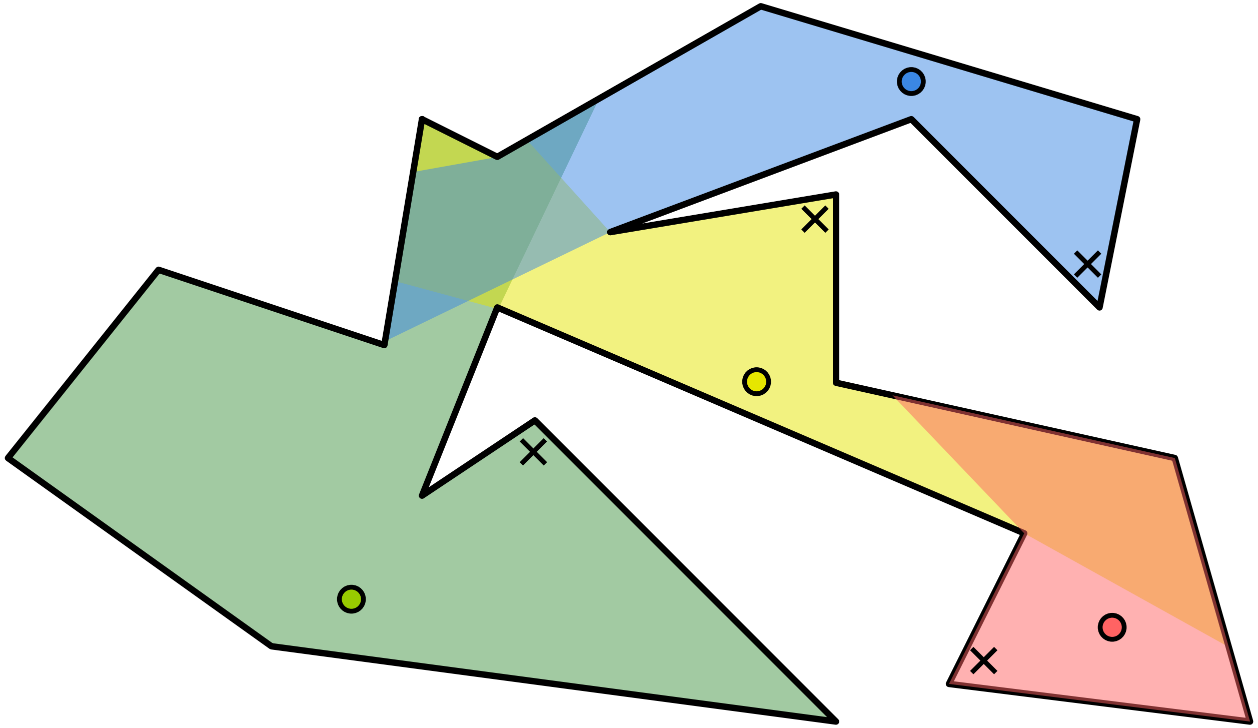
Motivation: The Art Gallery Problem

Viktor Klee 1973: How many guards are needed to cover a given art gallery (polygon) with n vertices?



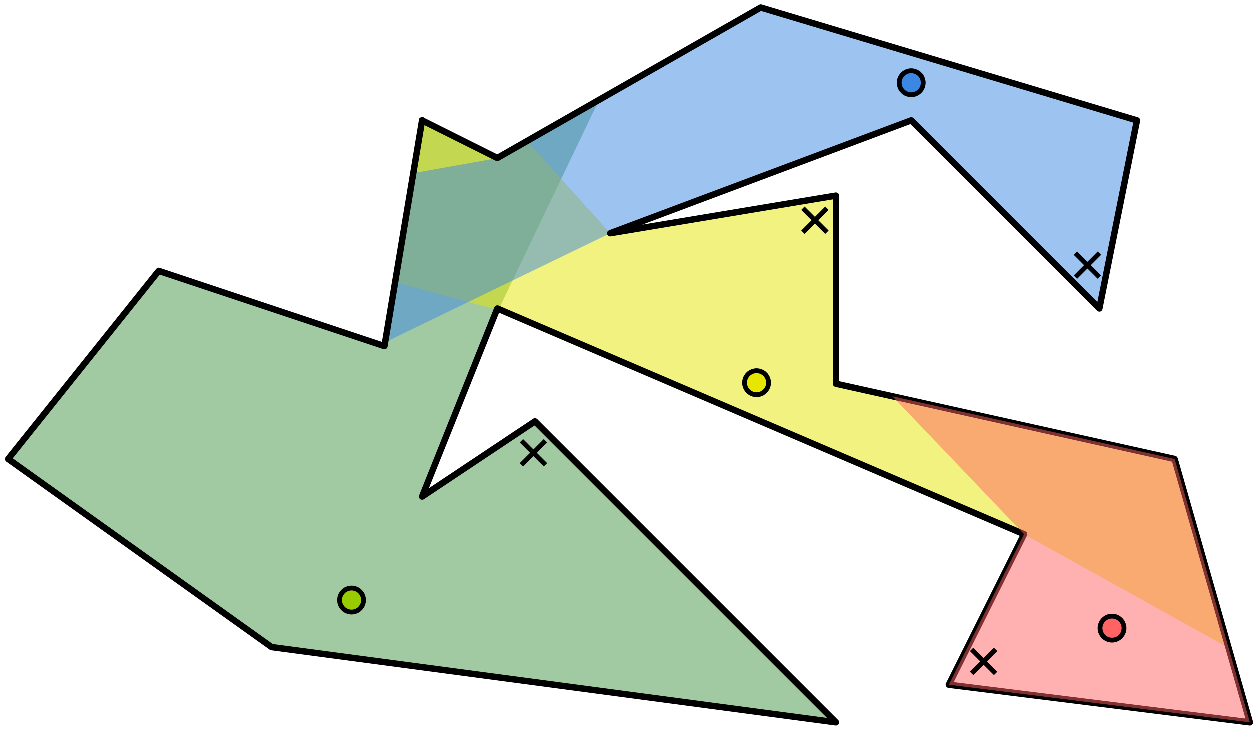
Motivation: The Art Gallery Problem

Viktor Klee 1973: How many guards are needed to cover a given art gallery (polygon) with n vertices?



Motivation: The Art Gallery Problem

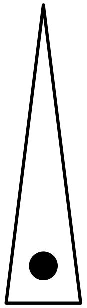
Viktor Klee 1973: How many guards are needed to cover a given art gallery (polygon) with n vertices?



Input. Array of points $(x_1, y_1), \ldots, (x_n, y_n)$: the corners of the art gallery in cyclic order, where $x_i, y_i \in \mathbb{Z}$. An integer k.
Output. Can we place k guards that see the entire art gallery?

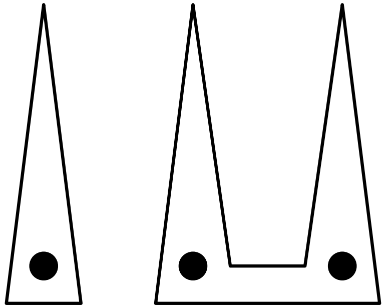
$\left\lfloor \frac{n}{3} \right\rfloor$ guards are sometimes needed

If $n = 3k$, then k guards can be needed!



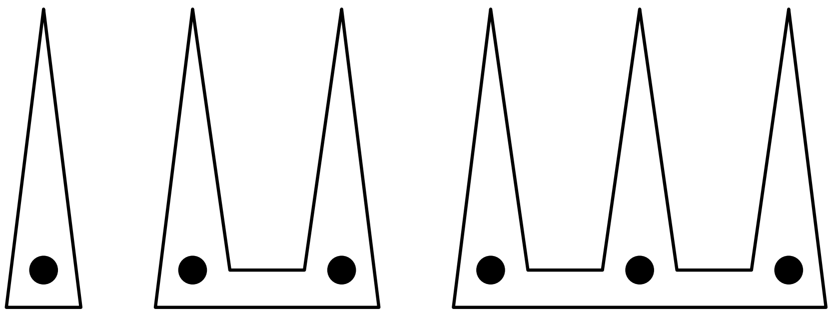
$\lfloor \frac{n}{3} \rfloor$ guards are sometimes needed

If $n = 3k$, then k guards can be needed!



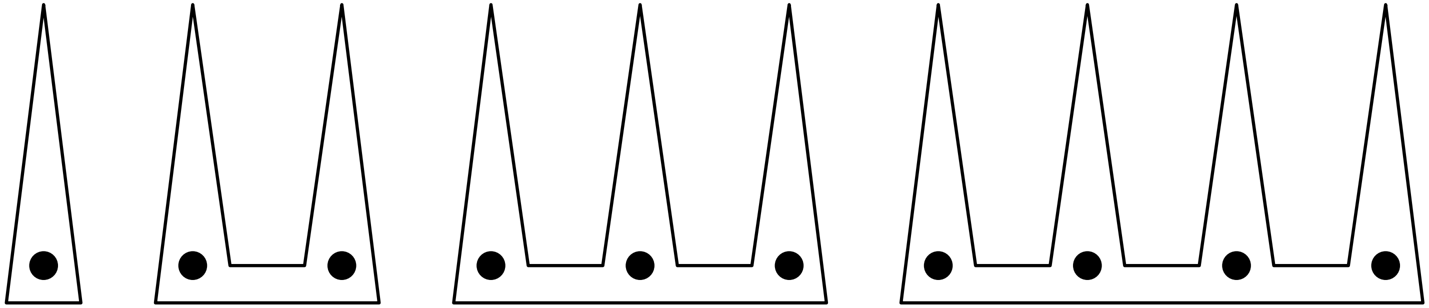
$\lfloor \frac{n}{3} \rfloor$ guards are sometimes needed

If $n = 3k$, then k guards can be needed!



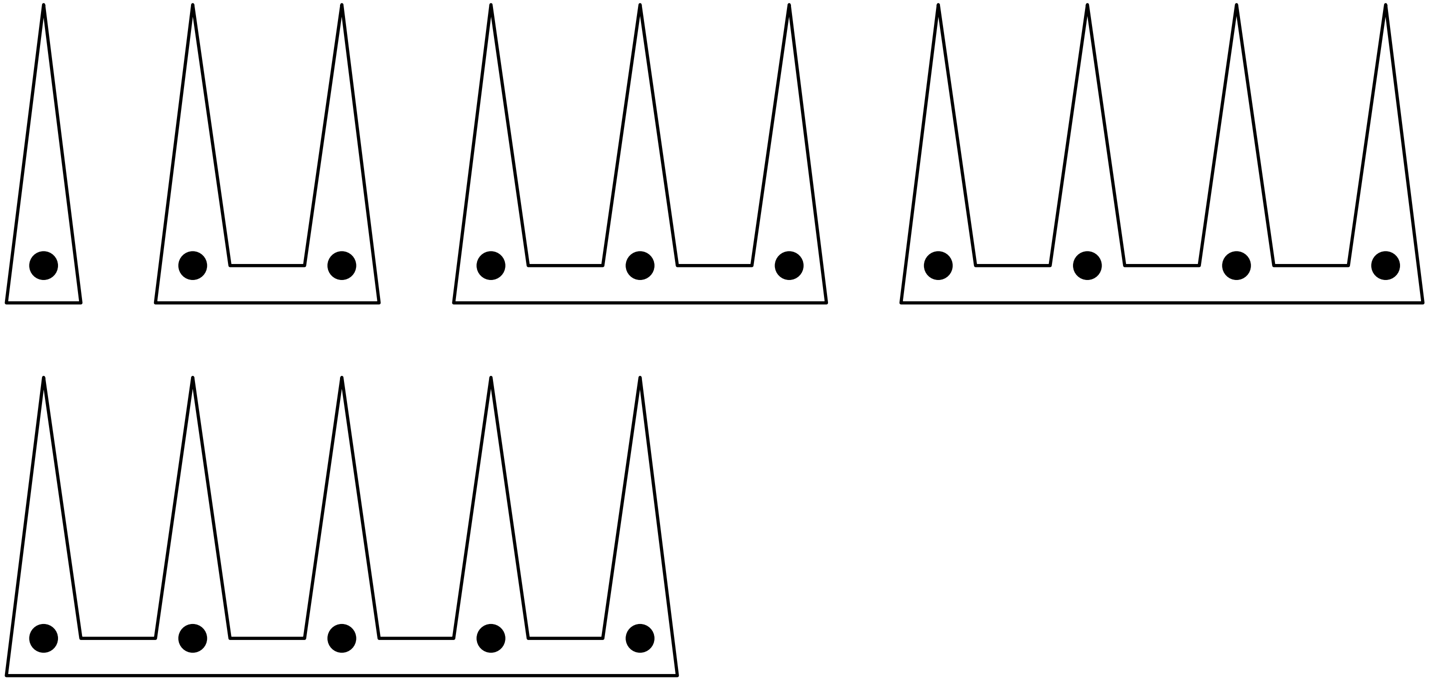
$\left\lfloor \frac{n}{3} \right\rfloor$ guards are sometimes needed

If $n = 3k$, then k guards can be needed!



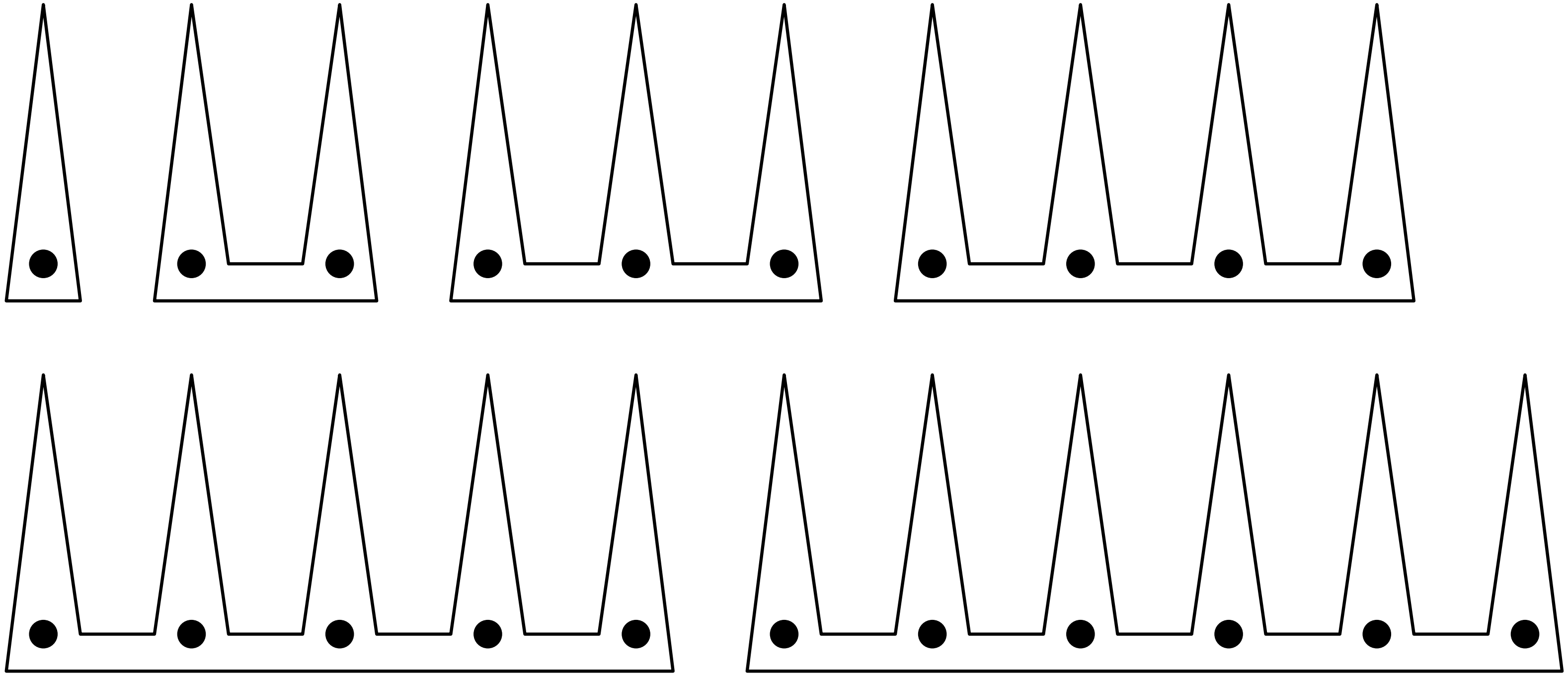
$\lfloor \frac{n}{3} \rfloor$ guards are sometimes needed

If $n = 3k$, then k guards can be needed!



$\lfloor \frac{n}{3} \rfloor$ guards are sometimes needed

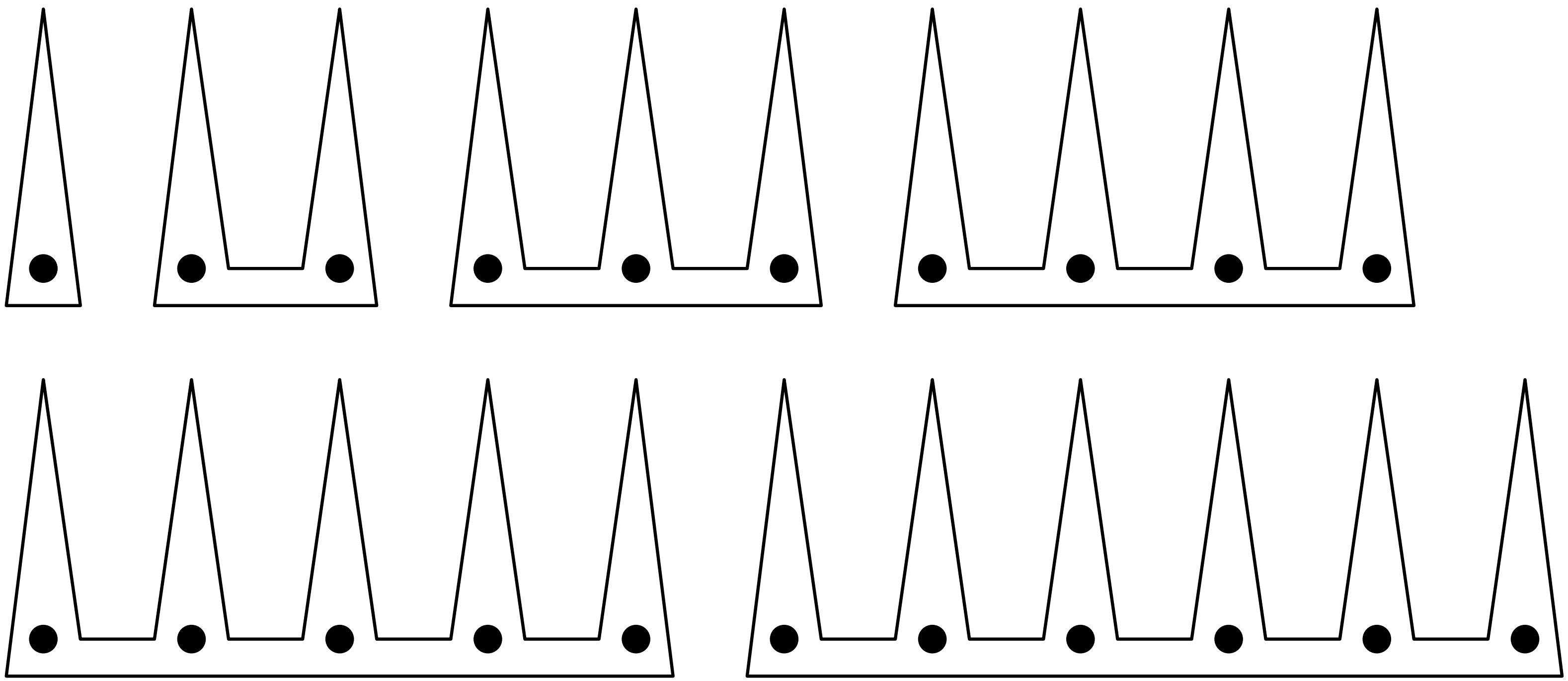
If $n = 3k$, then k guards can be needed!



Conclusion: $\lfloor \frac{n}{3} \rfloor$ guards are sometimes needed.

$\lfloor \frac{n}{3} \rfloor$ guards are sometimes needed

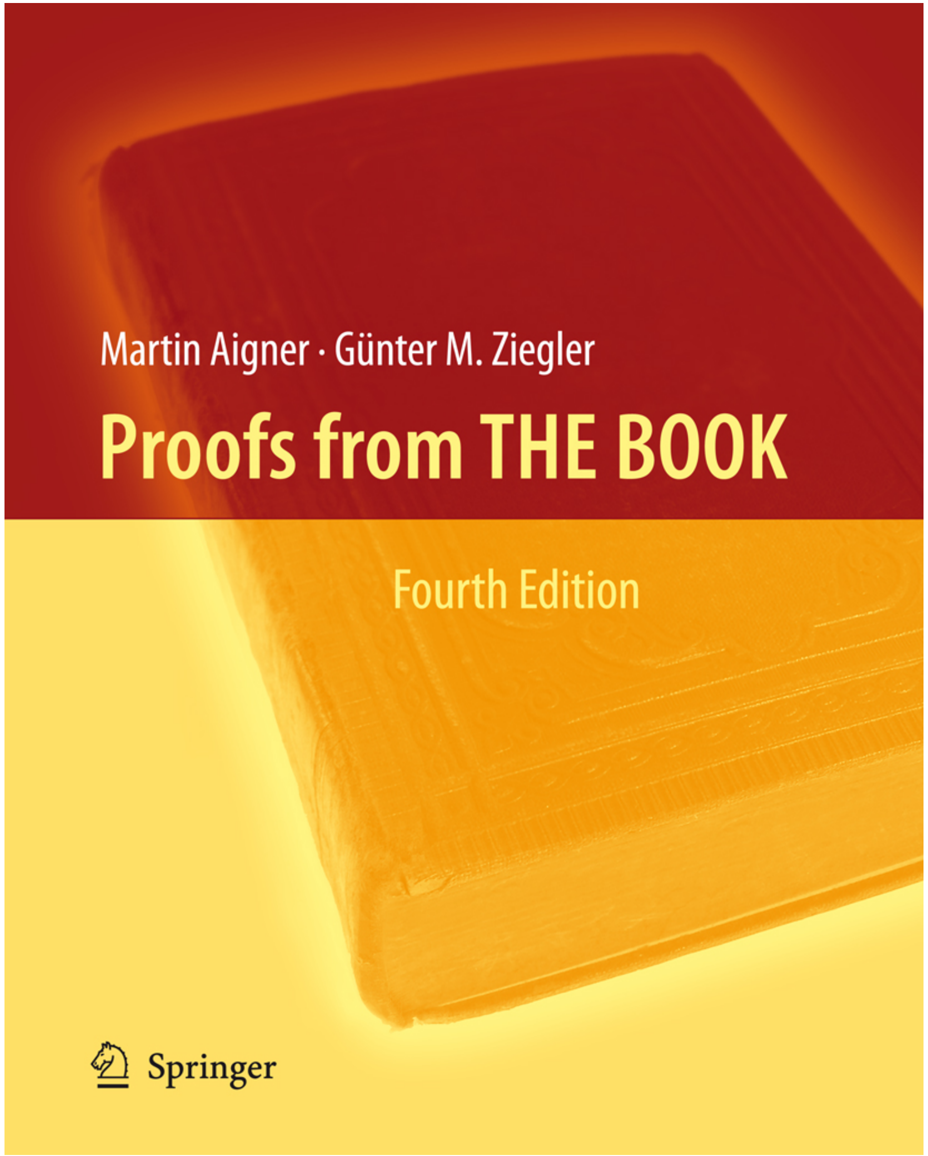
If $n = 3k$, then k guards can be needed!

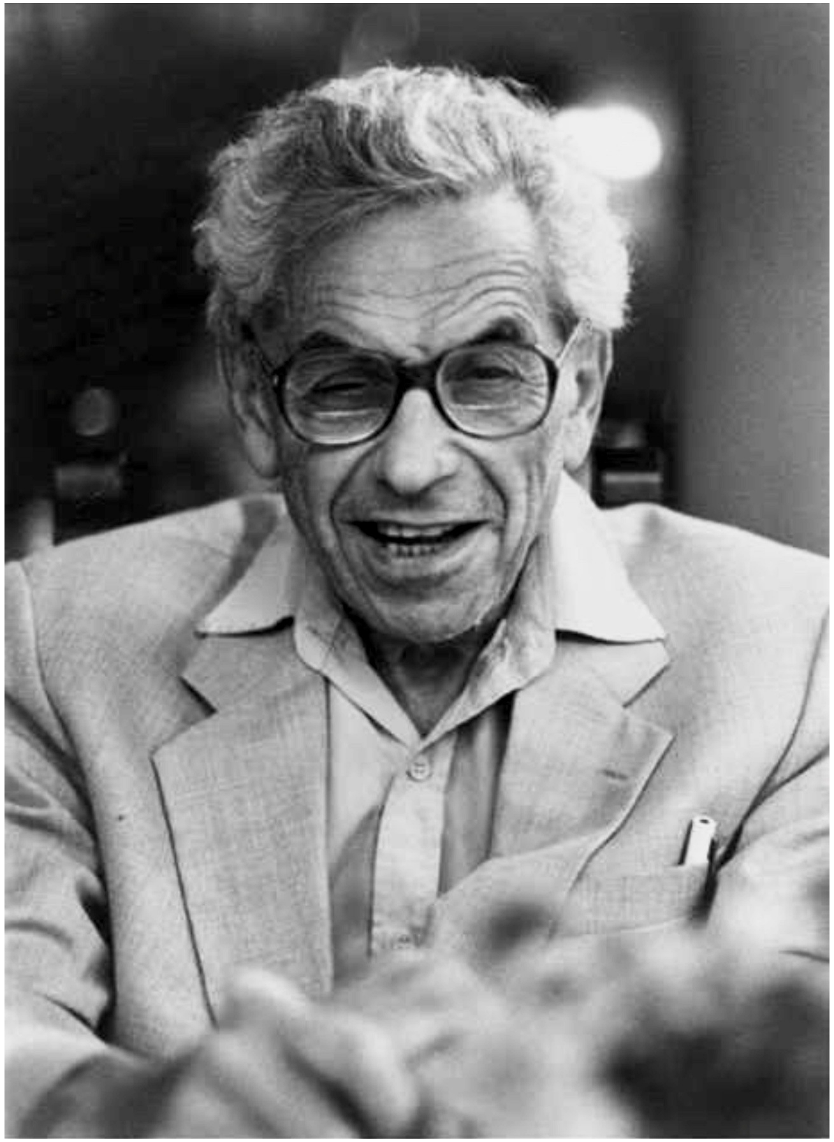


Conclusion: $\lfloor \frac{n}{3} \rfloor$ guards are sometimes needed.

Question: Is it always enough?

Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

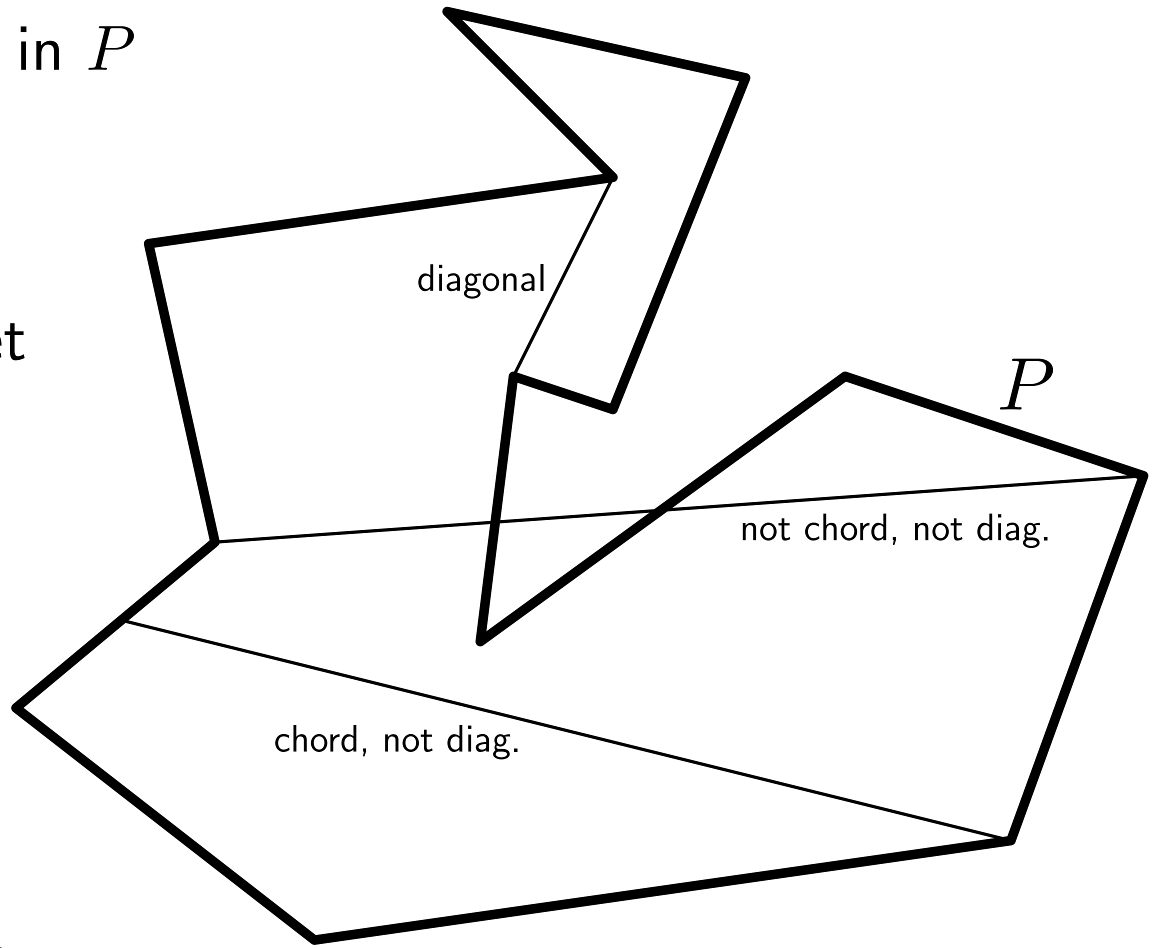


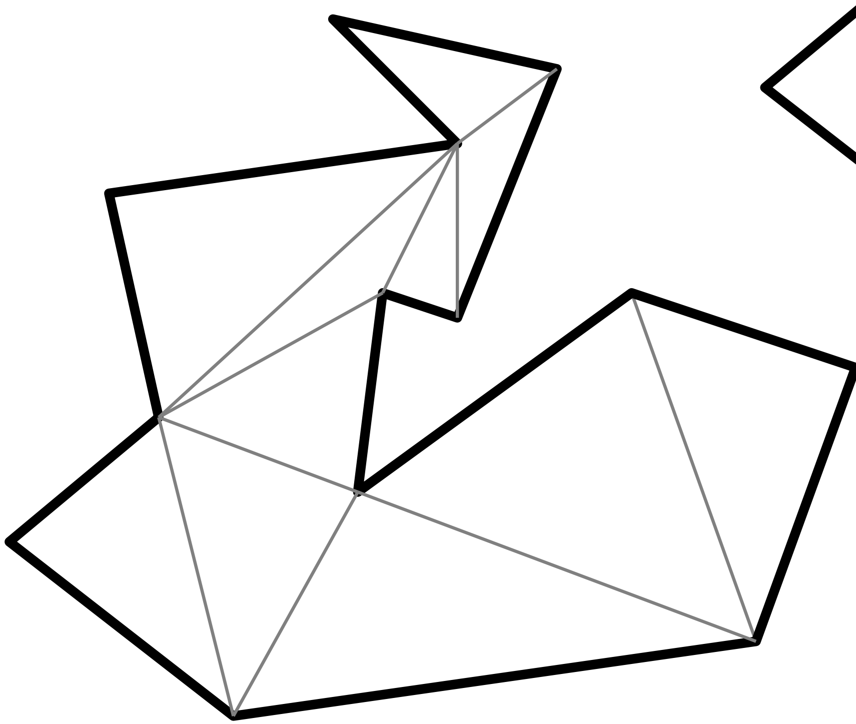


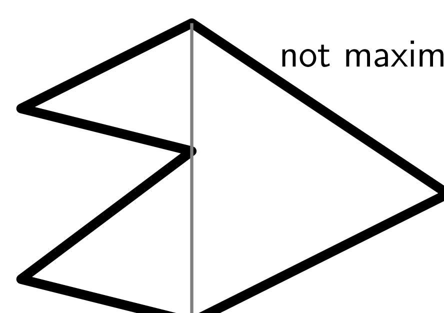
What is a triangulation?

Diagonal: Segment contained in P between two vertices.

Triangulation: Partition of P into triangles by a maximal set of non-intersecting diagonals.





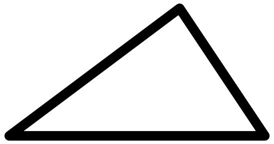


Any polygon can be triangulated

Lemma: A polygon P with n vertices can be triangulated, and any triangulation has $n-2$ triangles using $n-3$ diagonals.

Any polygon can be triangulated

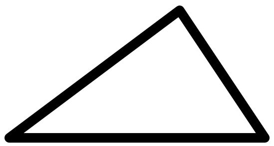
Lemma: A polygon P with n vertices can be triangulated, and any triangulation has $n-2$ triangles using $n-3$ diagonals.



Proof: Induction on n. Base case $n=3$ is trivial.

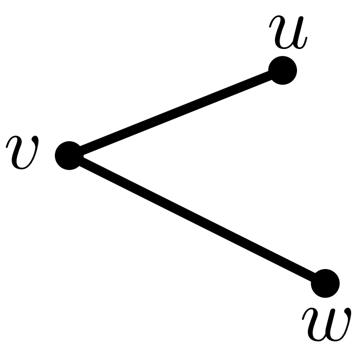
Any polygon can be triangulated

Lemma: A polygon P with n vertices can be triangulated, and any triangulation has $n-2$ triangles using $n-3$ diagonals.



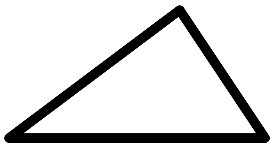
Proof: Induction on n. Base case $n = 3$ is trivial.

Induction step: v leftmost vertex, u and w neighbours.



Any polygon can be triangulated

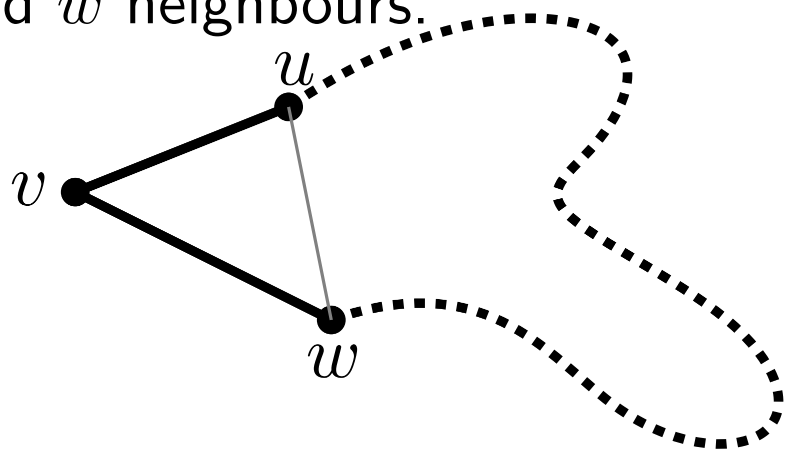
Lemma: A polygon P with n vertices can be triangulated, and any triangulation has $n-2$ triangles using $n-3$ diagonals.



Proof: Induction on n. Base case $n=3$ is trivial.

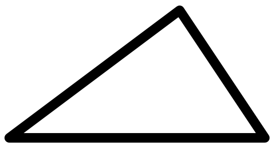
Induction step: v leftmost vertex, u and w neighbours.

Case 1: uw is a diagonal.



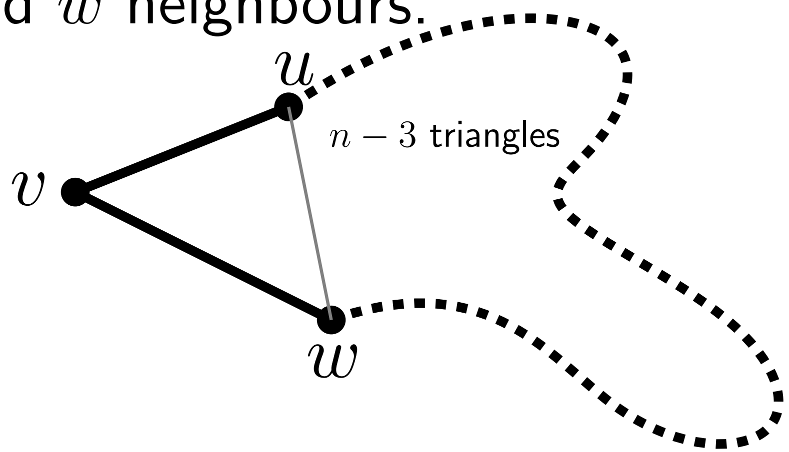
Any polygon can be triangulated

Lemma: A polygon P with n vertices can be triangulated, and any triangulation has $n-2$ triangles using $n-3$ diagonals.



Proof: Induction on n. Base case $n=3$ is trivial.

Induction step: v leftmost vertex, u and w neighbours.

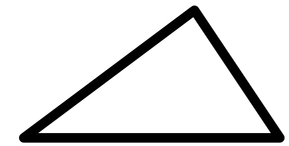


Case 1: uw is a diagonal.

Induction hypothesis $\Rightarrow$ triangulation with $n-3$ triangles on the other side of uw.

Any polygon can be triangulated

Lemma: A polygon P with n vertices can be triangulated, and any triangulation has $n-2$ triangles using $n-3$ diagonals.

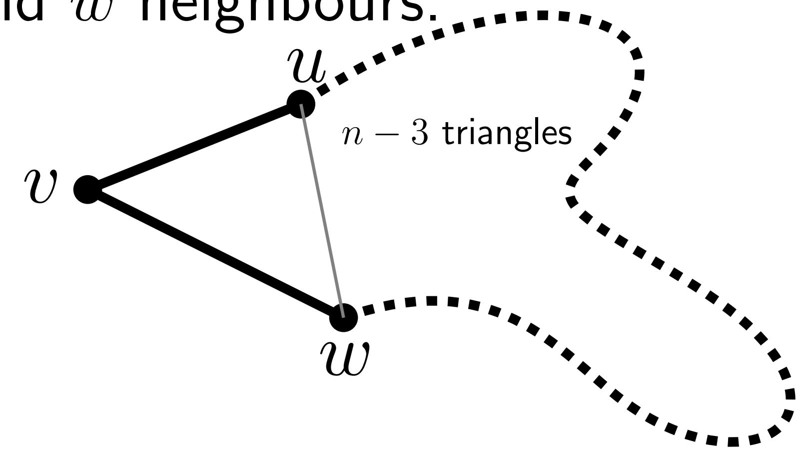


Proof: Induction on n. Base case $n=3$ is trivial.

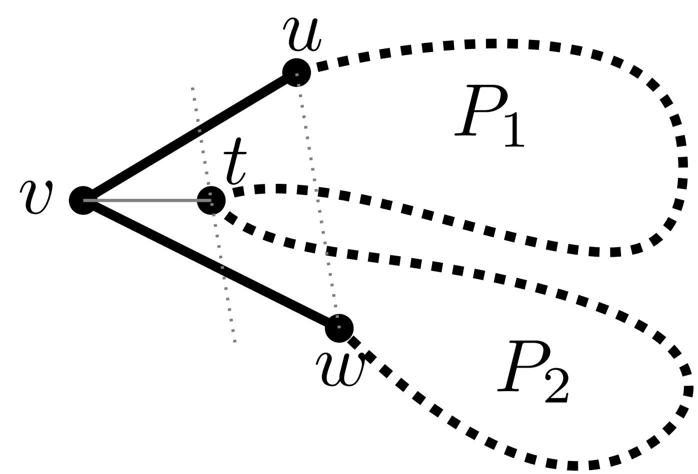
Induction step: v leftmost vertex, u and w neighbours.

Case 1: uw is a diagonal.

Induction hypothesis $\Rightarrow$ triangulation with $n-3$ triangles on the other side of uw.

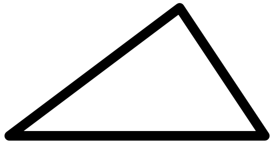


Case 2: uw is no diagonal. Let t be corner in uvw farthest from uw. vt is a diagonal, splits P into P_1 and P_2 with $m_1 < n$ and $m_2 < n$ vertices, $m_1 + m_2 = n + 2$.



Any polygon can be triangulated

Lemma: A polygon P with n vertices can be triangulated, and any triangulation has $n-2$ triangles using $n-3$ diagonals.

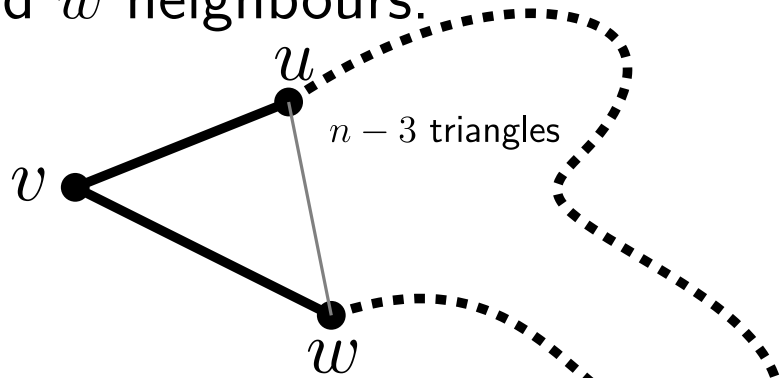


Proof: Induction on n. Base case $n=3$ is trivial.

Induction step: v leftmost vertex, u and w neighbours.

Case 1: uw is a diagonal.

Induction hypothesis $\Rightarrow$ triangulation with $n-3$ triangles on the other side of uw.



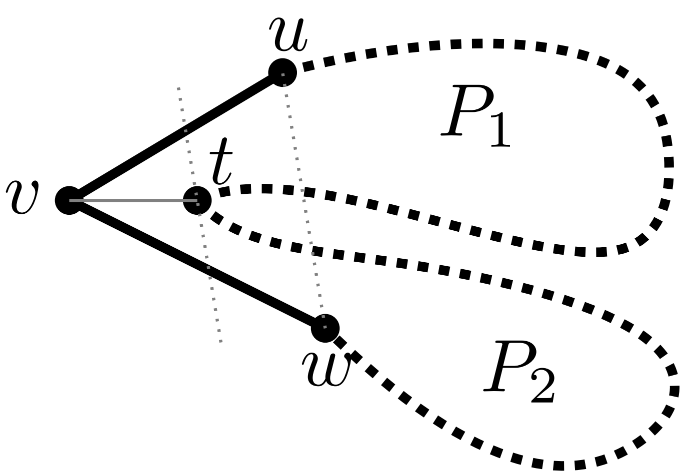
Case 2: uw is no diagonal. Let t be corner in uvw farthest from uw. vt is a diagonal, splits P into P_1 and P_2 with $m_1 < n$ and $m_2 < n$ vertices, $m_1 + m_2 = n + 2$.

Induction hypothesis $\Rightarrow$

P_1: $m_1 - 2$ triangles.

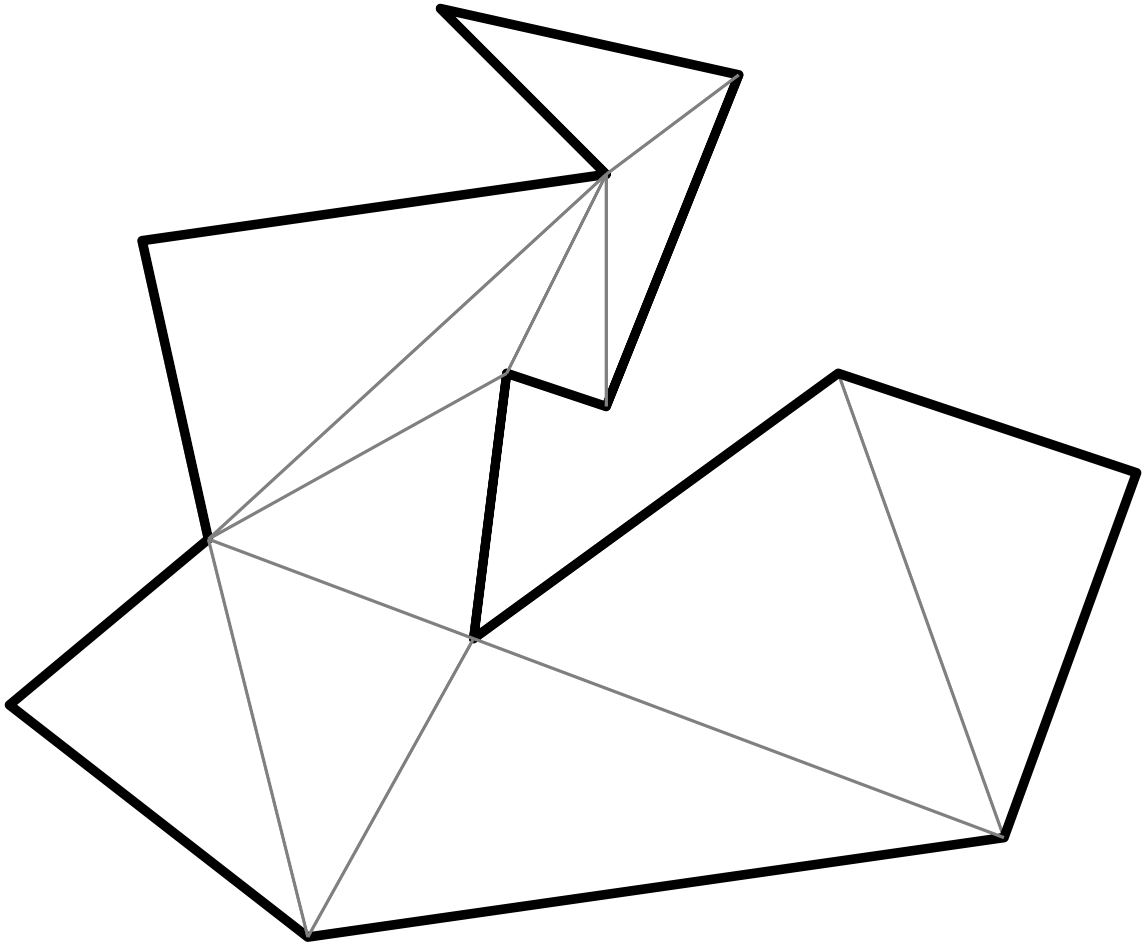
P_2: $m_2 - 2$ triangles.

In total: $m_1 + m_2 - 4 = n + 2 - 4 = n - 2$.



Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

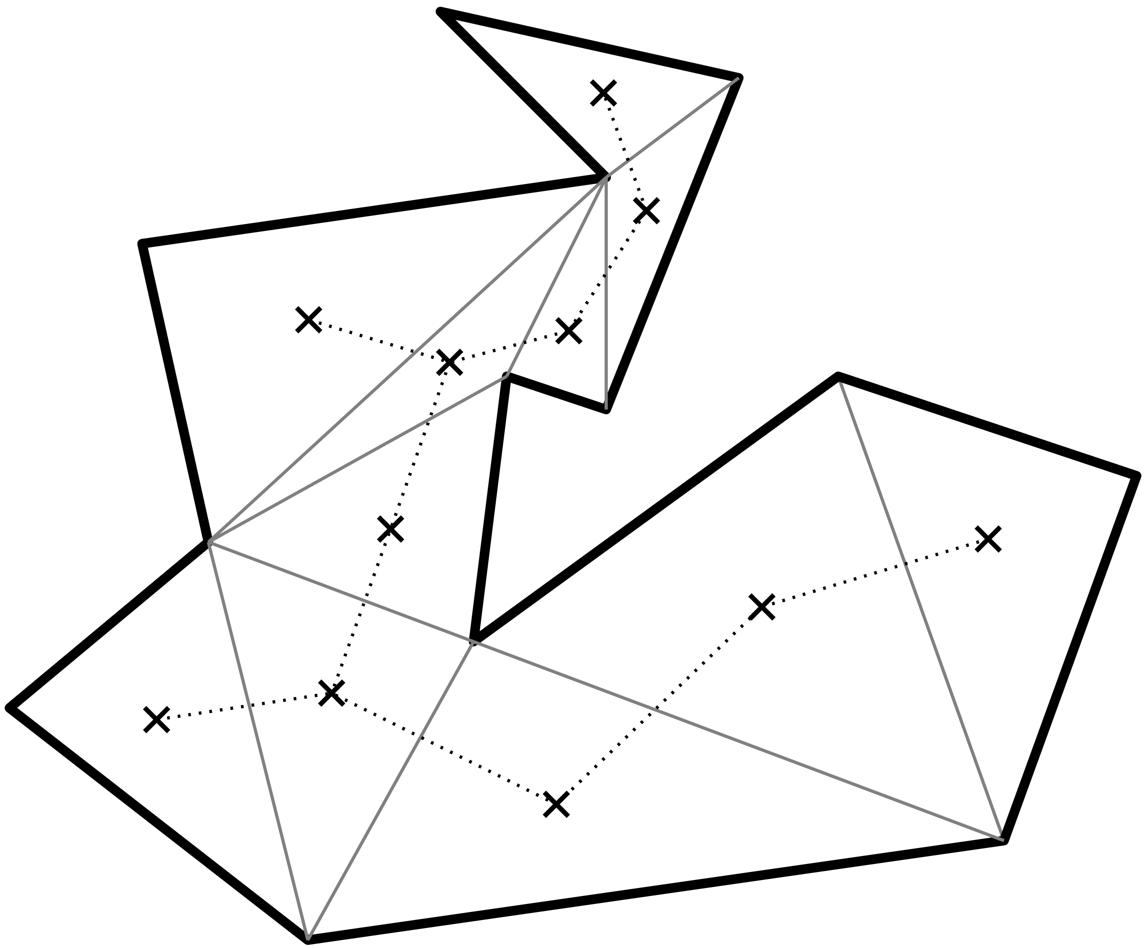
Consider a triangulation.



Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

Consider a triangulation.

Consider dual tree.

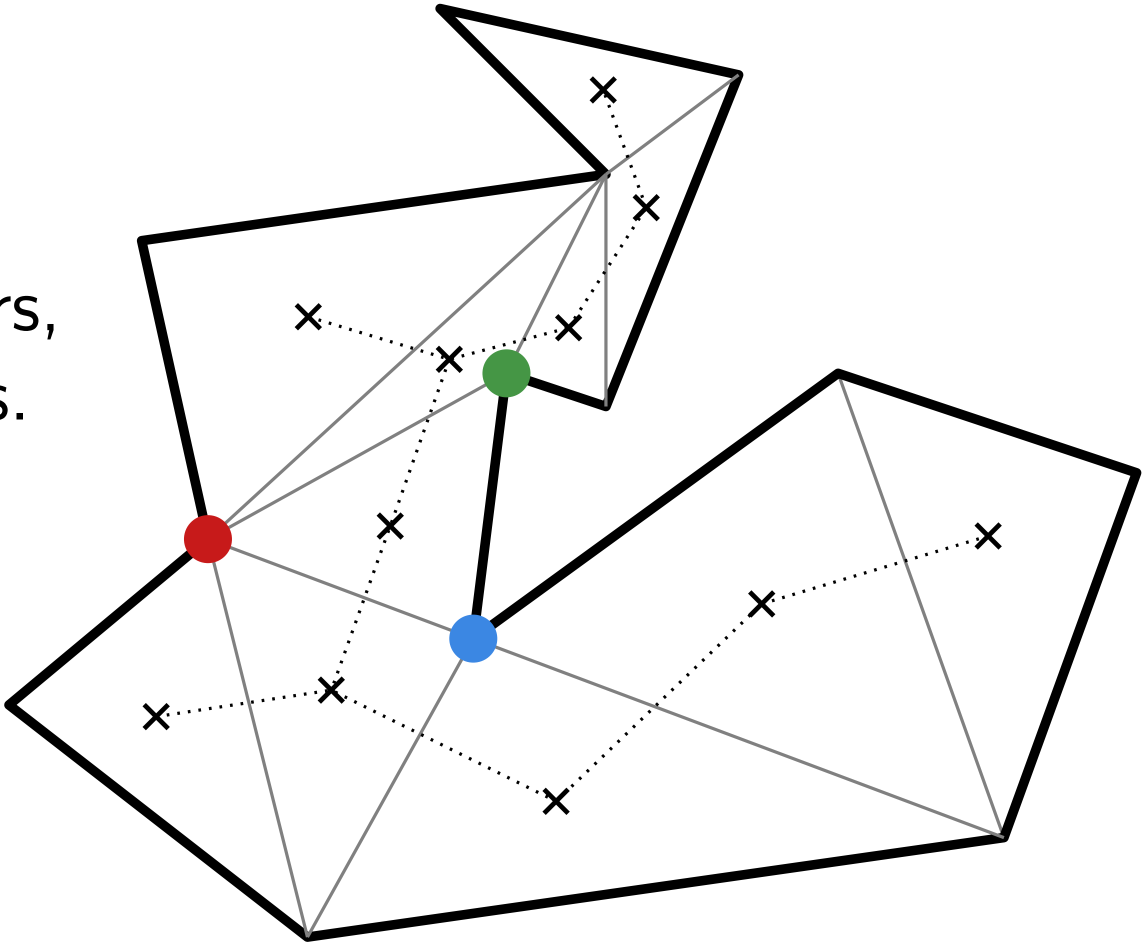


Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

Consider a triangulation.

Consider dual tree.

Color vertices with 3 colors, each triangle gets 3 colors.

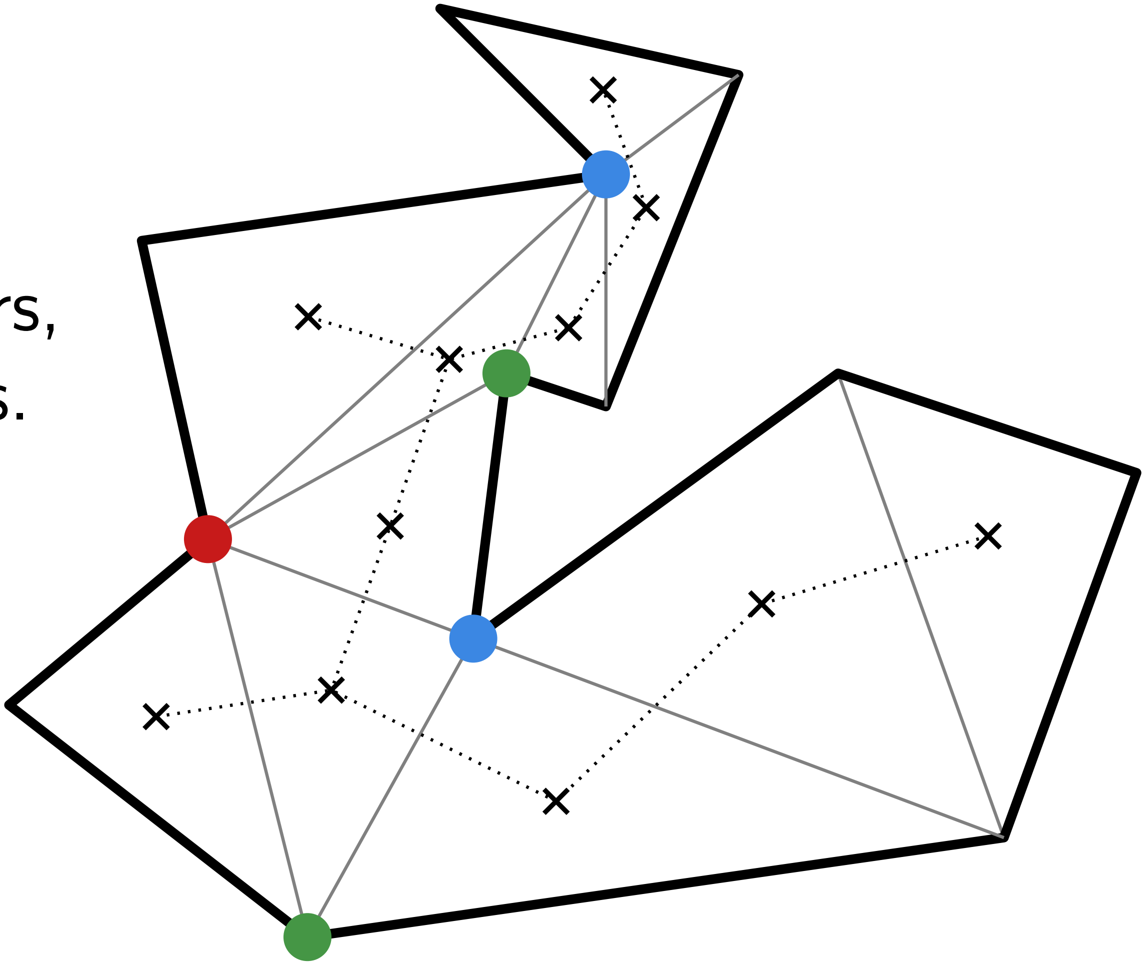


Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

Consider a triangulation.

Consider dual tree.

Color vertices with 3 colors, each triangle gets 3 colors.

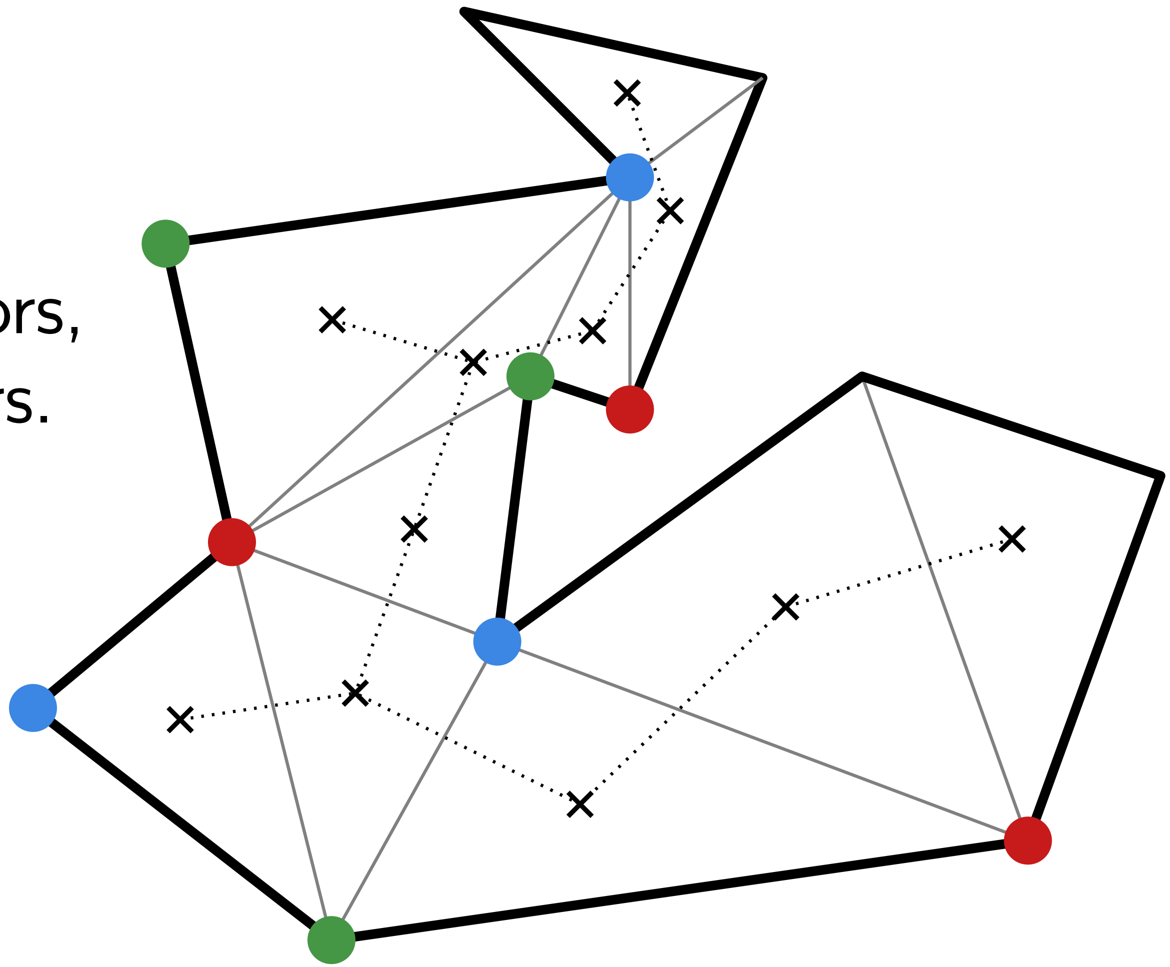


Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

Consider a triangulation.

Consider dual tree.

Color vertices with 3 colors, each triangle gets 3 colors.

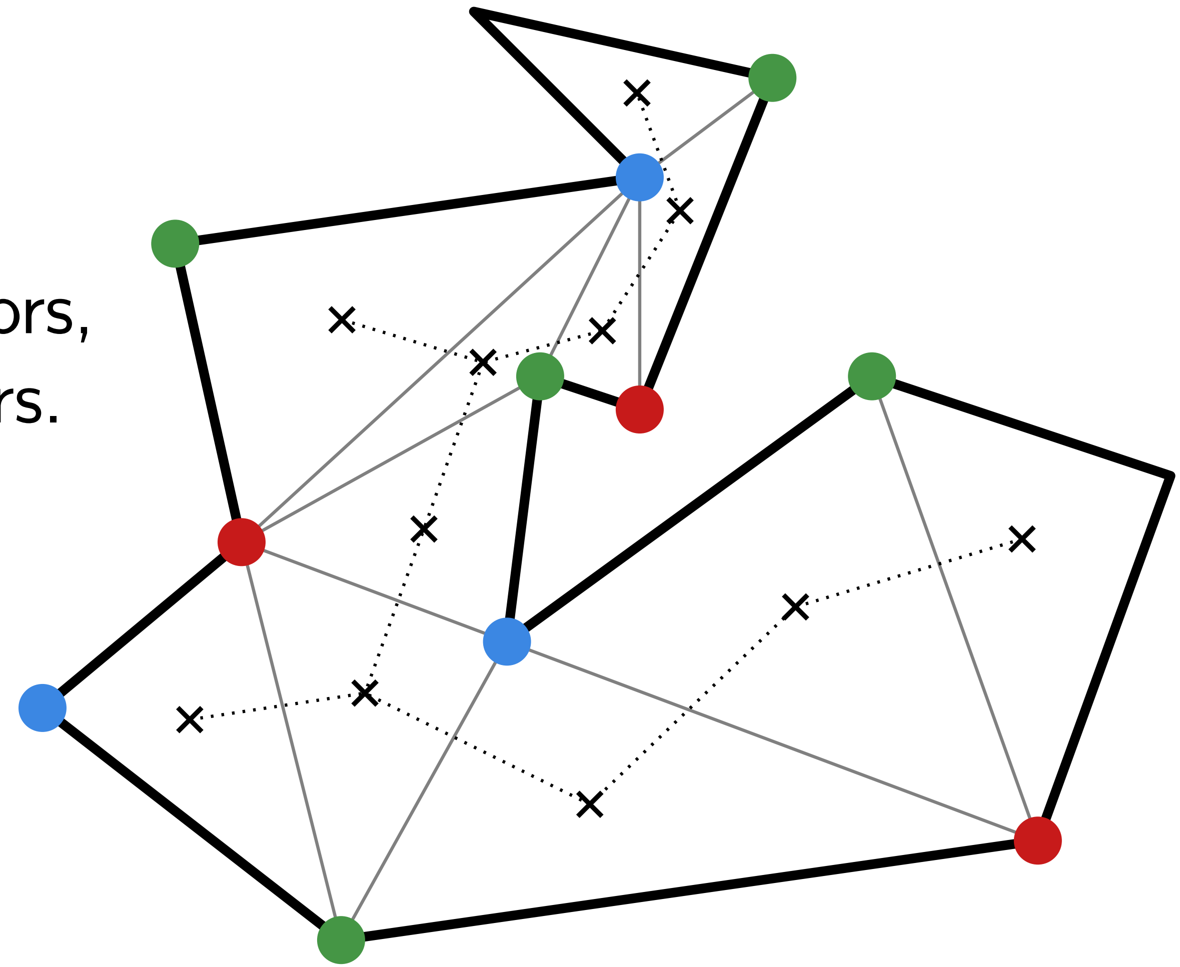


Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

Consider a triangulation.

Consider dual tree.

Color vertices with 3 colors, each triangle gets 3 colors.

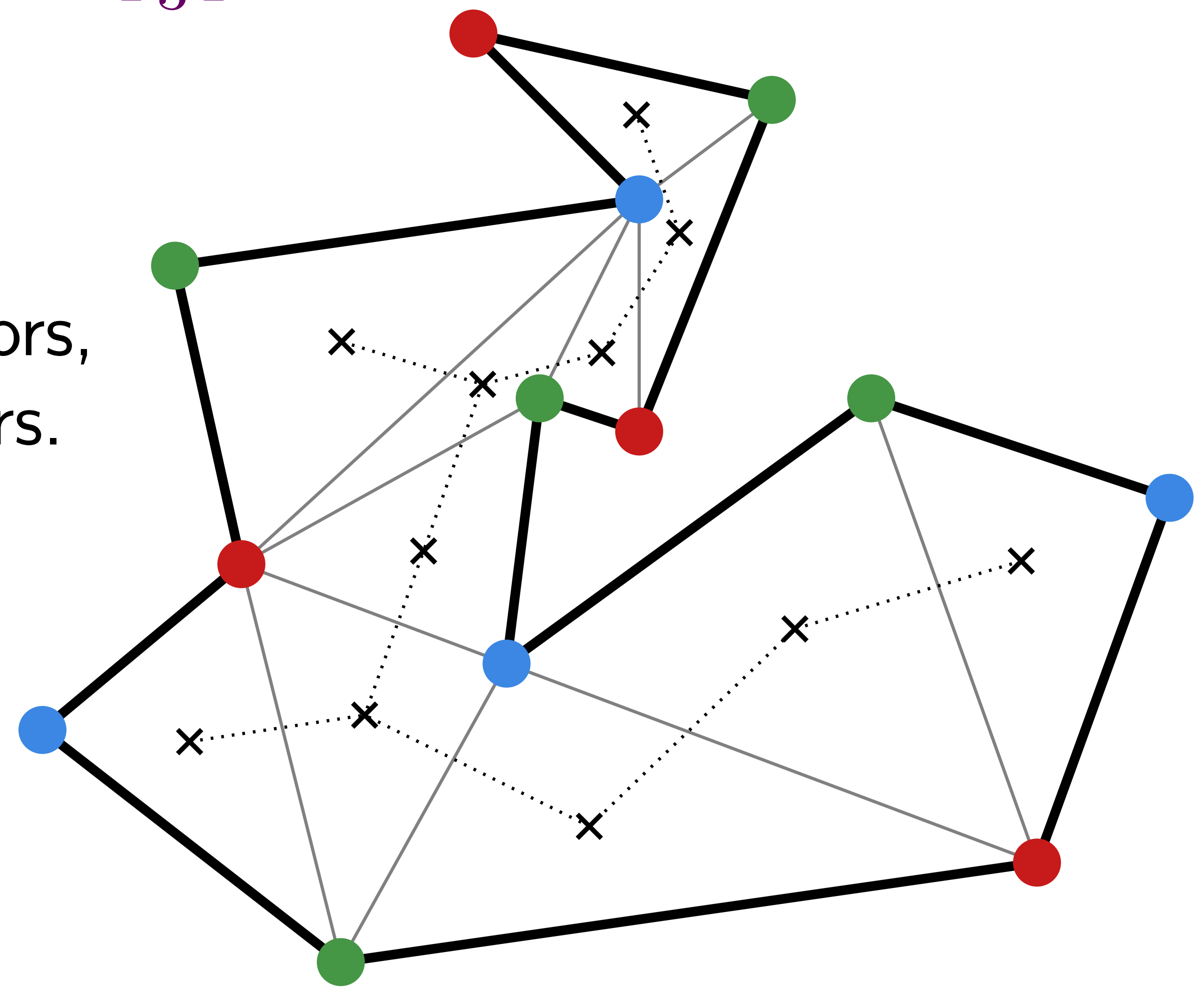


Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

Consider a triangulation.

Consider dual tree.

Color vertices with 3 colors, each triangle gets 3 colors.



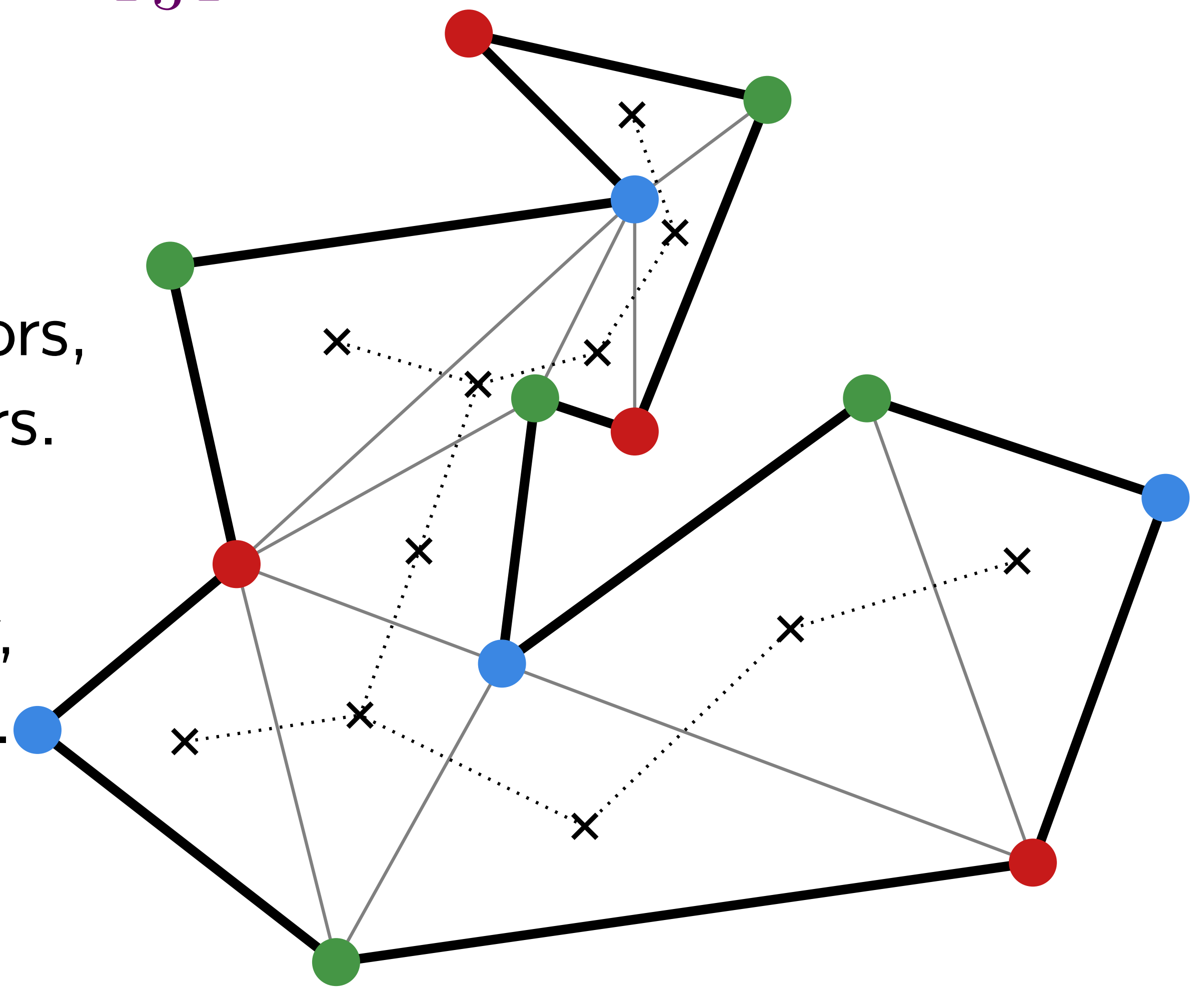
Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

Consider a triangulation.

Consider dual tree.

Color vertices with 3 colors, each triangle gets 3 colors.

Observe that the red vertices guard the gallery, as do the green and blue.



Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

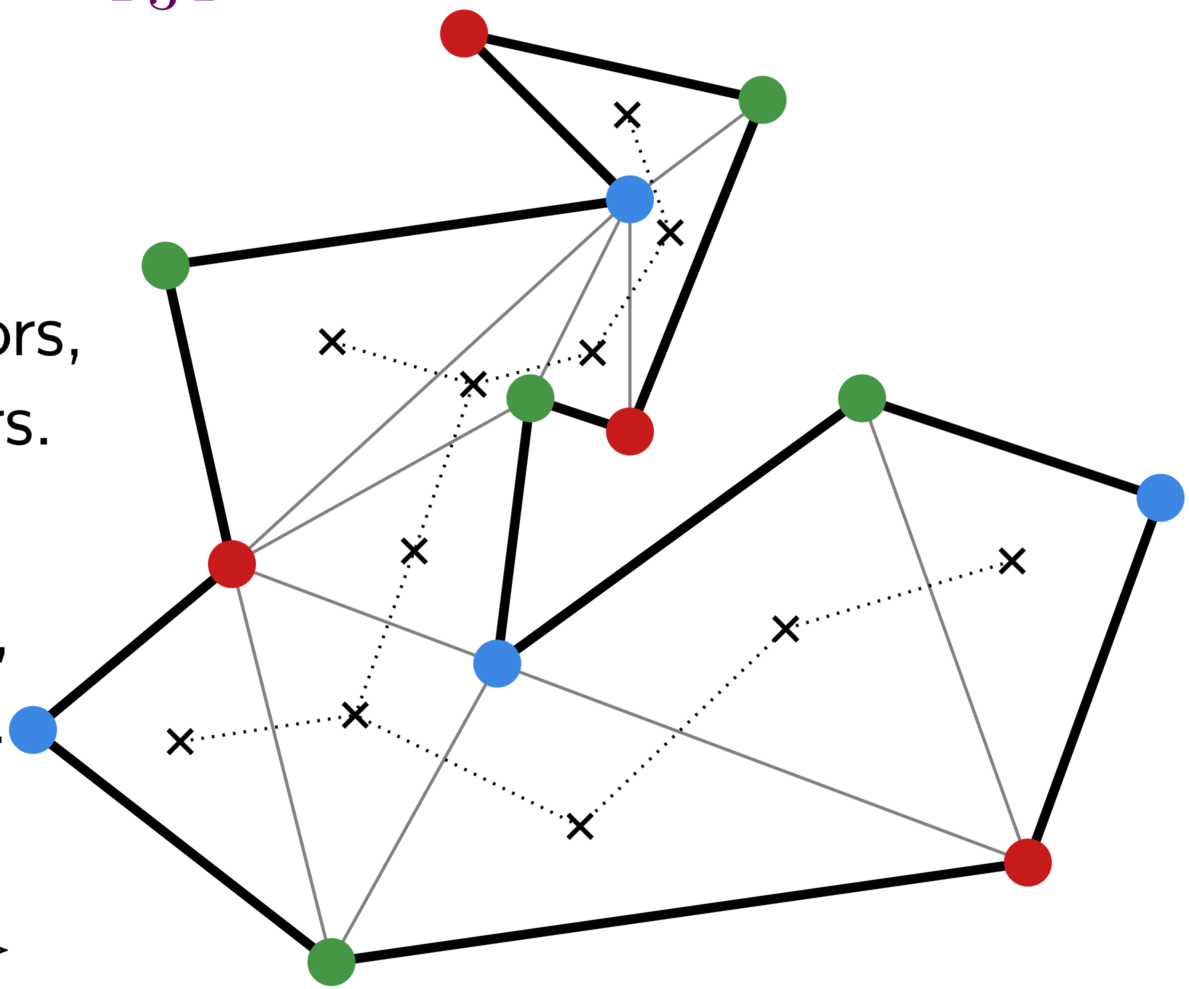
Consider a triangulation.

Consider dual tree.

Color vertices with 3 colors, each triangle gets 3 colors.

Observe that the red vertices guard the gallery, as do the green and blue.

$$n = n_r + n_g + n_b \implies$$
$$\min\{n_r, n_g, n_b\} \leq \frac{n}{3} \implies$$
$$\min\{n_r, n_g, n_b\} \leq \lfloor \frac{n}{3} \rfloor$$



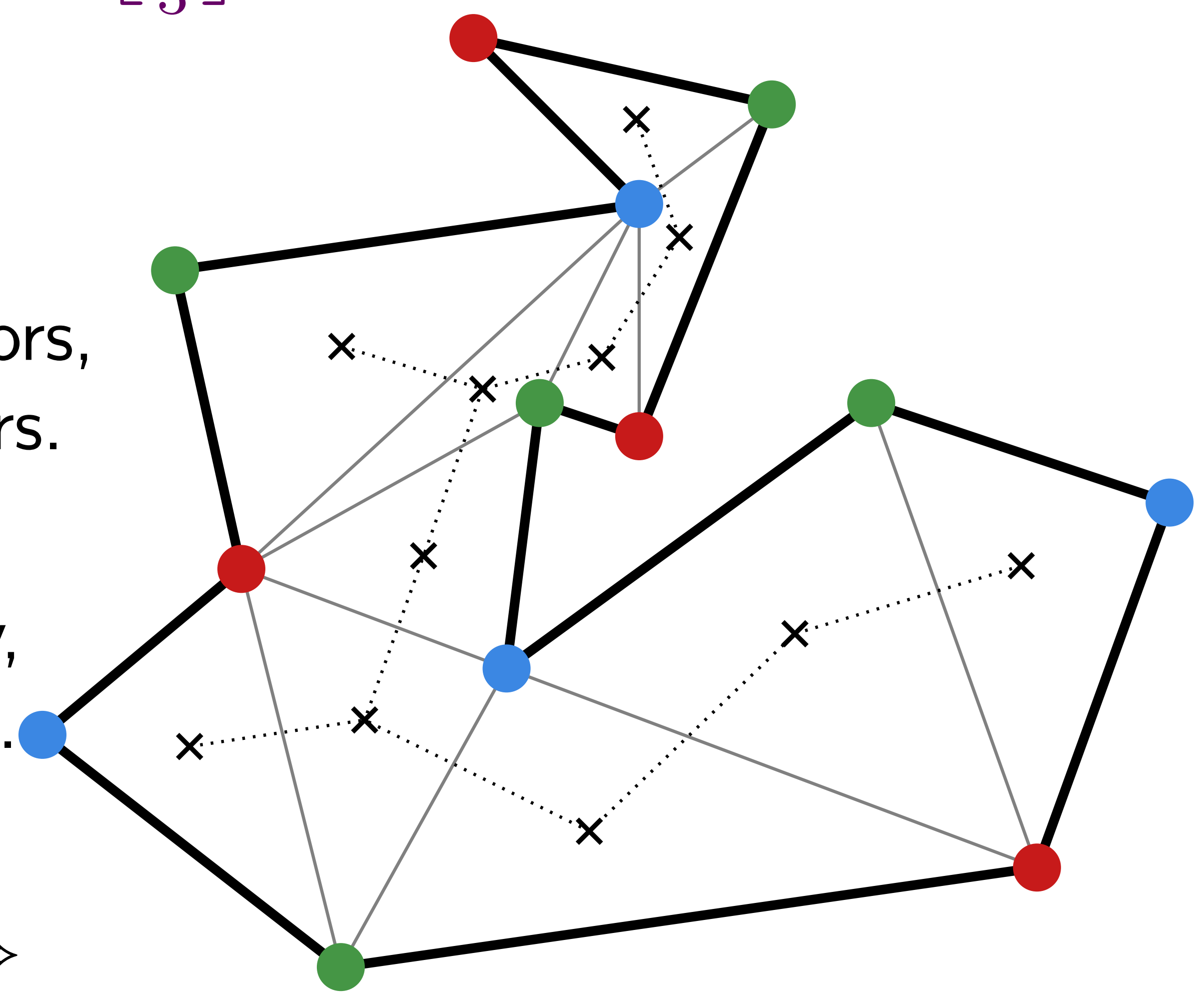
Proof that $\lfloor \frac{n}{3} \rfloor$ are sufficient

Consider a triangulation.

Consider dual tree.

Color vertices with 3 colors,
each triangle gets 3 colors.

Observe that the red vertices guard the gallery, as do the green and blue.



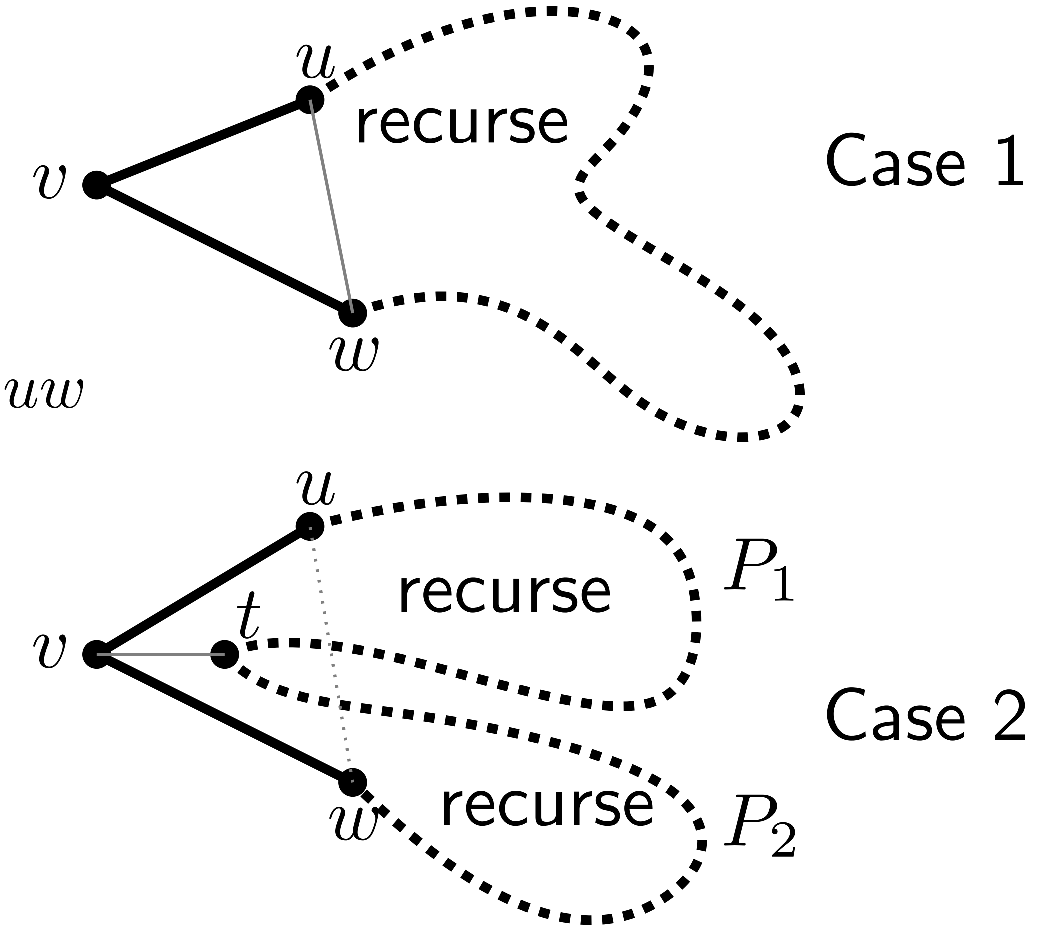
$$n = n_r + n_g + n_b \implies$$
$$\min\{n_r, n_g, n_b\} \leq \frac{n}{3} \implies$$
$$\min\{n_r, n_g, n_b\} \leq \lfloor \frac{n}{3} \rfloor$$

Why does the proof fail if P has holes?

Simple triangulation algorithm

Use proof that a triangulation exists!

triangulate polygon P
1. find points u, v, w
2. if uw is a diagonal // case 1
3. add diagonal uw
4. recurse on the other side of uw
5. else // case 2
6. find point t
7. add diagonal vt
8. recurse on P_1 and P_2

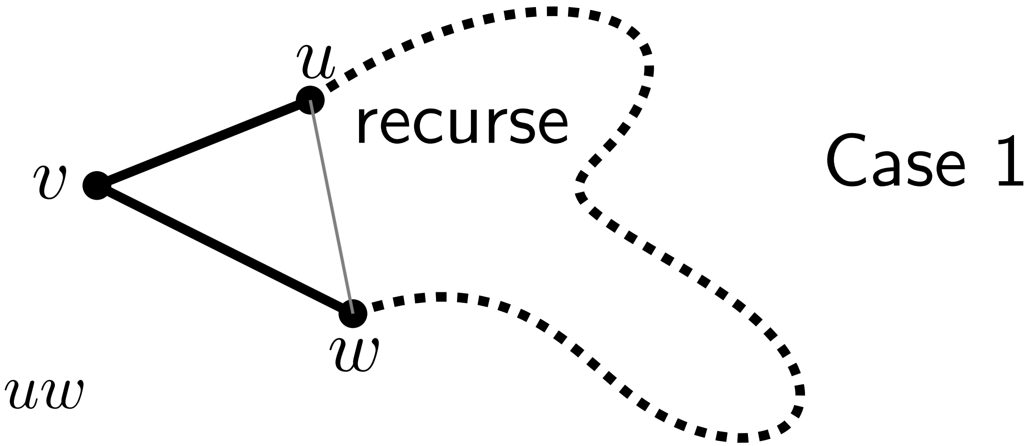


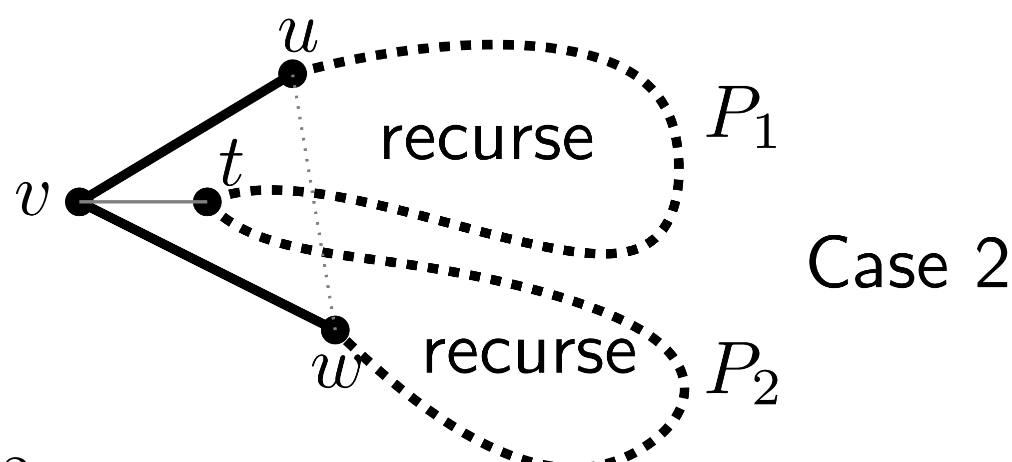
Simple triangulation algorithm

Use proof that a triangulation exists!

triangulate polygon P

1. find points u, v, w
2. if uw is a diagonal // case 1
3. add diagonal uw
4. recurse on the other side of uw
5. else // case 2
6. find point t
7. add diagonal vt
8. recurse on P_1 and P_2

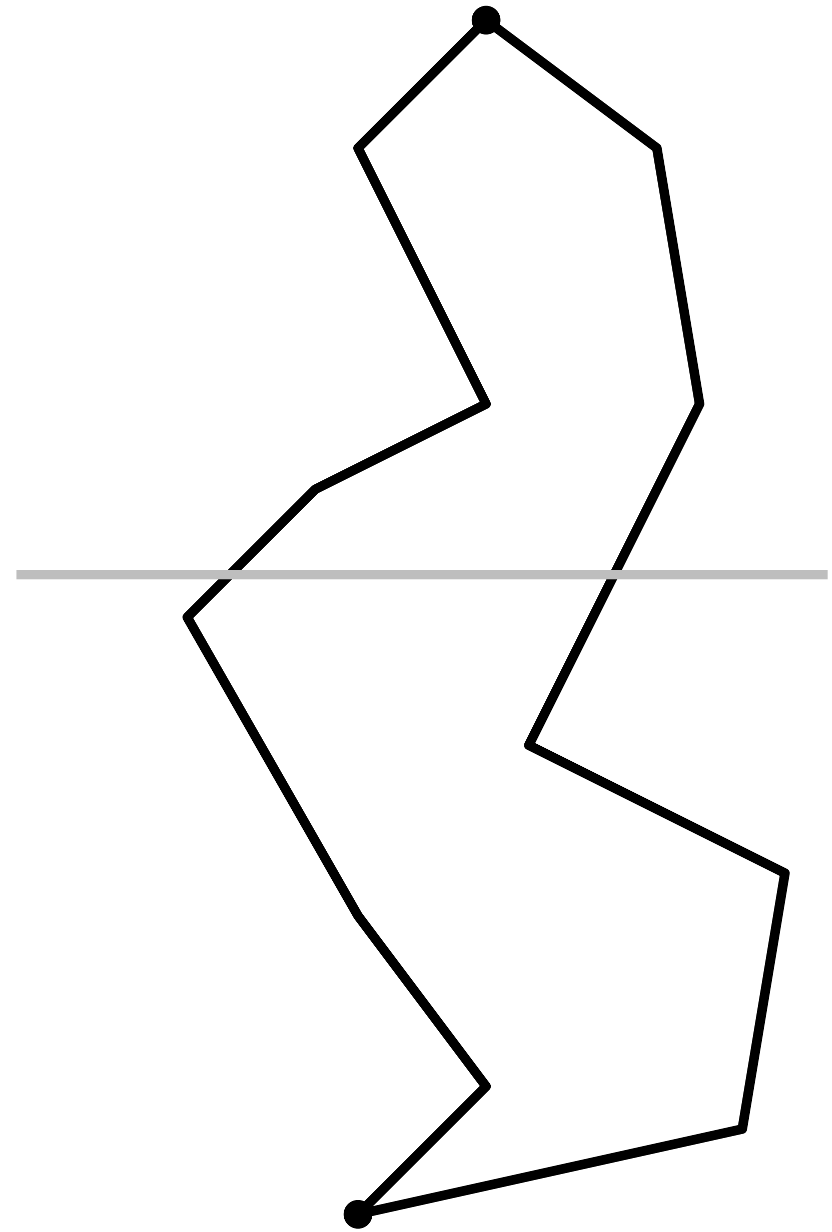




Worst-case running time: $O(n^2)$.

Faster algorithm

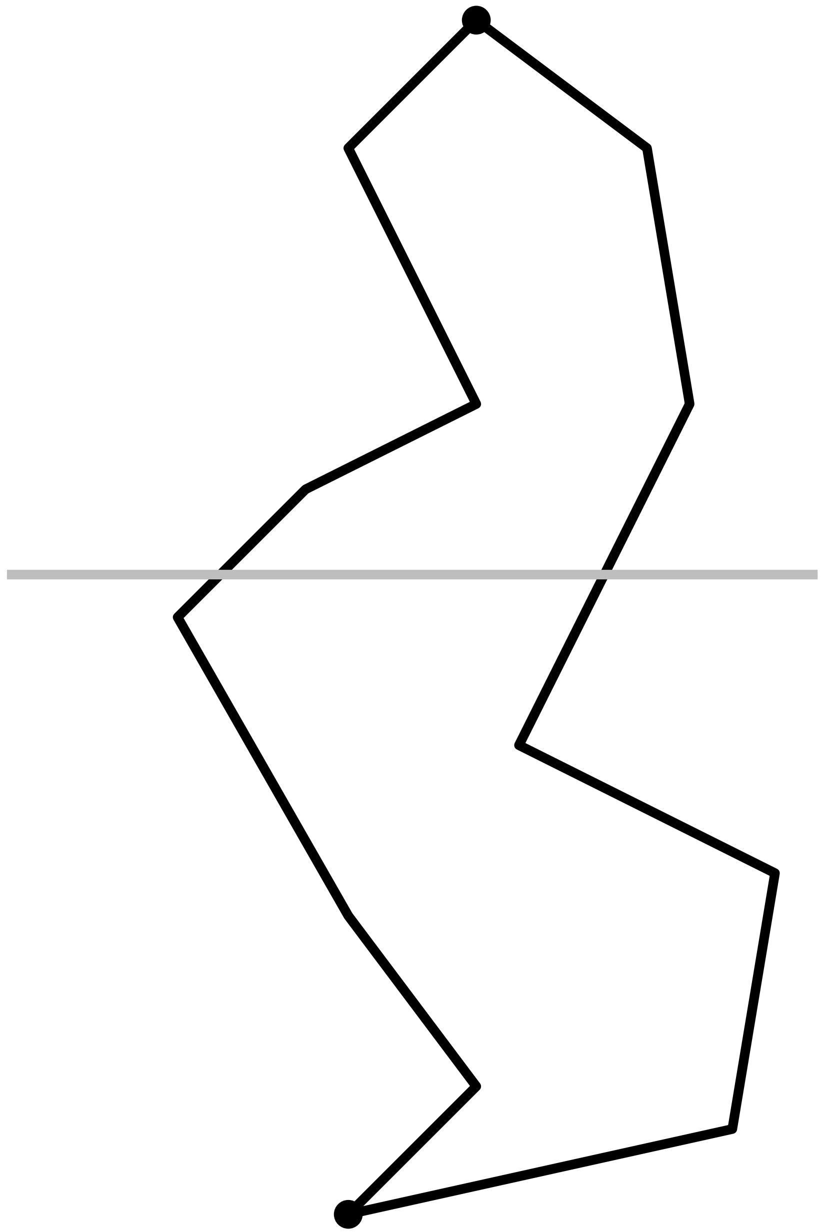
y-monotone polygon: Intersection with any horizontal line is connected.
Equivalent: One vertex with both edges going down and one with both edges going up.



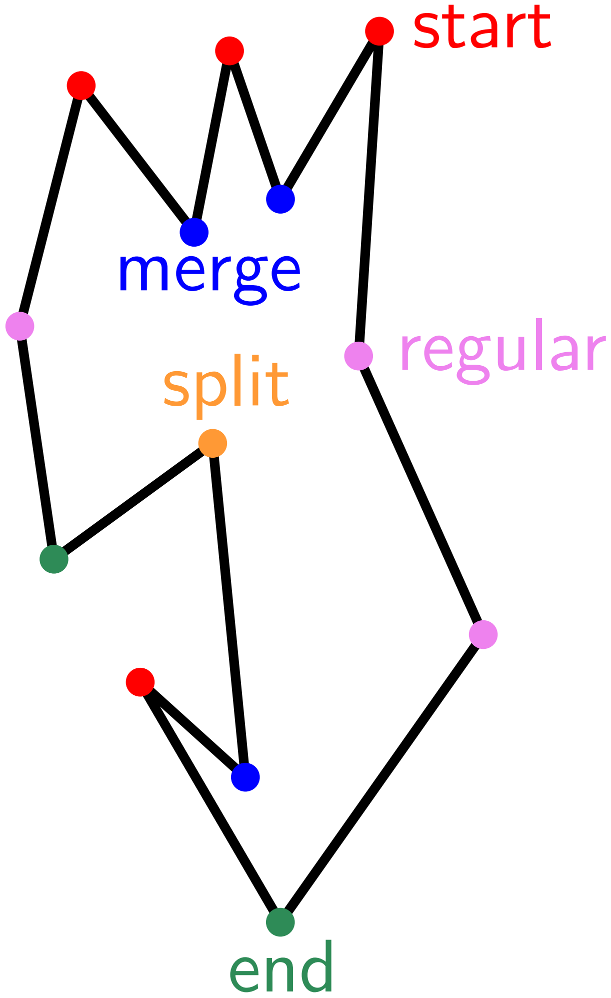
Faster algorithm

y-monotone polygon: Intersection with any horizontal line is connected.
Equivalent: One vertex with both edges going down and one with both edges going up.

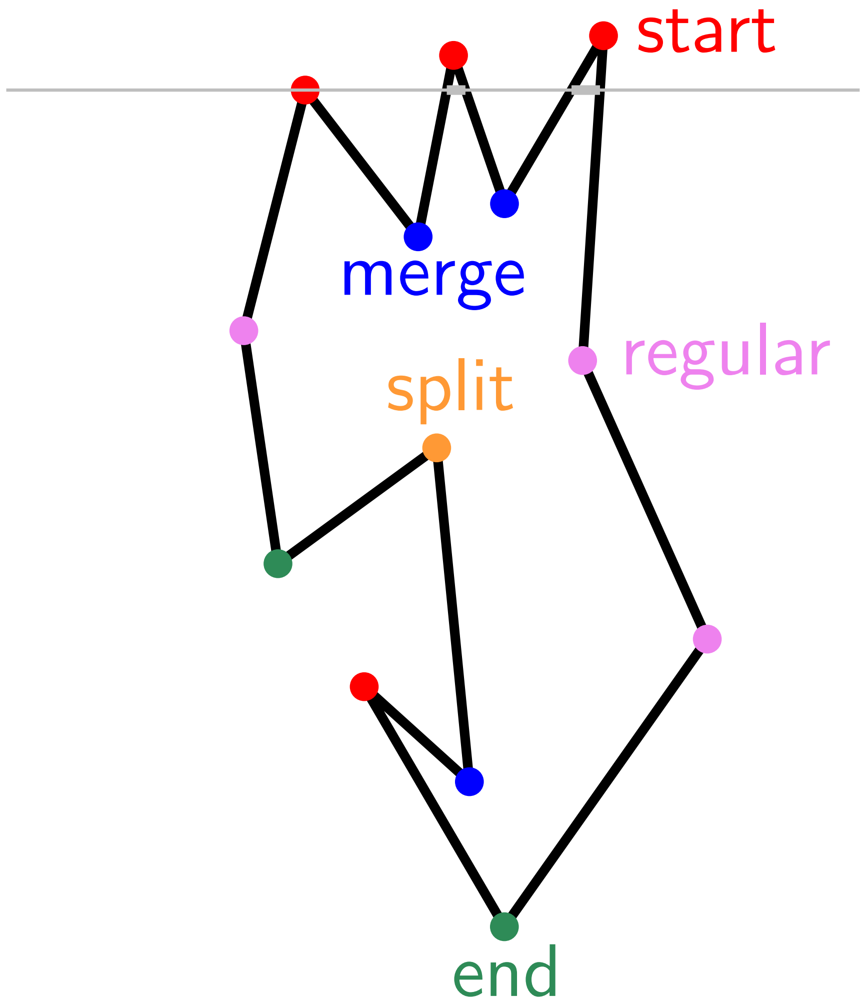
Idea:
Phase 1: Split polygon P into y-monotone polygons.
Phase 2: Triangulate each y-monotone polygon.



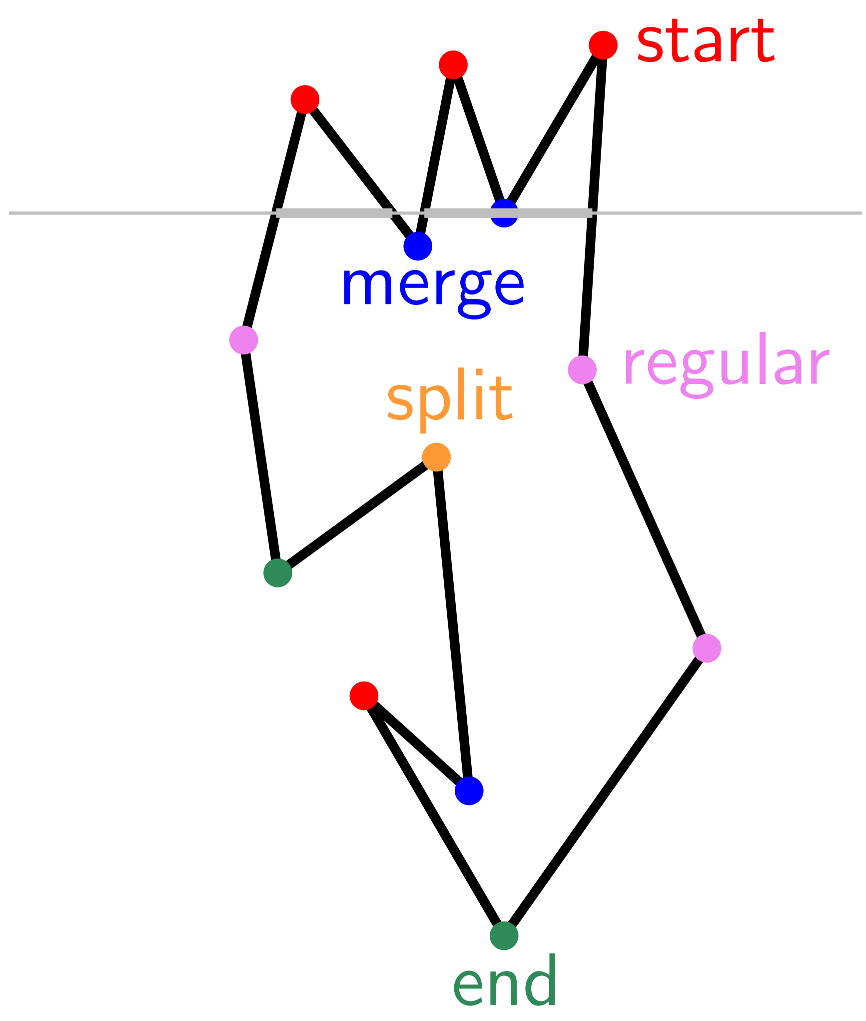
4 types of vertices



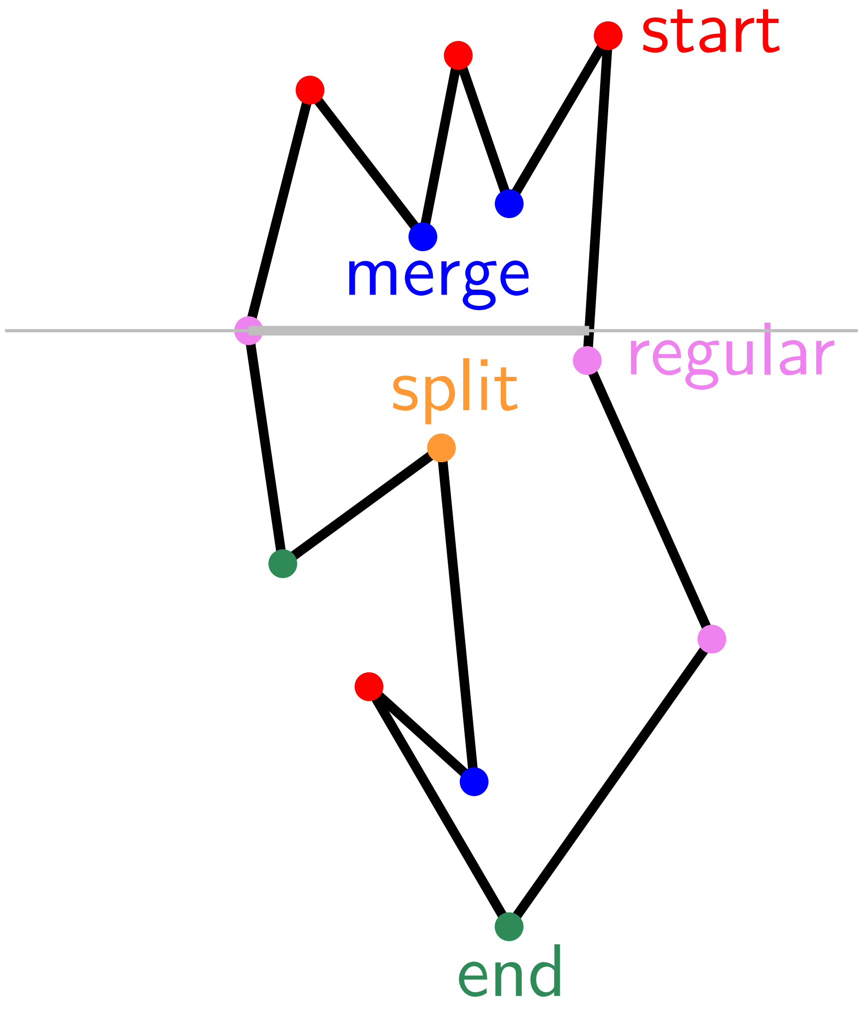
4 types of vertices



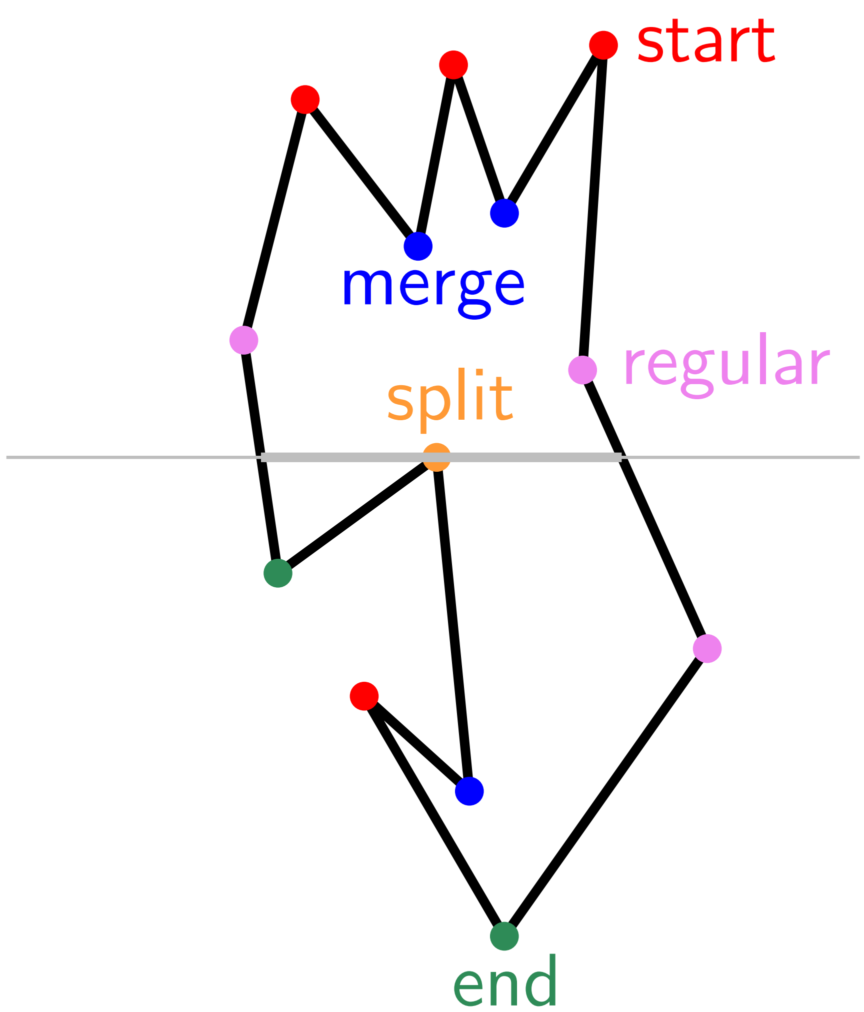
4 types of vertices



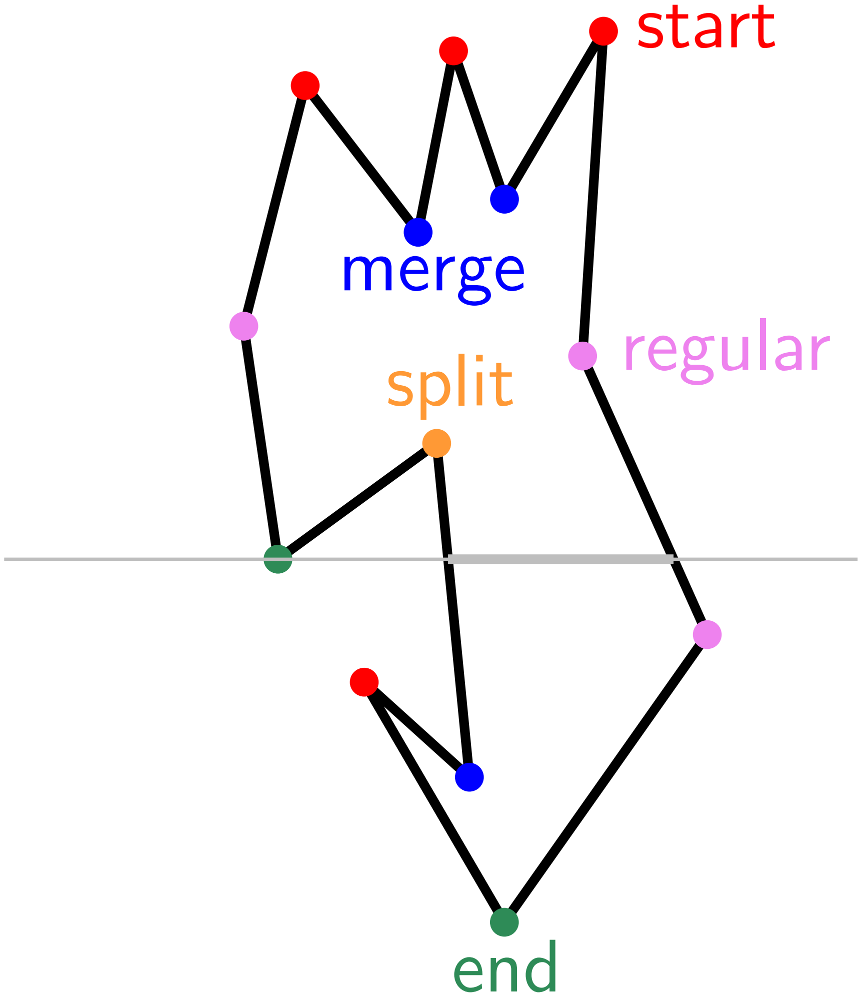
4 types of vertices



4 types of vertices

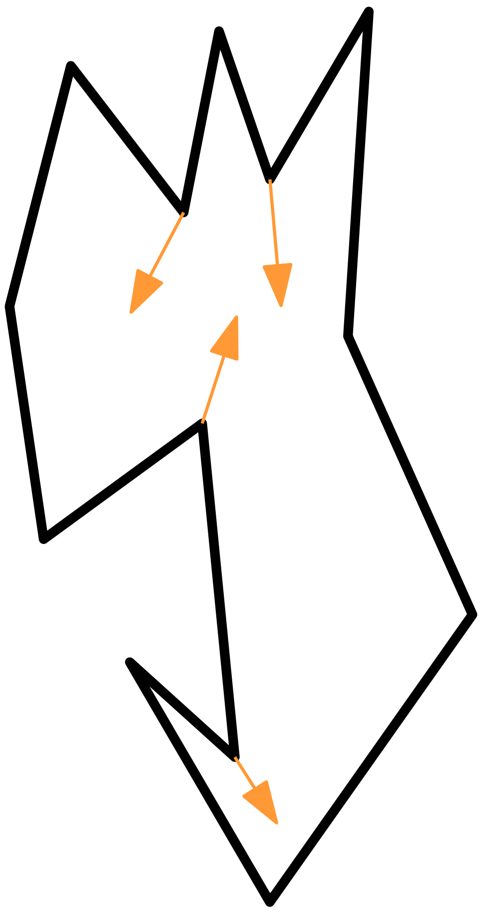


4 types of vertices



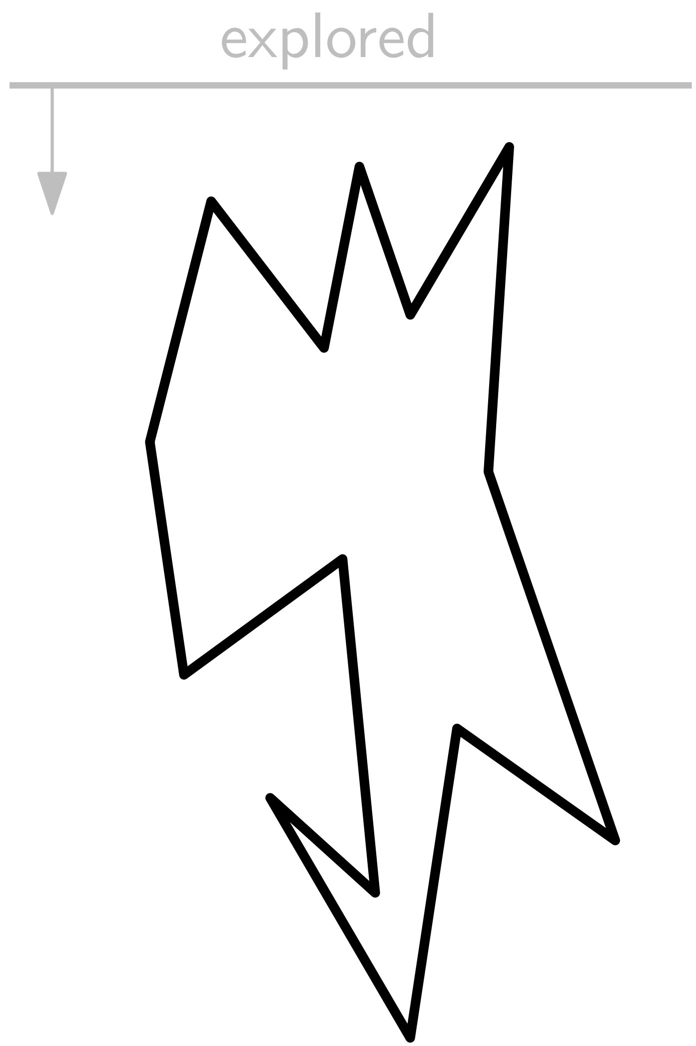
Idea

Idea: “Shoot” a diagonal from every split and merge vertex.
No split and merge $\Rightarrow$ monotone polygon!



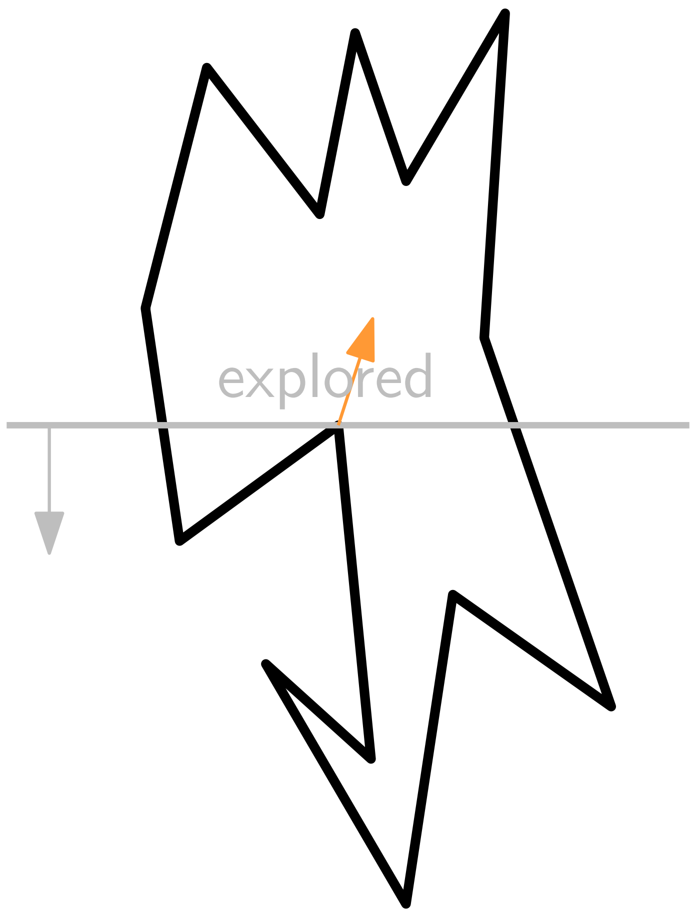
Idea

Use horizontal sweep line going down.
Fix split vertices first.
Then go up and fix merge vertices.



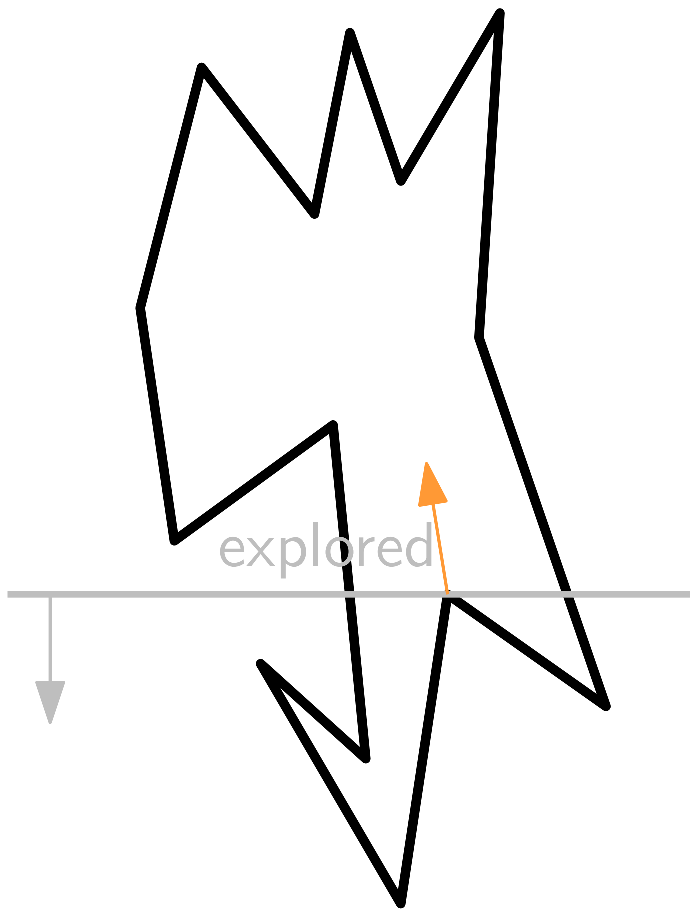
Idea

Use horizontal sweep line going down.
Fix split vertices first.
Then go up and fix merge vertices.



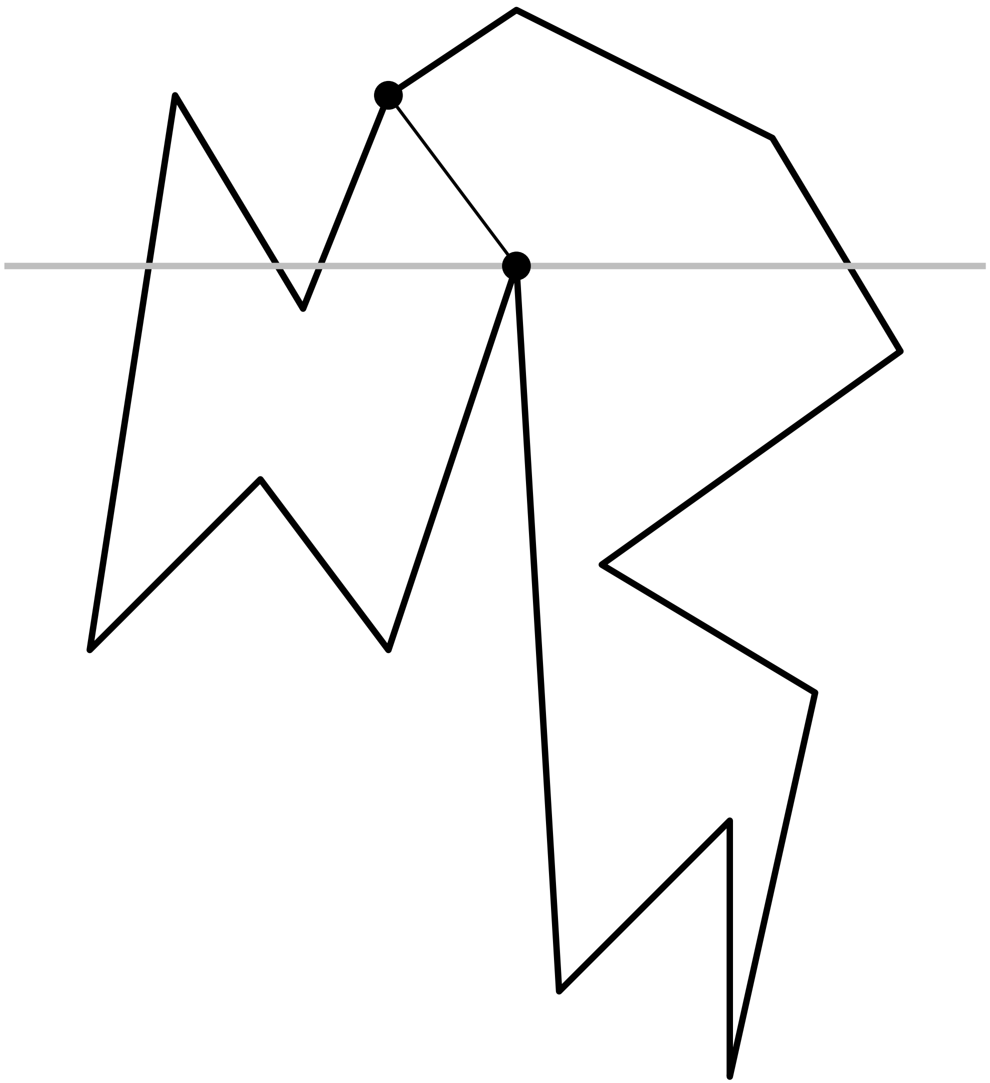
Idea

Use horizontal sweep line going down.
Fix split vertices first.
Then go up and fix merge vertices.



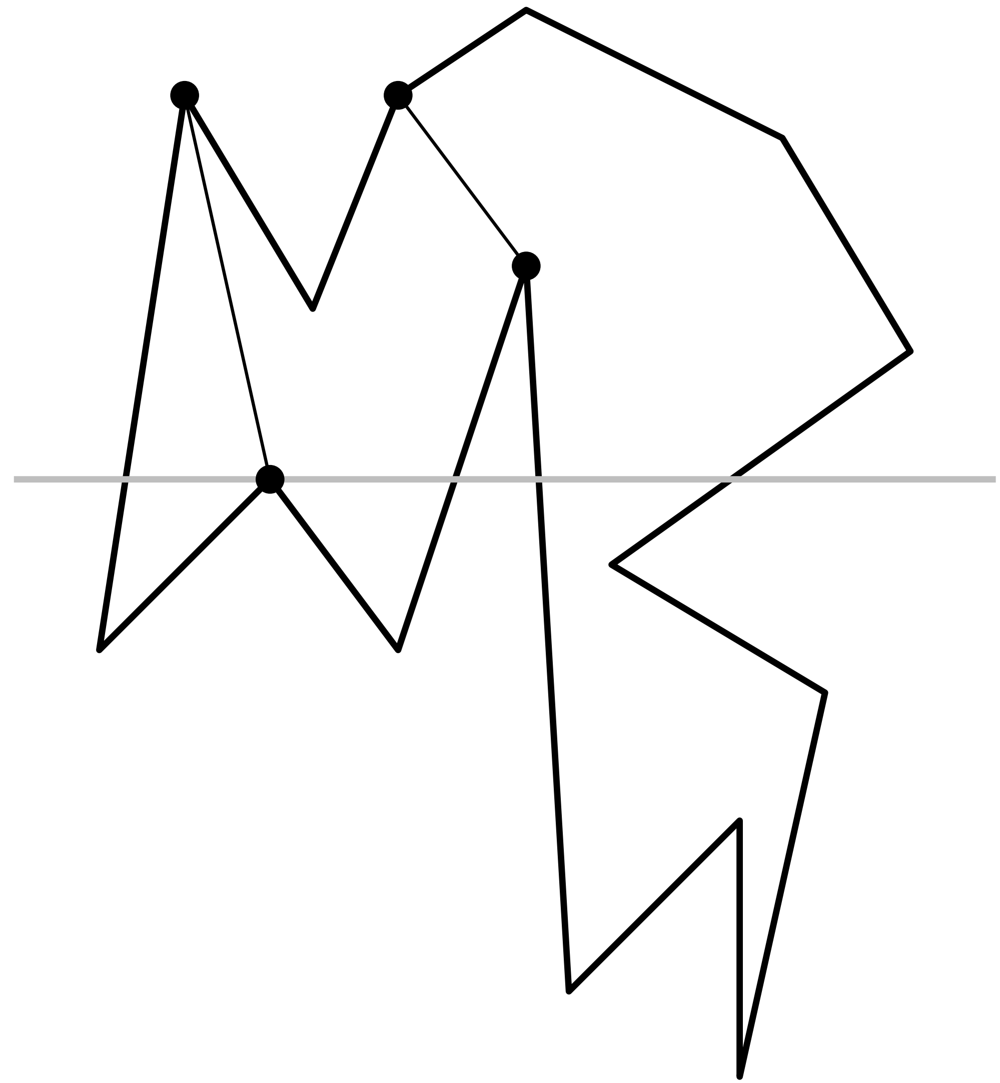
First idea

Make segment to top vertex of line segment to the left?



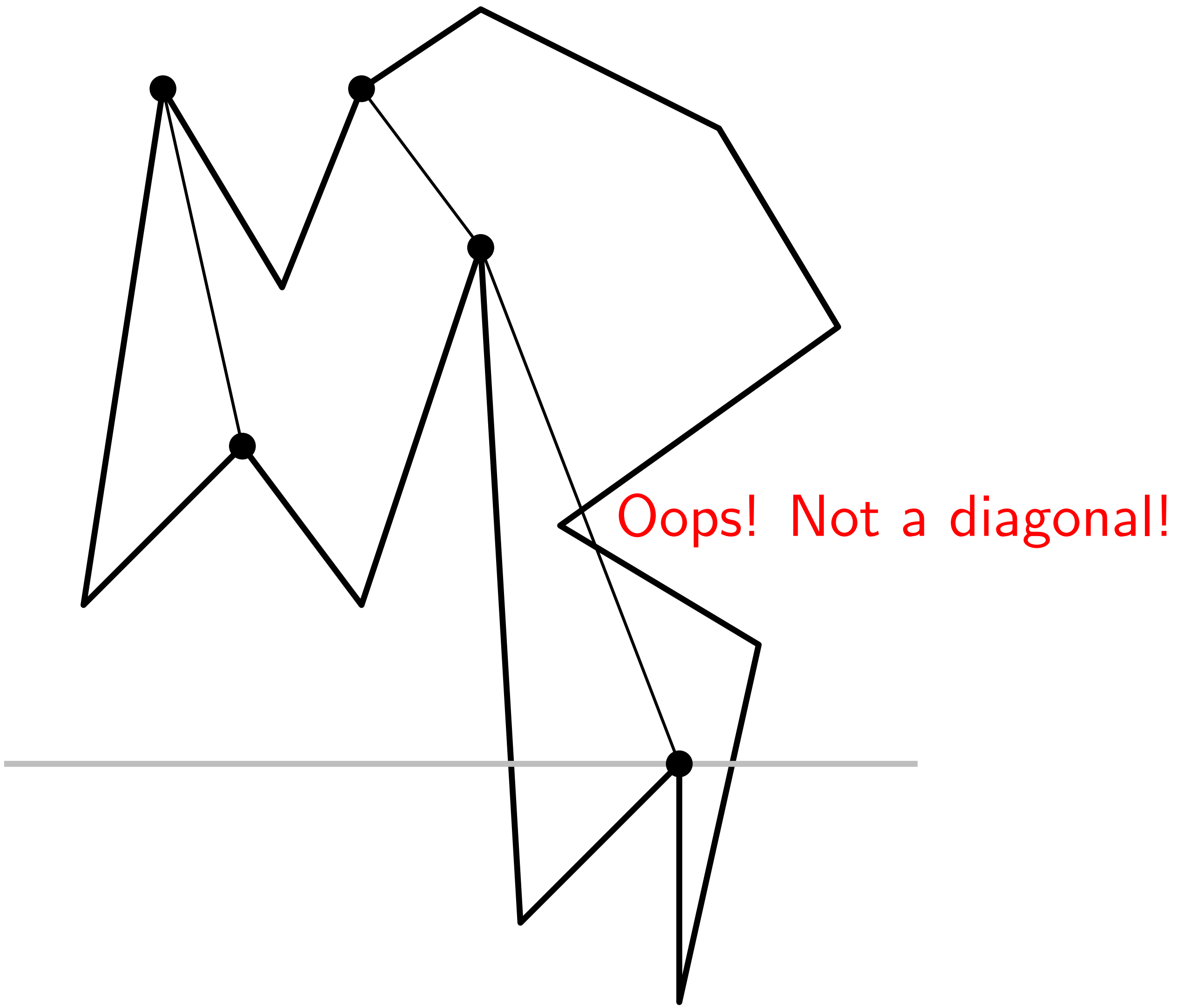
First idea

Make segment to top vertex of line segment to the left?



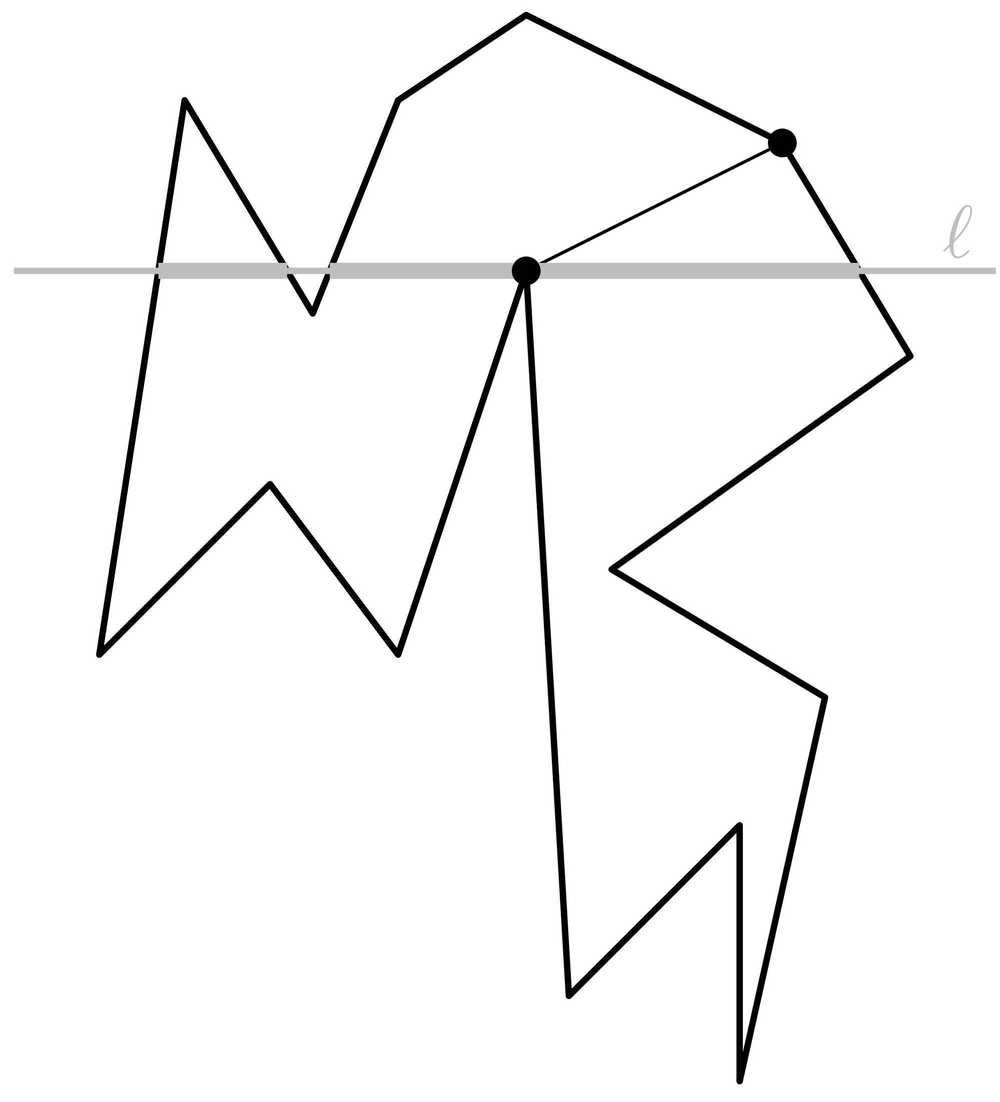
First idea

Make segment to top vertex of line segment to the left?



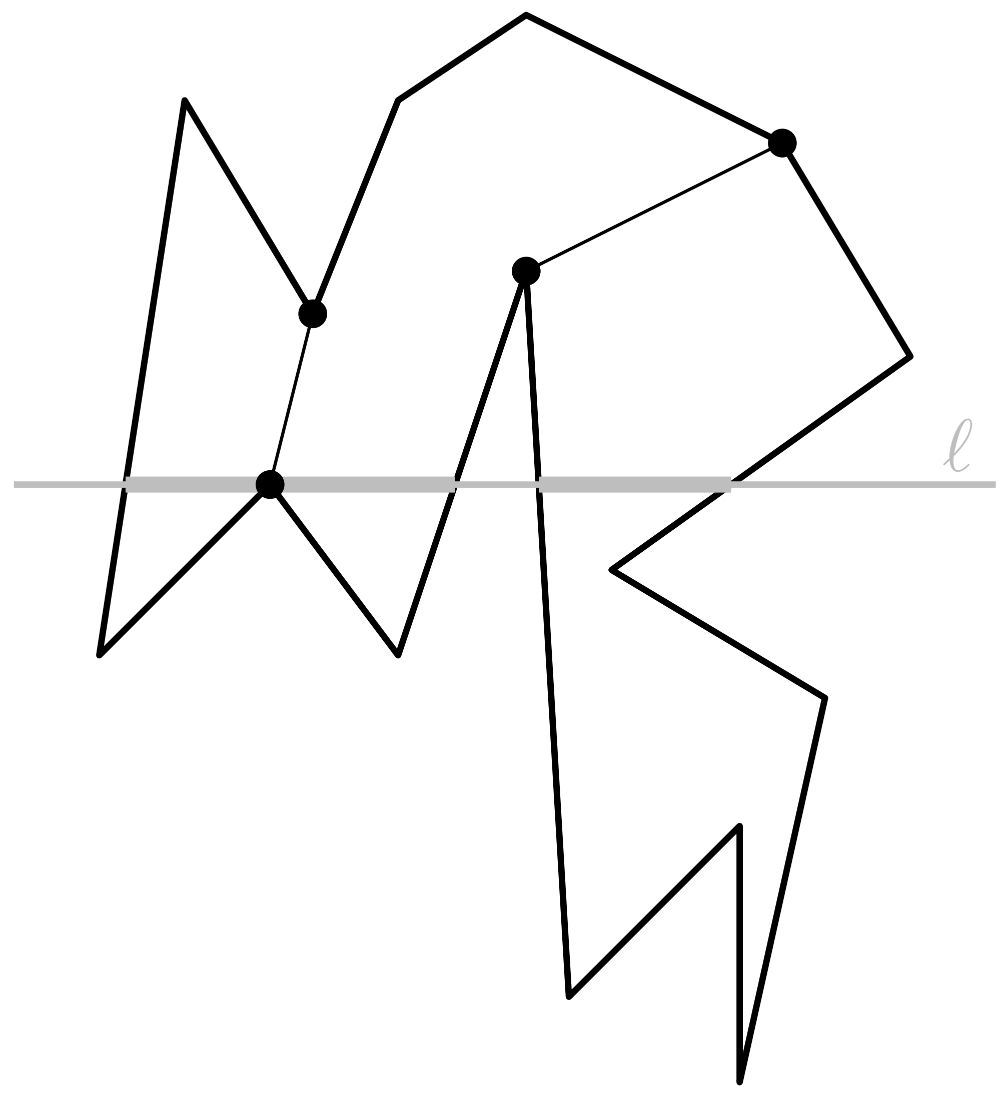
Refined idea

Make segment to previous vertex visited by the component of $\ell \cap P$.



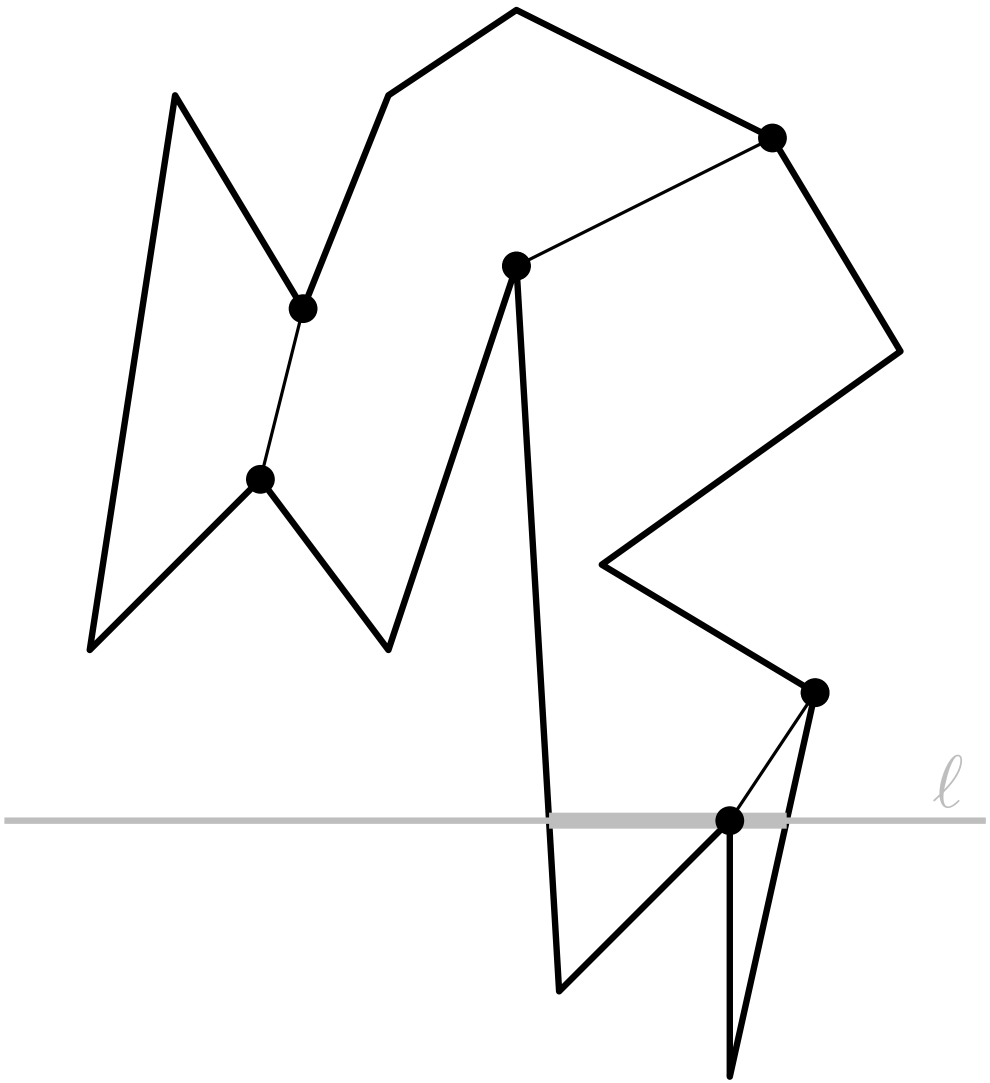
Refined idea

Make segment to previous vertex visited by the component of $\ell \cap P$.



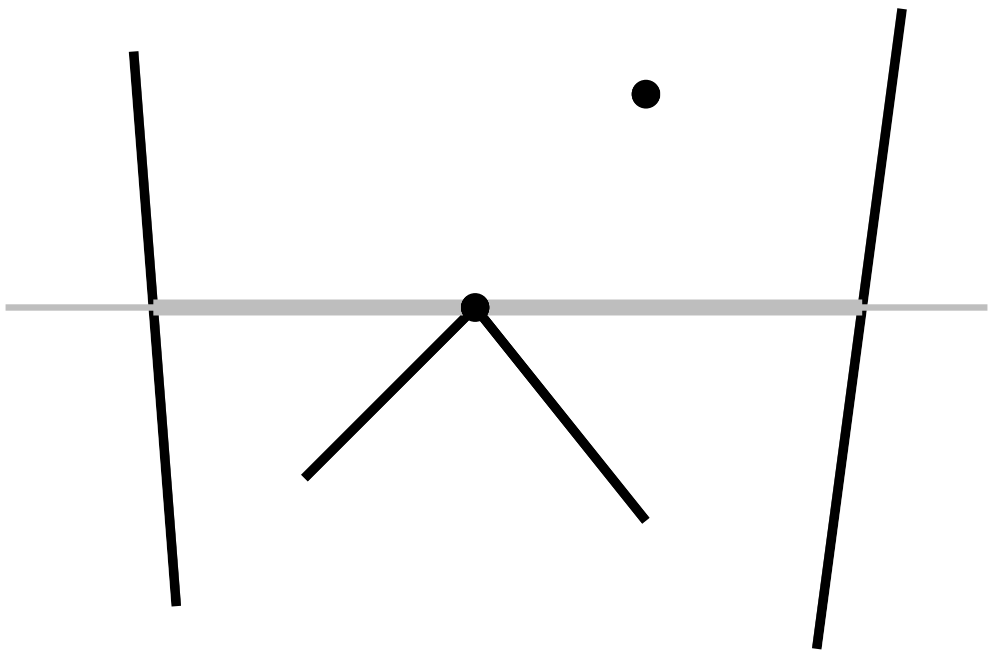
Refined idea

Make segment to previous vertex visited by the component of $\ell \cap P$.

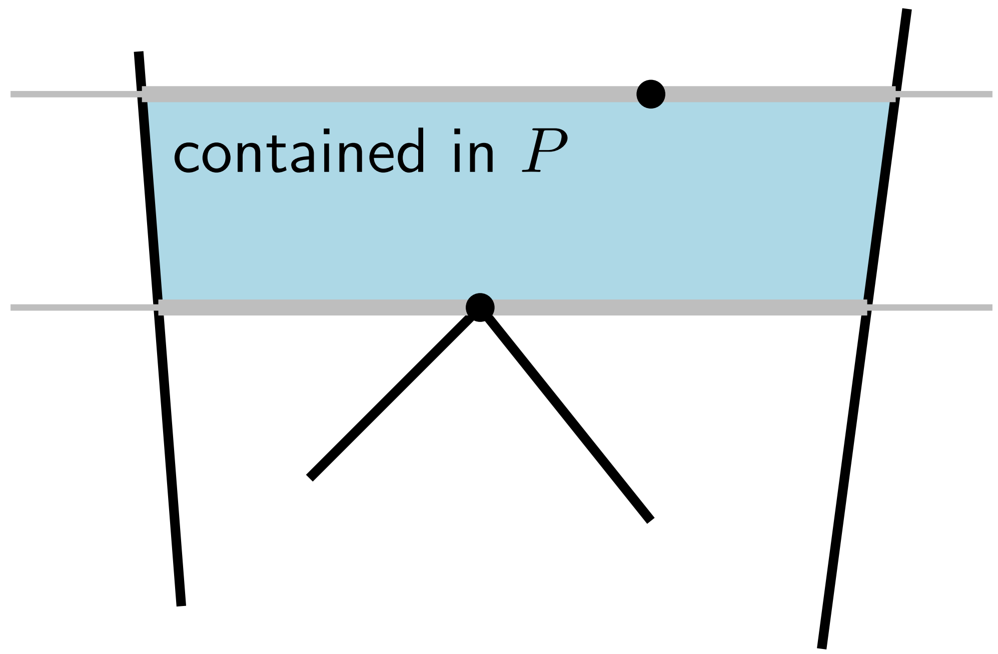


Always a diagonal?
Non-intersecting diagonals?

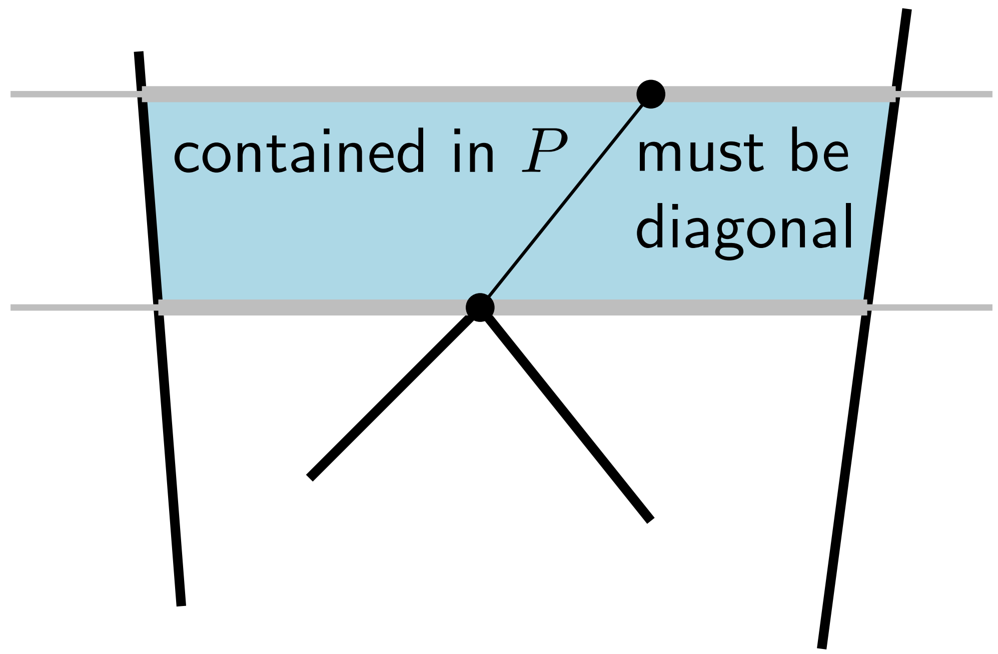
Idea works!



Idea works!

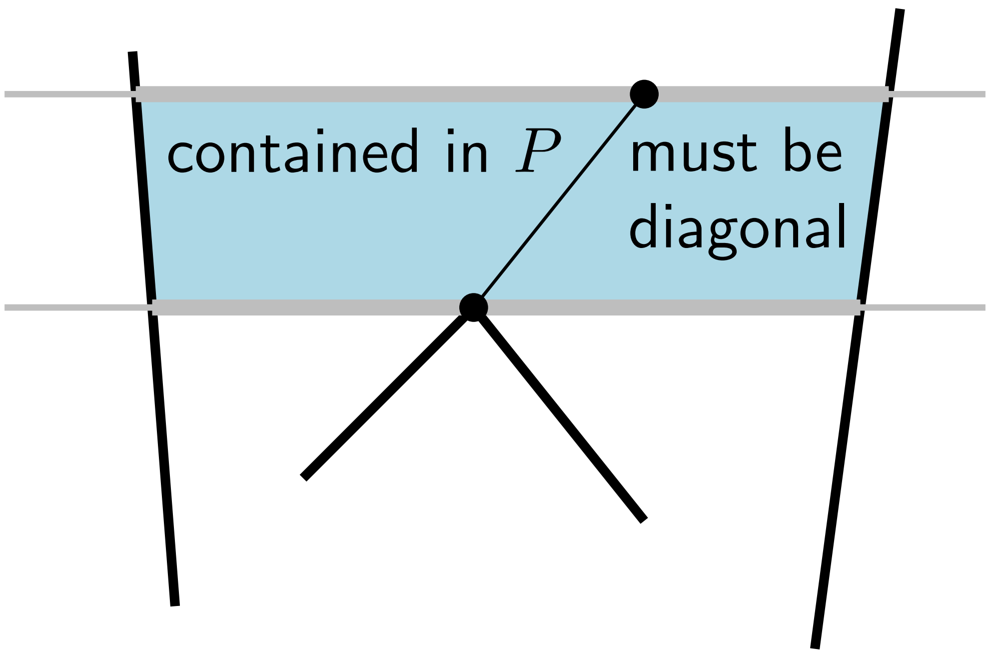


Idea works!



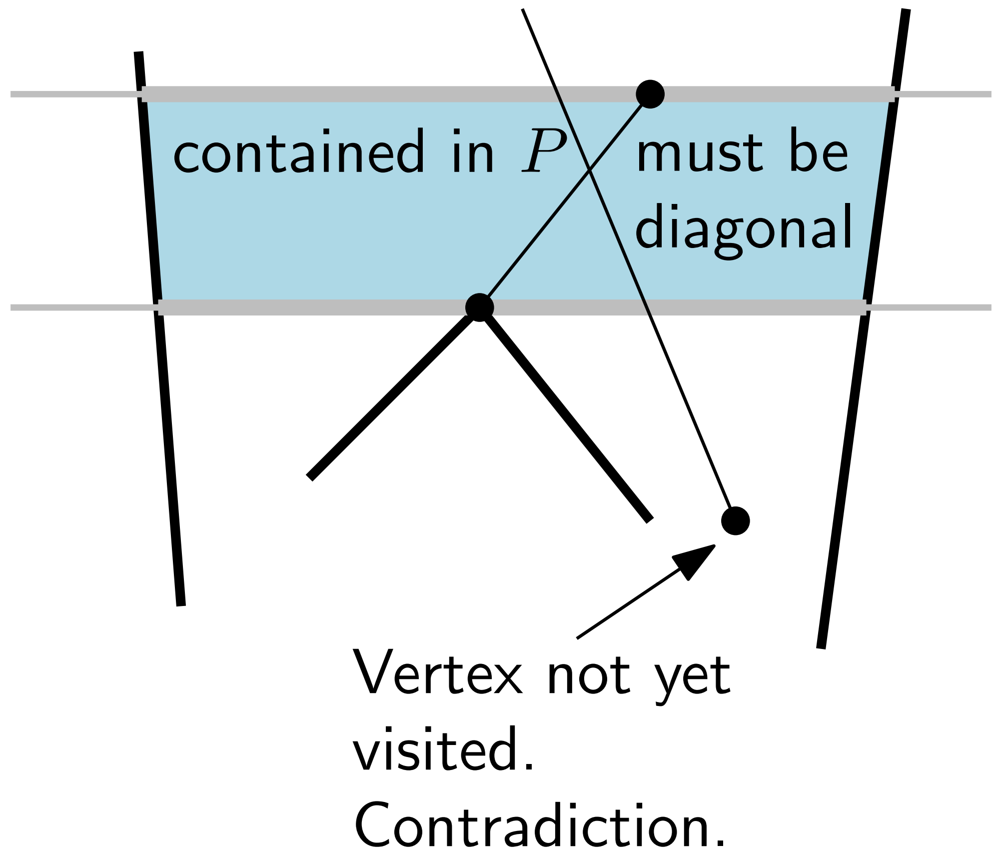
Idea works!

Can new diagonal intersect a previously added?



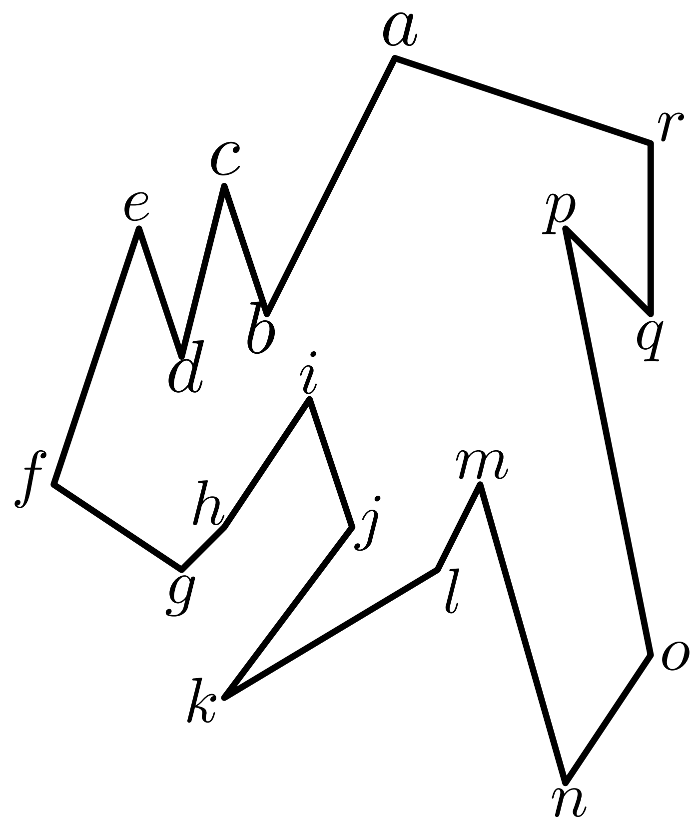
Idea works!

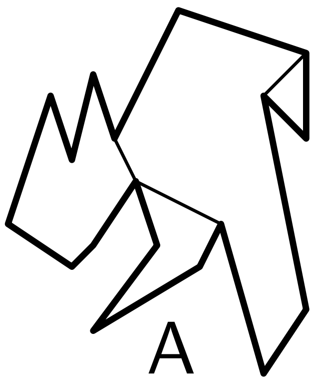
Can new diagonal intersect a previously added?



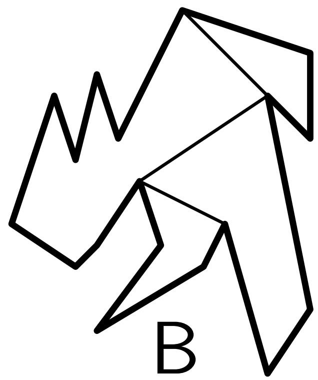
What diagonals are introduced after sweeping down?

socrative.com $\rightarrow$ Student login,
Room name: ABRAHAMSEN3464

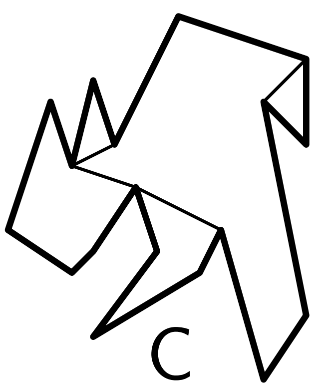




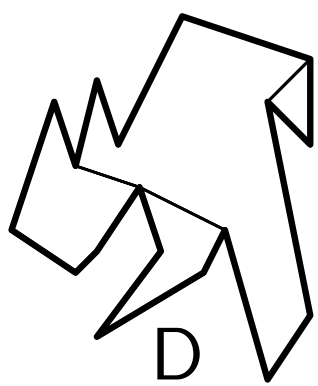
A



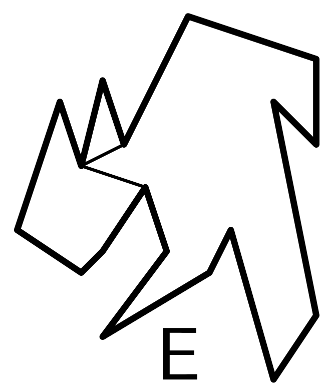
B



C



D

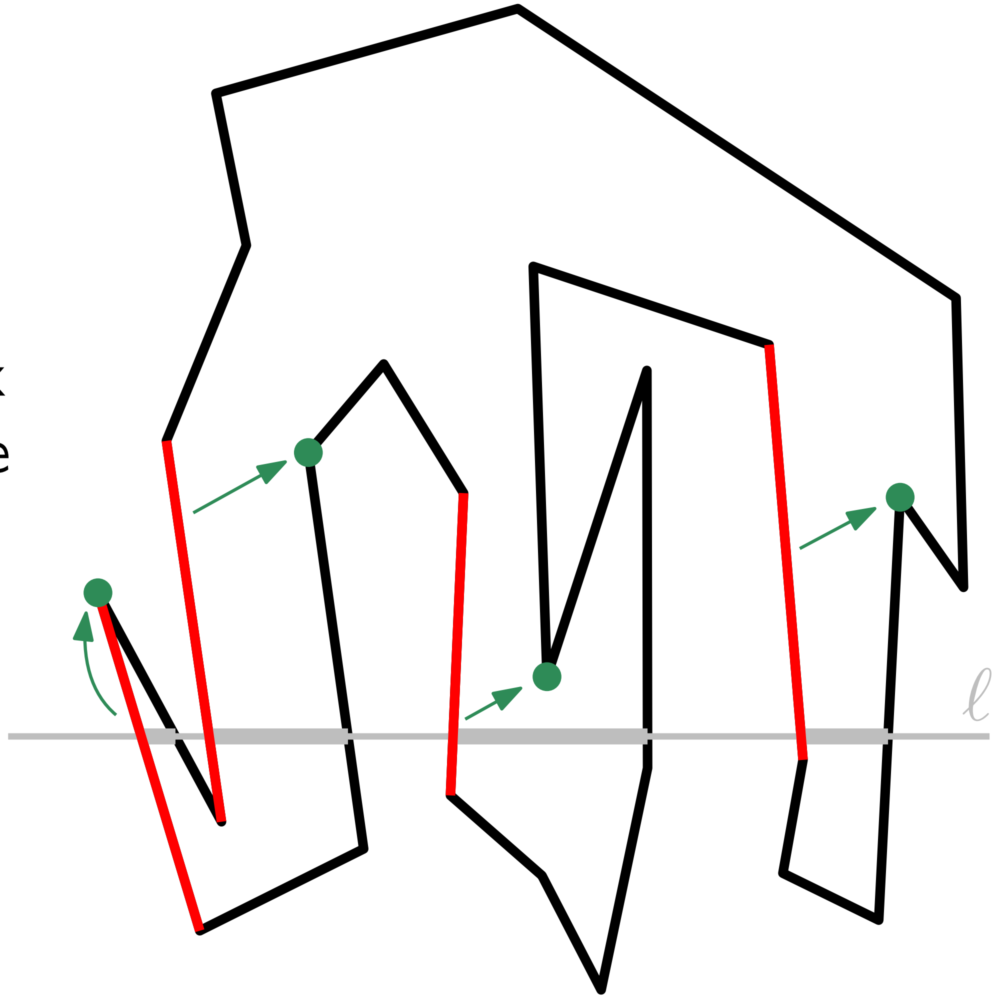


E

Helpers

Helper of edge e intersected by ℓ with interior of P to the right: Previous vertex visited by this connected component of $\ell \cap P$.

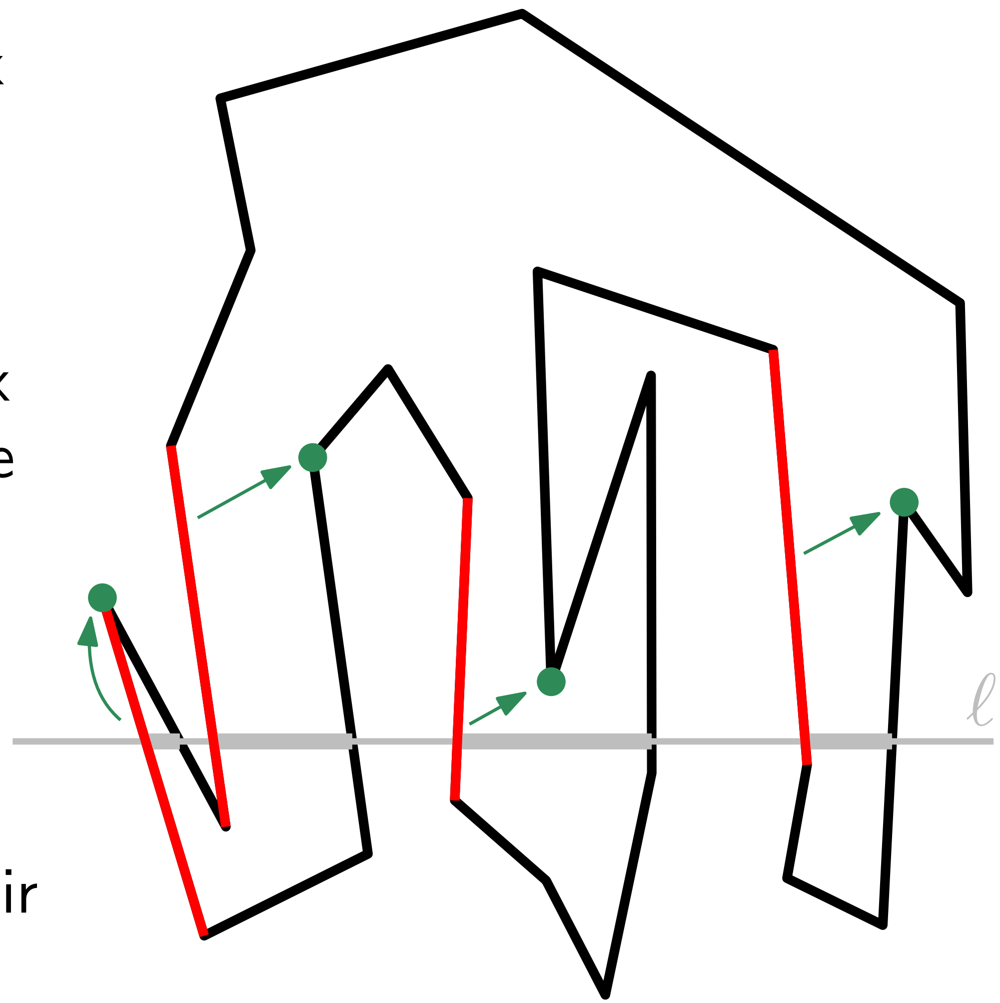
Equivalent: Lowest vertex above ℓ that sees e to the left.



Helpers

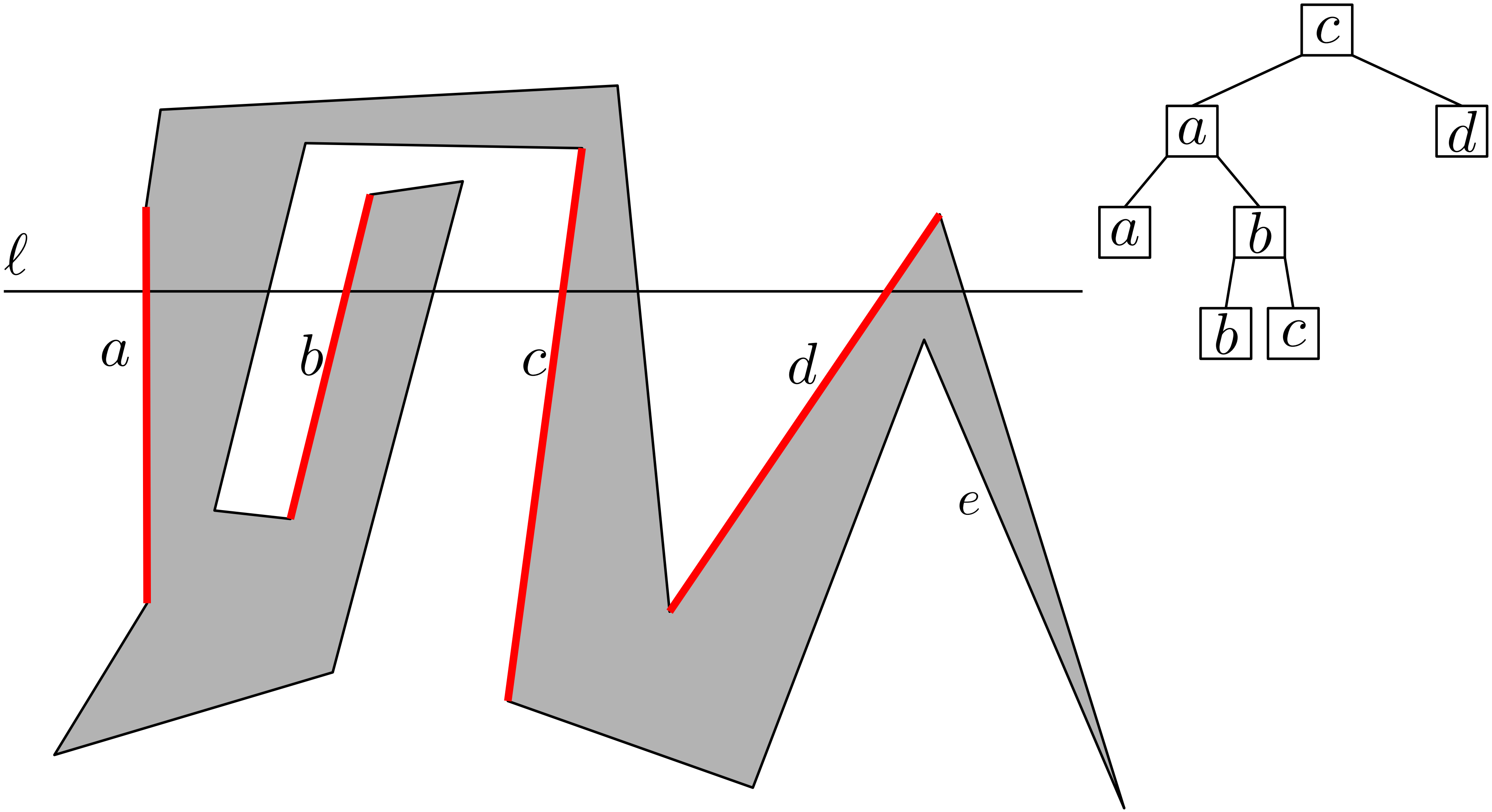
Helper of edge e intersected by ℓ with interior of P to the right: Previous vertex visited by this connected component of $\ell \cap P$.

Equivalent: Lowest vertex above ℓ that sees e to the left.



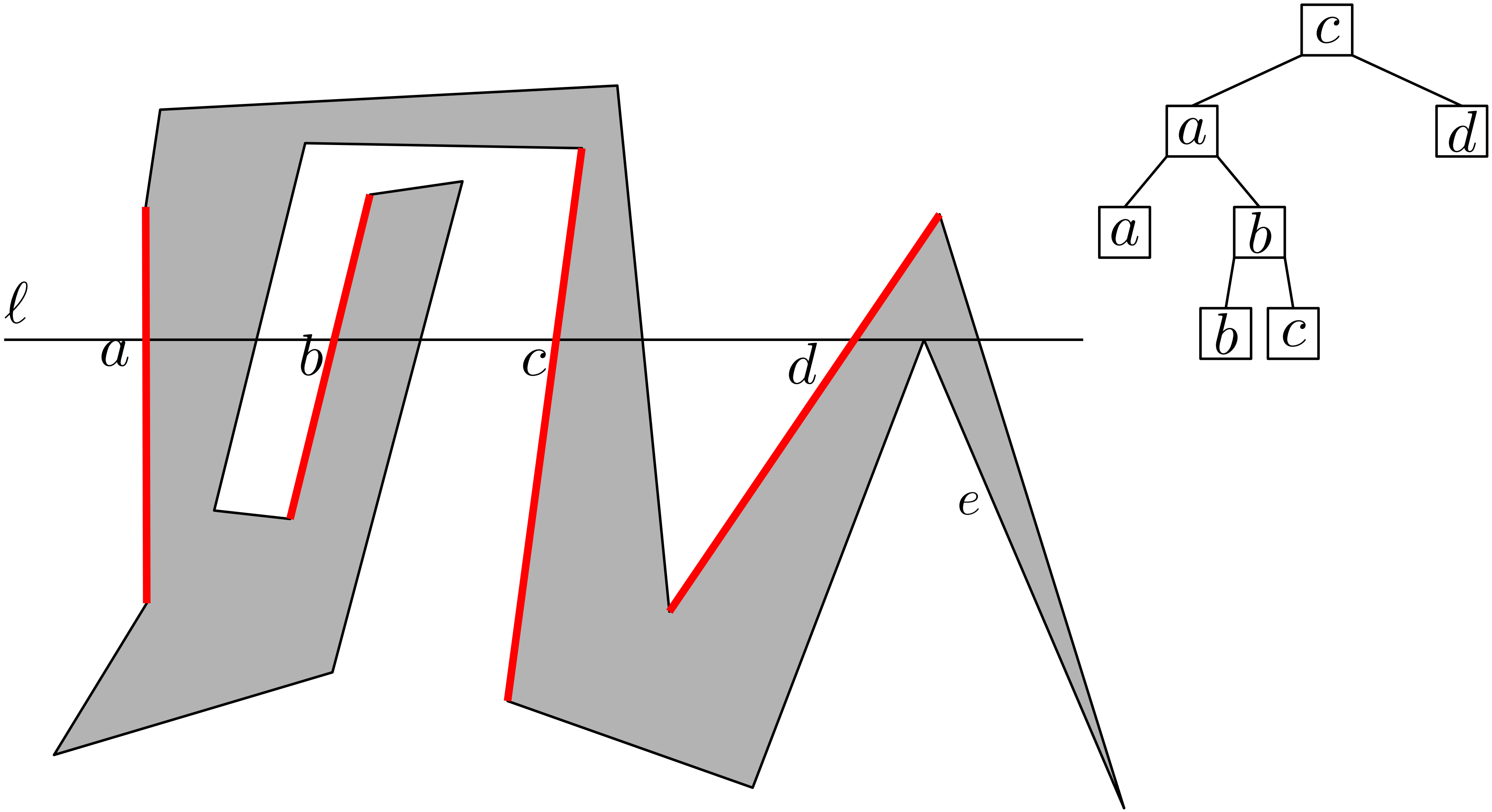
Intersected edges and their helpers are stored in the *status* structure T.

Status



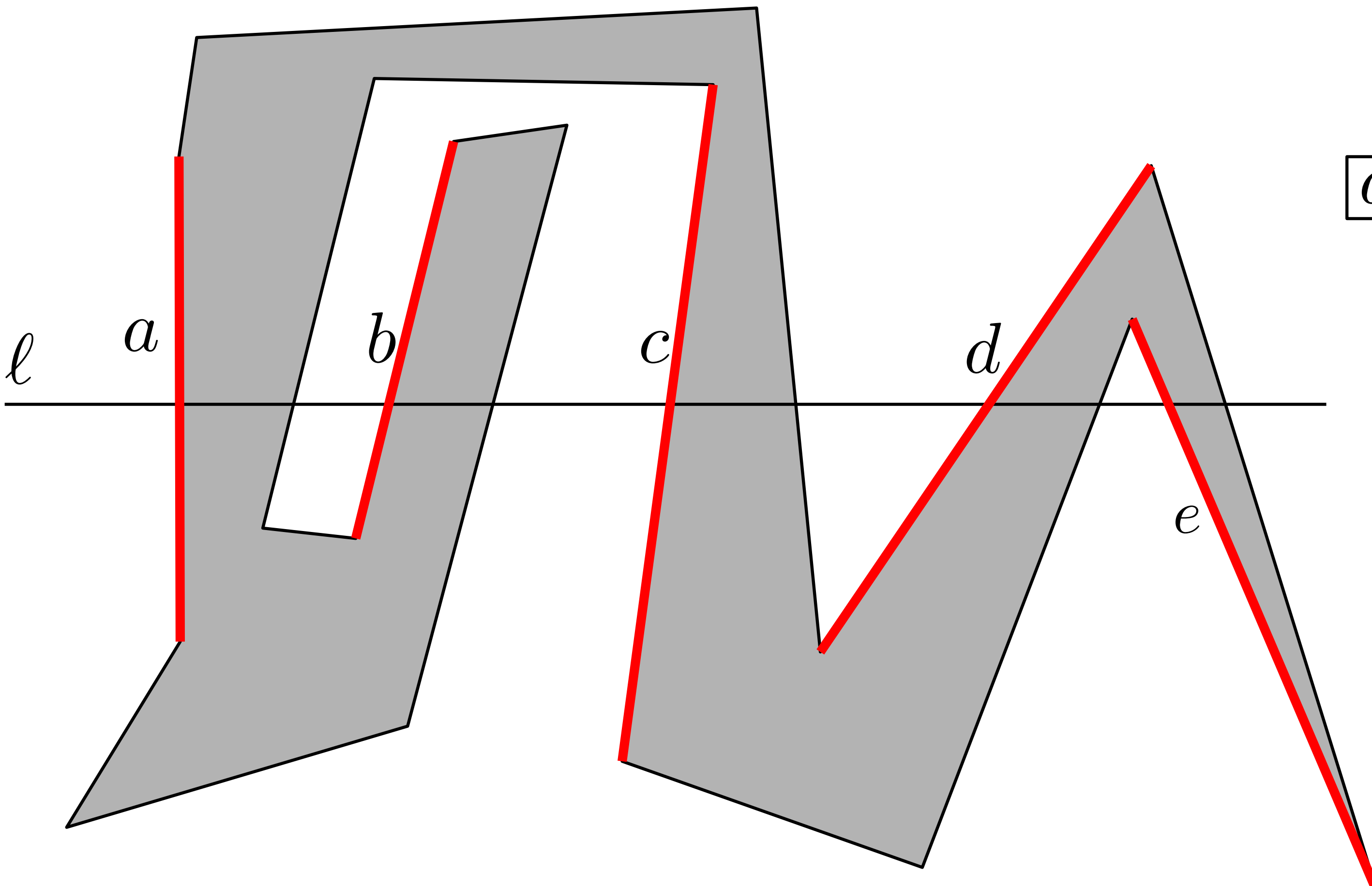
Implement status with balanced binary search tree T.
Sorting order: intersection points with ℓ from left to right.

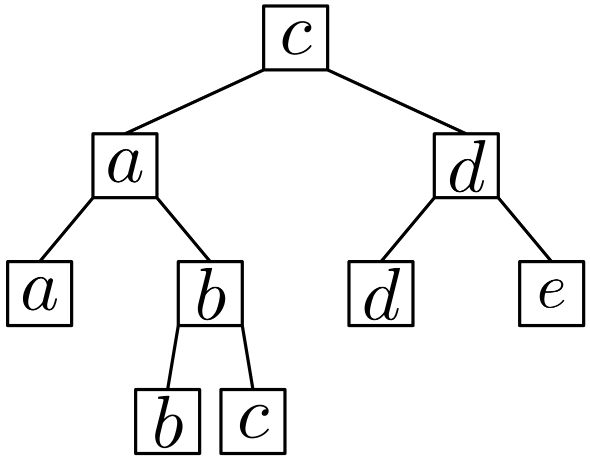
Status



Implement status with balanced binary search tree T.
Sorting order: intersection points with ℓ from left to right.

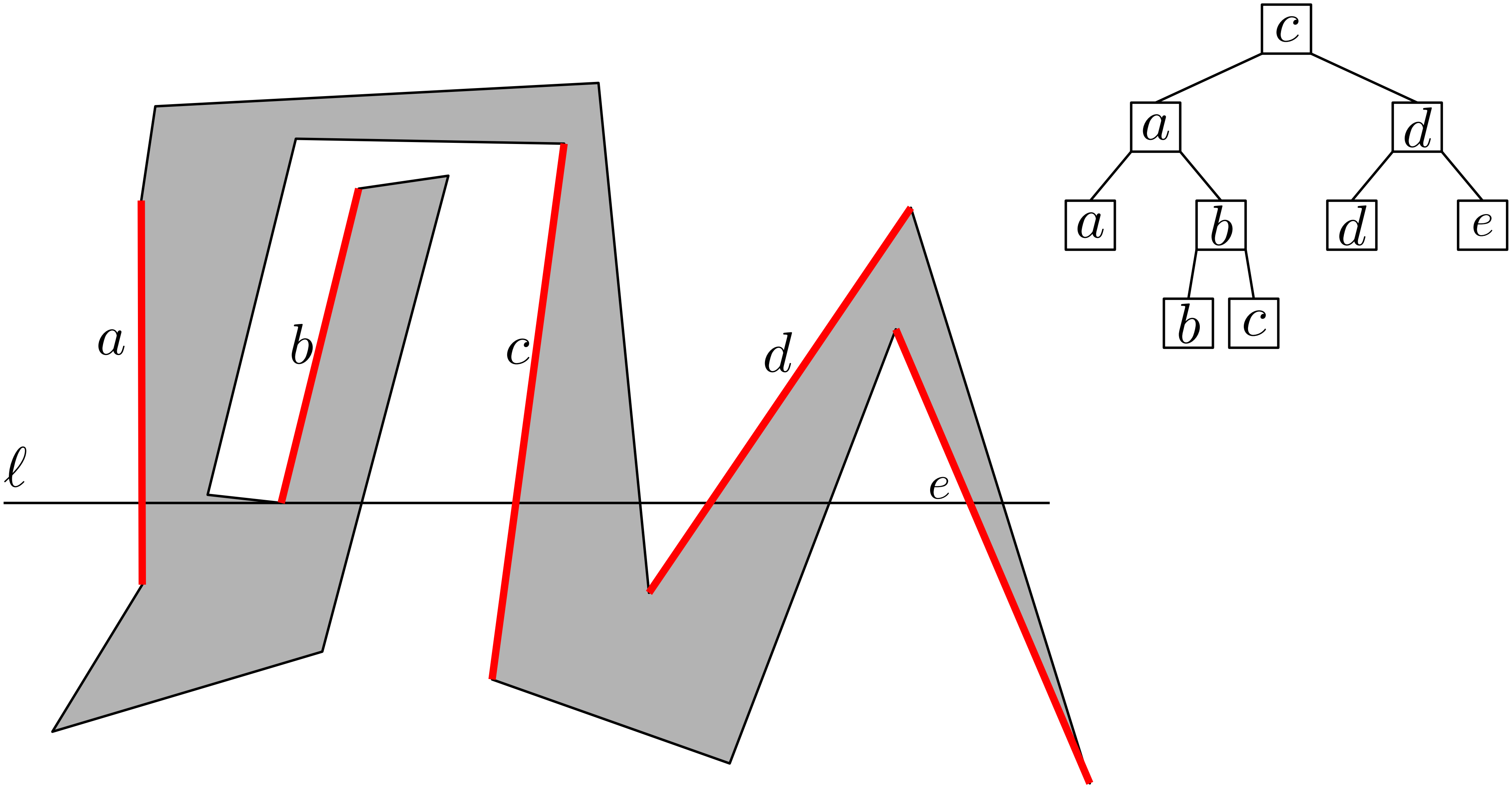
Status





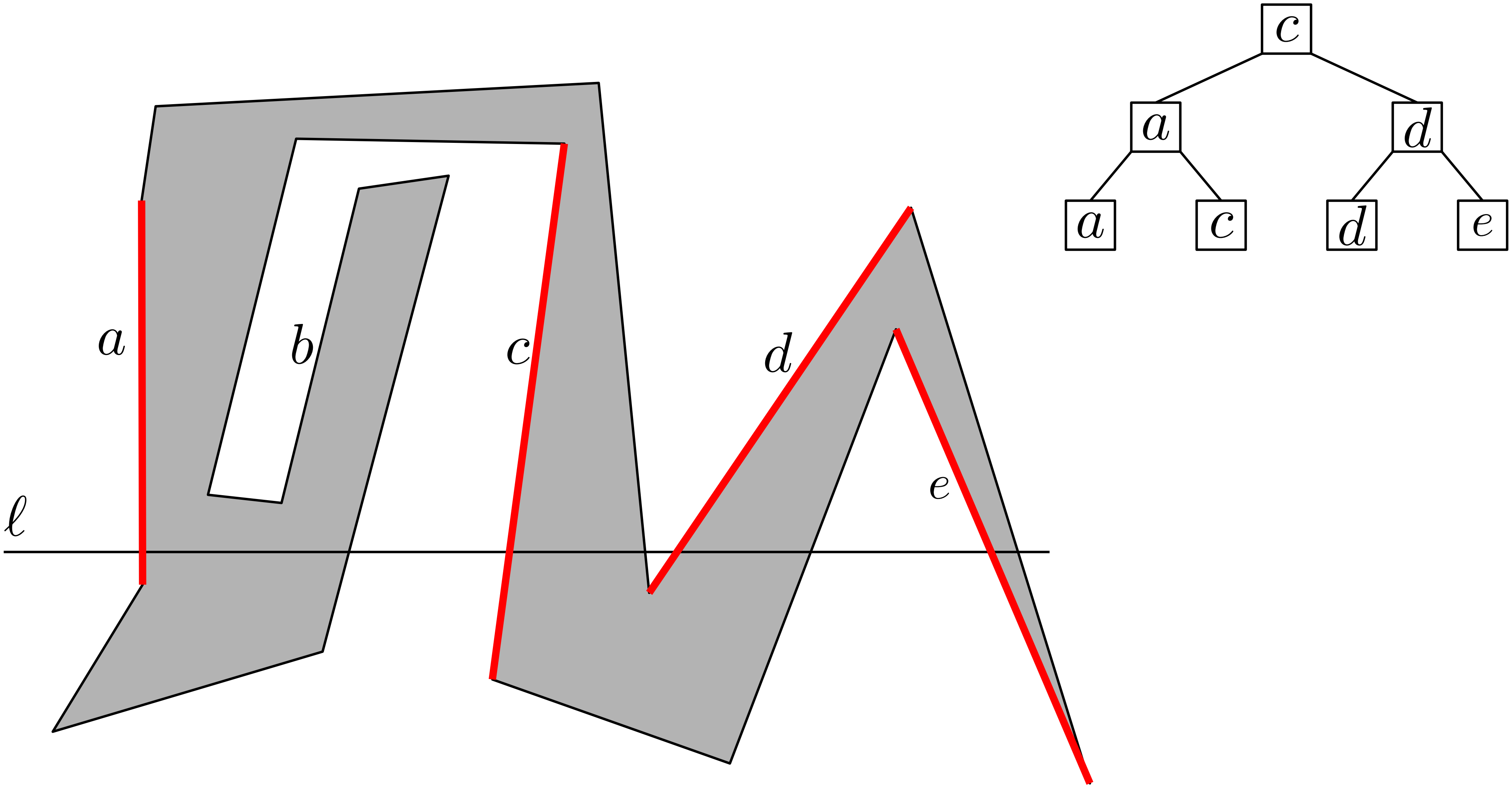
Implement status with balanced binary search tree T.
Sorting order: intersection points with ℓ from left to right.

Status



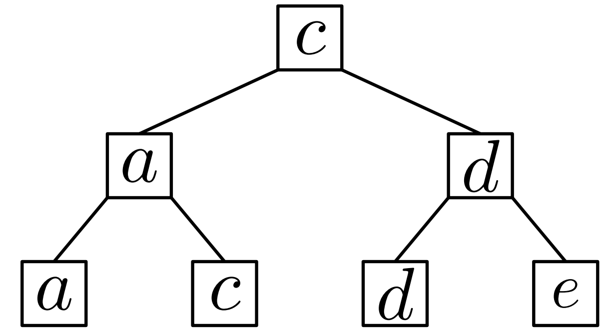
Implement status with balanced binary search tree T.
Sorting order: intersection points with ℓ from left to right.

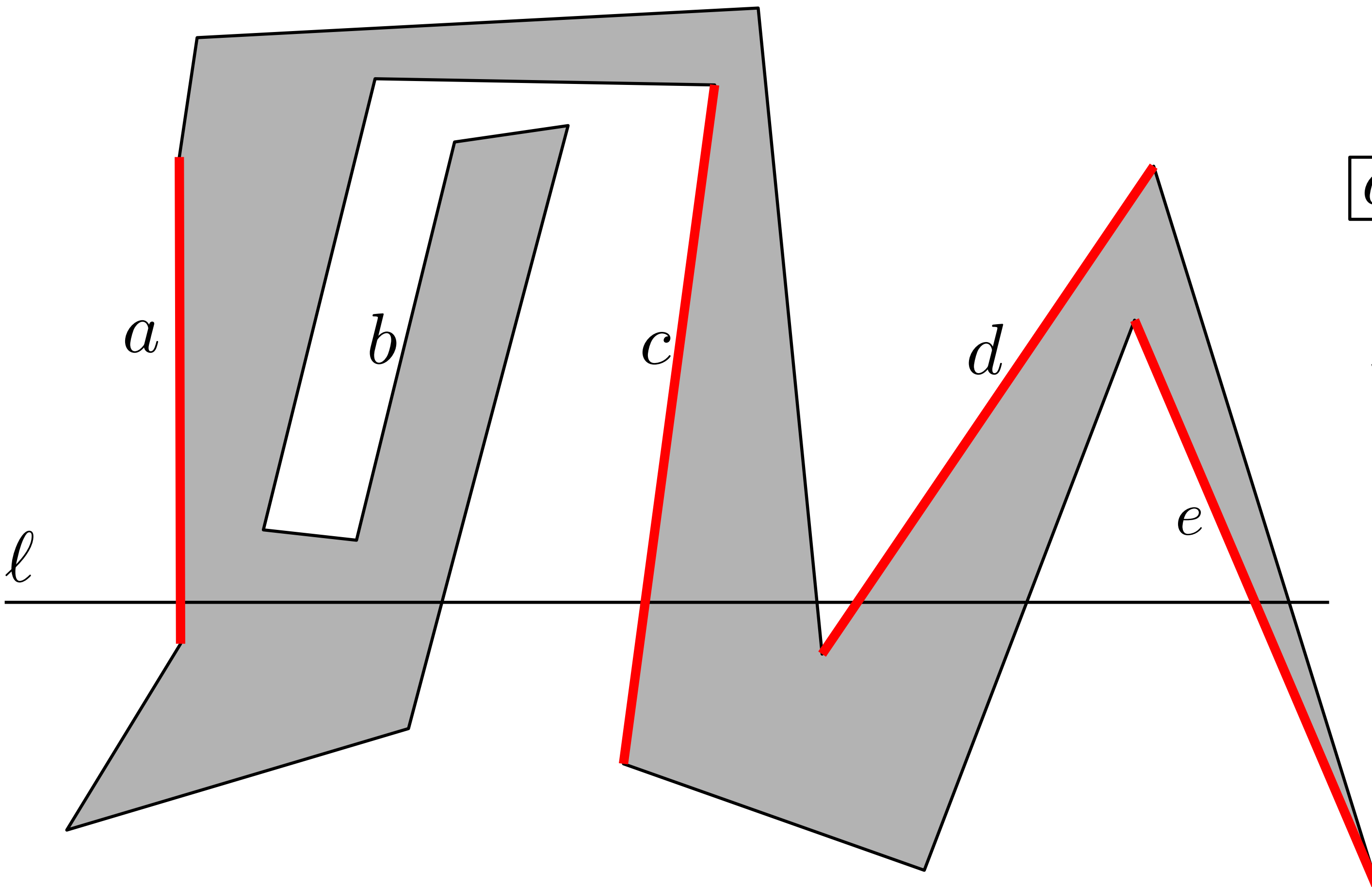
Status



Implement status with balanced binary search tree T.
Sorting order: intersection points with ℓ from left to right.

Status



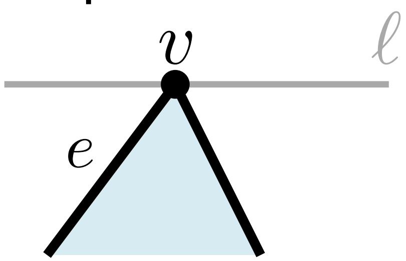


When reaching a new vertex, use $O(\log n)$ time to find the edge in T immediately to the left.

Implement status with balanced binary search tree T.
Sorting order: intersection points with ℓ from left to right.

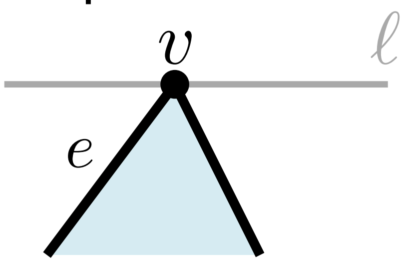
Events

Start vertex: Insert e in T with helper v.

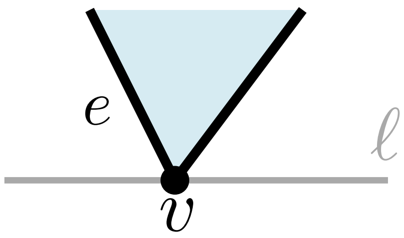


Events

Start vertex: Insert e in T with helper v.

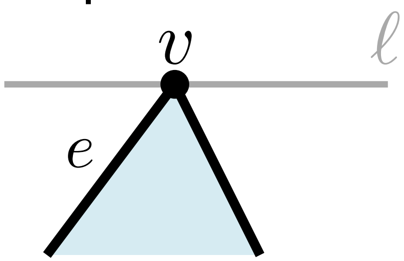


End vertex: Remove e from T.

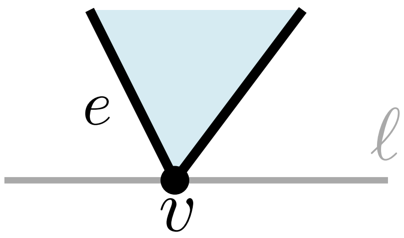


Events

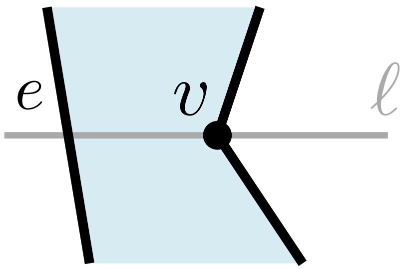
Start vertex: Insert e in T with helper v.



End vertex: Remove e from T.

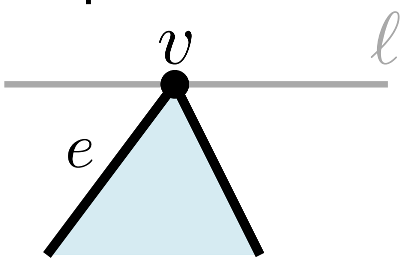


Regular vertex with P to the left: Update helper of e to v.

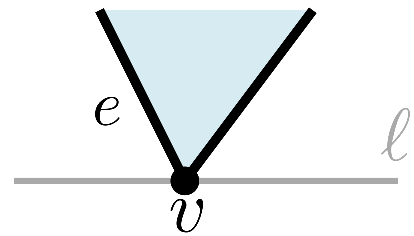


Events

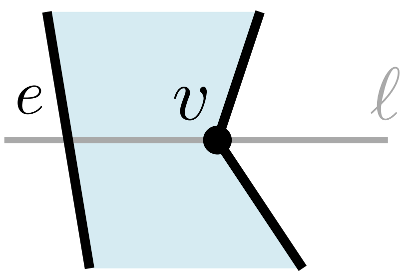
Start vertex: Insert e in T with helper v.



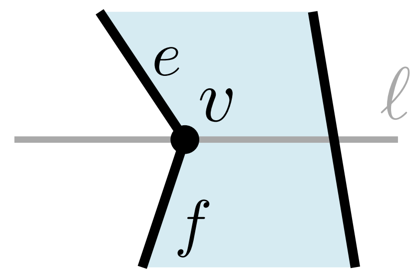
End vertex: Remove e from T.



Regular vertex with P to the left: Update helper of e to v.

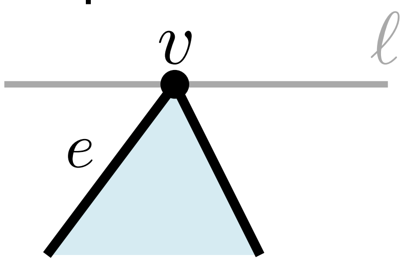


Regular vertex with P to the right: Replace e by f in T with helper v.

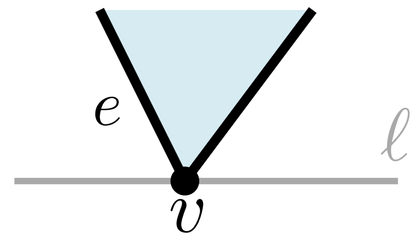


Events

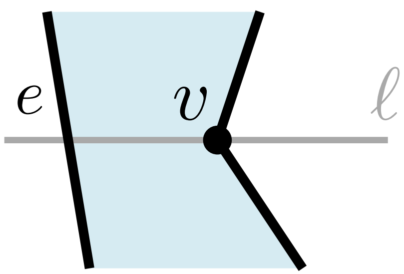
Start vertex: Insert e in T with helper v.



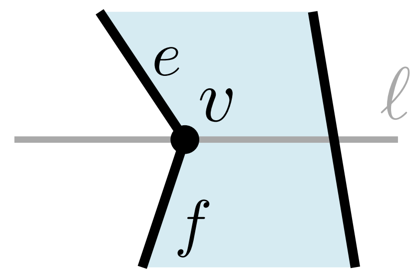
End vertex: Remove e from T.



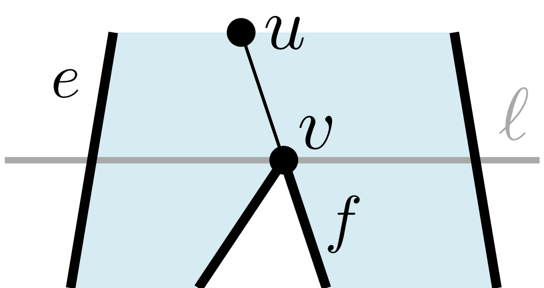
Regular vertex with P to the left: Update helper of e to v.



Regular vertex with P to the right: Replace e by f in T with helper v.

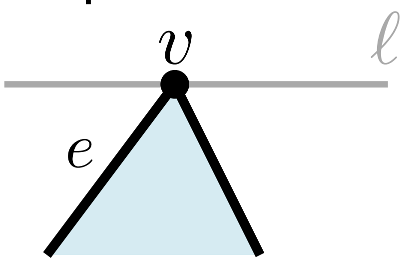


Split vertex: Add diagonal to helper u of e. Update helper of e to v. Add f to T with helper v.

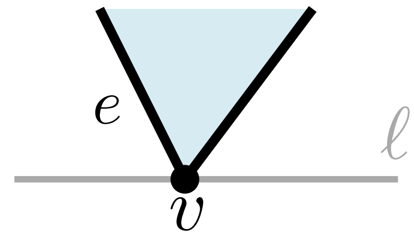


Events

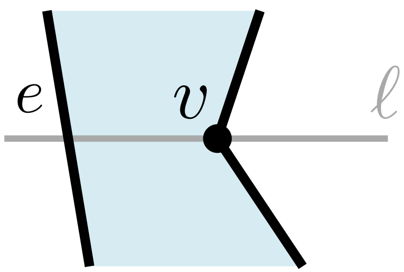
Start vertex: Insert e in T with helper v.



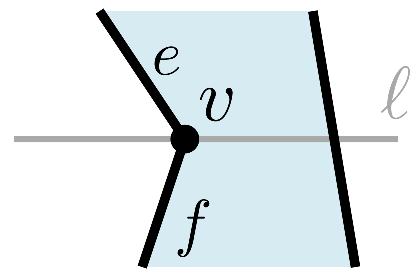
End vertex: Remove e from T.



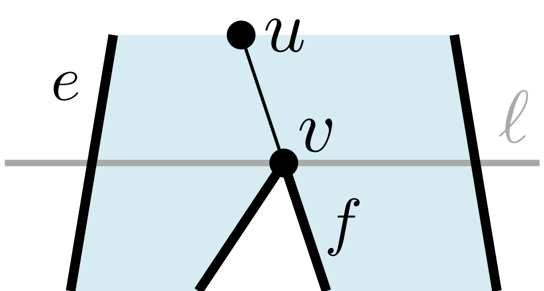
Regular vertex with P to the left: Update helper of e to v.



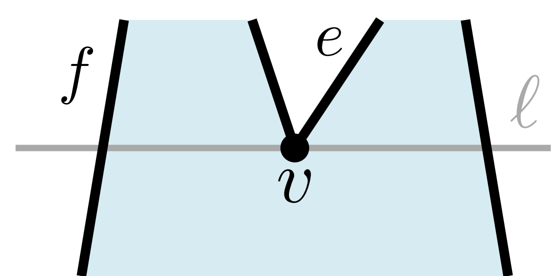
Regular vertex with P to the right: Replace e by f in T with helper v.



Split vertex: Add diagonal to helper u of e. Update helper of e to v. Add f to T with helper v.

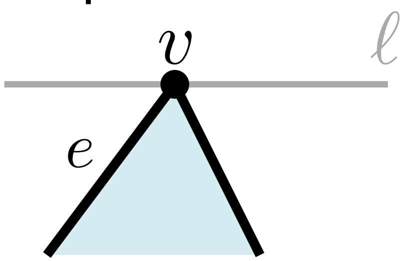


Merge vertex: Remove e from T. Update helper of f to v.

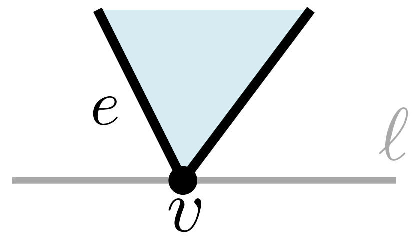


Events

Start vertex: Insert e in T with helper v.

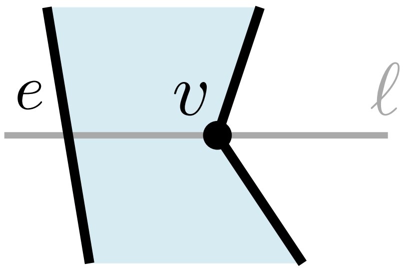


End vertex: Remove e from T.

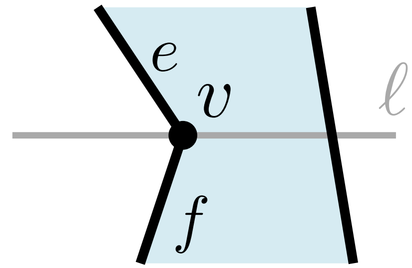


Each event takes time $O(\log n)$.

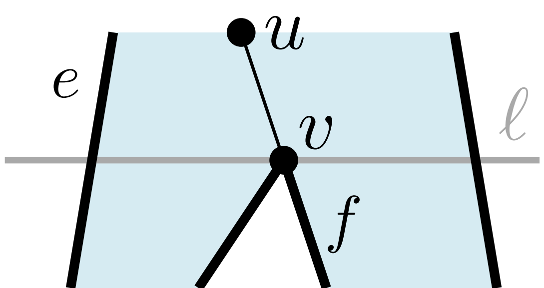
Regular vertex with P to the left: Update helper of e to v.



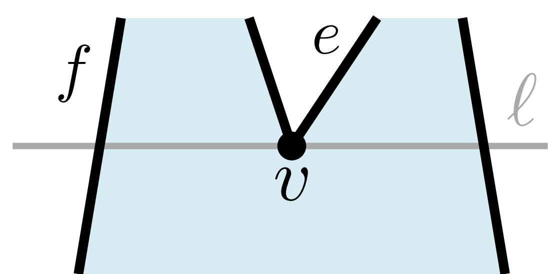
Regular vertex with P to the right: Replace e by f in T with helper v.



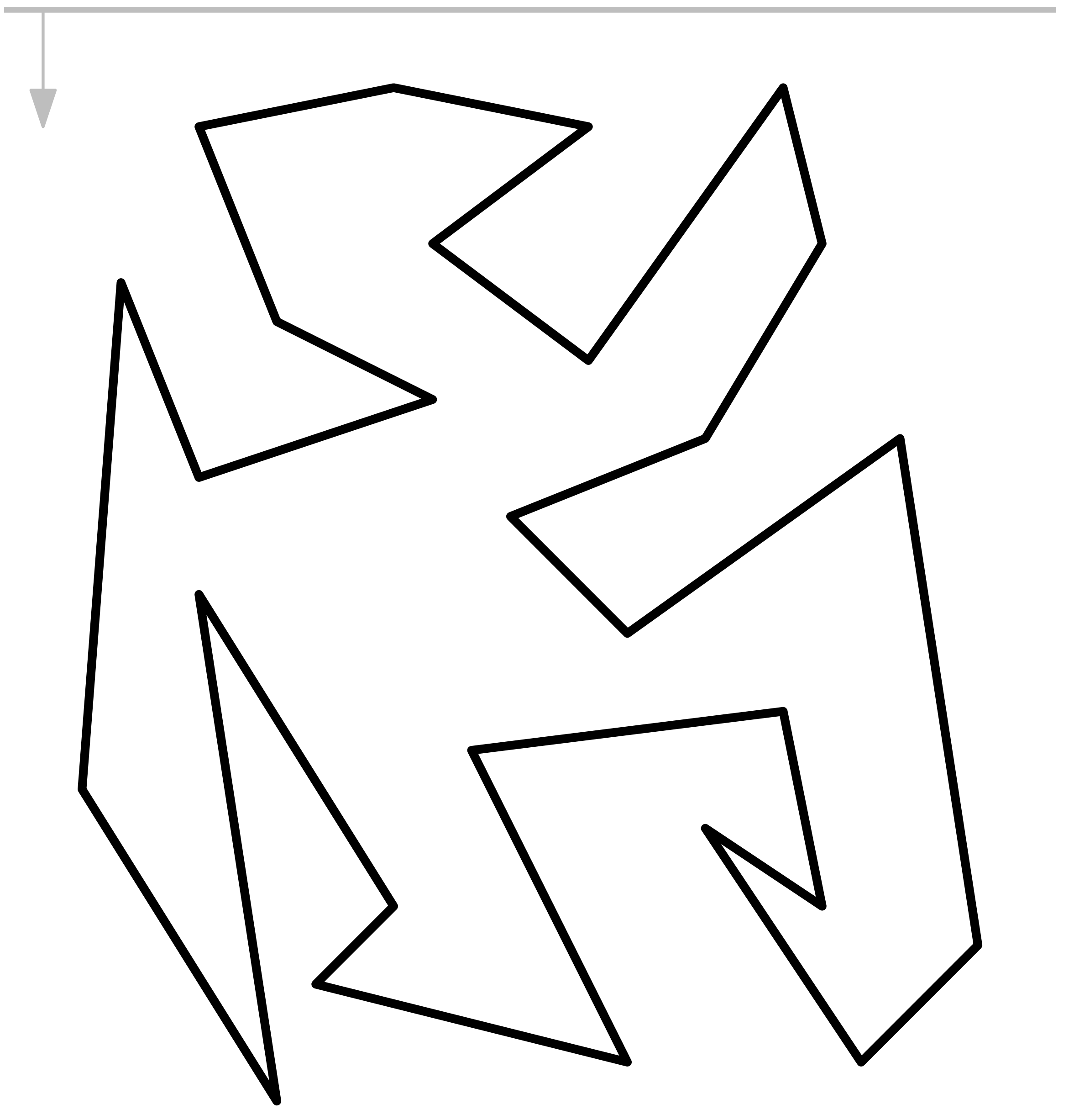
Split vertex: Add diagonal to helper u of e. Update helper of e to v. Add f to T with helper v.



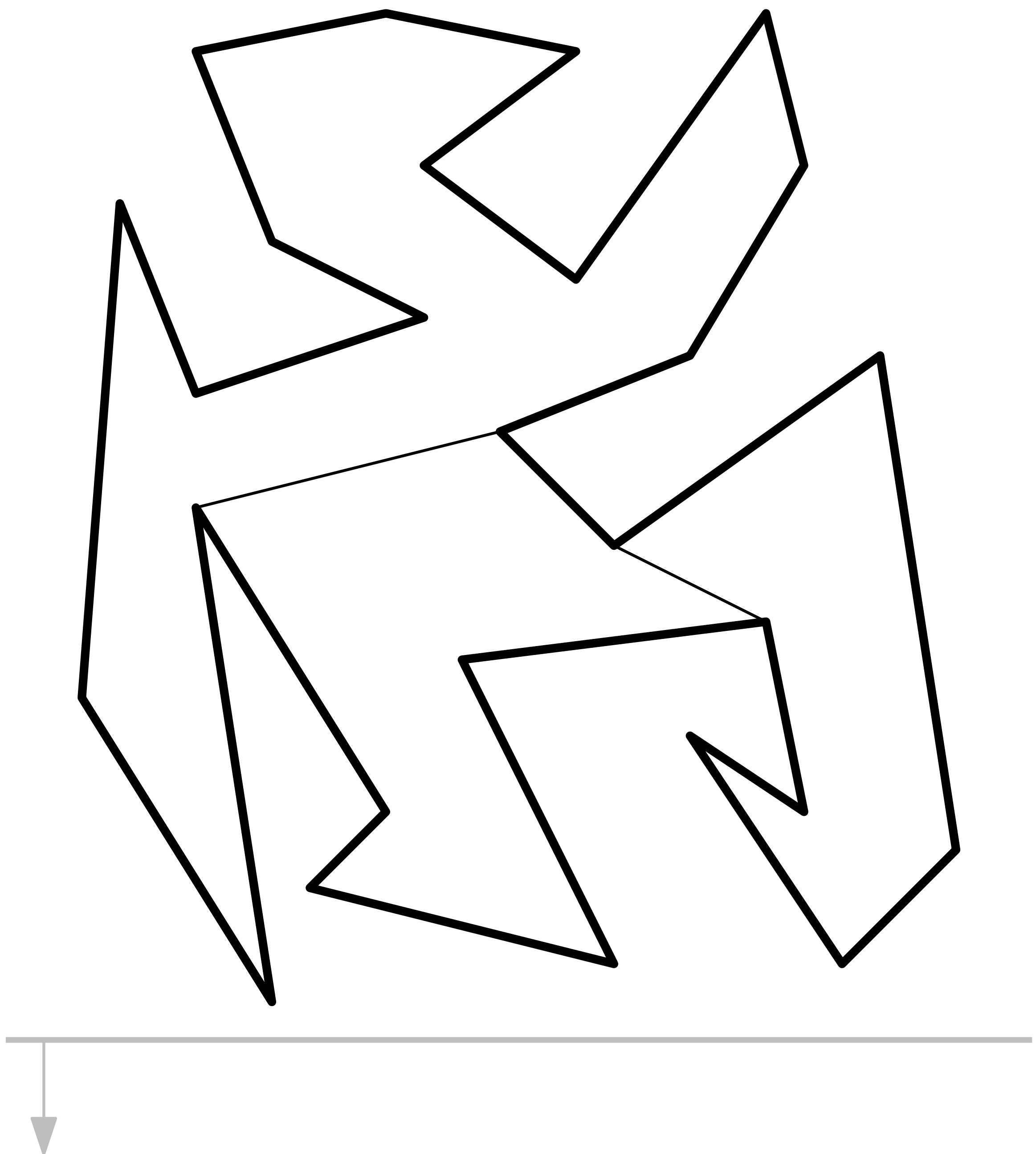
Merge vertex: Remove e from T. Update helper of f to v.



Sweep down, then up

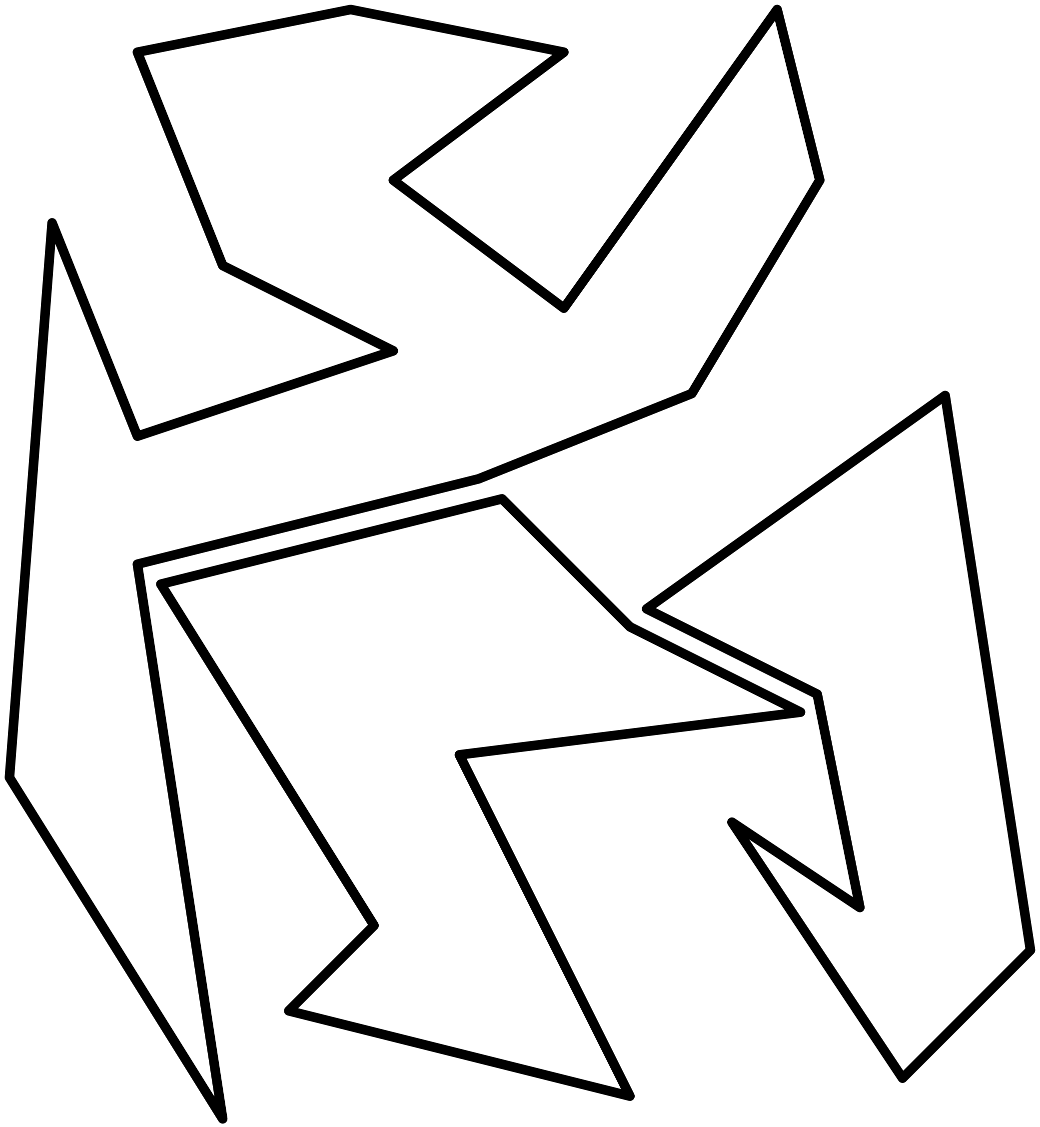


Sweep down, then up



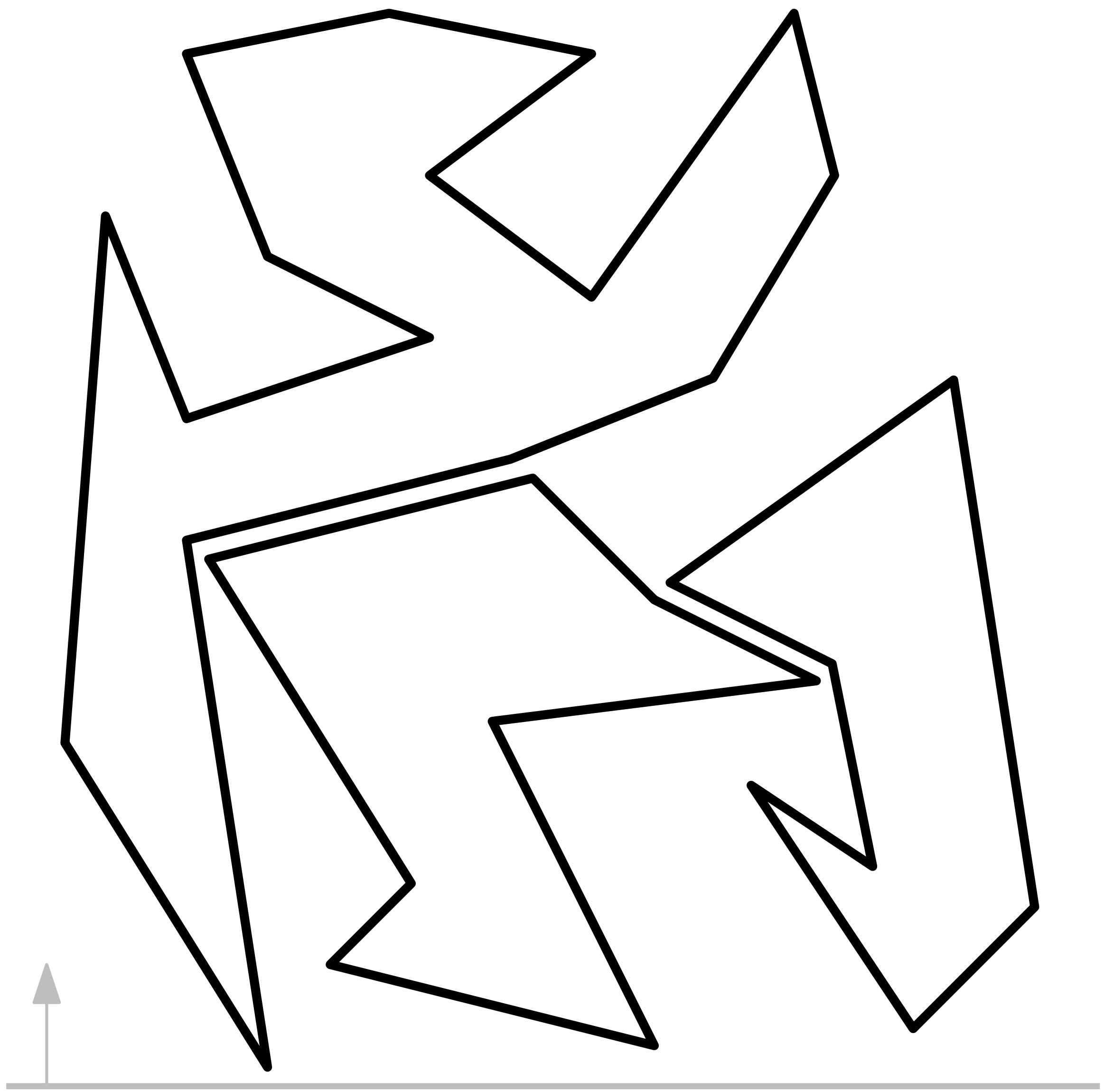
Sweep down, then up

From below:
Each polygon separately!



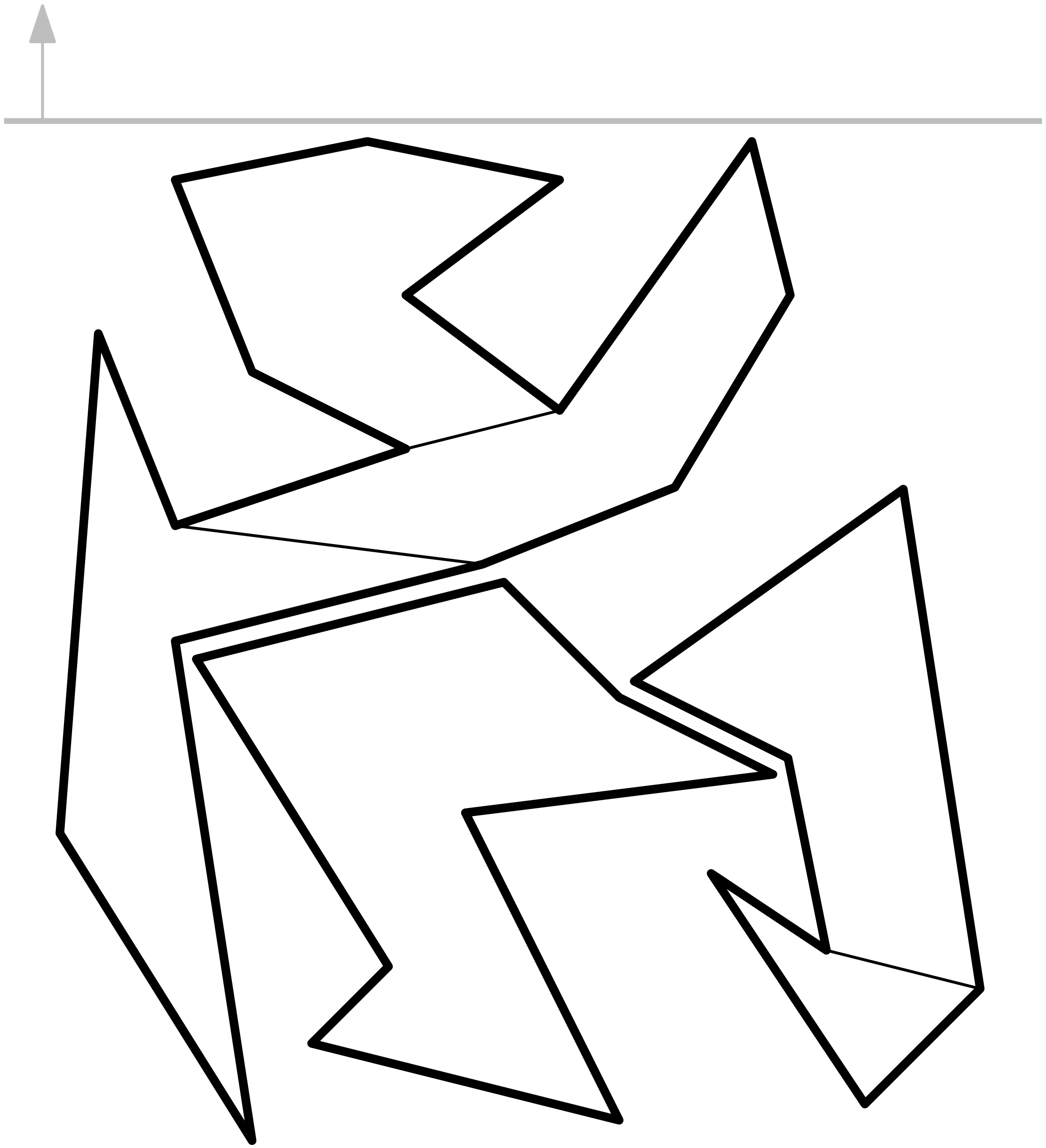
Sweep down, then up

From below:
Each polygon
separately!

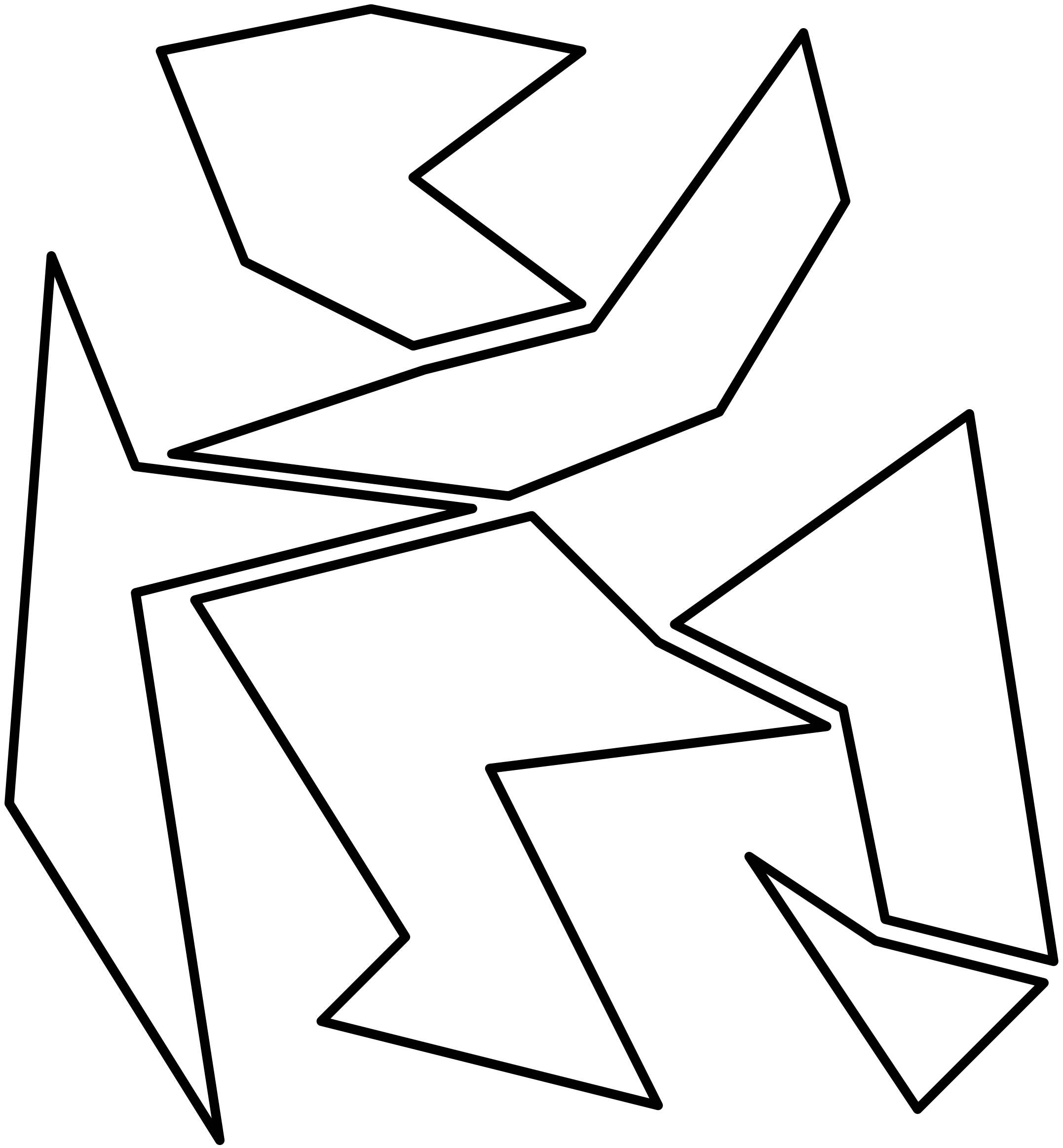


Sweep down, then up

From below:
Each polygon
separately!

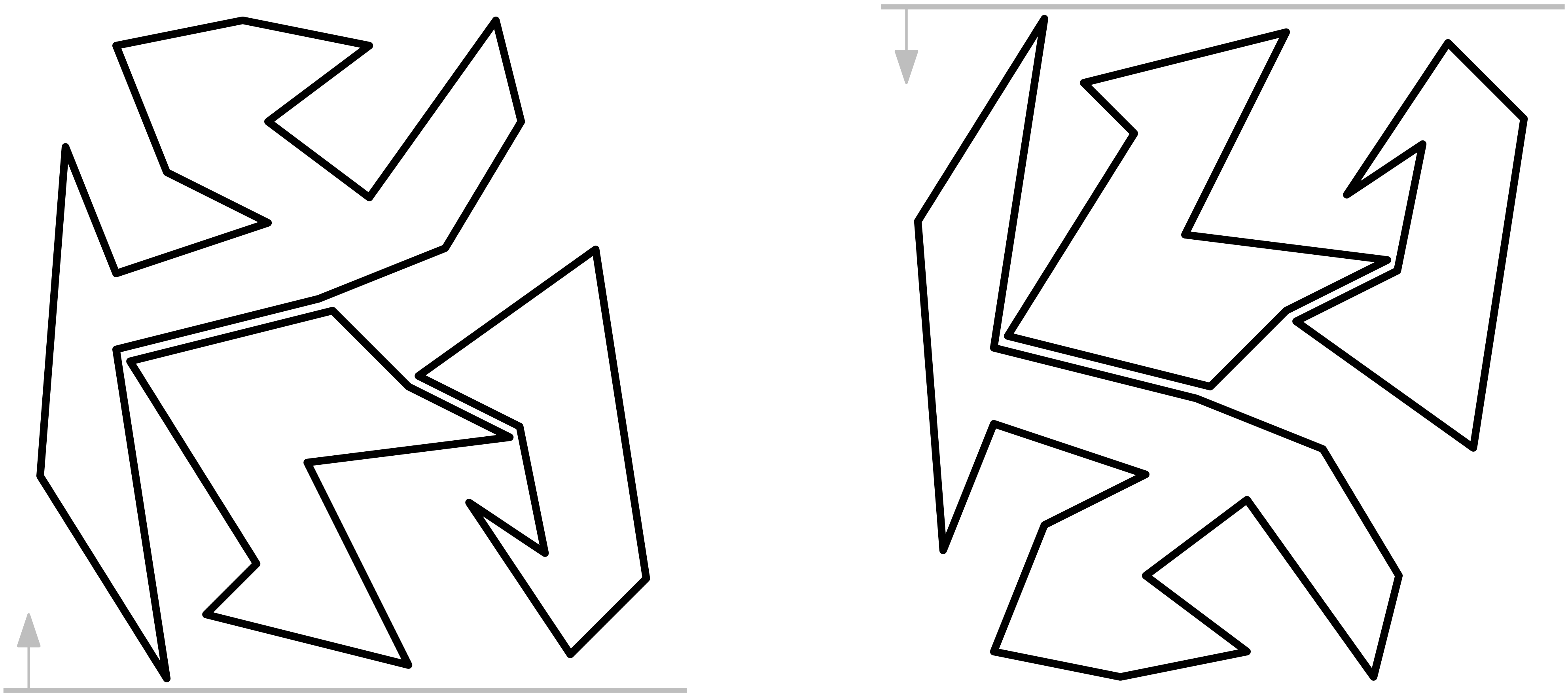


Sweep down, then up



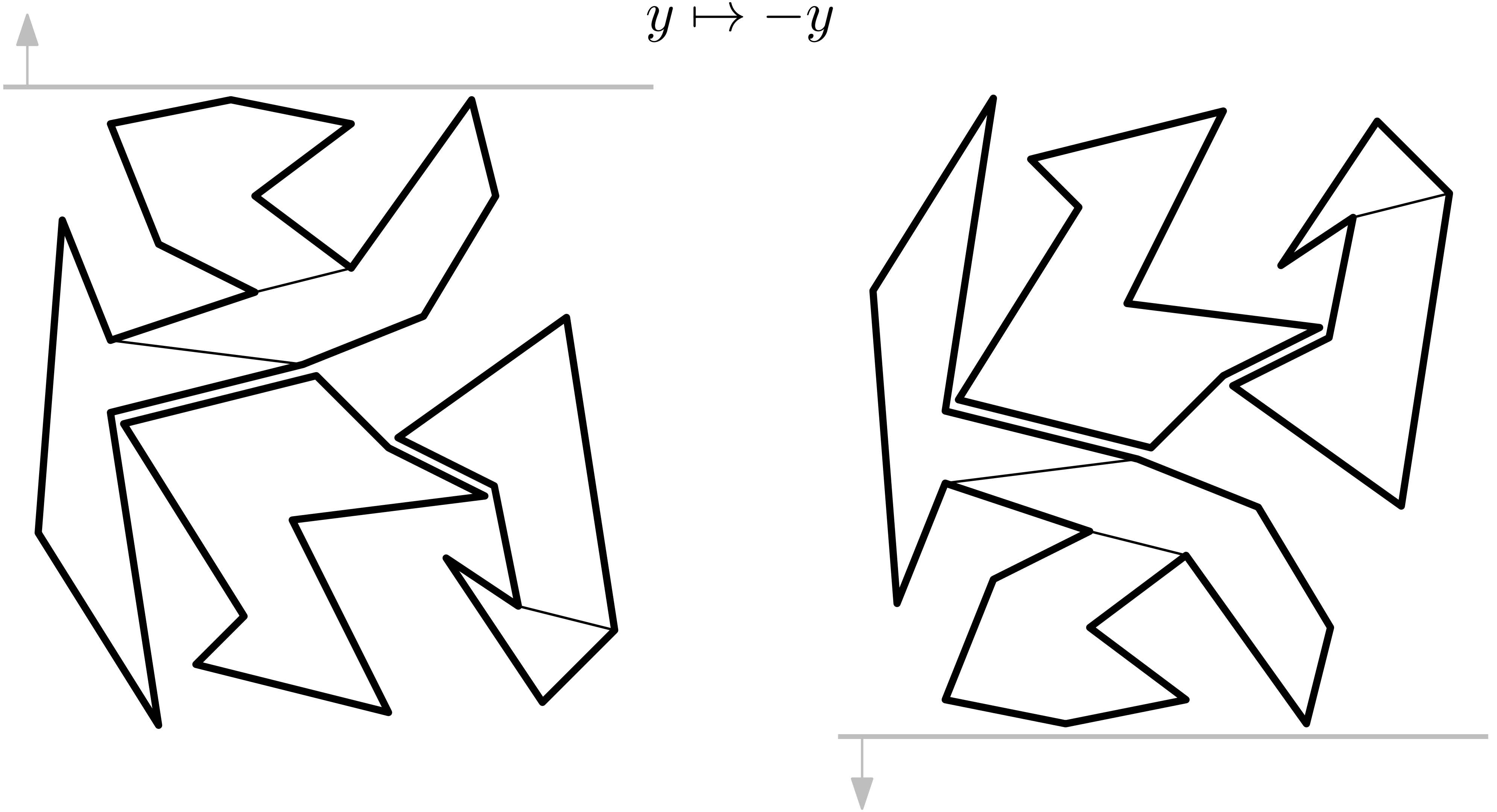
Sweep down, then up

$$y \mapsto -y$$

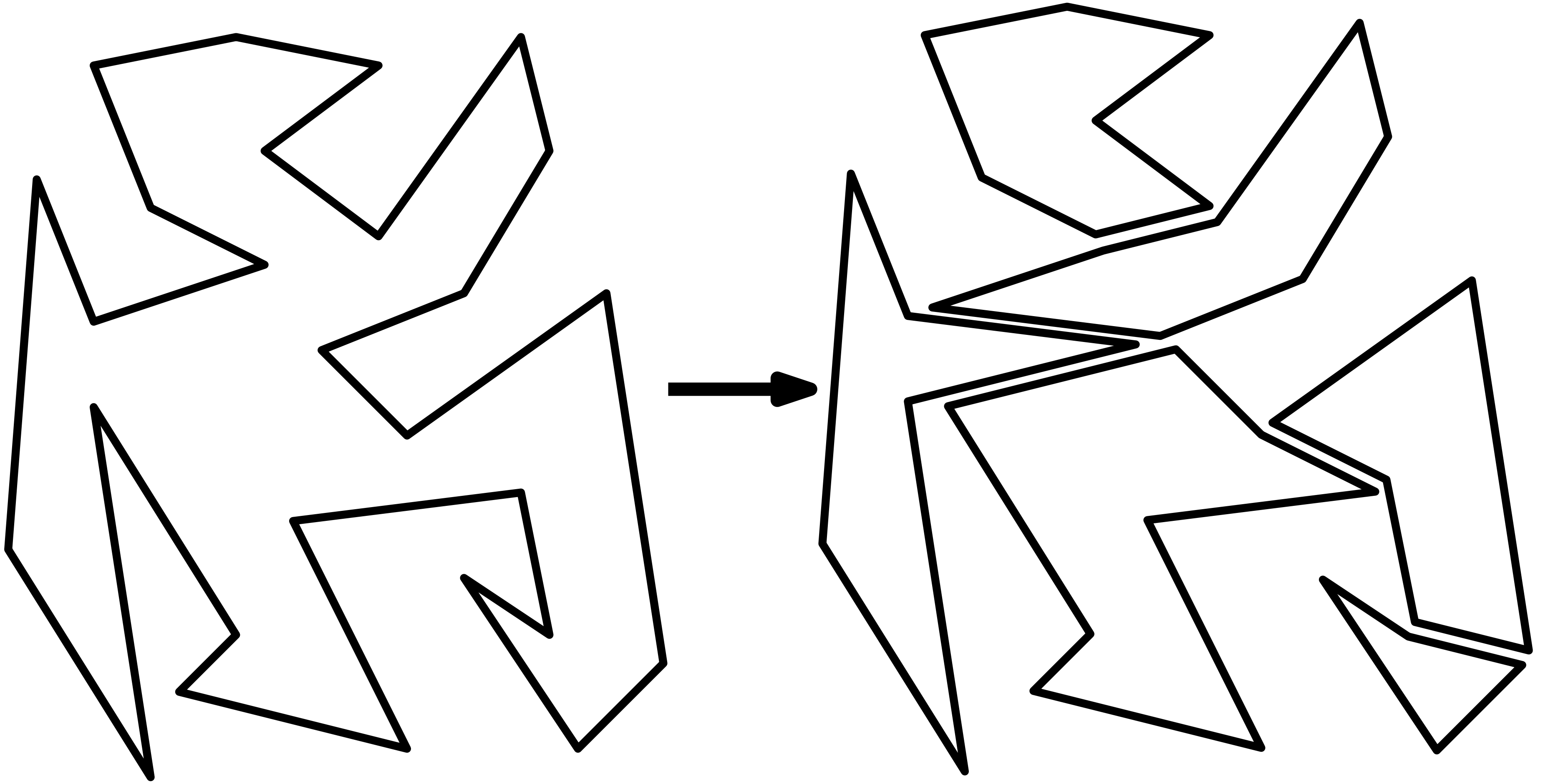


Sweep down, then up

$$y \mapsto -y$$



How many vertices?



n vertices

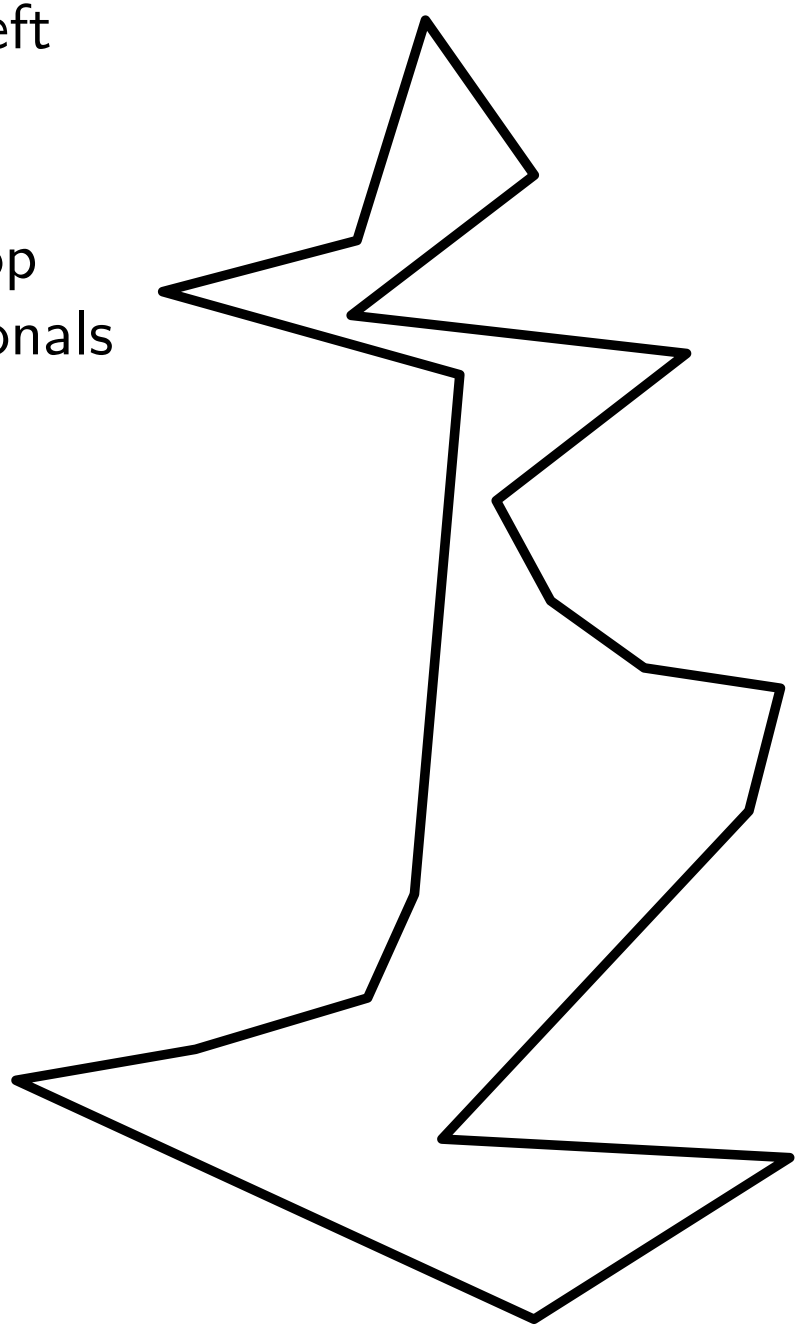
2 extra vertices per diagonal and $\leq n-3$ diagonals $\Rightarrow$ $\leq n+2(n-3)=3n-6$ vertices

Efficiency

Sorting vertices by y-coordinates: $O(n \log n)$.
Handling n events: $O(n \log n)$ time.
Doing it again from below (at most $3n - 6$ vertices now).
In total $O(n \log n)$.

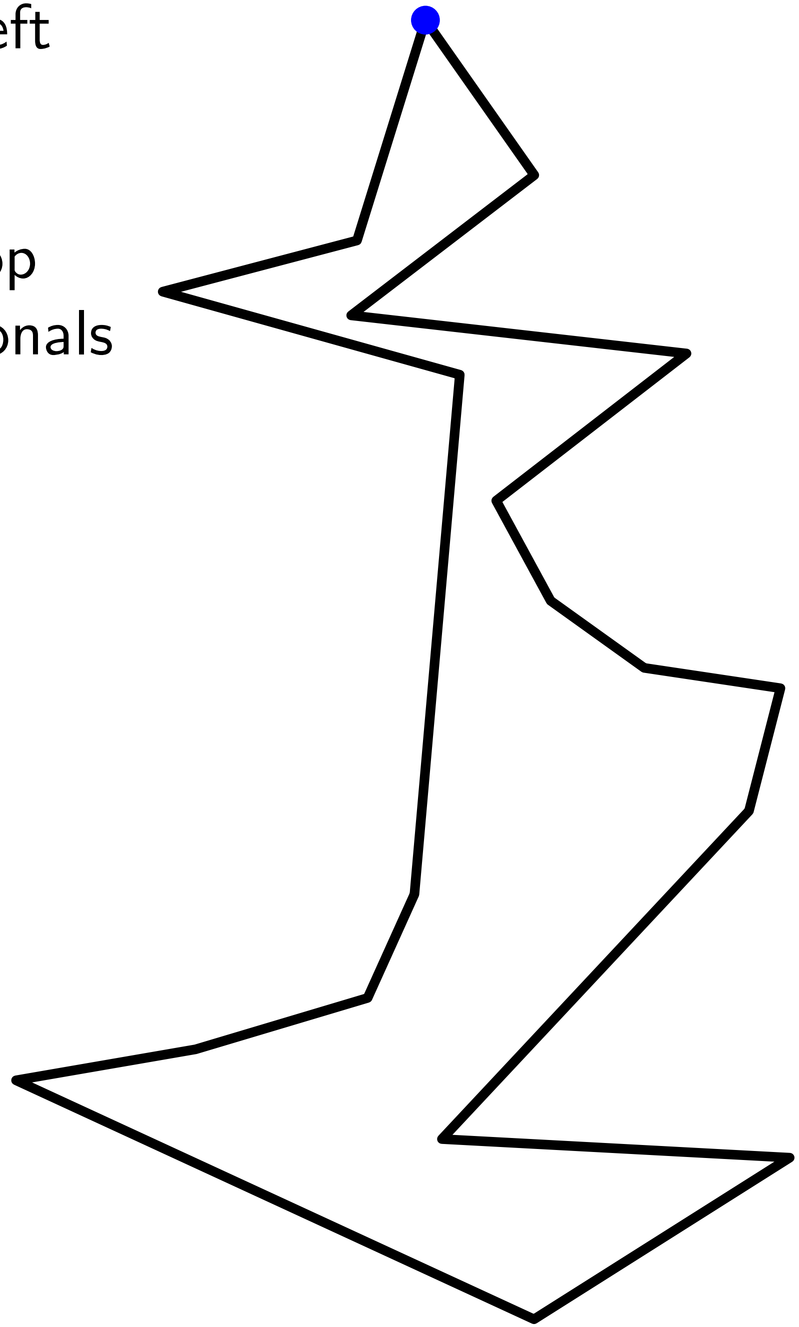
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



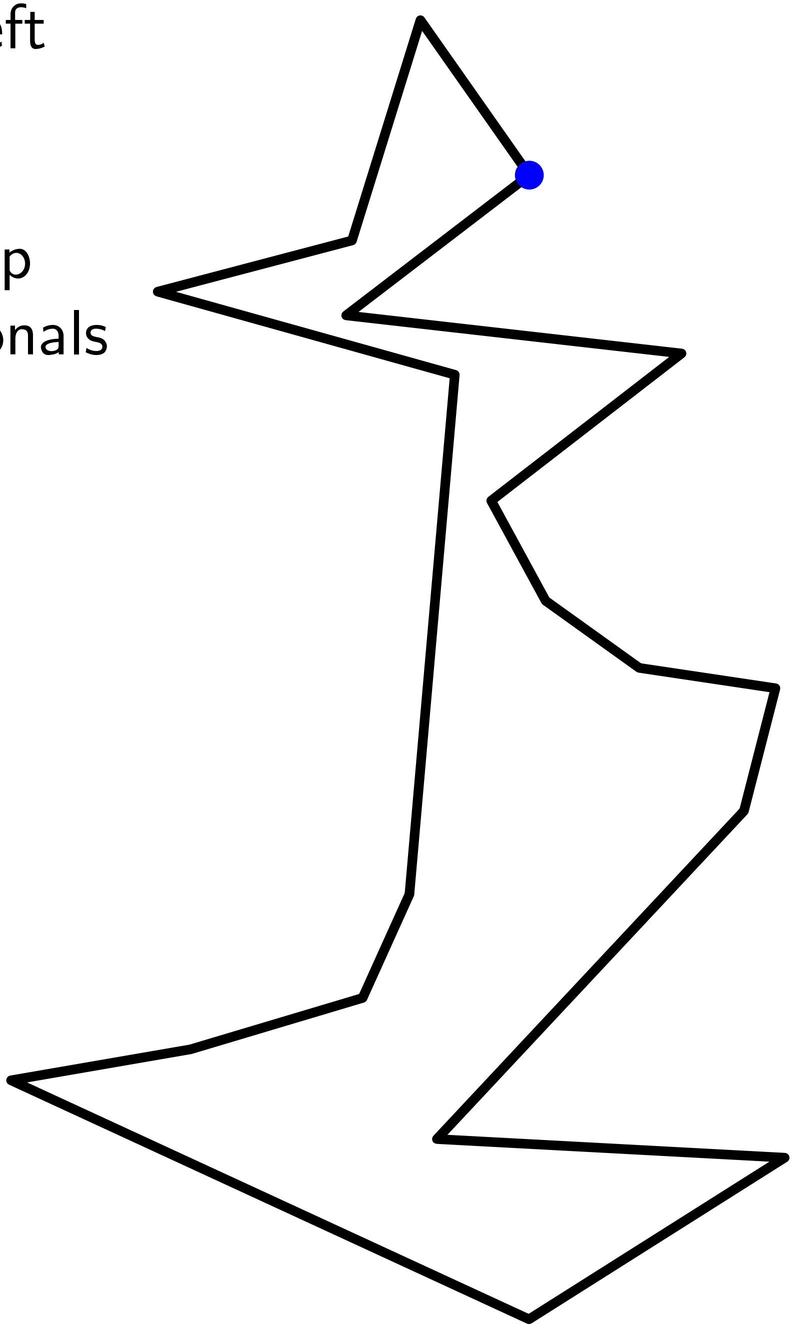
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



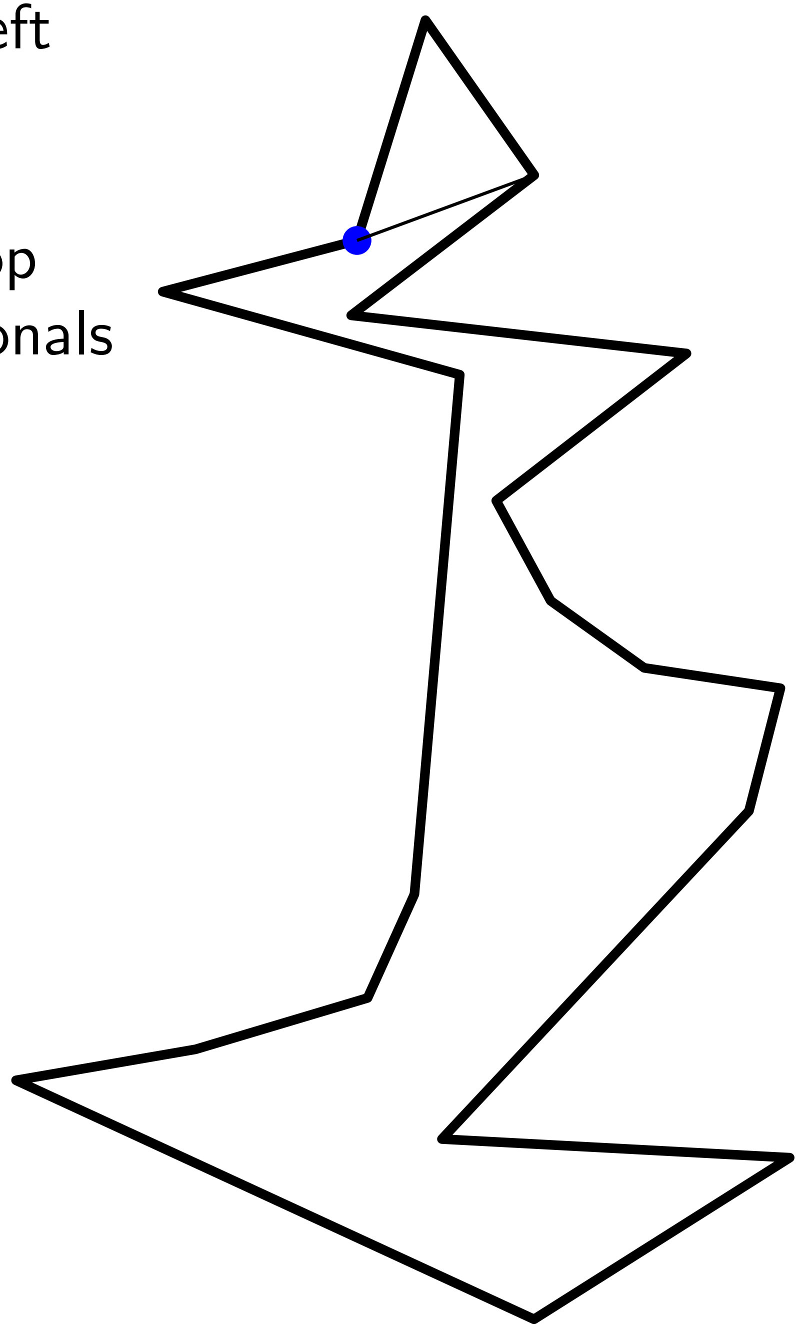
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



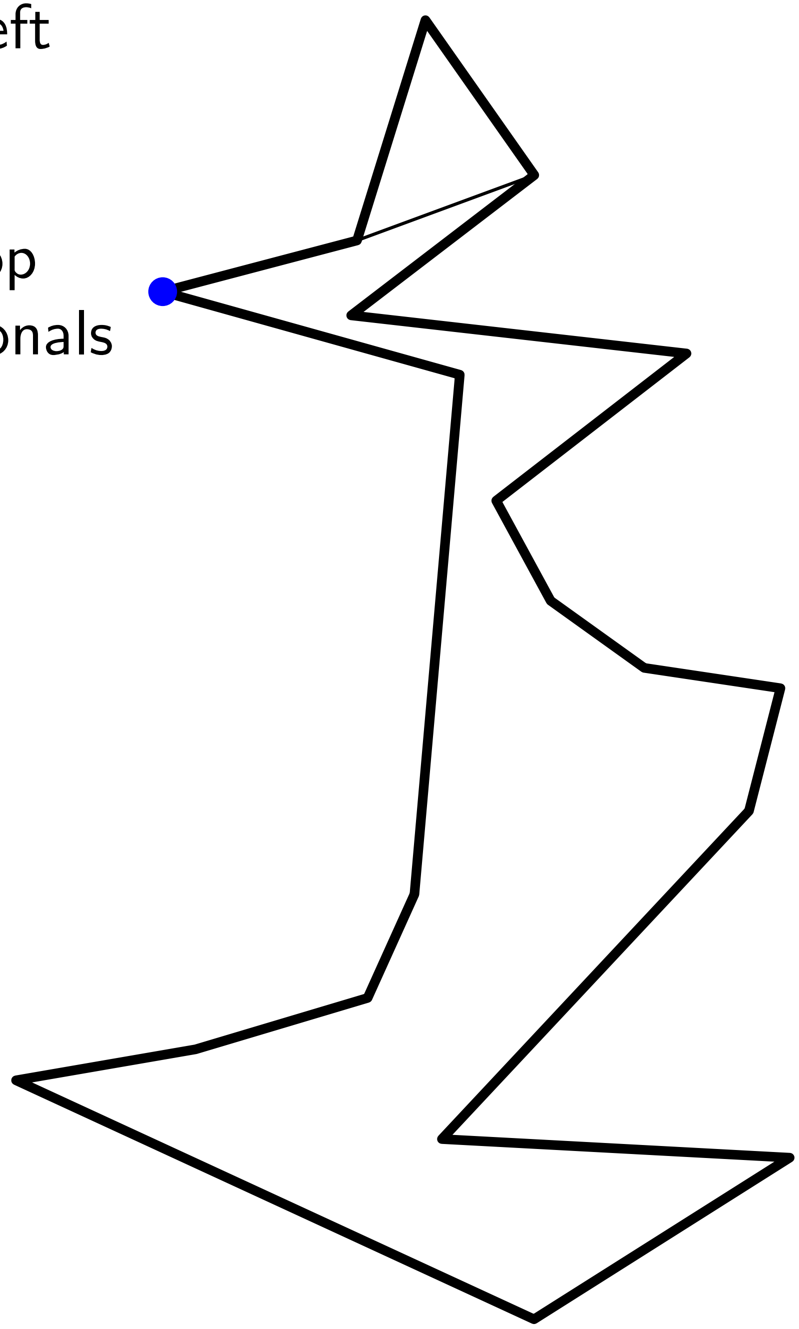
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



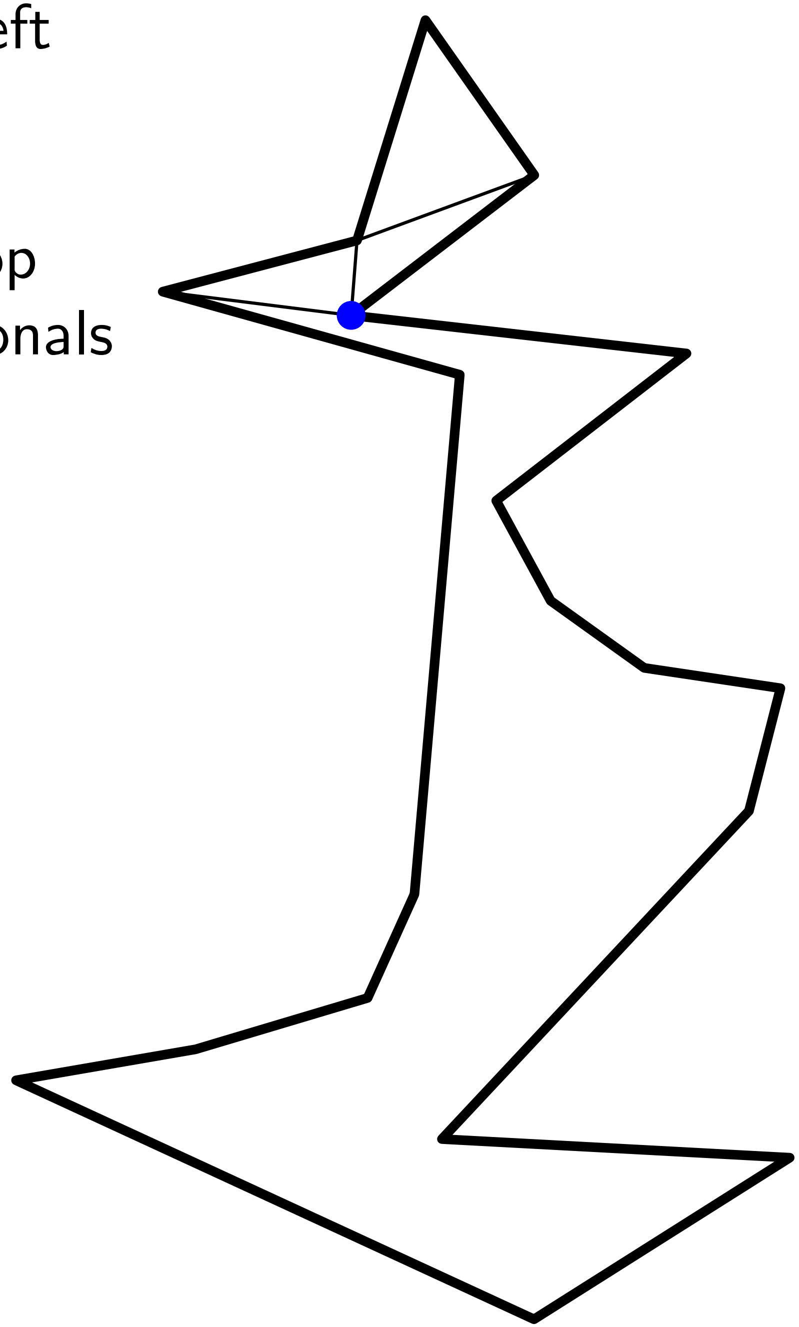
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



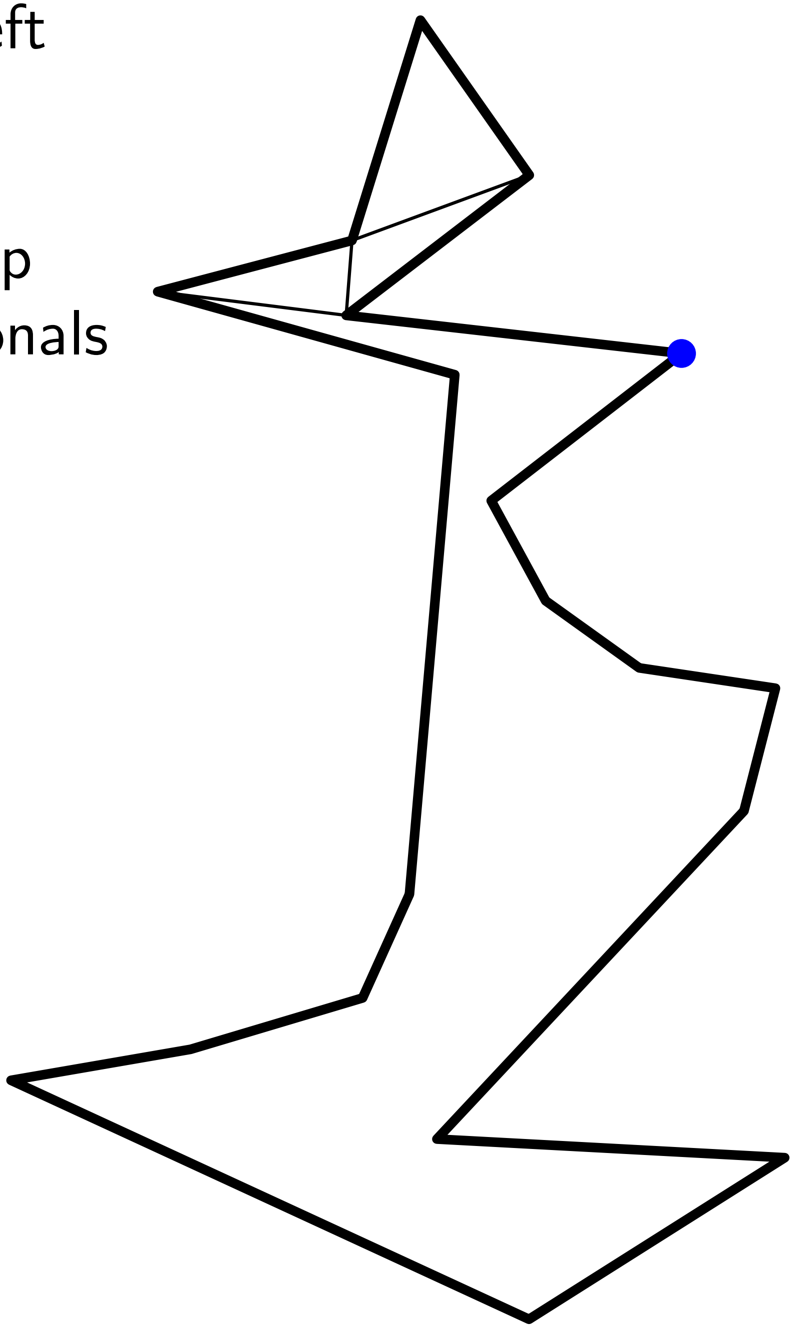
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



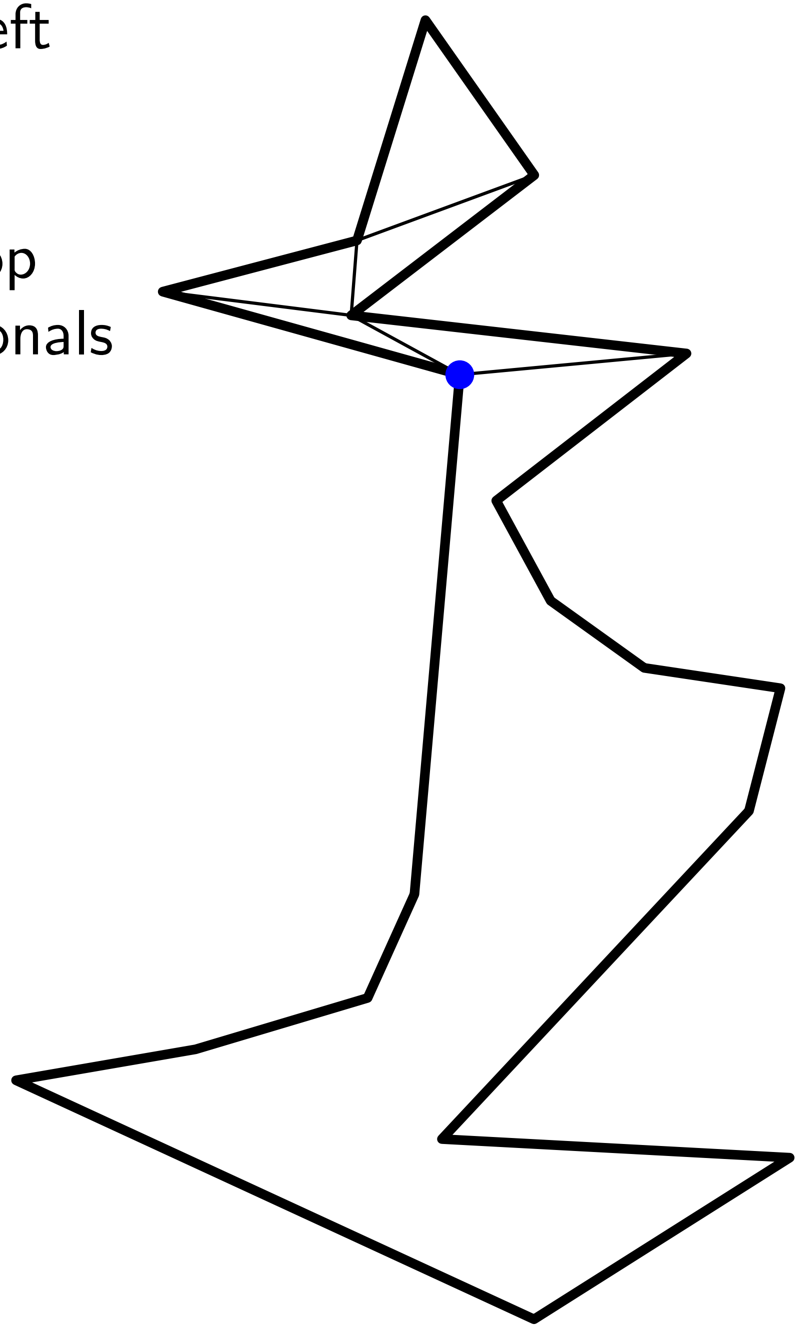
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



Triangulating a y-monotone polygon

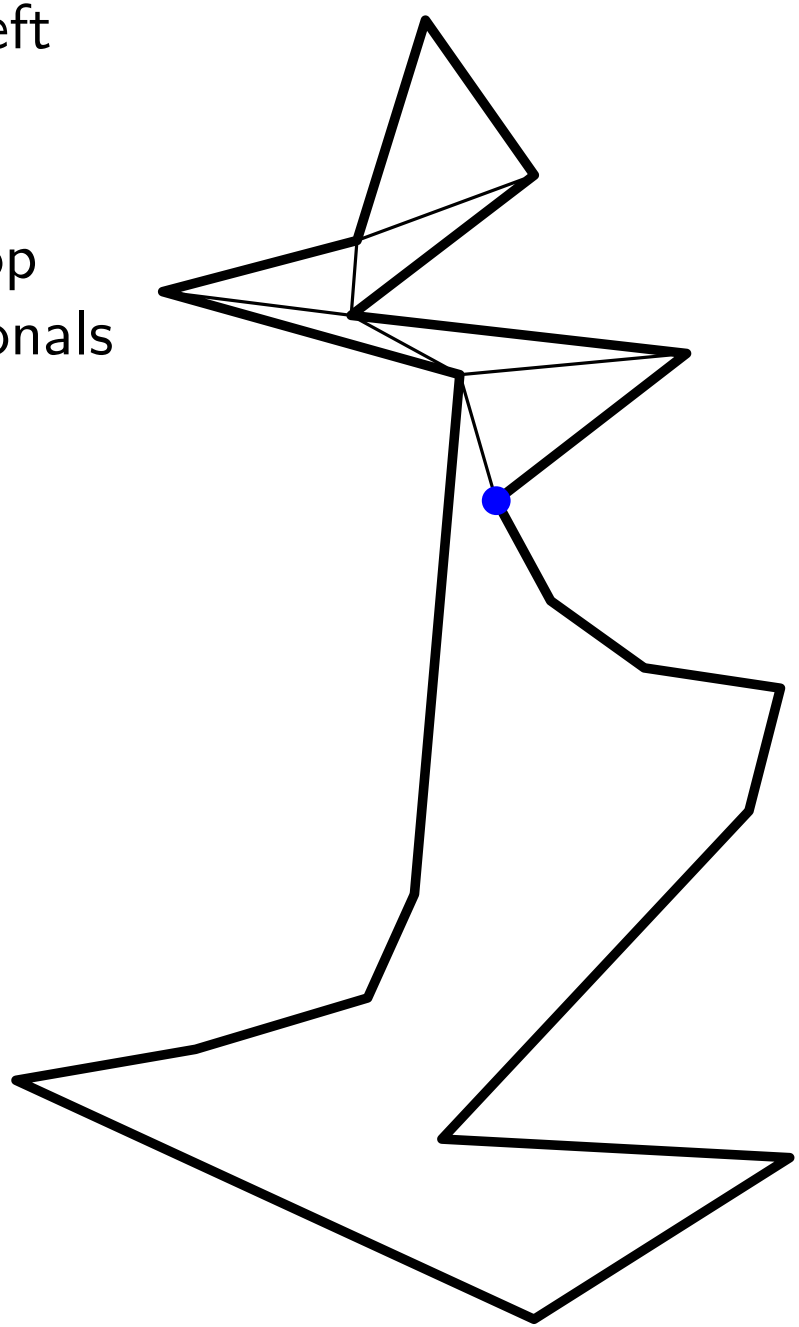
1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



Triangulating a y-monotone polygon

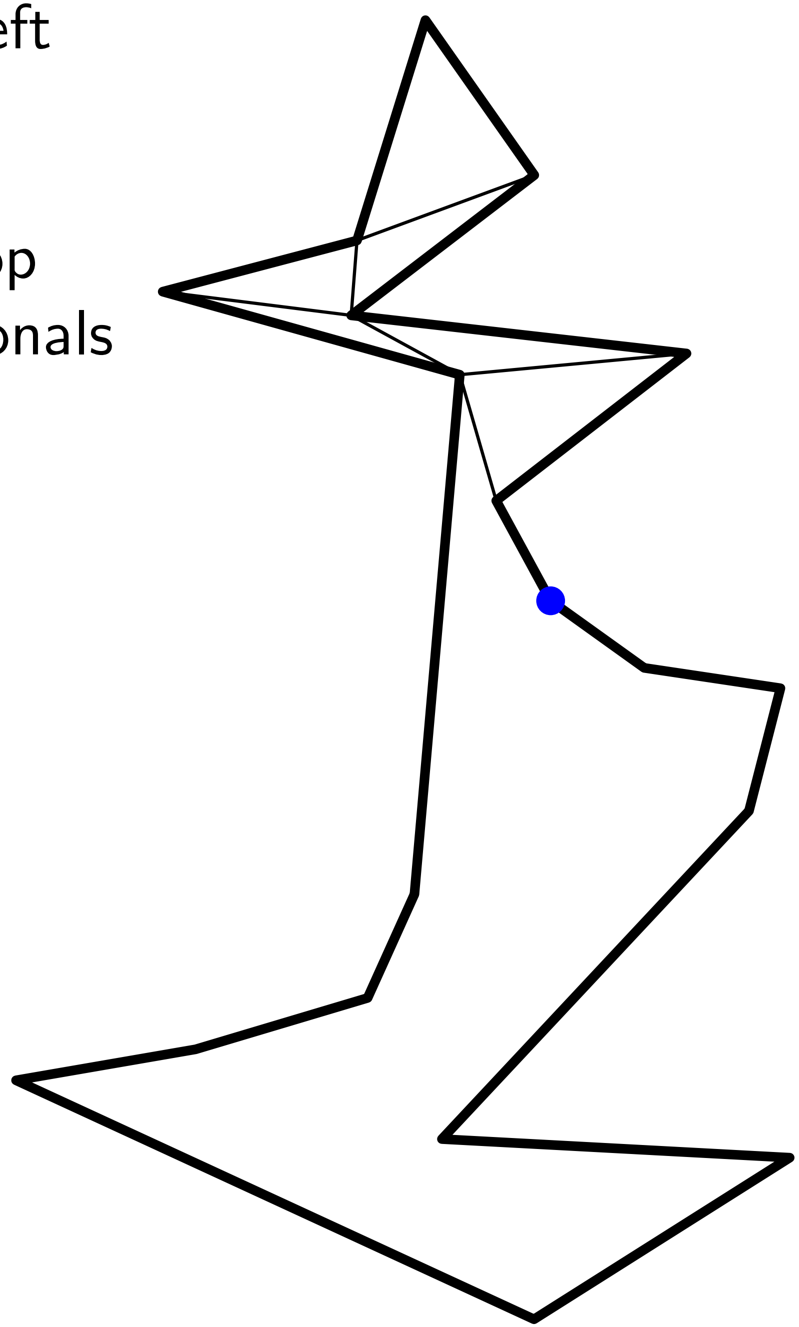
1: Merge vertices of left and right chain to get sorted order

2: Traverse vertices top down and create diagonals to all possible vertices above



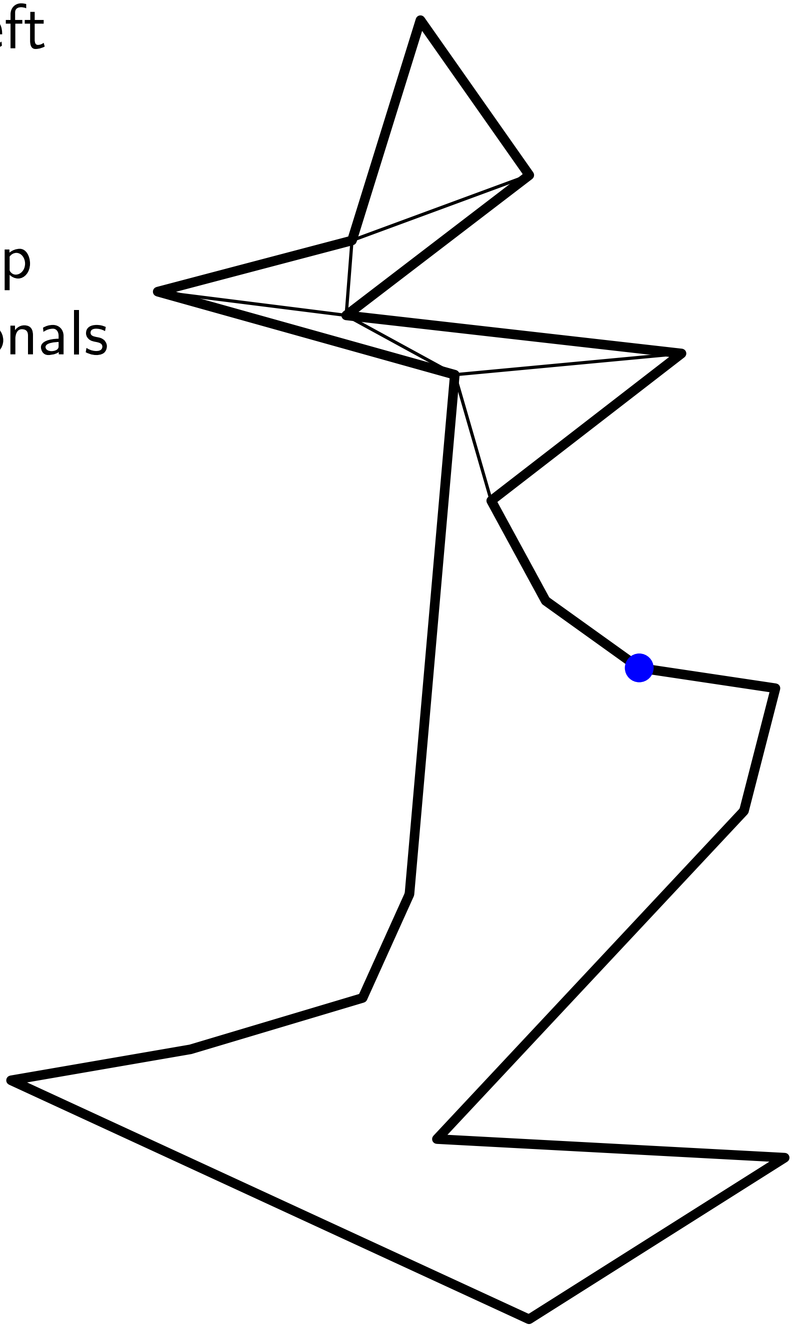
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



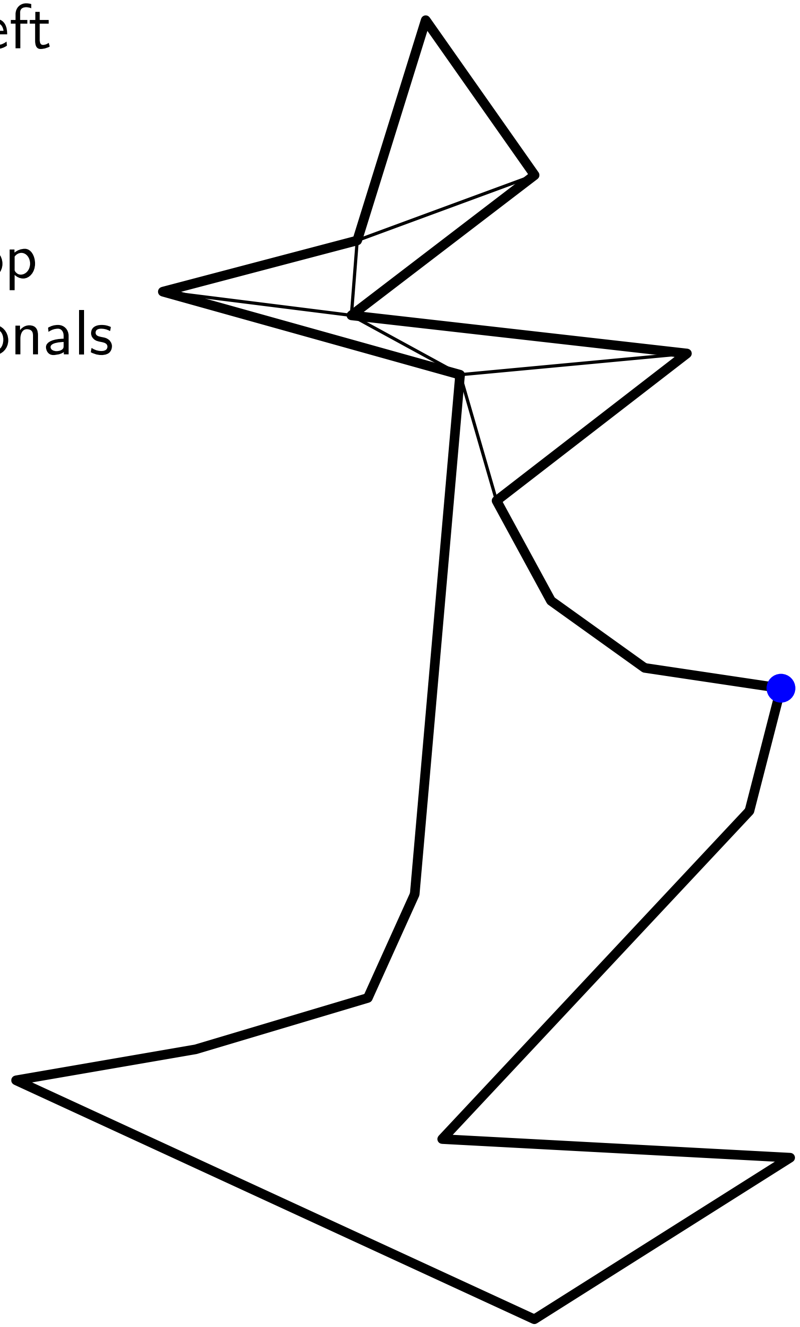
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



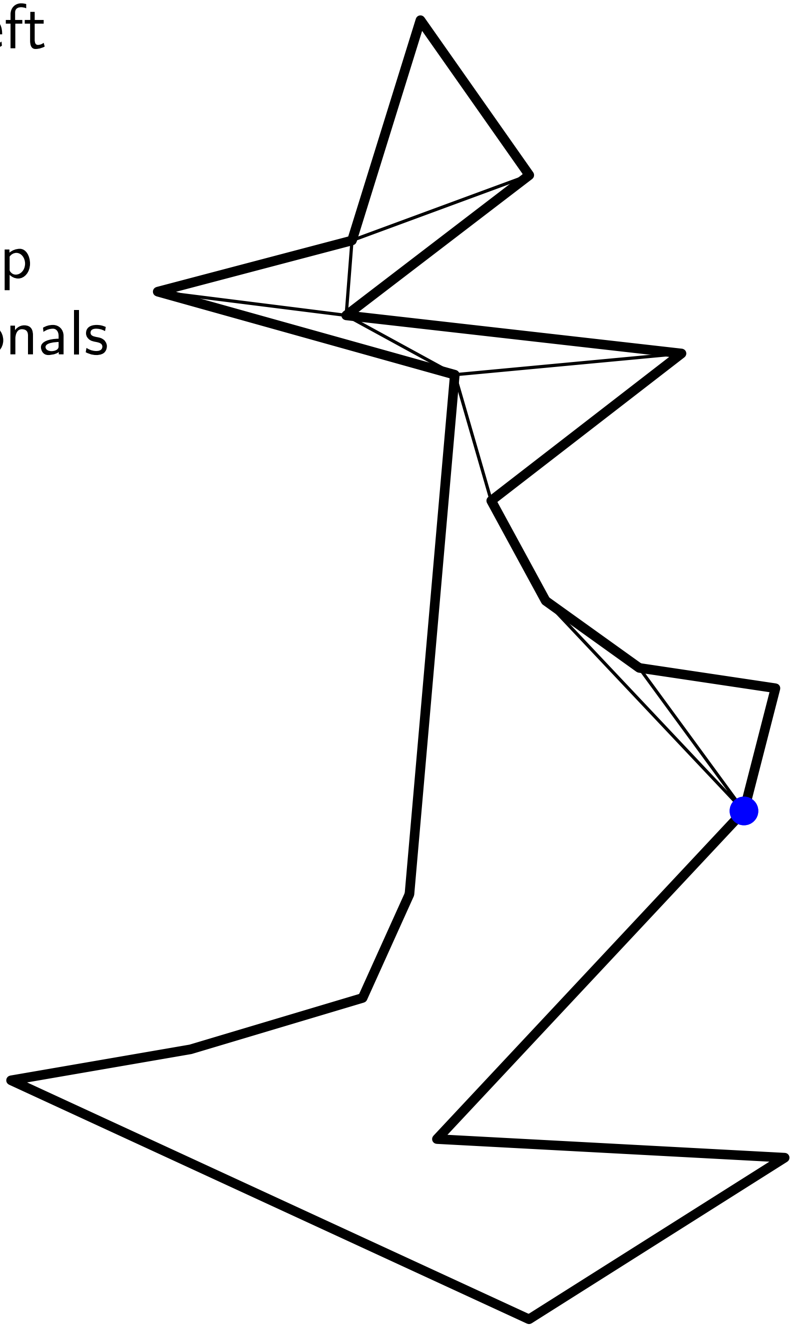
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



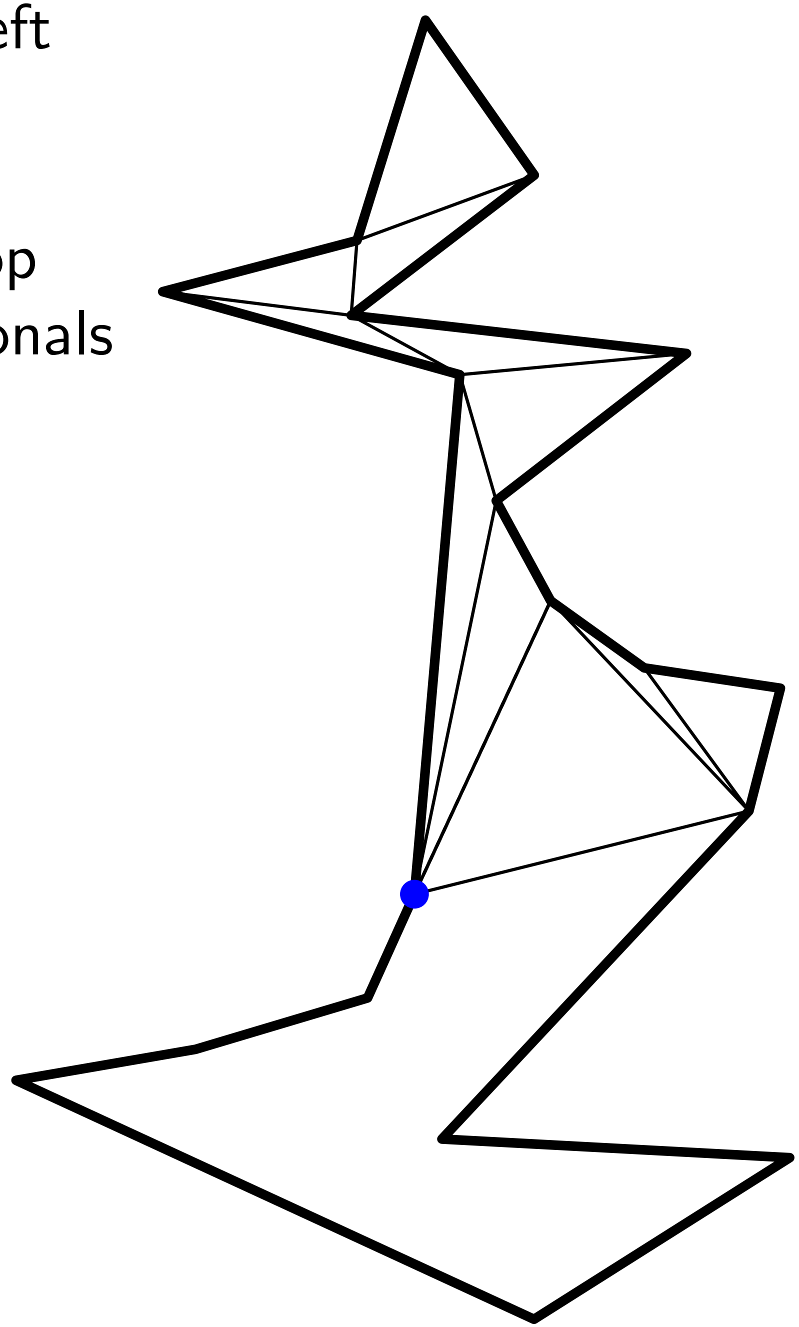
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



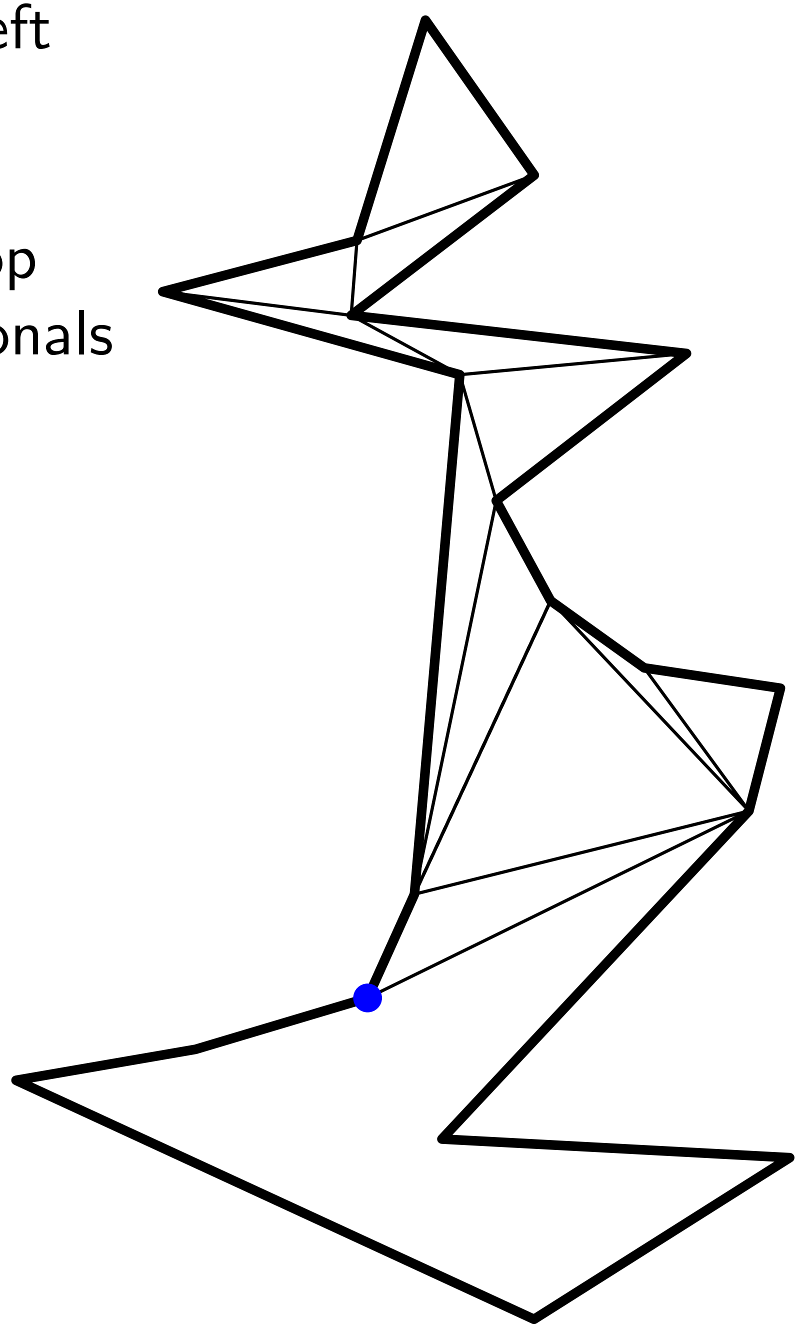
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



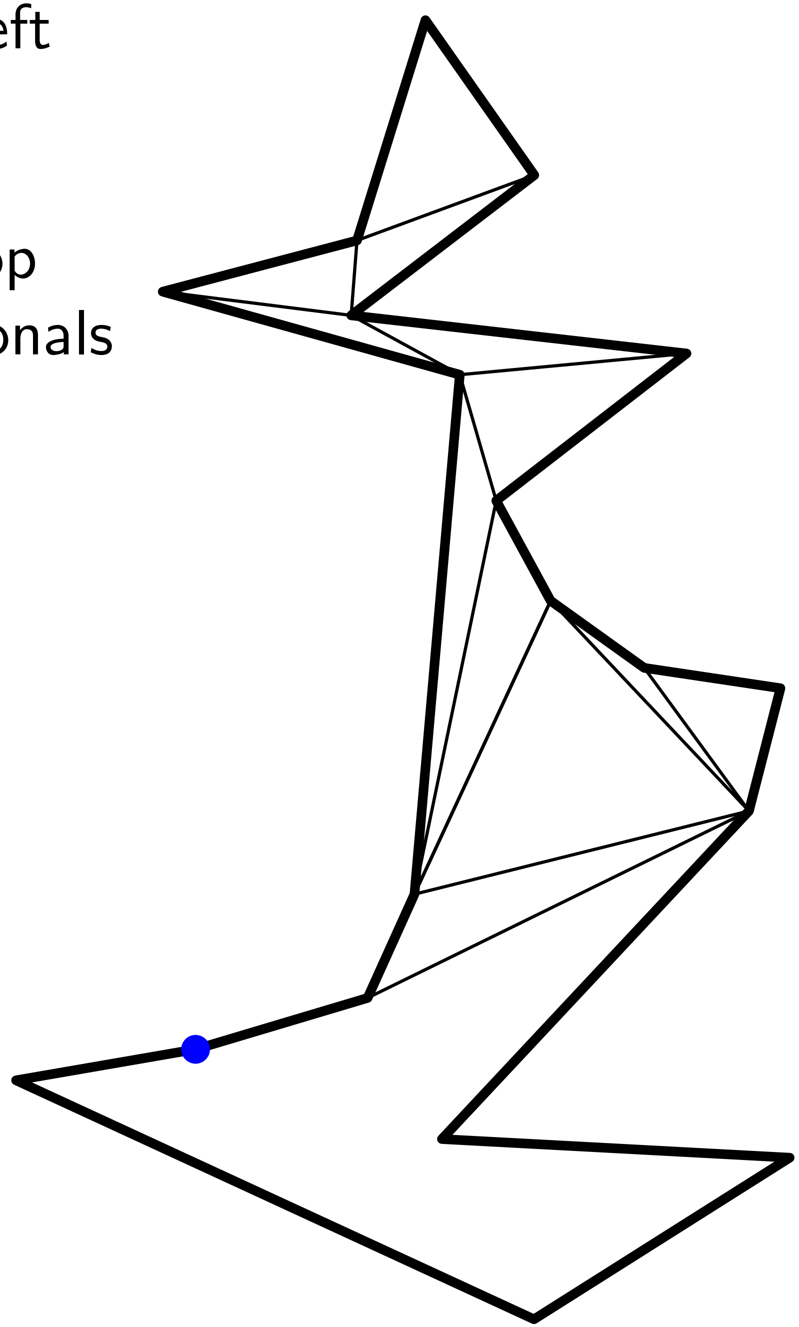
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



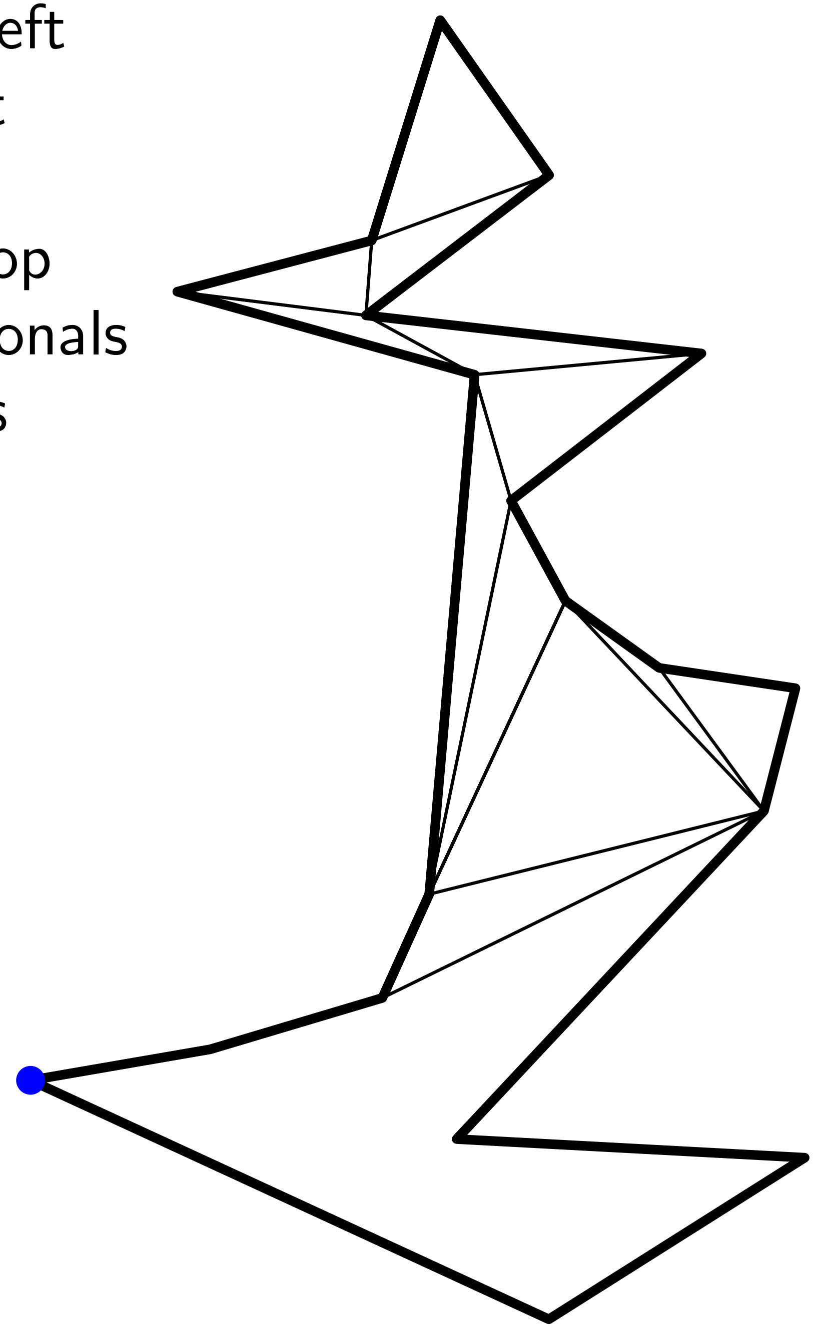
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



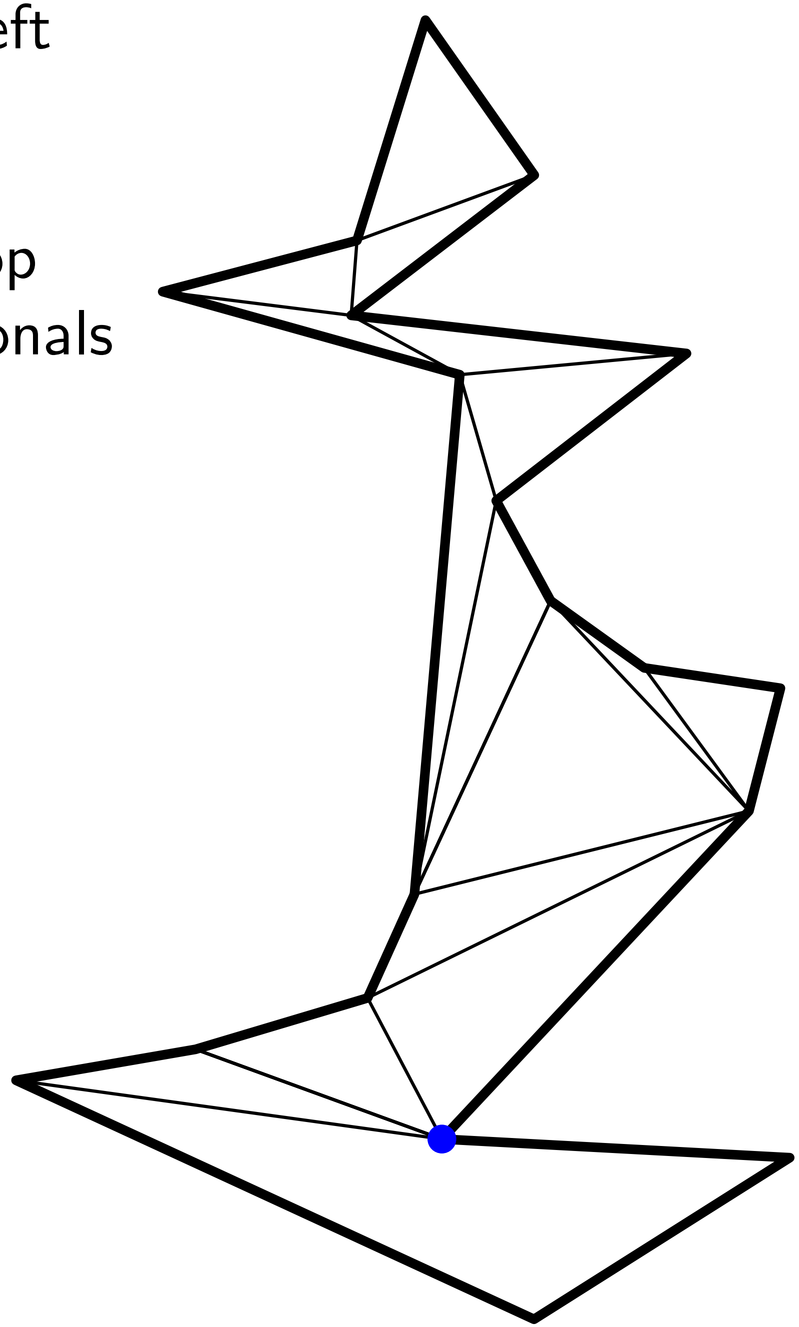
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



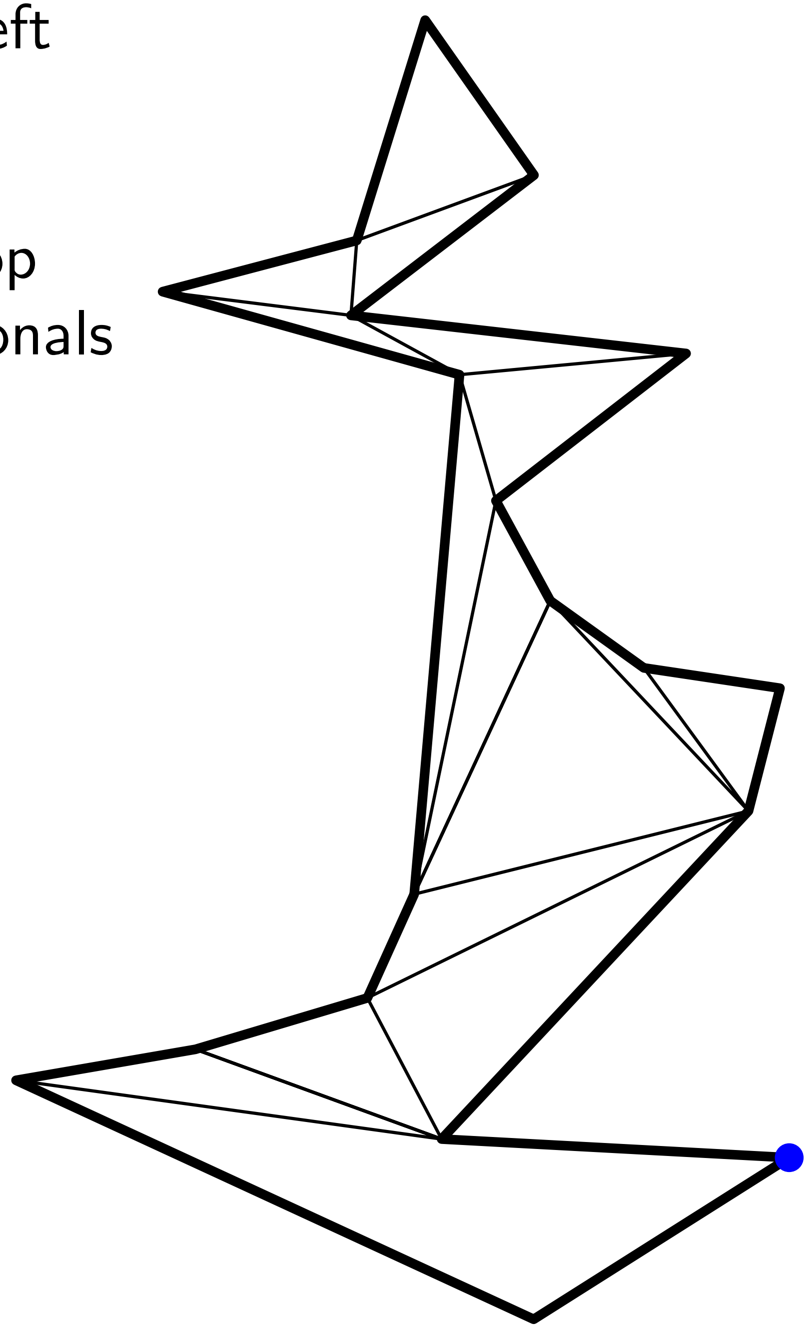
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



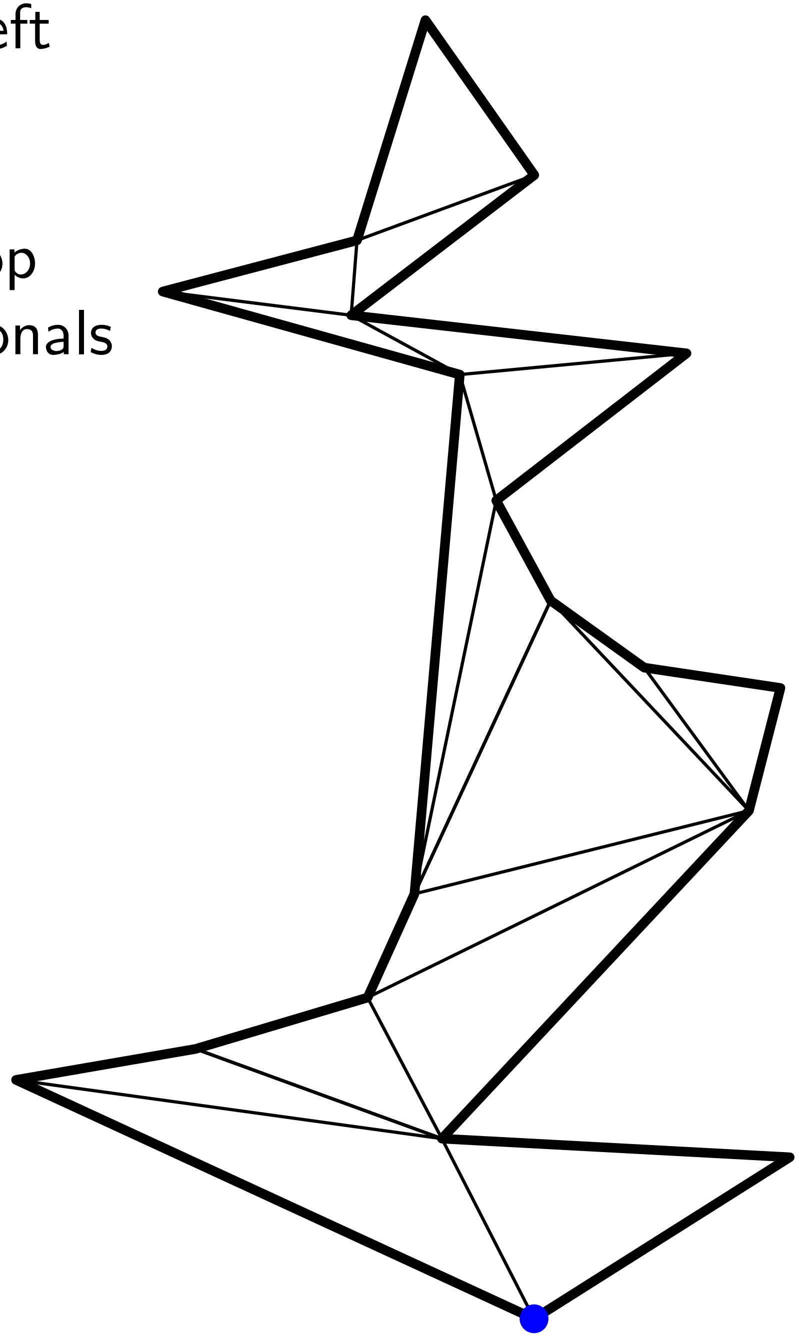
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above



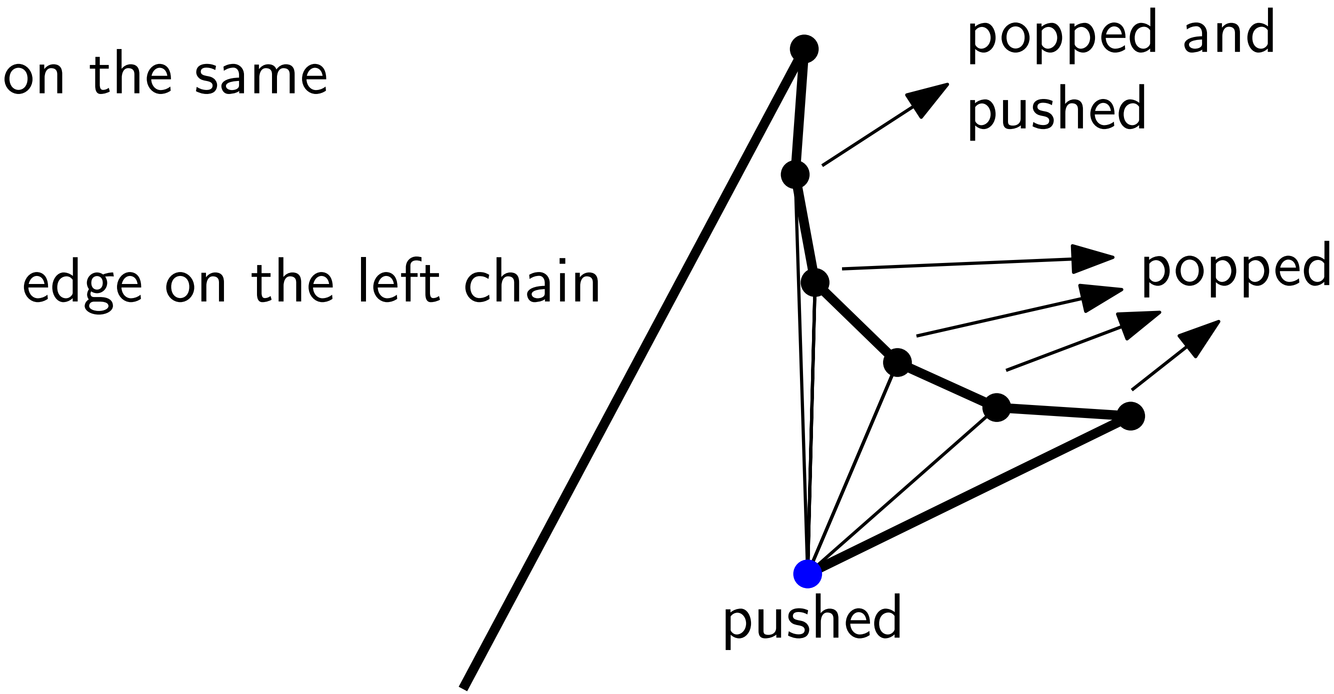
Triangulating a y-monotone polygon

1: Merge vertices of left and right chain to get sorted order
2: Traverse vertices top down and create diagonals to all possible vertices above

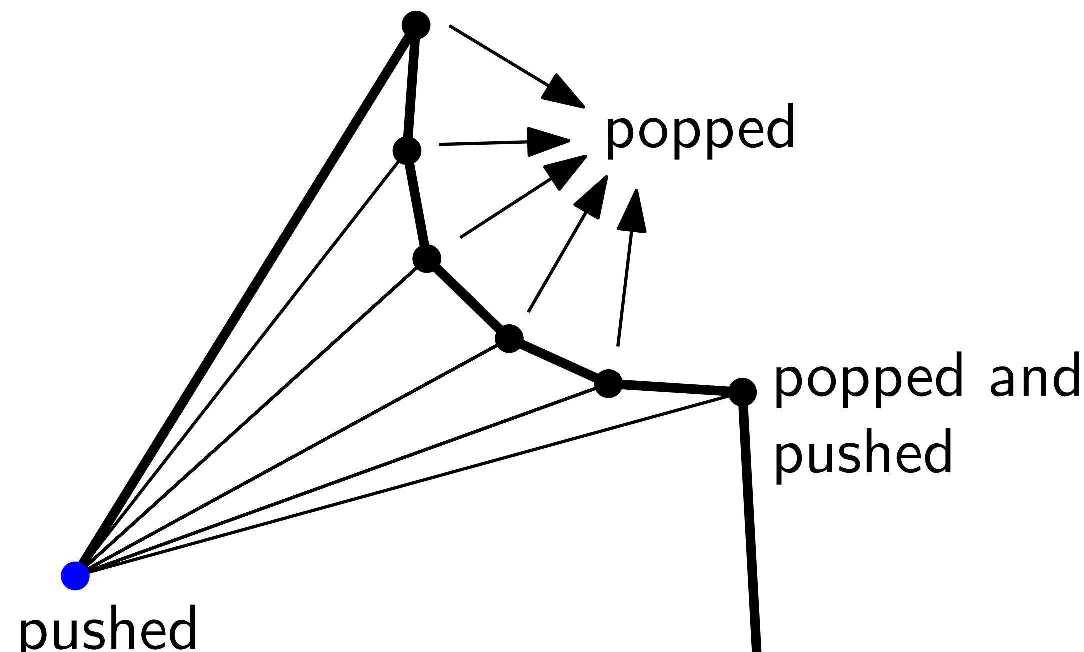


Using a stack to make diagonals

Case 1: New vertex on the same chain (here right).

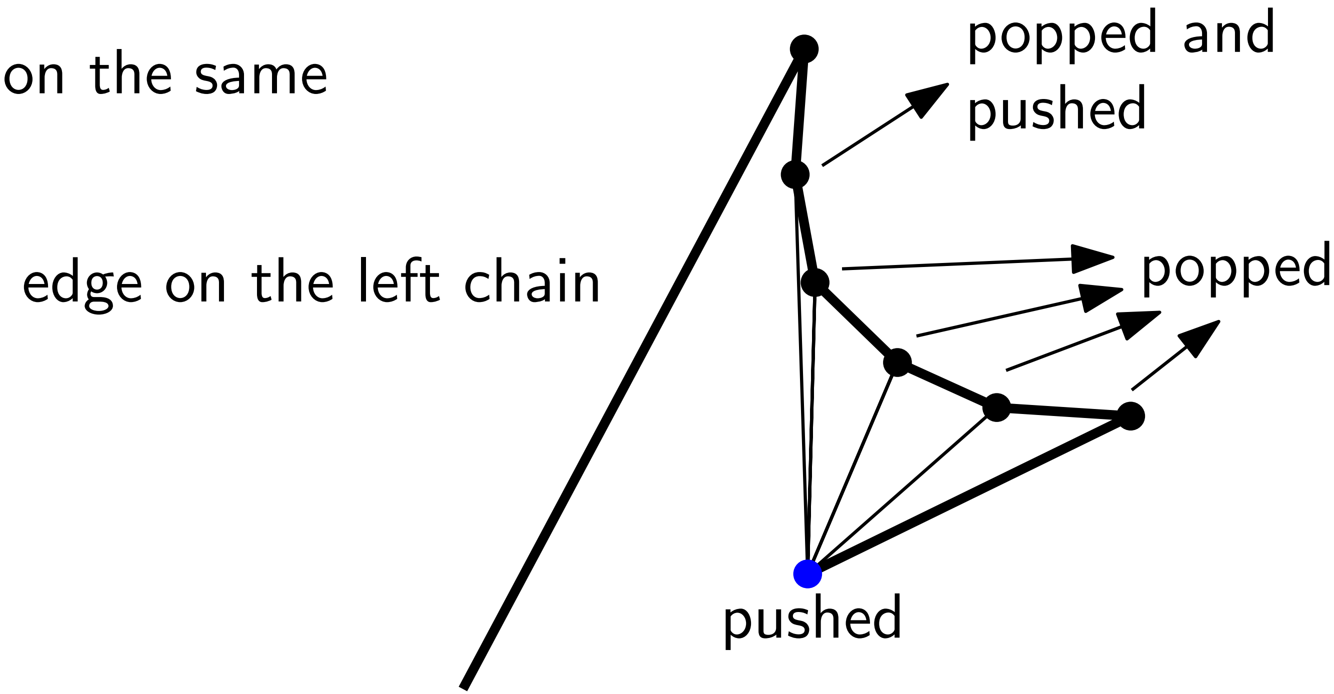


Case 2: New vertex on the other chain.



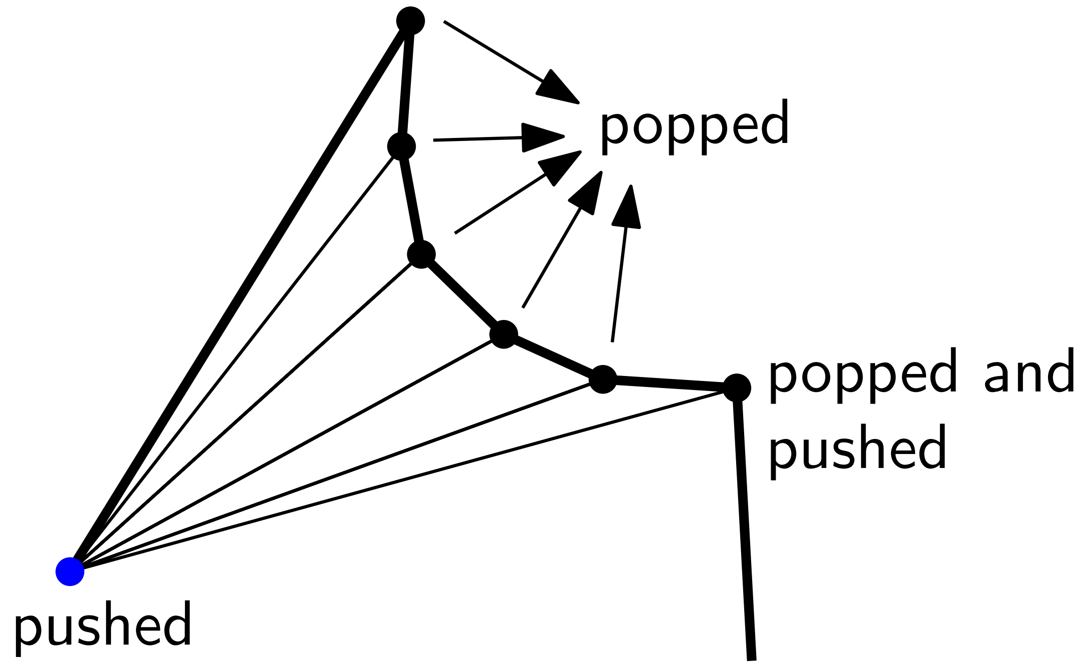
Using a stack to make diagonals

Case 1: New vertex on the same chain (here right).

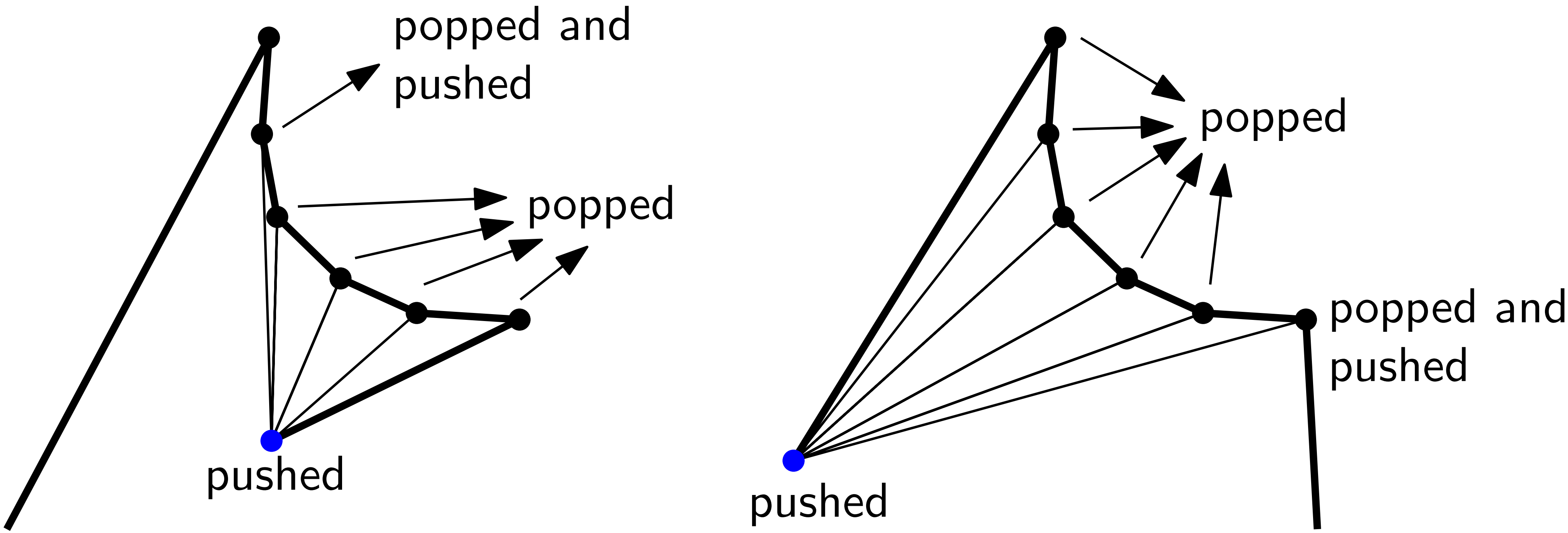


Case 2: New vertex on the other chain.

Why are all diagonals OK?



Time complexity

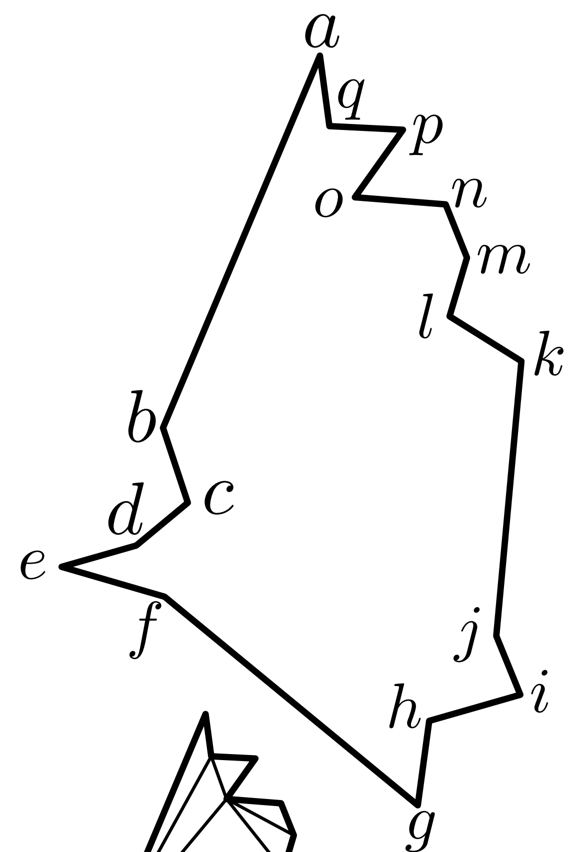


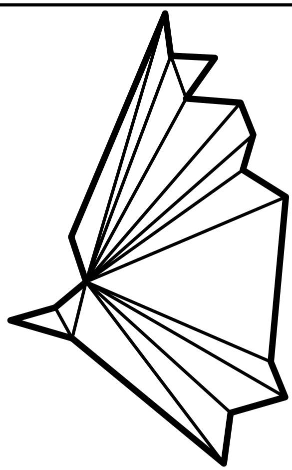
Note: In either case, we push two vertices $\Rightarrow \leq 2n$ pushes, pops and diagonal checks.

Time complexity: $O(n)$ for merging and traversing.
n: number of vertices of this y-monotone polygon

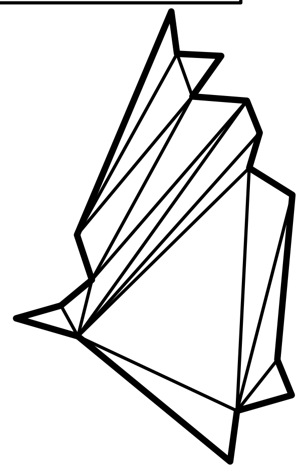
How does the triangulation look?

socrative.com → Student login,
Room name: ABRAHAMSEN3464

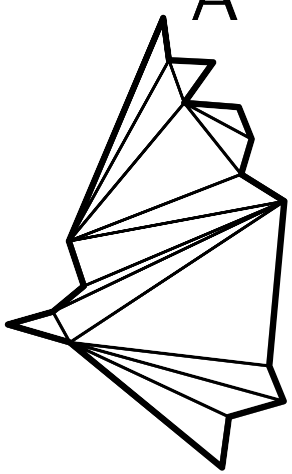




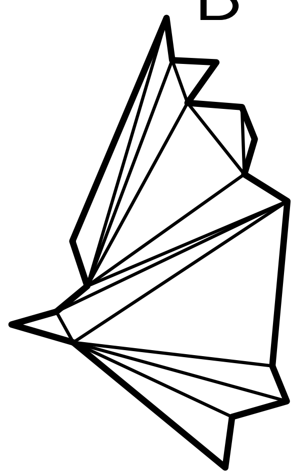
A



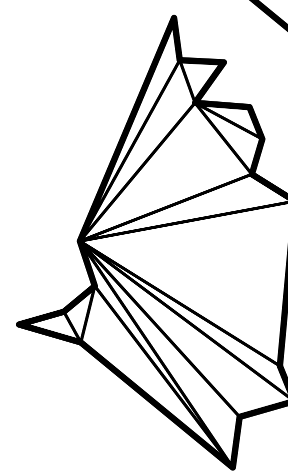
B



C

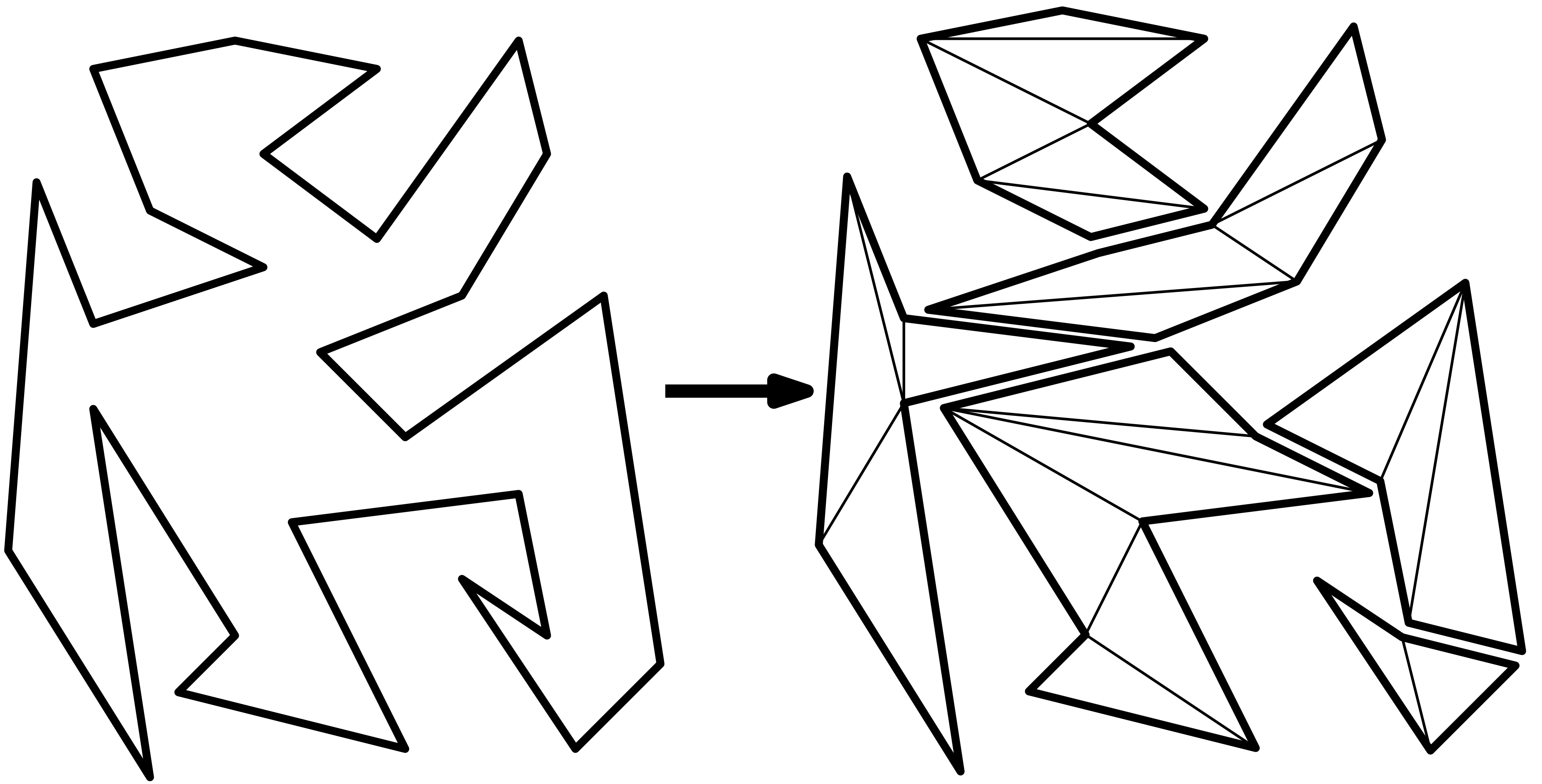


D



E

Total time complexity



$\leq 3n - 6$ vertices $\Rightarrow O(n)$ time to triangulate monotone polygons.
In total $O(n \log n)$ time (monotone partition dominates).

History of Triangulation Algorithms

1979: Garey, Johnson, Preparata, Tarjan. $O(n \log n)$ sweep-line algorithm, similar to this.
1982: Chazelle. $O(n \log n)$ divide-and-conquer algorithm.
1986: Tarjan and Van Wyk. $O(n \log \log n)$ algorithm.
1988: Clarkson, Tarjan, and Van Wyk. Randomized $O(n \log^* n)$ algorithm. Two other algorithms with same running time around the same time.
1990: Chazelle. Optimal $O(n)$ algorithm.

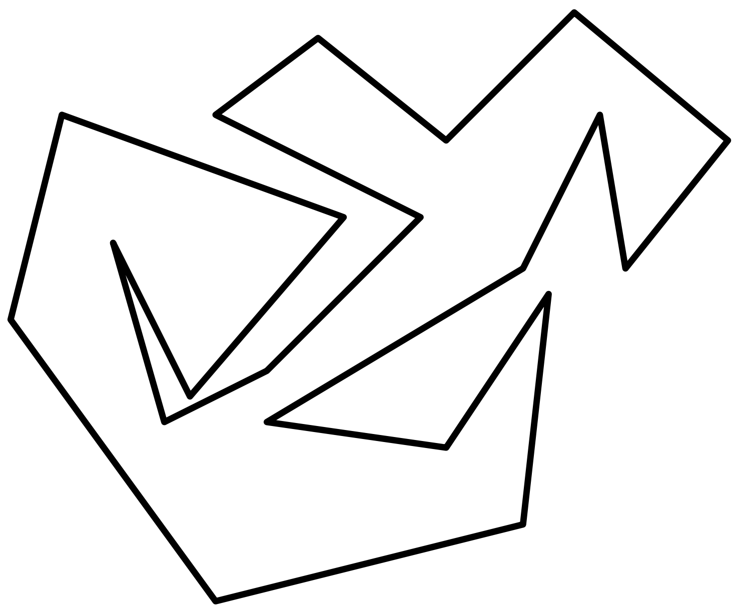
$\log^* n$: number of times to apply $\log$ before we get to ≤ 1.

$$\log^* n = \begin{cases} 0, & \text{if } n \leq 1, \\ 1 + \log^*(\log n), & \text{if } n > 1. \end{cases}$$

n	$\log^* n$
$(-\infty, 1]$	0
$(1, 2]$	1
$(2, 4]$	2
$(4, 16]$	3
$(16, 65536]$	4
$(65536, 2^{65536}]$	5

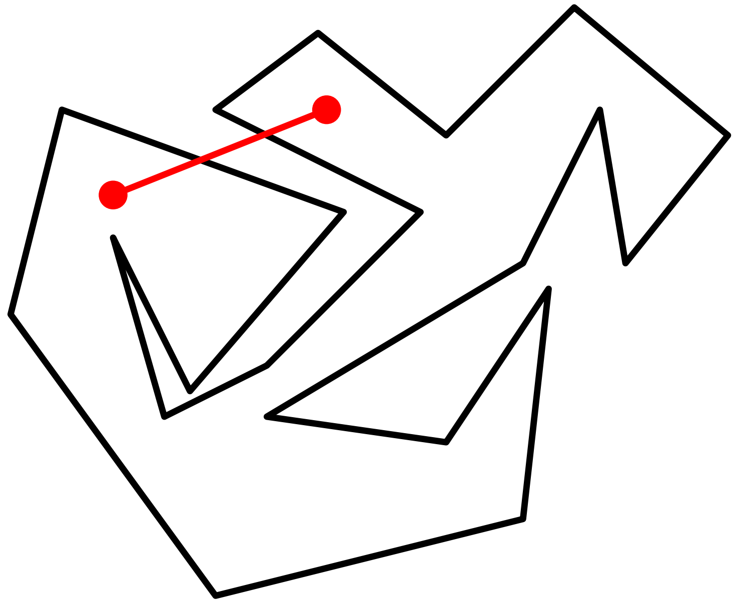
Use of Triangulations

Visibility problems.



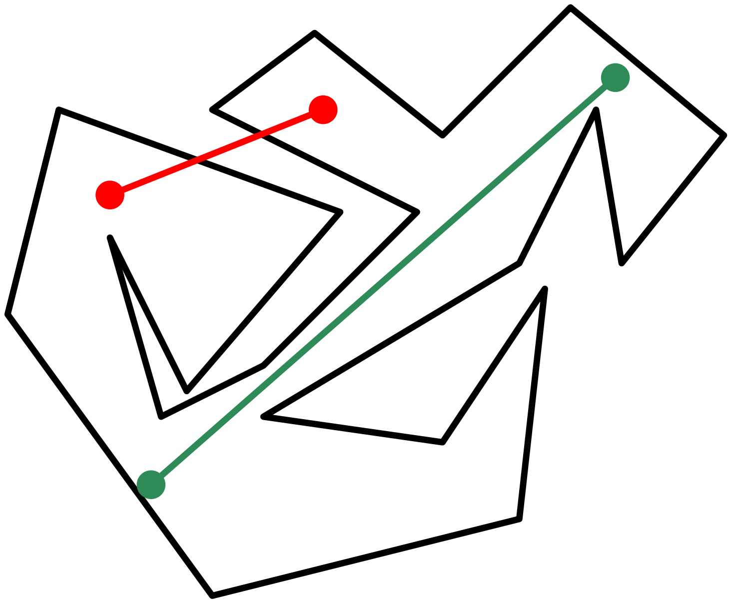
Use of Triangulations

Visibility problems.



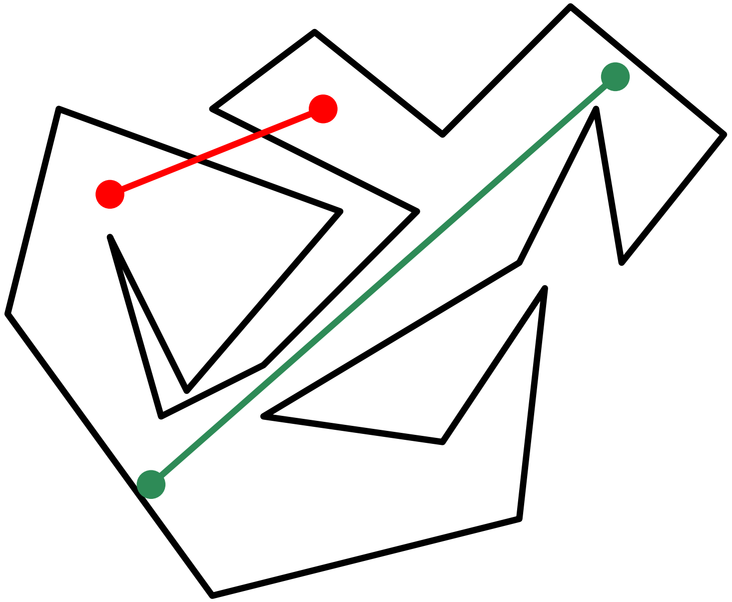
Use of Triangulations

Visibility problems.

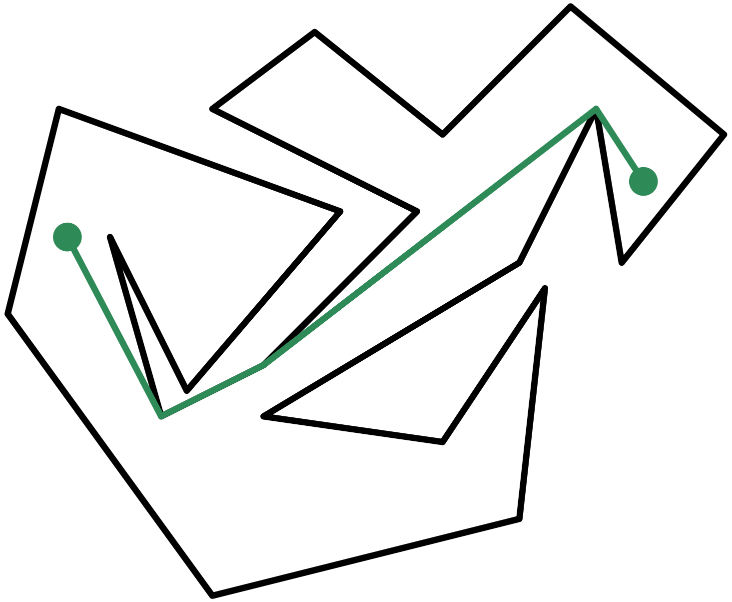


Use of Triangulations

Visibility problems.

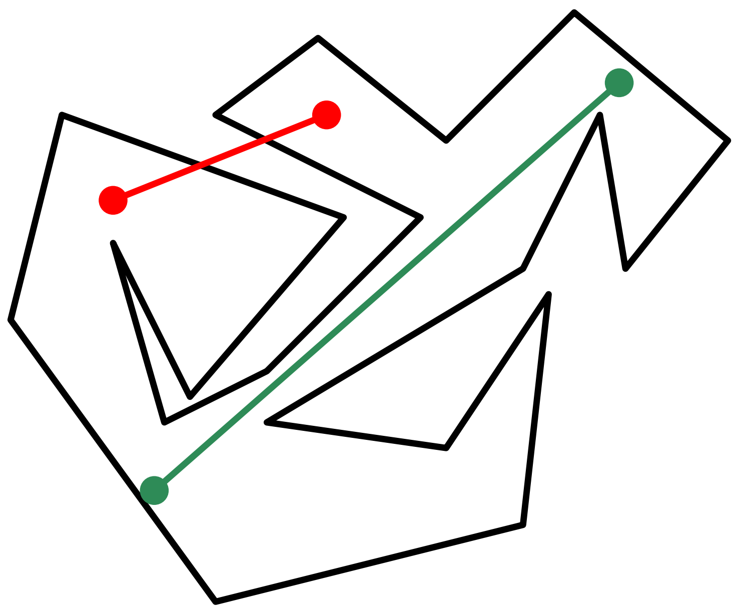


Shortest paths.

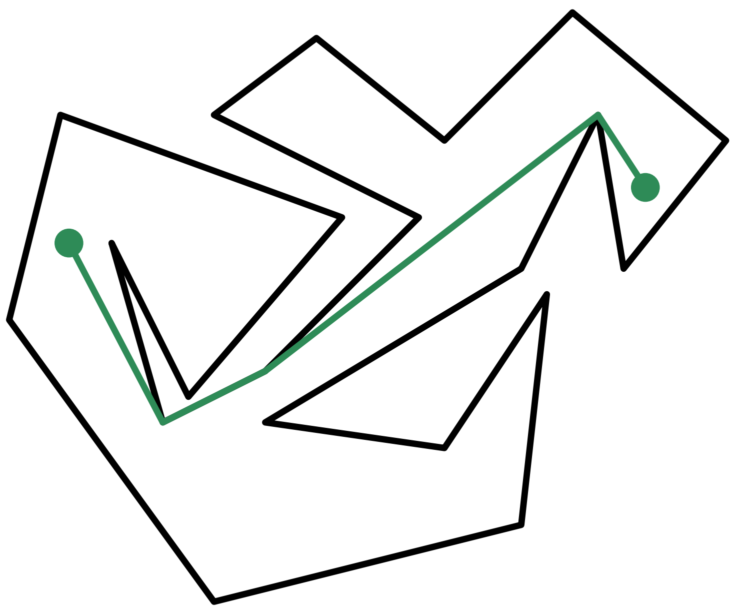


Use of Triangulations

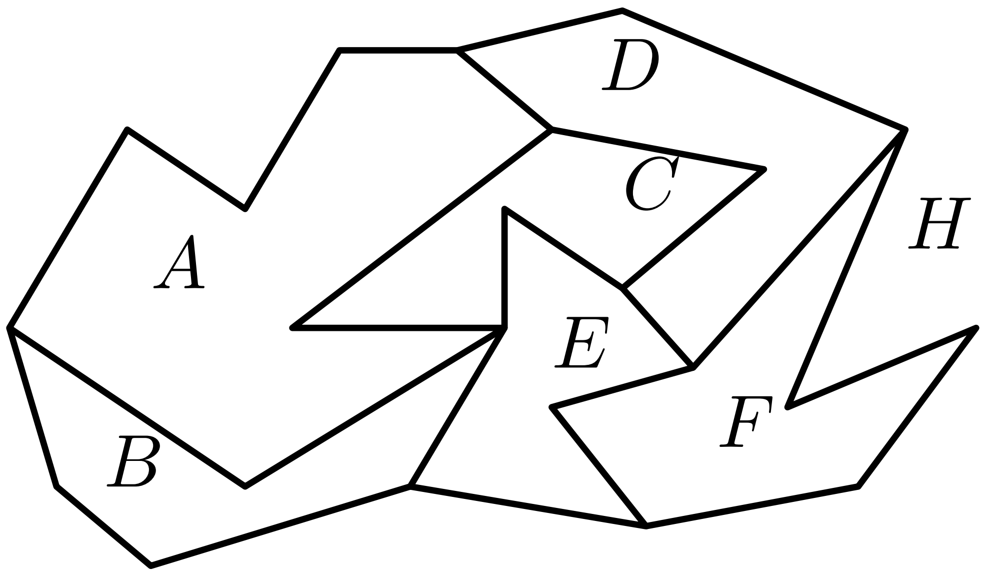
Visibility problems.



Shortest paths.

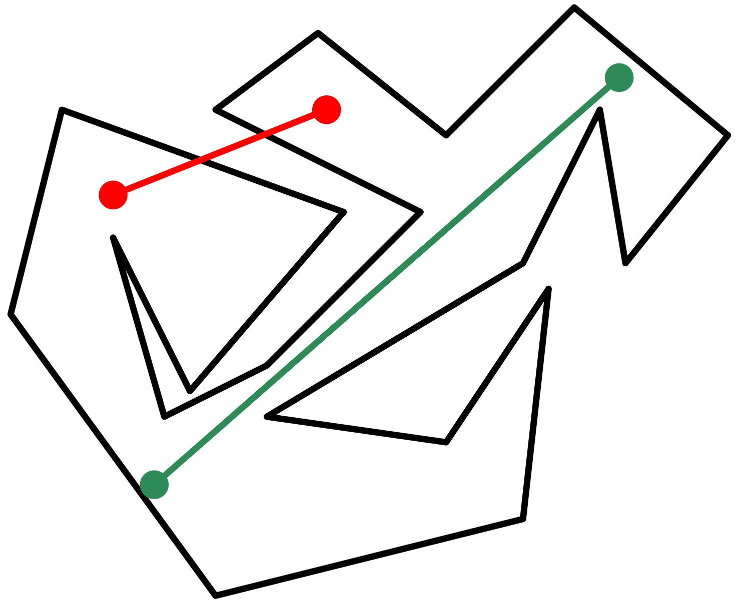


Point location.

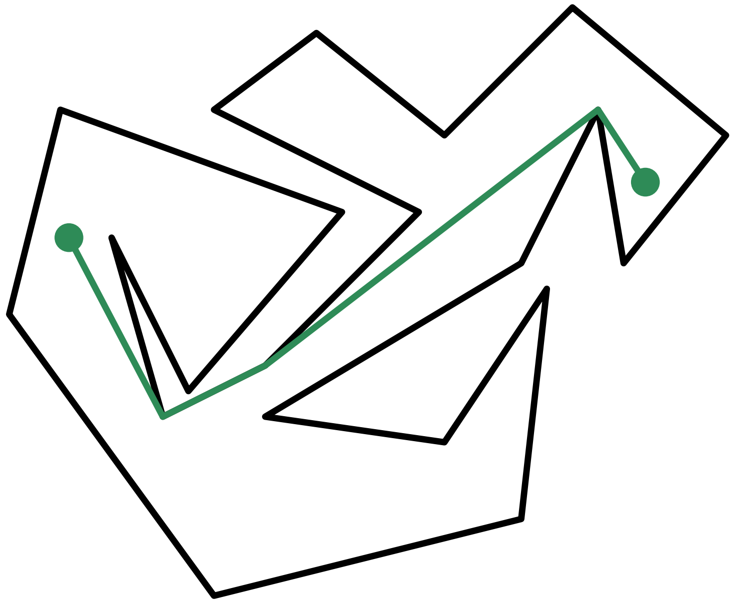


Use of Triangulations

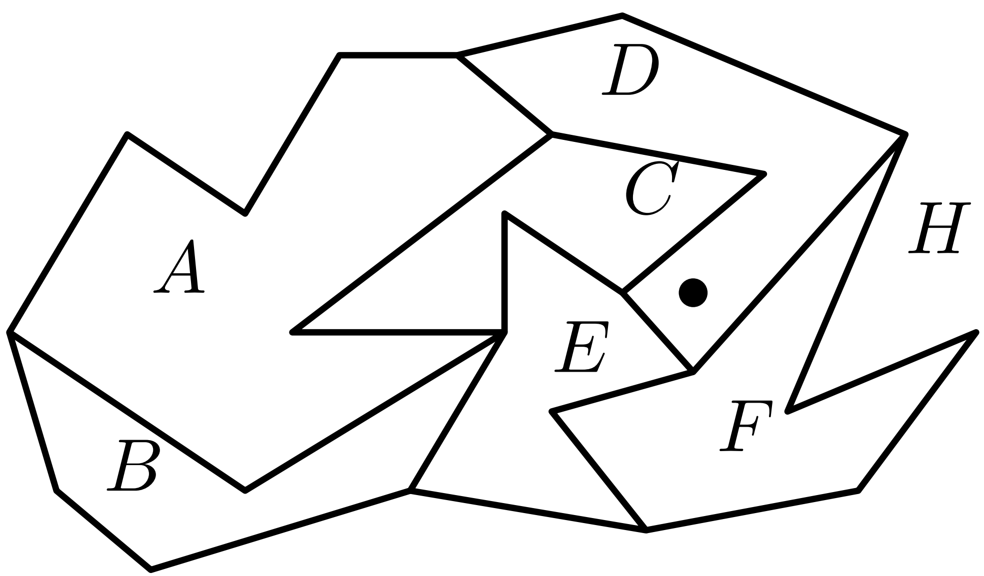
Visibility problems.



Shortest paths.

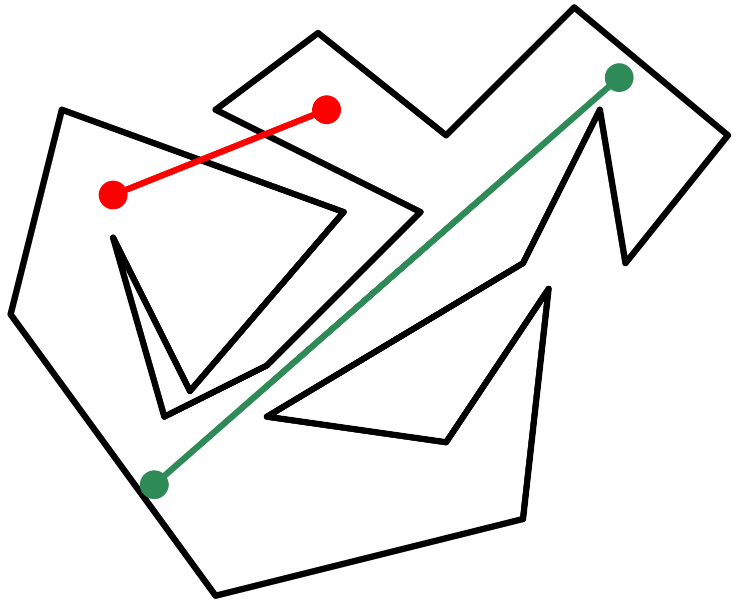


Point location.

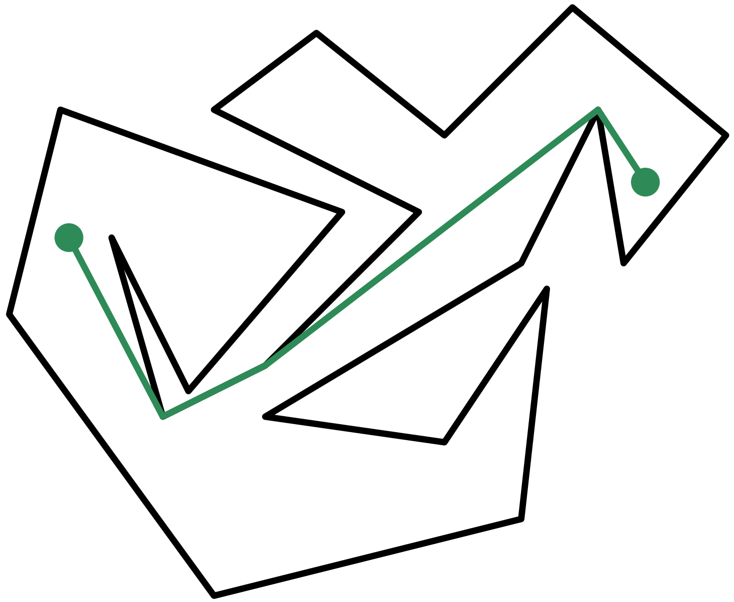


Use of Triangulations

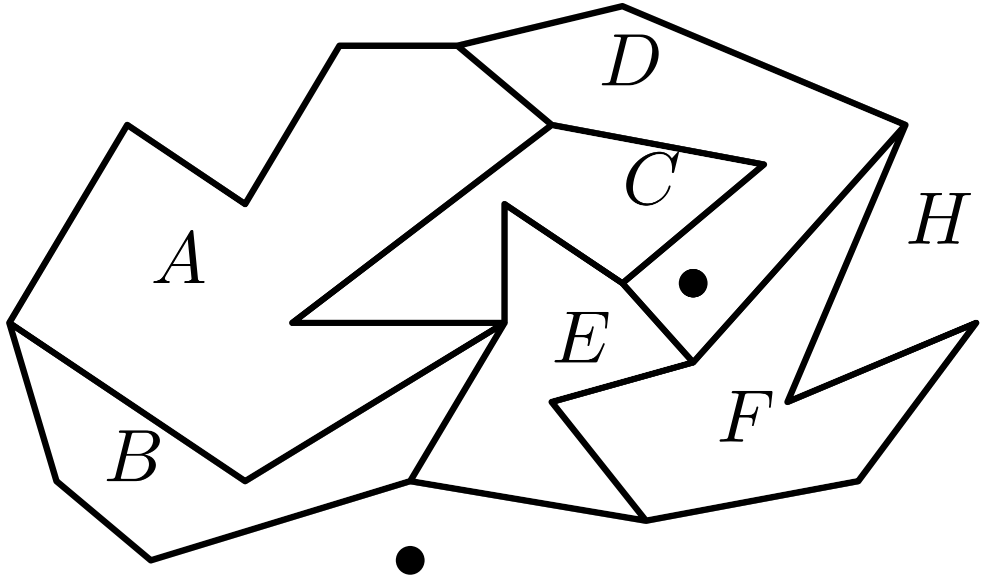
Visibility problems.



Shortest paths.

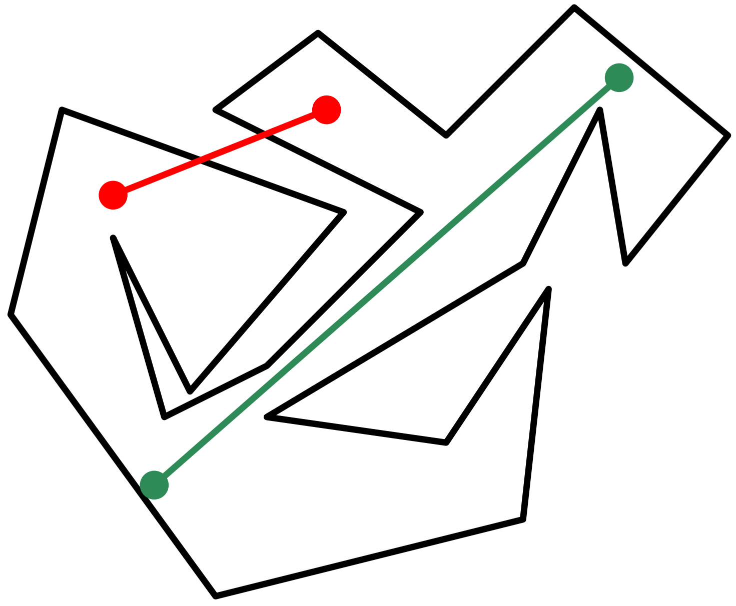


Point location.

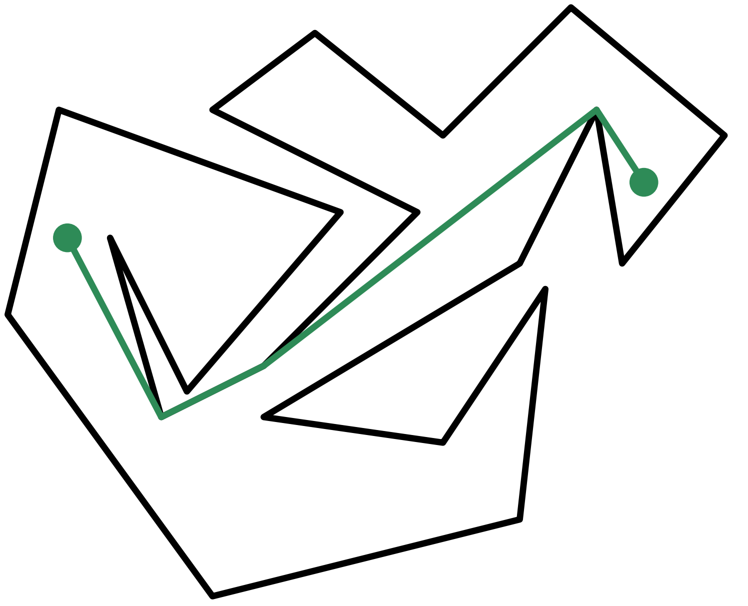


Use of Triangulations

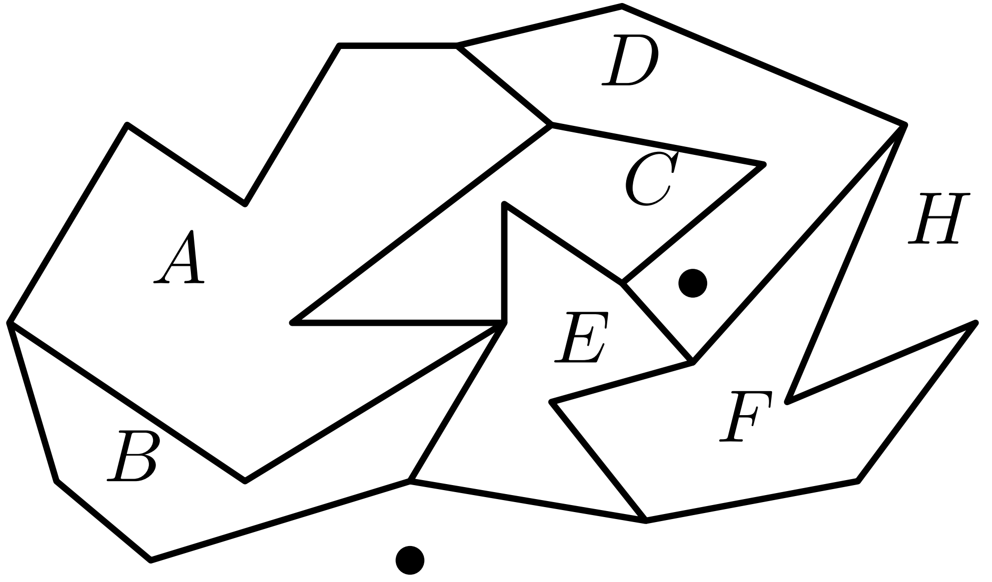
Visibility problems.



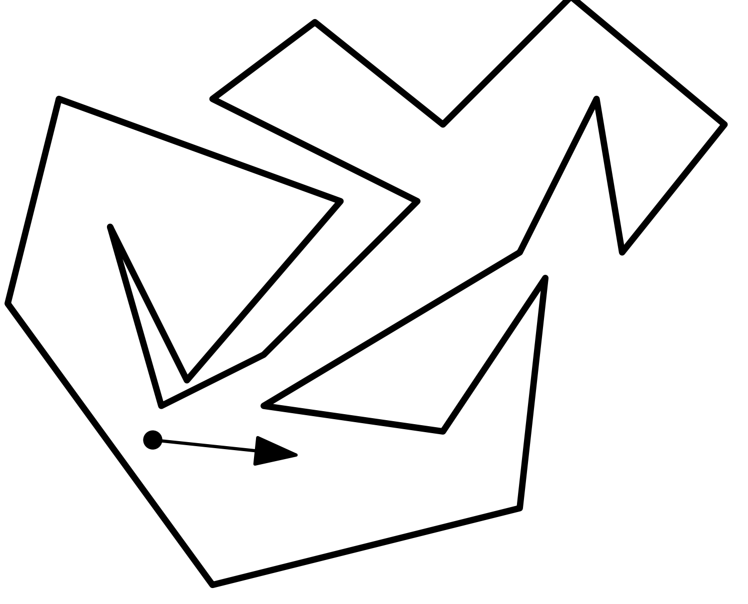
Shortest paths.



Point location.

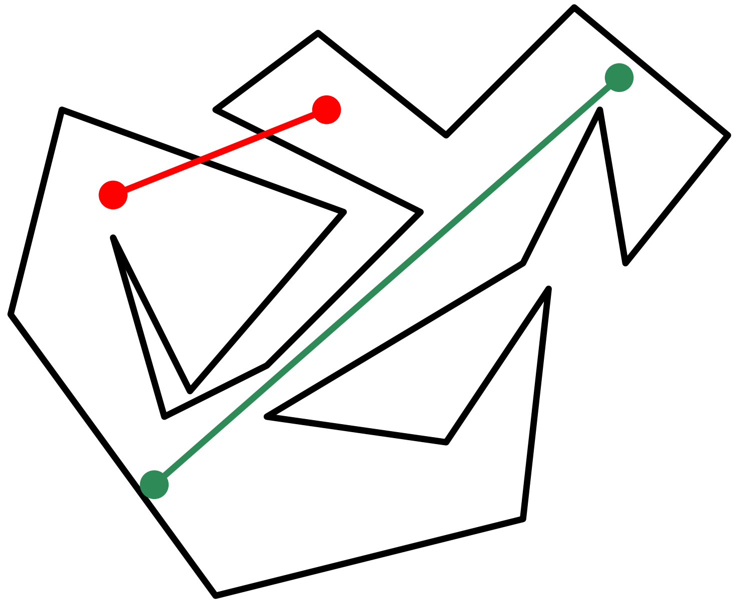


Ray shooting.

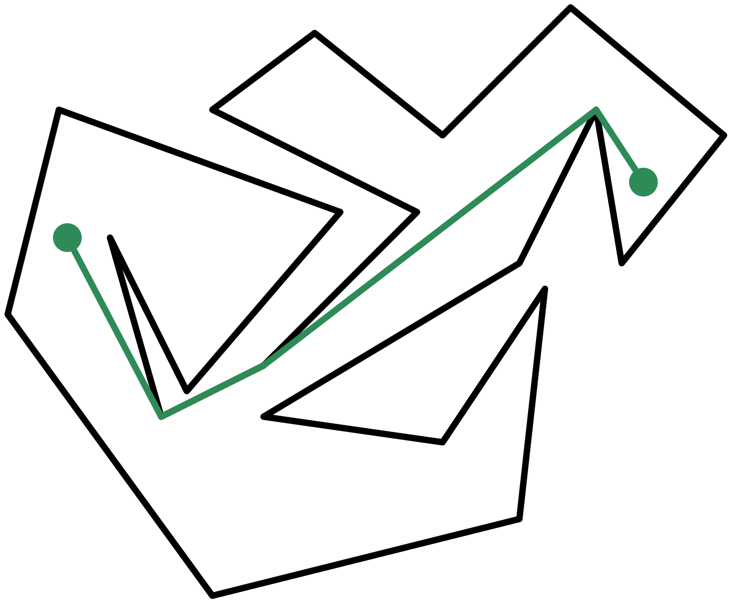


Use of Triangulations

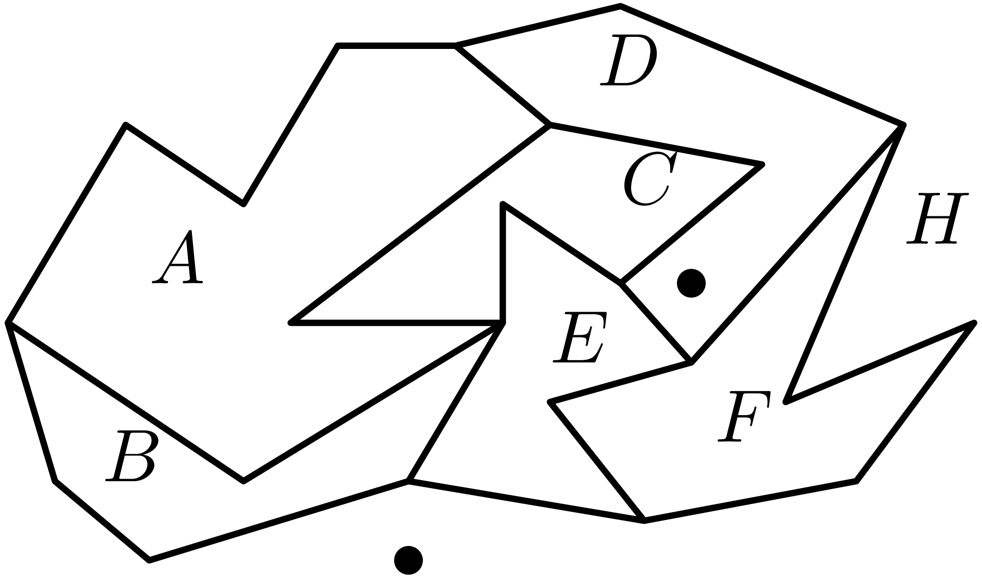
Visibility problems.



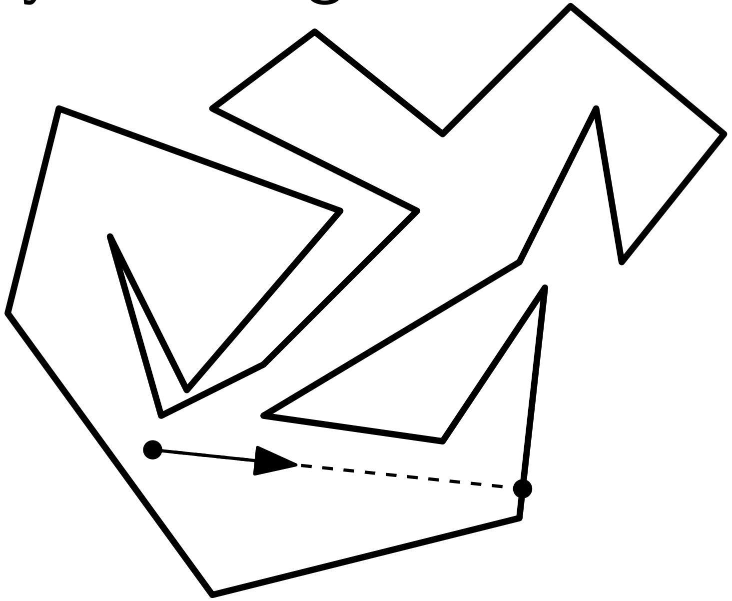
Shortest paths.



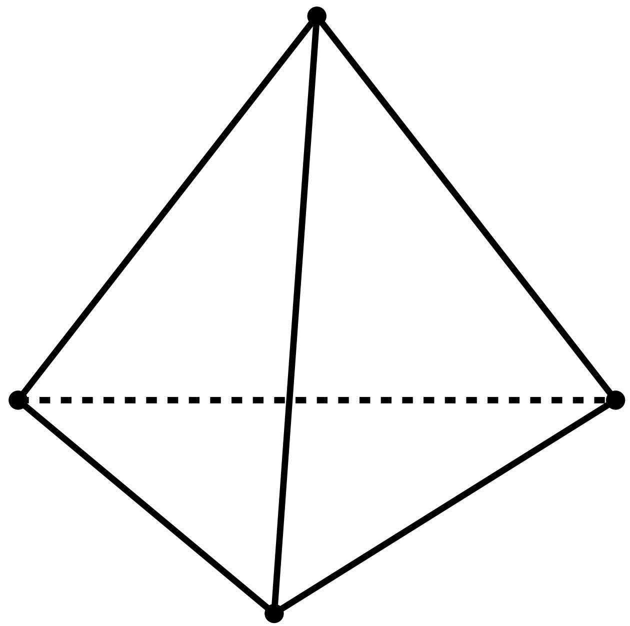
Point location.

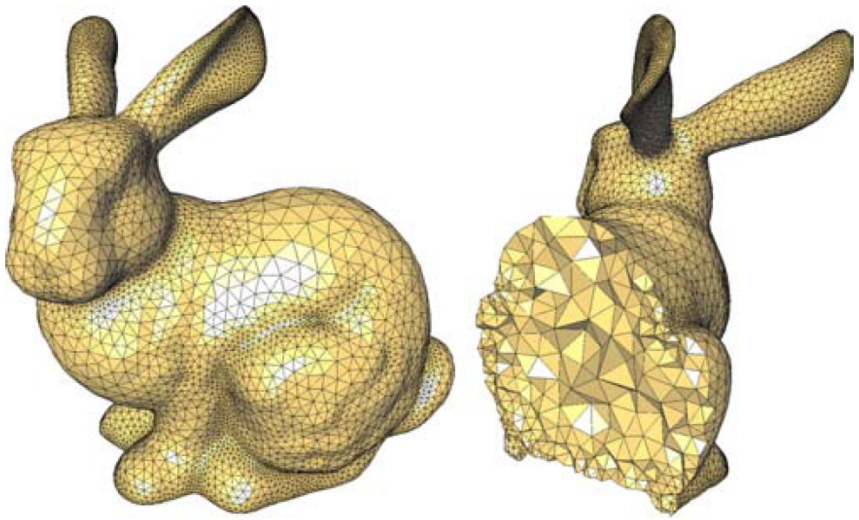


Ray shooting.

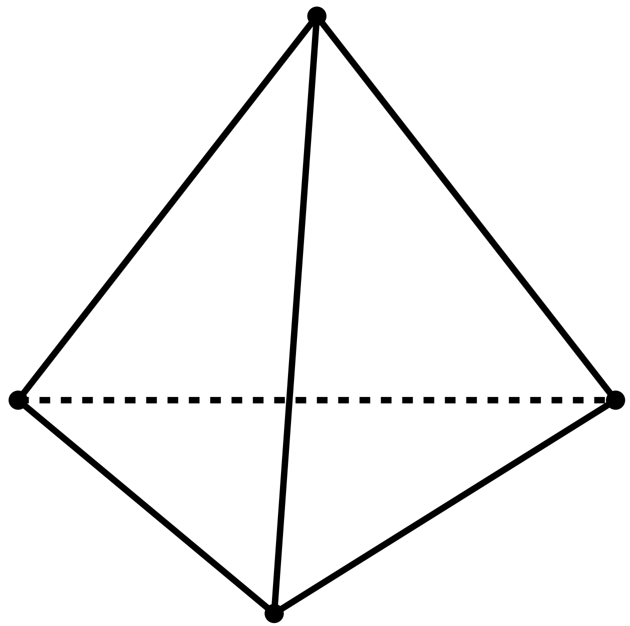


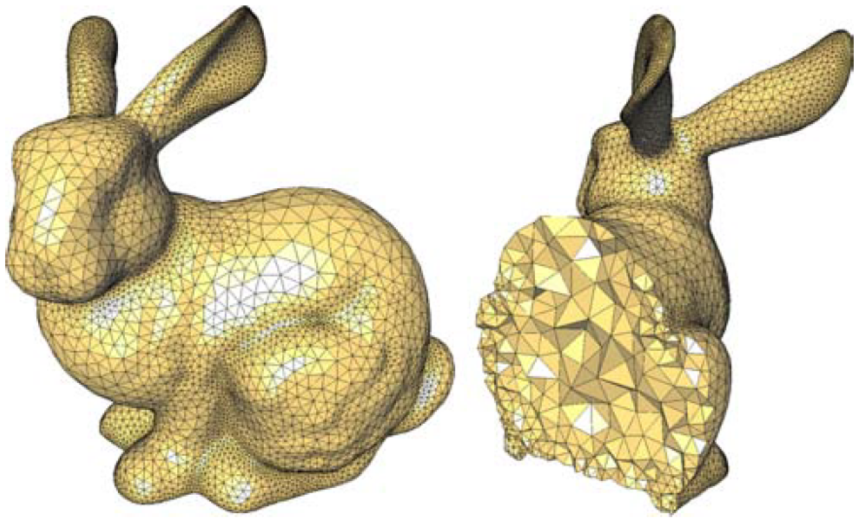
3D

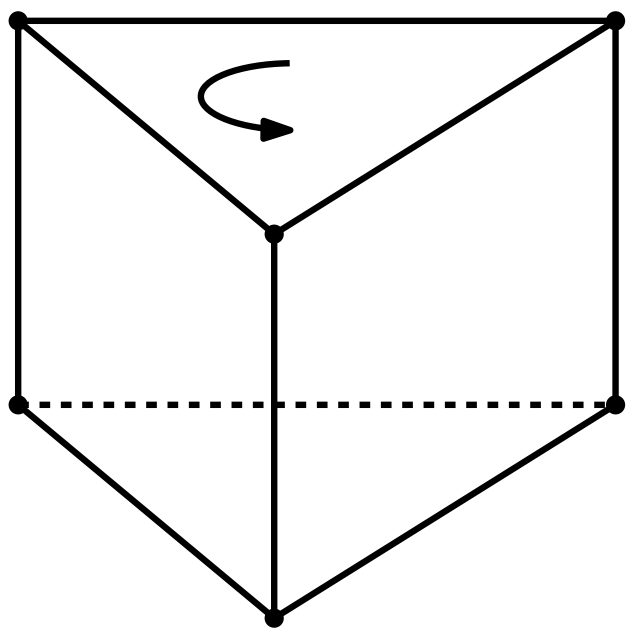




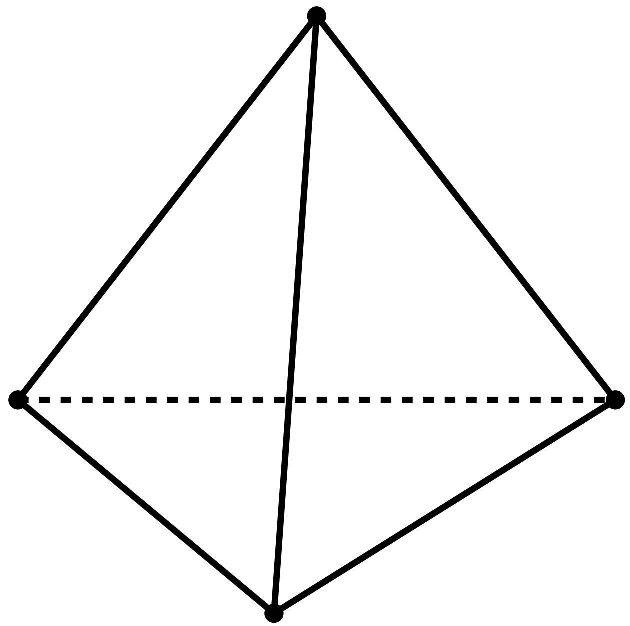
3D

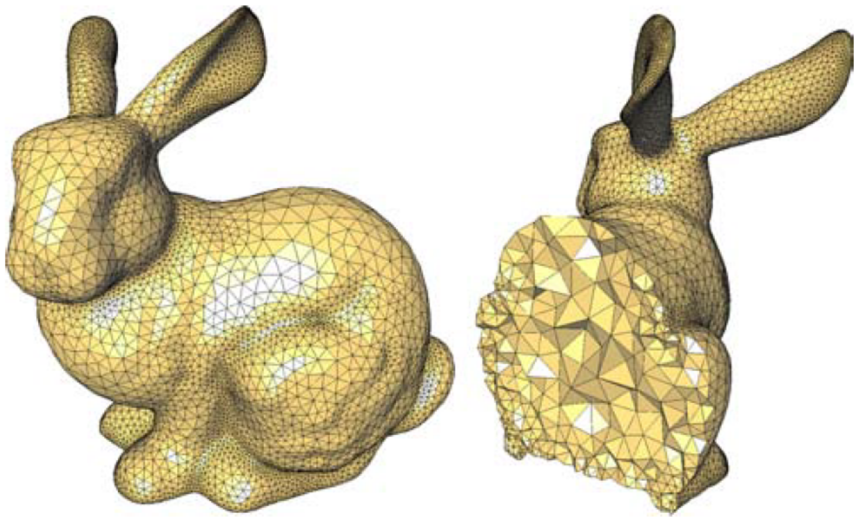


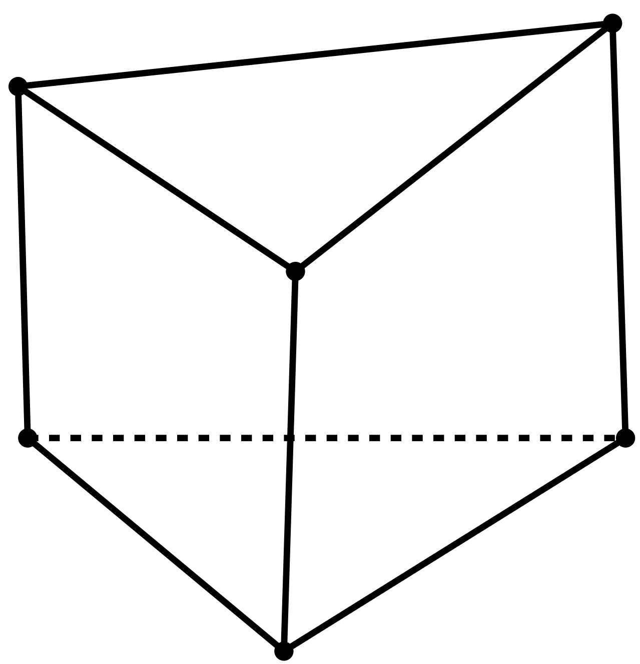




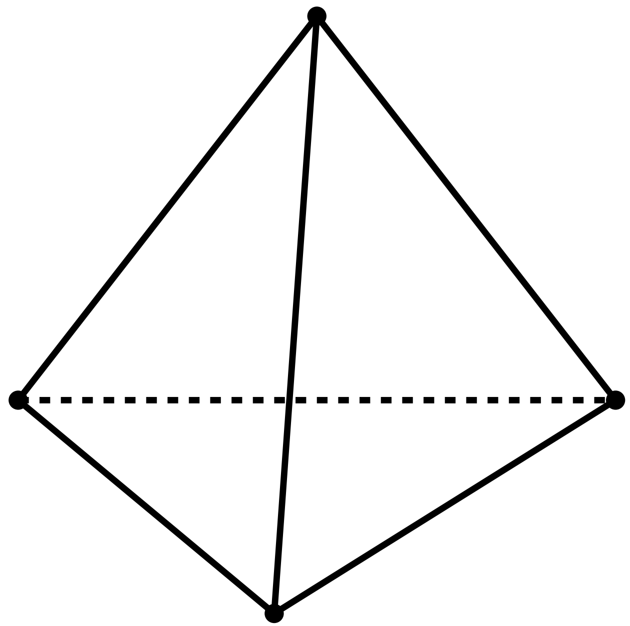
3D

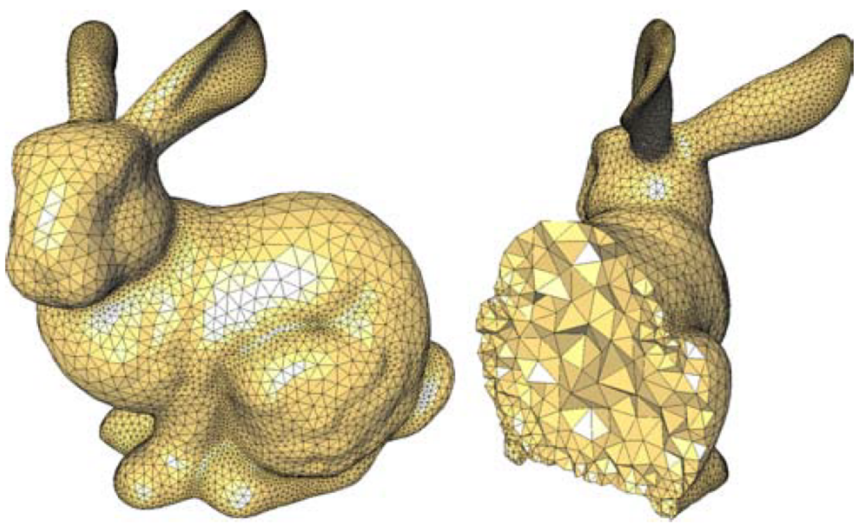


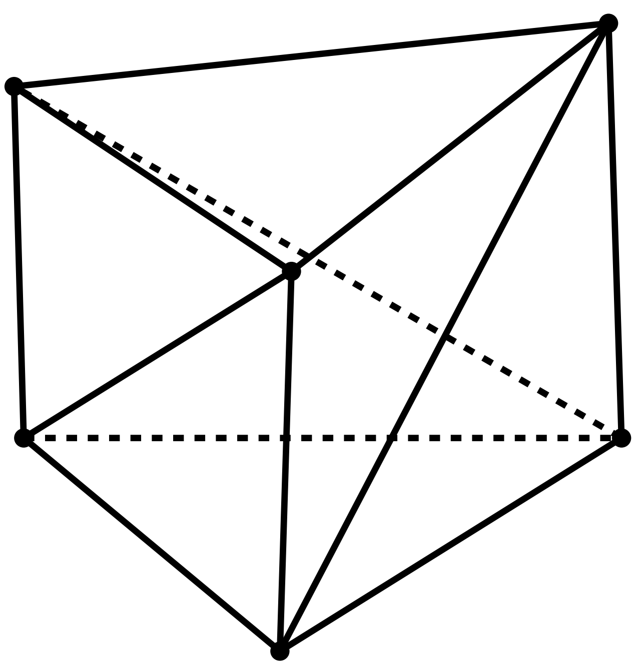




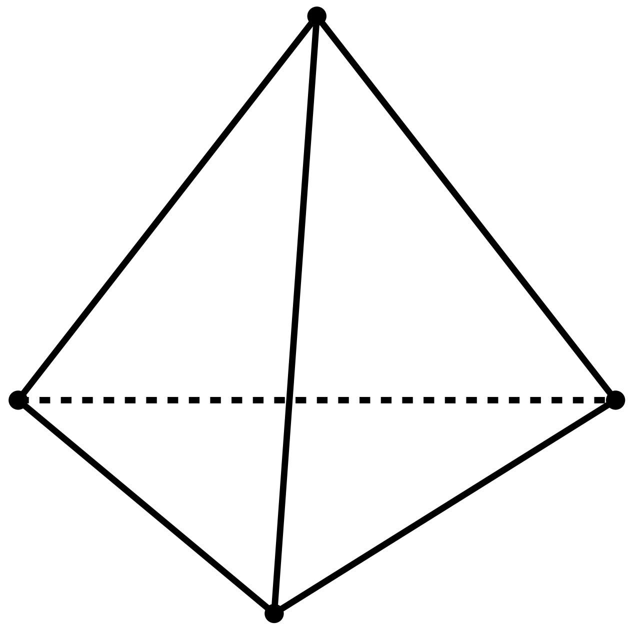
3D

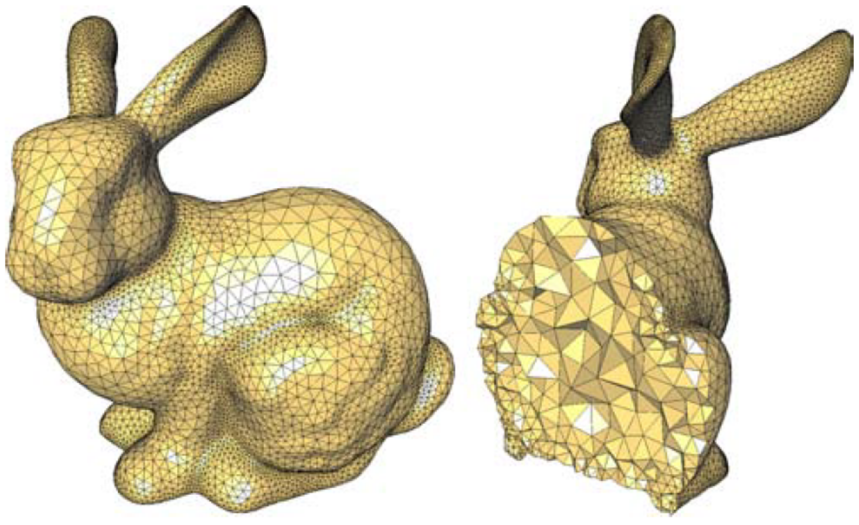


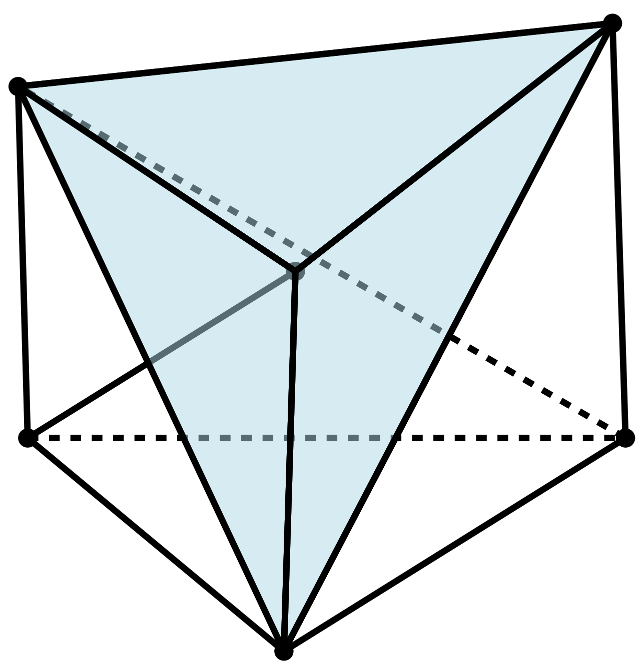




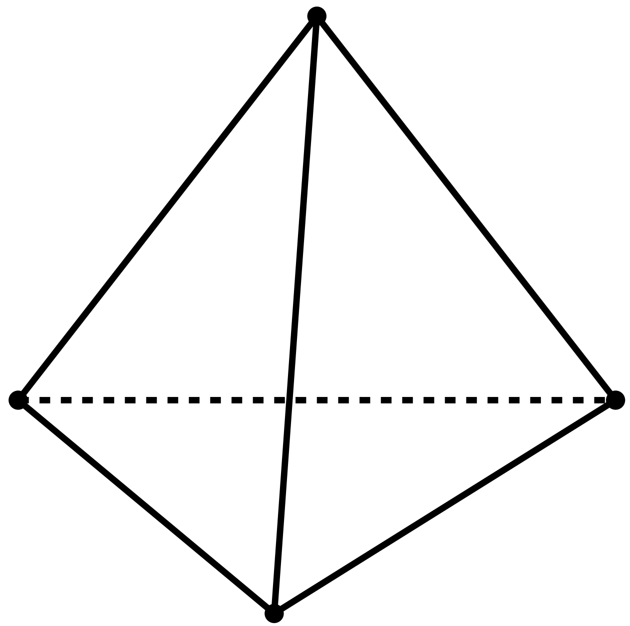
3D

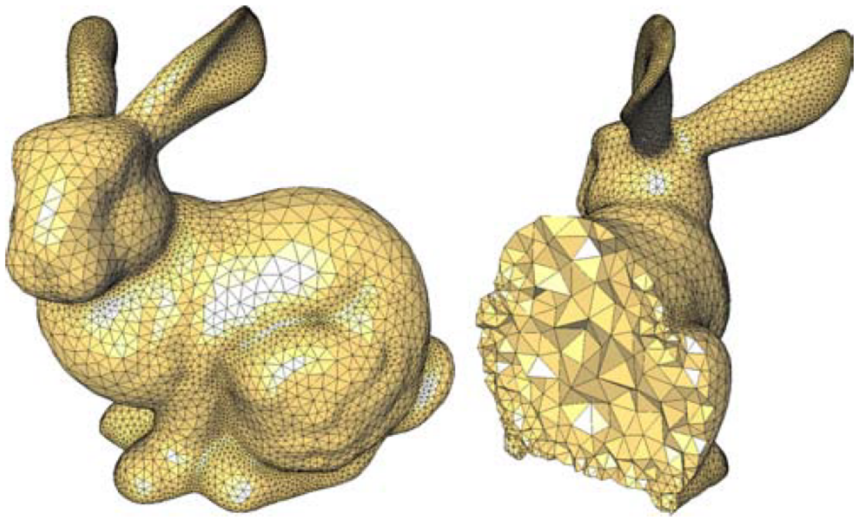


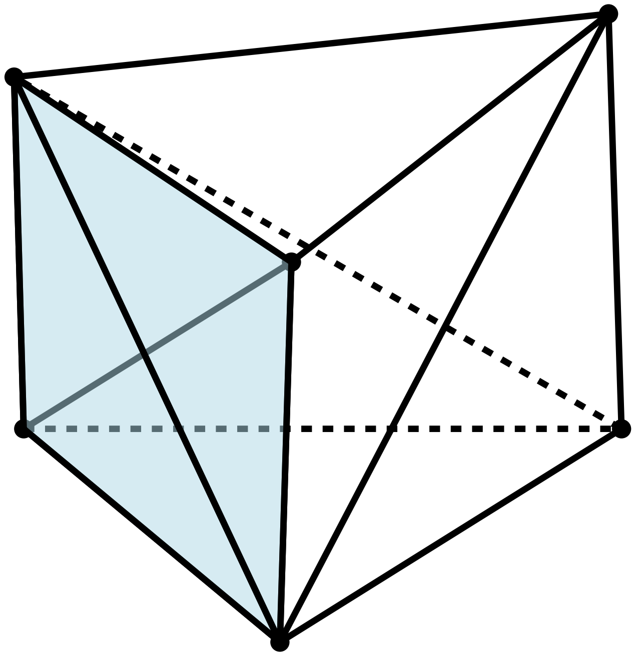




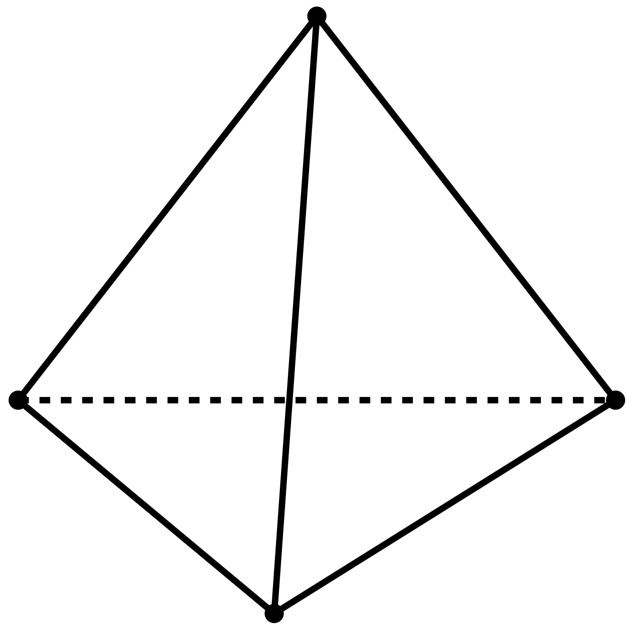
3D

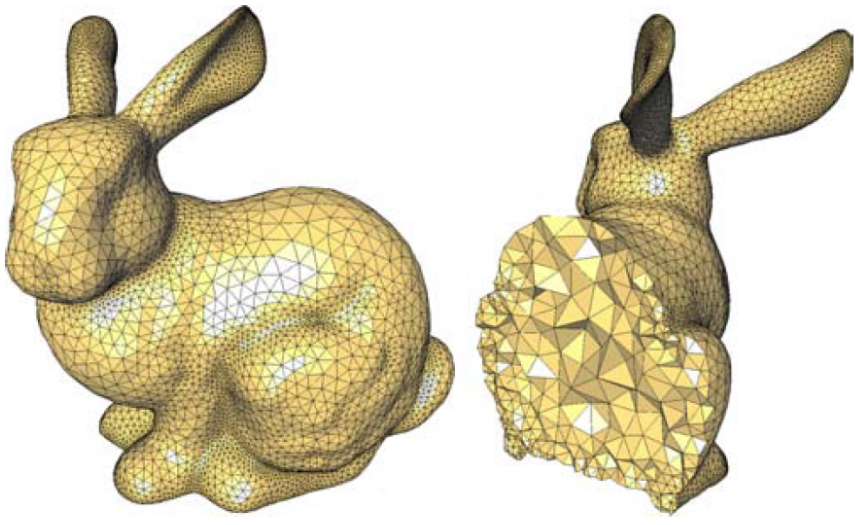


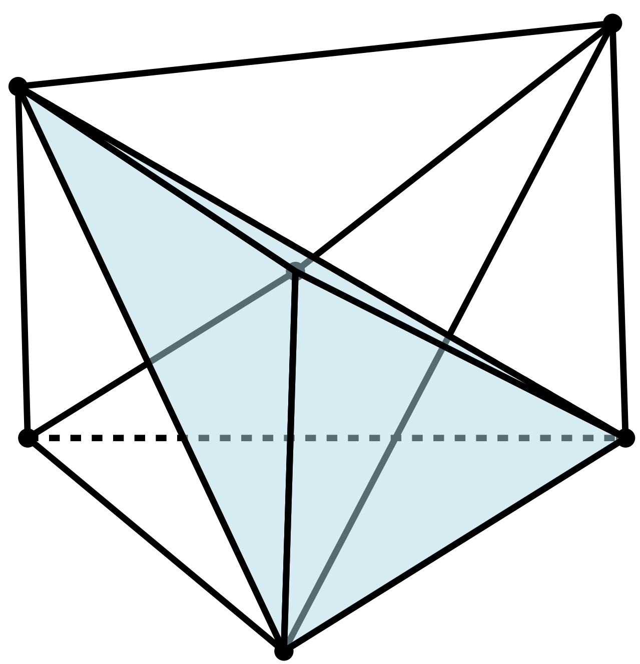




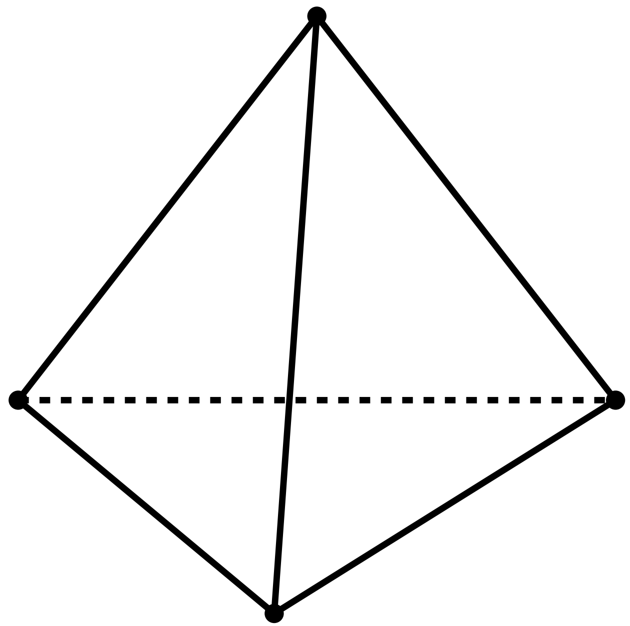
3D

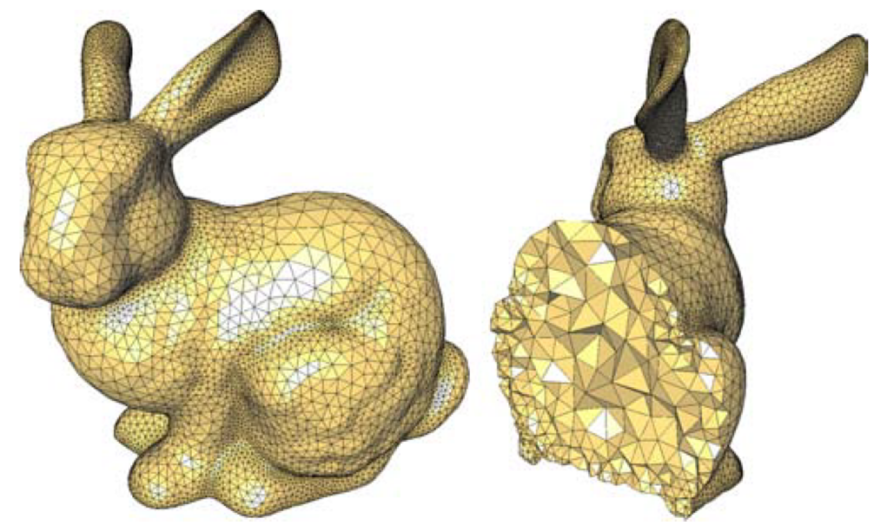


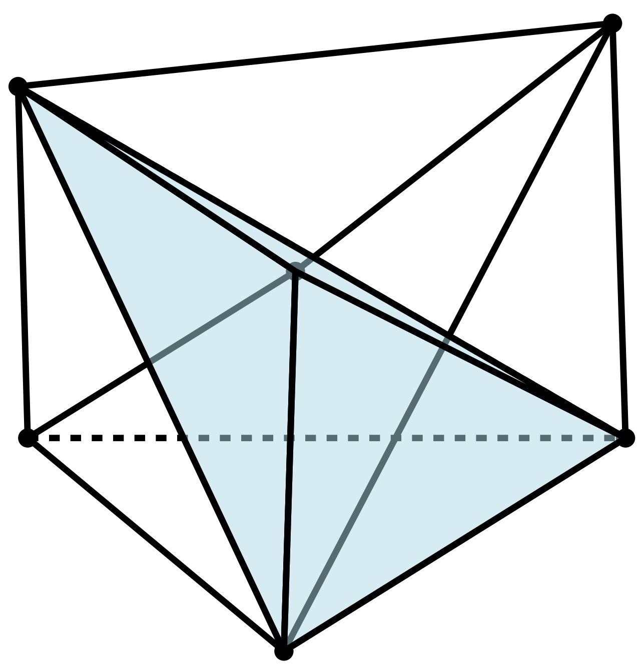




3D







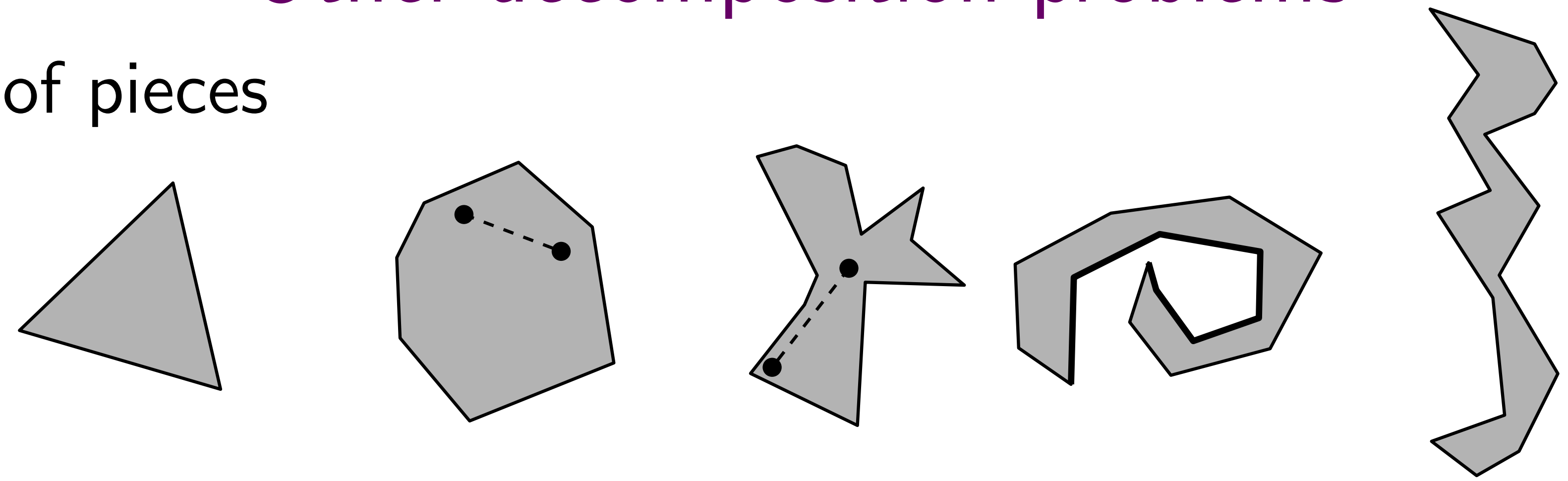
Sometimes $\Theta(n^2)$ Steiner points are needed. NP-complete to decide if possible without Steiner points.

Even when possible without Steiner points, it differs how many tetrahedra are needed.

For some convex polyhedra with n corners, there exist different tetrahedralizations without Steiner points of sizes $\Theta(n)$ and $\Theta(n^2)$.

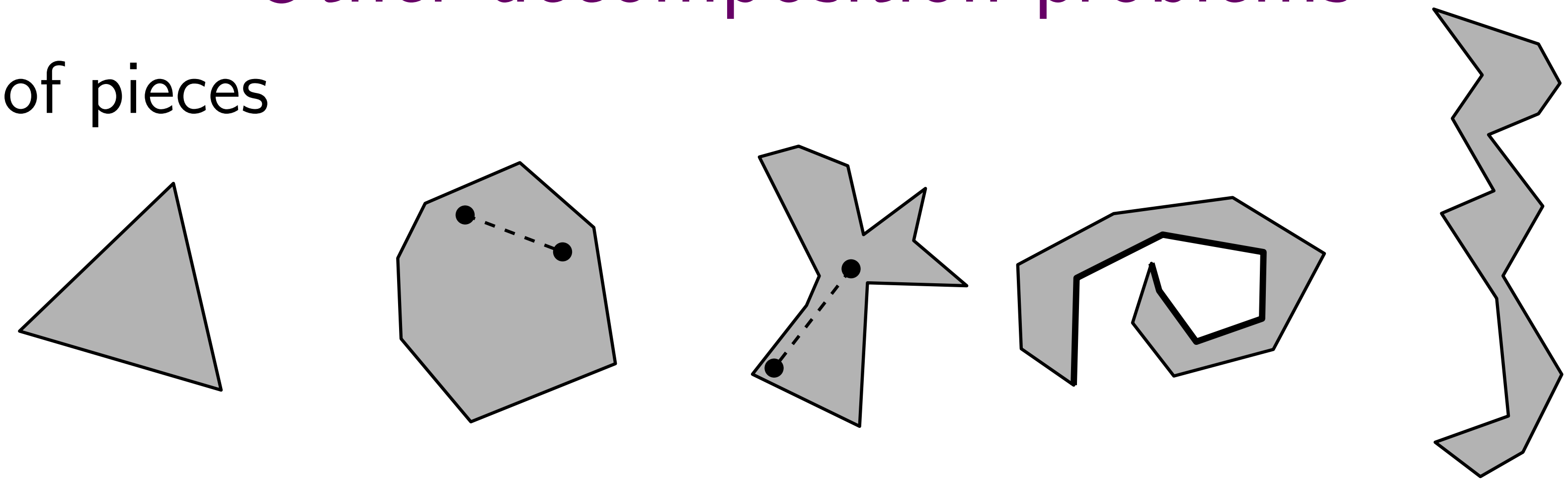
Other decomposition problems

Type of pieces

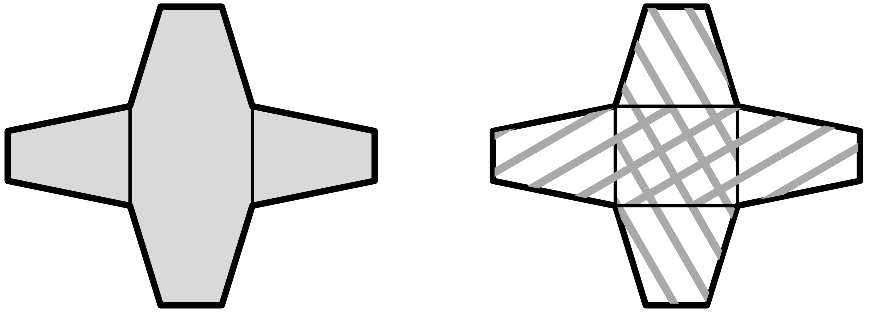


Other decomposition problems

Type of pieces

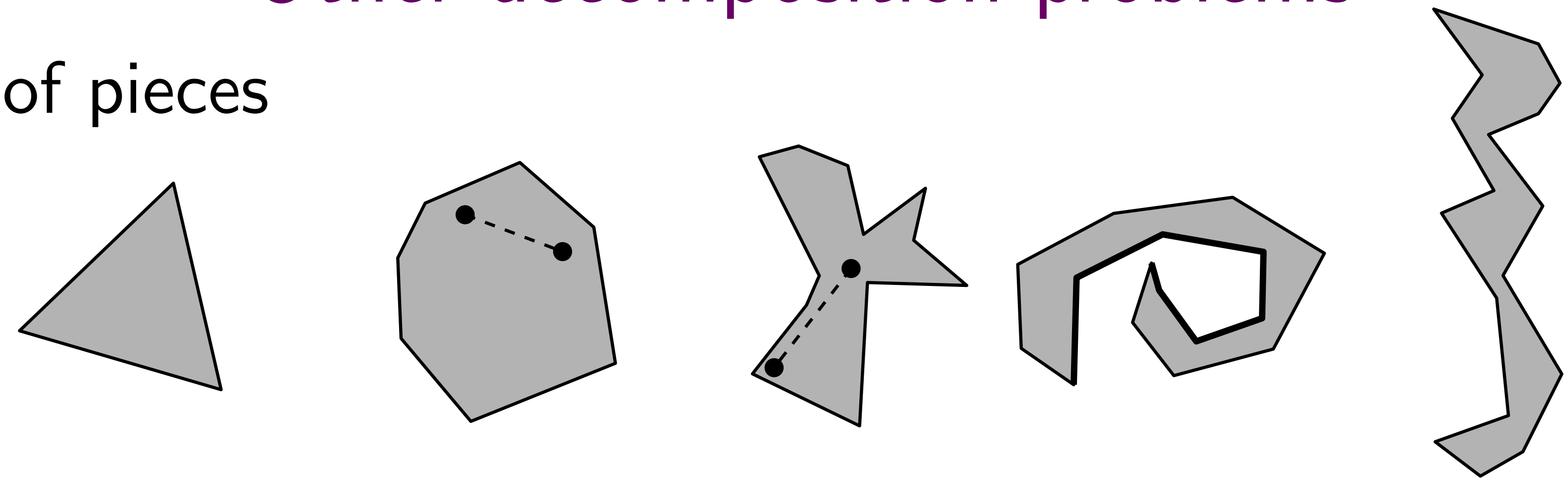


Partition vs. covering

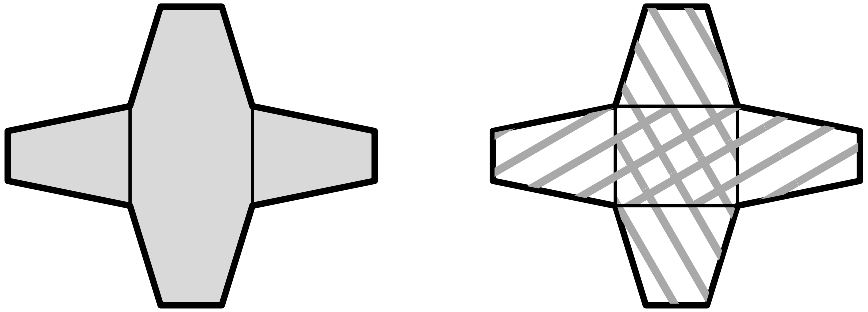


Other decomposition problems

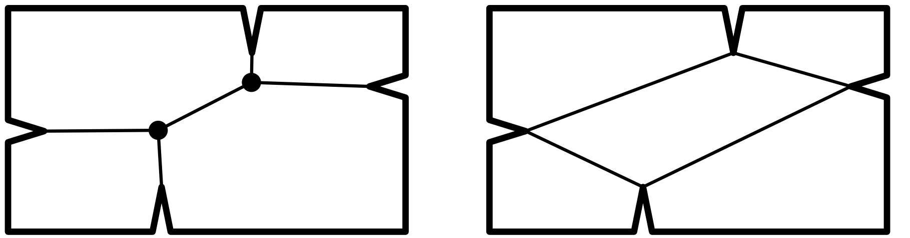
Type of pieces



Partition vs. covering

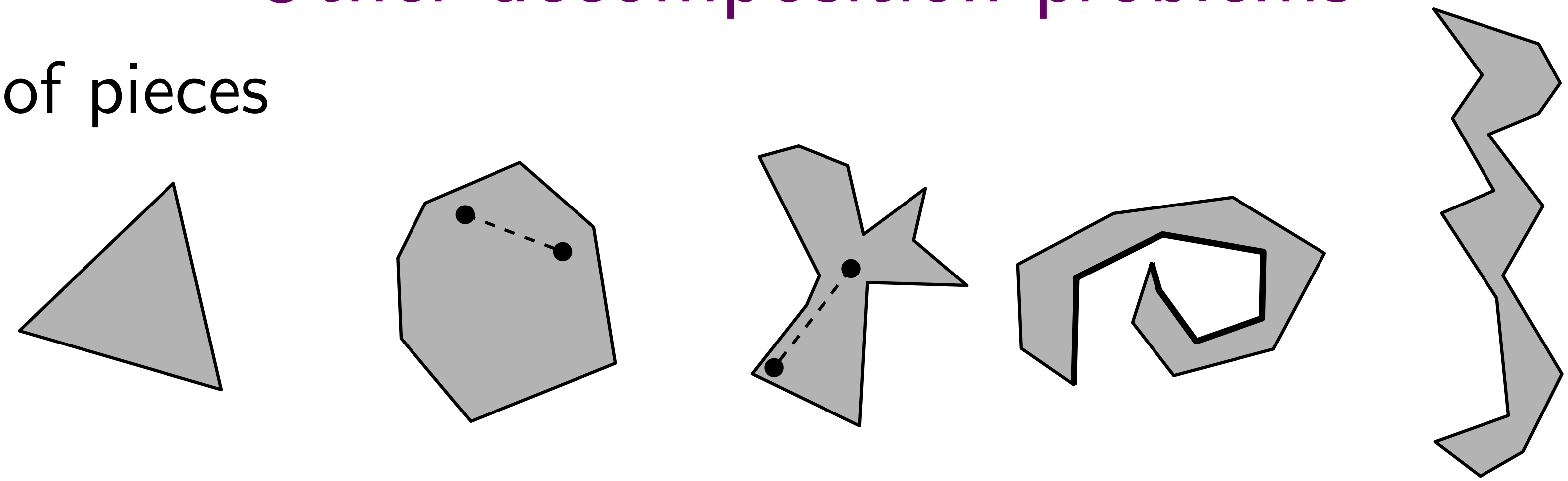


With or without Steiner points

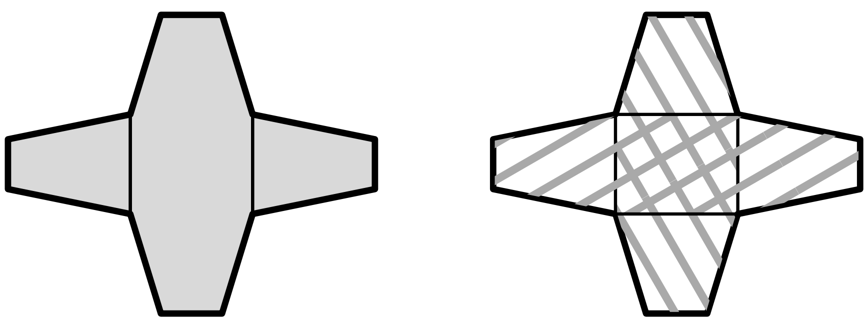


Other decomposition problems

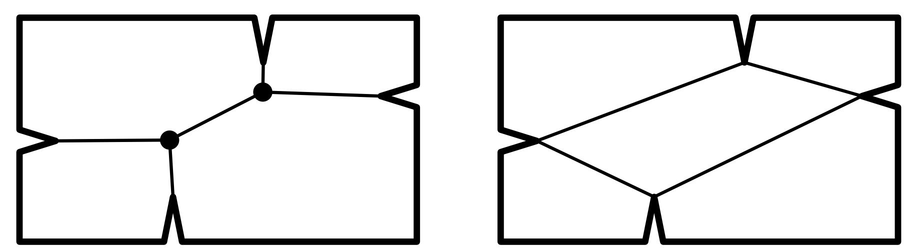
Type of pieces



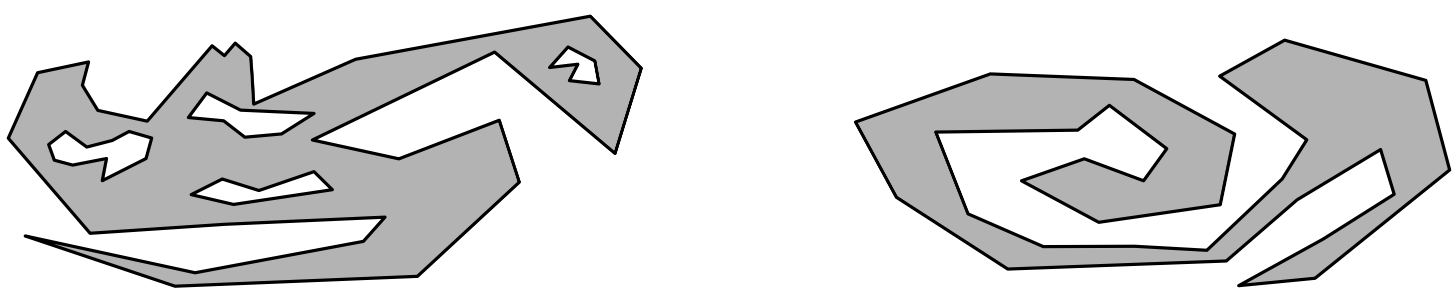
Partition vs. covering



With or without Steiner points

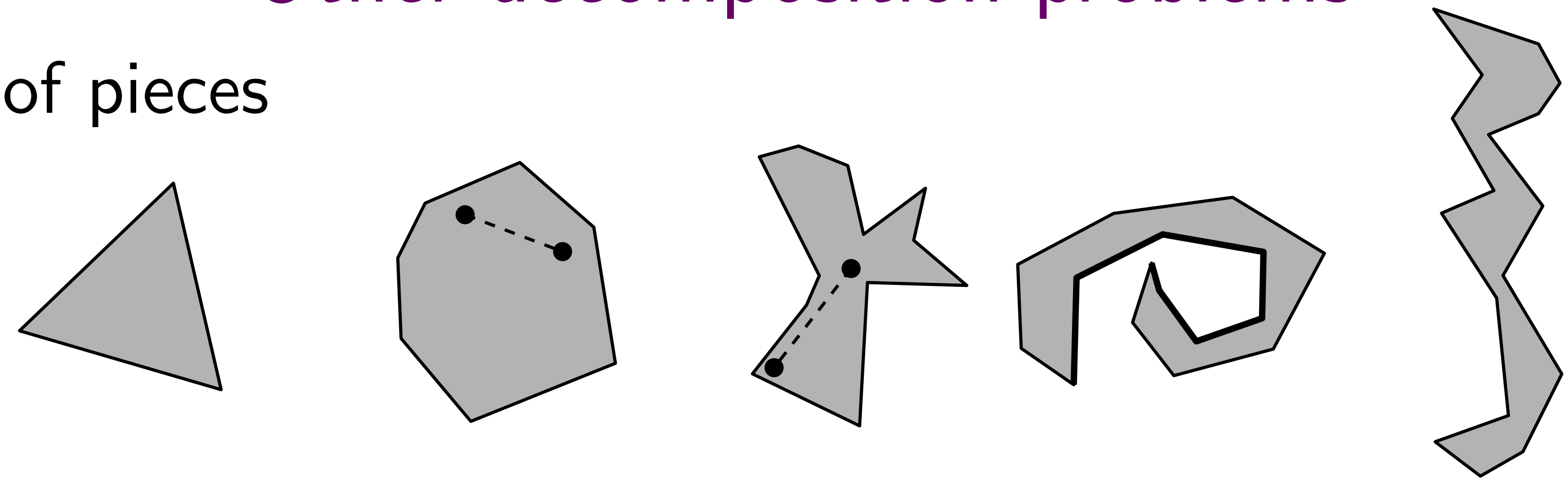


With or without holes

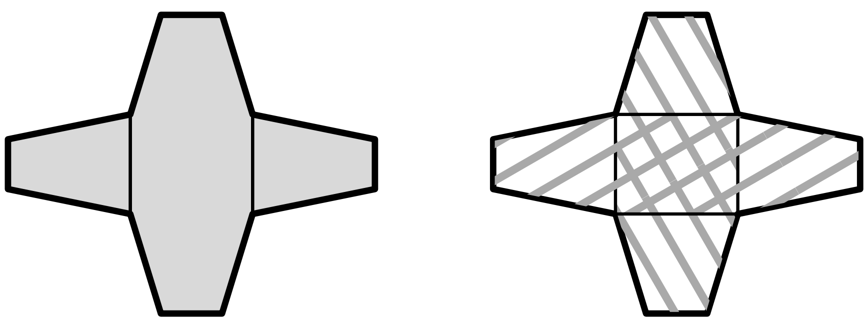


Other decomposition problems

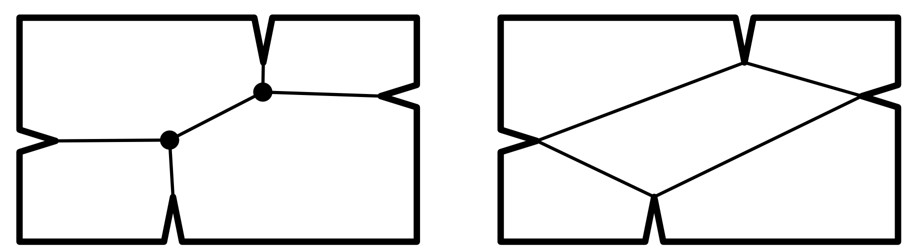
Type of pieces



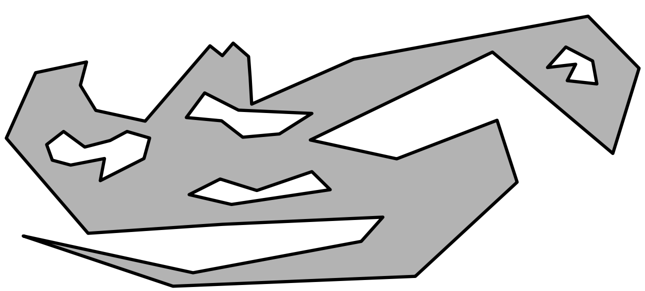
Partition vs. covering



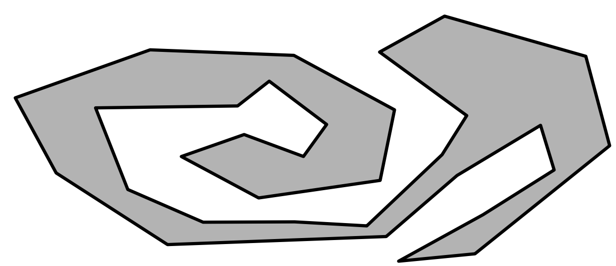
With or without Steiner points



With or without holes

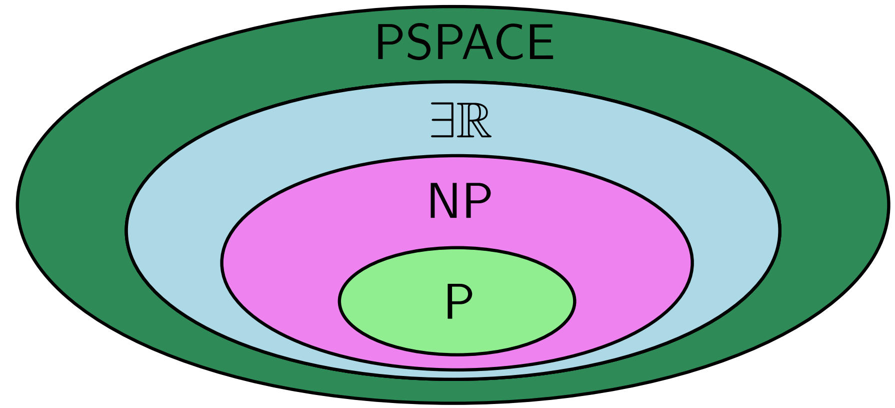


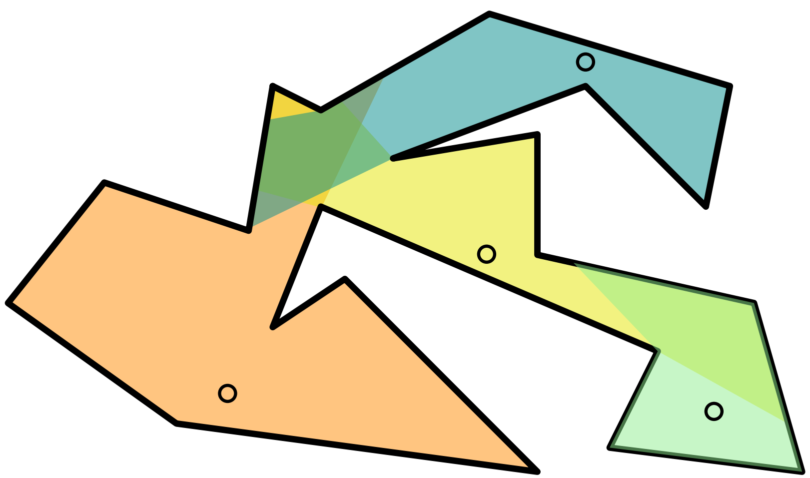
$5 \cdot 2 \cdot 2 \cdot 2 = 40$ problems!

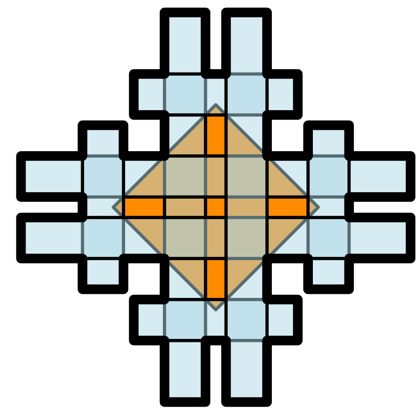


Some of my own work

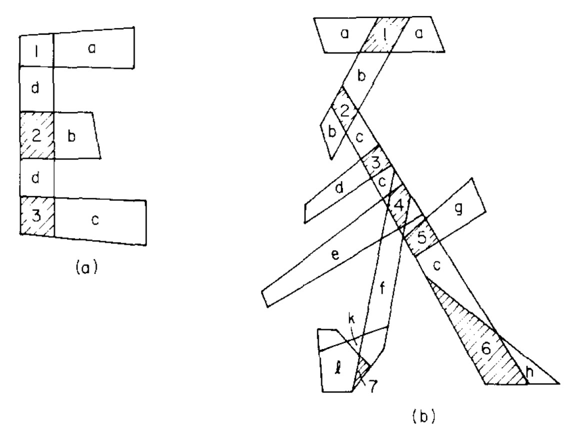
$\exists\mathbb{R}$: Given polynomial $P(x_1, \ldots, x_k)$ with integer coefficients, exists real solution to $P(x_1, \ldots, x_k) = 0$?







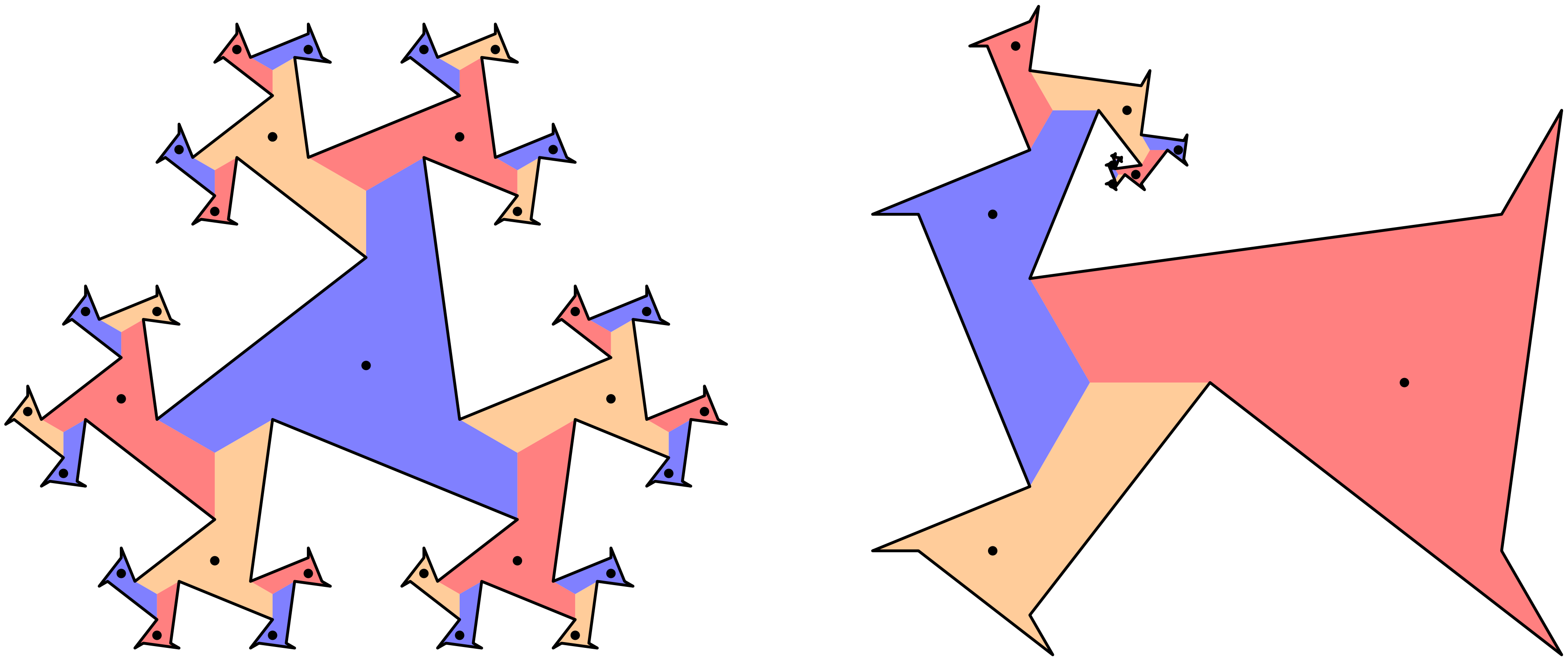
Art Gallery Problem is $\exists\mathbb{R}$-complete
(**A.**, Adamaszek, Miltzow, 2018)



Minimum Convex Cover is $\exists\mathbb{R}$-complete
(**A.**, 2021)

Recent work on decomposition problems

Minimum star partitions in polynomial time ($O(n^{107})$)



A., Blikstad, Nusser, Zhang, 2023