Triangulating a polygon

Computational Geometry

Lecture 4: Triangulating a polygon

对多边形进行三角测量

Polygons and visibility

one of motivation

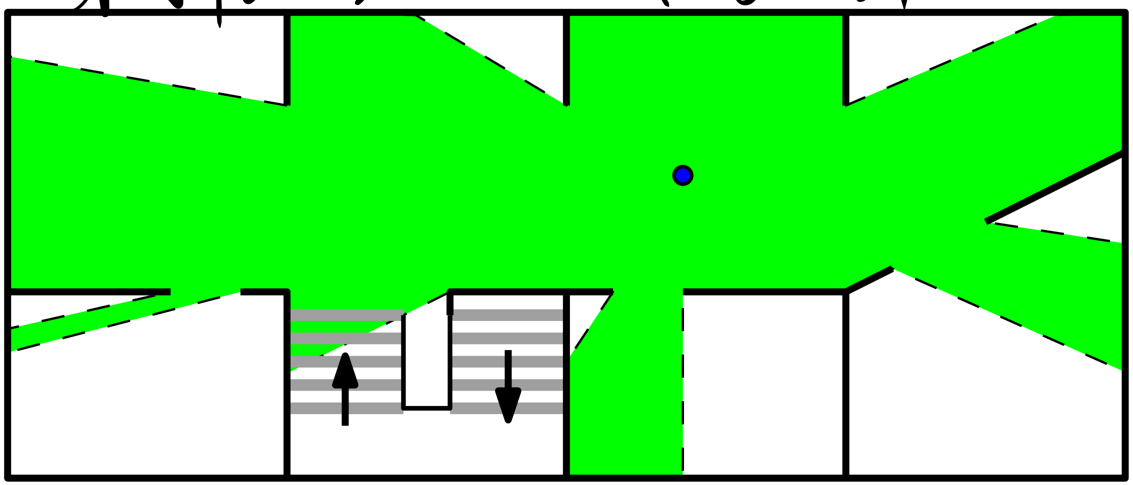
则能看到彼此。

Two points in a simple polygon can see each other if their connecting line segment is in the polygon

如果简单多边形的两点连线在多边形中

Art gallery

放一些摄像头能保护文物不被偷

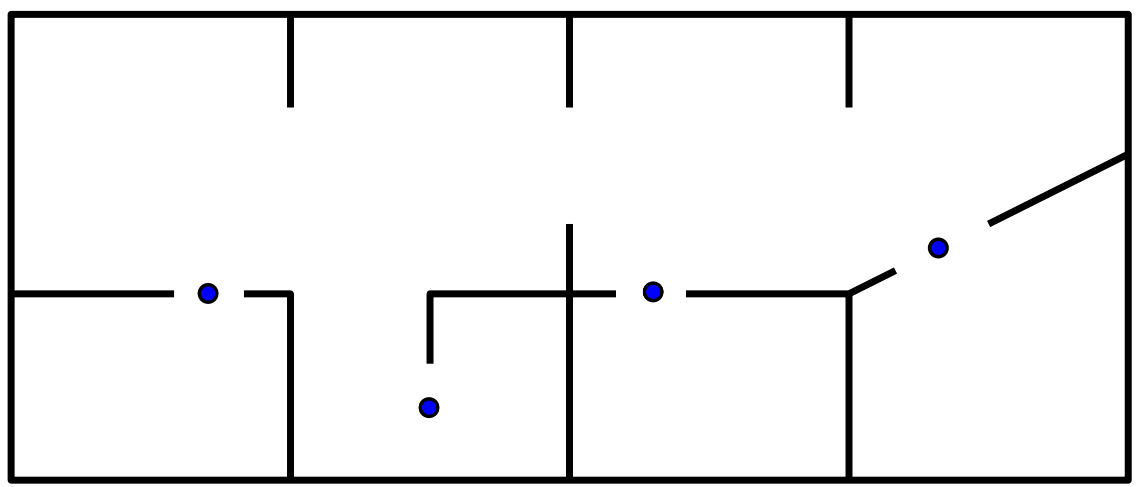


墙会挡住视野范围。 保证可见，→ find a way to place a camera so that everything is protected.

Art gallery problem

本节课 只看2维

Art Gallery Problem: How many cameras are needed to guard a given art gallery so that every point is seen?



把房间想成多边形，不想了录摄像头

简单

Art gallery problem

We don't want the line of visibility to be blocked by a wall. → 看不见

minimize. → NP-hard.

In geometry terminology: How many points are needed in a simple polygon with n vertices so that every point in the polygon is seen?

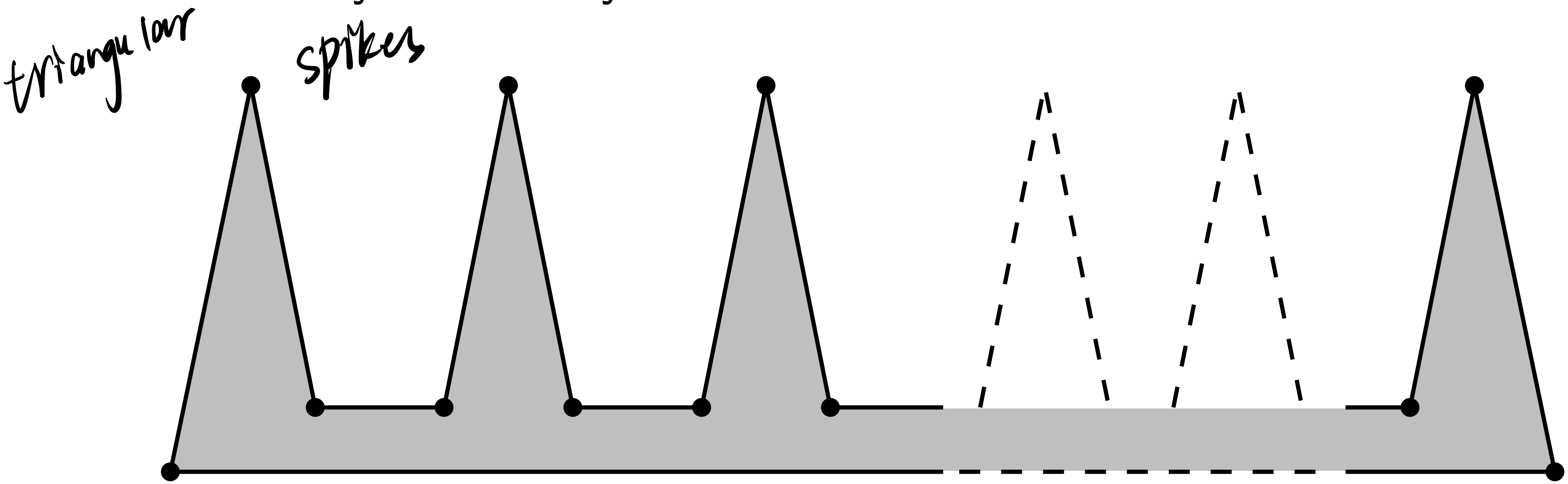
to find the minimum number of these camers or points that can see everything.

The optimization problem is computationally difficult

Art Gallery Theorem: $\lfloor n/3 \rfloor$ cameras are occasionally necessary but always sufficient

Art gallery problem

Art Gallery Theorem: $\lfloor n/3 \rfloor$ cameras are occasionally necessary but always sufficient



need one guard for each of these spikes, and each spike adds 3 corners to the polygon $\Rightarrow$ need at least $\frac{n}{3}$ guards.

Triangulation, diagonal

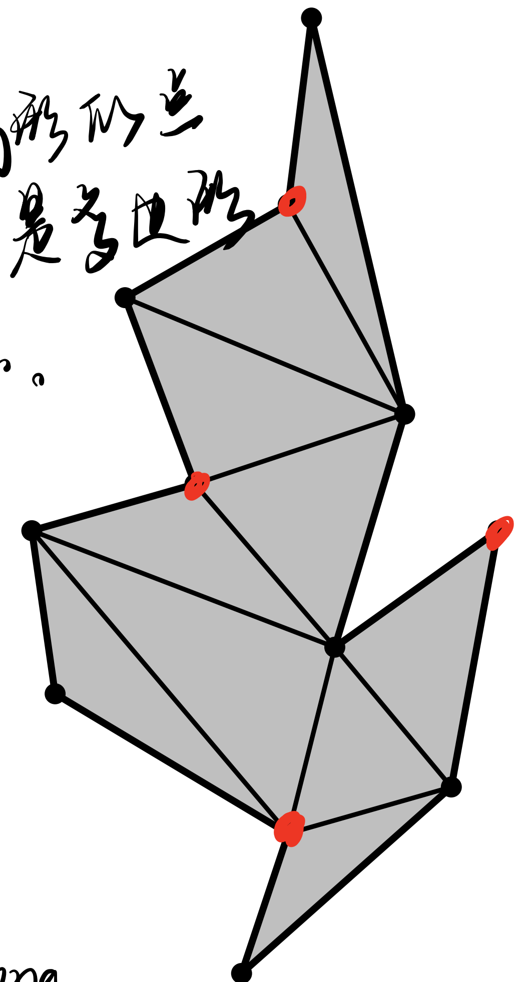
Why are $\lfloor n/3 \rfloor$ cameras always enough?

partitioning

Assume polygon P is triangulated: a decomposition of P into disjoint triangles by a maximal set of → 分解 non-intersecting diagonals

Diagonal of P: open line segment that connects two vertices of P and fully lies in the interior of P

三角形的点同样是多边形的点。



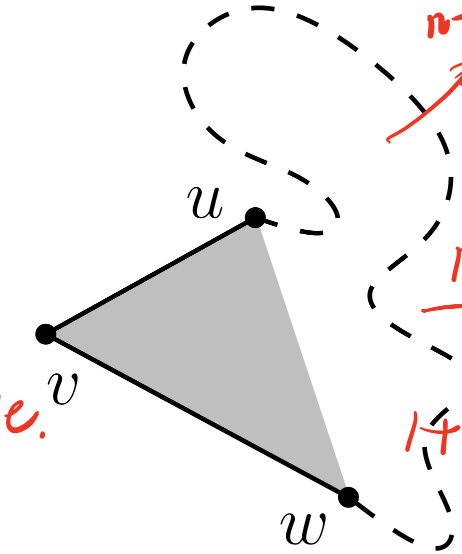
We have cut the polygon in pieces along diagonals.

A triangulation always exists

Lemma: A simple polygon with n vertices can always be triangulated, and always with $n-2$ triangles

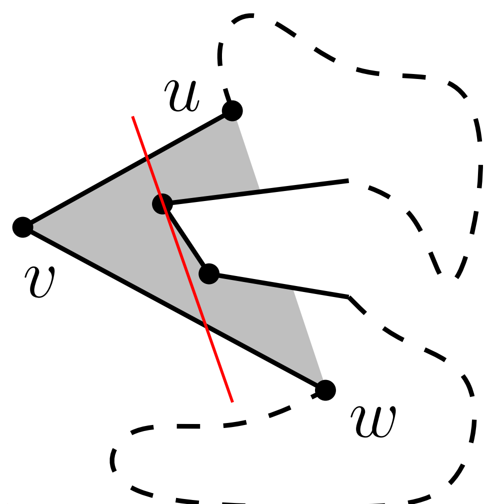
Proof: Induction on n. If $n=3$, it is trivial → Already triangulated, n-2-1 个 triangle.

Assume $n>3$. Consider the leftmost vertex v and its two neighbors u and w. Either uw is a diagonal (case 1), or part of the boundary of P is in $\triangle uvw$ (case 2)



n-1 corners or vertices
推导
n-3个△
1+n-3=n-2个△

Case 2: choose the vertex t in $\triangle uvw$ farthest from the line through u and w, then $\overline{vt}$ must be a diagonal



We can use this diagonal to split the polygon into 2 pieces. VT↑, VT↓

A triangulation always exists

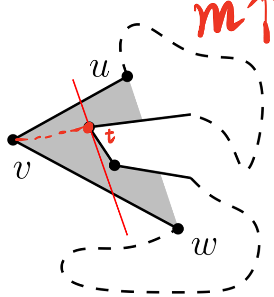
因为 w不可能在上部分，而u也不可能在下方(vt)，所以 ⇐ 它们中的每一个都有少于n个 vertices？
至多 n-1 个

In case 1, $\overline{uw}$ cuts the polygon into a triangle and a simple polygon with $n-1$ vertices, and we apply induction

In case 2, $\overline{vt}$ cuts the polygon into two simple polygons with m and $n-m+2$ vertices, $3 \leq m \leq n-1$, and we also apply induction

By induction, the two polygons can be triangulated using $m-2$ and $n-m+2-2 = n-m$ triangles. So the original polygon is triangulated using $m-2 + n-m = n-2$ triangles □

m个



1个v, 1个u

$n-(m-2) \Rightarrow n-m+2$

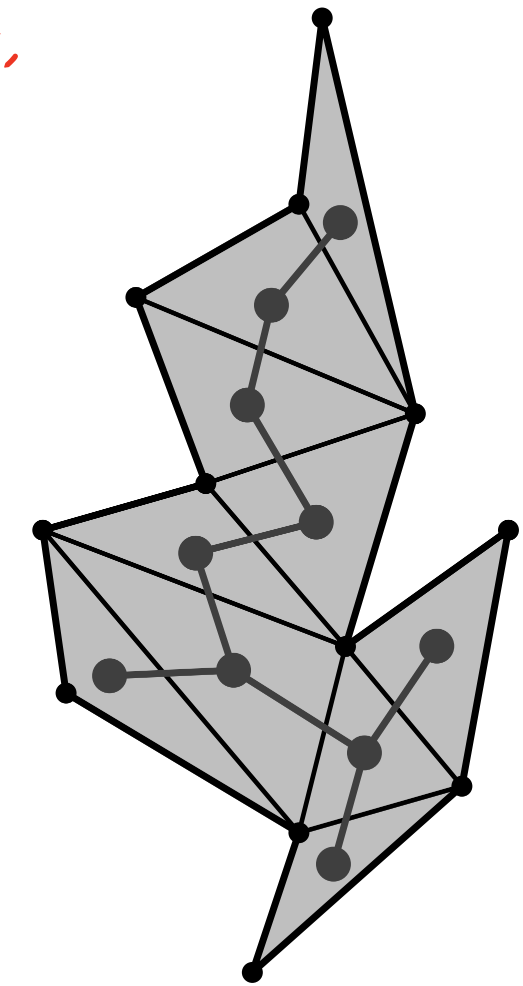
A 3-coloring always exists

We can always triangulate a polygon.

如果你在房间里，你总能去一些角落并且从那个角落，你能看见另一个角落。

Observe: the dual graph of a triangulated simple polygon is a tree

Dual graph: each face gives a node; two nodes are connected if the faces are adjacent



A 3-coloring always exists

Lemma: The vertices of a triangulated simple polygon can always be 3-colored

可以通过 induction 推导

Proof: Induction on the number of triangles in the triangulation. Base case: True for a triangle

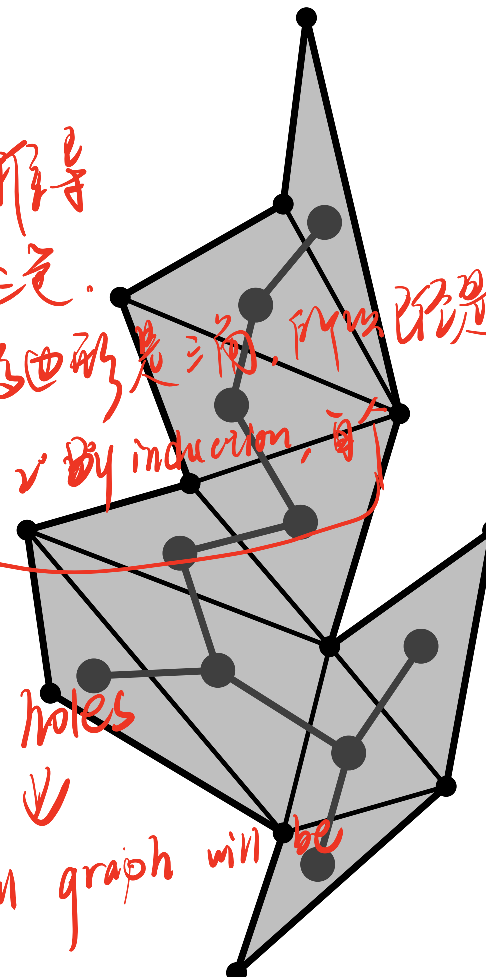
每个顶点都有RGB三选.

1. 如果多边形是三角, 所以只提个三角

2. By induction, 每个

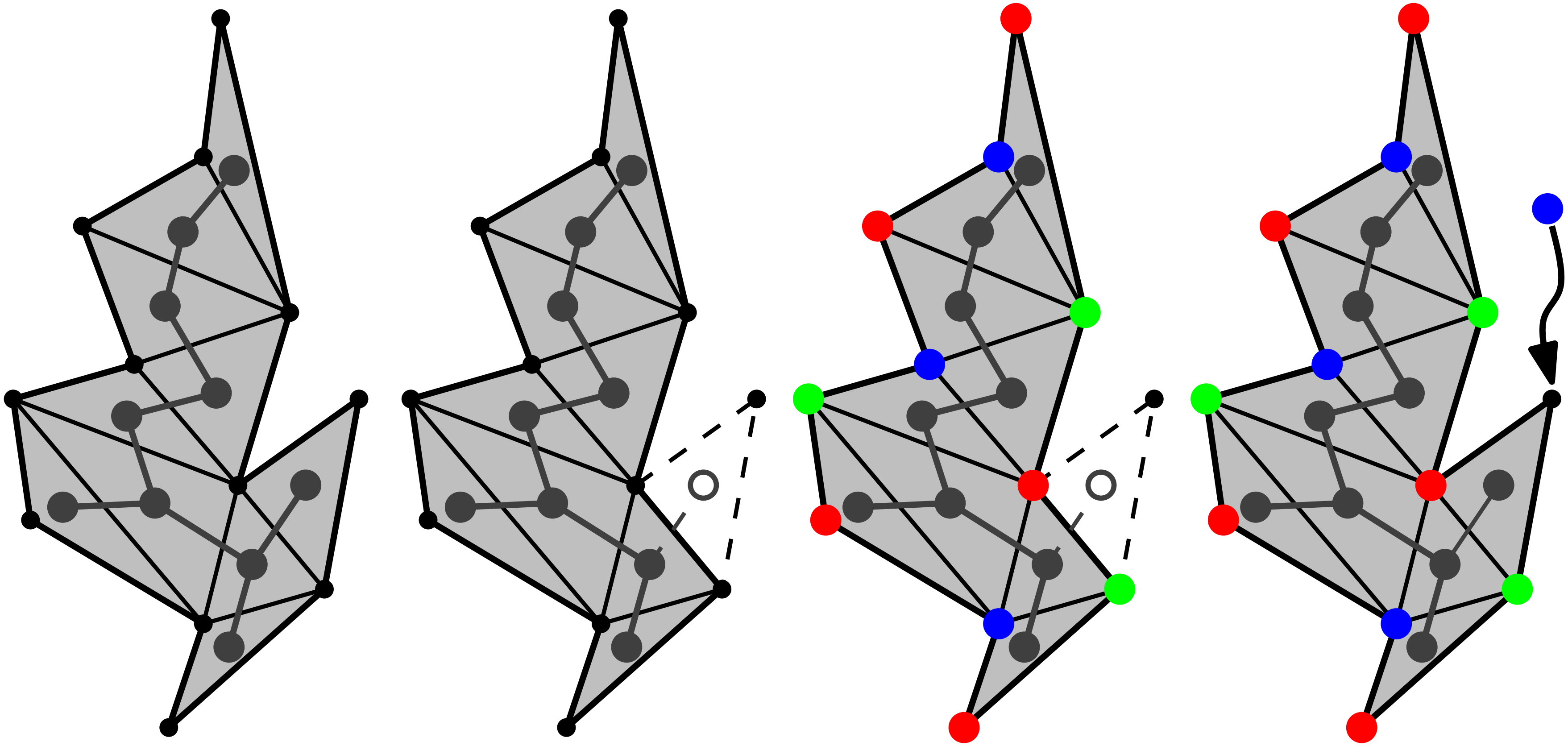
是证其

Every tree has a leaf, in particular the one that is the dual graph. Remove the corresponding triangle from the triangulated polygon, color its vertices, add the triangle back, and let the extra vertex have the color different from its neighbors



No holes ⇓ dual graph will be a tree

A 3-coloring always exists



$\lfloor n/3 \rfloor$ cameras are enough

For a 3-colored, triangulated simple polygon, one of the color classes is used by at most $\lfloor n/3 \rfloor$ colors. Place the cameras at these vertices

we can regard the art gallery problem within n/3 cameras

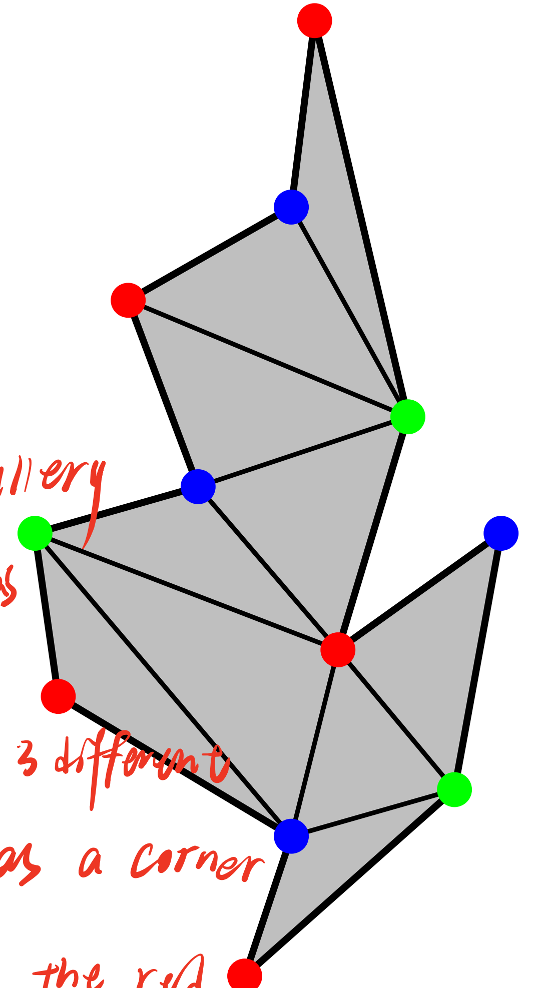
⇓

因为 we have vertices of 3 different colours and then each triangle has a corner of each color ⇒ if we take all the red vertices, then will guard everything as each triangle will be guarded by some red point.

This argument is called
the pigeon-hole principle

$\min\{|R|, |G|, |B|\}$

integers "$\lfloor\ \rfloor$"



$\lfloor n/3 \rfloor$ cameras are enough

Question: Why does the proof fail when the polygon has holes?

When we do the inductive step, we say that we pick a leaf from the tree induced by the triangulation, but if there is a hole in the polygon, then the triangulation will not induce a tree then there can be circles and then there is no leaf we can choose.
then you need more guards.

Two-ears for triangulation

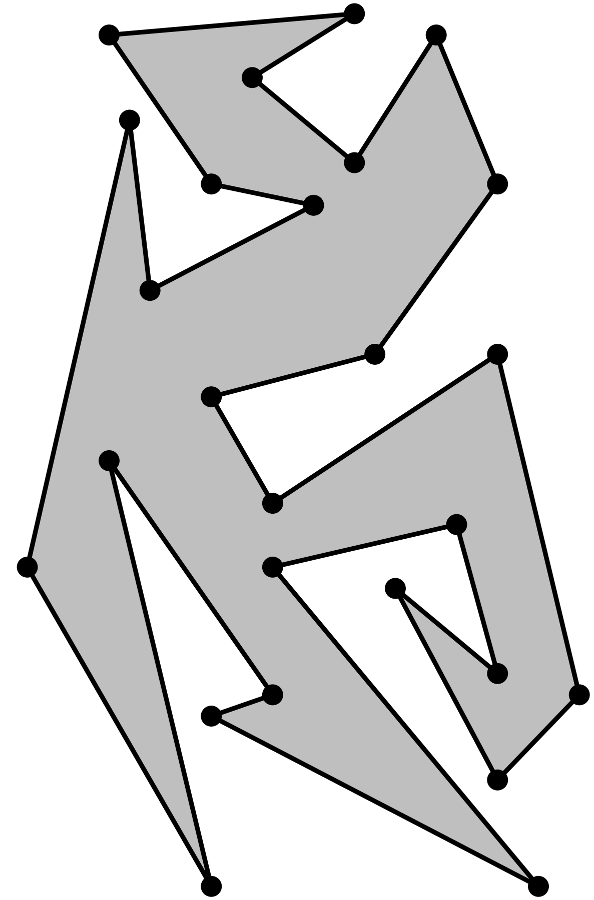
一种得到 ↑ 的方法.

Using the two-ears theorem:
(an ear consists of three consecutive vertices u, v, w where $\overline{uw}$ is a diagonal)

Find an ear, cut it off with a diagonal, triangulate the rest iteratively

Question: Why does every simple polygon have an ear?

Question: How efficient is this algorithm?



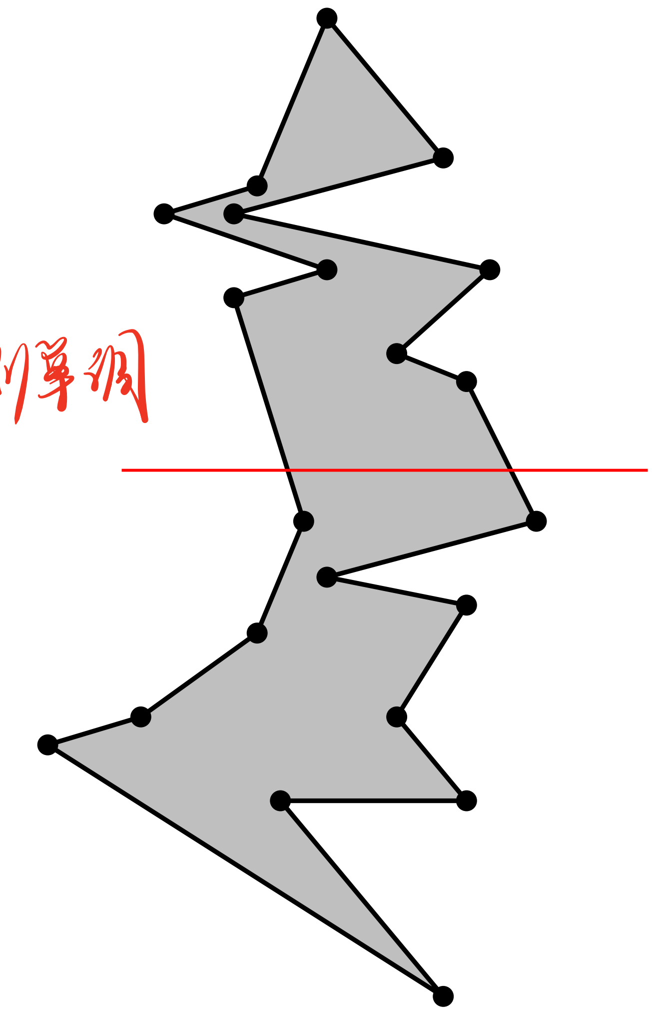
Overview

单调的

A simple polygon is y-monotone iff any horizontal line intersects it in a connected set (or not at all) ⇒ 比如一条线相交了2个交点则单调

Use plane sweep to partition the polygon into y-monotone polygons

Then triangulate each y-monotone polygon

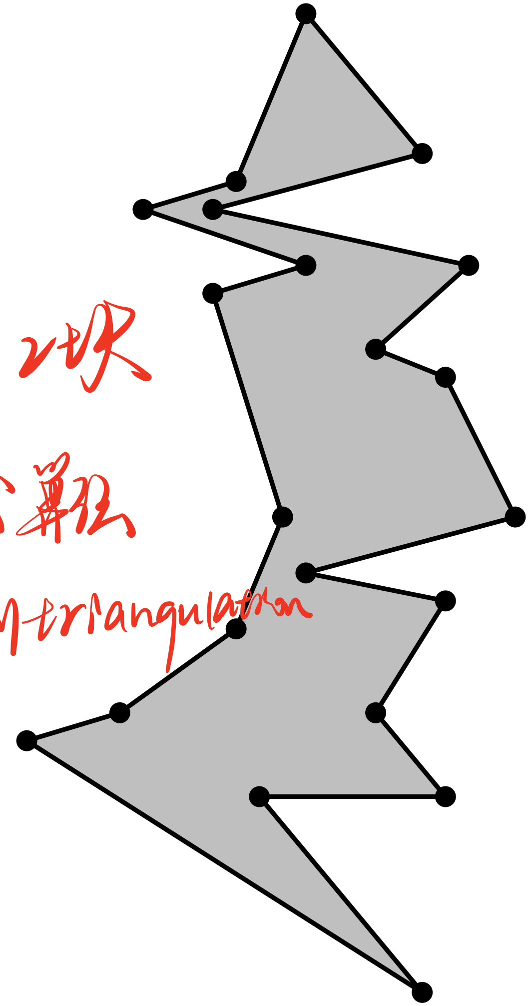


Monotone polygons

A y-monotone polygon has a top vertex, a bottom vertex, and two y-monotone chains between top and bottom as its boundary

分成2块
有多一个单调
能得到triangulation

Any simple polygon with one top vertex and one bottom vertex is y-monotone

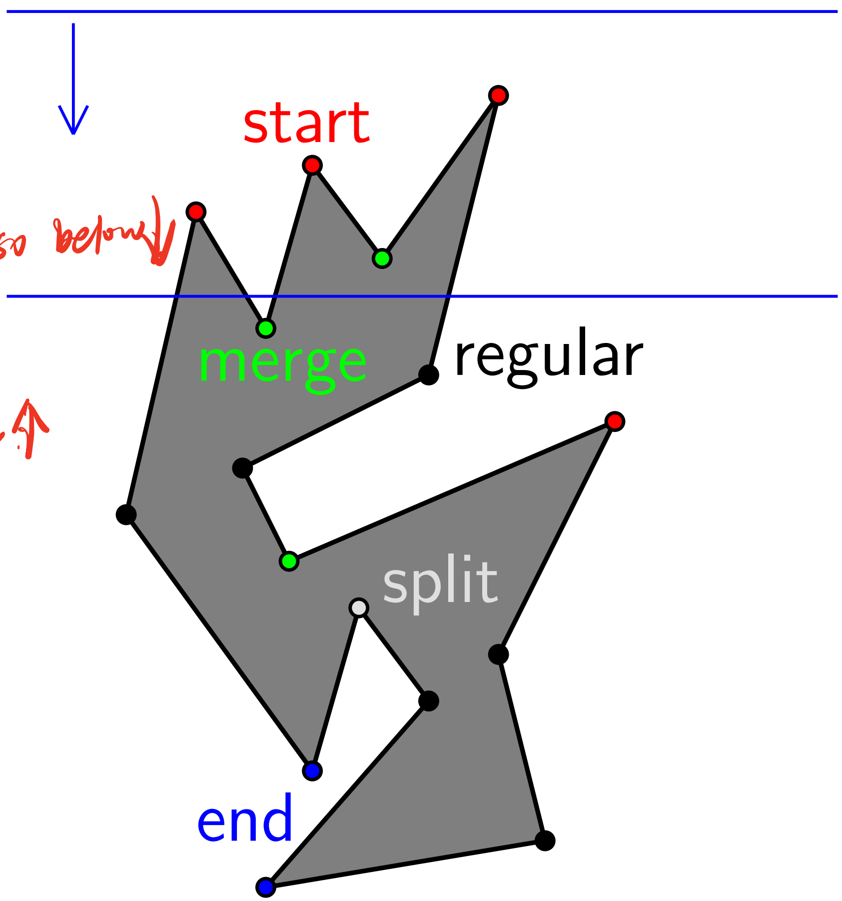


Vertex types

What types of vertices does a simple polygon have?

- ↓↓ start → go down↓ and the polygon is also below↓
- ↑↑ stop → edges go up↑, polygon ↑
- ↓↑ split → edges go down↓. the polygon above↑
- ↑↓ merge → edges go up, polygon ↓
- ↑↓ or ↓↑ regular → one ↑, one ↓

... imagining a sweep line going top to bottom

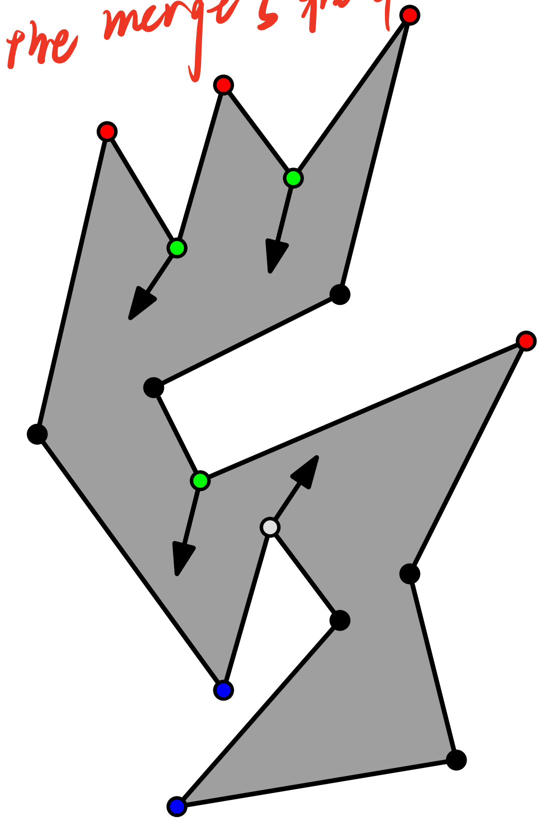


Sweep ideas

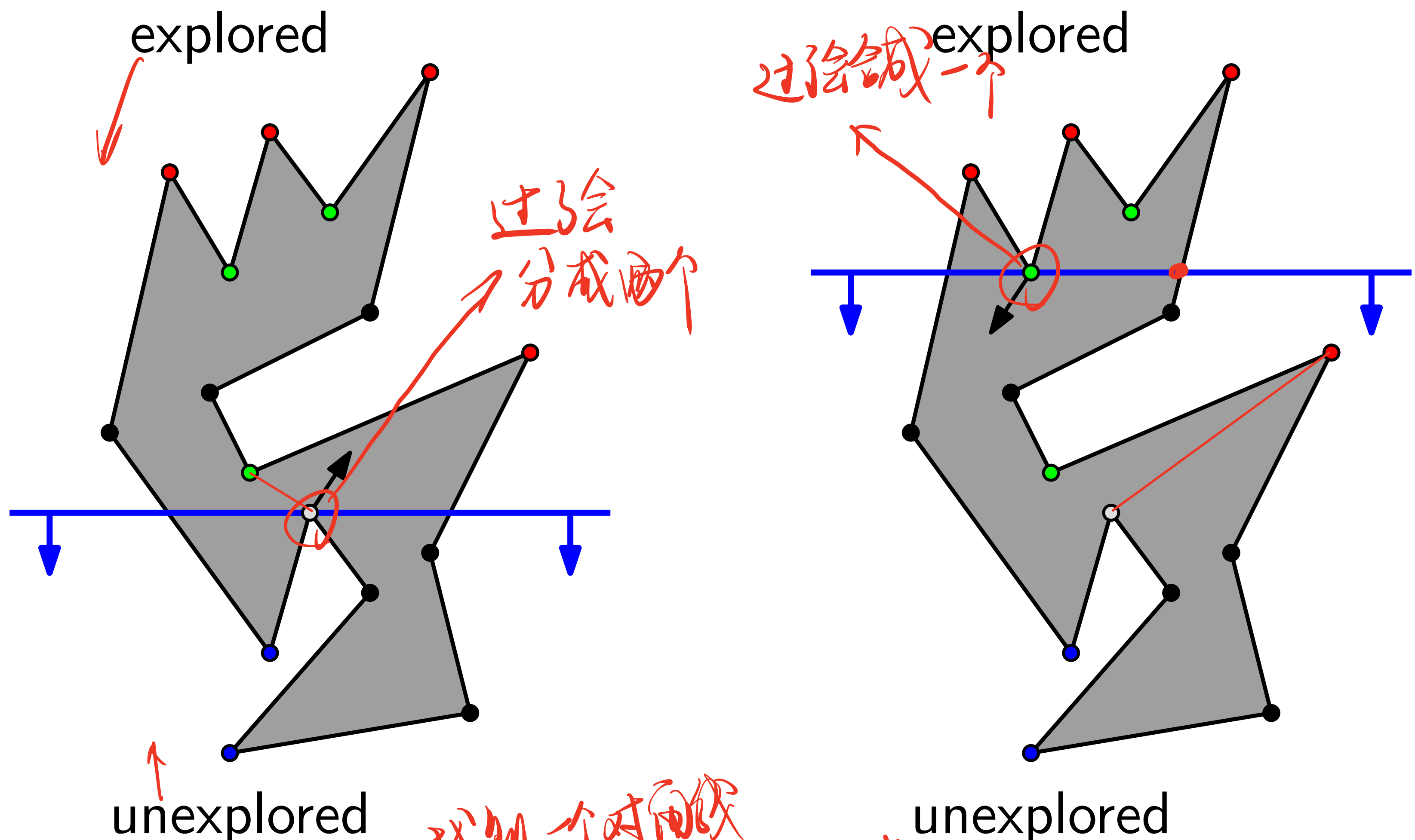
为了得到 y-monotone pieces,
我们得消除 all the merges 和 split.

Find diagonals from each merge vertex down, and from each split vertex up

A simple polygon with no split or merge vertices can have at most one start and one end vertex, so it is y-monotone



Sweep ideas



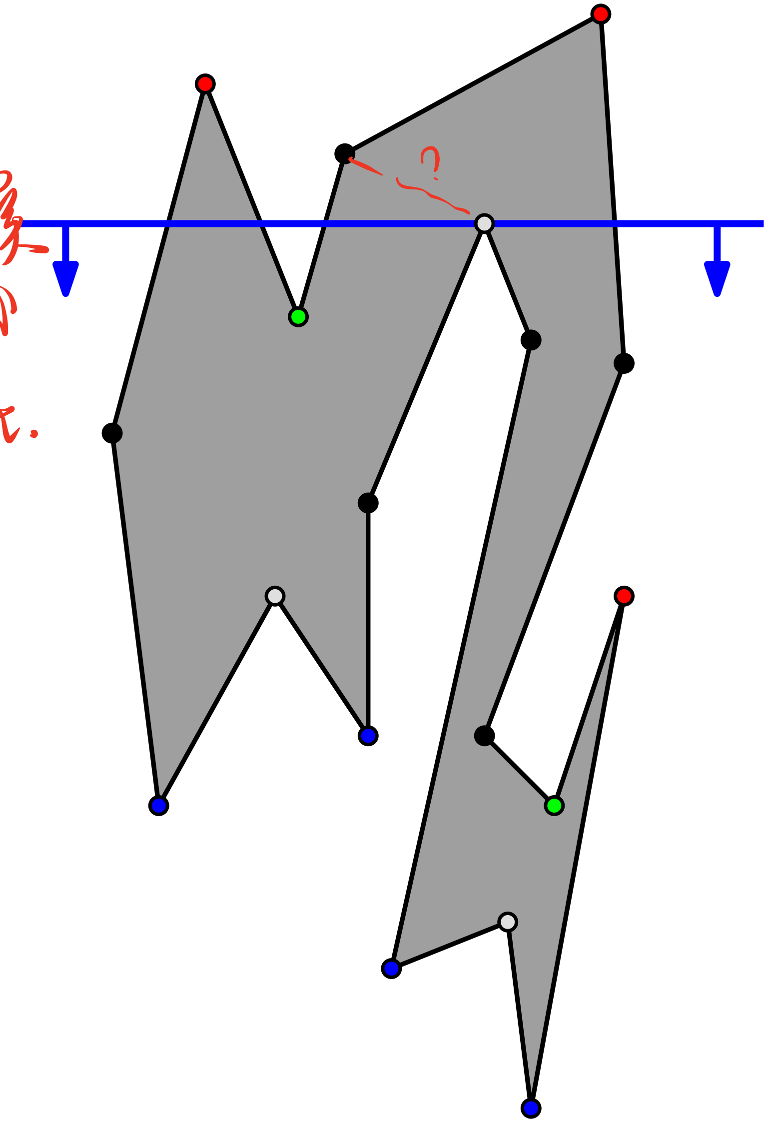
找到一个对角线
消除split,成1个无split
的多边形

Sweep ideas

从上往下扫 split
从下往上扫 merge
因为反方向观察 merge 就是 split.

Where can a diagonal from a split vertex go?

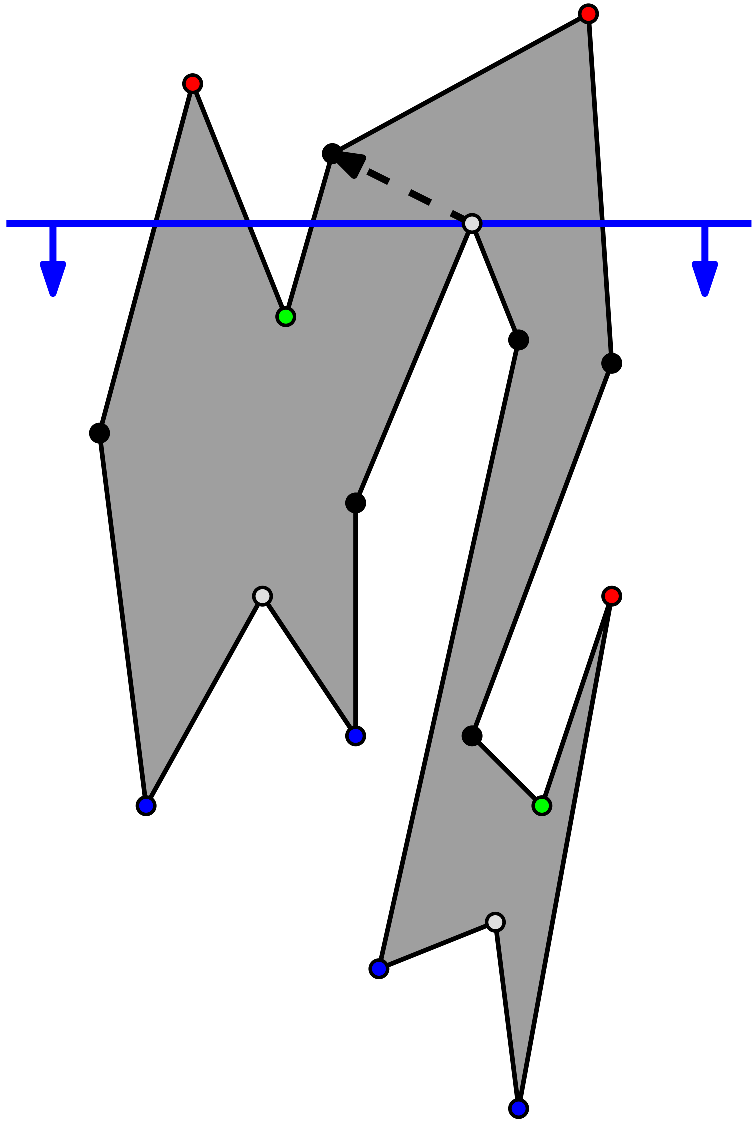
Perhaps the upper endpoint of the edge immediately left of the split vertex?



Sweep ideas

Where can a diagonal from a split vertex go?

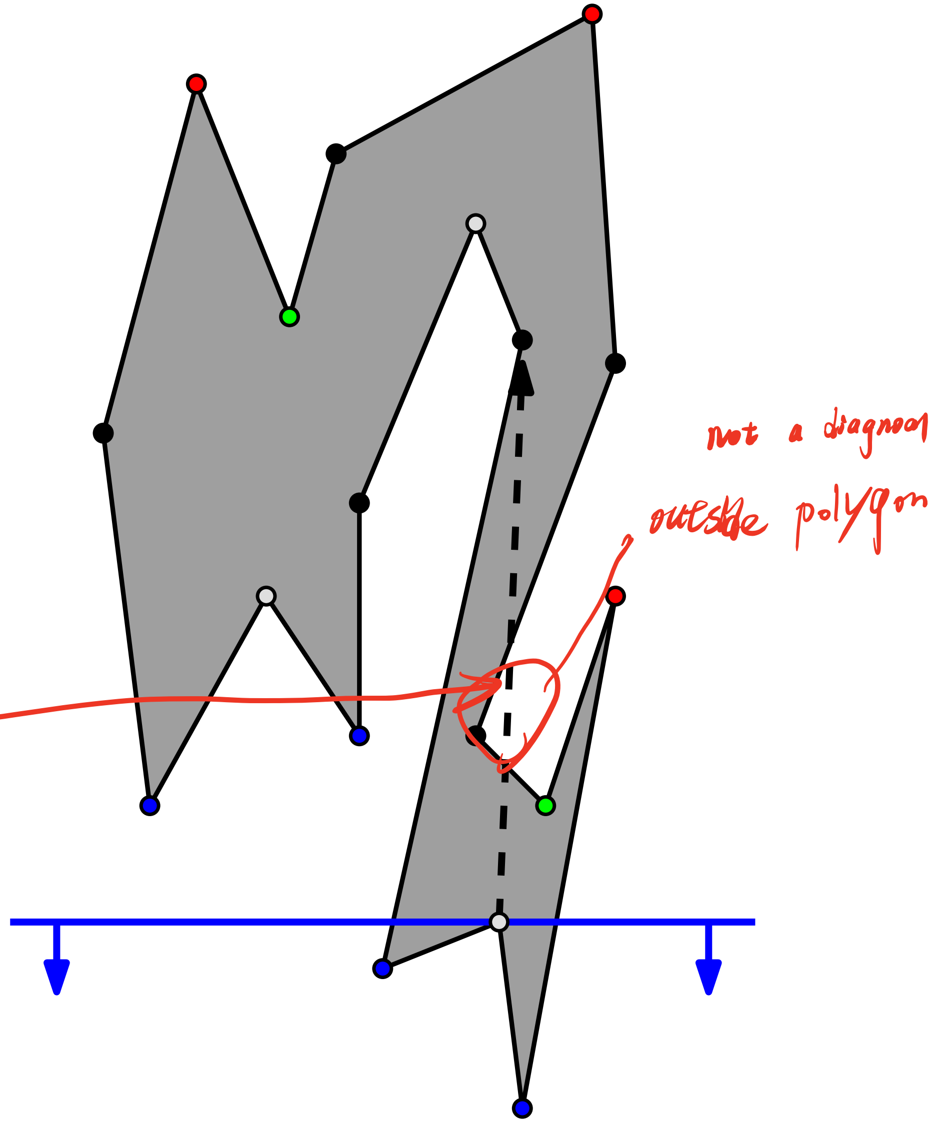
Perhaps the upper endpoint of the edge immediately left of the split vertex?



Sweep ideas

Where can a diagonal from a split vertex go?

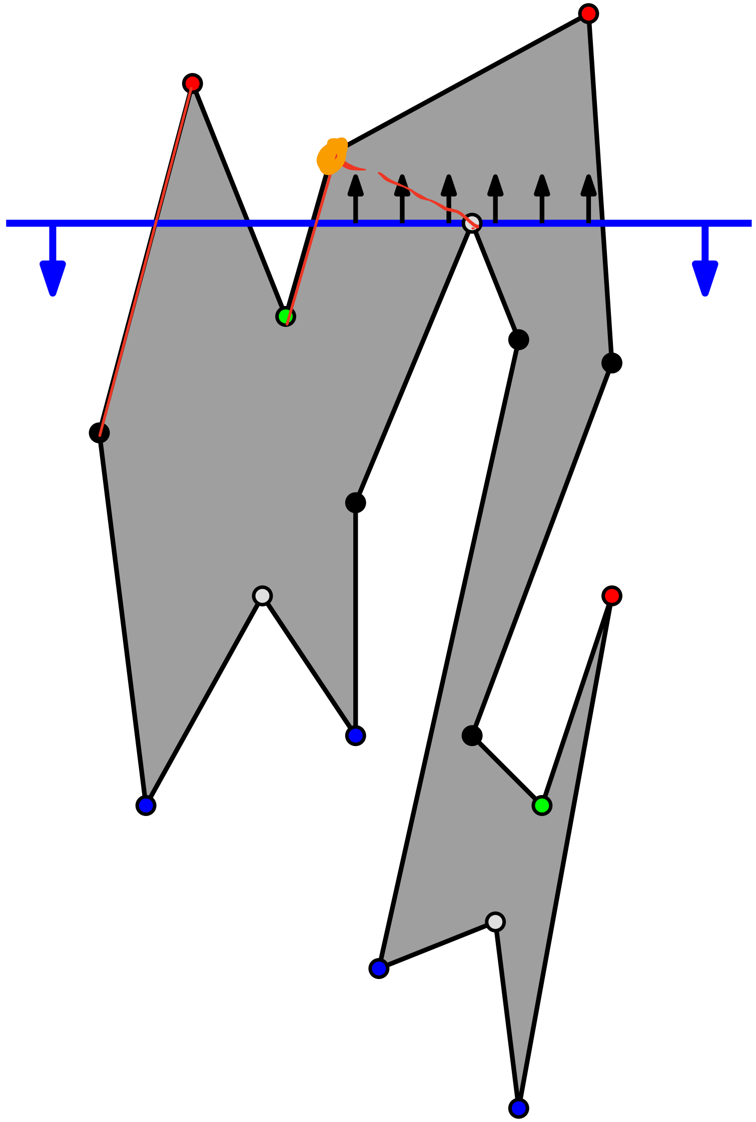
Perhaps the upper endpoint of the edge immediately left of the split vertex?



Sweep ideas

Where can a diagonal from a split vertex go?

Perhaps the last vertex passed in the same “component”?



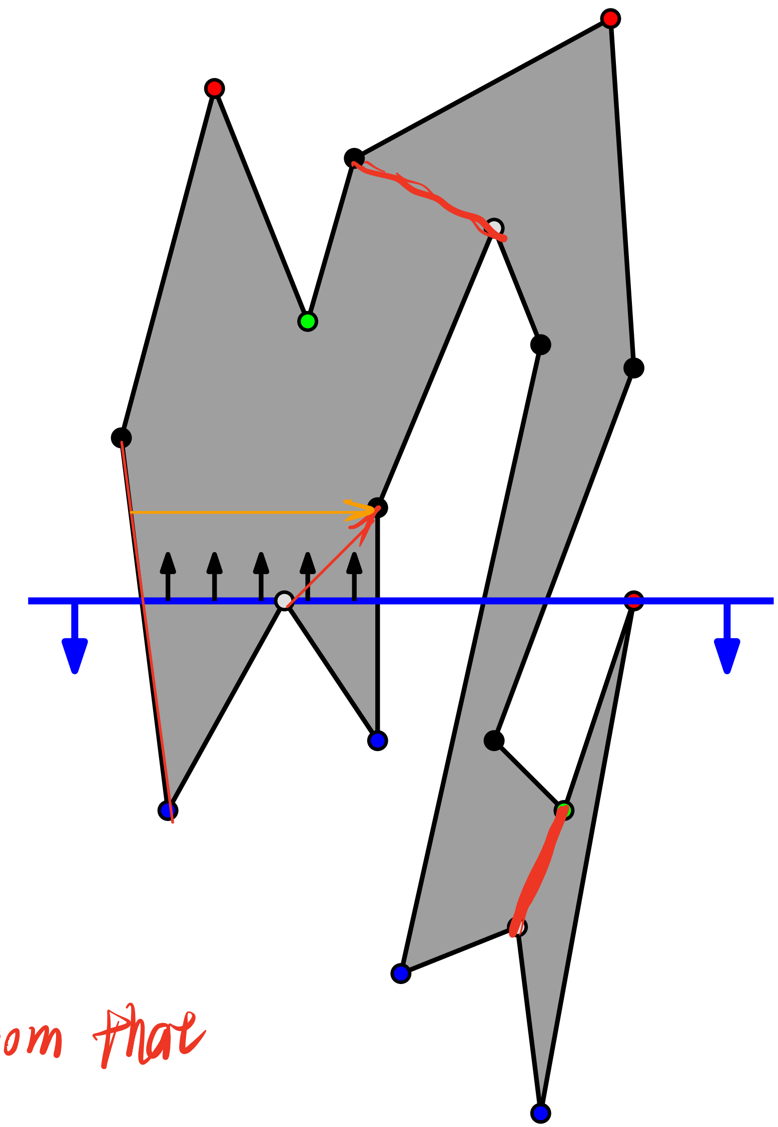
Sweep ideas

看到

Where can a diagonal from a split vertex go?

Perhaps the last vertex passed in the same "component"?

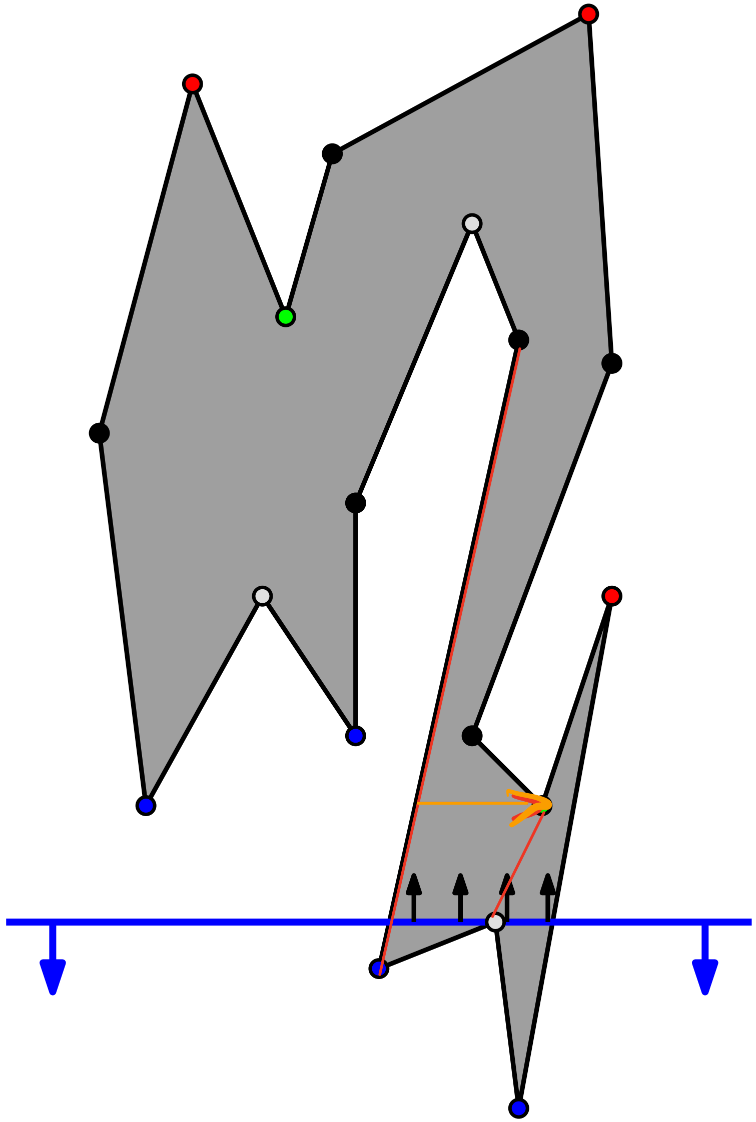
The first vertex we need when we look up from that



Sweep ideas

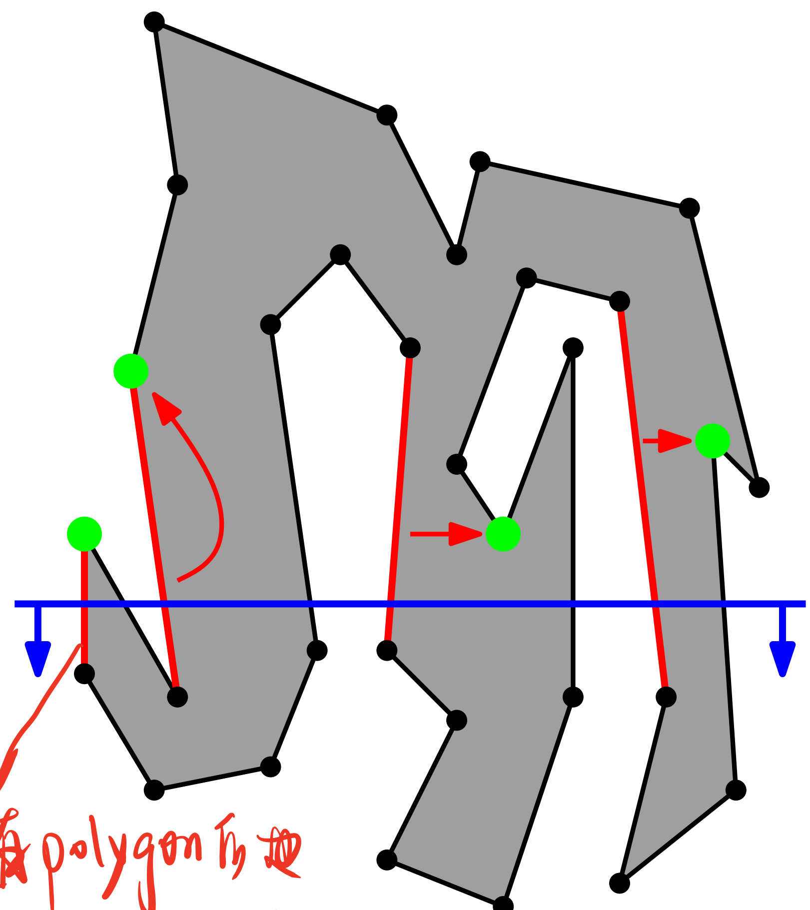
Where can a diagonal from a split vertex go?

Perhaps the last vertex passed in the same “component”?



Helpers of edges

The helper for an edge e that has the polygon right of it, and a position of the sweep line, is the lowest vertex v above the sweep line such that the horizontal line segment connecting e and v is inside the polygon



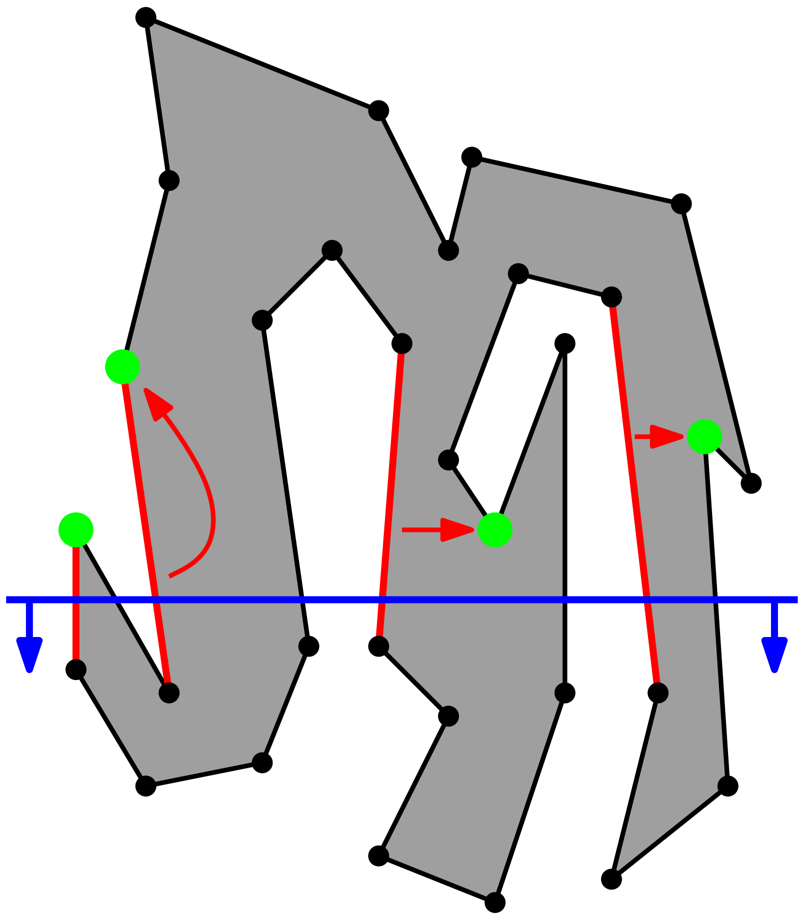
右边有polygon的边

helper of these edges ← 绿点是 这条边能看到的最低点

Status of sweep

The status is the set of edges intersecting the sweep line that have the polygon to their right, sorted from left to right, and each with their *helper*: the last vertex passed in that component

r+g
edge+ vertex



we need to update this structure as we go along.

Status structure, event list

当我们遇到split时，连接status中的那个helper.

The status structure stores all edges that have the polygon to the right, with their helper, sorted from left to right in the leaves of a balanced binary search tree

事件仅发生在顶点

The events happen only at the vertices: sort them by y-coordinate and put them in a list (or array, or tree)

按y坐标排序并放入列表里。

存成一个平衡二叉搜索树.

Main algorithm

Initialize the event list (all vertices sorted by decreasing y-coordinate) and the status structure (empty)

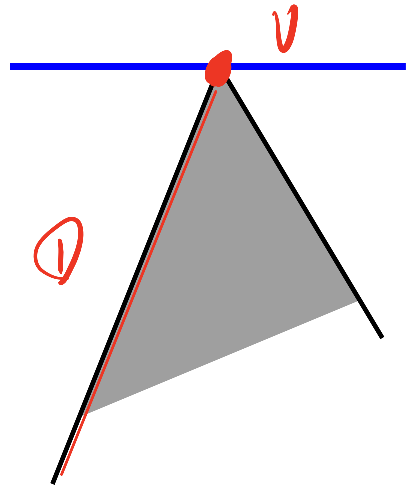
While there are still events in the event list, remove the first (topmost) one and handle it

Event handling

当迭到这个点，将这个边和点塞进去

Start vertex v:

- Insert the counterclockwise incident edge in T with v as the helper



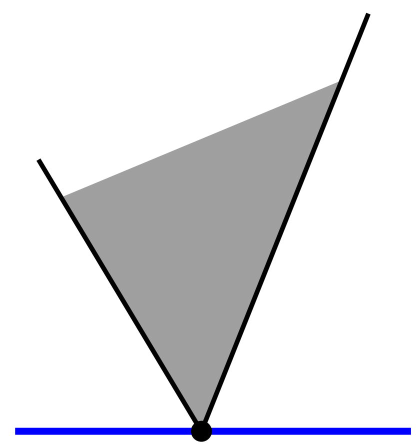
塞v是因为这个点是最低的 能看见这个边的

Status $\{①, v\}$

Event handling

End vertex v:

- Delete the clockwise incident edge and its helper from T



过了最低点，

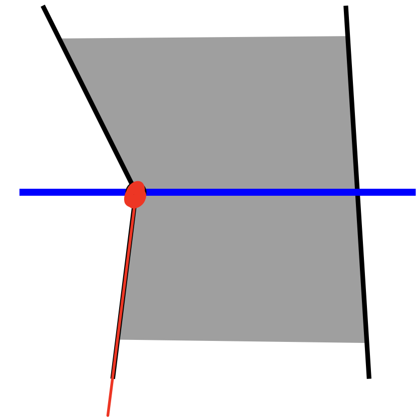
删掉这个边和它的 helper，

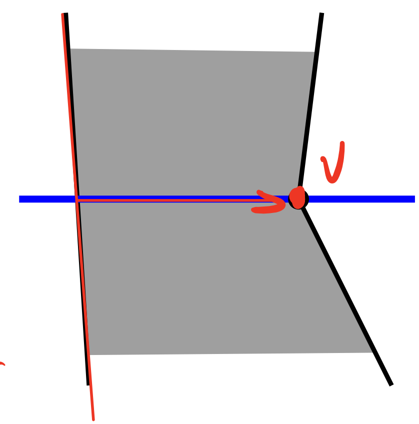
Event handling

Regular vertex v:

- If the polygon is right of the two incident edges, then replace the upper edge by the lower edge in T, and make v the helper
- If the polygon is left of the two incident edges, then find the edge e directly left of v, and replace its helper by v

更新它
helper

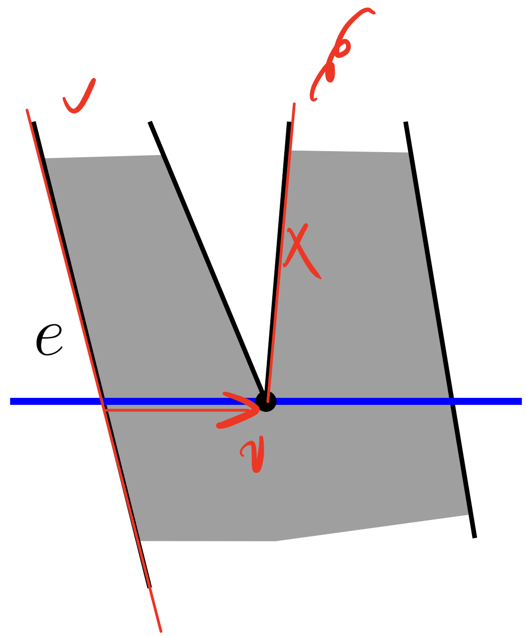




Event handling

Merge vertex v:

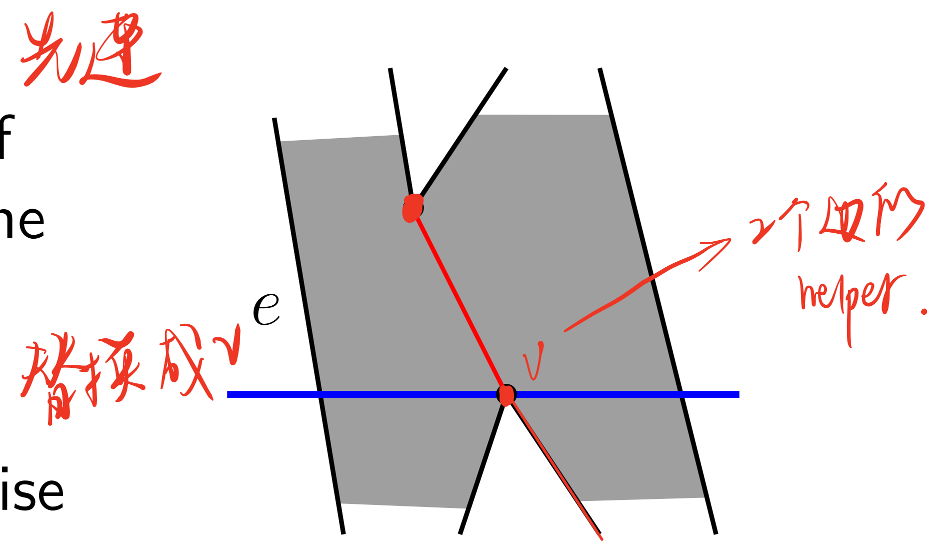
- Remove the edge clockwise from v from T
- Find the edge e directly left of v, and replace its helper by v



Event handling

Split vertex v:

- Find the edge e directly left of v, and choose as a diagonal the edge between its helper and v
- Replace the helper of e by v
- Insert the edge counterclockwise from v in T, with v as its helper



Efficiency

points

decreasing

Sorting all events by y-coordinate takes $O(n \log n)$ time

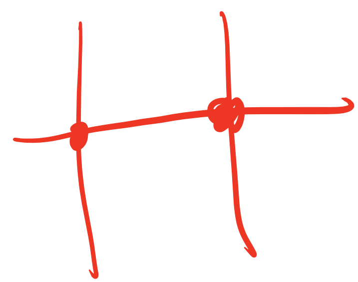
Every event takes $O(\log n)$ time, because it only involves querying, inserting and deleting in T

BST

Degenerate cases

Question: Which degenerate cases arise in this algorithm?

两个点
都在 sweep
line 上



但这不会更复杂 只须考虑时间
想如何更好
确保

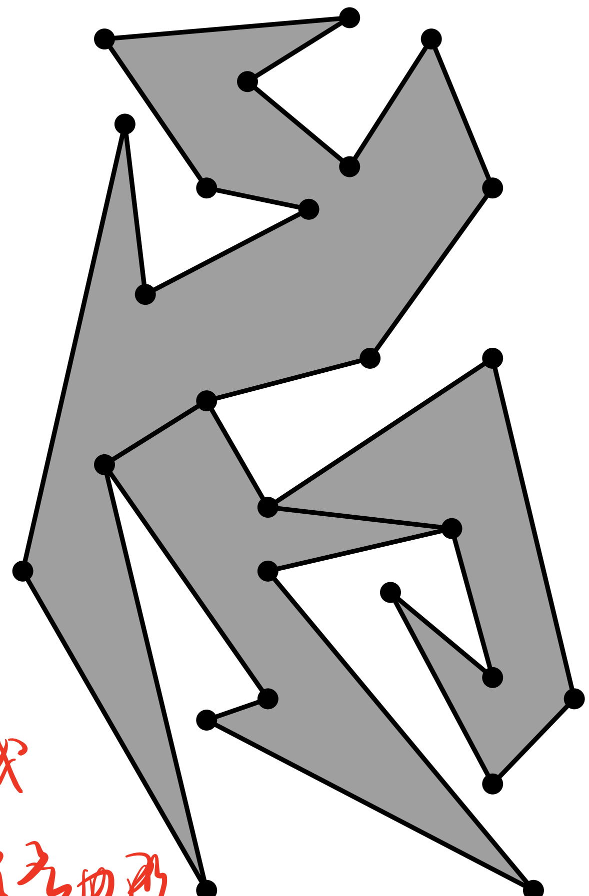
Representation

代表性

A simple polygon with some diagonals is a subdivision ⇒ use a DCEL

→ Duplicated Connected Edge List.

Question: How many diagonals may be chosen to the same vertex?



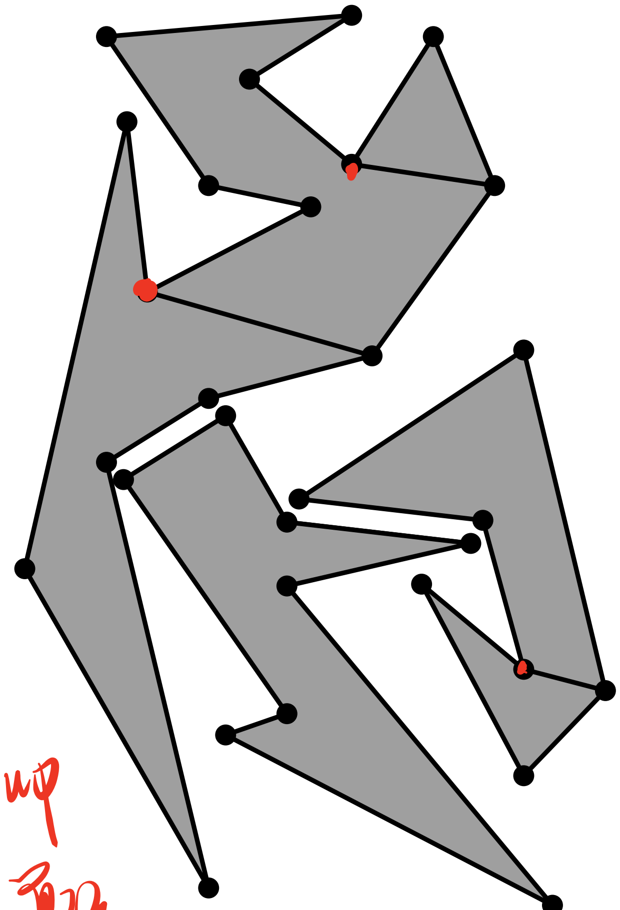
我们加这些对角线
然后分割成更多多边形
然后之后对这些多边形操作

More sweeping

With an upward sweep in each subpolygon, we can find a diagonal down from every merge vertex (which is a split vertex for an upward sweep!)

This makes all subpolygons y-monotone

但是没有merge
所以需要 bottom up
然后再跑一次算法



Result

Theorem: A simple polygon with n vertices can be partitioned into y-monotone pieces in $O(n \log n)$ time

因为我们用 nlogn time 来 sort 点 corners

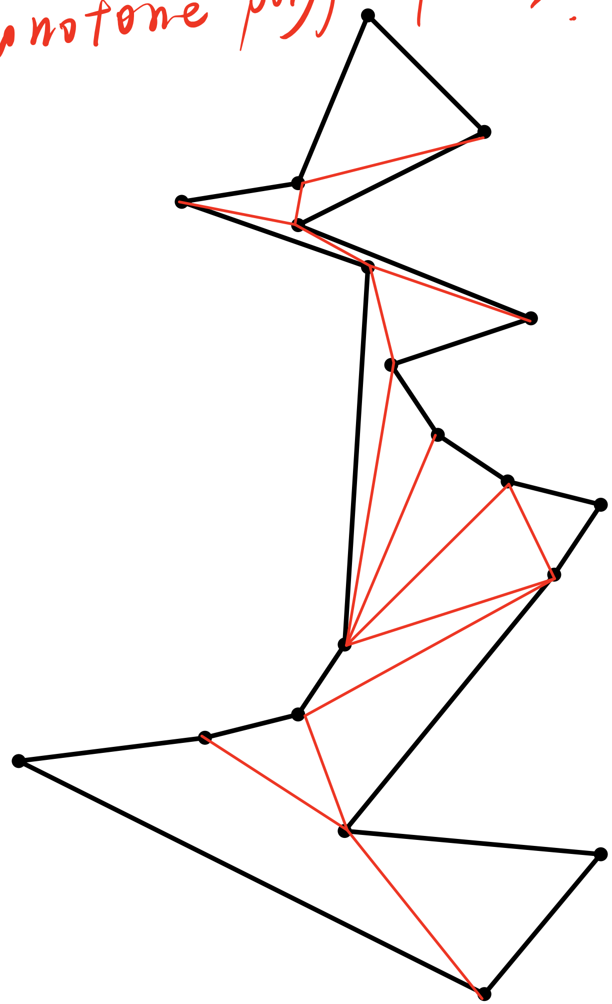
同时也用 logn 时间来更新数据结构(对于每个点)

所以 nlogn

Triangulating a monotone polygon

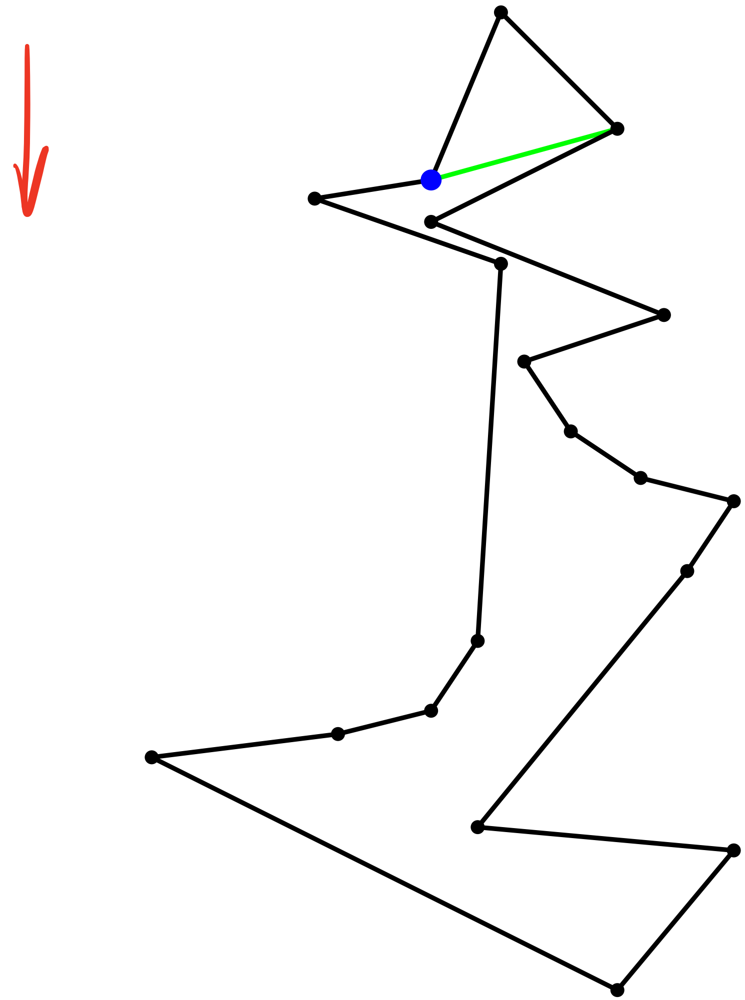
上面是如何分成 y-monotone polygon pieces.

How to triangulate a y-monotone polygon?



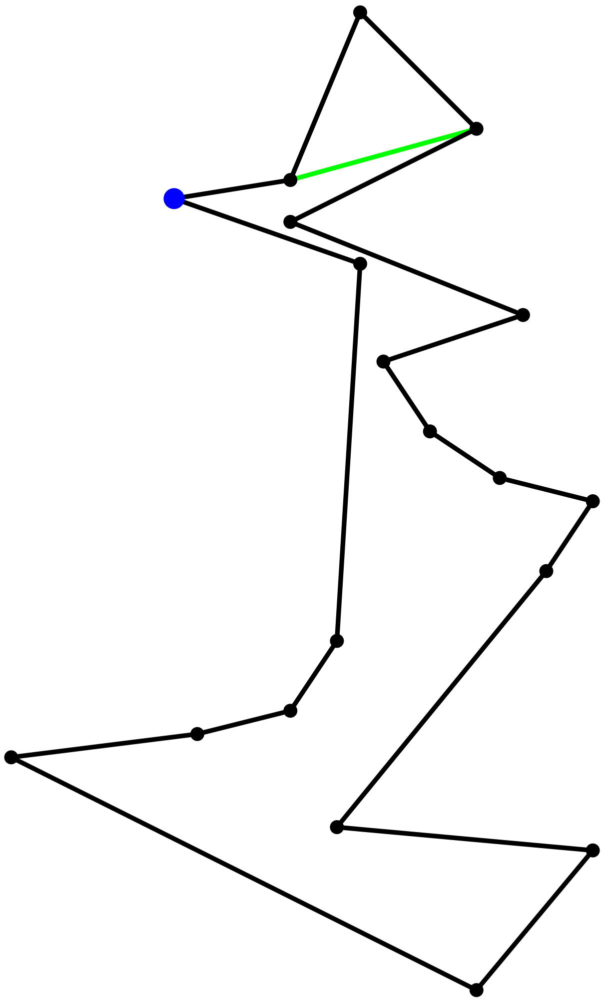
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



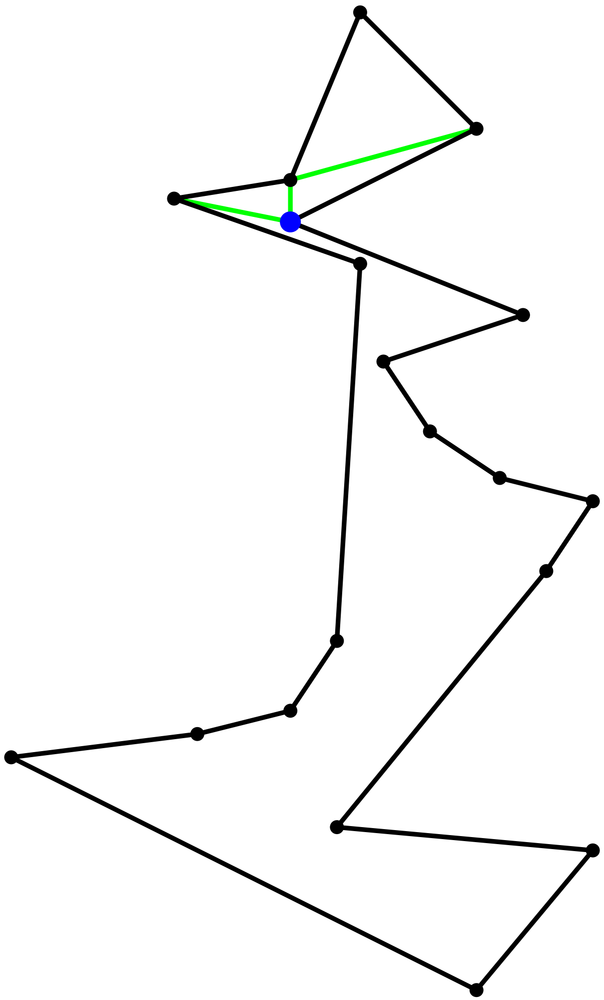
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



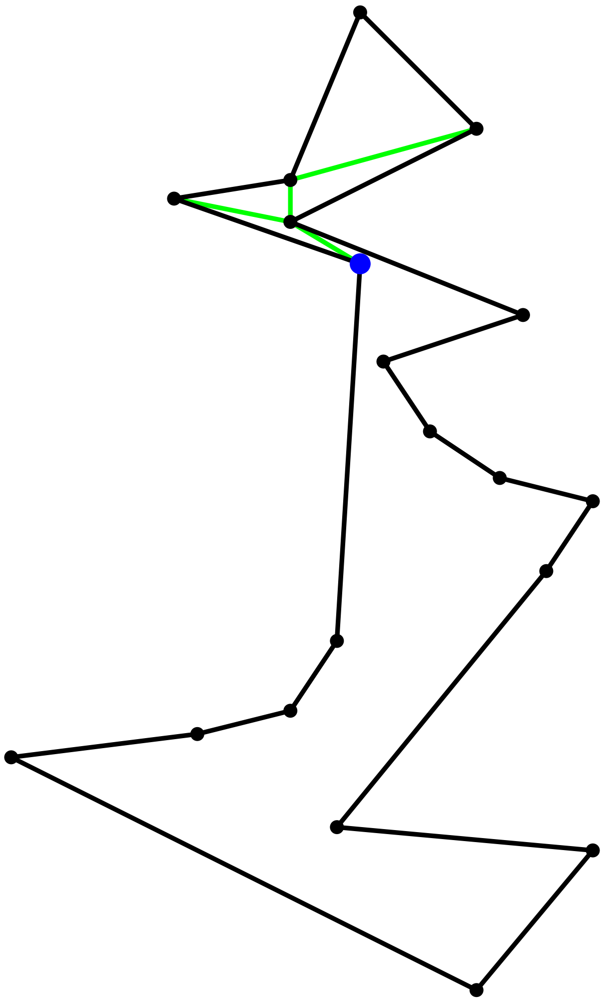
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



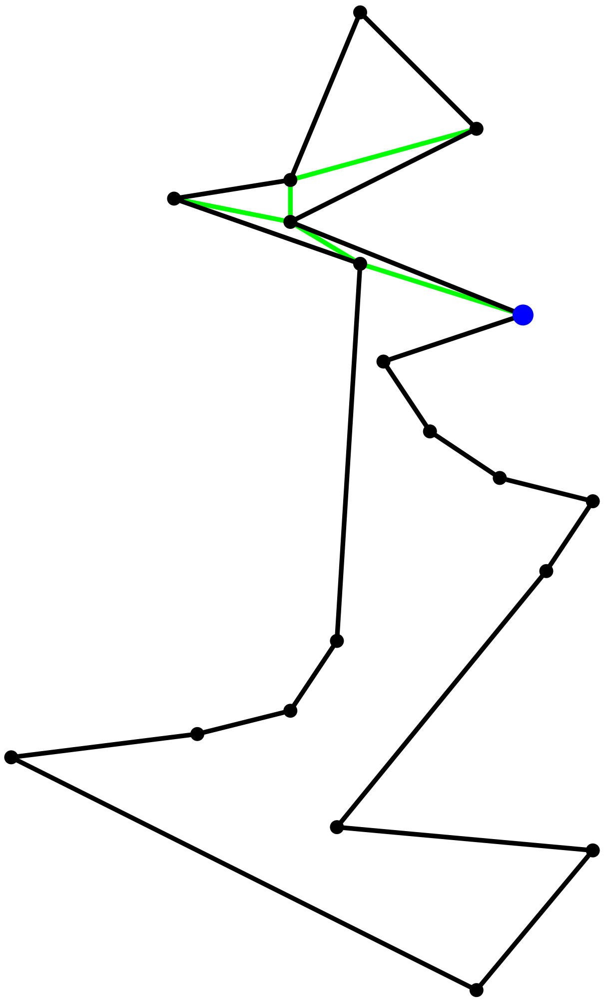
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



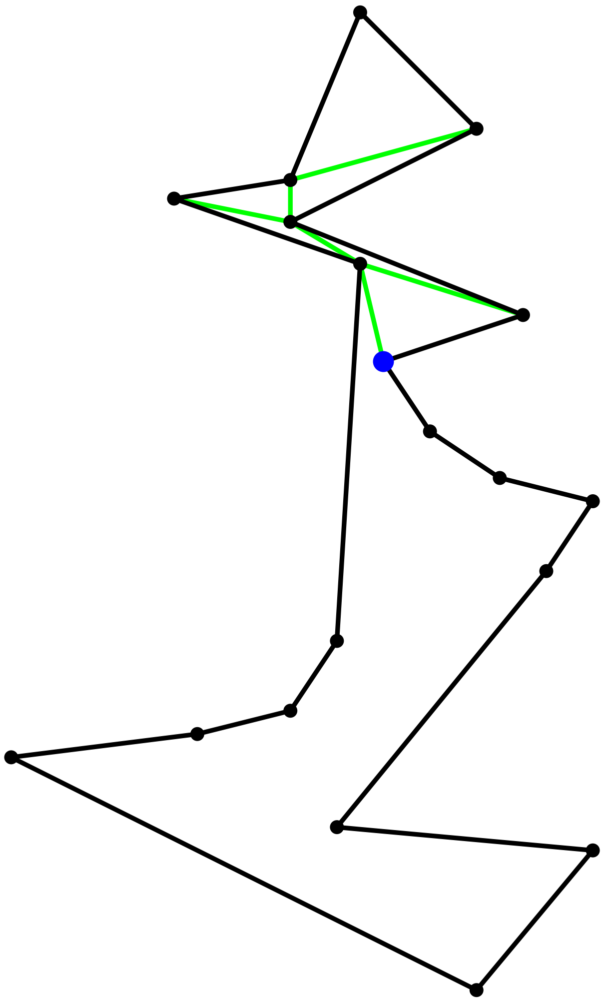
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



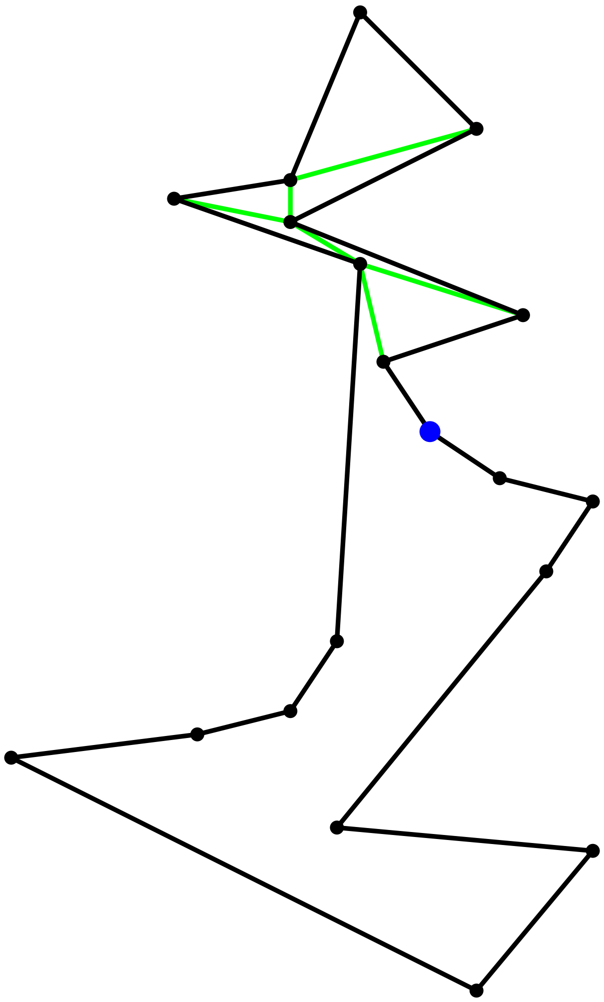
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



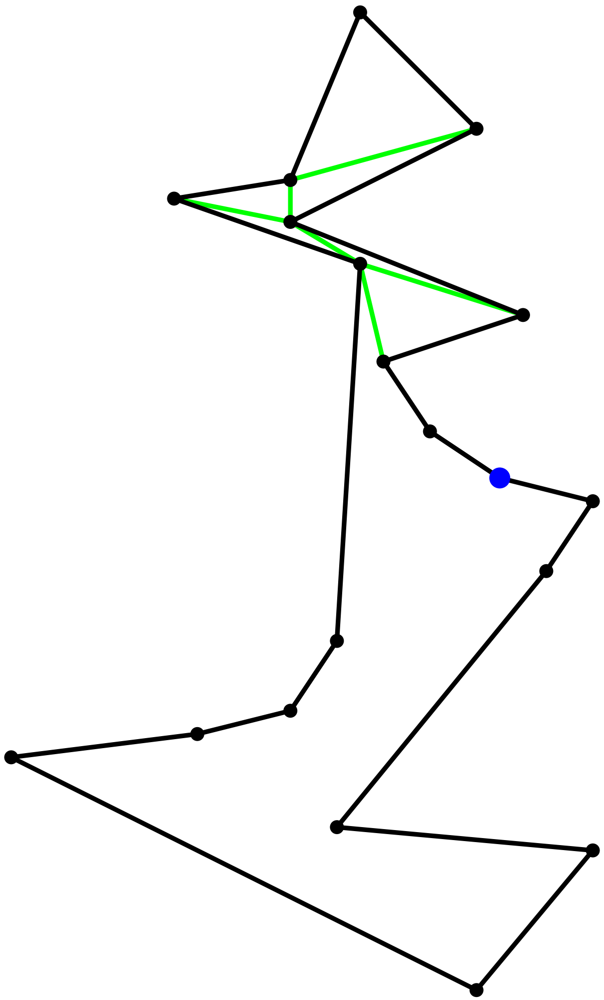
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?

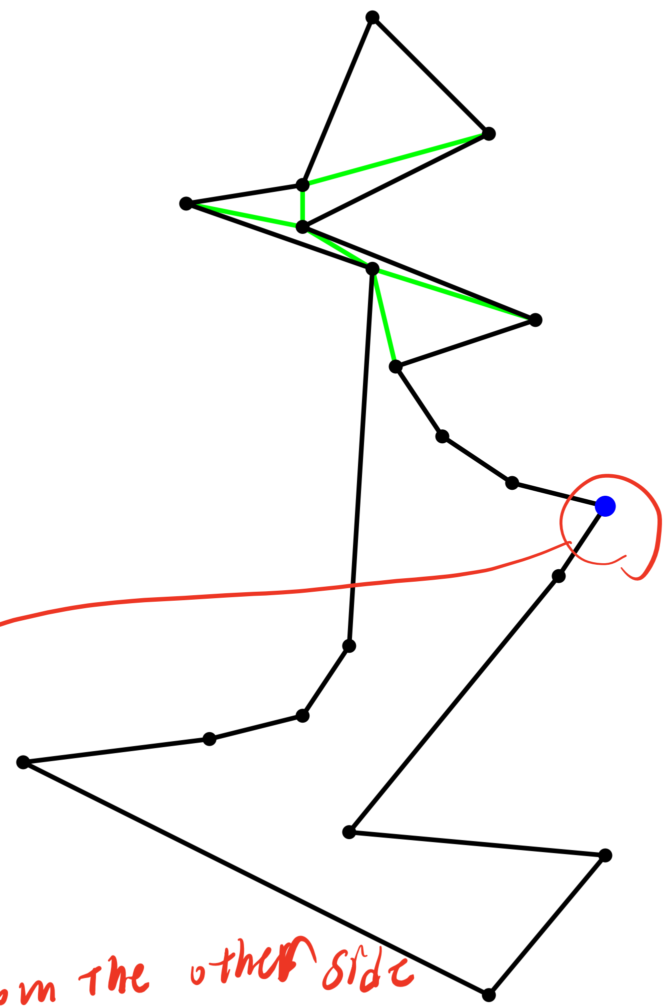


Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



Triangulating a monotone polygon

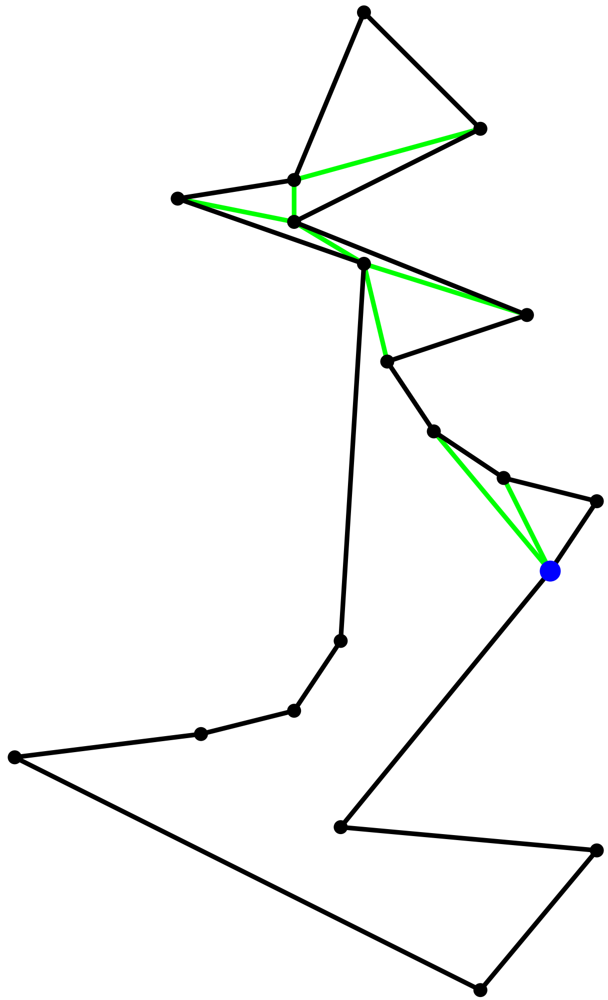


How to triangulate a y-monotone polygon?

we name a chain of concave vertices from one side of the polygons and it ends in a corner from the other side of the polygon

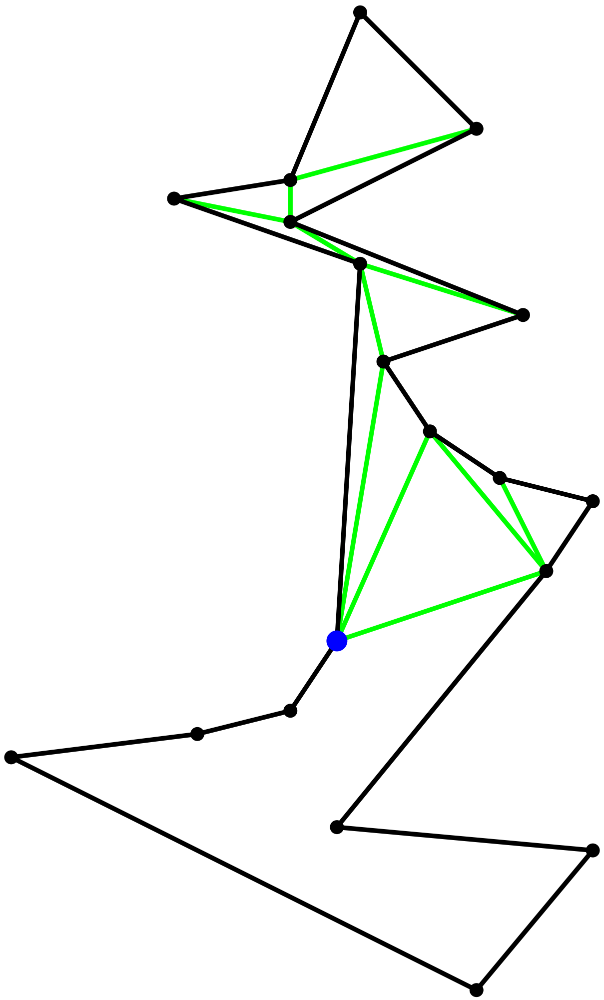
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



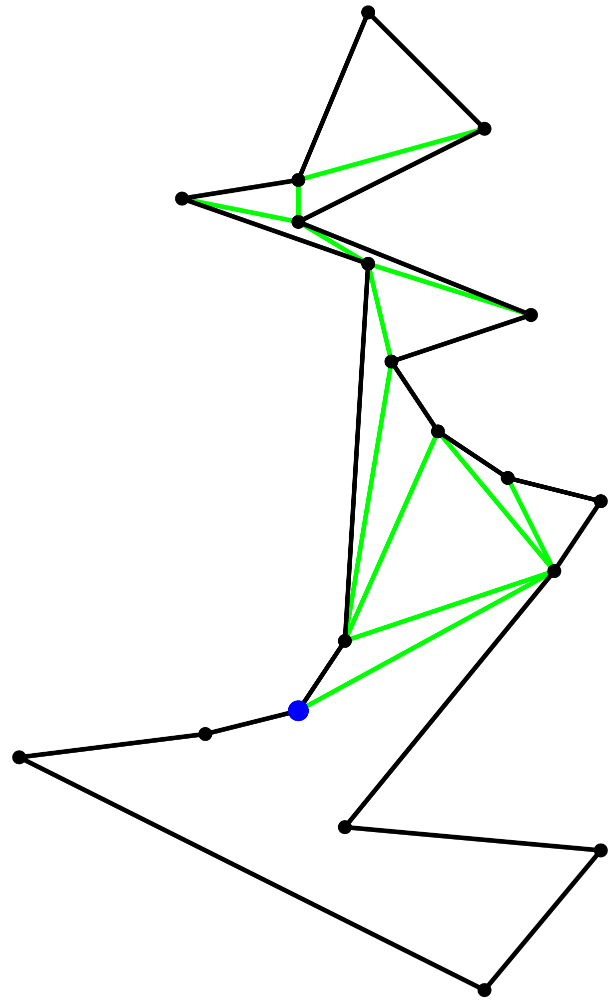
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



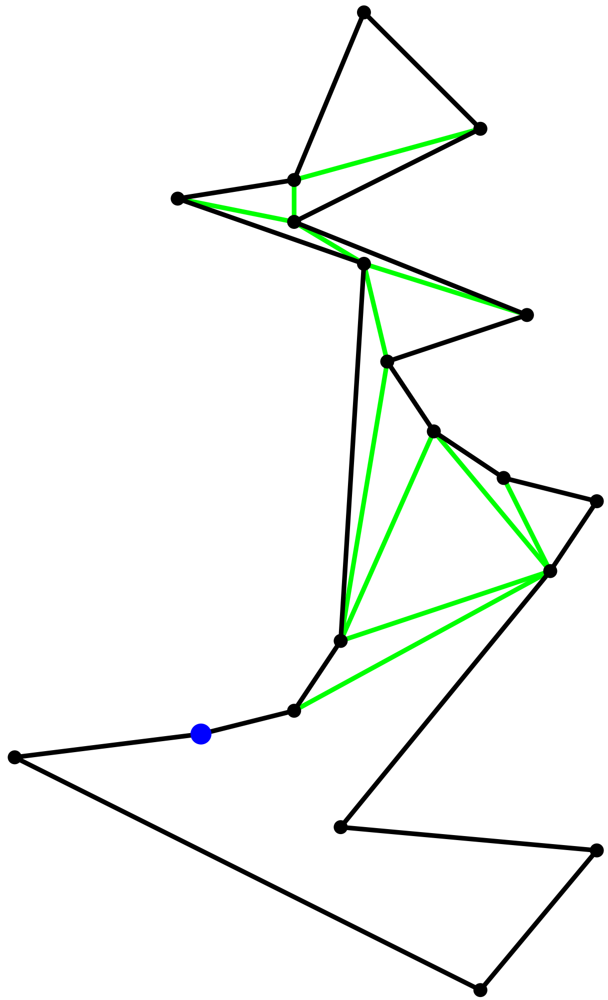
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



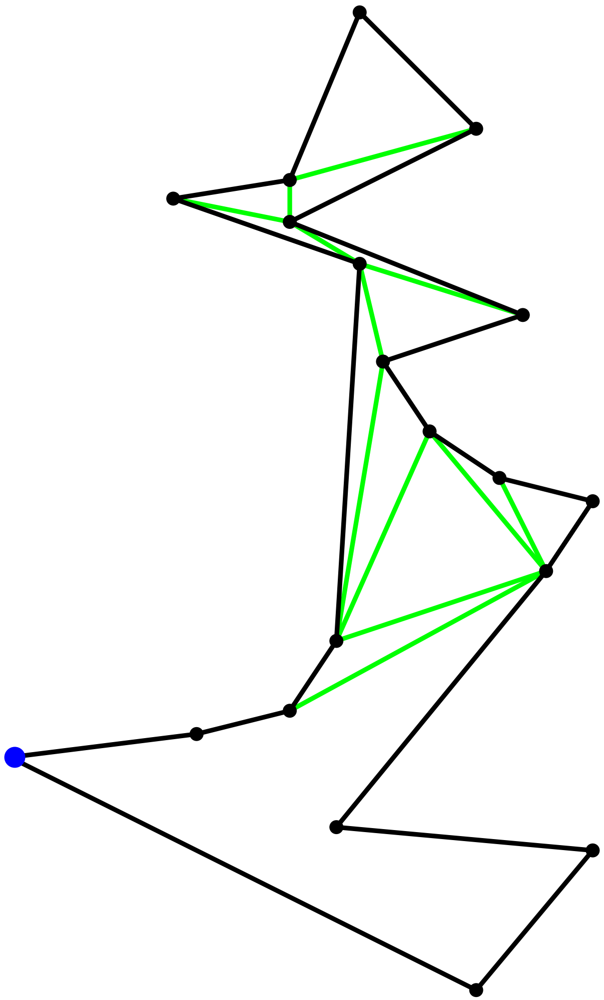
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



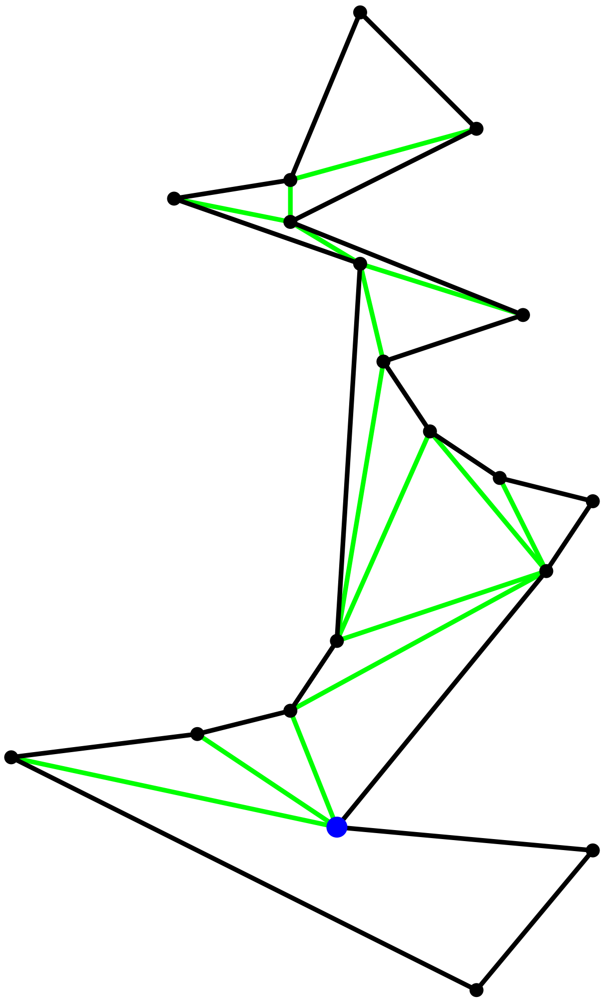
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



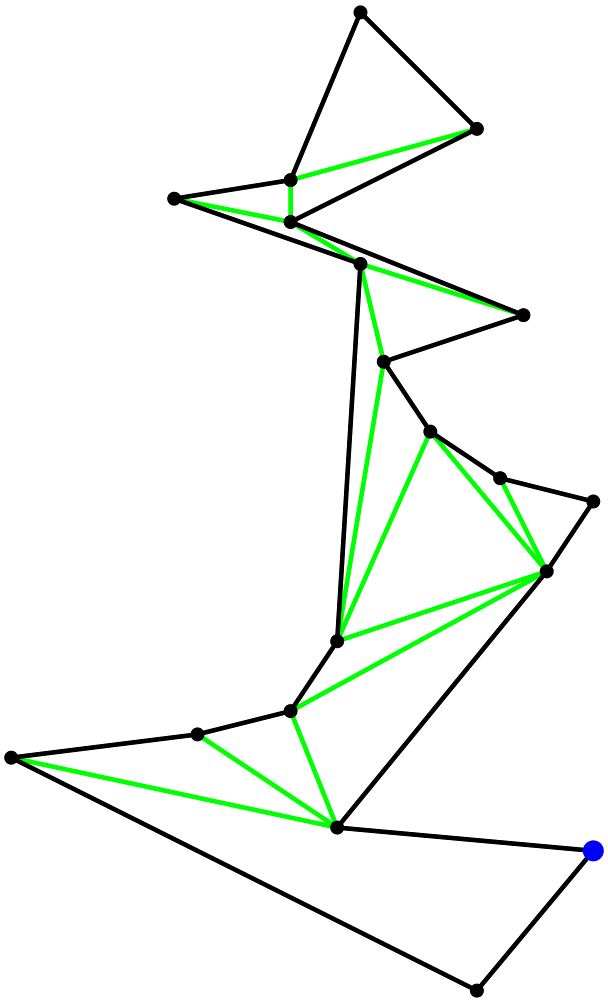
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



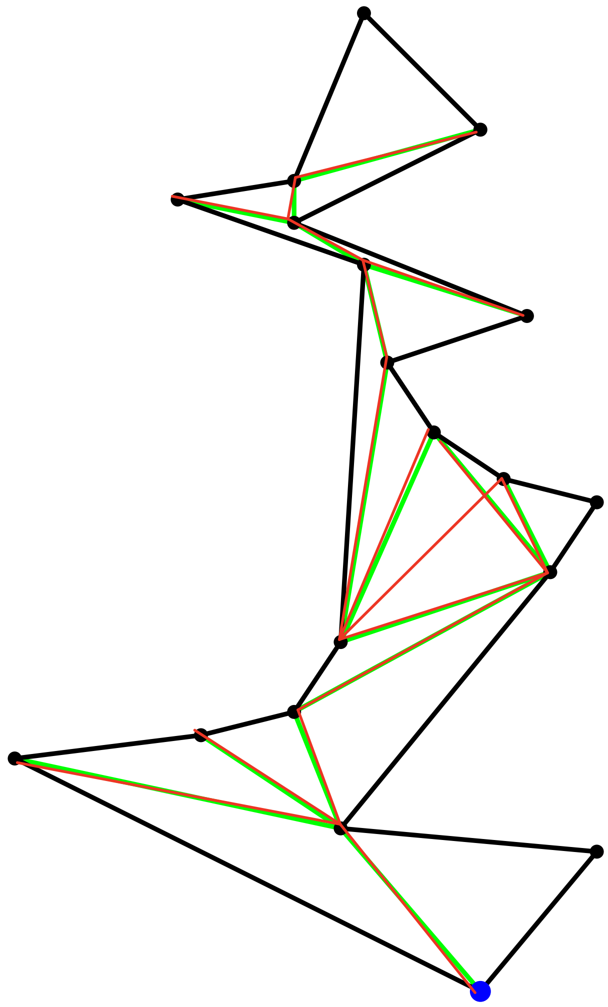
Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



Triangulating a monotone polygon

How to triangulate a y-monotone polygon?



The algorithm

- Sort the vertices top-to-bottom by a merge of the two chains
- Initialize a stack. Push the first two vertices
- Take the next vertex v, and triangulate as much as possible, top-down, while popping the stack
- Push v onto the stack

Result

Theorem: A simple polygon with n vertices can be partitioned into y-monotone pieces in $O(n \log n)$ time

Theorem: A monotone polygon with n vertices can be triangulated $O(n)$ time

for each vertex will be added to the stack and remove from some point. each will be handled twice, each of these handle take constant time. So therefore all in all it will just take linear time.

Can we immediately conclude:

A simple polygon with n vertices can be triangulated $O(n \log n)$ time ???

最后的n不是分割的n了。

因为每一次分割ች split和merge都会加一条对角线以便多出2个点，但是每次都是这样，最终点还是$O(n)$个点。

Result

We need to argue that all y-monotone polygons together that we will triangulate have $O(n)$ vertices

Initially we had n edges. We add at most $n-3$ diagonals in the sweeps. These diagonals are used on both sides as edges. So all monotone polygons together have at most $3n-6$ edges, and therefore at most $3n-6$ vertices

Hence we can conclude that triangulating all monotone polygons together takes only $O(n)$ time

Theorem: A simple polygon with n vertices can be triangulated $O(n \log n)$ time