Using orientation tests to solve basic problems on polygons

Vera Sacristán

Computational Geometry
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Point in polygon test

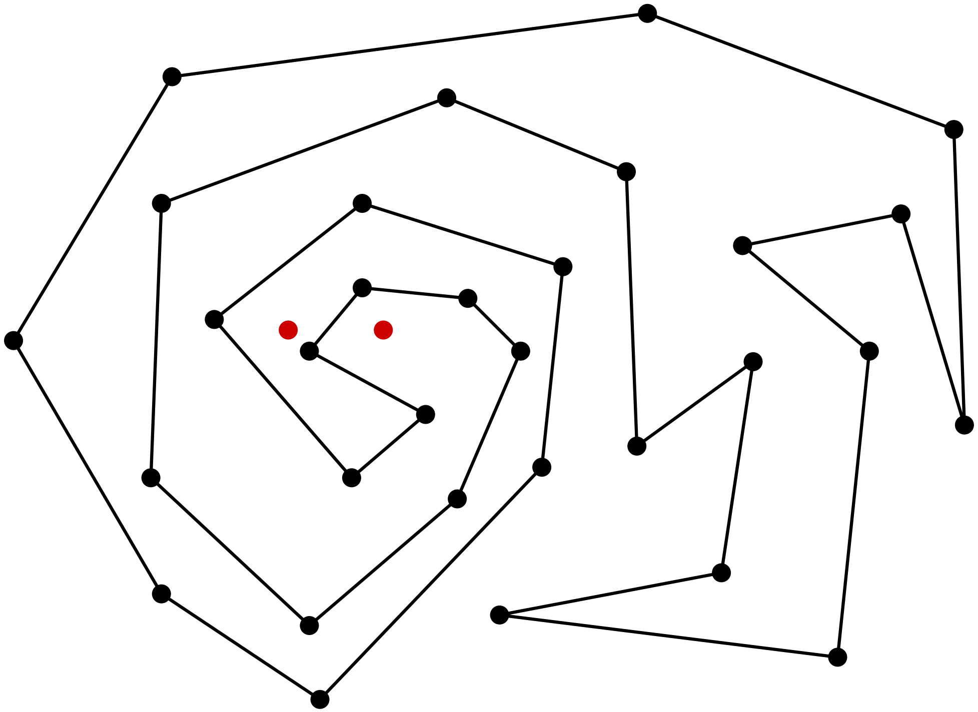
Input:

A polygon $p_1, p_2, \ldots, p_n$

A query point q

Output:

Yes/No $q \in P$



Point in polygon test

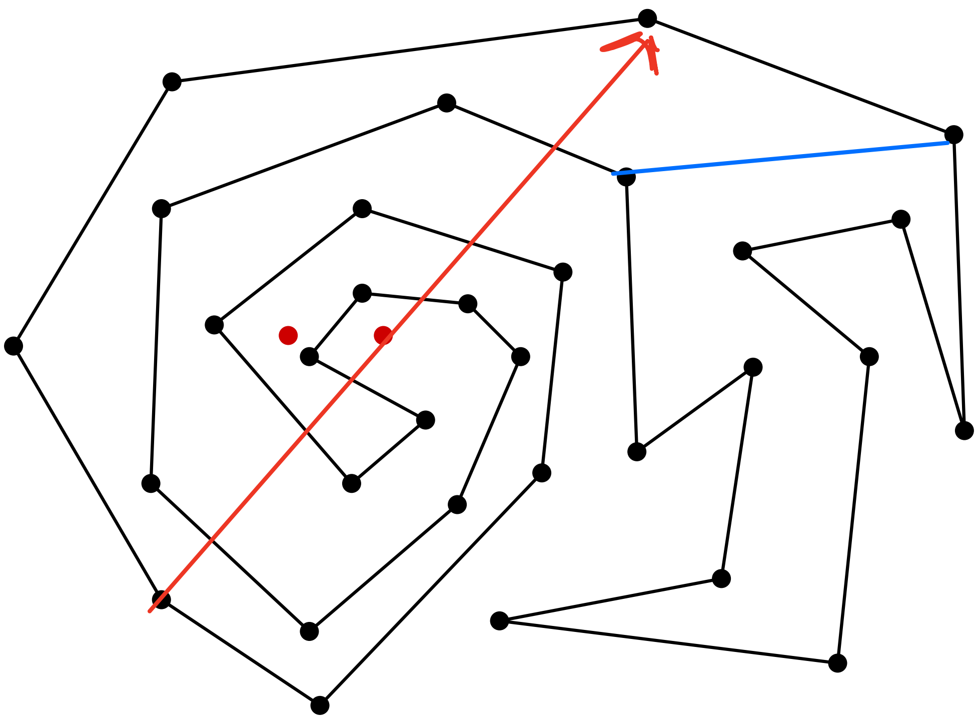
Input:

A polygon $p_1, p_2, \ldots, p_n$

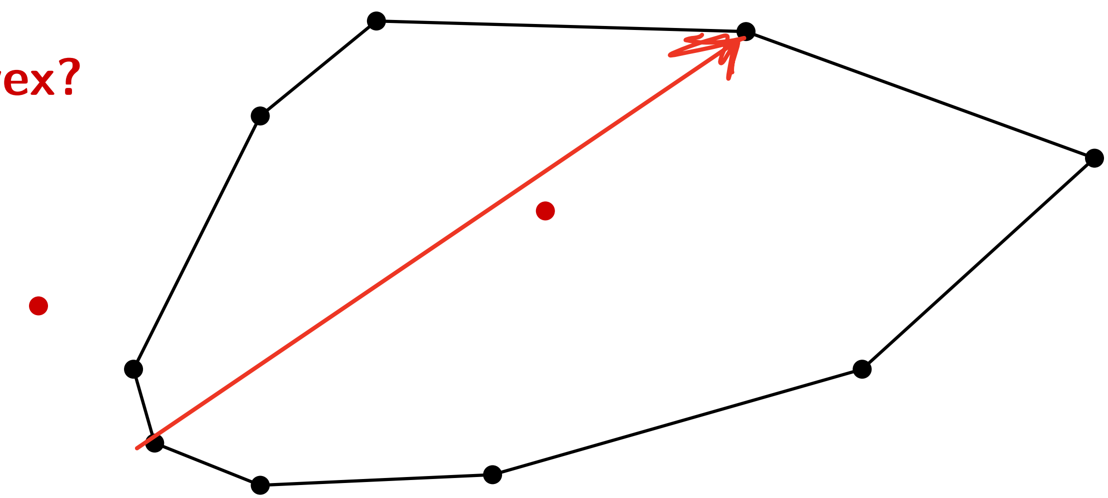
A query point q

Output:

Yes/No $q \in P$

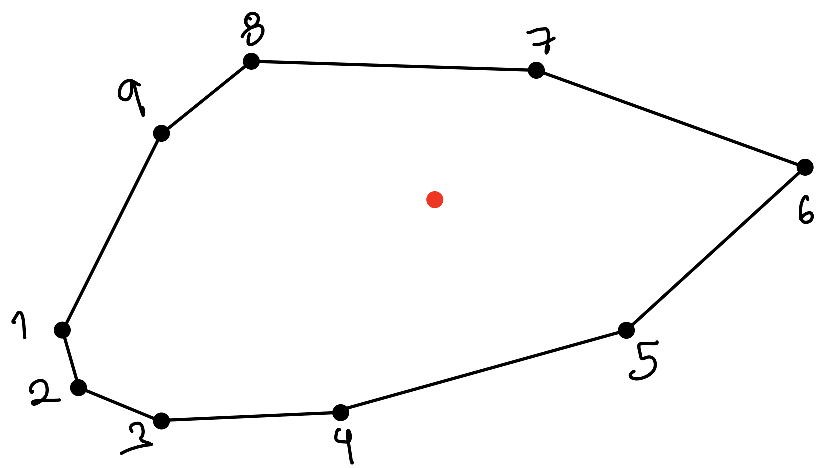


What if the polygon is convex?

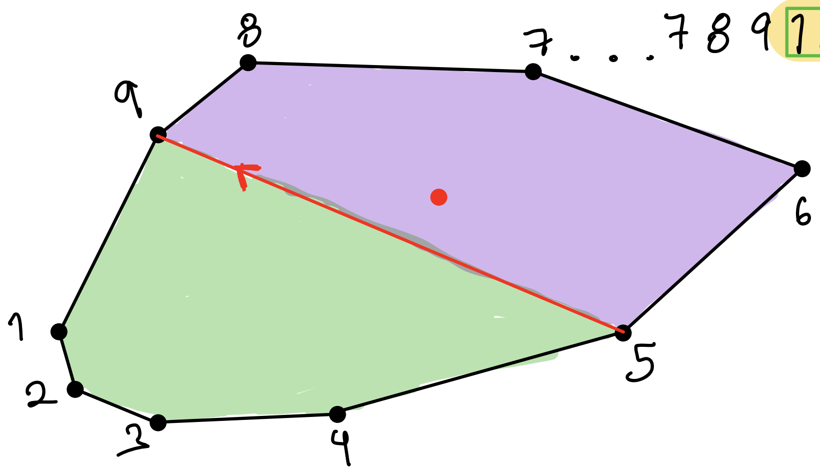


Ejemplo: BB.

. . . 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 . . .



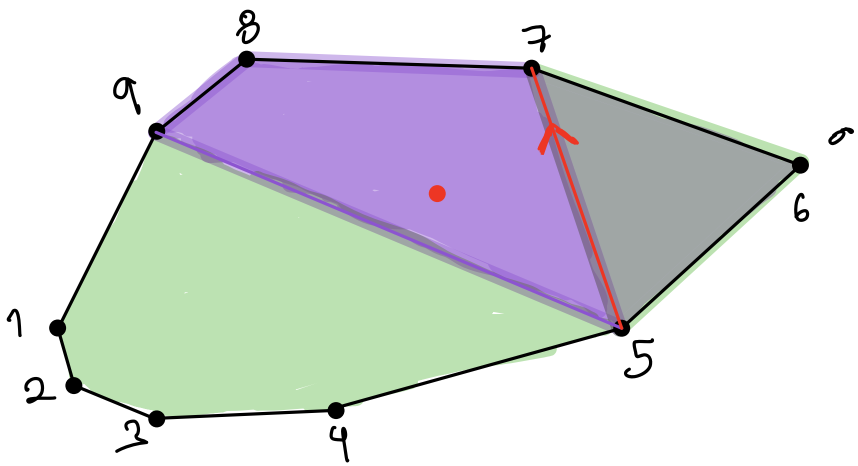
. . . 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 . . .



polígonos: 1 2 3 4 5 9

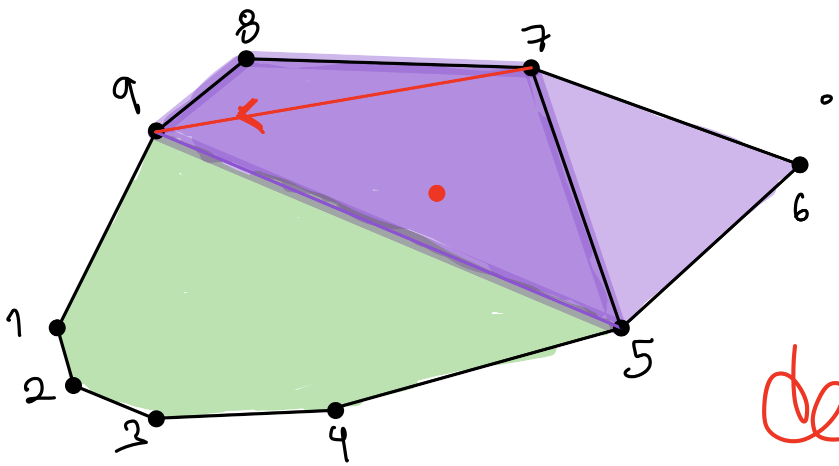
5 6 7 8 9

$$9 = 5 + 4.$$



...789 567 89 5678...

polígonos: 567, 5789.



...89 57 89 578...

polígonos: 579
789

Caso base: polígono de tamaño 3 (oreja)

¿Cómo dividimos el polígono en tiempo constante?

O(n) pre-pro. O(lg n).

Shamos: O(lg n).

USING ORIENTATION TESTS ON POLYGONS

Intersection test line - polygon

Input:

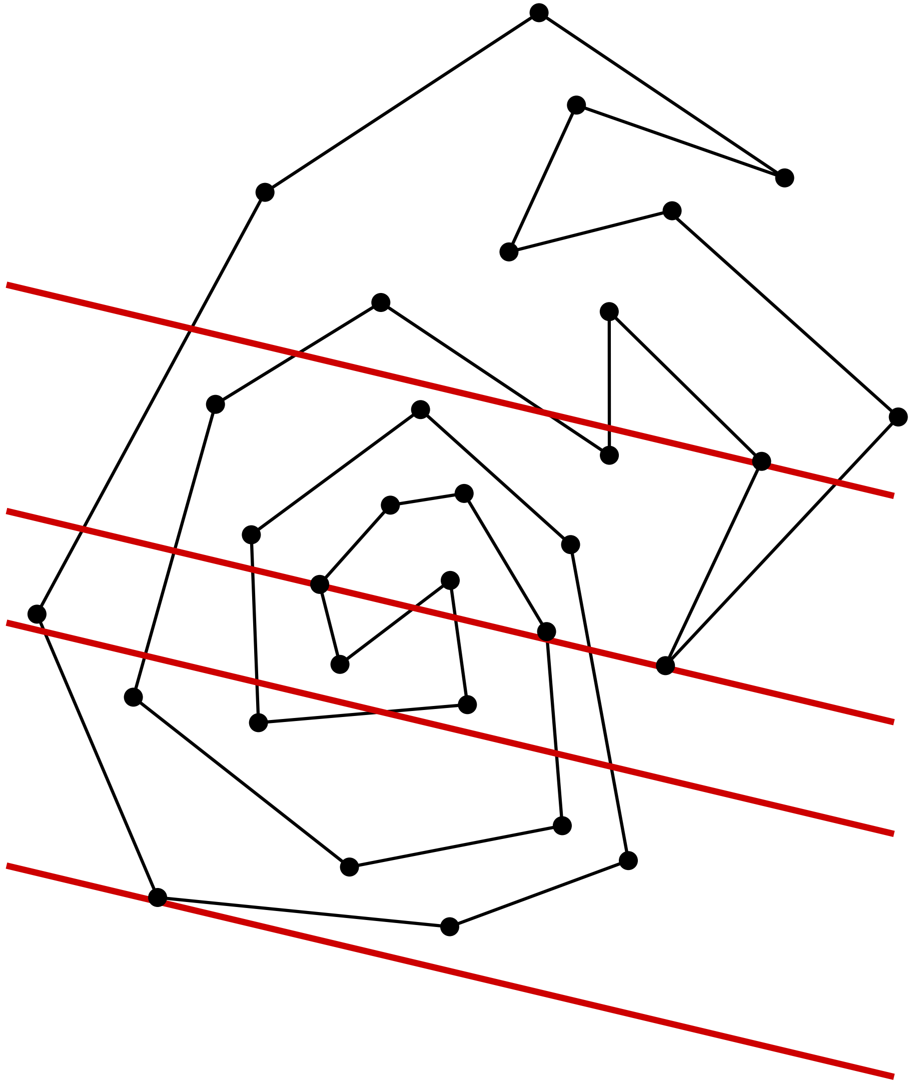
ℓ: a line (through p and q)

P: a polygon (with vertices $p_1, p_2, \ldots, p_m$)

Yes/No they intersect.

If they do, the edges of P intersecting ℓ

$O(n)$



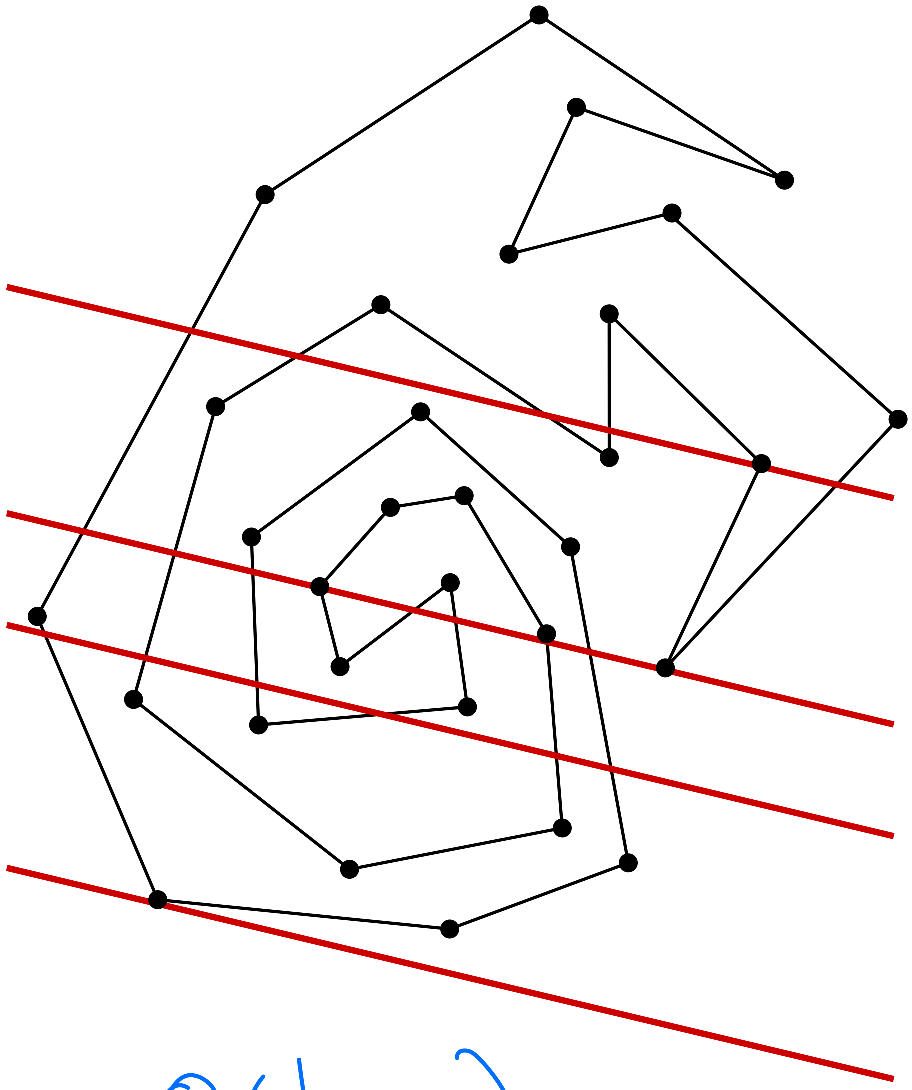
USING ORIENTATION TESTS ON POLYGONS

Intersection test line - polygon

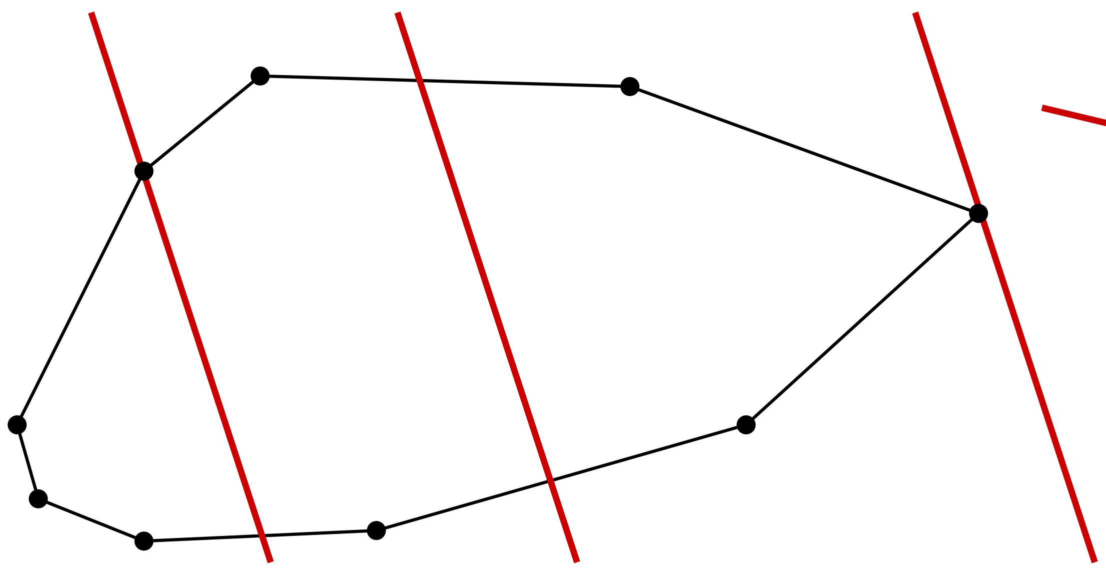
Input:

ℓ: a line (through p and q)
P: a polygon (with vertices $p_1, p_2, \ldots, p_m$)

Yes/No they intersect.
If they do, the edges of P intersecting ℓ



What if the polygon is convex?



O(lg n).

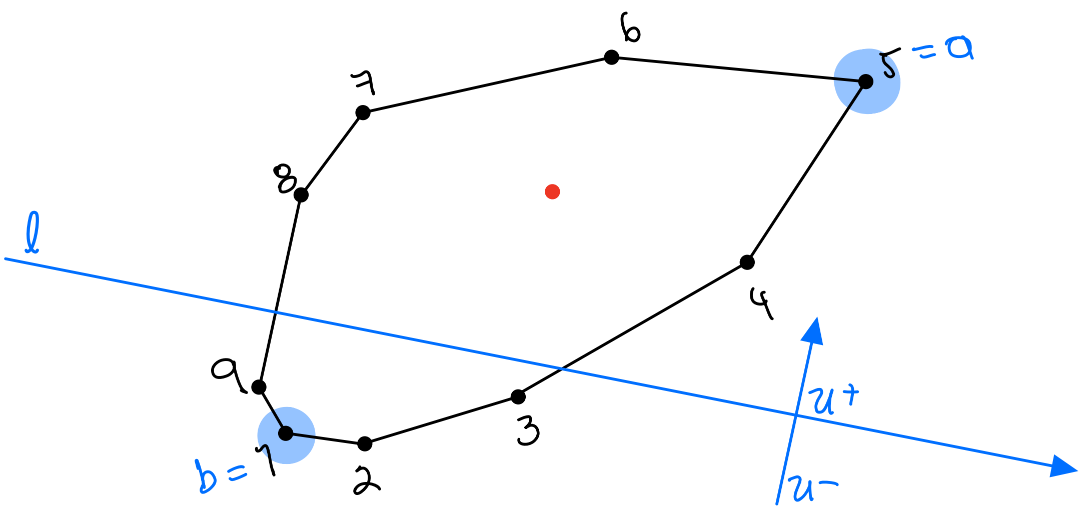
Problema: Dado un polígono P con vértices $p_1, p_1, p_2, \ldots, p_n$ (en sentido anti-horario)

Dada una recta l: p, q.

Determinar: Si P y l se intersectan.

Idea:

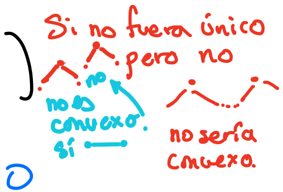
1. Encontrar los extremos de P en dirección u^+ y u^-. Sean a y b.
2. Si a y b están del mismo lado de $l \Rightarrow$ no hay intersección.
   Si a y b están en lados distintos de $l \Rightarrow$ hay intersección.

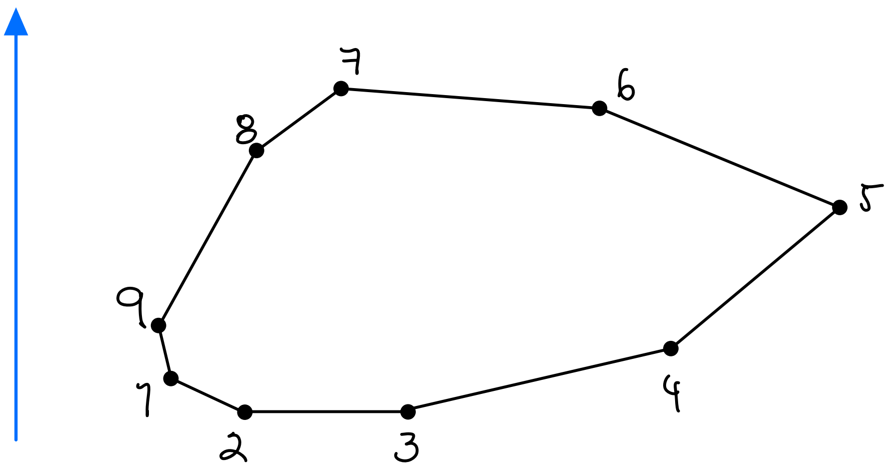


Deseamos $O(\lg n)$.

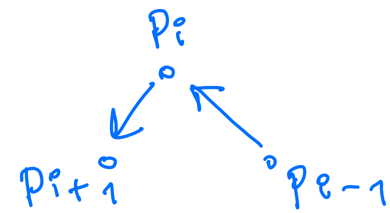
¿Cómo encontramos los extremos? (En $O(\lg n)$)

Vamos a resolver el problema de encontrar el vértice de P con mayor coordenada y. (Supondremos que es único.)



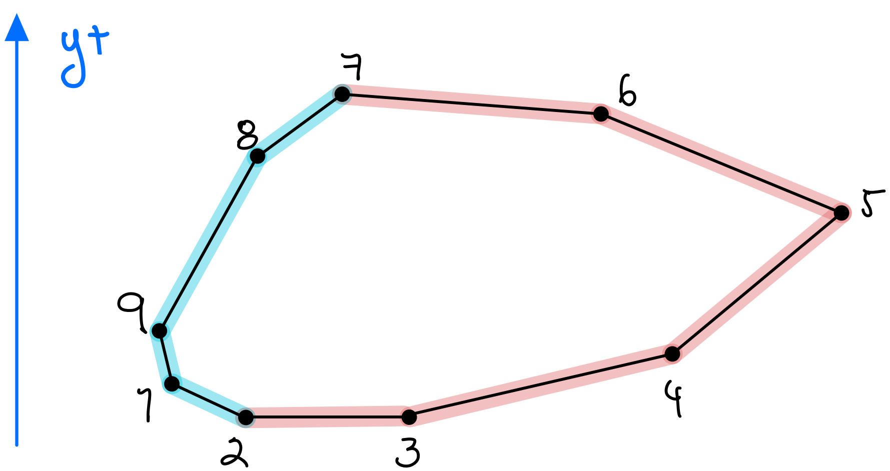


Idea: el vértice más alto necesariamente se ve así:



- Recorrer el polígono y descartar subcadenas usando búsqueda binaria.

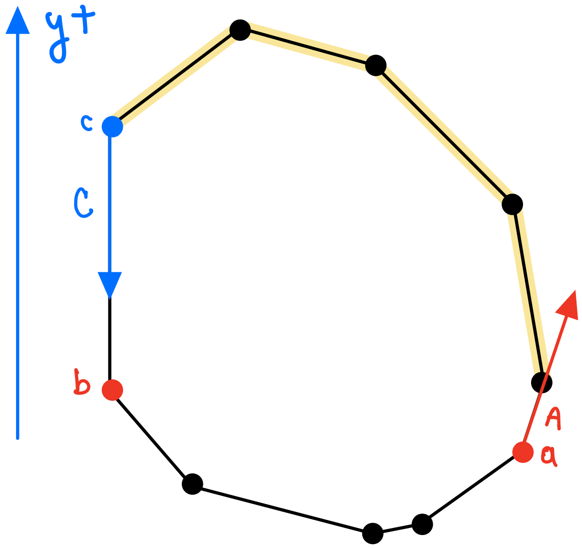
Notemos:



- Criterio de búsqueda:

Defino: 1. Cadena derecha = los vértices entre el punto más bajo y el punto más alto.

2. Cadena izquierda = los vértices entre el punto más alto y el punto más bajo.



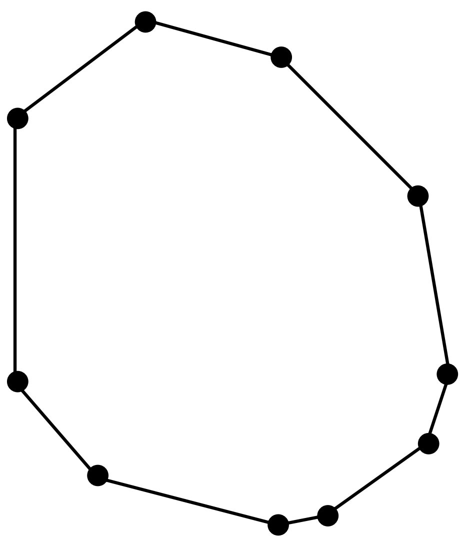
Sean a, b y c vertices del poligono, tales que c está justo ala mitad entre a y b.

Sea A y C los vectores:

$$A = (a, a+1)$$

$$C = (c, c+1)$$

Considerando las direcciónes de A y C tenemos 4 posibles casos:

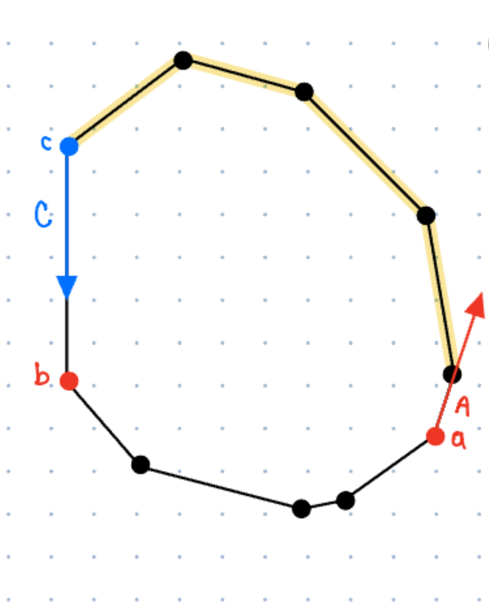


① A↑, C↓ (a cadena derecha, c izquierda)

② A↓, C↑ (a izquierda, c derecha)

③ A↓, C↓ (izquierda)

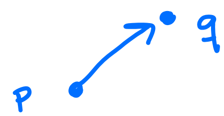
④ A↑, C↑ (derecha)



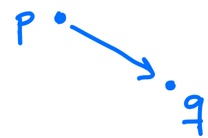
① A ↑ , C ↓ (a cadena derecha, c izquierda)

⟶ busco en [a, c]

- Un vector $\overrightarrow{(p, q)}$ apunta hacia arriba si $p_y < q_y$

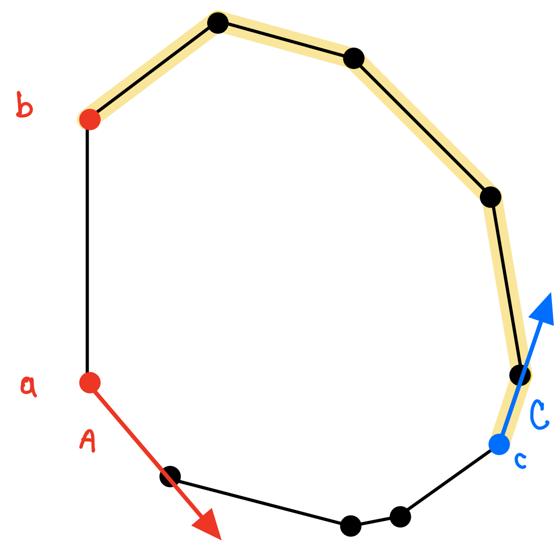


apunta hacia abajo si $p_y > q_y$



② A ↓ , C ↑ (a izquierda, c derecha)

⟶ busco en [c, b]



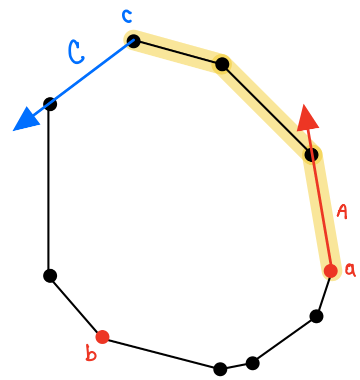
③ A↓, C↓ (izquierda)

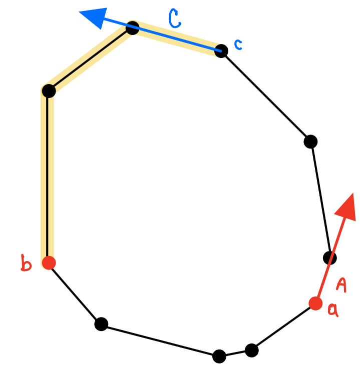
i. si a está encima de c

→ busco en [a, c]

ii. de lo contrario

→ busco en [c, b]





si a y b son adyacentes ⇒ a = c.

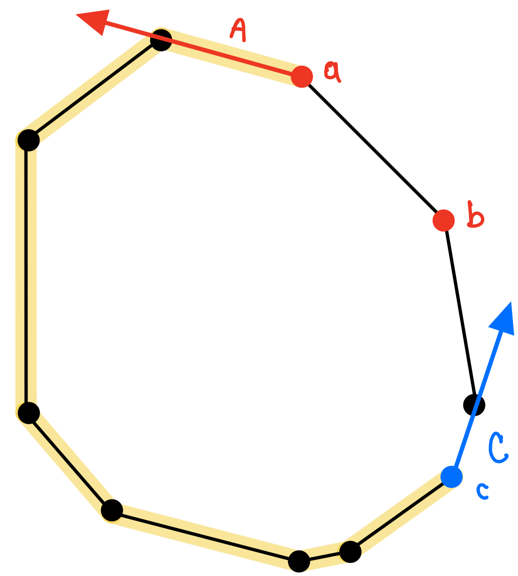
④ A↑, C↑ (derecha)

i. si a está abajo de c

→ busco en [a, c]

ii. de lo contrario

→ busco en ~~[a, c]~~ [c, b]



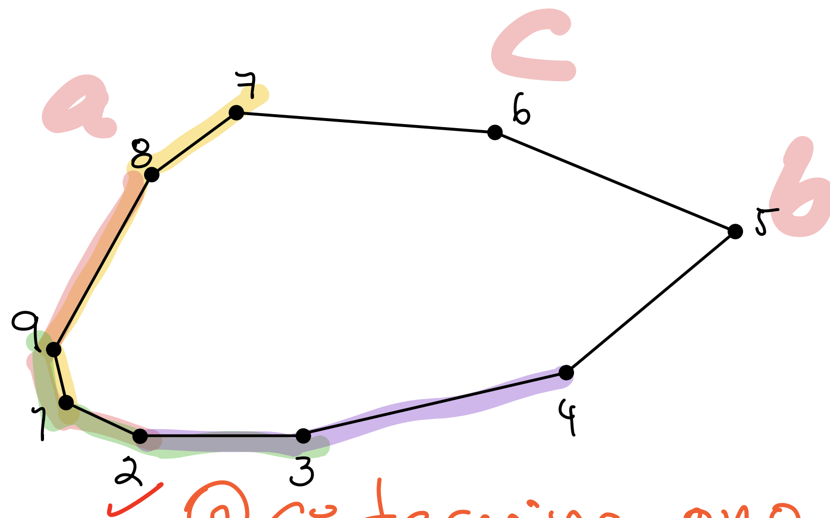
= cómo calcular =

```
int      Midway( int a, int b, int n )
{
  if (a < b) return ( a + b ) / 2;          ← piso
  else       return ( ( a + b + n ) / 2 ) % n;  ← piso.
}
```

Algorithm: HIGHEST POINT OF CONVEX POLYGON
Initialize a and b.
repeat forever
  $c \leftarrow$ index midway from a to b. → O(1)
  if $P[c]$ is locally highest then return c → O(1)
  if A points up and C points down → O(1)
    then $[a, b] \leftarrow [a, c]$ → O(1)
  else if A points down and C points up
    then $[a, b] \leftarrow [c, b]$
  else if A points up and C points up
    if $P[a]$ is above $P[c]$
      then $[a, b] \leftarrow [a, c]$
      else $[a, b] \leftarrow [c, b]$
  else if A points down and C points down
    if $P[a]$ is below $P[c]$
      then $[a, b] \leftarrow [a, c]$
      else $[a, b] \leftarrow [c, b]$

¿Cuál es su complejidad?

Ejemplo:



① Si termina o no
② ¿Cuántas veces se repite?

Si deseo encontrar los extremos en dirección u, reemplazamos el test sobre un vector V por:

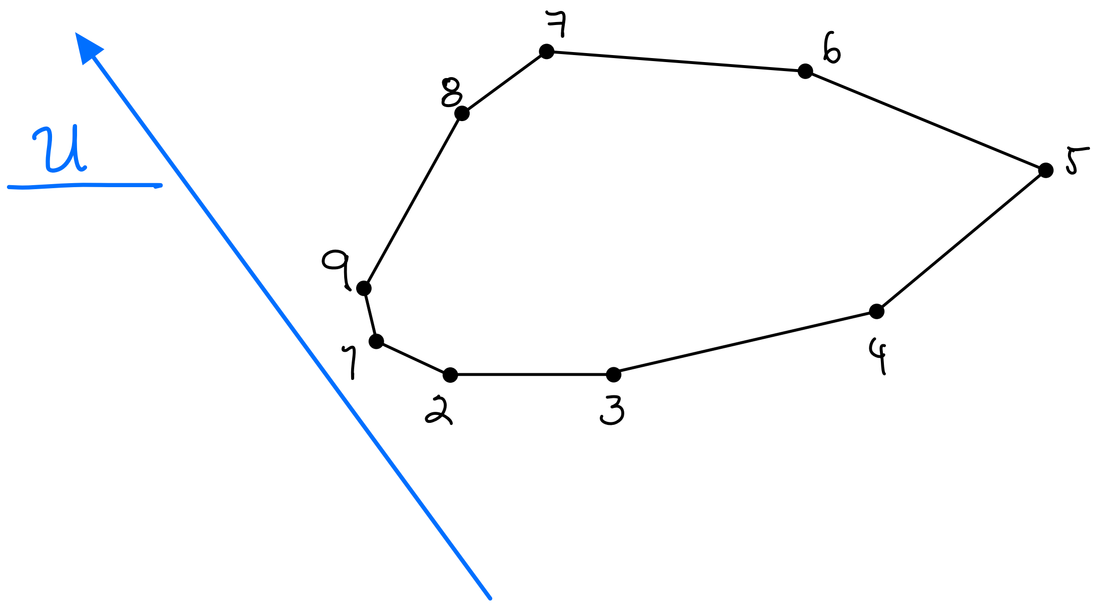
V apunta hacia abajo → $u \cdot V < 0$

V apunta hacia arriba → $u \cdot V > 0$

a arriba de c → $u \cdot (a-c) > 0$

de otra forma → $u \cdot (a-c) \leq 0$

etc...



Supporting lines point - polygon

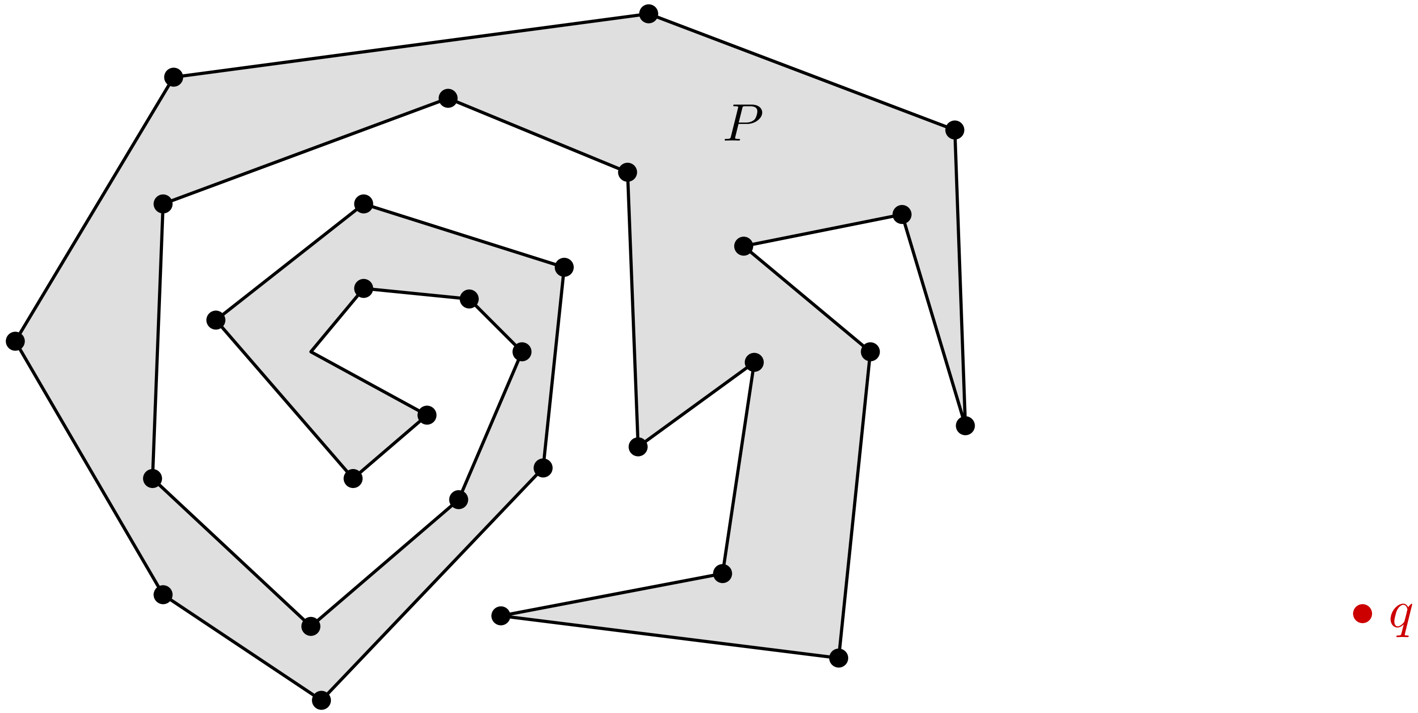
Input:

A polygon P with vertices $p_1, p_2, \ldots, p_n$

A point q not belonging to the convex hull of P

Output:

Lines through q and P that leave all of P to one side



Supporting lines point - polygon

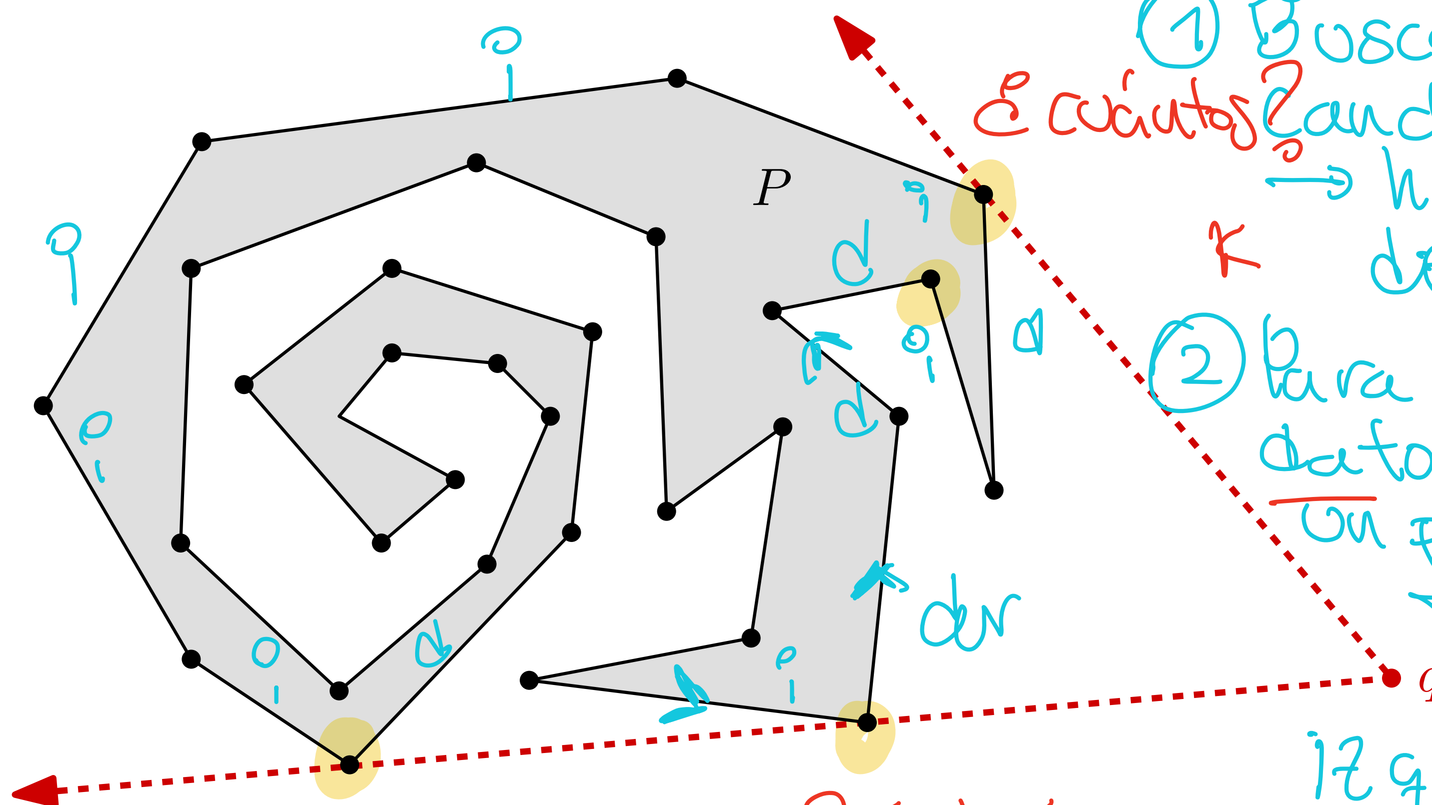
Input:

A polygon P with vertices $p_1, p_2, \ldots, p_n$

A point q not belonging to the convex hull of P

Output:

Lines through q and P that leave all of P to one side



Algoritmo:

(1) Busco los pts candidatos ¿cuántos? k

→ hay un cambio de dirección.

(2) Para cada candidato reviso si es un punto de tangencia o no.

$2 \leq k \leq n$

∀q → O(kn).

Supporting lines point - polygon

Input:

A polygon P with vertices $p_1, p_2, \ldots, p_n$

A point q not belonging to the convex hull of P

Output:

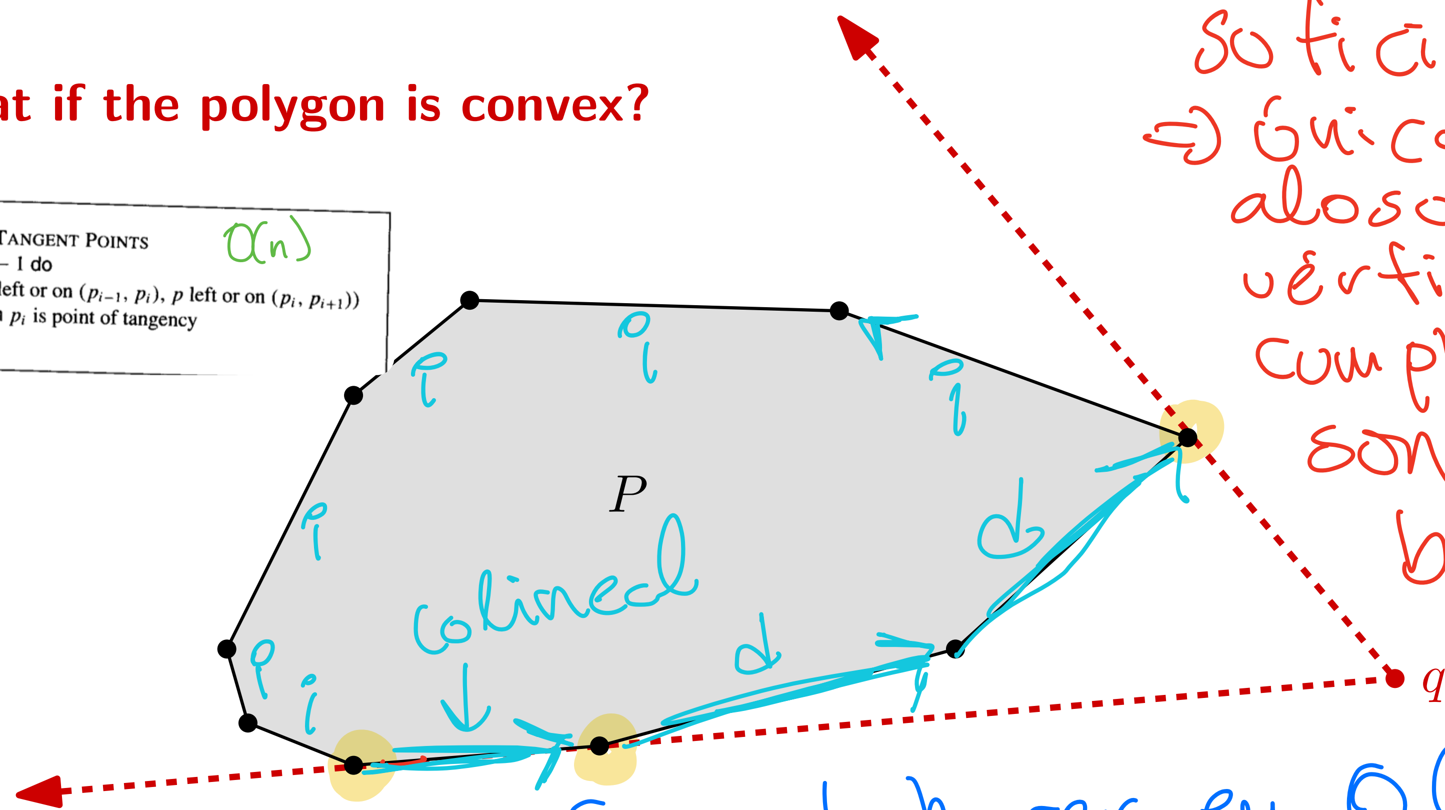
Lines through q and P that leave all of P to one side

En este caso la condición es suficiente.
⇒ Únicamente hay a lo sumo 3 vértices que la cumplen y esos son los que buscamos.

What if the polygon is convex?

Algorithm: TANGENT POINTS
for $i = 0$ to $n-1$ do
if Xor (p left or on (p_{i-1}, p_i), p left or on (p_i, p_{i+1}))
then p_i is point of tangency

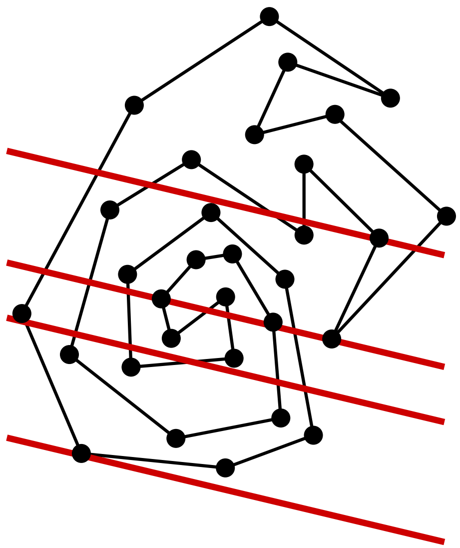
$O(n)$

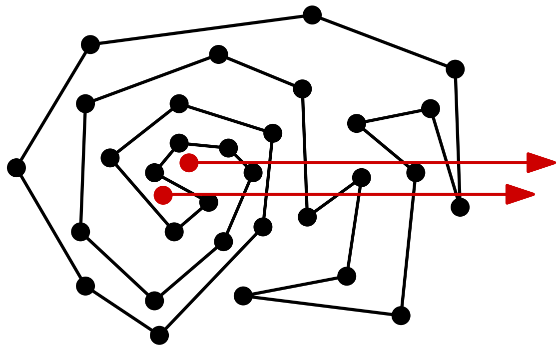


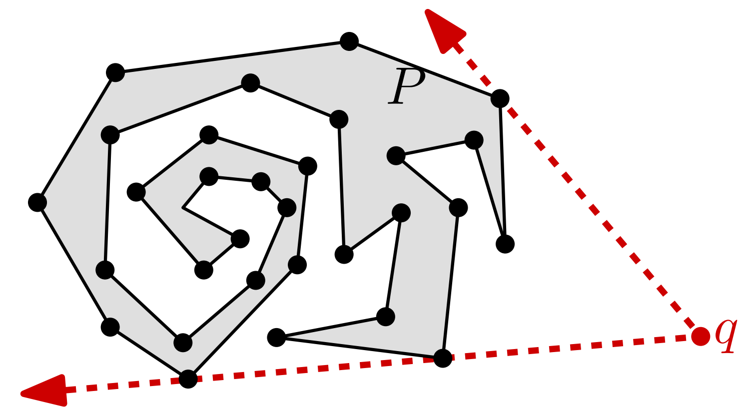
Se puede hacer en $O(\lg n)$.

How did we prove the correctness of our solutions?

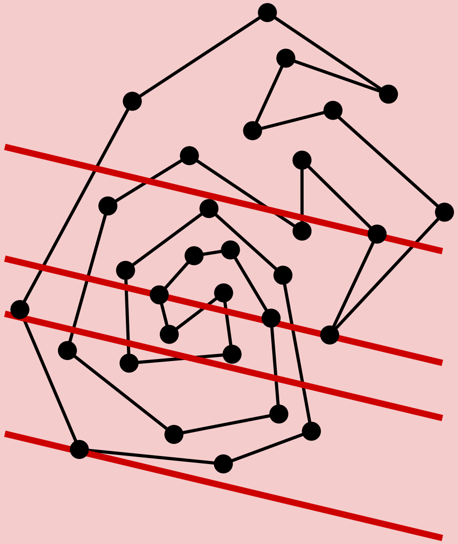
How did we prove the correctness of our solutions?



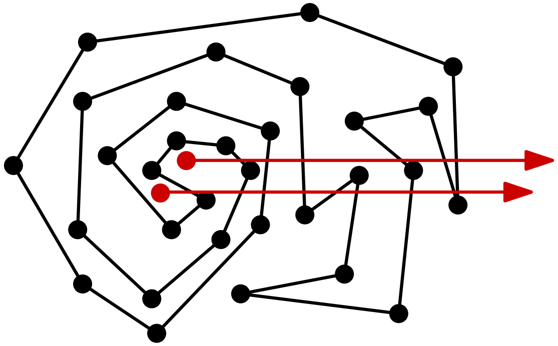


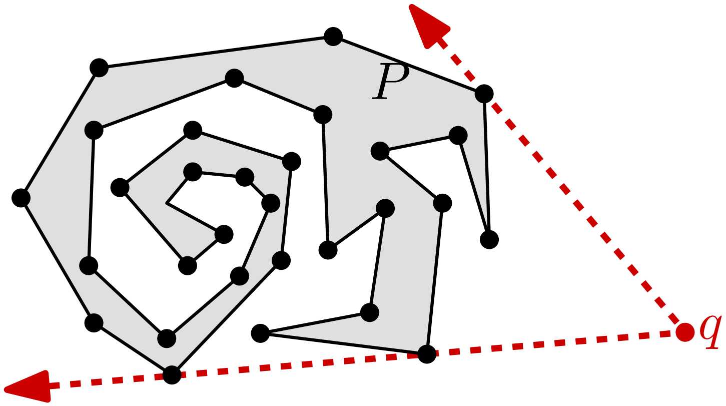


How did we prove the correctness of our solutions?

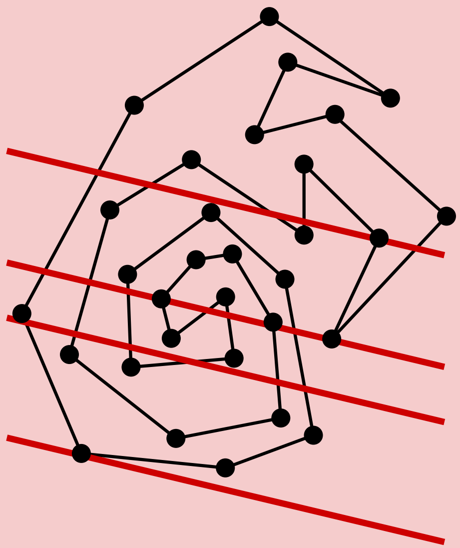


Geometric property:
No particular one.





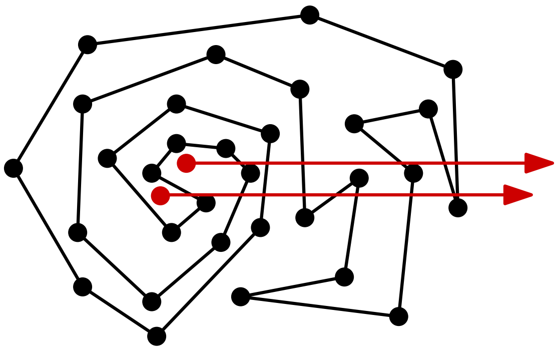
How did we prove the correctness of our solutions?

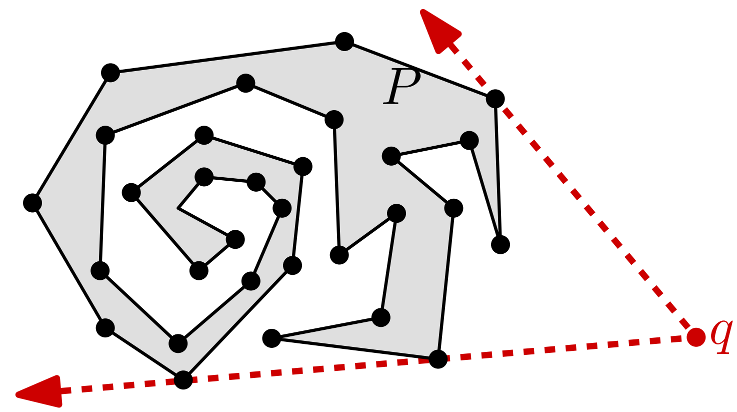


Geometric property:
No particular one.

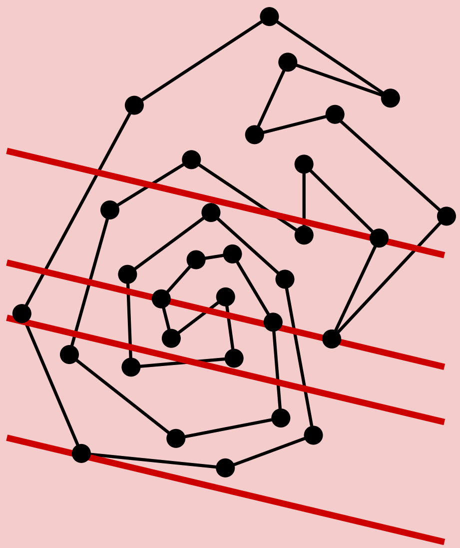
Brute-force solution

$O(n)$ time
$O(n)$ space





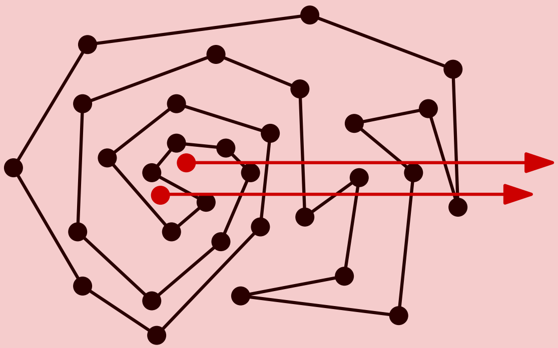
How did we prove the correctness of our solutions?



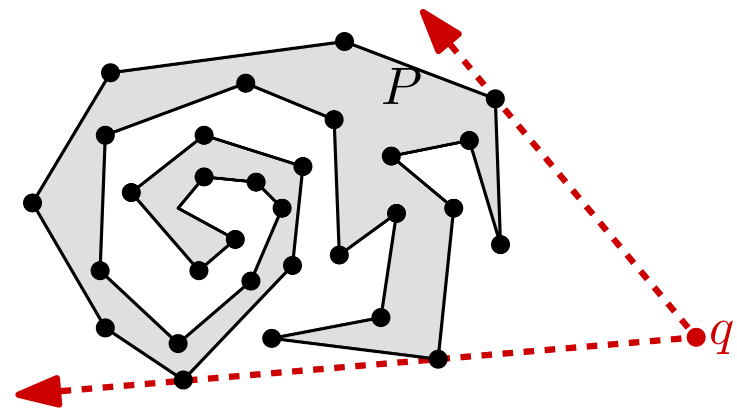
Geometric property:
No particular one.

Brute-force solution

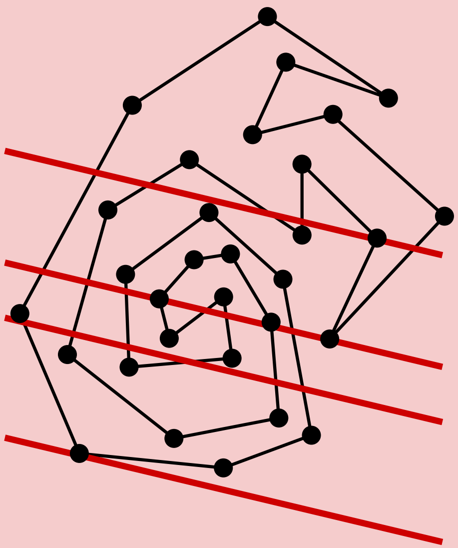
$O(n)$ time
$O(n)$ space



Geometric property:
$p \in P \Leftrightarrow$ The number of intersections of ∂P and any halfline with origin at p is odd.



How did we prove the correctness of our solutions?

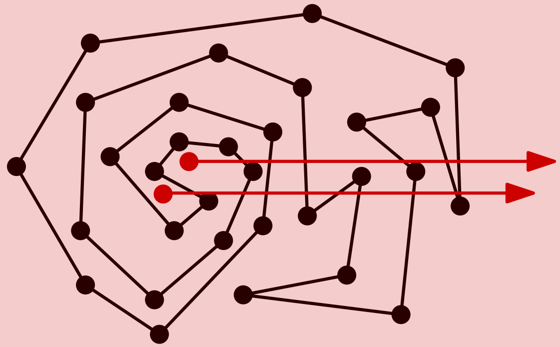


Geometric property:
No particular one.

⬇

Brute-force solution

$O(n)$ time
$O(n)$ space

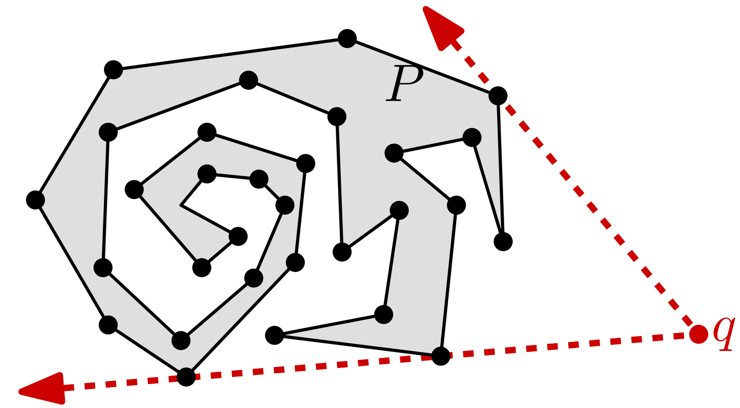


Geometric property:
$p \in P \Leftrightarrow$ The number of intersections of ∂P and any halfline with origin at p is odd.

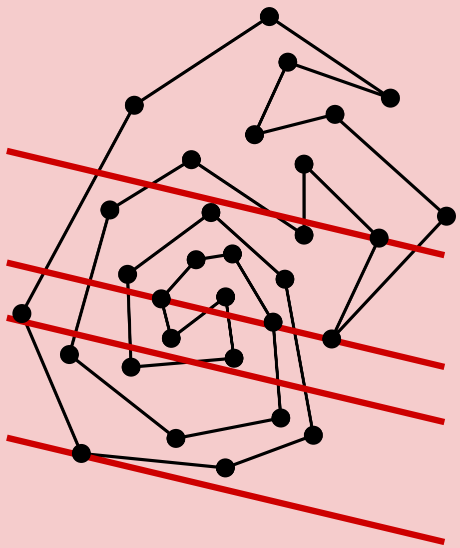
⬇

Brute-force solution

$O(n)$ time
$O(n)$ space



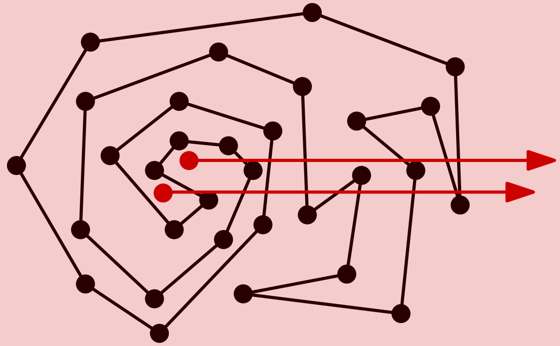
How did we prove the correctness of our solutions?



Geometric property:
No particular one.

Brute-force solution

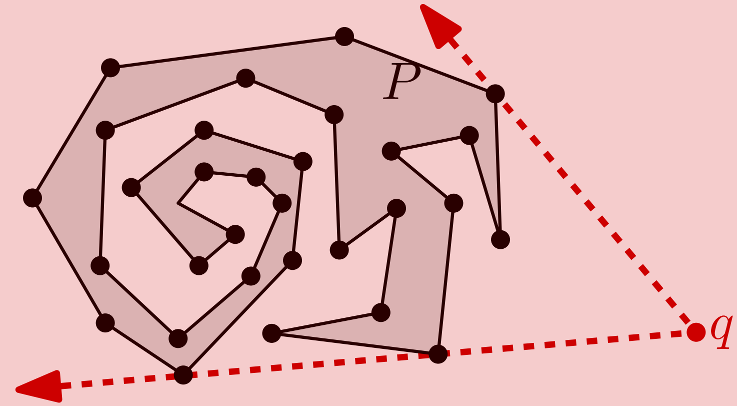
$O(n)$ time
$O(n)$ space



Geometric property:
$p \in P \Leftrightarrow$ The number of intersections of ∂P and any halfline with origin at p is odd.

Brute-force solution

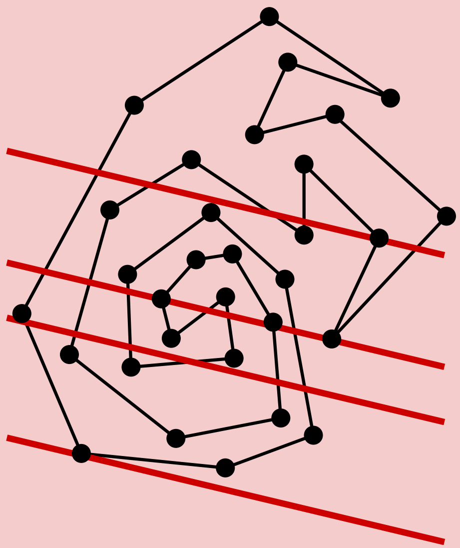
$O(n)$ time
$O(n)$ space



Geometric property:
The solutions are the angularly extreme vertices of P as seen from p.

USING ORIENTATION TESTS ON POLYGONS

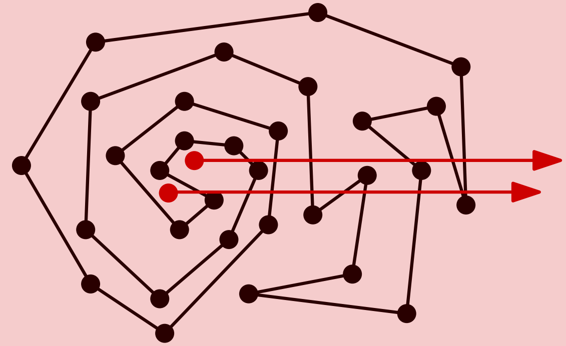
How did we prove the correctness of our solutions?



Geometric property:
No particular one.

Brute-force solution

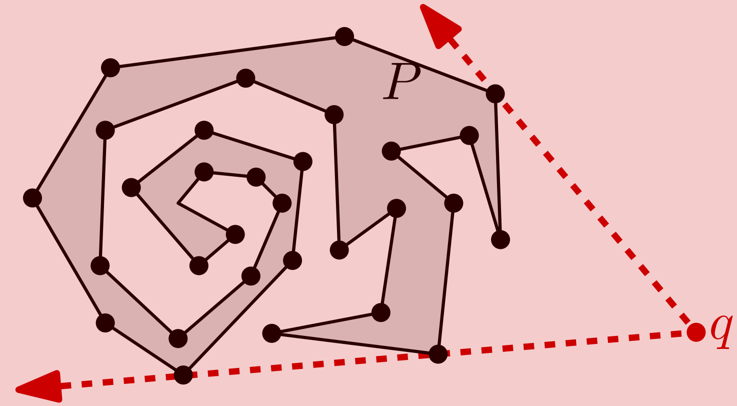
$O(n)$ time
$O(n)$ space



Geometric property:
$p \in P \Leftrightarrow$ The number of intersections of ∂P and any halfline with origin at p is odd.

Brute-force solution

$O(n)$ time
$O(n)$ space

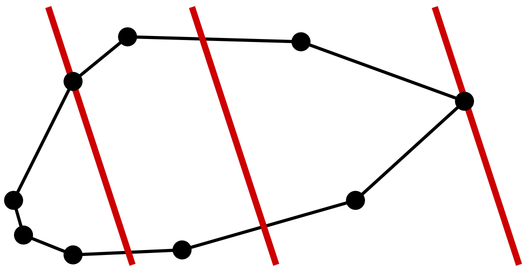


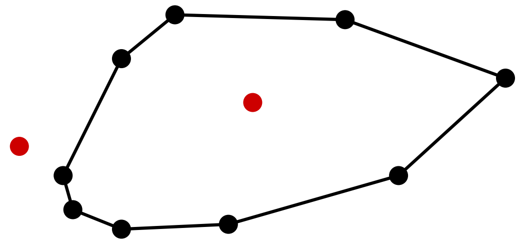
Geometric property:
The solutions are the angularly extreme vertices of P as seen from p.

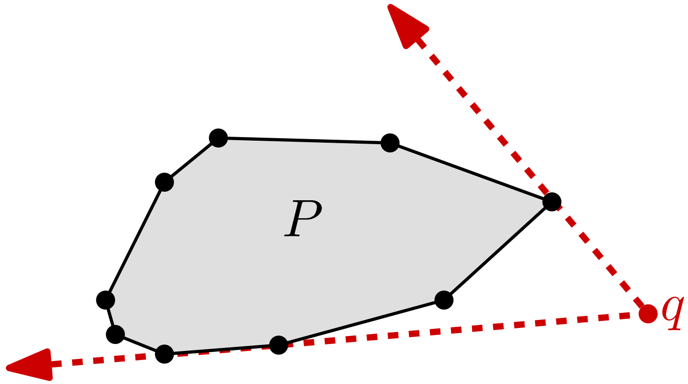
Use a max/min algorithm

$O(n)$ time
$O(n)$ space

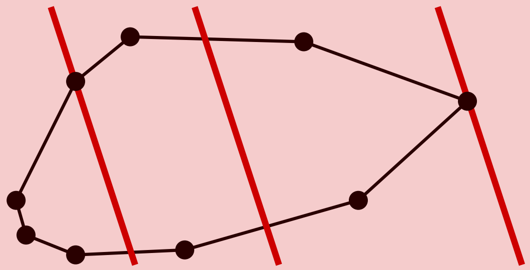
How did we prove the correctness of our solutions?

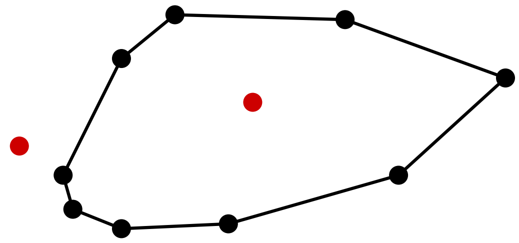


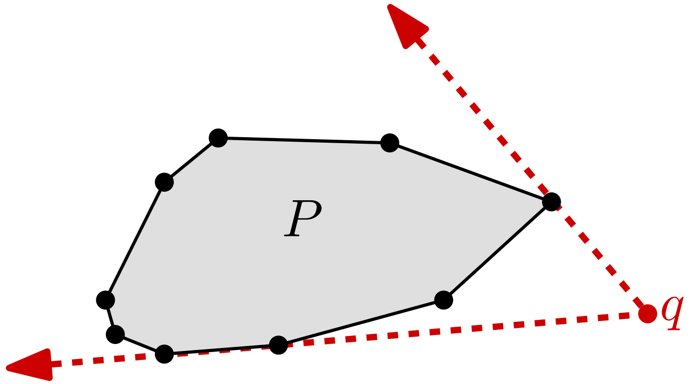




How did we prove the correctness of our solutions?

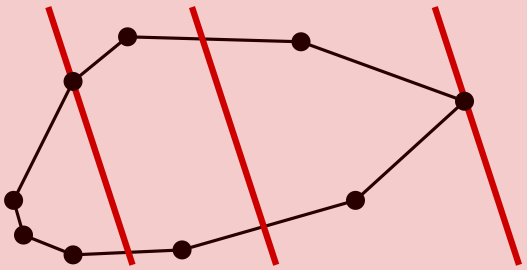






Geometric property: Distance to line is unimodal along each chain of ∂P.

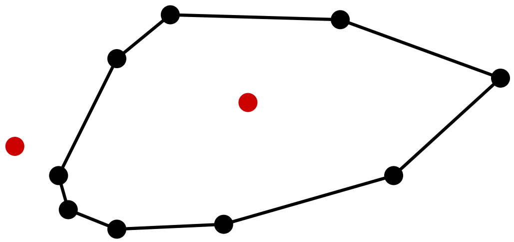
How did we prove the correctness of our solutions?

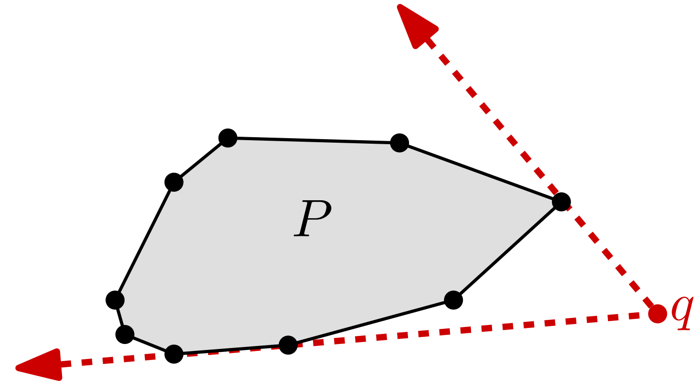


Geometric property:
Distance to line is unimodal along each chain of ∂P.

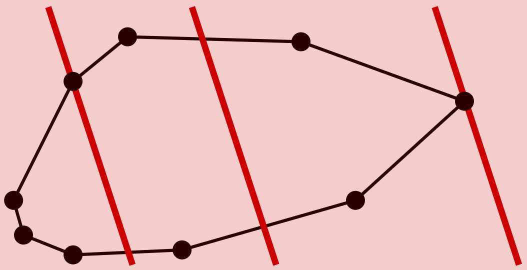
Binary search solution

$O(\log n)$ time
$O(n)$ space
(after preprocess)

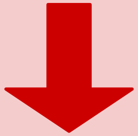




How did we prove the correctness of our solutions?

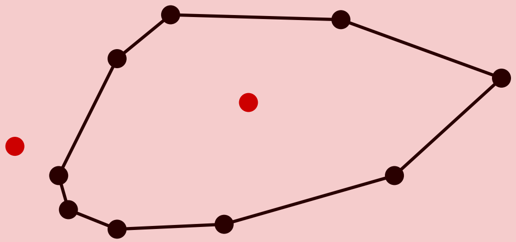


Geometric property: Distance to line is unimodal along each chain of ∂P.

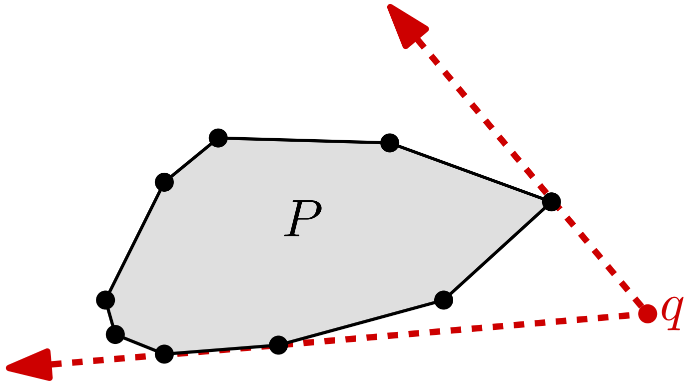


Binary search solution

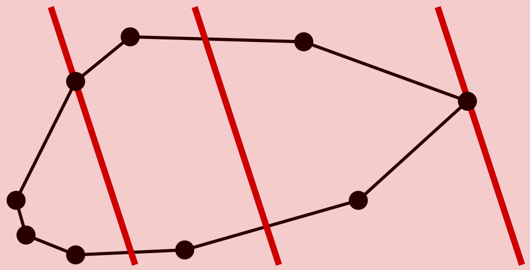
$O(\log n)$ time
$O(n)$ space
(after preprocess)



Geometric property: Segments connecting two vertices decompose P into two convex subpolygons.



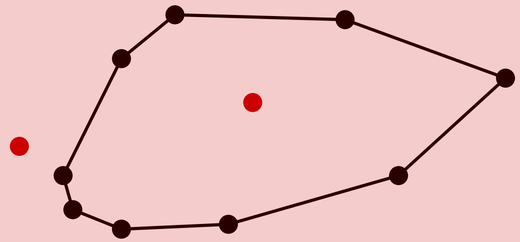
How did we prove the correctness of our solutions?



Geometric property:
Distance to line is unimodal along each chain of ∂P.

Binary search solution

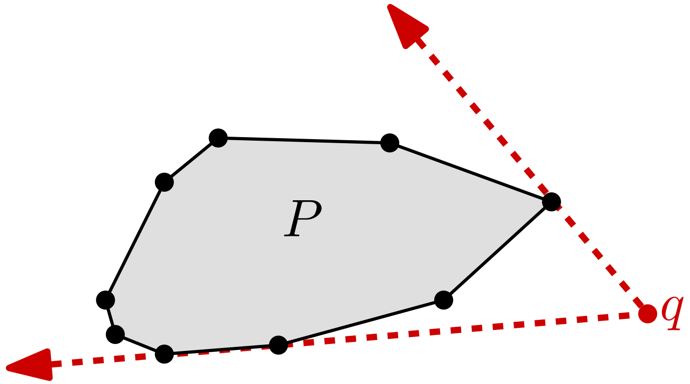
$O(\log n)$ time
$O(n)$ space
(after preprocess)



Geometric property:
Segments connecting two vertices decompose P into two convex subpolygons.

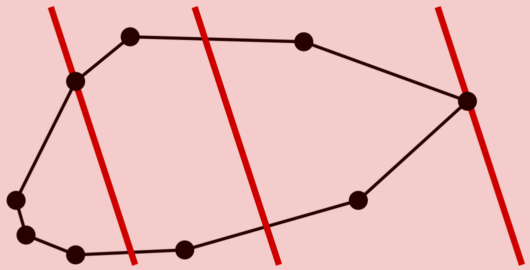
Binary search solution

$O(\log n)$ time
$O(n)$ space
(after preprocess)



USING ORIENTATION TESTS ON POLYGONS

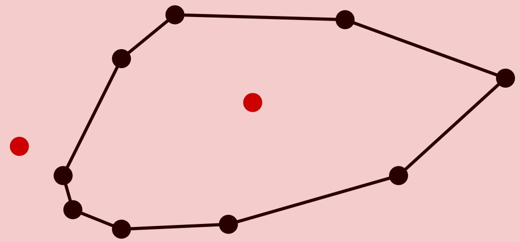
How did we prove the correctness of our solutions?



Geometric property: Distance to line is unimodal along each chain of ∂P.

Binary search solution

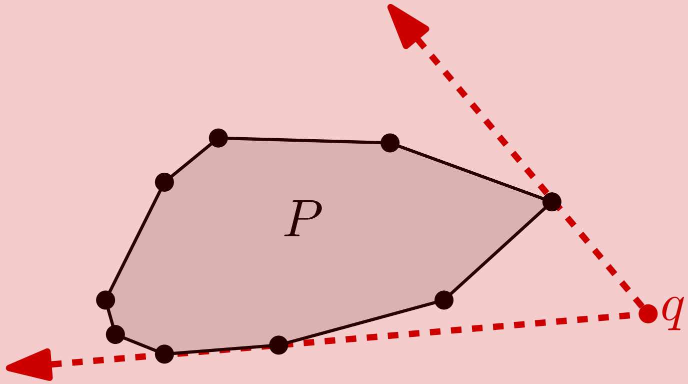
$O(\log n)$ time
$O(n)$ space
(after preprocess)



Geometric property: Segments connecting two vertices decompose P into two convex subpolygons.

Binary search solution

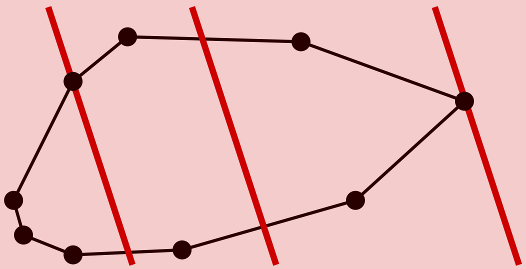
$O(\log n)$ time
$O(n)$ space
(after preprocess)



Geometric property: Angle wrt q is unimodal along ∂P.

USING ORIENTATION TESTS ON POLYGONS

How did we prove the correctness of our solutions?

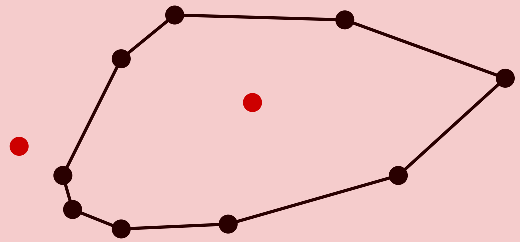


Geometric property: Distance to line is unimodal along each chain of ∂P.

⬇

Binary search solution

$O(\log n)$ time
$O(n)$ space
(after preprocess)

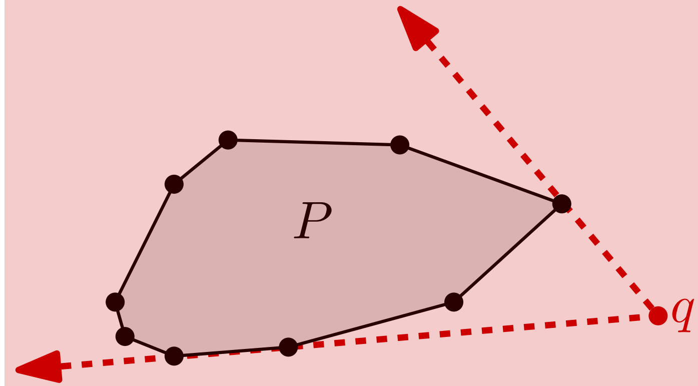


Geometric property: Segments connecting two vertices decompose P into two convex subpolygons.

⬇

Binary search solution

$O(\log n)$ time
$O(n)$ space
(after preprocess)



Geometric property: Angle wrt q is unimodal along ∂P.

⬇

Binary search solution

$O(\log n)$ time
$O(n)$ space
(after preprocess)

FURTHER READING

J. O'Rourke
Computational Geometry in C
Cambridge University Press, 1994 (2nd ed. 1998), pp. 17-35.

F. P. Preparata and M. I. Shamos
Computational Geometry: An Introduction
Springer-Verlag, 1985, pp. 36-45.