

# Point Visibility

**Arturo González Peñaloza**

**Dulce Julieta Mora Hernández**

Universidad Nacional Autónoma de México

16 de mayo de 2024

# 1. Introducción

- 1.1 Definiciones Fundamentales
- 1.2 Algoritmo para cierre convexo en  $O(n)$
- 1.3 Cierre Convexo en  $O(n)$
- 1.4 Punto en polígono

# 2. Calculando la visibilidad de un punto

# 3. Algoritmo para calcular $V(q)$

- 3.1 Algoritmo
- 3.2 Complejidad
- 3.3 Ejemplo

# Introducción

---

## Polígono de visibilidad

El *polígono de visibilidad*  $V(q)$  de un punto  $q$  en un polígono simple  $P$  es el conjunto de todos los puntos de  $P$  que son visibles desde  $q$ .

$$V(q) = \{p \in P \mid q \text{ sees } p\}$$

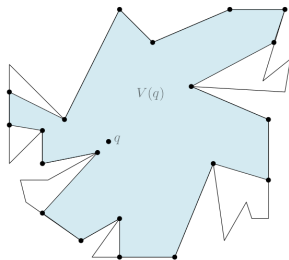


Figura: Polígono simple

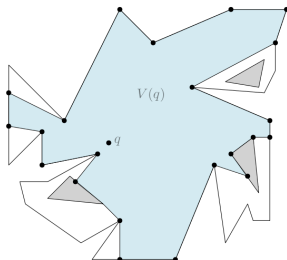


Figura: Polígono con hoyos

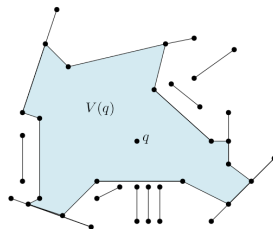


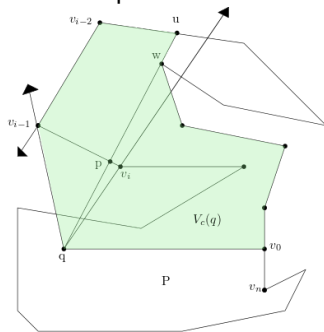
Figura: Conjunto de líneas

# Arista construida

Sea  $ab$  una arista en el perímetro de  $V(q)$  de manera que

- Ningún punto de  $ab$ , excepto  $a$  y  $b$ , pertenecen al perímetro de  $P$
- $q$ ,  $a$  y  $b$  son colineales
- $a$  o  $b$  es un vértice de  $P$

La arista  $ab$  se llama *arista construida* de  $V(q)$

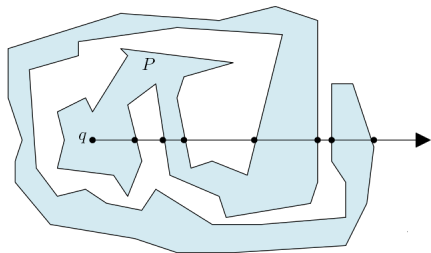


# Revoluciones

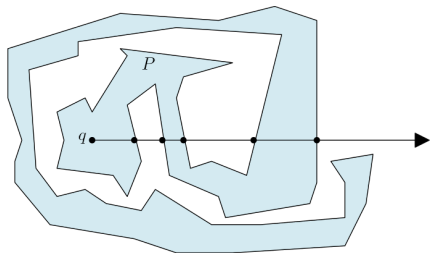
Para un polígono simple  $P$  y un punto  $z \in P$ , el *número de revoluciones* de  $P$  con respecto a  $z$  es el número de revoluciones que el perímetro de  $P$  hace alrededor de  $z$ .

Si el número de revoluciones de  $P$  respecto a  $z$  es uno,  $P$  es llamado *non-winding polygon*.

El número de revoluciones de  $P$  respecto a  $q$  es dos



El número de revoluciones de  $P$  respecto a  $q$  es una





## Algoritmo para calcular CH en $O(n)$

1.  $t \leftarrow -1$ ;  $b \leftarrow 0$ ;  
   $v_1 \leftarrow \text{input}$ ;  $v_2 \leftarrow \text{input}$ ;  $v_3 \leftarrow \text{input}$ ;  
  if  $(v_1, v_2, v_3) > 0$   
    then begin push  $v_1$ ; push  $v_2$  ; end  
    else begin push  $v_2$ ; push  $v_1$  ; end  
  push  $v_3$  ; insert  $v_3$ ;
2.  $v \leftarrow \text{input}$ ;  
  until  $(v, d_b, d_{b+1}) < 0$  or  $(d_{t-1}, d_t, v) < 0$   
    do  $v \leftarrow \text{input}$  end;
3. until  $(d_{t-1}, d_t, v) > 0$  do pop  $d_t$  end;  
  push  $v$ ;
4. until  $(v, d_b, d_{b+1}) > 0$  do remove  $d_b$  end;  
  insert  $v$ ;  
  goto 2

# Algoritmo de Melkman

Los polígonos simples simplifican el proceso de calcular su cierre convexo. El algoritmo de Melkman es una herramienta que aprovecha esta propiedad.

# Algoritmo de Melkman

1.  $D \leftarrow (p_2, p_1, p_1)$  // Se agregan a una deque dos puntos consecutivos del polígono.
2. Para  $i \leftarrow 3$  a  $n$  hacer:
  - 2.1 Si  $p_i$  está fuera del ángulo  $v_{t-1}v_tv_{b+1}$ , entonces
    - 2.1.1 Mientras  $p_i$  esté a la izquierda de  $\overrightarrow{v_b v_{b+1}}$ , entonces se saca desde abajo de  $D$ .
    - 2.1.2 Mientras  $p_i$  esté a la derecha de  $\overrightarrow{v_t v_{t-1}}$ , entonces se saca desde arriba de  $D$ .
  - 2.2 Se agrega a  $p_i$ . al inicio y al final de  $D$ .

## Observación

Si tomamos los primeros tres vértices de la deque (leídos de izquierda a derecha), obtenemos una vuelta a la derecha. Si tomamos los últimos tres vértices de la deque (leídos de derecha a izquierda), obtenemos una vuelta a la izquierda.

# Ejemplo

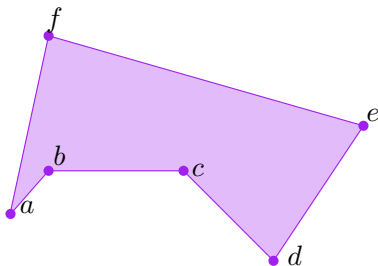


Figura: Polígono simple

# Ejemplo

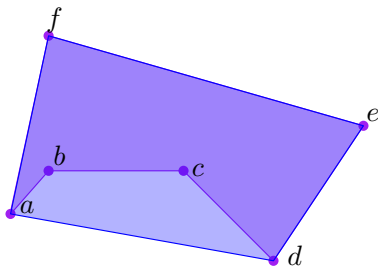
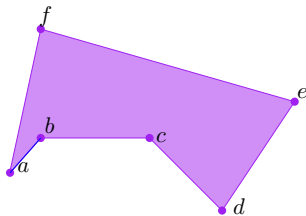


Figura: Polígono simple con su cierre convexo

## Ejemplo

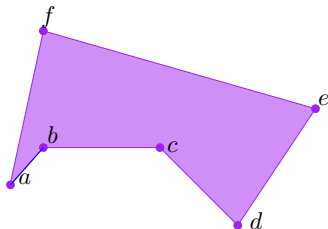
Se agrega a la deque dos puntos consecutivos del polígono.



$a$	$b$	$a$
-----	-----	-----

## Ejemplo

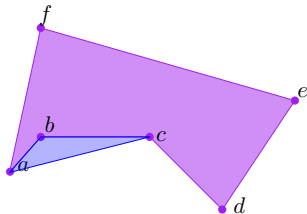
$c$  está a la derecha de  $\overrightarrow{ab}$ , entonces se saca desde abajo de  $D$ .



$a$	$b$
-----	-----

## Ejemplo

Se agrega a  $c$  al inicio y al final de la deque.

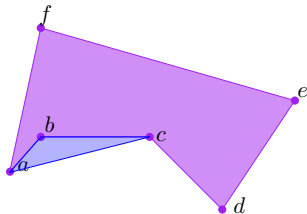


$c$	$a$	$b$	$c$
-----	-----	-----	-----



## Ejemplo

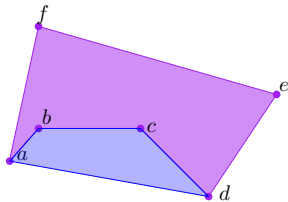
$d$  está a la izquierda de  $\overrightarrow{ca}$ , entonces se saca desde abajo de la deque.



$a$	$b$	$c$
-----	-----	-----

## Ejemplo

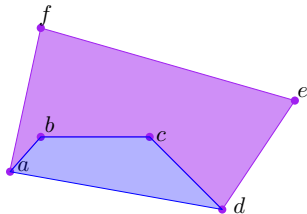
Se agrega a  $d$  al inicio y al final de la deque.



$d$	$a$	$b$	$c$	$d$
-----	-----	-----	-----	-----

## Ejemplo

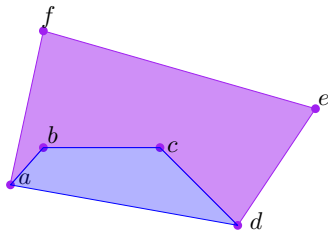
e se encuentra a la derecha de  $\overrightarrow{dc}$ , entonces se saca desde arriba de la deque.



$d$	$a$	$b$	$c$
-----	-----	-----	-----

## Ejemplo

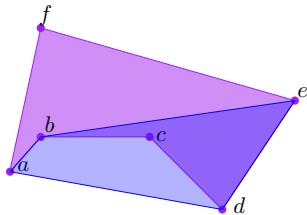
e se encuentra a la derecha de  $\overrightarrow{cb}$ , entonces se saca desde arriba de la deque.



$d$	$a$	$b$
-----	-----	-----

## Ejemplo

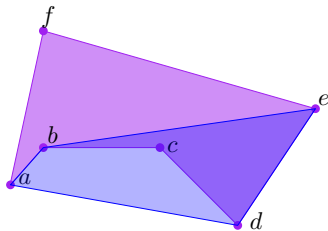
Se agrega a  $e$  al inicio y al final de la deque.



$e$	$d$	$a$	$b$	$e$
-----	-----	-----	-----	-----

## Ejemplo

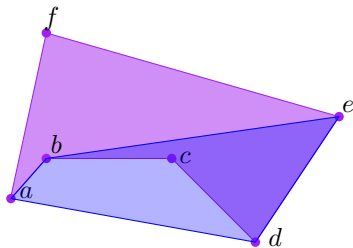
$f$  está a la derecha de  $\overrightarrow{eb}$ , así que se saca desde arriba de la deque.



$e$	$d$	$a$	$b$
-----	-----	-----	-----

## Ejemplo

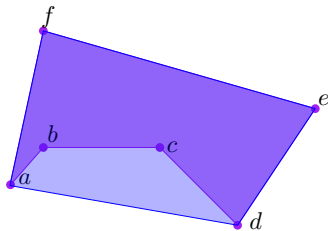
$f$  está a la derecha de  $\overrightarrow{ba}$ , así que se saca desde arriba de la deque.



$e$	$d$	$a$
-----	-----	-----

## Ejemplo

Se agrega a  $f$  al inicio y al final de la deque.



$f$	$e$	$d$	$a$	$f$
-----	-----	-----	-----	-----



# Complejidad

Si el polígono tiene  $n$  vértices, entonces se realizan a lo más  $2n$  inserciones y  $2n - 3$  eliminaciones. Así, el algoritmo es  $O(n)$ .

## Punto en polígono

Un problema geométrico es determinar si un punto específico está dentro o fuera de un polígono dado.

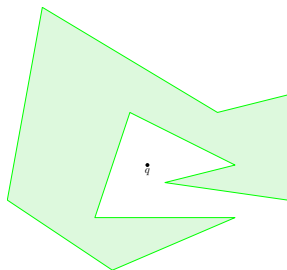


Figura: Punto  $q$  fuera de polígono.

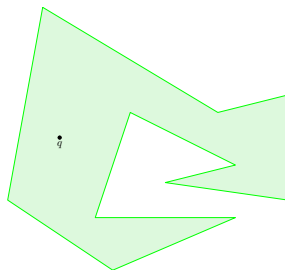
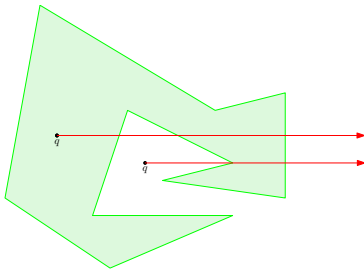


Figura: Punto  $q$  dentro de polígono.

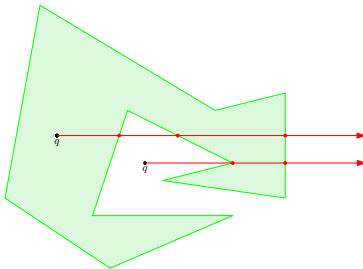
# Método del rayo

1. Se traza una línea horizontal desde el punto hacia la derecha a lo largo del eje x.



# Método del rayo

1. Se traza una línea horizontal desde el punto hacia la derecha a lo largo del eje x.
2. Se cuenta el número de intersecciones con las aristas del polígono:
  - 2.1 Si el número de intersecciones es par, entonces el punto está afuera del polígono.
  - 2.2 En otro caso, el punto está dentro del polígono.



# Complejidad

1. Se traza una línea horizontal desde el punto hacia la derecha a lo largo del eje x.  
 **$O(1)$**
2. Se cuenta el número de intersecciones con las aristas del polígono:  **$O(n)$** 
  - 2.1 Si el número de intersecciones es par, entonces el punto está afuera del polígono.  
 **$O(1)$**
  - 2.2 En otro caso, el punto está dentro del polígono.  **$O(1)$**

**Complejidad:**  $O(n)$

## Calculando la visibilidad de un punto

---

# El problema

**Non-winding polygon:  $O(n)$  algorithm**

El primer paso del algoritmo es determinar si  $q$  se encuentra dentro o fuera de  $P$ .

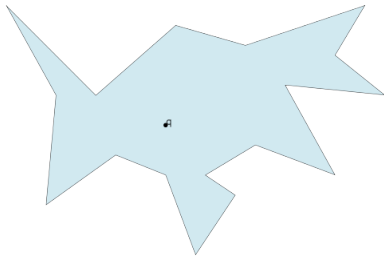


Figura:  $q$  se encuentra dentro de  $P$

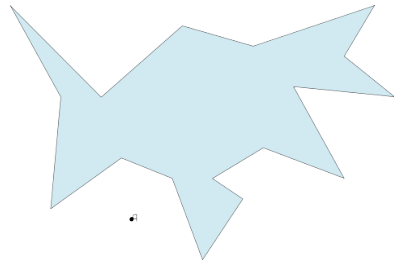


Figura:  $q$  se encuentra fuera de  $P$



Existen dos situaciones:

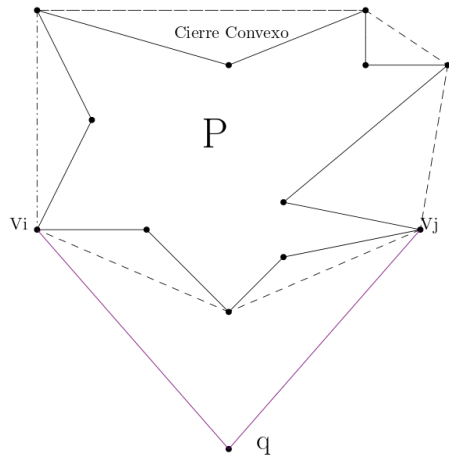
- $q$  se encuentra fuera del cierre convexo de  $P$
- $q$  se encuentra fuera de  $P$  pero dentro del cierre convexo de  $P$

## Si $q$ se encuentra fuera del cierre convexo de $P$

1. Trazamos dos tangentes (digamos,  $qv_i$  y  $qv_j$ ) a partir de  $q$  hacia el cierre convexo de  $P$ .
2. Ahora  $q$  es un punto interno de  $P'$ .

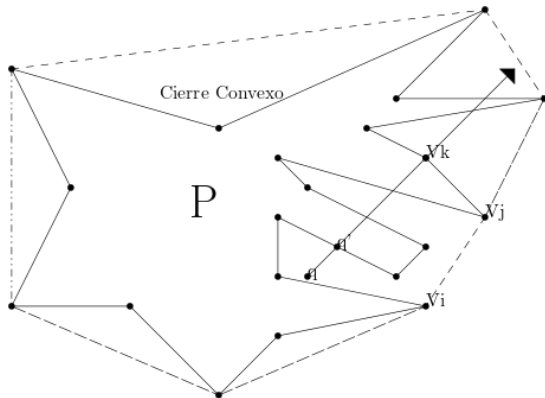
### Observación

Sea  $bd(P)$  el perímetro de  $P$ . Notemos que todos los puntos visibles del  $bd(P)$  a partir de  $q$  se encuentran entre  $v_i$  y  $v_j$  viendo hacia  $q$ .



## Si $q$ se encuentra fuera de $P$ pero dentro del cierre convexo de $P$

1. Trazamos una línea a partir de  $q$  que pase por cualquier vértice  $v_k$  de  $P$  (denotado como  $\overrightarrow{qv_k}$ ).
2. Sea  $q'$  el punto más cercano a  $q$  entre todos los puntos de las intersecciones de  $\overrightarrow{qv_k}$  con  $bd(P)$ .
3. A partir de  $q'$  recorremos  $bd(P)$  en el sentido de las manecillas del reloj (y en sentido contrario) hasta que un vértice  $v_i$  del cierre convexo (respectivamente,  $v_j$ ) se alcanza.
4. Ahora,  $q$  es un punto dentro de  $P'$



A partir de ahora, se considera que el punto  $q$  es un punto interno de  $P$ . Por lo que, de ahora en adelante, se asume que  $bd(P)$  no tiene *winding* alrededor de  $q$ .

**El problema es calcular  $V(q)$  de  $P$  de  $q$ .**

## Observación

Sea  $bd(v_j, v_k)$  el límite en sentido antihorario de  $P$  desde  $v_j$  hasta  $v_k$ .

También asumimos que los vértices (y los puntos finales de las *aristas construidas*) en  $bd(v_0 v_{i-1})$ , las cuales se encuentran para ser visibles desde  $q$  por el procedimiento, son colocadas en un stack en el orden en que son encontradas, donde  $v_0$  y  $v_{i-1}$  están en la parte inferior y superior del stack, respectivamente.

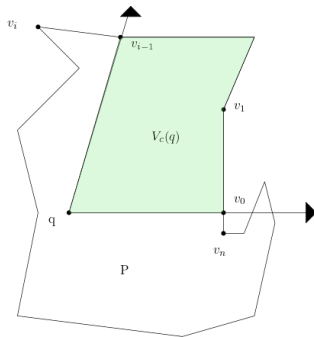
Contamos con los siguientes casos

1. El vértice  $v_i$  se encuentra a la izquierda de  $\overrightarrow{qv_{i-1}}$
2. El vértice  $v_i$  se encuentra a la derecha de  $\overrightarrow{qv_{i-1}}$ 
  - 2.1 El vértice  $v_i$  se encuentra a la derecha de  $\overrightarrow{v_{i-2}v_{i-1}}$
  - 2.2 El vértice  $v_i$  se encuentra a la izquierda de  $\overrightarrow{v_{i-2}v_{i-1}}$

## Caso 1

El vértice  $v_i$  se encuentra a la izquierda de  $\overrightarrow{qv_{i-1}}$

Como  $v_i$  y los vértices y puntos en el stack se encuentran ordenados por el ordenamiento angular respecto a  $q$ ,  $v_i$  es ingresado al stack.







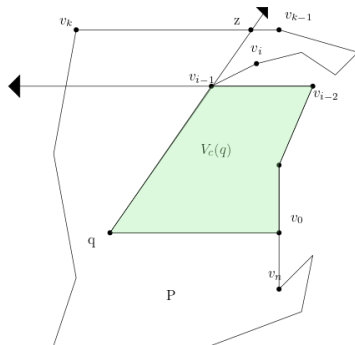
## Caso 2a

El vértice  $v_i$  se encuentra a la derecha de  $\overrightarrow{v_{i-2}v_{i-1}}$

El vértice  $v_i$  y algunos de los vértices subsecuentes de  $v_i$  (que serán revisados) no son visibles desde  $q$ .

Sea  $v_{k-1}v_k$  la primer arista desde  $v_{i+1}$  en  $bd(v_{i+1}, v_n)$ , en sentido antihorario de manera que  $v_{k+1}v_k$  intersecta  $\overrightarrow{qv_{i-1}}$ .

Sea  $z$  el punto de intersección.



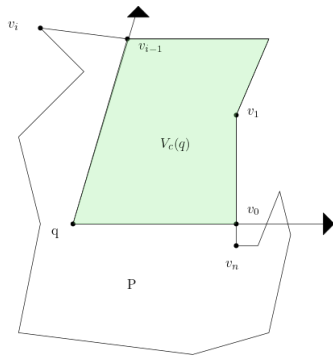


## Caso 2b

El vértice  $v_i$  se encuentra a la izquierda de  $\overrightarrow{v_{i-2}v_{i-1}}$

El vértice  $v_{i-1}$  y algunos de los vértices anteriores de  $v_i$  (quien está actualmente en el stack) no son visibles desde  $q$ . Sacamos a  $v_i$  del stack.

Sea  $u$  vértice que se encuentra en la parte superior del stack. La arista  $v_{i-1}v_i$  es conocida como *arista frontal*. Mientras  $v_{i-1}v_i$  intersecta  $uq$  y  $u$  es un vértice de  $P$ , realizamos pop al stack.



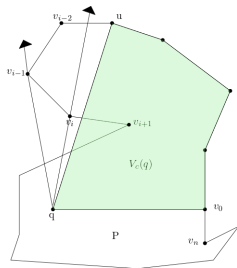
Después de ejecutar el backtracking, pueden suceder dos situaciones

- i.  $v_{i-1}v_i$  no interseca  $uq$
- ii.  $v_{i-1}v_i$  interseca  $uq$

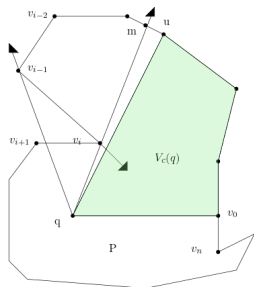
## Caso 2b.i

$v_{i-1}v_i$  **no intersecta**  $uq$

Si  $v_{i+1}$  se encuentra a la derecha de  $\overrightarrow{qv_i}$ , el backtracking continua con  $v_iv_{i+1}$  como la *arista frontal* actual.



De otra forma,  $v_{i+1}$  se encuentra a la izquierda de  $\overrightarrow{qv_i}$ .

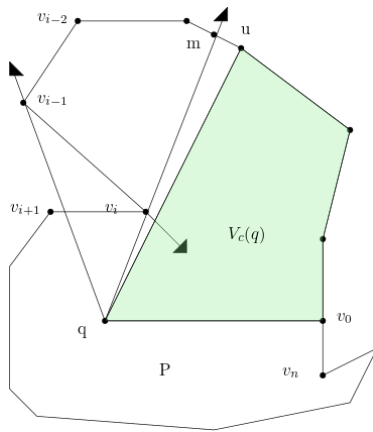


## Caso 2b.i

$v_{i-1}v_i$  **no interseca**  $uq$

Sea  $m$  el punto de intersección de  $\overrightarrow{qv_i}$  con la arista del polígono que contiene  $u$ .

Si  $v_{i+1}$  se encuentra a la derecha de  $\overrightarrow{v_{i-1}v_i}$ , entonces termina el backtracking.

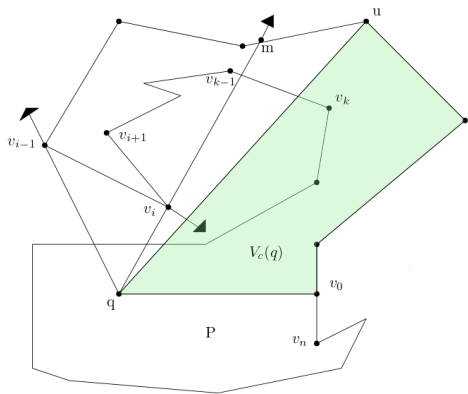


## Caso 2b.i

$v_{i-1}v_i$  **no interseca**  $uq$

Ingresamos  $m$  y  $v_{i+1}$  al stack y  $v_{i+1}$  se convierte en el nuevo  $v_i$ .

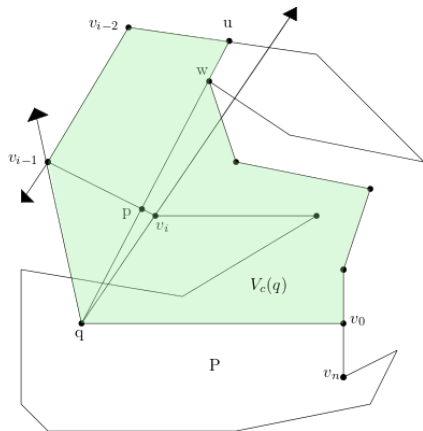
Si  $v_{i+1}$  se encuentra a la izquierda de  $\overrightarrow{v_{i-1}v_i}$ , revisamos  $bd(v_{i+1}, v_n)$  desde  $v_{i+1}$  hasta que un vértice  $v_k$  es encontrado de manera que la arista  $v_{k-1}v_k$  interseca  $mv_i$ . El backtracking continua con  $v_{k-1}v_k$  como la *arista frontal* actual.



## Caso 2b.ii

$v_{i-1}v_i$  **intersecta**  $uq$

$u$  no es un vértice de  $P$ . Sea  $w$  el vértice que se encuentra justo debajo de  $u$  en el stack. Por lo que,  $uw$  es una *arista construida*



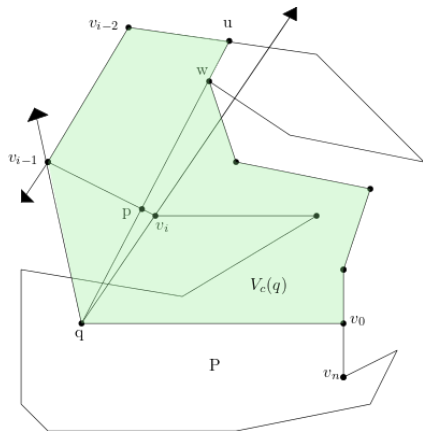


## Caso 2b.ii

$v_{i-1}v_i$  **intersecta**  $uq$

Sea  $p$  el punto de intersección de  $uq$  y  $v_{i-1}v_i$ . Si  $p \in qw$ , la visibilidad de ambos,  $u$  y  $w$  desde  $q$  esta bloqueada por  $v_{i-1}v_i$ . Vaciamos el stack.

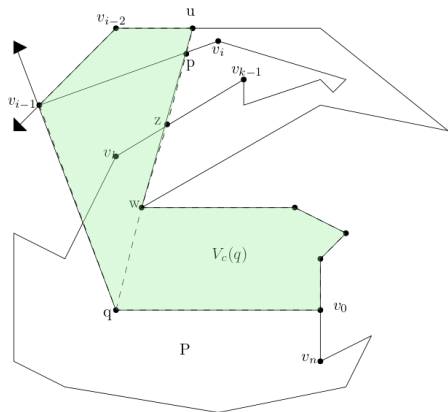
El backtracking continua y  $v_{i-1}v_i$  permanece como la *arista frontal*.



$V_{j-1} V_j$  intersecta  $uq$

De otra forma,  $v_{i-1}v_i$  ha intersectado  $uw$  como  $p$  pertenece a  $uw$ .

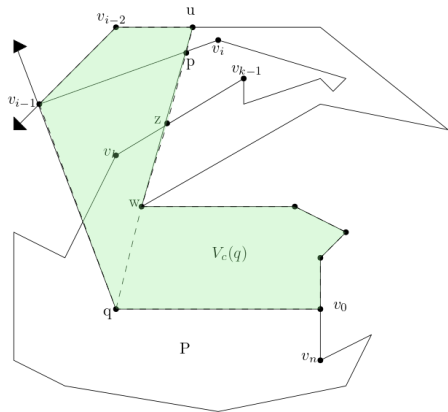
Checamos  $bd(v_{i+1}, v_n)$  desde  $v_{i+1}$  hasta encontrar un vértice  $v_k$  tal que la arista  $v_{k-1}v_k$  ha sido intersectada por  $w$  en algún punto (digamos,  $z$ ). Así que, todo  $bd(w, z)$  (a excepción de  $w$  y  $z$ ) no es visible por  $q$ . Vaciamos el stack.



## Caso 2b.ii

$v_{i-1}v_i$  **intersecta**  $uq$

Ingresamos a  $z$  y a  $v_k$  al stack. Por lo que,  $v_{k+1}$  se convierten en el nuevo  $v_i$ . Puede suceder que la *arista construida* que termina en  $u$  (digamos,  $uu'$ ) haya sido calculada por el Caso 2b al final de la fase de backtracking. Esto significa que el vértice  $u'$  es el último vértice en el stack que va a ser sacado en la fase de backtracking actual. Por lo tanto,  $q$ ,  $w$  y  $z$  no son colineales. Los sacamos del stack y notemos que nos encontramos la primera situación del backtracking actual.



## Algoritmo para calcular $V(q)$

---

## Algoritmo para calcular $V(q)$

1. Ingresamos a  $v_1$  al stack y  $i := i + 1$ . Si  $i = n + 1$  *Ir al Paso 8*.
2. Si  $v_i$  se encuentra a la izquierda de  $\overrightarrow{qv_{i-1}}$  entonces *Ir al Paso 1*
3. Si  $v_i$  se encuentra a la derecha de  $\overrightarrow{qv_{i-1}}$  y  $\overrightarrow{v_{i-2}v_{i-1}}$  entonces
  - 3.1 Checar desde  $v_{i+1}$  en sentido antihorario hasta encontrar un vértice  $v_k$  tal que  $v_{k-1}v_k$  intersecta  $\overrightarrow{qv_{i-1}}$ . Sea  $z$  el punto de intersección.
  - 3.2 Ingresamos  $z$  al stack.  $i := k$  e *Ir al Paso 1*.
4. Si  $v_i$  se encuentra a la derecha de  $\overrightarrow{qv_{i-1}}$  y a la izquierda de  $\overrightarrow{v_{i-2}v_{i-1}}$  entonces
  - 4.1 Sea  $u$  el elemento que se encuentra en la parte superior del stack. Realizamos *pop* al stack.
  - 4.2 Mientras  $u$  sea un vértice y  $v_{i-1}v_i$  intersecte  $uq$ , realizamos *pop* al stack.

5. Si  $v_{i-1}v_i$  no intersecta  $uq$  entonces
  - 5.1 Si  $v_{i+1}$  se encuentra a la derecha de  $\overrightarrow{qv_i}$  entonces  $i := i + 1$  e *Ir al Paso 4b.*
  - 5.2 Sea  $m$  el punto de intersección de  $\overrightarrow{qv_i}$  y la arista que contiene a  $u$ . Si  $v_{i+1}$  se encuentra a la derecha de  $\overrightarrow{v_{i-1}v_i}$  entonces ingresamos  $m$  al stack y *vamos al Paso 1.*
  - 5.3 Checamos desde  $v_{i+1}$  en orden antihorario hasta encontrar un vértice  $v_k$  tal que  $v_{k-1}v_k$  intersecte  $mv_i$ . Asignamos  $k$  a  $i$  y *vamos al Paso 4b.*
6. Sea  $w$  el vértice que se encuentra justo debajo de  $u$  en el stack. Sea  $p$  el punto de intersección entre  $v_{i-1}v_i$  y  $uq$ . Si  $p \in qw$  o  $q, w$  y  $u$  no son colineales entonces realizamos *pop* al stack y *vamos al paso 4b.*
7. Checamos desde  $v_{i+1}$  en sentido antihorario hasta encontrar un vértice  $v_k$  tal que  $v_{k-1}v_k$  intersecte  $wp$ . Insertamos el punto de intersección al stack, asignamos  $k$  a  $i$  y *vamos al Paso 1.*
8. Generamos  $V(q)$  sacando todos los vértices y puntos del stack y nos detenemos.

# Ejemplo

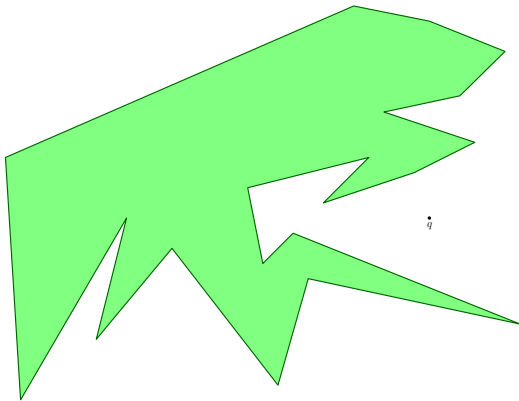
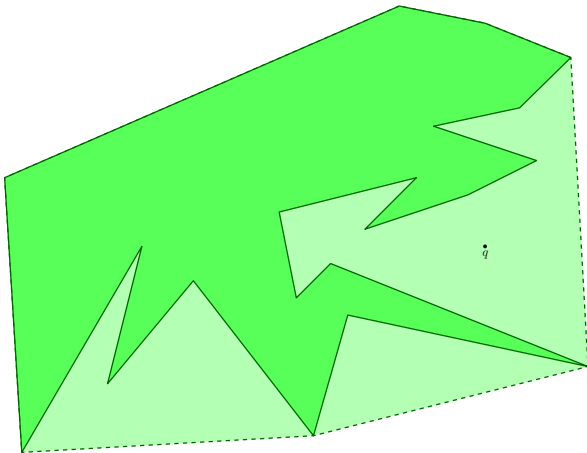


Figura: Polígono con punto  $q$

## Ejemplo

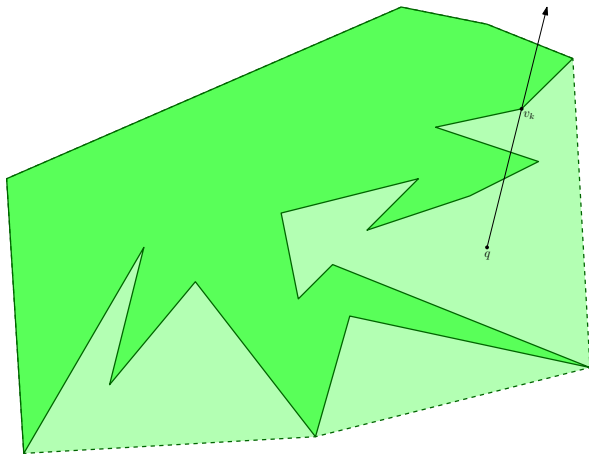
Calculamos el cierre convexo.





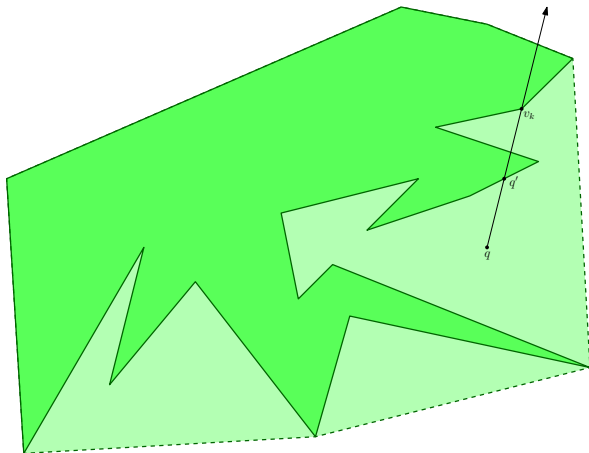
## Ejemplo

Trazamos una línea a partir de  $q$  que pase por un vértice de  $P$ .



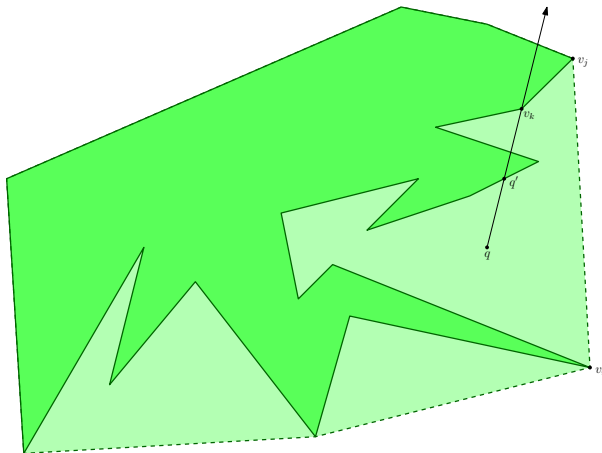
## Ejemplo

Encontramos al punto más cercano a  $q$  de las intersecciones de  $\overrightarrow{qv_k}$  con  $bd(P)$ .



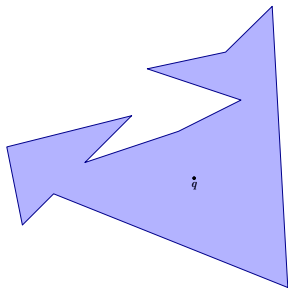
# Ejemplo

Encontramos a  $v_i$  y  $v_j$ .



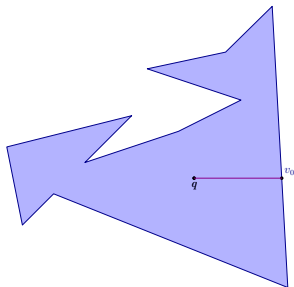
## Ejemplo

Ahora  $q$  es un punto interno del polígono conseguido.



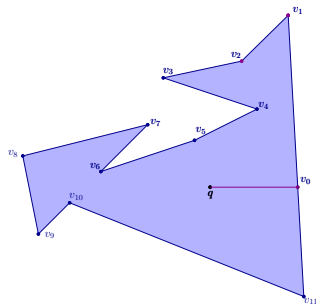
# Ejemplo

TODO



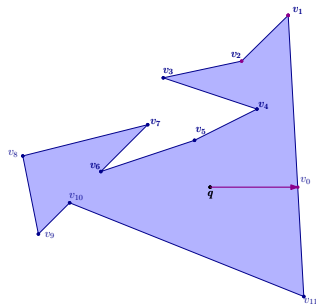
# Ejemplo

TODO



# Ejemplo

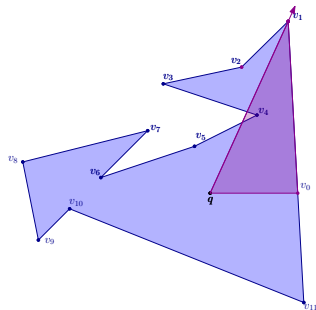
TODO



$v_0$

# Ejemplo

TODO

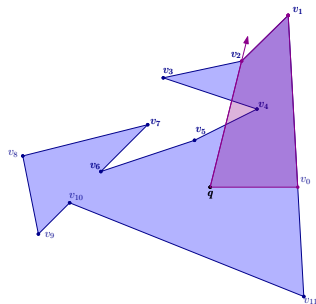


$v_1$
$v_0$



# Ejemplo

TODO

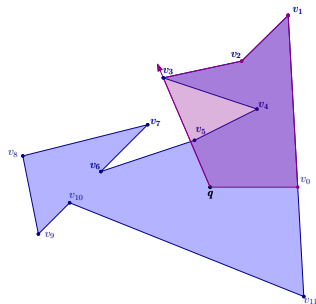


$v_2$
$v_1$
$v_0$

# Ejemplo

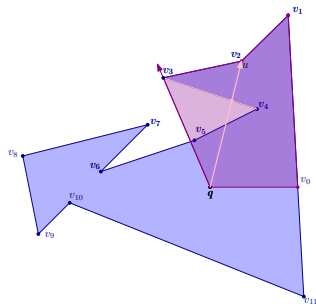
TODO

$v_3$
$v_2$
$v_1$
$v_0$



# Ejemplo

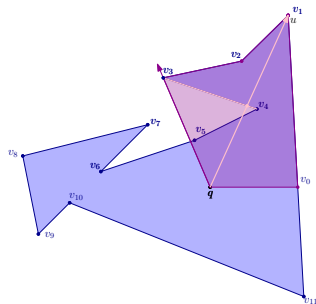
TODO



$v_2$
$v_1$
$v_0$

# Ejemplo

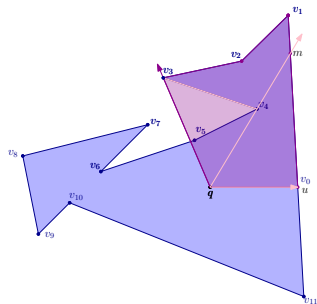
TODO



$v_1$
$v_0$

# Ejemplo

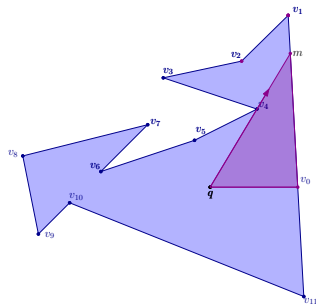
TODO



$v_0$

# Ejemplo

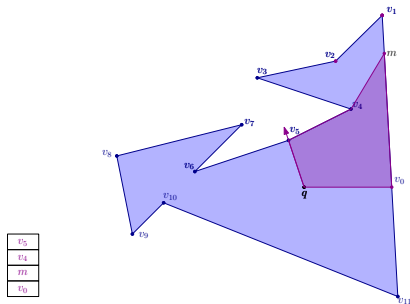
TODO



$v_4$
$m$
$v_0$

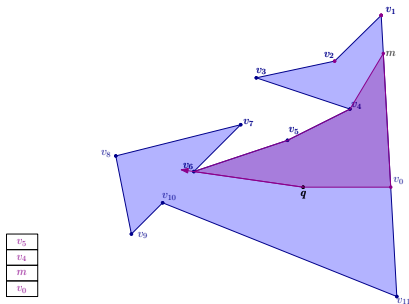
# Ejemplo

TODO



# Ejemplo

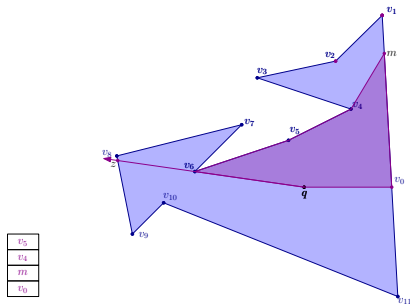
TODO





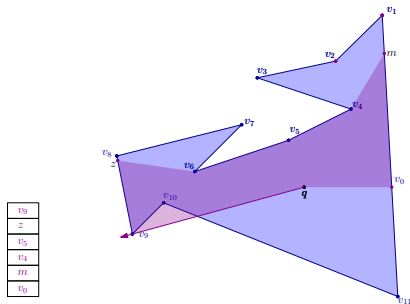
# Ejemplo

TODO



# Ejemplo

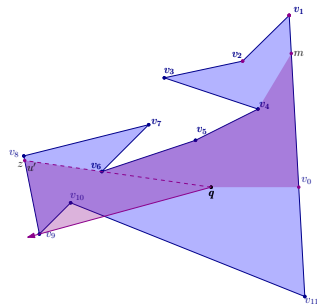
TODO



# Ejemplo

TODO

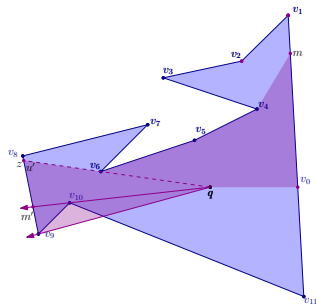
$z$
$v_5$
$v_4$
$m$
$v_0$



# Ejemplo

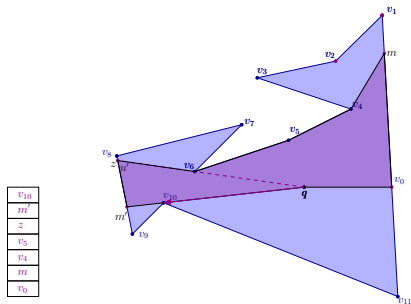
TODO

$z$
$v_5$
$v_4$
$m$
$v_0$



## Ejemplo

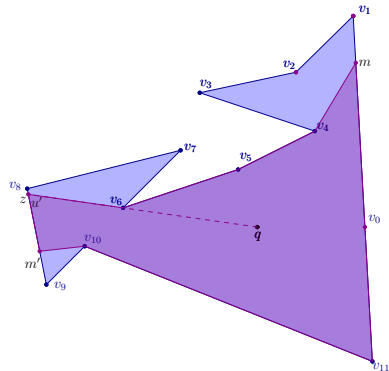
TODO



# Ejemplo

TODO fdsaf dfs asf sad  
 fas fas fsad TODO fdsaf  
 dfs asf sad fas fas  
 fsadTODO fdsaf dfs asf  
 sad fas fas fsadTODO  
 fdsaf dfs asf sad fas fas  
 fsadTODO fdsaf dfs asf  
 sad fas fas fsadTODO  
 fdsaf dfs asf sad fas fas  
 fsadTODO fdsaf dfs asf  
 sad fas fas fsad

$v_{11}$
$v_{10}$
$m'$
$z$
$v_5$
$v_4$
$m$
$v_0$



# Complejidad del algoritmo

## Invariante

El algoritmo mantiene una invariante de que los vértices y puntos en la pila en cualquier etapa están ordenados angularmente con respecto a  $q$ .

Revisemos paso a paso la complejidad del algoritmo

1. Inicialización:  $O(1)$
2. La ejecución recorre los  $n$  vértices una vez y para cada vértice se realizan operaciones en el stack:  $O(n)$

**Complejidad Total:**  $O(n)$

# Gracias por su atención

**Arturo González Peñaloza**

**Dulce Julieta Mora Hernández**

Universidad Nacional Autónoma de México

16 de mayo de 2024