

Geometria Computacional

Cristina G. Fernandes

Departamento de Ciência da Computação do IME-USP

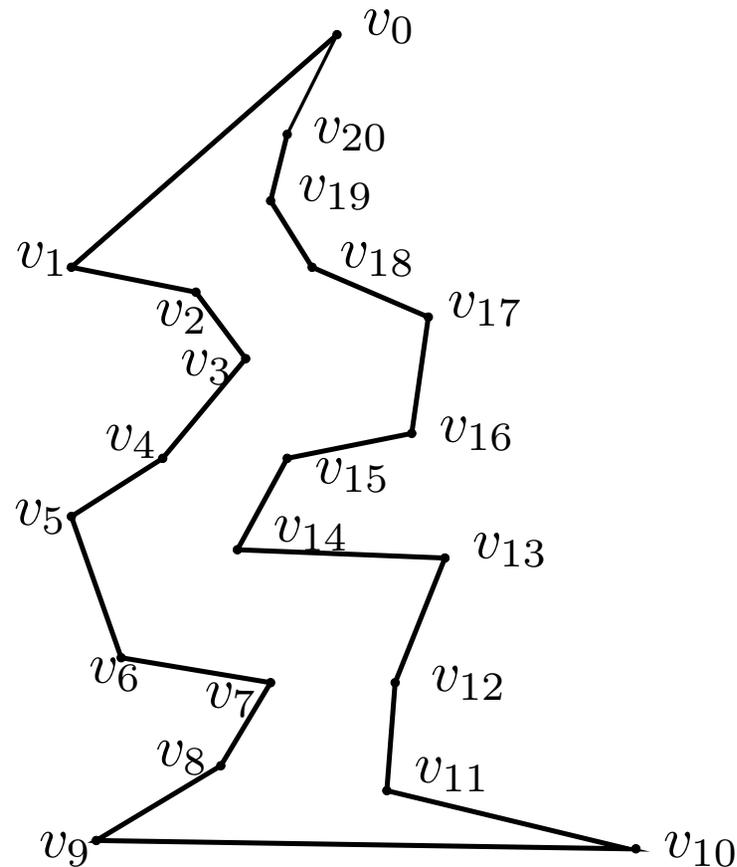
<http://www.ime.usp.br/~cris/>

segundo semestre de 2018

Polígonos monótonos

Um polígono P é **monótono** em relação a uma reta L se $P \cap L'$ é conexo para toda reta L' perpendicular a L .

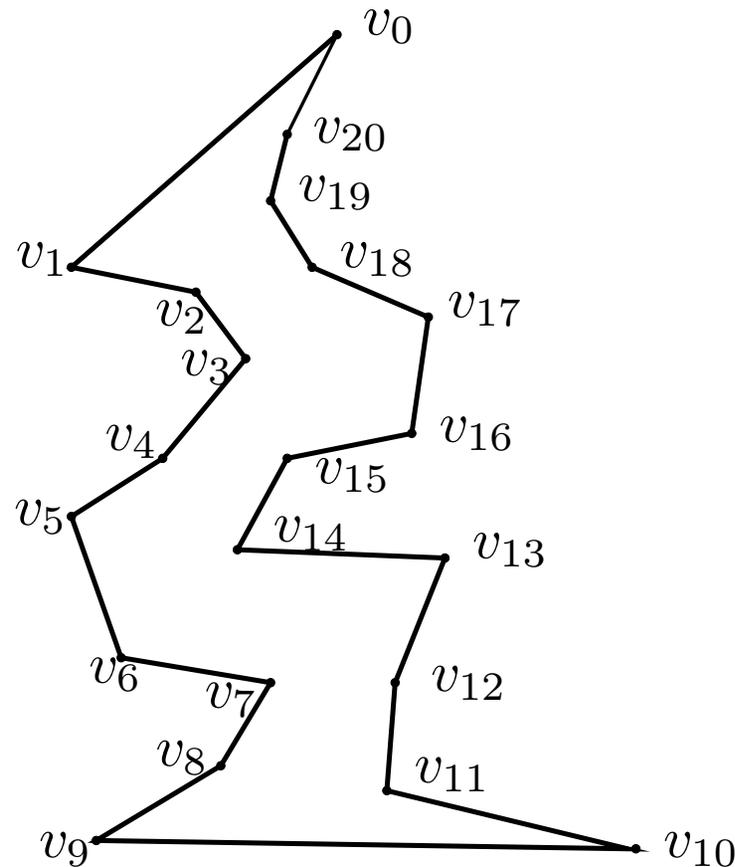
Se L é o eixo y , dizemos que P é **y -monótono**.



Polígonos monótonos

Um polígono P é **monótono** em relação a uma reta L se $P \cap L'$ é conexo para toda reta L' perpendicular a L .

Se L é o eixo y , dizemos que P é **y -monótono**.



Sabemos **triangular** P em tempo linear.

Triangulação em $O(n \lg n)$

P : polígono arbitrário com n vértices

Idéia do algoritmo:

Triangulação em $O(n \lg n)$

P : polígono arbitrário com n vértices

Idéia do algoritmo:

- particionar P em polígonos monótonos
- triangular cada um deles em tempo linear

Triangulação em $O(n \lg n)$

P : polígono arbitrário com n vértices

Idéia do algoritmo:

- particionar P em polígonos monótonos
- triangular cada um deles em tempo linear

Partição tem que consumir tempo $O(n \lg n)$!

Triangulação em $O(n \lg n)$

P : polígono arbitrário com n vértices

Idéia do algoritmo:

- particionar P em polígonos monótonos
- triangular cada um deles em tempo linear

Partição tem que consumir tempo $O(n \lg n)$!

Como fazemos isso?

Triangulação em $O(n \lg n)$

P : polígono arbitrário com n vértices

Idéia do algoritmo:

- particionar P em polígonos monótonos
- triangular cada um deles em tempo linear

Partição tem que consumir tempo $O(n \lg n)$!

Como fazemos isso?

Usando uma **trapezoidação especial** de P .

Trapezoidação

Trapézio: quadrilátero com duas arestas paralelas

Trapezoidação

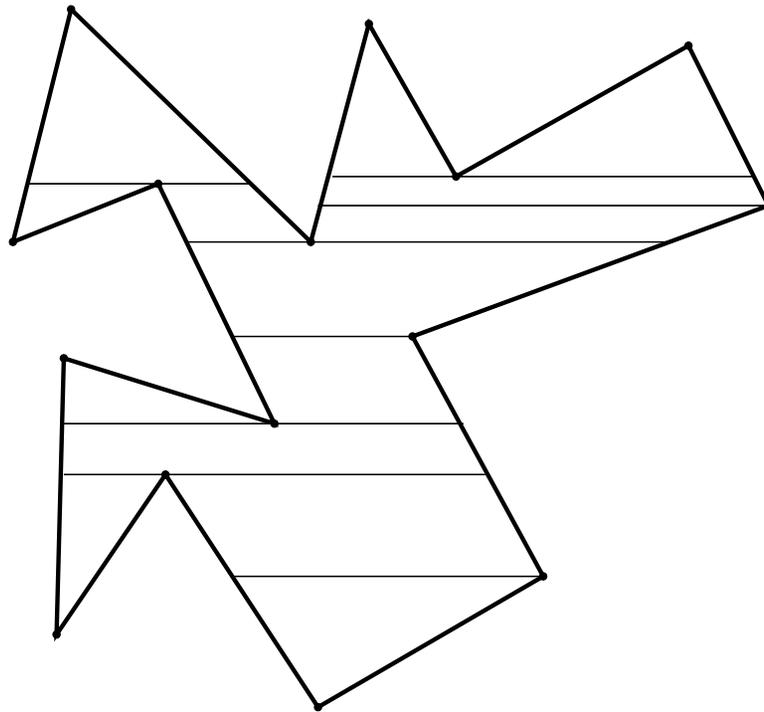
Trapézio: quadrilátero com duas arestas paralelas

Trapezoidação horizontal de um polígono P :
resultado de traçar segmentos horizontais maximais
contidos em P , passando por cada vértice de P .

Trapezoidação

Trapézio: quadrilátero com duas arestas paralelas

Trapezoidação horizontal de um polígono P :
resultado de traçar segmentos horizontais maximais
contidos em P , passando por cada vértice de P .



Trapezoidação

Hipótese simplificadora:

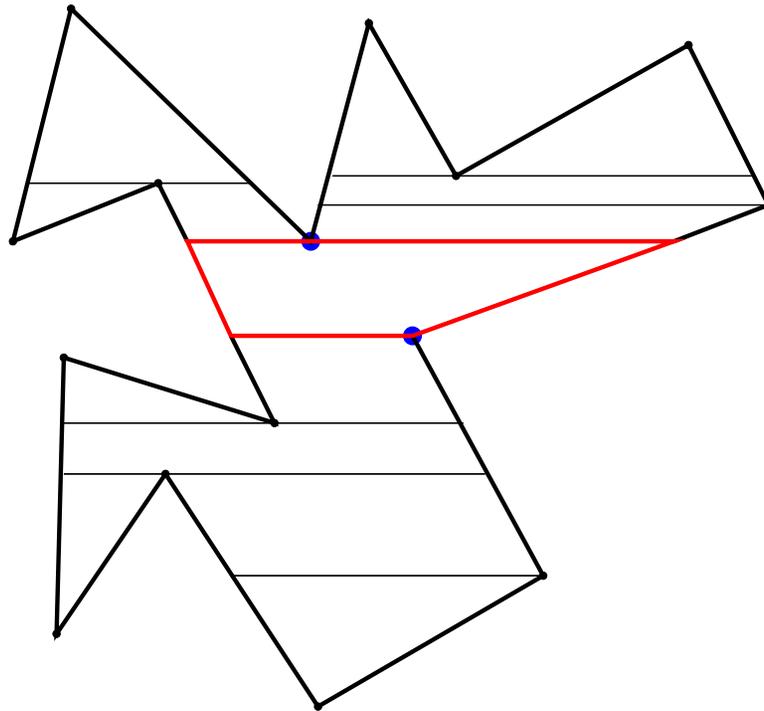
não há dois vértices com a mesma Y -coordenada.

Trapezoidação

Hipótese simplificadora:

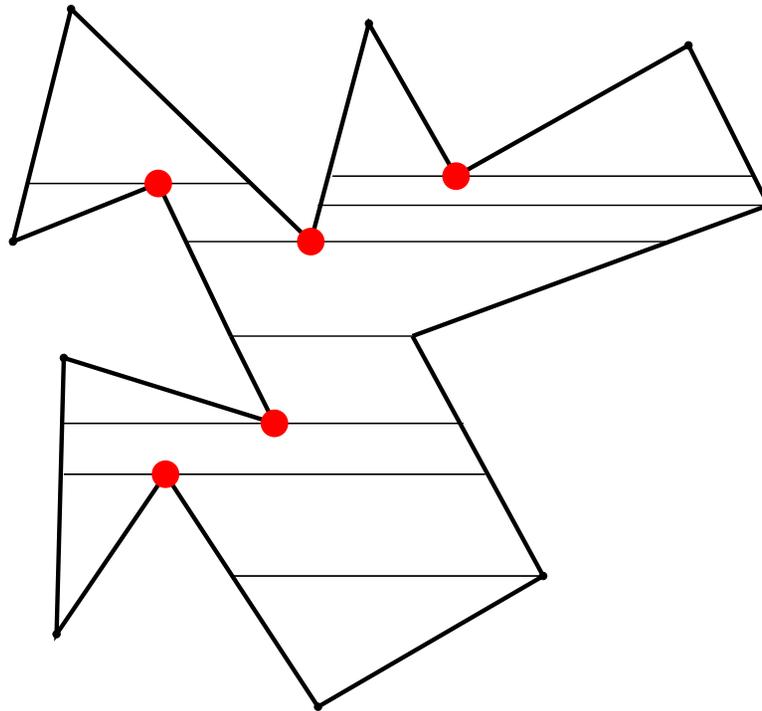
não há dois vértices com a mesma Y -coordenada.

Afirmação: todo trapézio tem exatamente dois vértices de P em sua fronteira (**vértices de suporte**), um na aresta superior, outro na inferior.



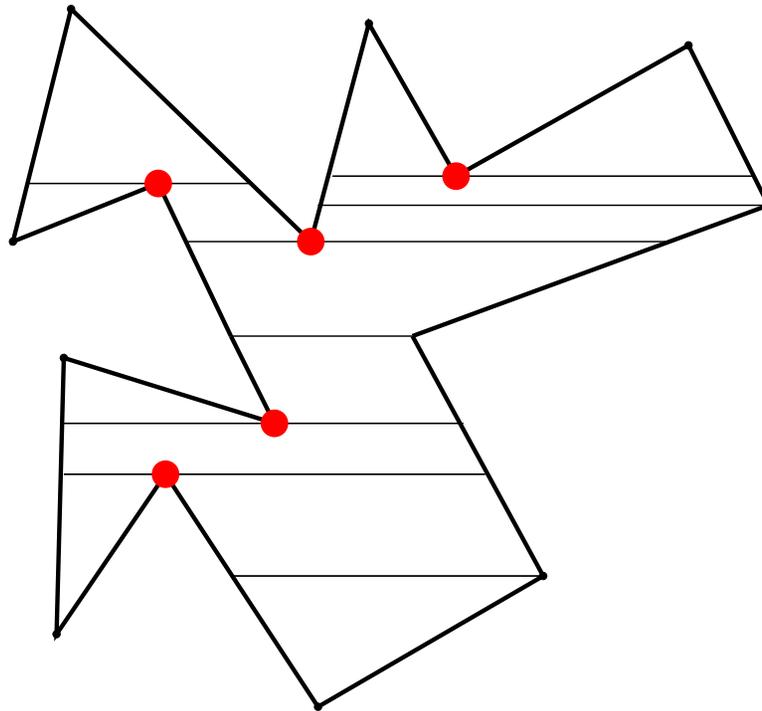
Pontas interiores

Ponta interior de P : vértice v reflexo cujos vizinhos em δP estão ambos acima ou ambos abaixo de v .



Pontas interiores

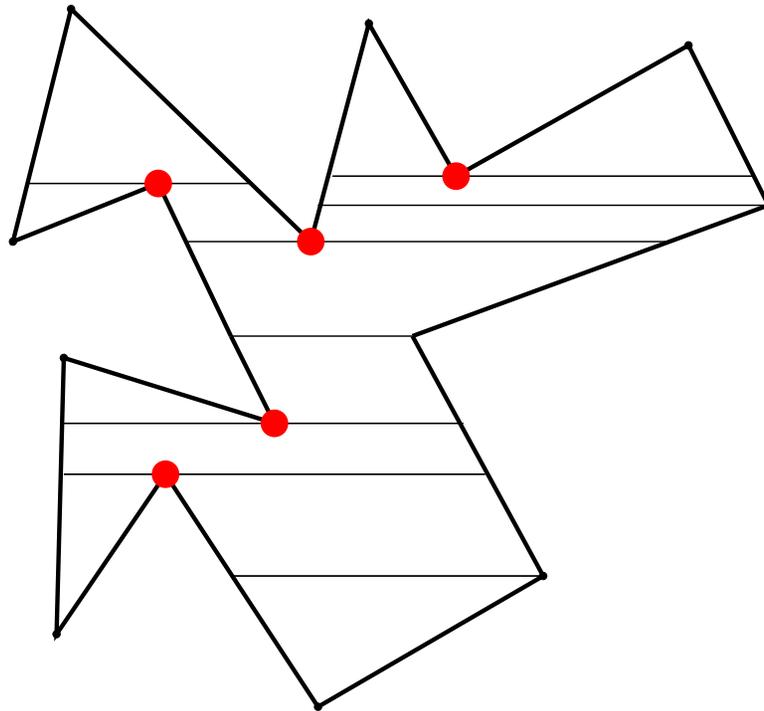
Ponta interior de P : vértice v reflexo cujos vizinhos em δP estão ambos acima ou ambos abaixo de v .



Lema: Se P não tem pontas interiores, P é y -monótono.

Pontas interiores

Ponta interior de P : vértice v reflexo cujos vizinhos em δP estão ambos acima ou ambos abaixo de v .



Lema: Se P não tem pontas interiores, P é y -monótono.

Ponta interior de P :

vértice de suporte no **interior da aresta** do seu trapézio.

Partição em polígonos monótonos

Lema: Se P não tem pontas interiores, P é monótono.

Idéia: acabar com as pontas interiores!

Partição em polígonos monótonos

Lema: Se P não tem pontas interiores, P é monótono.

Idéia: acabar com as pontas interiores!

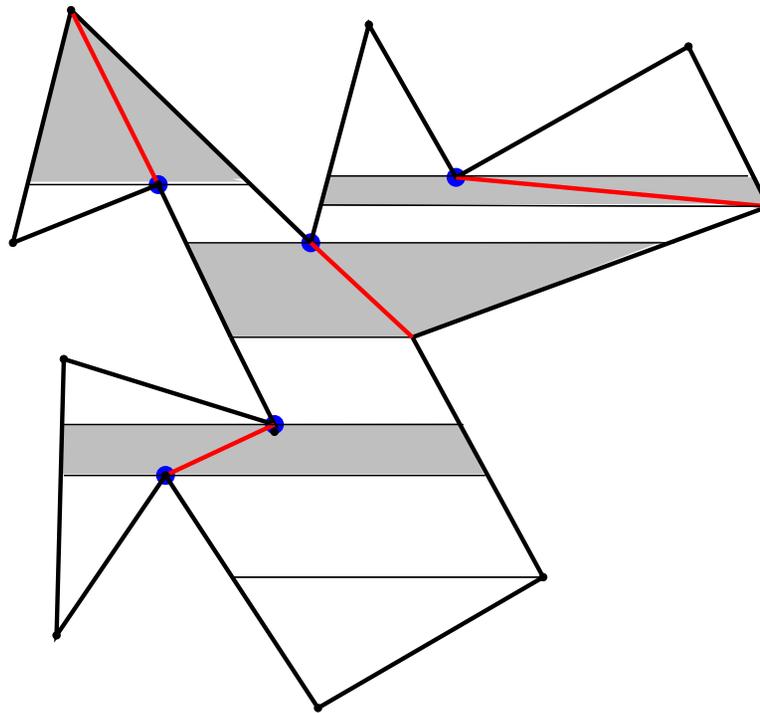
Como?

Partição em polígonos monótonos

Lema: Se P não tem pontas interiores, P é monótono.

Idéia: acabar com as pontas interiores!

Como?



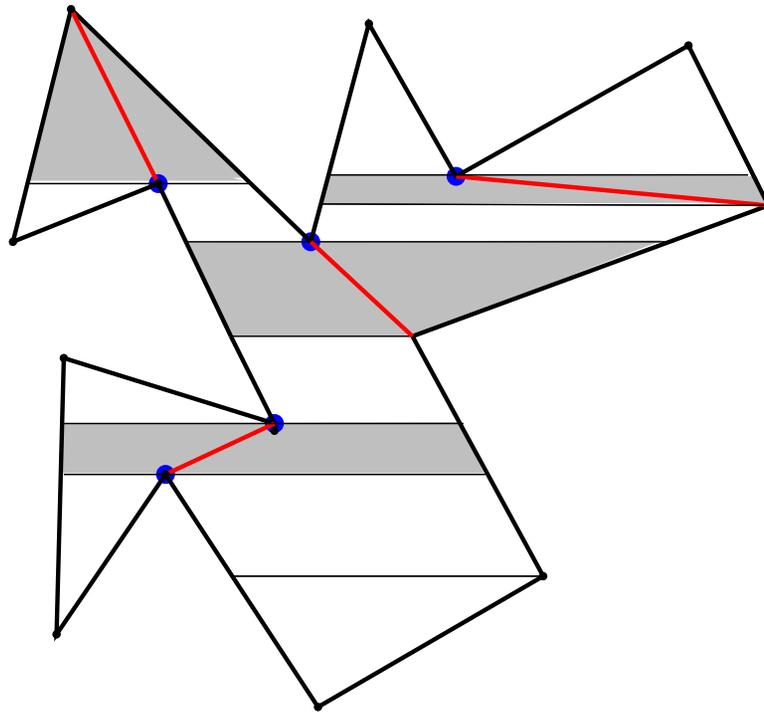
Uma diagonal a partir de cada ponta interior:

Partição em polígonos monótonos

Lema: Se P não tem pontas interiores, P é monótono.

Idéia: acabar com as pontas interiores!

Como?



Uma diagonal a partir de cada ponta interior:
diagonal entre a ponta e o outro vértice de suporte.

Algoritmo de Lee e Preparata

Entrada: polígono P com n vértices

Saída: triangulação de P

Algoritmo de Lee e Preparata

Entrada: polígono P com n vértices

Saída: triangulação de P

Técnica: linha de varredura

Eventos: vértices de P , ordenados por Y -coordenada

Algoritmo de Lee e Preparata

Entrada: polígono P com n vértices

Saída: triangulação de P

Técnica: linha de varredura

Eventos: vértices de P , ordenados por Y -coordenada

ED para a linha de varredura ℓ : ABBB ou skip list

O que é guardado na ED da linha de varredura?

Algoritmo de Lee e Preparata

Entrada: polígono P com n vértices

Saída: triangulação de P

Técnica: linha de varredura

Eventos: vértices de P , ordenados por Y -coordenada

ED para a linha de varredura ℓ : AB BB ou skip list

O que é guardado na ED da linha de varredura?

Trapézios que cruzam ℓ , dados por triplas (e, u, f) , onde

- e e f são as arestas de P que contém respectivamente o lado esquerdo e direito do trapézio
- u é o vértice de suporte superior do trapézio

Algoritmo de Lee e Preparata

Entrada: polígono P com n vértices

Saída: triangulação de P

Técnica: linha de varredura

Eventos: vértices de P , ordenados por Y -coordenada

ED para a linha de varredura ℓ : AB BB ou skip list

O que é guardado na ED da linha de varredura?

Trapézios que cruzam ℓ , dados por triplas (e, u, f) , onde

- e e f são as arestas de P que contém respectivamente o lado esquerdo e direito do trapézio
- u é o vértice de suporte superior do trapézio

(u : candidato a extremo de uma diagonal particionadora)

Algoritmo de Lee e Preparata

Em cada iteração, um evento (vértice) v é processado.
Linha de varredura ℓ sobre v .

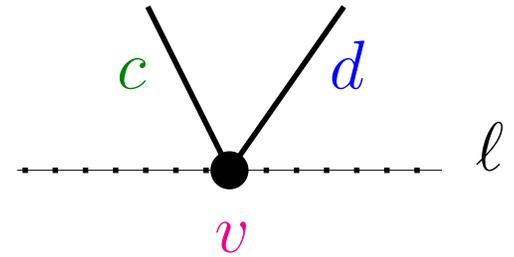
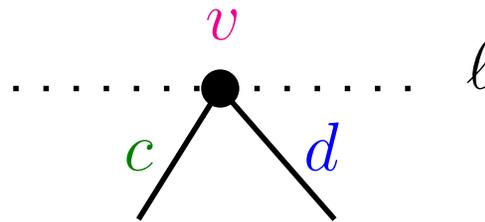
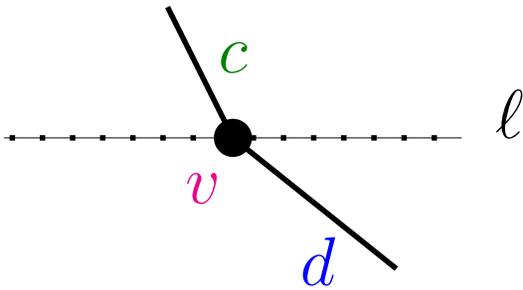
Algoritmo de Lee e Preparata

Em cada iteração, um evento (vértice) v é processado.

Linha de varredura ℓ sobre v .

c e d : arestas do polígono incidentes a v

Três casos a considerar:



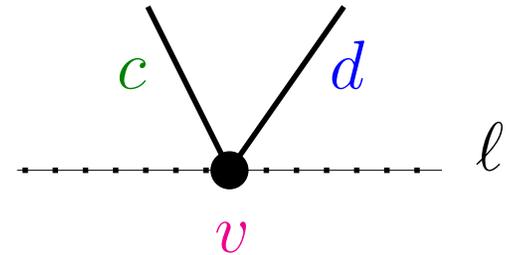
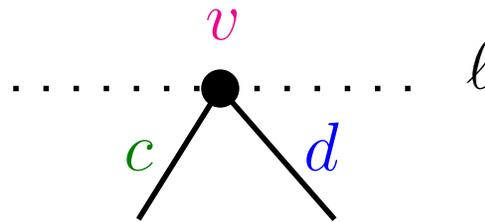
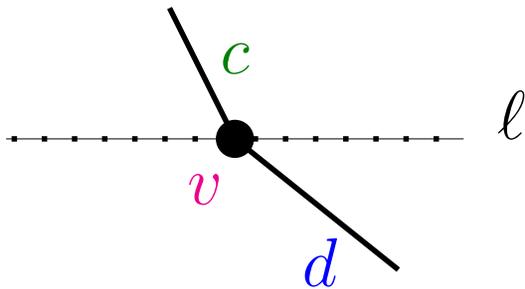
Algoritmo de Lee e Preparata

Em cada iteração, um evento (vértice) v é processado.

Linha de varredura ℓ sobre v .

c e d : arestas do polígono incidentes a v

Três casos a considerar:



Caso 1. Aresta c está acima de ℓ e d abaixo

Caso 2. Arestas c e d estão abaixo de ℓ

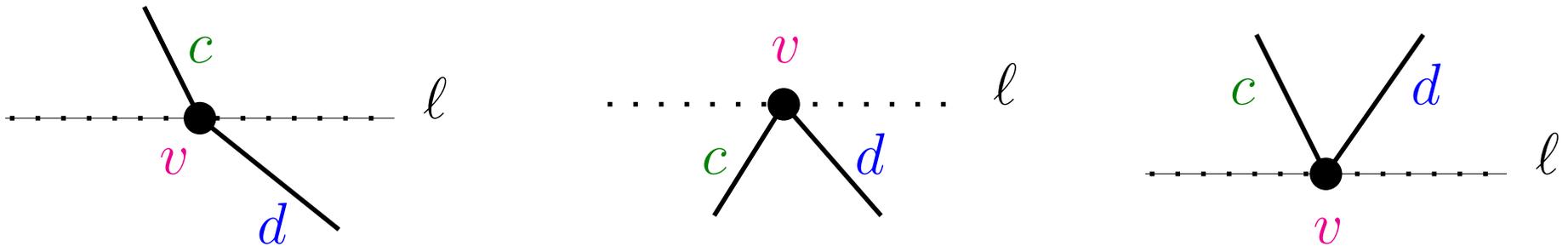
Caso 3. Arestas c e d estão acima de ℓ

Algoritmo de Lee e Preparata

Em cada iteração, um evento (vértice) v é processado.
Linha de varredura ℓ sobre v .

c e d : arestas do polígono incidentes a v

Três casos a considerar:



Caso 1. Aresta c está acima de ℓ e d abaixo

Caso 2. Arestas c e d estão abaixo de ℓ

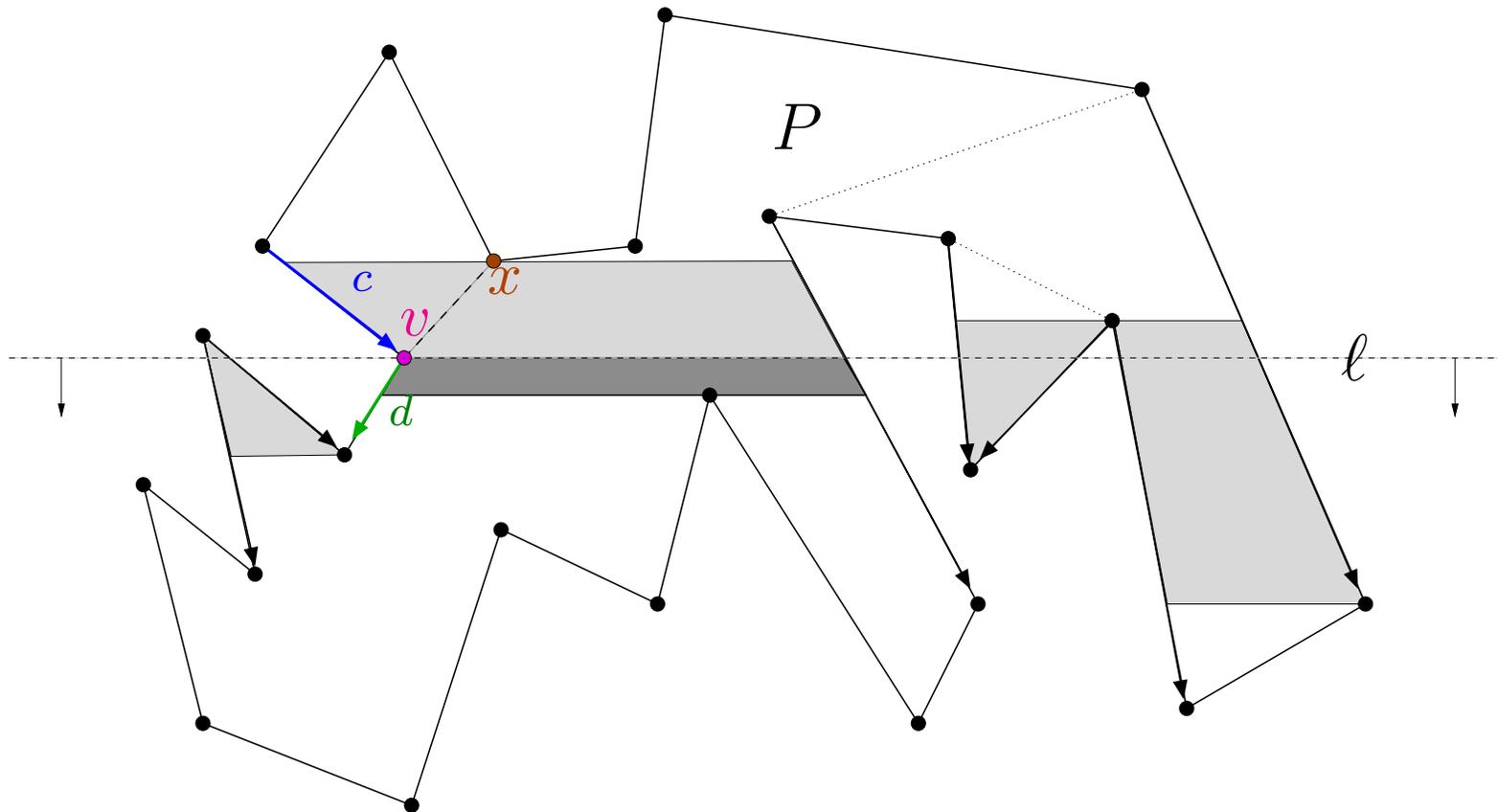
Caso 3. Arestas c e d estão acima de ℓ

No que segue, ED da linha de varredura: AB \overline{BB} T

Caso 1

Caso 1. Aresta c está acima de ℓ e d abaixo

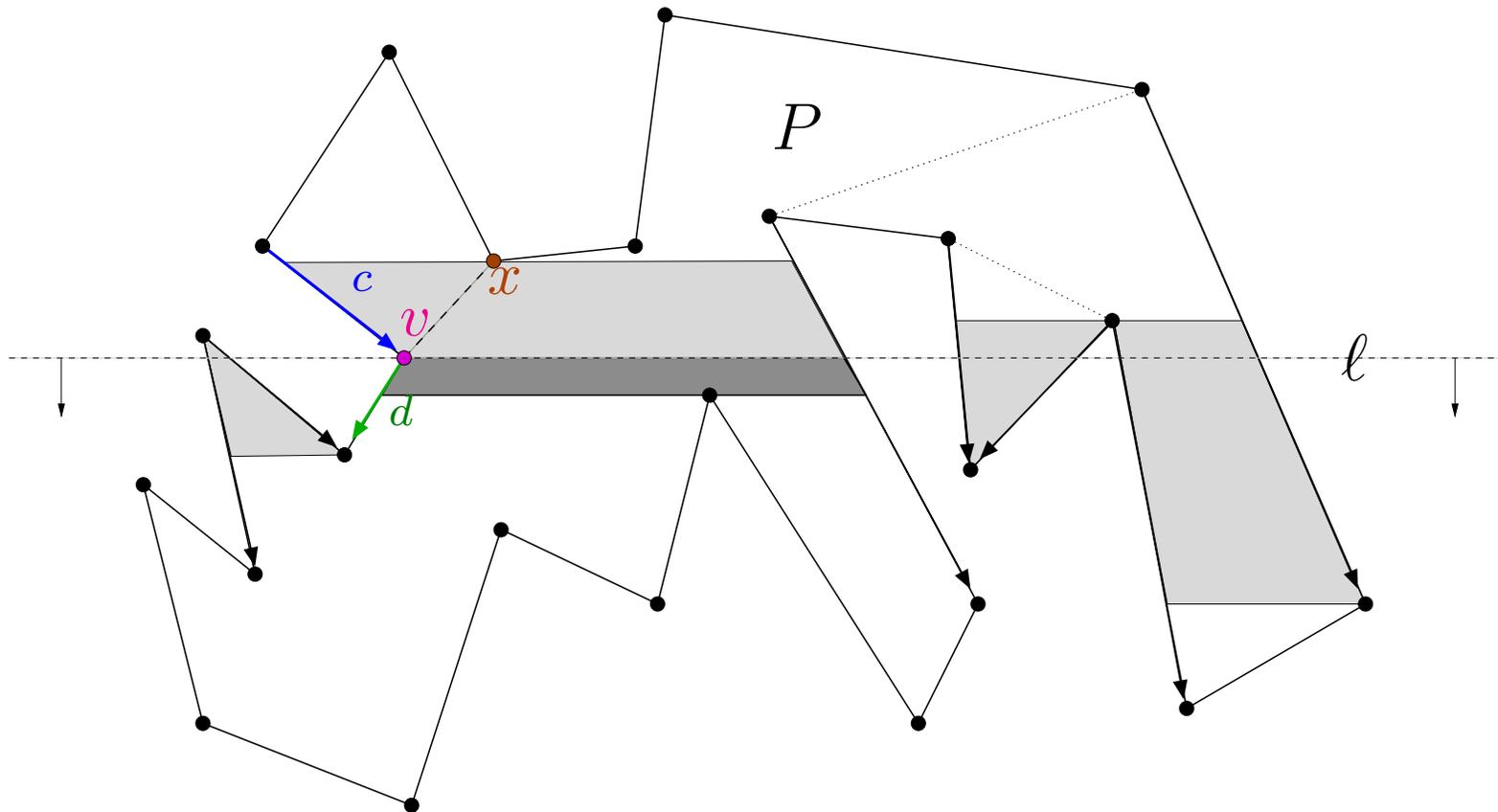
Remova o trapézio (c, x, e) ou (e, x, c)
e insira o trapézio (d, v, e) ou (e, v, d) em T .



Caso 1

Caso 1. Aresta c está acima de ℓ e d abaixo

Remova o trapézio (c, x, e) ou (e, x, c)
e insira o trapézio (d, v, e) ou (e, v, d) em T .



Se x for ponta para baixo, acrescente a diagonal (x, v) .

Teste de ponta para baixo

PONTAPARABAIXO(x, Y, n)

1 $x^- \leftarrow x - 1$ $x^+ \leftarrow x + 1$

2 se $x^- = 0$ então $x^- \leftarrow n$

3 se $x^+ = n + 1$ então $x^+ \leftarrow 1$

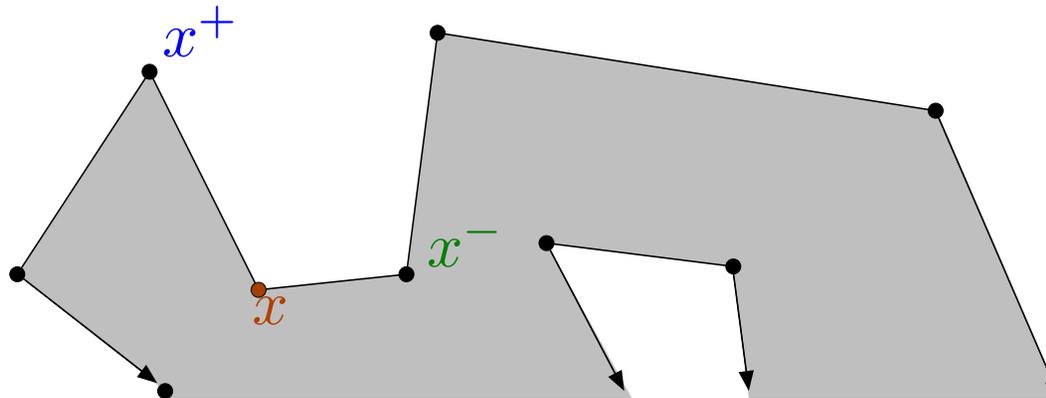
▷ x^- e x^+ são o predecessor e o sucessor de x em δP

4 se $Y[x^-] > Y[x]$ e $Y[x^+] > Y[x]$

▷ x é ponta interior para baixo?

5 então devolva VERDADE

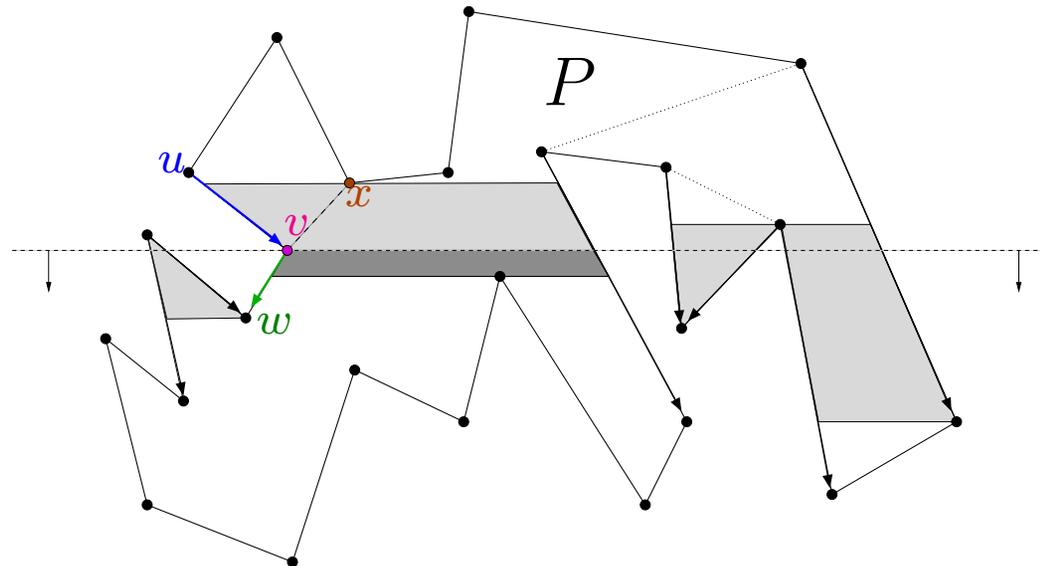
6 senão devolva FALSO



Caso 1

TRATACASO1(T, u, v, w, Y, n, D, t)

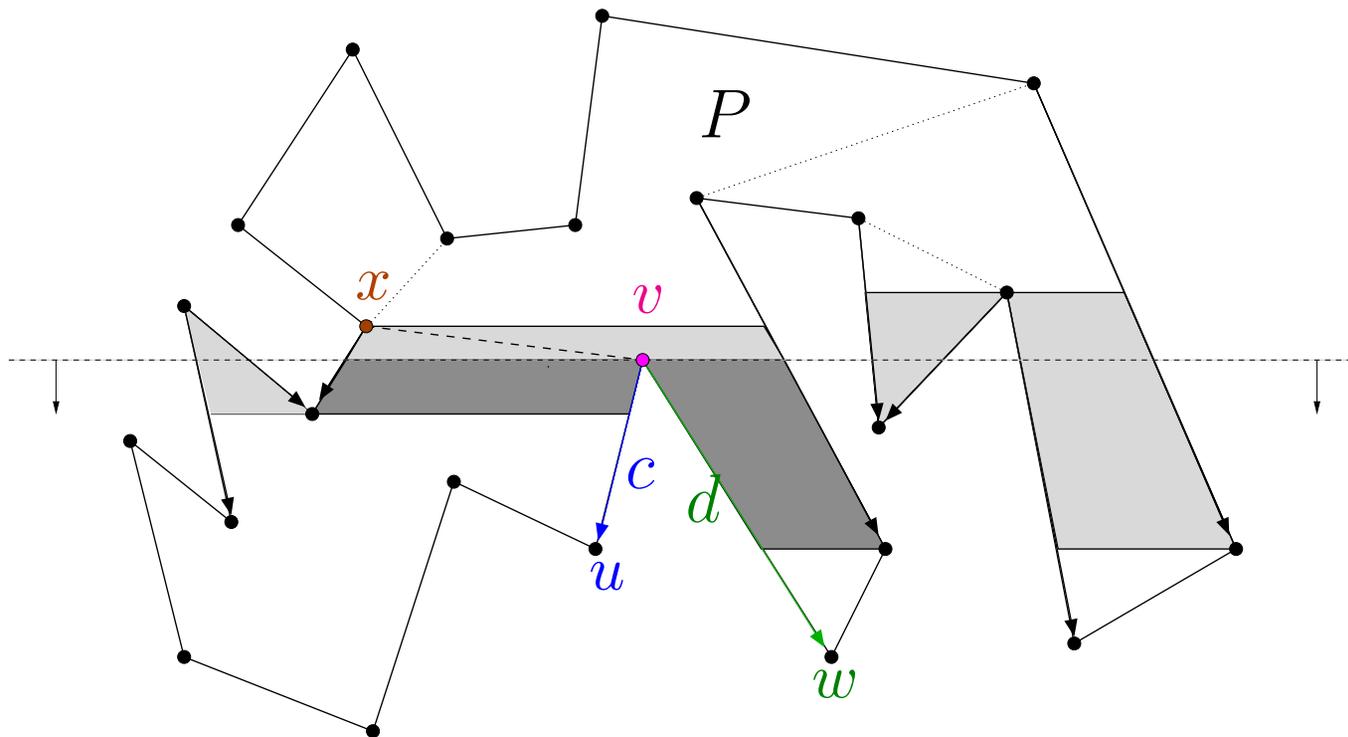
- 1 se $Y[u] < Y[w]$ então $u \leftrightarrow w$
- 2 $((i, j), x, (k, l)) \leftarrow \text{Remove}(T, v)$
- 3 se $v = j$ \triangleright o trapézio está à direita de v ?
- 4 então $\text{Insere}(T, (v, w), v, (k, l))$
- 5 senão $\text{Insere}(T, (i, j), v, (v, w))$
- 6 se $\text{PONTAPARABAIXO}(x, Y, n)$
- 7 então $t \leftarrow t + 1$ $D[t] \leftarrow (x, v)$



Caso 2

Caso 2. Arestas c e d estão abaixo de ℓ

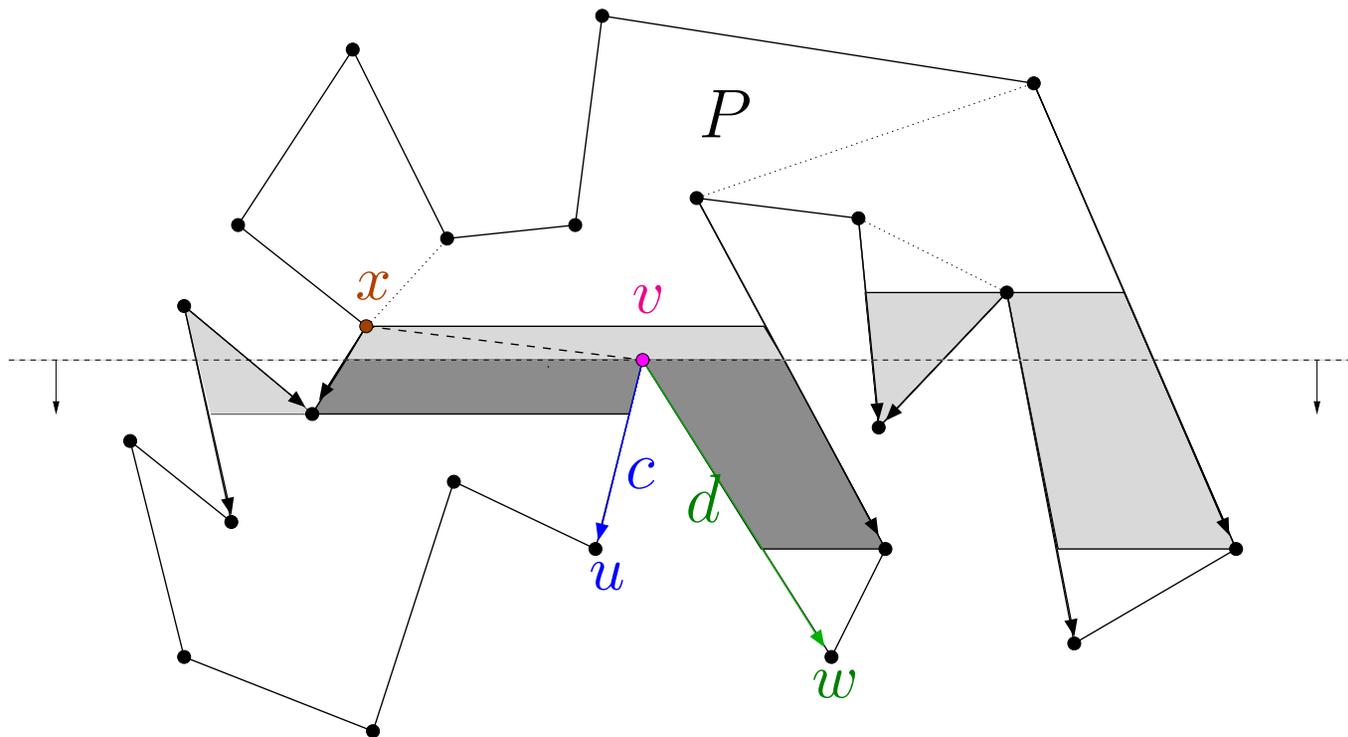
Se há trapézio com v em T ,
então substitua-o por dois trapézios:
um com lado direito em c e outro com lado esquerdo em d .



Caso 2

Caso 2. Arestas c e d estão abaixo de ℓ

Se há trapézio com v em T ,
então substitua-o por dois trapézios:
um com lado direito em c e outro com lado esquerdo em d .

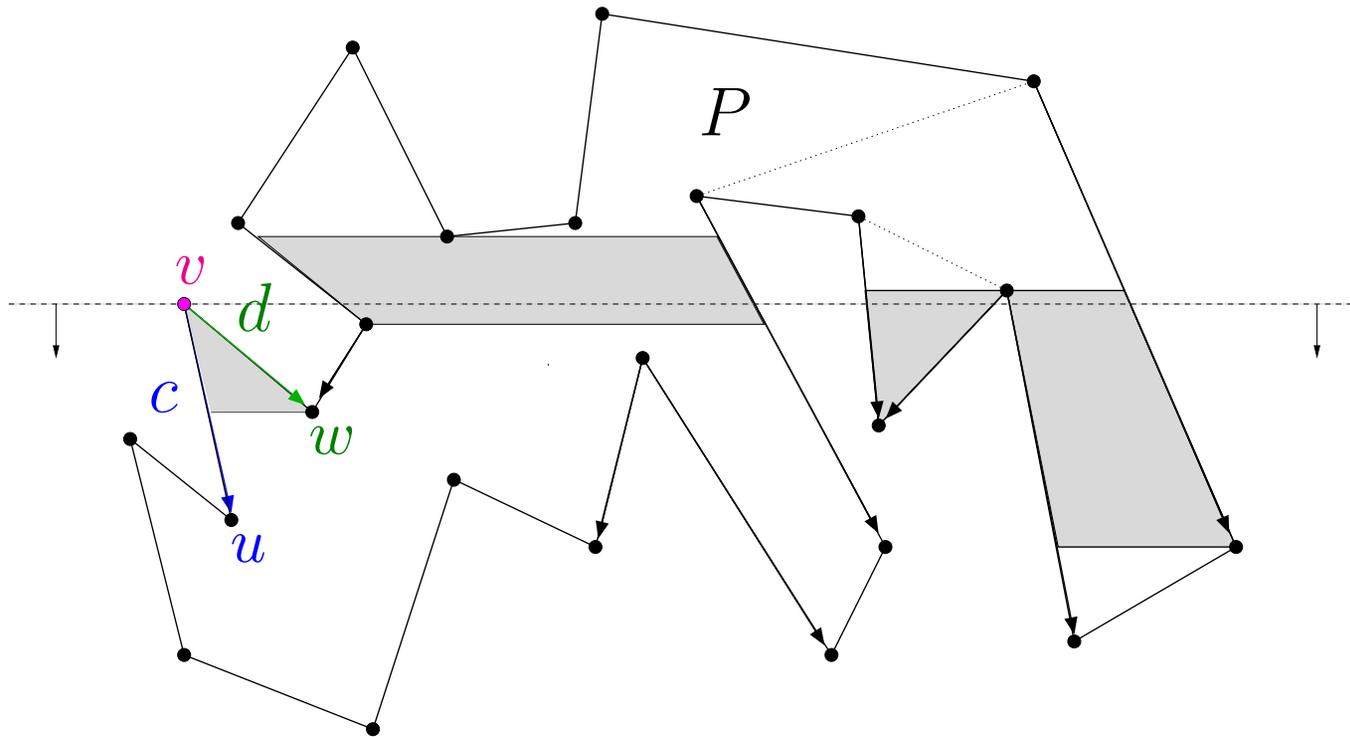


Acrescente a diagonal (x, v) .

Caso 2

Caso 2. Arestas c e d estão abaixo de ℓ

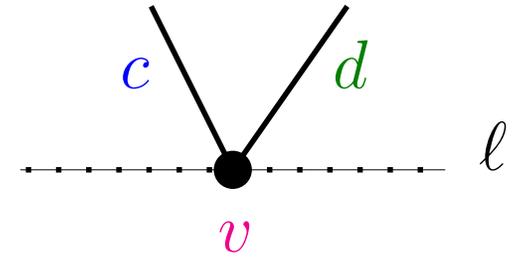
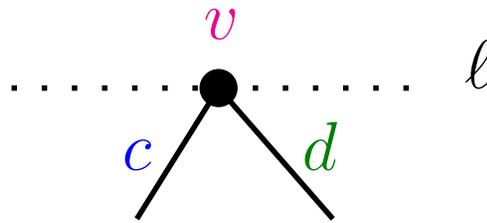
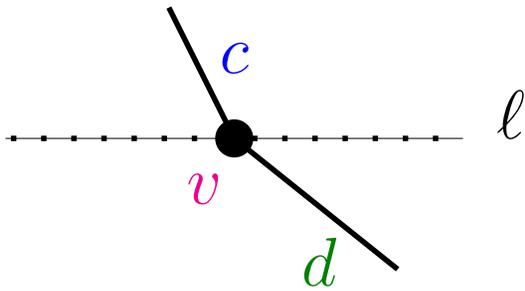
Se há trapézio com v em T ,
então substitua-o por dois trapézios:
um com lado direito em c e outro com lado esquerdo em d .
Se não há trapézio com v em T , insira (c, v, d) em T .



Algoritmo de Lee e Preparata

T : ED da linha de varredura ℓ

Três casos a considerar:



Caso 1. Aresta c está acima de ℓ e d abaixo

Caso 2. Arestas c e d estão abaixo de ℓ

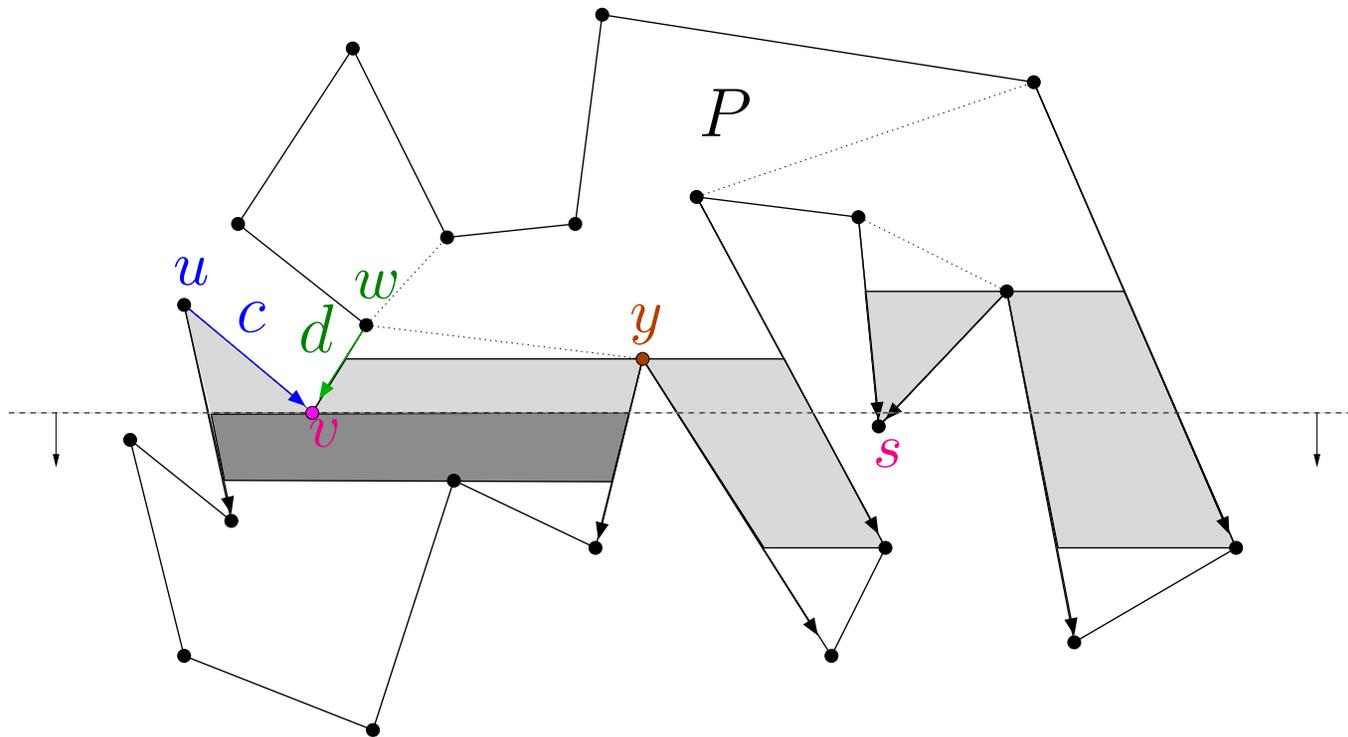
Caso 3. Arestas c e d estão acima de ℓ

Caso 3

Caso 3. Arestas c e d estão acima de ℓ

Pode haver um ou dois trapézios contendo v em T .

Se houver dois, remova (e, x, c) e (d, y, f) de T e insira (e, v, f) .

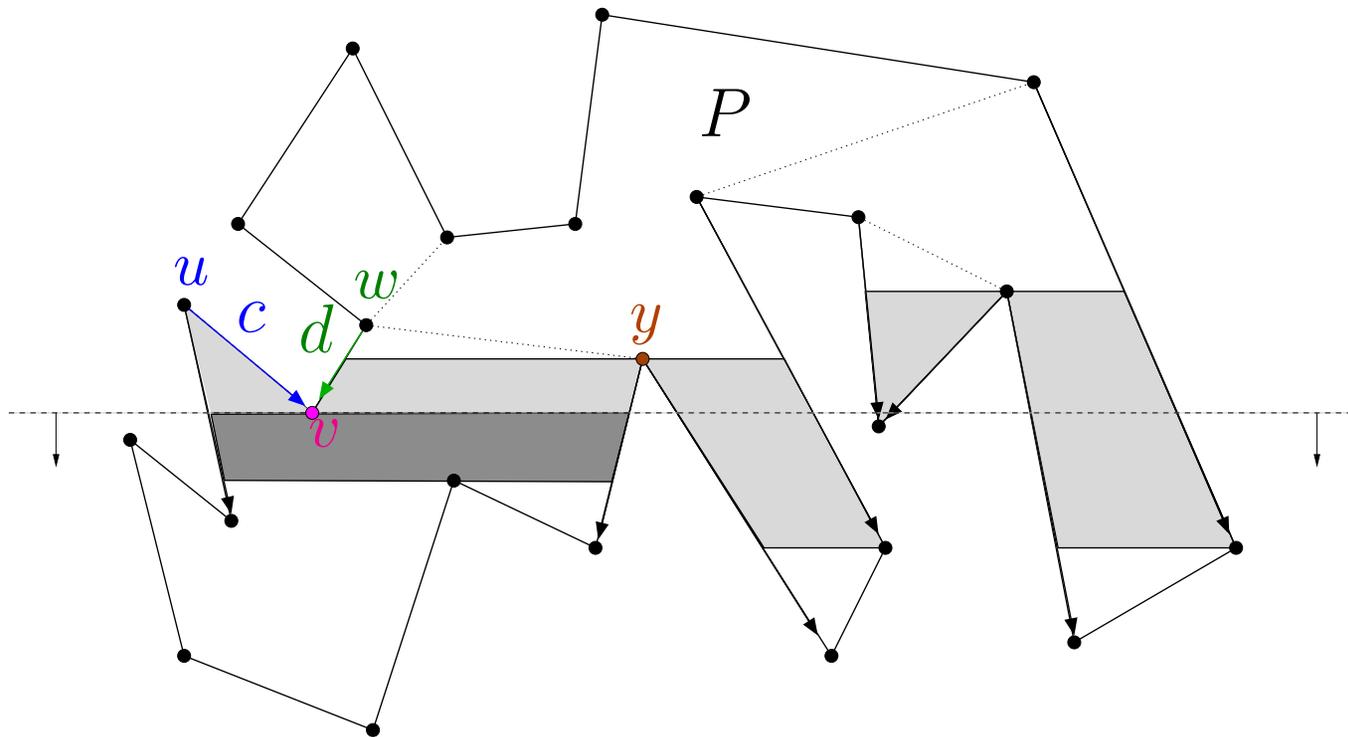


Caso 3

Caso 3. Arestas c e d estão acima de ℓ

Pode haver um ou dois trapézios contendo v em T .

Se houver dois, remova (e, x, c) e (d, y, f) de T e insira (e, v, f) .



Se x e/ou y forem ponta para baixo,
acrescente as diagonais (x, v) e/ou (y, v) .

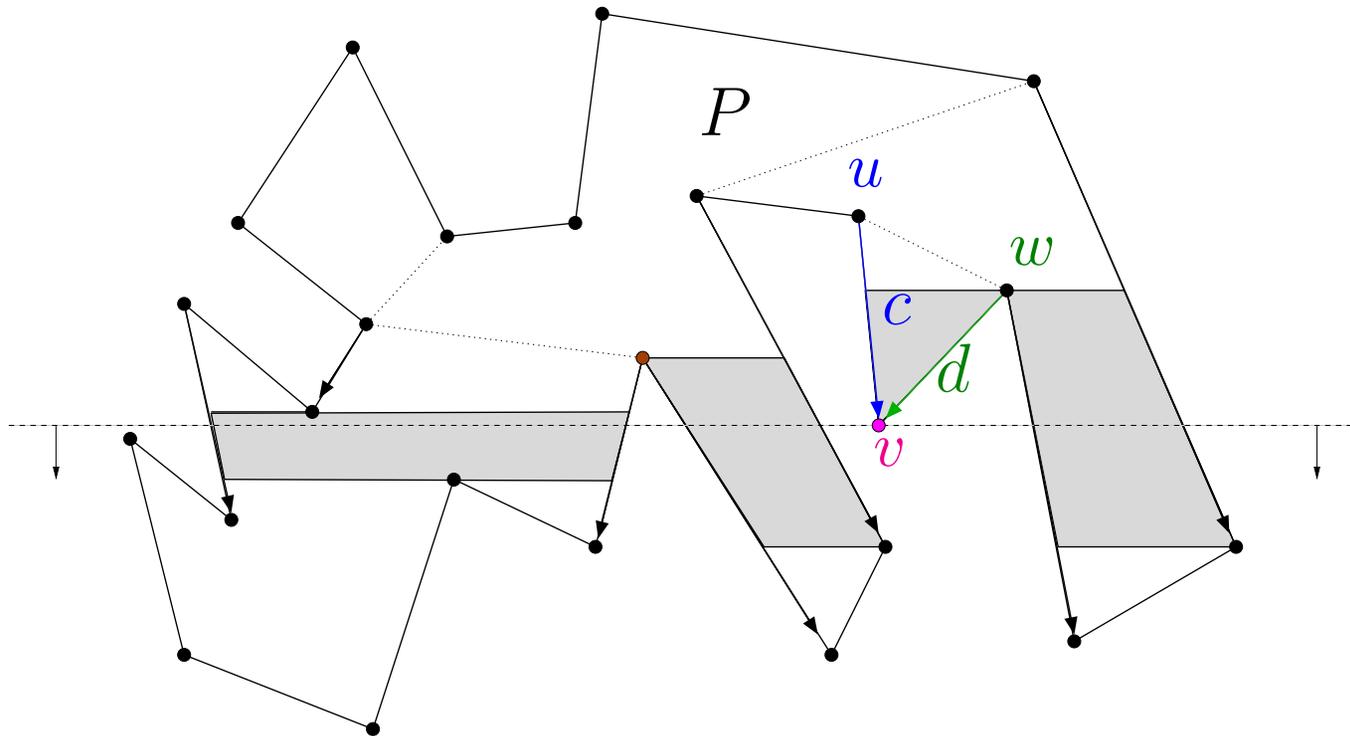
Caso 3

Caso 3. Arestas c e d estão acima de ℓ

Pode haver um ou dois trapézios contendo v em T .

Se houver dois, remova (e,x,c) e (d,y,f) de T e insira (e,v,f) .

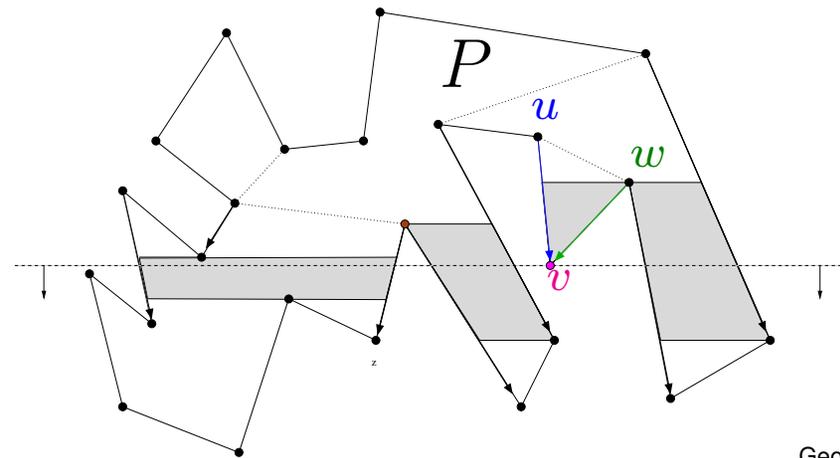
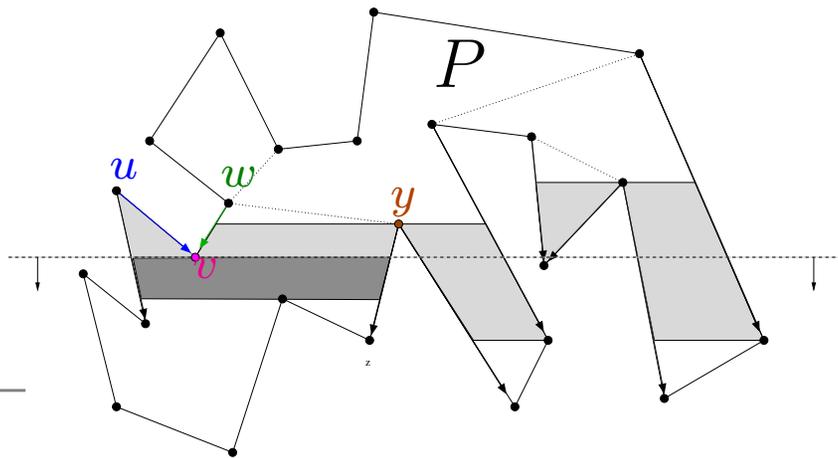
Se houver um, remova-o.



Caso 3

TRATACASO3(T, v, Y, n, D, t)

- 1 $((i, j), x, (k, l)) \leftarrow \text{Remove}(T, v)$
- 2 se PONTAPARABAIXO(x, Y, n)
- 3 então $t \leftarrow t + 1$ $D[t] \leftarrow (x, v)$
- 4 se $j \neq v$ ou $k \neq v$ \triangleright há um outro trapézio em T ?
- 5 então $((i', j'), y, (k', l')) \leftarrow \text{Remove}(T, v)$
- 6 se PONTAPARABAIXO(y, Y, n)
- 7 então $t \leftarrow t + 1$ $D[t] \leftarrow (y, v)$
- 8 se $l = v$
- 9 então **Inserere**(i, j, v, k', l')
- 10 senão **Inserere**(i', j', v, k, l)



Algoritmo de Lee e Preparata

DIVIDEEMMONÓTONO-LP(X, Y, n)

```
1  para  $k \leftarrow 1$  até  $n$  faça
2       $E[k] \leftarrow k$ 
3  MERGESORT( $Y, X, 1, n, E$ )    ▷ ordenação indireta decrescente de
4  CRIE( $T$ )     $t \leftarrow 0$ 
5  para  $k \leftarrow 1$  até  $n$  faça
6       $v \leftarrow E[k]$ 
7       $v^- \leftarrow v - 1$      $v^+ \leftarrow v + 1$ 
8      se  $v^- = 0$  então  $v^- \leftarrow n$ 
9      se  $v^+ = n + 1$  então  $v^+ \leftarrow 1$ 
10     se  $Y[v^-] < Y[v] < Y[v^+]$  ou  $Y[v^+] < Y[v] < Y[v^-]$ 
11         então TRATACASO1( $T, v^-, v, v^+, Y, n, D, t$ )
12     senão se  $Y[v^-] < Y[v]$ 
13         então TRATACASO2( $T, v^-, v, v^+, X, D, t$ )
14     senão TRATACASO3( $T, v, Y, n, D, t$ )
15 devolva ( $D, t$ )
```

Consumo de tempo

O tratamento de cada caso consome tempo $O(\lg n)$.

Assim, o algoritmo `DIVIDEEMMONÓTONO-LP` consome tempo $O(\lg n)$ por iteração.

São n iterações, logo o consumo de tempo total é $O(n \lg n)$.

Consumo de tempo

O tratamento de cada caso consome tempo $O(\lg n)$.

Assim, o algoritmo `DIVIDEEMMONÓTONO-LP` consome tempo $O(\lg n)$ por iteração.

São n iterações, logo o consumo de tempo total é $O(n \lg n)$.

Como se livrar da hipótese simplificadora?

Consumo de tempo

O tratamento de cada caso consome tempo $O(\lg n)$.

Assim, o algoritmo `DIVIDEEMMONÓTONO-LP` consome tempo $O(\lg n)$ por iteração.

São n iterações, logo o consumo de tempo total é $O(n \lg n)$.

Como se livrar da hipótese simplificadora?

Como acoplar isso com o algoritmo linear para triangular polígonos monótonos?