Geometria Computacional

Cristina G. Fernandes
Departamento de Ciência da Computação do IME-USP
`http://www.ime.usp.br/~cris/`

Escola de Verão da Maratona de Programação

janeiro de 2020

Descrição da aula

Parte I: Algoritmos para fecho convexo

Descrição da aula

Parte I: Algoritmos para fecho convexo

Parte II: Algoritmos de linha de varredura

Descrição da aula

Parte I: Algoritmos para fecho convexo

Parte II: Algoritmos de linha de varredura

Parte III: Animação dos algoritmos

Descrição da aula

Parte I: Algoritmos para fecho convexo

Parte II: Algoritmos de linha de varredura

Parte III: Animação dos algoritmos

Combinação convexa

P: coleção de pontos do plano, dada por $X[1\,..\,n]$, $Y[1\,..\,n]$.

Combinação convexa

P: coleção de pontos do plano, dada por $X[1\,..\,n], Y[1\,..\,n]$.

Combinação convexa de pontos de P: soma da forma

$$\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]),$$

com $\alpha_i \geq 0$, para $i = 1, \ldots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.

Combinação convexa

P: coleção de pontos do plano, dada por $X[1 \mathinner{.\,.} n]$, $Y[1 \mathinner{.\,.} n]$.

Combinação convexa de pontos de P: soma da forma

$$\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]),$$

com $\alpha_i \geq 0$, para $i = 1, \ldots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.

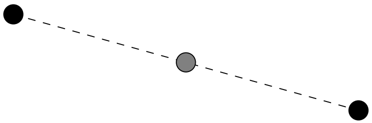
Combinação convexa

P: coleção de pontos do plano, dada por $X[1\,..\,n]$, $Y[1\,..\,n]$.

Combinação convexa de pontos de P: soma da forma

$$\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]),$$

com $\alpha_i \geq 0$, para $i = 1, \ldots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



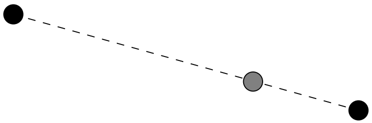
Combinação convexa

P: coleção de pontos do plano, dada por $X[1\,.\,.\,n]$, $Y[1\,.\,.\,n]$.

Combinação convexa de pontos de P: soma da forma

$$\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]),$$

com $\alpha_i \geq 0$, para $i = 1, \ldots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



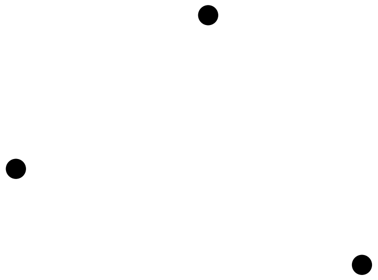
Combinação convexa

P: coleção de pontos do plano, dada por $X[1\,..\,n]$, $Y[1\,..\,n]$.

Combinação convexa de pontos de P: soma da forma

$$\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]),$$

com $\alpha_i \geq 0$, para $i = 1, \ldots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



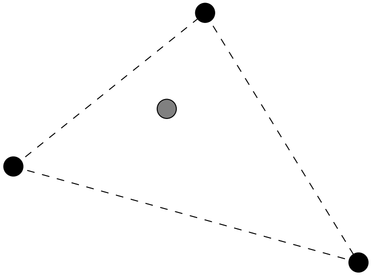
Combinação convexa

P: coleção de pontos do plano, dada por $X[1\,..\,n], Y[1\,..\,n]$.

Combinação convexa de pontos de P: soma da forma

$$\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]),$$

com $\alpha_i \geq 0$, para $i = 1, \ldots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



Fecho convexo

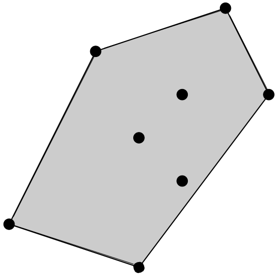
Fecho convexo de P: conjunto de combinações convexas de pontos de P, ou seja,

$$\begin{aligned}\mathrm{conv}(P) := \{&\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]) : \\ &\alpha_1 + \cdots + \alpha_n = 1, \text{ e } \alpha_i \geq 0 \ (i = 1, \ldots, n)\}.\end{aligned}$$

Fecho convexo

Fecho convexo de P: conjunto de combinações convexas de pontos de P, ou seja,

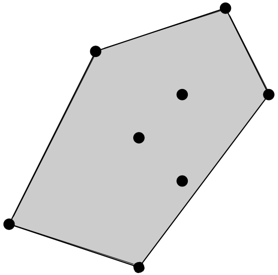
$$\mathrm{conv}(P) := \{\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]) :$$
$$\alpha_1 + \cdots + \alpha_n = 1, \text{ e } \alpha_i \geq 0 \ (i = 1, \ldots, n)\}.$$



Fecho convexo

Fecho convexo de P: conjunto de combinações convexas de pontos de P, ou seja,

$$\mathrm{conv}(P) := \{\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]) : \\ \alpha_1 + \cdots + \alpha_n = 1, \text{ e } \alpha_i \geq 0 \ (i = 1, \ldots, n)\}.$$



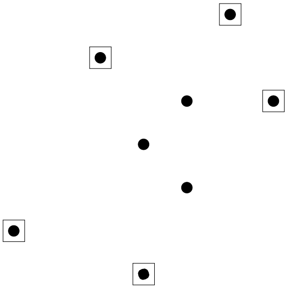
Problema: Dada uma coleção P de pontos do plano, determinar o fecho convexo de P.

Pontos extremos

Ponto (x, y) de P é extremo
se não é combinação convexa de pontos de $P \setminus \{(x, y)\}$.

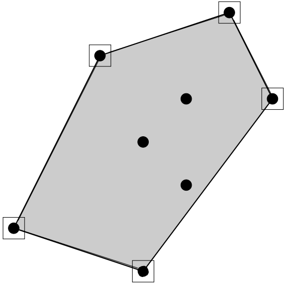
Pontos extremos

Ponto (x, y) de P é extremo
se não é combinação convexa de pontos de $P \setminus \{(x, y)\}$.



Pontos extremos

Ponto (x, y) de P é extremo
se não é combinação convexa de pontos de $P \setminus \{(x, y)\}$.



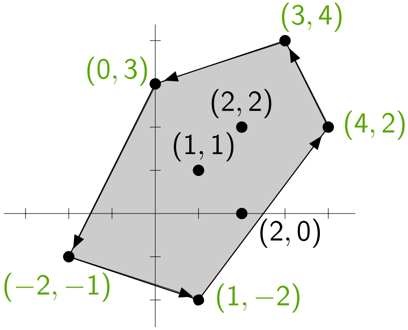
Pontos extremos de $\mathrm{conv}(P)$ são pontos extremos de P.

Representação do fecho convexo

Representação do fecho convexo: vetor $H[1\,..\,h]$ com índices dos pontos extremos na ordem em que aparecem na fronteira do fecho convexo (sentido anti-horário).

Representação do fecho convexo

Representação do fecho convexo: vetor $H[1\,..\,h]$ com índices dos pontos extremos na ordem em que aparecem na fronteira do fecho convexo (sentido anti-horário).

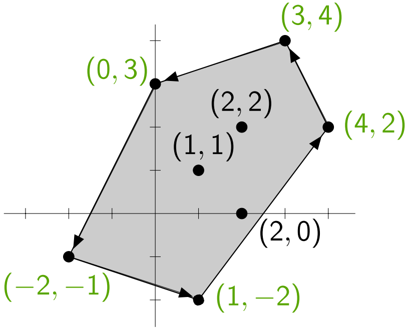


	1	2	3	4	5	6	7	8
X	1	3	2	−2	1	2	4	0
Y	1	4	0	−1	−2	2	2	3

	1	2	3	4	5
H	2	8	4	5	7

Representação do fecho convexo

Representação do fecho convexo: vetor $H[1\,.\,.\,h]$ com índices dos pontos extremos na ordem em que aparecem na fronteira do fecho convexo (sentido anti-horário).



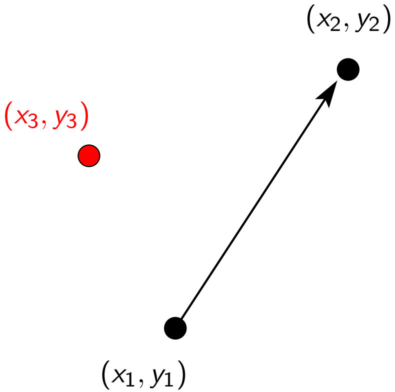
	1	2	3	4	5	6	7	8
X	1	3	2	−2	1	2	4	0
Y	1	4	0	−1	−2	2	2	3

	1	2	3	4	5
H	2	8	4	5	7

Os pontos de índice 2, 4, 5, 7 e 8 são extremos.

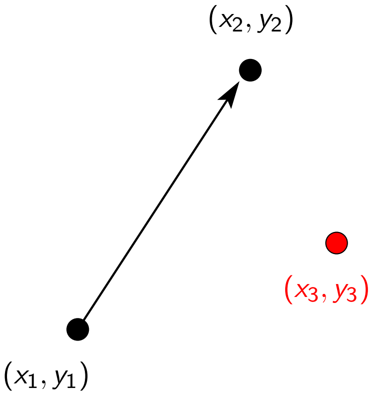
Predicados geométricos

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3))$ = verdade
Direita$((x_1, y_1), (x_2, y_2), (x_3, y_3))$ = falso



Predicados geométricos

$\text{Esquerda}((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \text{falso}$
$\text{Direita}((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \text{verdade}$



Predicados geométricos

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3))$

Predicados geométricos

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3))$: sinal do determinante

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Predicados geométricos

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3))$: sinal do determinante

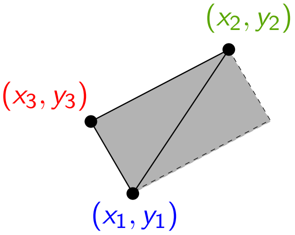
$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1).$$

Predicados geométricos

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3))$: sinal do determinante

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1).$$

O valor absoluto deste número é duas vezes
a área do triângulo de extremos (x_1, y_1), (x_2, y_2) e (x_3, y_3).



Predicados geométricos

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3))$

```
se (x2 − x1)(y3 − y1) − (x3 − x1)(y2 − y1) ≥ 0
    então devolva verdade
    senão devolva falso
```

$$\text{se } (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \geq 0$$

Predicados geométricos

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3))$

se $(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \geq 0$
    então devolva verdade
    senão devolva falso

Esquerda$^+((x_1, y_1), (x_2, y_2), (x_3, y_3))$

se $(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) > 0$
    então devolva verdade
    senão devolva falso

Predicados geométricos

$\text{Esquerda}((x_1, y_1), (x_2, y_2), (x_3, y_3))$

```
se (x2 − x1)(y3 − y1) − (x3 − x1)(y2 − y1) ≥ 0
     então devolva verdade
     senão devolva falso
```

$\text{Esquerda}^{+}((x_1, y_1), (x_2, y_2), (x_3, y_3))$

```
se (x2 − x1)(y3 − y1) − (x3 − x1)(y2 − y1) > 0
     então devolva verdade
     senão devolva falso
```

Podemos definir similarmente

$\text{Direita}((x_1, y_1), (x_2, y_2), (x_3, y_3))$
$\text{Direita}^{+}((x_1, y_1), (x_2, y_2), (x_3, y_3))$
$\text{Colinear}((x_1, y_1), (x_2, y_2), (x_3, y_3))$

Predicados geométricos

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3))$

```
se (x2 − x1)(y3 − y1) − (x3 − x1)(y2 − y1) ≥ 0
    então devolva verdade
    senão devolva falso
```

Esquerda$^+((x_1, y_1), (x_2, y_2), (x_3, y_3))$

```
se (x2 − x1)(y3 − y1) − (x3 − x1)(y2 − y1) > 0
    então devolva verdade
    senão devolva falso
```

Podemos definir similarmente

Direita$((x_1, y_1), (x_2, y_2), (x_3, y_3))$
Direita$^+((x_1, y_1), (x_2, y_2), (x_3, y_3))$
Colinear$((x_1, y_1), (x_2, y_2), (x_3, y_3))$

Com números reais, cuidado com problemas de precisão!

Abreviaturas

Coleção $X[1\,..\,n]$, $Y[1\,..\,n]$ de pontos.

Abreviaturas

Coleção $X[1\,..\,n]$, $Y[1\,..\,n]$ de pontos.

$\text{Esq}(X, Y, i, j, k) = \text{Esquerda}((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

Abreviaturas

Coleção $X[1\,..\,n], Y[1\,..\,n]$ de pontos.

$\text{Esq}(X, Y, i, j, k) = \text{Esquerda}((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

Em pseudocódigo:

```
Esq(X, Y, i, j, k)
  devolva Esquerda((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))
```

Abreviaturas

Coleção $X[1..n], Y[1..n]$ de pontos.

$\text{Esq}(X, Y, i, j, k) = \text{Esquerda}((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

Em pseudocódigo:

Esq(X, Y, i, j, k)
1 devolva Esquerda($(X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k])$)

Similarmente

$\text{Dir}(X, Y, i, j, k) = \text{Direita}((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

Em pseudocódigo:

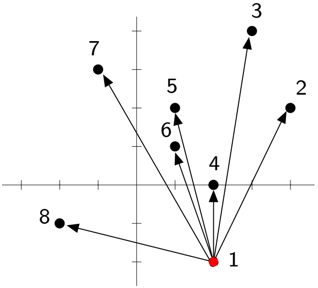
Dir(X, Y, i, j, k)
1 devolva Direita($(X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k])$)

Algoritmo de Graham

Ideia: primeiro fazemos uma "ordenação angular" dos pontos em torno do ponto de menor coordenada Y.

Algoritmo de Graham

Ideia: primeiro fazemos uma "ordenação angular" dos pontos em torno do ponto de menor coordenada Y.



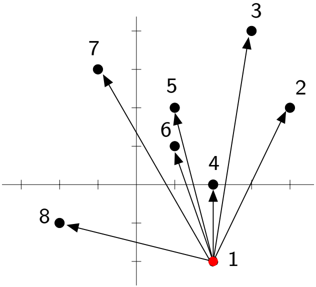
X	1	3	2	-2	2	1	4	-1
Y	1	4	0	-1	-2	2	2	3

Depois deste pré-processamento:

X	2	4	3	2	1	1	-1	-2
Y	-2	2	4	0	2	1	3	-1
	1	2	3	4	5	6	7	8

Algoritmo de Graham

Ideia: primeiro fazemos uma "ordenação angular" dos pontos em torno do ponto de menor coordenada Y.



X	1	3	2	−2	2	1	4	−1
Y	1	4	0	−1	−2	2	2	3

Depois deste pré-processamento:

X	2	4	3	2	1	1	−1	−2
Y	−2	2	4	0	2	1	3	−1
	1	2	3	4	5	6	7	8

Hipótese simplificadora:
a coleção não contém três pontos colineares.

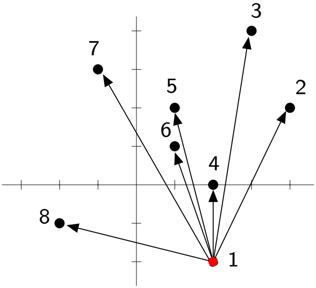
Pré-processamento do Graham

Ordena-G(X, Y, n)

$k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j], 1 \leq j \leq n\}$

$(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$

MergeSort-G$(X, Y, 2, n)$

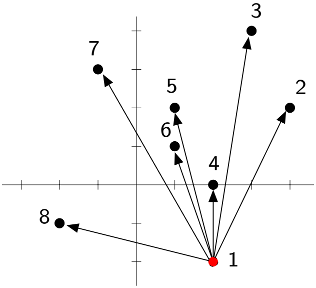


	1	2	3	4	5	6	7	8
X	2	4	3	2	1	1	−1	−2
Y	−2	2	4	0	2	1	3	−1

Pré-processamento do Graham

Ordena-G(X, Y, n)

$k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j], 1 \leq j \leq n\}$
$(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$
MergeSort-G$(X, Y, 2, n)$



X	2	4	3	2	1	1	−1	−2
Y	−2	2	4	0	2	1	3	−1
	1	2	3	4	5	6	7	8

Dir$(X, Y, 1, j, i)$ diz
se o ponto $(X[i], Y[i])$ é "menor" ou não que $(X[j], Y[j])$.

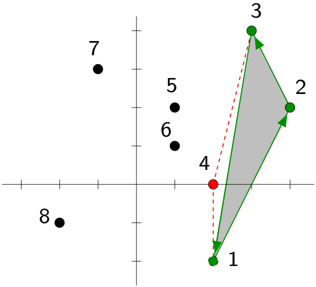
Algoritmo de Graham

Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.

Algoritmo de Graham

Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.

Começamos com os três primeiros pontos.

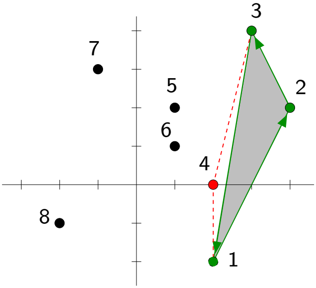


H	1	2	3
	1	2	3

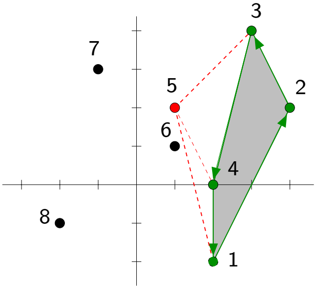
Algoritmo de Graham

Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.

Começamos com os três primeiros pontos.



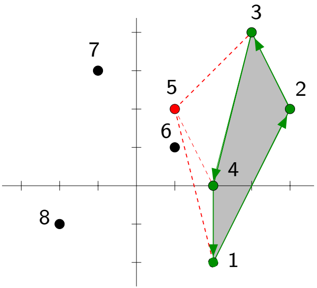
	1	2	3
H	1	2	3



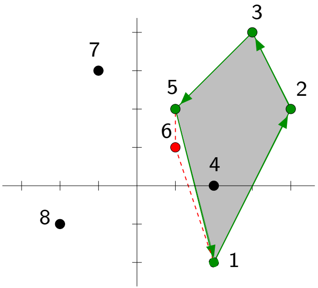
	1	2	3	4
H	1	2	3	4

Algoritmo de Graham

Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.



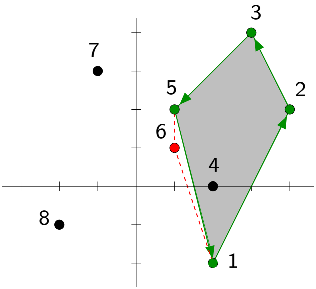
H	1	2	3	4
	1	2	3	4



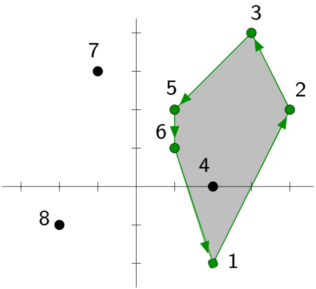
H	1	2	3	5
	1	2	3	4

Algoritmo de Graham

Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.



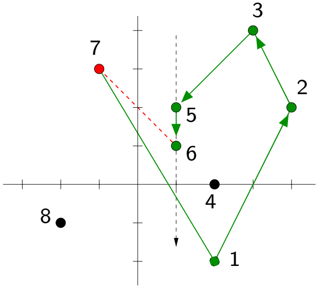
H	1	2	3	5
	1	2	3	4



H	1	2	3	5	6
	1	2	3	4	5

Algoritmo de Graham

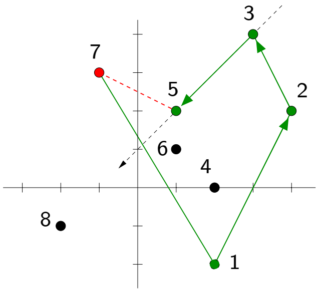
Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.



H					
	1	2	3	5	$\cancel{6}$
	1	2	3	4	5

Algoritmo de Graham

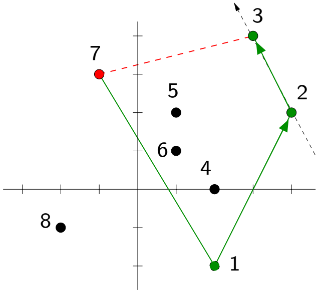
Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.



H	1	2	3	5
	1	2	3	4

Algoritmo de Graham

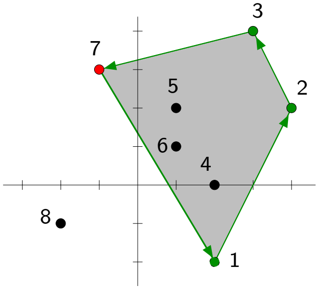
Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.



H			
	1	2	3
	1	2	3

Algoritmo de Graham

Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.



H	1	2	3	7
	1	2	3	4

Algoritmo de Graham

Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.

Graham(X, Y, n)
  Ordena-G(X, Y, n)

Algoritmo de Graham

Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.

Graham(X, Y, n)
1 Ordena-G(X, Y, n)
2 $H[1] \leftarrow 1 \quad H[2] \leftarrow 2 \quad H[3] \leftarrow 3 \quad h \leftarrow 3$

Algoritmo de Graham

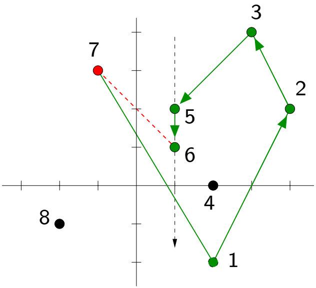
Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.

Graham(X, Y, n)

Ordena-G(X, Y, n)
$H[1] \leftarrow 1 \quad H[2] \leftarrow 2 \quad H[3] \leftarrow 3 \quad h \leftarrow 3$
para $k \leftarrow 4$ até n faça
    $j \leftarrow h$
    enquanto Dir$(X, Y, H[j-1], H[j], k)$ faça
        $j \leftarrow j - 1$
    $h \leftarrow j + 1 \quad H[h] \leftarrow k$
devolva (H, h)

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
    $j \leftarrow h$
    enquanto $\text{Dir}(X, Y, H[j-1], H[j], k)$ faça
        $j \leftarrow j - 1$
    $h \leftarrow j + 1$ $H[h] \leftarrow k$

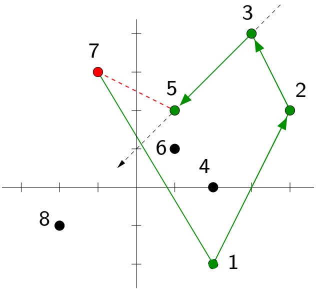


$\text{Dir}(X, Y, 5, 6, 7) =$ verdade

H					
	1	2	3	5	$\cancel{6}$
	1	2	3	4	5

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
    $j \leftarrow h$
    enquanto $\text{Dir}(X, Y, H[j{-}1], H[j], k)$ faça
        $j \leftarrow j - 1$
    $h \leftarrow j + 1 \quad H[h] \leftarrow k$



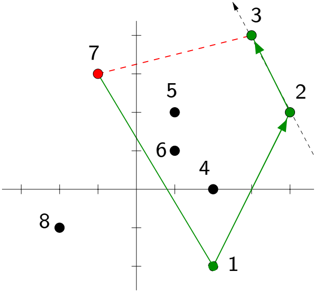
$\text{Dir}(X, Y, 3, 5, 7) =$ verdade

H

1	2	3	4
1	2	3	$\cancel{5}$

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
    $j \leftarrow h$
    enquanto $\mathrm{Dir}(X, Y, H[j-1], H[j], k)$ faça
        $j \leftarrow j - 1$
    $h \leftarrow j + 1 \quad H[h] \leftarrow k$

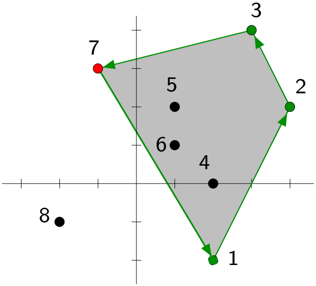


$\mathrm{Dir}(X, Y, 2, 3, 7) =$ falso

H	1	2	3
	1	2	3

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
    $j \leftarrow h$
    enquanto Dir$(X, Y, H[j-1], H[j], k)$ faça
        $j \leftarrow j - 1$
    $h \leftarrow j + 1 \quad H[h] \leftarrow k$

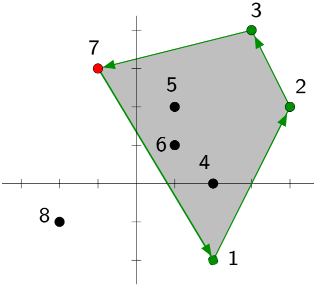


$\text{Esq}(X, Y, 3, 2, 7) = \text{falso}$

H	1	2	3	4
	1	2	3	7

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
  $j \leftarrow h$
  enquanto $\text{Dir}(X, Y, H[j-1], H[j], k)$ faça
    $j \leftarrow j - 1$
  $h \leftarrow j + 1 \quad H[h] \leftarrow k$

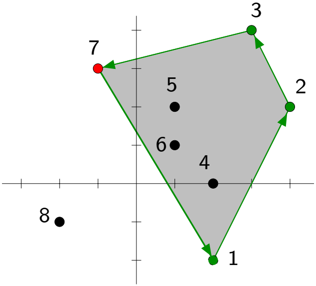


$\text{Esq}(X, Y, 3, 2, 7) =$ falso

H	1	2	3	7
	1	2	3	4

Algoritmo de Graham

```
para k ← 4 até n faça
    j ← h
    enquanto Dir(X, Y, H[j−1], H[j], k) faça
        j ← j − 1
    h ← j + 1     H[h] ← k
```



$\text{Esq}(X, Y, 3, 2, 7) = \text{falso}$

H	1	2	3	7
	1	2	3	4

$H[1\,..\,h]$ funciona como uma pilha.

Algoritmo de Graham

```
Graham(X, Y, n)
Ordena-G(X, Y, n)
H[1] ← 1    H[2] ← 2    H[3] ← 3    h ← 3
para k ← 4 até n faça
    j ← h
    enquanto Dir(X, Y, H[j−1], H[j], k) faça
        j ← j − 1
    h ← j + 1    H[h] ← k
devolva (H, h)
```

Consumo de tempo:

Pré-processamento: $\Theta(n \lg n)$

Algoritmo de Graham

Graham(X, Y, n)

Ordena-G(X, Y, n)
$H[1] \leftarrow 1 \quad H[2] \leftarrow 2 \quad H[3] \leftarrow 3 \quad h \leftarrow 3$
para $k \leftarrow 4$ até n faça
    $j \leftarrow h$
    enquanto Dir($X, Y, H[j-1], H[j], k$) faça
        $j \leftarrow j - 1$
    $h \leftarrow j + 1 \quad H[h] \leftarrow k$
devolva (H, h)

Consumo de tempo:

Pré-processamento: $\Theta(n \lg n)$

Restante: $\Theta(n)$.

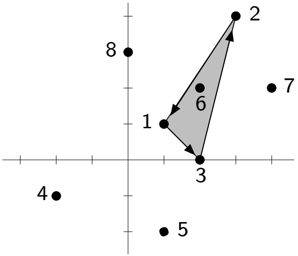
Algoritmo incremental

Ideia: examinar um a um os pontos da coleção,
mantendo o fecho convexo dos pontos já examinados.

Algoritmo incremental

Ideia: examinar um a um os pontos da coleção,
mantendo o fecho convexo dos pontos já examinados.

Começamos com os três primeiros pontos.



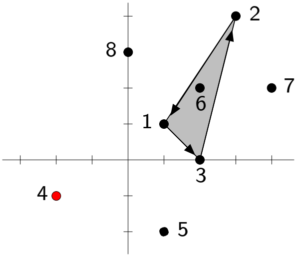
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	1	3	2
	1	2	3

Algoritmo incremental

Ideia: examinar um a um os pontos da coleção, mantendo o fecho convexo dos pontos já examinados.

Começamos com os três primeiros pontos.



	1	2	3	4	5	6	7	8
X	1	3	2	−2	1	2	4	0
Y	1	4	0	−1	−2	2	2	3

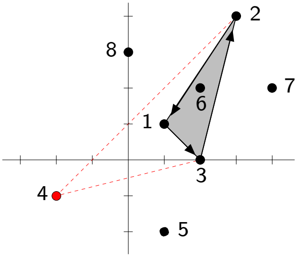
	1	2	3
H	1	3	2

O quarto ponto, $(-2, -1)$, pertence ao fecho corrente?

Algoritmo incremental

Ideia: examinar um a um os pontos da coleção,
mantendo o fecho convexo dos pontos já examinados.

Começamos com os três primeiros pontos.



	1	2	3	4	5	6	7	8
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3

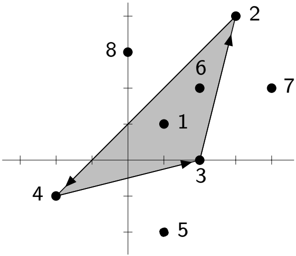
	1	2	3
H	1	3	2

O quarto ponto, $(-2, -1)$, pertence ao fecho corrente? Não.

Algoritmo incremental

Ideia: examinar um a um os pontos da coleção,
mantendo o fecho convexo dos pontos já examinados.

Começamos com os três primeiros pontos.



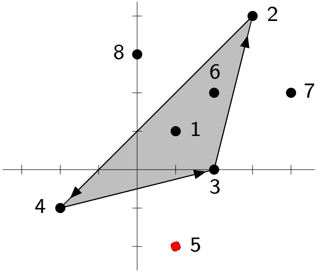
	1	2	3	4	5	6	7	8
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3

	1	2	3
H	3	2	4

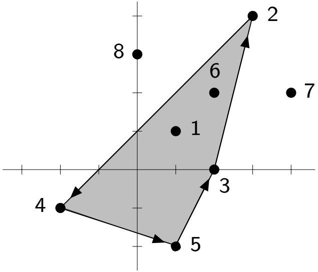
Atualizamos o fecho para incluir o ponto $(-2, -1)$.

Algoritmo incremental

Próximas iterações...



H	3	2	4
	1	2	3

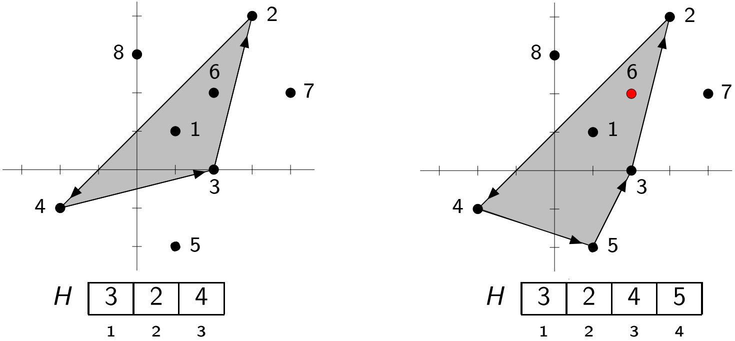


H	3	2	4	5
	1	2	3	4

O quinto ponto, $(1, -2)$, pertence ao fecho corrente? Não.

Algoritmo incremental

Próximas iterações...

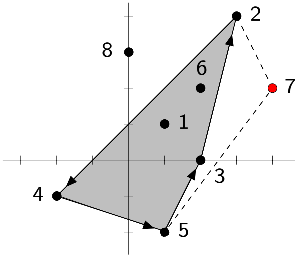


O quinto ponto, $(1, -2)$, pertence ao fecho corrente? Não.

O sexto ponto, $(2, 2)$, pertence ao fecho corrente? Sim.

Algoritmo incremental

Próximas iterações...



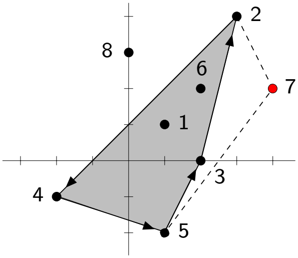
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	3	2	4	5
	1	2	3	4

O sétimo ponto, $(4, 2)$, pertence ao fecho corrente? Não.

Algoritmo incremental

Próximas iterações...



X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	2	4	5	7
	1	2	3	4

O sétimo ponto, $(4, 2)$, pertence ao fecho corrente? Não.
Atualizamos o fecho para incluir o ponto $(4, 2)$.

Algoritmo incremental

Incremental(X, Y, n)

se Esq($X, Y, 1, 2, 3$)
    então $H[1] \leftarrow 1$ $\quad H[2] \leftarrow 2$ $\quad H[3] \leftarrow 3$ $\quad h \leftarrow 3$
    senão $H[1] \leftarrow 1$ $\quad H[2] \leftarrow 3$ $\quad H[3] \leftarrow 2$ $\quad h \leftarrow 3$

Algoritmo incremental

Incremental(X, Y, n)

se Esq($X, Y, 1, 2, 3$)
    então $H[1] \leftarrow 1 \quad H[2] \leftarrow 2 \quad H[3] \leftarrow 3 \quad h \leftarrow 3$
    senão $H[1] \leftarrow 1 \quad H[2] \leftarrow 3 \quad H[3] \leftarrow 2 \quad h \leftarrow 3$
para $k \leftarrow 4$ até n faça
    se não Pertence($H, h, X, Y, X[k], Y[k]$)
        então $(H, h) \leftarrow$ InserePonto(H, h, X, Y, k)
devolva (H, h)

Algoritmo incremental

Incremental(X, Y, n)

se Esq$(X, Y, 1, 2, 3)$
    então $H[1] \leftarrow 1 \quad H[2] \leftarrow 2 \quad H[3] \leftarrow 3 \quad h \leftarrow 3$
    senão $H[1] \leftarrow 1 \quad H[2] \leftarrow 3 \quad H[3] \leftarrow 2 \quad h \leftarrow 3$
para $k \leftarrow 4$ até n faça
    se não Pertence$(H, h, X, Y, X[k], Y[k])$
        então $(H, h) \leftarrow$ InserePonto(H, h, X, Y, k)
devolva (H, h)

Pertence(H, h, X, Y, x, y): devolve verdade se o ponto (x, y) está no fecho convexo, descrito por $H[1 \,.\,.\, h]$, da coleção $X[1 \,.\,.\, k]$, $Y[1 \,.\,.\, k]$ de pontos, falso caso contrário.

Algoritmo incremental

Incremental(X, Y, n)

se Esq$(X, Y, 1, 2, 3)$
    então $H[1] \leftarrow 1$ $\quad H[2] \leftarrow 2$ $\quad H[3] \leftarrow 3$ $\quad h \leftarrow 3$
    senão $H[1] \leftarrow 1$ $\quad H[2] \leftarrow 3$ $\quad H[3] \leftarrow 2$ $\quad h \leftarrow 3$
para $k \leftarrow 4$ até n faça
    se não Pertence$(H, h, X, Y, X[k], Y[k])$
        então $(H, h) \leftarrow$ InserePonto(H, h, X, Y, k)
devolva (H, h)

Pertence(H, h, X, Y, x, y): devolve verdade se o ponto (x, y) está no fecho convexo, descrito por $H[1\,..\,h]$, da coleção $X[1\,..\,k]$, $Y[1\,..\,k]$ de pontos, falso caso contrário.

Consumo de tempo:

Jeito fácil: no pior caso, $\mathrm{O}(h)$.

Algoritmo incremental

```
Incremental(X, Y, n)
se Esq(X, Y, 1, 2, 3)
    então  H[1] ← 1   H[2] ← 2   H[3] ← 3   h ← 3
    senão  H[1] ← 1   H[2] ← 3   H[3] ← 2   h ← 3
para k ← 4 até n faça
    se não Pertence(H, h, X, Y, X[k], Y[k])
        então (H, h) ← InserePonto(H, h, X, Y, k)
devolva (H, h)
```

Pertence(H, h, X, Y, x, y): devolve verdade se o ponto (x, y) está no fecho convexo, descrito por $H[1\,..\,h]$, da coleção $X[1\,..\,k]$, $Y[1\,..\,k]$ de pontos, falso caso contrário.

Consumo de tempo:

Jeito fácil: no pior caso, $\mathrm{O}(h)$.

Com busca binária: no pior caso, $\mathrm{O}(\lg h)$.

Algoritmo incremental

Incremental(X, Y, n)

se Esq$(X, Y, 1, 2, 3)$
    então $H[1] \leftarrow 1 \quad H[2] \leftarrow 2 \quad H[3] \leftarrow 3 \quad h \leftarrow 3$
    senão $H[1] \leftarrow 1 \quad H[2] \leftarrow 3 \quad H[3] \leftarrow 2 \quad h \leftarrow 3$
para $k \leftarrow 4$ até n faça
    se não Pertence$(H, h, X, Y, X[k], Y[k])$
        então $(H, h) \leftarrow$ InserePonto(H, h, X, Y, k)
devolva (H, h)

InserePonto(H, h, X, Y, k): recebe o fecho convexo $H[1\,..\,h]$ da coleção $X[1\,..\,k{-}1]$, $Y[1\,..\,k{-}1]$ de pontos e devolve o fecho convexo da coleção $X[1\,..\,k]$, $Y[1\,..\,k]$.

Algoritmo incremental

Incremental(X, Y, n)

se Esq$(X, Y, 1, 2, 3)$
    então $H[1] \leftarrow 1$ $\quad H[2] \leftarrow 2$ $\quad H[3] \leftarrow 3$ $\quad h \leftarrow 3$
    senão $H[1] \leftarrow 1$ $\quad H[2] \leftarrow 3$ $\quad H[3] \leftarrow 2$ $\quad h \leftarrow 3$
para $k \leftarrow 4$ até n faça
    se não Pertence$(H, h, X, Y, X[k], Y[k])$
        então $(H, h) \leftarrow$ InserePonto(H, h, X, Y, k)
devolva (H, h)

InserePonto(H, h, X, Y, k): recebe o fecho convexo $H[1\,..\,h]$ da coleção $X[1\,..\,k-1]$, $Y[1\,..\,k-1]$ de pontos e devolve o fecho convexo da coleção $X[1\,..\,k]$, $Y[1\,..\,k]$.

Consumo de tempo: no pior caso, $\Theta(h)$.

Algoritmo incremental

Incremental(X, Y, n)

1 se Esq$(X, Y, 1, 2, 3)$
2 então $H[1] \leftarrow 1$ $H[2] \leftarrow 2$ $H[3] \leftarrow 3$ $h \leftarrow 3$
3 senão $H[1] \leftarrow 1$ $H[2] \leftarrow 3$ $H[3] \leftarrow 2$ $h \leftarrow 3$
4 para $k \leftarrow 4$ até n faça
5 se não Pertence$(H, h, X, Y, X[k], Y[k])$
6 então $(H, h) \leftarrow$ InserePonto(H, h, X, Y, k)
7 devolva (H, h)

Invariante do para da linha 4:

$H[1\,..\,h]$ é fecho convexo da coleção $X[1\,..\,k-1]$, $Y[1\,..\,k-1]$.

Algoritmo incremental

Incremental(X, Y, n)
1 se Esq$(X, Y, 1, 2, 3)$
2 então $H[1] \leftarrow 1$ $H[2] \leftarrow 2$ $H[3] \leftarrow 3$ $h \leftarrow 3$
3 senão $H[1] \leftarrow 1$ $H[2] \leftarrow 3$ $H[3] \leftarrow 2$ $h \leftarrow 3$
4 para $k \leftarrow 4$ até n faça
5 se não Pertence$(H, h, X, Y, X[k], Y[k])$
6 então $(H, h) \leftarrow$ InserePonto(H, h, X, Y, k)
7 devolva (H, h)

Invariante do para da linha 4:
$H[1\,..\,h]$ é fecho convexo da coleção $X[1\,..\,k-1], Y[1\,..\,k-1]$.

Consumo de tempo: no pior caso, $\Theta(n^2)$, pois $h \leq n$.

Teste linear de pertinência a polígono convexo

Pertence(H, h, X, Y, x, y)

$H[h+1] \leftarrow H[1]$ ▷ sentinela

para $i \leftarrow 1$ até h faça

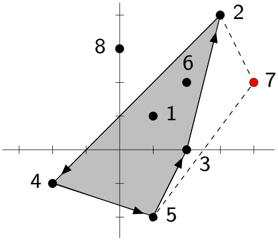
  se Direita($X[H[i]], Y[H[i]], X[H[i+1]], Y[H[i+1]], x, y$)

    então devolva falso

devolva verdade

Teste linear de pertinência a polígono convexo

Pertence(H, h, X, Y, x, y)
$H[h+1] \leftarrow H[1]$ ▷ sentinela
para $i \leftarrow 1$ até h faça
    se Direita($X[H[i]], Y[H[i]], X[H[i+1]], Y[H[i+1]], x, y$)
        então devolva falso
devolva verdade



	1	2	3	4	5	6	7	8
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3

	1	2	3	4
H	3	2	4	5

Pertence($H, 4, X, Y, 2, 2$) $=$ verdade
Pertence($H, 4, X, Y, 4, 2$) $=$ falso

Teste linear de pertinência a polígono convexo

Pertence(H, h, X, Y, x, y)
1 $H[h+1] \leftarrow H[1]$ $\triangleright$ sentinela
2 para $i \leftarrow 1$ até h faça
3 se Direita($X[H[i]], Y[H[i]], X[H[i+1]], Y[H[i+1]], x, y$)
4 então devolva falso
5 devolva verdade

Consumo de tempo: $\Theta(h)$, ou seja, linear.

Teste linear de pertinência a polígono convexo

```
Pertence(H, h, X, Y, x, y)
  H[h+1] ← H[1]     ▷ sentinela
  para i ← 1 até h faça
      se Direita(X[H[i]], Y[H[i]], X[H[i+1]], Y[H[i+1]], x, y)
          então devolva falso
  devolva verdade
```

Consumo de tempo: $\Theta(h)$, ou seja, linear.

Dá para fazer em $\Theta(\lg h)$, ou seja, logarítmico...

Teste linear de pertinência a polígono convexo

Pertence(H, h, X, Y, x, y)

```
H[h+1] ← H[1]    ▷ sentinela
para i ← 1 até h faça
    se Direita(X[H[i]], Y[H[i]], X[H[i+1]], Y[H[i+1]], x, y)
        então devolva falso
devolva verdade
```

Consumo de tempo: $\Theta(h)$, ou seja, linear.

Dá para fazer em $\Theta(\lg h)$, ou seja, logarítmico... com busca binária!

Se Esquerda$(X[H[1]], Y[H[1]], X[H[2]], Y[H[2]], x, y)$ e
Esquerda$(X[H[\frac{h}{2}]], Y[H[\frac{h}{2}]], X[H[\frac{h}{2}+1]], Y[H[\frac{h}{2}+1]], x, y)$

Teste linear de pertinência a polígono convexo

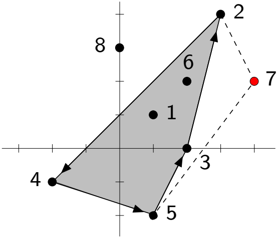
```
Pertence(H, h, X, Y, x, y)
  H[h+1] ← H[1]      ▷ sentinela
  para i ← 1 até h faça
      se Direita(X[H[i]], Y[H[i]], X[H[i+1]], Y[H[i+1]], x, y)
          então devolva falso
  devolva verdade
```

Consumo de tempo: $\Theta(h)$, ou seja, linear.

Dá para fazer em $\Theta(\lg h)$, ou seja, logarítmico... com busca binária!

Se Esquerda$(X[H[1]], Y[H[1]], X[H[2]], Y[H[2]], x, y)$ e
Esquerda$(X[H[\frac{h}{2}]], Y[H[\frac{h}{2}]], X[H[\frac{h}{2}+1]], Y[H[\frac{h}{2}+1]], x, y)$
então se Esquerda$(x, y, X[H[2]], Y[H[2]], X[H[h/2]], Y[H[h/2]])$
então joga fora o trecho de 2 a $h/2$
senão joga fora o trecho de $h/2 + 1$ a 1.

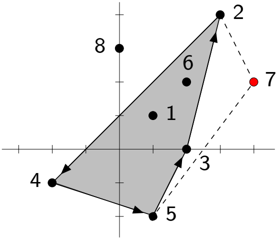
Insere Ponto



O InserePonto tem que encontrar os pontos 2 e 5 acima.

Que características estes pontos têm?

Insere Ponto

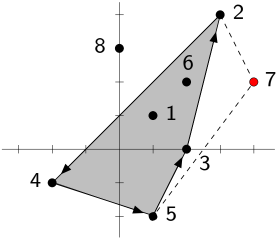


O InserePonto tem que encontrar os pontos 2 e 5 acima.

Que características estes pontos têm?

Cada uma das arestas incidentes a eles deixa o ponto 7 de um lado diferente: uma à esquerda, a outra à direita.

Insere Ponto



O InserePonto tem que encontrar os pontos 2 e 5 acima.

Que características estes pontos têm?

Cada uma das arestas incidentes a eles deixa o ponto 7 de um lado diferente: uma à esquerda, a outra à direita.

A fronteira do fecho atualizado consiste num dos trechos de 2 e 5 junto com o 7.

No algoritmo à frente, $H[i] = 2$ e $H[j] = 5$ na linha 9.

Insere Ponto

InserePonto$(H, h, X, Y, {\color{red}k})$

$H[0] \leftarrow H[h] \quad H[h{+}1] \leftarrow H[1]$ $\quad \triangleright$ sentinelas
$i \leftarrow 1$
enquanto $\text{Esq}(X, Y, H[i{-}1], H[i], {\color{red}k}) = \text{Esq}(X, Y, H[i], H[i{+}1], {\color{red}k})$ faça
    $i \leftarrow i + 1$
$j \leftarrow i + 1$
enquanto $\text{Esq}(X, Y, H[j{-}1], H[j], {\color{red}k}) = \text{Esq}(X, Y, H[j], H[j{+}1], {\color{red}k})$ faça
    $j \leftarrow j + 1$
se $\text{Esq}(X, Y, H[i{-}1], H[i], {\color{red}k})$ então $i \leftrightarrow j$

Insere Ponto

InserePonto(H, h, X, Y, k)

$H[0] \leftarrow H[h] \quad H[h{+}1] \leftarrow H[1]$ $\triangleright$ sentinelas

$i \leftarrow 1$

enquanto Esq$(X, Y, H[i{-}1], H[i], k) =$ Esq$(X, Y, H[i], H[i{+}1], k)$ faça

  $i \leftarrow i + 1$

$j \leftarrow i + 1$

enquanto Esq$(X, Y, H[j{-}1], H[j], k) =$ Esq$(X, Y, H[j], H[j{+}1], k)$ faça

  $j \leftarrow j + 1$

se Esq$(X, Y, H[i{-}1], H[i], k)$ então $i \leftrightarrow j$

$t \leftarrow 1$

enquanto $i \neq j$ faça

  $F[t] \leftarrow H[i] \quad t \leftarrow t + 1 \quad i \leftarrow (i \bmod h) + 1$

$F[t] \leftarrow H[i] \quad t \leftarrow t + 1 \quad F[t] \leftarrow k$

devolva (F, t)

Insere Ponto

InserePonto(H, h, X, Y, k)

1 $H[0] \leftarrow H[h]$ $\quad H[h{+}1] \leftarrow H[1]$ $\quad \triangleright$ sentinelas

2 $i \leftarrow 1$

3 enquanto Esq$(X, Y, H[i{-}1], H[i], k) =$ Esq$(X, Y, H[i], H[i{+}1], k)$ faça

4 $\quad i \leftarrow i + 1$

5 $j \leftarrow i + 1$

6 enquanto Esq$(X, Y, H[j{-}1], H[j], k) =$ Esq$(X, Y, H[j], H[j{+}1], k)$ faça

7 $\quad j \leftarrow j + 1$

8 se Esq$(X, Y, H[i{-}1], H[i], k)$ então $i \leftrightarrow j$

9 $t \leftarrow 1$

10 enquanto $i \neq j$ faça

11 $\quad F[t] \leftarrow H[i]$ $\quad t \leftarrow t + 1$ $\quad i \leftarrow (i \bmod h) + 1$

12 $F[t] \leftarrow H[i]$ $\quad t \leftarrow t + 1$ $\quad F[t] \leftarrow k$

13 devolva (F, t)

As linhas 3-4 e 6-7 podem ser melhoradas,
e feitas em $\mathrm{O}(\lg h)$ com busca binária.

Casos degenerados

Como tratar os casos degenerados nos dois algoritmos anteriores?

Hipótese simplificadora:
a coleção não contém três pontos colineares.

Casos degenerados

Como tratar os casos degenerados nos dois algoritmos anteriores?

Hipótese simplificadora:
a coleção não contém três pontos colineares.

O que fazer quando há três ou mais pontos colineares?

Casos degenerados: Graham

Hipótese simplificadora:

a coleção não contém três pontos colineares.

Casos degenerados: Graham

Hipótese simplificadora:
a coleção não contém três pontos colineares.

Pré-processamento: ordenação angular dos pontos em torno do ponto de menor coordenada Y.

Ordena-G(X, Y, n)
$k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j], 1 \leq j \leq n\}$
$(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$
MergeSort-G($X, Y, 2, n$)

Casos degenerados: Graham

Hipótese simplificadora:
a coleção não contém três pontos colineares.

Pré-processamento: ordenação angular dos pontos
em torno do ponto de menor coordenada Y.

Ordena-G(X, Y, n)
1 $k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j], 1 \leq j \leq n\}$
2 $(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$
3 MergeSort-G($X, Y, 2, n$)

Se houver três pontos colineares, tome como k
um ponto com Y mínimo e, dentre estes, o de X mínimo.

Casos degenerados: Graham

Hipótese simplificadora:
a coleção não contém três pontos colineares.

Pré-processamento: ordenação angular dos pontos
em torno do ponto de menor coordenada Y.

Ordena-G(X, Y, n)
1 $k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j], 1 \leq j \leq n\}$
2 $(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$
3 MergeSort-G($X, Y, 2, n$)

Se houver três pontos colineares, tome como k
um ponto com Y mínimo e, dentre estes, o de X mínimo.

1 $k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j]$ ou
$(Y[i] = Y[j]$ e $X[i] < X[j]),\ 1 \leq j \leq n\}$

Casos degenerados: Graham

Pré-processamento:

Ordena-G(X, Y, n)
1 $k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j]$ ou
$(Y[i] = Y[j]$ e $X[i] < X[j]),\ \ 1 \leq j \leq n\}$
2 $(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$
3 MergeSort-G($X, Y, 2, n$)

Casos degenerados: Graham

Pré-processamento:

Ordena-G(X, Y, n)

$k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j]$ ou
$\quad (Y[i] = Y[j]$ e $X[i] < X[j]),\ 1 \leq j \leq n\}$

$(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$

MergeSort-G$(X, Y, 2, n)$

Na ordenação, se houver três pontos colineares,
quando há empate no ângulo, desempate pela distância a k:
os com distância menor primeiro na ordem.

Casos degenerados: Graham

Pré-processamento:

Ordena-G(X, Y, n)

$k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j]$ ou
$(Y[i] = Y[j]$ e $X[i] < X[j]),\ 1 \leq j \leq n\}$

$(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$

MergeSort-G$(X, Y, 2, n)$

Na ordenação, se houver três pontos colineares, quando há empate no ângulo, desempate pela distância a k: os com distância menor primeiro na ordem.

No laço principal, desempilhe enquanto estiver à direita...

enquanto Dir$(X, Y, H[j-1], H[j], k)$ faça
    $j \leftarrow j - 1$

Casos degenerados: Graham

Pré-processamento:

Ordena-G(X, Y, n)
$k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \le Y[j]$ ou
$(Y[i] = Y[j]$ e $X[i] < X[j]),\ \ 1 \le j \le n\}$
$(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$
MergeSort-G($X, Y, 2, n$)

Na ordenação, se houver três pontos colineares,
quando há empate no ângulo, desempate pela distância a k:
os com distância menor primeiro na ordem.

No laço principal, desempilhe enquanto estiver à direita...

enquanto Dir($X, Y, H[j-1], H[j], k$) faça
$\quad j \leftarrow j - 1$

... ou for colinear.

Casos degenerados: Incremental

Inicialização:

Incremental(X, Y, n)

se Esq($X, Y, 1, 2, 3$)
    então $H[1] \leftarrow 1$ $H[2] \leftarrow 2$ $H[3] \leftarrow 3$ $h \leftarrow 3$
    senão $H[1] \leftarrow 1$ $H[2] \leftarrow 3$ $H[3] \leftarrow 2$ $h \leftarrow 3$

Casos degenerados: Incremental

Inicialização:

Incremental(X, Y, n)

1 se Esq$(X, Y, 1, 2, 3)$

2 então $H[1] \leftarrow 1$ $H[2] \leftarrow 2$ $H[3] \leftarrow 3$ $h \leftarrow 3$

3 senão $H[1] \leftarrow 1$ $H[2] \leftarrow 3$ $H[3] \leftarrow 2$ $h \leftarrow 3$

Percorrer os pontos até encontrar três não colineares.

Casos degenerados: Incremental

Inicialização:

Incremental(X, Y, n)

1 se Esq$(X, Y, 1, 2, 3)$

2 então $H[1] \leftarrow 1$ $H[2] \leftarrow 2$ $H[3] \leftarrow 3$ $h \leftarrow 3$

3 senão $H[1] \leftarrow 1$ $H[2] \leftarrow 3$ $H[3] \leftarrow 2$ $h \leftarrow 3$

Percorrer os pontos até encontrar três não colineares.

Se todos são colineares, o fecho consistirá de dois pontos extremos.

Casos degenerados: Incremental

Processamento dos demais pontos:

```
para k ← 4 até n faça
    se não Pertence(H, h, X, Y, X[k], Y[k])
        então (H, h) ← InserePonto(H, h, X, Y, k)
devolva (H, h)
```

Casos degenerados: Incremental

Processamento dos demais pontos:

para $k \leftarrow 4$ até n faça
    se não Pertence($H, h, X, Y, X[k], Y[k]$)
        então $(H, h) \leftarrow$ InserePonto(H, h, X, Y, k)
devolva (H, h)

Pertence(H, h, X, Y, x, y):
Se o ponto (x, y) estiver à direta estrita de alguma aresta do fecho, ele não percente ao fecho.

Casos degenerados: Incremental

Processamento dos demais pontos:

```
para k ← 4 até n faça
    se não Pertence(H, h, X, Y, X[k], Y[k])
        então (H, h) ← InserePonto(H, h, X, Y, k)
devolva (H, h)
```

Pertence(H, h, X, Y, x, y):
Se o ponto (x, y) estiver à direta estrita de alguma aresta do fecho, ele não percente ao fecho.

InserePonto(H, h, X, Y, k):
Os vértices de inflexão do fecho são tais que o ponto k está à esquerda estrita de uma de suas arestas e à direita da outra.

Moral da história

Pense nas primitivas que o problema exige.

Pense nas hipóteses simplificadoras:
quais condições representam posição geral para o problema.

Resolva o problema em posição geral.

Pense em como lidar com os casos degenerados,
que infringem as hipóteses simplificadoras,
modificando minimamente o seu algoritmo.

Moral da história

Pense nas primitivas que o problema exige.

Pense nas hipóteses simplificadoras:
quais condições representam posição geral para o problema.

Resolva o problema em posição geral.

Pense em como lidar com os casos degenerados,
que infringem as hipóteses simplificadoras,
modificando minimamente o seu algoritmo.

Sempre que possível, trabalhe com inteiros.

Caso necessite usar floats,
trabalhe com precisão ϵ nas comparações.