

Aula 17

Fecho convexo

Sec 7.4 do FP (JAI 2009), exceto pela sec 7.4.6.

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 $\alpha_j \geq 0$, para $i = 1, \dots, n$, 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 $\alpha_j \geq 0$, para $i = 1, \dots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



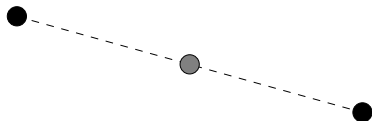
Combinação convexa

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 $\alpha_j \geq 0$, para $i = 1, \dots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



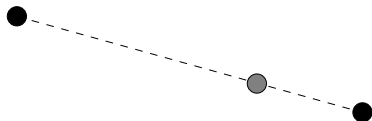
Combinação convexa

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 $\alpha_j \geq 0$, para $i = 1, \dots, n$, 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 $\alpha_j \geq 0$, para $i = 1, \dots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



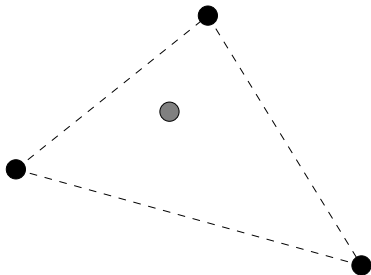
Combinação convexa

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 $\alpha_j \geq 0$, para $i = 1, \dots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



Fecho convexo

Fecho convexo de P :

conjunto de combinações convexas de pontos de P , ou seja,

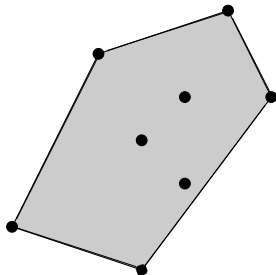
$$\text{conv}(P) := \left\{ \alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]) : \right. \\ \left. \alpha_1 + \cdots + \alpha_n = 1, \text{ e } \alpha_i \geq 0 \ (i = 1, \dots, n) \right\}.$$

Fecho convexo

Fecho convexo de P :

conjunto de combinações convexas de pontos de P , ou seja,

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

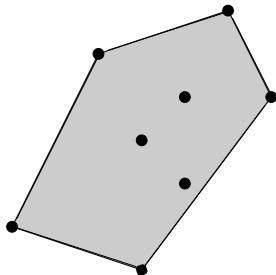


Fecho convexo

Fecho convexo de P :

conjunto de combinações convexas de pontos de P , ou seja,

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



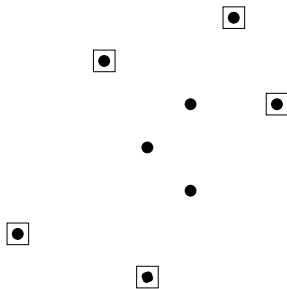
Problema: Dada uma coleção P de pontos do plano, determinar o fecho convexo de P .

Pontos extremos

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

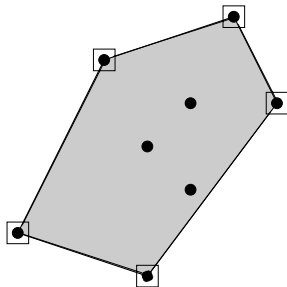
Pontos extremos

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



Pontos extremos

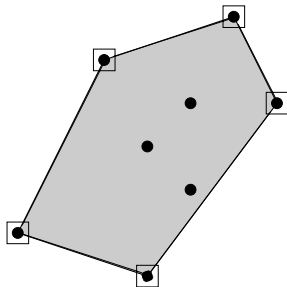
Ponto (x, y) de P é **extremo** se não é combinação convexa de pontos de $P \setminus \{(x, y)\}$.



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

Hipótese simplificadora:

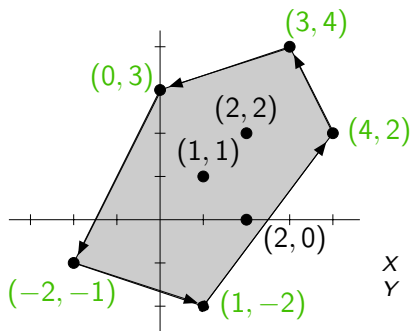
a coleção não contém três pontos colineares.

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**).

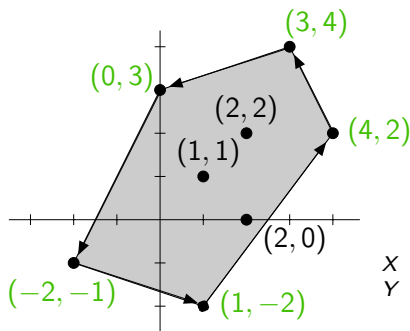


X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	2	8	4	5	7
	1	2	3	4	5

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**).



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

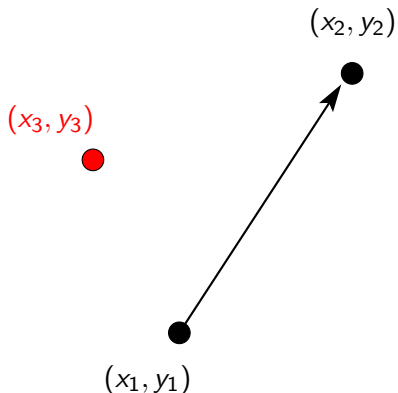
H	2	8	4	5	7
	1	2	3	4	5

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

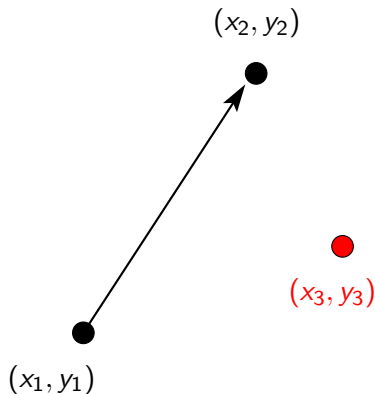
Direita((x_1, y_1) , (x_2, y_2) , (x_3, y_3)) = falso



Predicados geométricos

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

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



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:

$\text{Esq}(X, Y, i, j, k)$

1 devolva $\text{Esquerda}^+((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

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:

$\text{Esq}(X, Y, i, j, k)$

1 devolva $\text{Esquerda}^+((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

Similarmente

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

Em pseudocódigo:

$\text{Dir}(X, Y, i, j, k)$

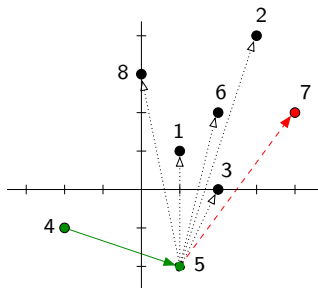
1 devolva $\text{Direita}^+((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

Embrulho de presente

Ideia: repetidamente, a partir de um ponto extremo do fecho convexo, encontrar o próximo no sentido anti-horário.

Embrulho de presente

Ideia: repetidamente, a partir de um ponto extremo do fecho convexo, encontrar o próximo no sentido anti-horário.



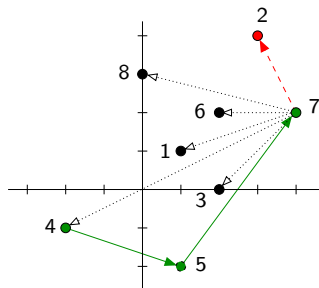
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	4	5	?	...
	1	2	3	4

$\text{Dir}(X, Y, 5, 7, j) = \text{falso}$ para $j = 1, \dots, 8$

Embrulho de presente

Ideia: repetidamente, a partir de um ponto extremo do fecho convexo, encontrar o próximo no sentido anti-horário.



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

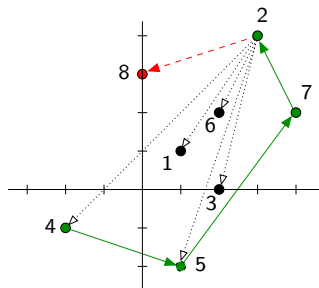
H	4	5	7	?	...
	1	2	3	4	5

$\text{Dir}(X, Y, 5, 7, j) = \text{falso}$ para $j = 1, \dots, 8$

$\text{Dir}(X, Y, 7, 2, j) = \text{falso}$ para $j = 1, \dots, 8$

Embrulho de presente

Ideia: repetidamente, a partir de um ponto extremo do fecho convexo, encontrar o próximo no sentido anti-horário.



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	4	5	7	2	?	...
	1	2	3	4	5	6

$\text{Dir}(X, Y, 5, 7, j) = \text{falso}$ para $j = 1, \dots, 8$

$\text{Dir}(X, Y, 7, 2, j) = \text{falso}$ para $j = 1, \dots, 8$

$\text{Dir}(X, Y, 2, 8, j) = \text{falso}$ para $j = 1, \dots, 8$

Embrulho de presente

Ideia: repetidamente, a partir de um ponto extremo do fecho convexo, encontrar o próximo no sentido anti-horário.

Embrulho de presente

Ideia: repetidamente, a partir de um ponto extremo do fecho convexo, encontrar o próximo no sentido anti-horário.

Embrulho(X, Y, n)

- 1 $h \leftarrow 0$
- 2 $H[0] \leftarrow \min\{i \in [1..n] : X[i] \leq X[j], 1 \leq j \leq n\}$ \triangleright sentinela
- 3 repita
- 4 $i \leftarrow (H[h] \bmod n) + 1$ \triangleright qualquer ponto distinto de $H[h]$
- 5 para $j \leftarrow 1$ até n faça
- 6 se $\text{Dir}(X, Y, H[h], i, j)$ então $i \leftarrow j$
- 7 $h \leftarrow h + 1$
- 8 $H[h] \leftarrow i$
- 9 até que $i = H[0]$ \triangleright fechou o polígono
- 10 devolva (H, h)

Embrulho de presente

Ideia: repetidamente, a partir de um ponto extremo do fecho convexo, encontrar o próximo no sentido anti-horário.

Embrulho(X, Y, n)

- 1 $h \leftarrow 0$
- 2 $H[0] \leftarrow \min\{i \in [1..n] : X[i] \leq X[j], 1 \leq j \leq n\}$ \triangleright sentinela
- 3 repita
- 4 $i \leftarrow (H[h] \bmod n) + 1$ \triangleright qualquer ponto distinto de $H[h]$
- 5 para $j \leftarrow 1$ até n faça
- 6 se $\text{Dir}(X, Y, H[h], i, j)$ então $i \leftarrow j$
- 7 $h \leftarrow h + 1$
- 8 $H[h] \leftarrow i$
- 9 até que $i = H[0]$ \triangleright fechou o polígono
- 10 devolva (H, h)

Consumo de tempo: $\Theta(nh)$,

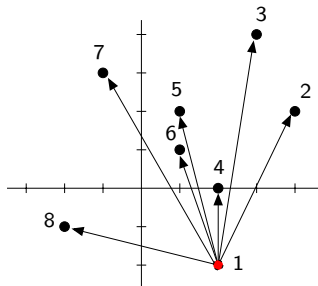
onde h é o número de pontos no fecho convexo.

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

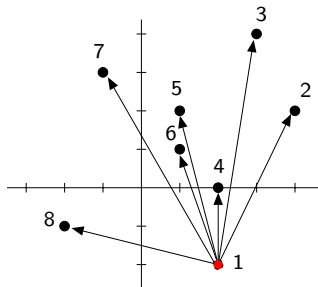
Depois deste pré-processamento:

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

Pré-processamento do Graham

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($X, Y, 2, n$)

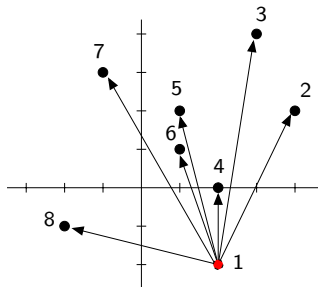


X	2	4	3	2	1	1	-1	-2
Y	-2	2	4	0	2	1	3	-1
	1	2	3	4	5	6	7	8

Pré-processamento do Graham

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($X, Y, 2, n$)



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

$\text{Dir}(X, Y, 1, j, i)$ diz

se o ponto $(X[i], Y[i])$ é “menor” ou não que $(X[j], Y[j])$.

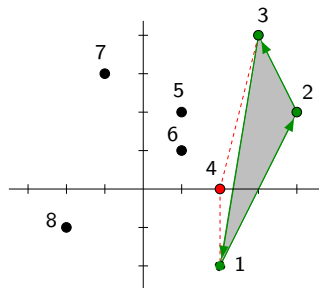
Algoritmo de Graham

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.



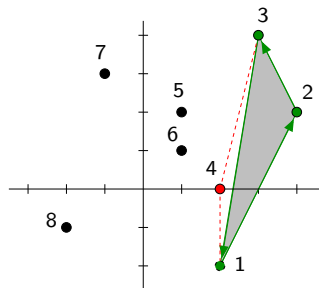
H

1	2	3
1	2	3

Algoritmo de Graham

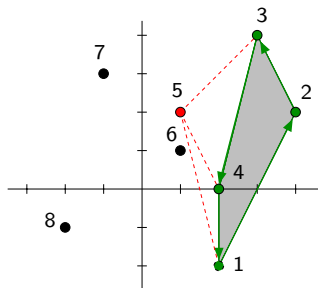
Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.

Começamos com os três primeiros pontos.



H

1	2	3
1	2	3

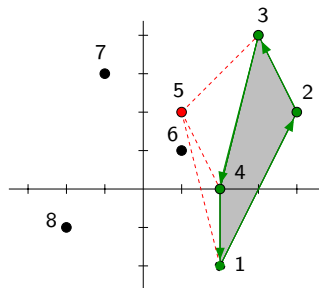


H

1	2	3	4
1	2	3	4

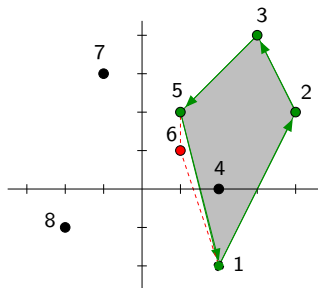
Algoritmo de Graham

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

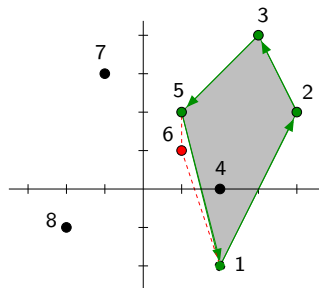


H

1	2	3	5
1	2	3	4

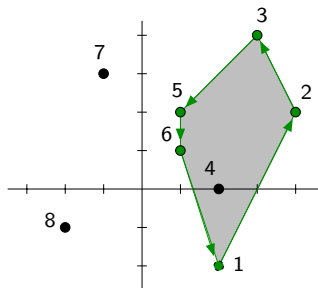
Algoritmo de Graham

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.

Graham(X, Y, n)

1 Ordena-G(X, Y, n)

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

3 para $k \leftarrow 4$ até n faça

4 $j \leftarrow h$

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

6 $j \leftarrow j - 1$

7 $h \leftarrow j + 1$ $H[h] \leftarrow k$

8 devolva (H, h)

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 $H[1] \leftarrow 1$ $H[2] \leftarrow 2$ $H[3] \leftarrow 3$ $h \leftarrow 3$
- 3 para $k \leftarrow 4$ até n faça
- 4 $j \leftarrow h$
- 5 enquanto Dir($X, Y, H[j-1], H[j], k$) faça
- 6 $j \leftarrow j - 1$
- 7 $h \leftarrow j + 1$ $H[h] \leftarrow k$
- 8 devolva (H, h)

Consumo de tempo:

Pré-processamento: $\Theta(n \lg n)$

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

3 para $k \leftarrow 4$ até n faça

4 $j \leftarrow h$

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

6 $j \leftarrow j - 1$

7 $h \leftarrow j + 1$ $H[h] \leftarrow k$

8 devolva (H, h)

Consumo de tempo:

Pré-processamento: $\Theta(n \lg n)$

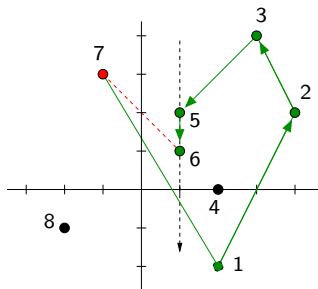
Restante: $\Theta(n)$.

Algoritmo de Graham

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

Algoritmo de Graham

- 3 para $k \leftarrow 4$ até n faça
- 4 $j \leftarrow h$
- 5 enquanto $\text{Dir}(X, Y, H[j-1], H[j], k)$ faça
- 6 $j \leftarrow j - 1$
- 7 $h \leftarrow j + 1$ $H[h] \leftarrow k$

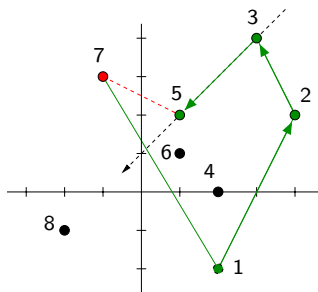


$\text{Dir}(X, Y, 5, 6, 7) = \text{verdade}$

H	1	2	3	5	6
	1	2	3	4	5

Algoritmo de Graham

- 3 para $k \leftarrow 4$ até n faça
- 4 $j \leftarrow h$
- 5 enquanto $\text{Dir}(X, Y, H[j-1], H[j], k)$ faça
- 6 $j \leftarrow j - 1$
- 7 $h \leftarrow j + 1$ $H[h] \leftarrow k$



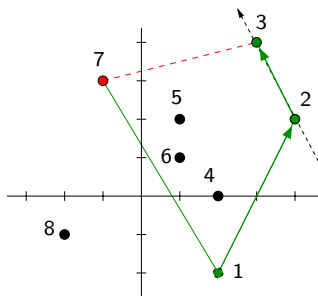
$\text{Dir}(X, Y, 3, 5, 7) = \text{verdade}$

H

1	2	3	5
1	2	3	4

Algoritmo de Graham

- 3 para $k \leftarrow 4$ até n faça
- 4 $j \leftarrow h$
- 5 enquanto $\text{Dir}(X, Y, H[j-1], H[j], k)$ faça
- 6 $j \leftarrow j - 1$
- 7 $h \leftarrow j + 1$ $H[h] \leftarrow k$



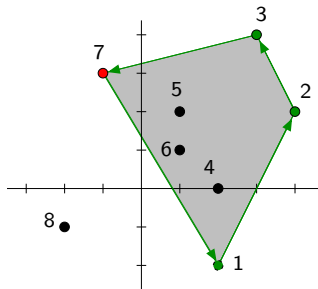
$\text{Dir}(X, Y, 2, 3, 7) = \text{falso}$

H

1	2	3
1	2	3

Algoritmo de Graham

- 3 para $k \leftarrow 4$ até n faça
- 4 $j \leftarrow h$
- 5 enquanto $\text{Dir}(X, Y, H[j-1], H[j], k)$ faça
- 6 $j \leftarrow j - 1$
- 7 $h \leftarrow j + 1$ $H[h] \leftarrow k$

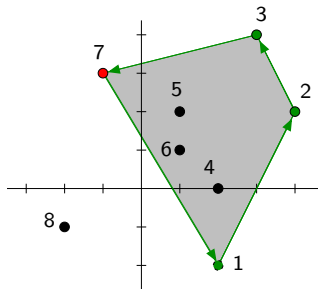


$\text{Esq}(X, Y, 3, 2, 7) = \text{falso}$

H	1	2	3	7
	1	2	3	4

Algoritmo de Graham

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



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

H	1	2	3	7
	1	2	3	4

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

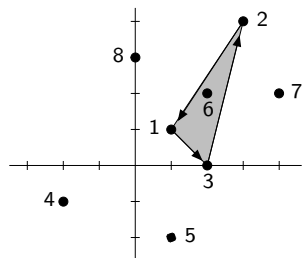
Um algoritmo incremental

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

Um 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.



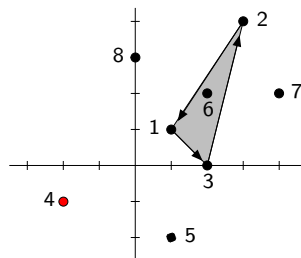
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	1	3	2
	1	2	3

Um 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

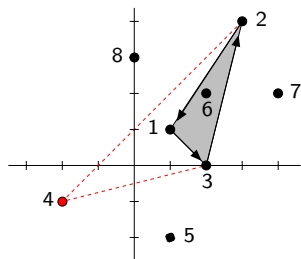
H	1	3	2
	1	2	3

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

Um 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

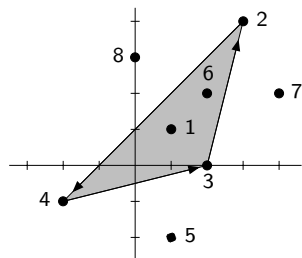
H	1	3	2
	1	2	3

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

Um 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.



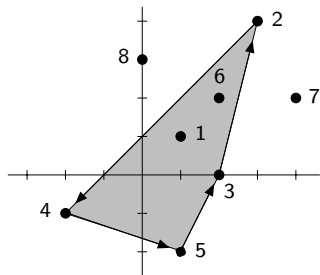
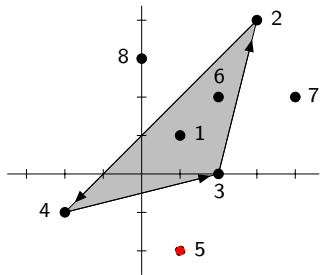
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	3	2	4
	1	2	3

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

Um algoritmo incremental

Próximas iterações...



H

3	2	4	5
1	2	3	4

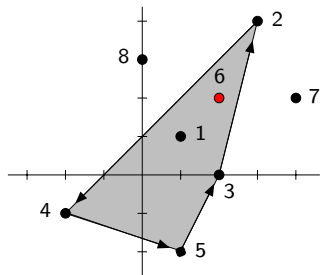
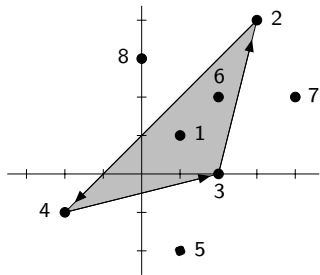
H

3	2	4
1	2	3

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

Um algoritmo incremental

Próximas iterações...



H

3	2	4	5
1	2	3	4

H

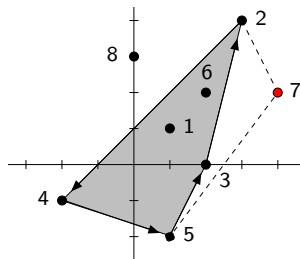
3	2	4
1	2	3

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

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

Um algoritmo incremental

Próximas iterações...



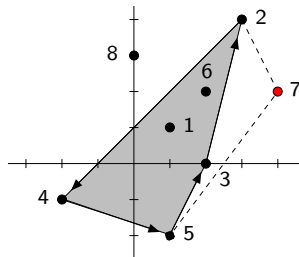
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	3	2	4	5
	1	2	3	4

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

Um 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 $(4, 2)$.

Um algoritmo incremental

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

Um algoritmo incremental

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

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  $(H, h)$ 
```

Um algoritmo incremental

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

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  $(H, h) \leftarrow$  InsererPonto( $H, h, X, Y, k$ )
7 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.

Um algoritmo incremental

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

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  $(H, h) \leftarrow$  InsererPonto( $H, h, X, Y, k$ )
7 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: no pior caso, $O(h)$.

Um algoritmo incremental

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

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  $(H, h) \leftarrow$  InsererPonto( $H, h, X, Y, k$ )
7 devolva  $(H, h)$ 
```

InsererPonto(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 $X[1..k], Y[1..k]$.

Um algoritmo incremental

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

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  $(H, h) \leftarrow$  InsererPonto( $H, h, X, Y, k$ )
7 devolva  $(H, h)$ 
```

InsererPonto(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, $\Theta(h)$.

Um algoritmo incremental

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

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 $X[1..k-1], Y[1..k-1]$.

Um algoritmo incremental

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

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 $h \leq n$.

Pertence

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

1 $H[h+1] \leftarrow H[1]$ ▷ sentinela

2 para $i \leftarrow 1$ até h faça

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

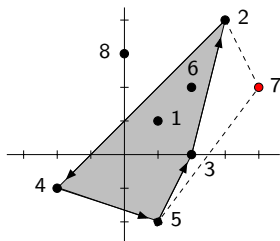
4 então devolva falso

5 devolva verdade

Pertence

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

- 1 $H[h+1] \leftarrow H[1]$ ▷ 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

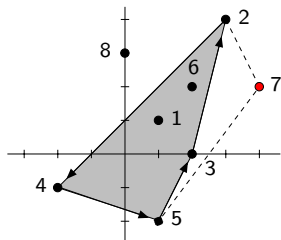


X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8
H	3	2	4	5				
	1	2	3	4				

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

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

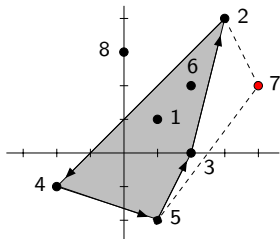
Insero Ponto



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

Que características estes pontos têm?

Insero Ponto

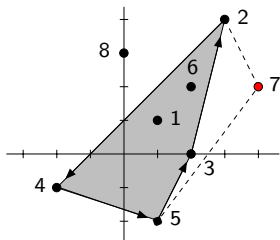


O InseroPonto tem que encontrar os pontos 2 e 5 acima.

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.

Inserir Ponto



O InserirPonto 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.

Inserer Ponto

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

```
1   $H[0] \leftarrow H[h]$     $H[h+1] \leftarrow H[1]$    ▷ 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      $i \leftarrow i + 1$ 
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      $j \leftarrow j + 1$ 
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      $F[t] \leftarrow H[i]$     $t \leftarrow t + 1$     $i \leftarrow (i \bmod h) + 1$ 
12      $F[t] \leftarrow H[j]$     $t \leftarrow t + 1$     $F[t] \leftarrow k$ 
13 devolva ( $F, t$ )
```


Casos degenerados

Como tratar os casos degenerados nos três algoritmos acima?