Geometria Computacional

Cristina G. Fernandes

Departamento de Ciência da Computação do IME-USP

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

segundo semestre de 2018

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 $X[1..n], Y[1..n]$.

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 $\alpha_1 + \cdots + \alpha_n = 1$.

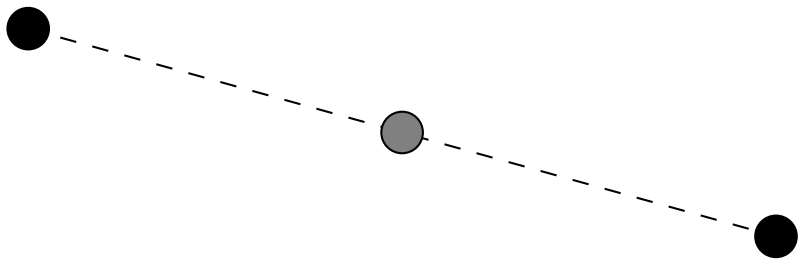
Combinação convexa

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 $\alpha_1 + \cdots + \alpha_n = 1$.



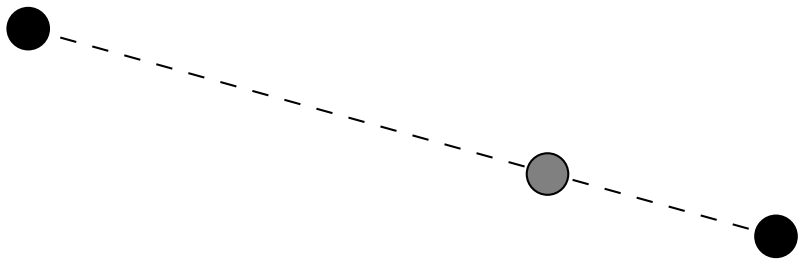
Combinação convexa

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 $\alpha_1 + \cdots + \alpha_n = 1$.



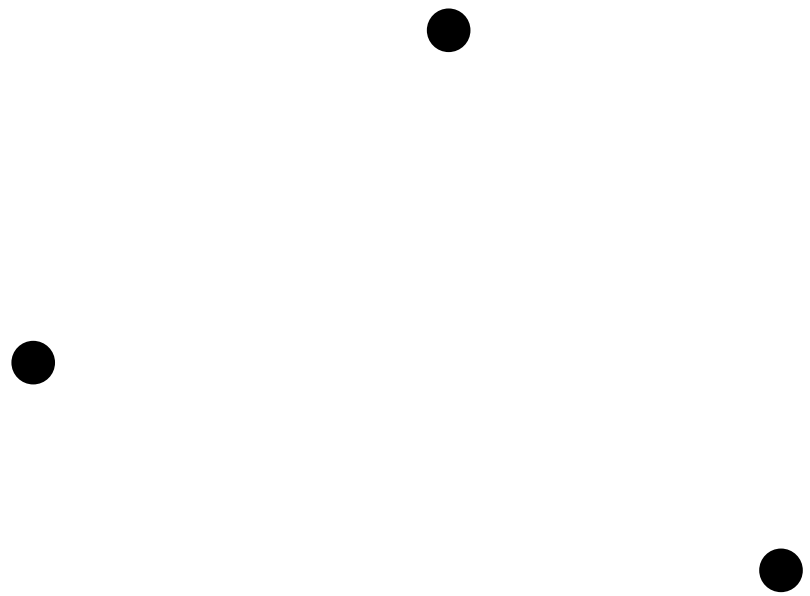
Combinação convexa

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 $\alpha_1 + \cdots + \alpha_n = 1$.



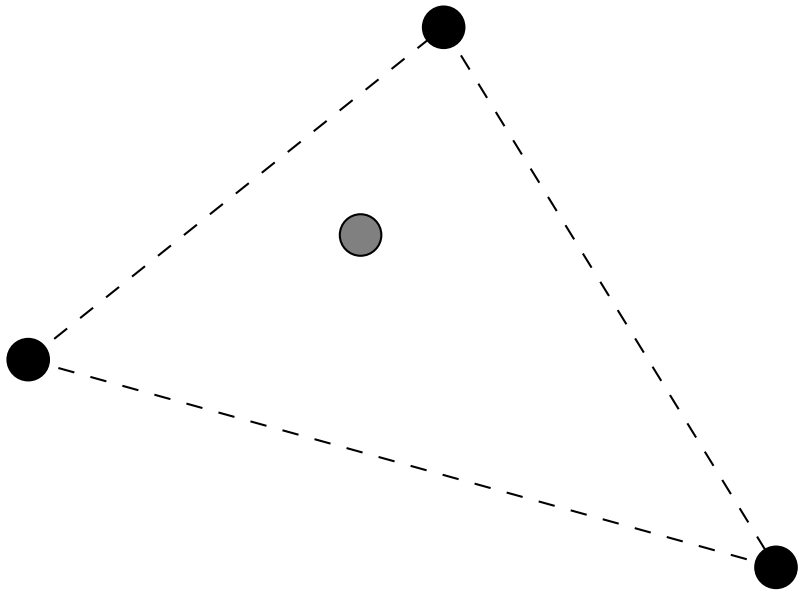
Combinação convexa

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

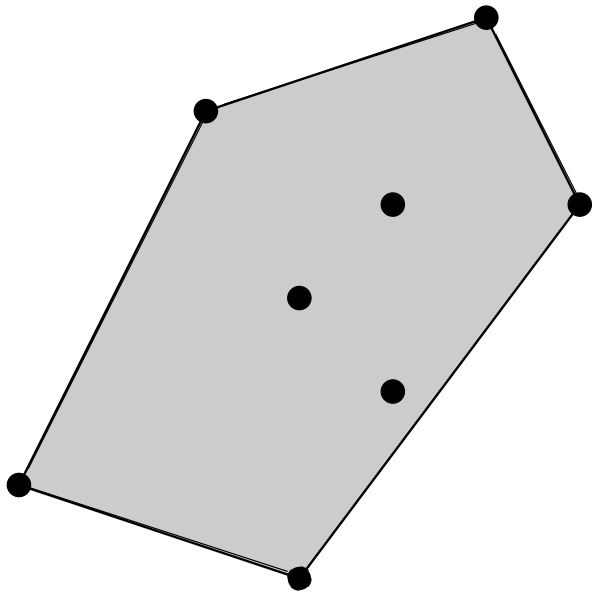
Fecho convexo de P: conjunto de combinações convexas de pontos de P, ou seja,

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

Fecho convexo

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

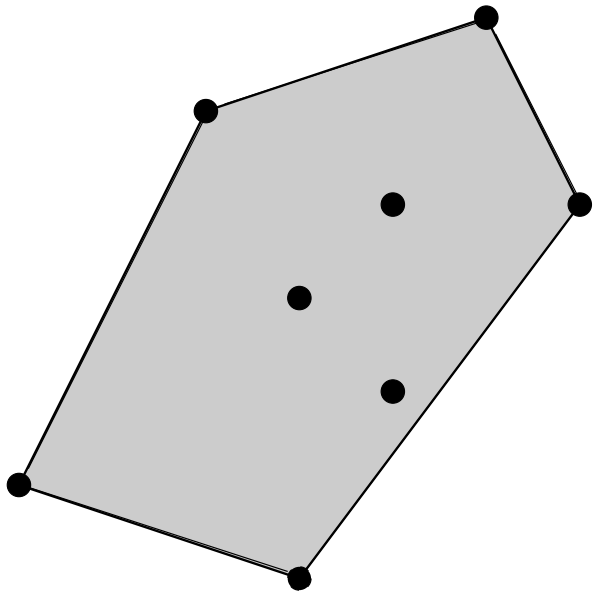
$$\text{conv}(P) := \{\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]) : \alpha_1 + \cdots + \alpha_n = 1, \text{ e } \alpha_i \geq 0 \ (i = 1, \ldots, n)\}.$$



Fecho convexo

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

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



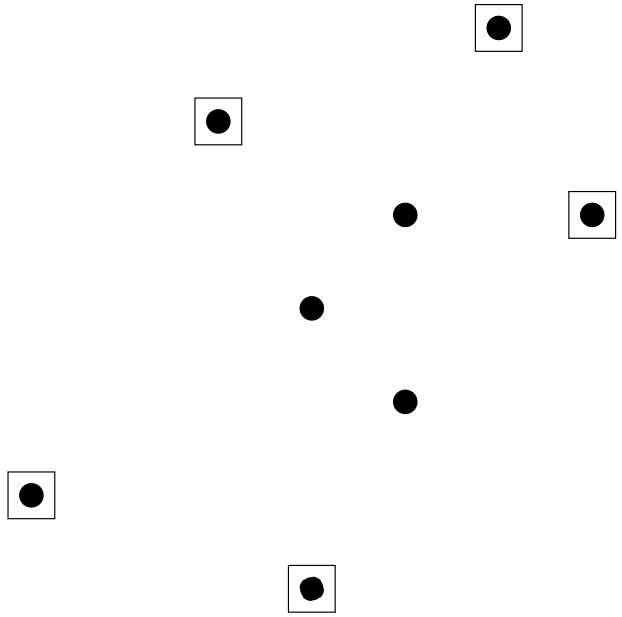
Problema: Dada uma coleção P de pontos do plano, determinar o fecho convexo de P.

Pontos extremos

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

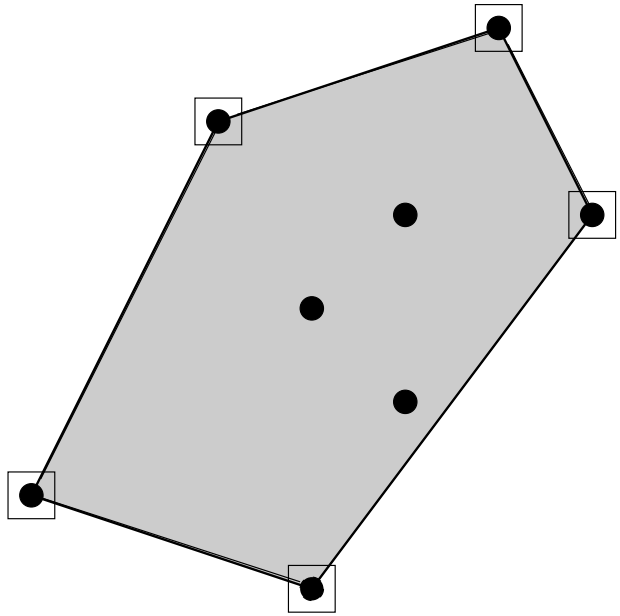
Pontos extremos

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



Pontos extremos

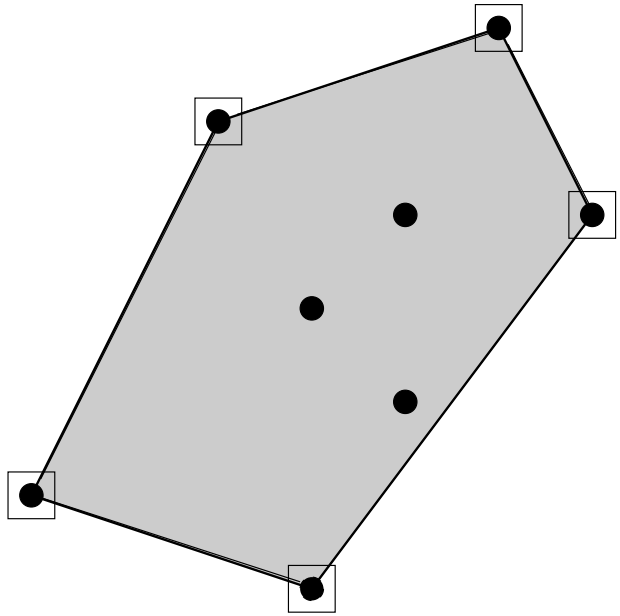
Ponto (x, y) de P é extremo se não é combinação convexa de pontos de $P \setminus \{(x, y)\}$.



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

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.

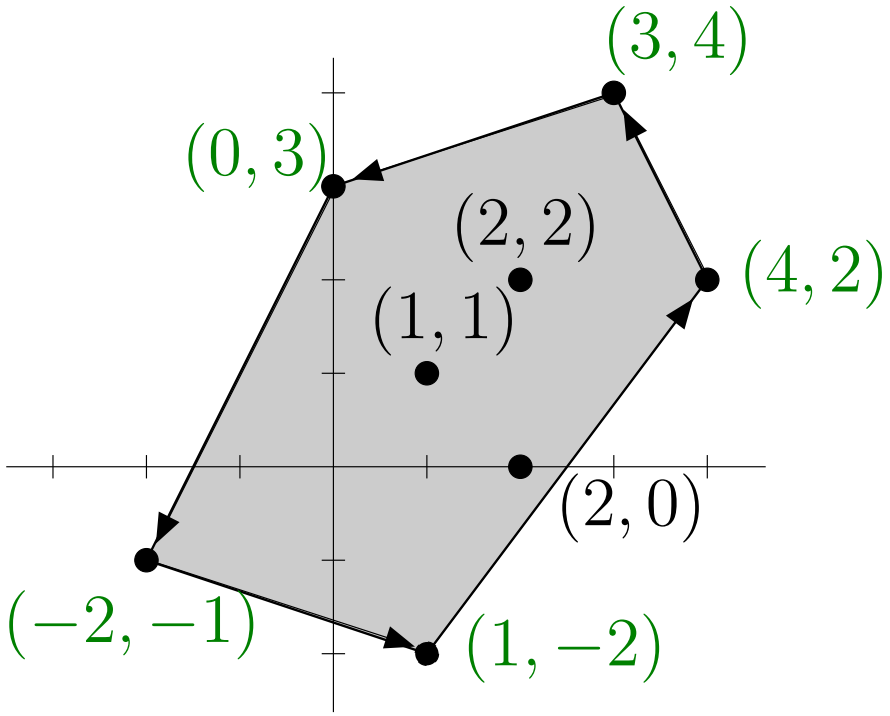
Hipótese simplificadora: a coleção não contém três pontos colineares.

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).

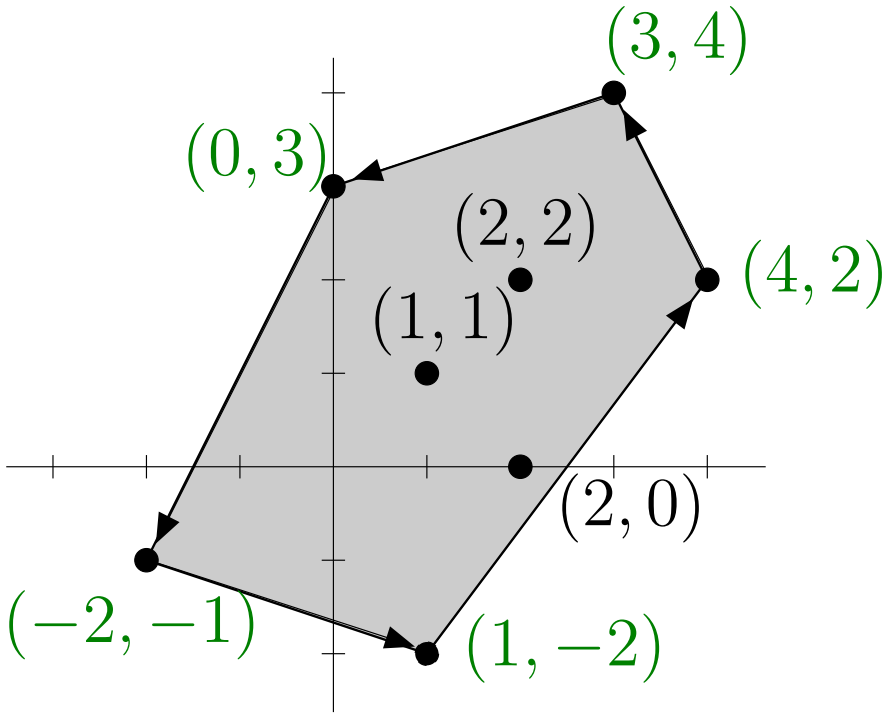


X	1	3	2	−2	1	2	4	0
Y	1	4	0	−1	−2	2	2	3
	1	2	3	4	5	6	7	8

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).



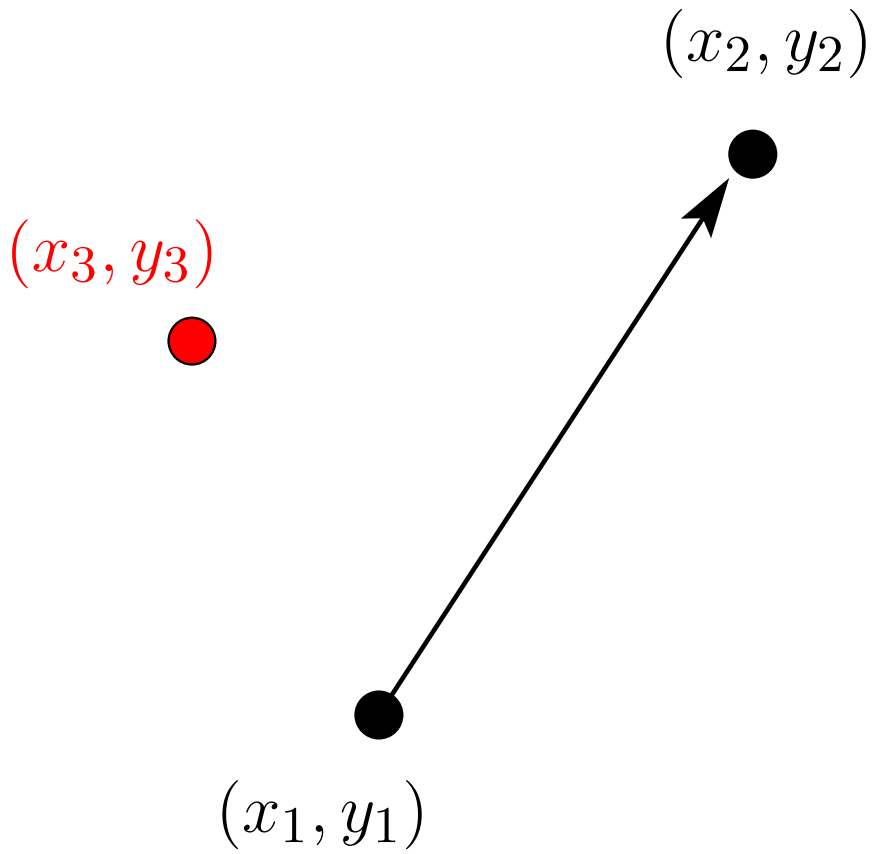
X	1	3	2	−2	1	2	4	0
Y	1	4	0	−1	−2	2	2	3
	1	2	3	4	5	6	7	8

H	2	8	4	5	7
	1	2	3	4	5

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

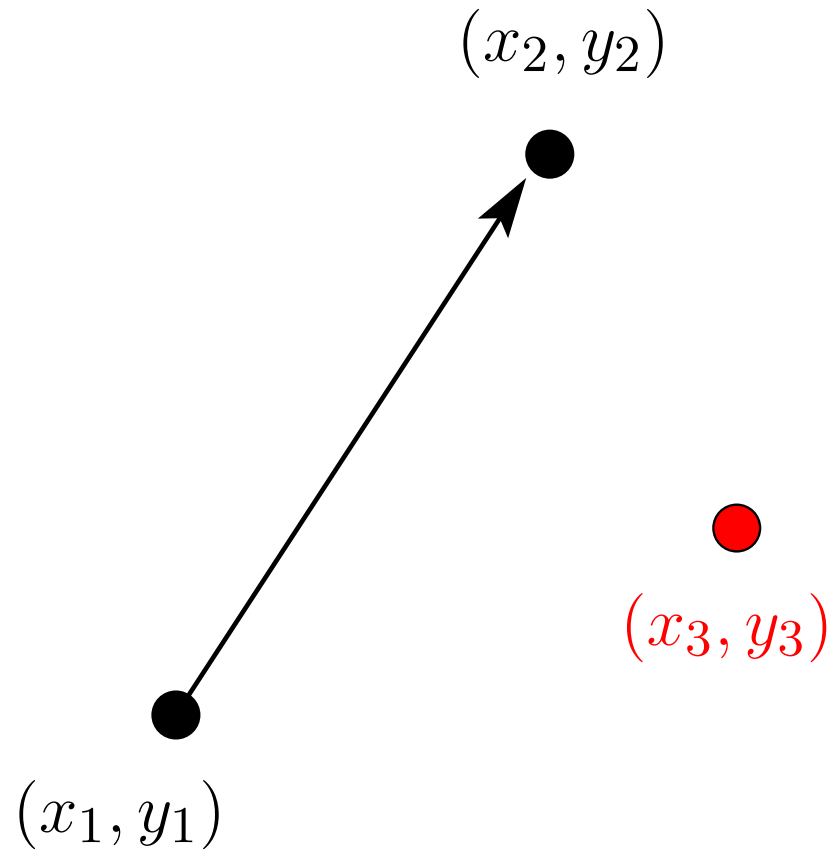
Predicados geométricos

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.

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

Abreviaturas

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

ESQ$(X, Y, i, j, k) =$
ESQUERDA$((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

Em pseudocódigo:

ESQ(X, Y, i, j, k)
1 devolva ESQUERDA$((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

Abreviaturas

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

ESQ$(X, Y, i, j, k) =$
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
DIR$(X, Y, i, j, k) =$
DIREITA$((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

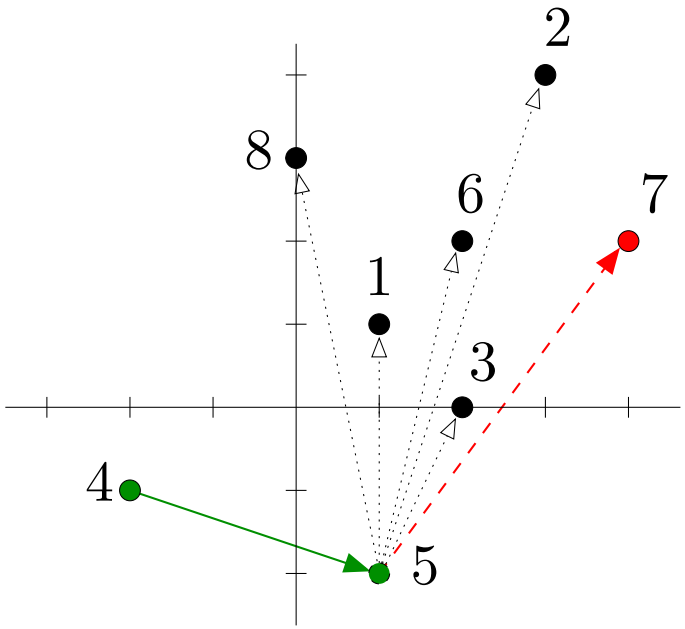
DIR(X, Y, i, j, k)
1 devolva DIREITA$((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

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.



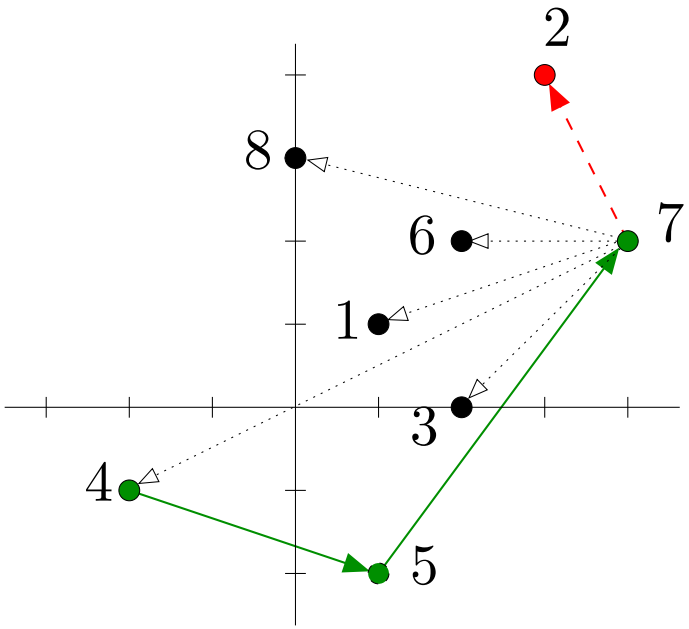
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	4	5	?	...
	1	2	3	4

DIR$(X, Y, 5, 7, j)$ = FALSO para $j = 1, \ldots, 8$

Embrulho de presente

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



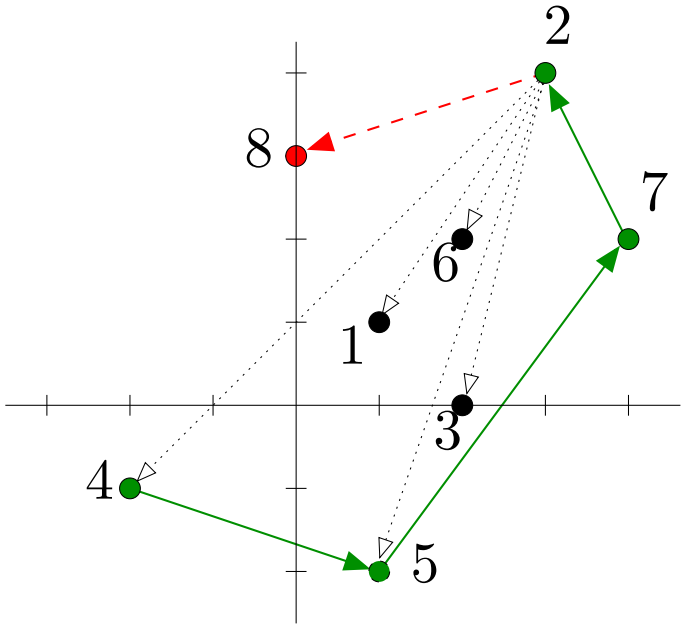
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

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

DIR$(X, Y, 5, 7, j)$ = FALSO para $j = 1, \ldots, 8$
DIR$(X, Y, 7, 2, j)$ = FALSO para $j = 1, \ldots, 8$

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

DIR$(X, Y, 5, 7, j)$ = FALSO para $j = 1, \ldots, 8$
DIR$(X, Y, 7, 2, j)$ = FALSO para $j = 1, \ldots, 8$
DIR$(X, Y, 2, 8, j)$ = FALSO para $j = 1, \ldots, 8$

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 ← 0
 2  H[0] ← min{i ∈ [1..n] : X[i] ≤ X[j], 1 ≤ j ≤ n}      ▷ senti
 3  repita
 4      i ← (H[h] mod n) + 1      ▷ qualquer ponto distinto de H[
 5      para j ← 1 até n faça
 6          se DIR(X, Y, H[h], i, j) então i ← j
 7      h ← h + 1
 8      H[h] ← i
 9  até que i = H[0]      ▷ fechou o polígono
10  devolva (H, h)
```

Embrulho de presente

```
EMBRULHO(X, Y, n)
 1  h ← 0
 2  H[0] ← min{i ∈ [1..n] : X[i] ≤ X[j], 1 ≤ j ≤ n}
 3  repita
 4      i ← (H[h] mod h) + 1      ▷ qualquer ponto distinto de H[
 5      para j ← 1 até n faça
 6          se DIR(X, Y, H[h], i, j) então i ← j
 7      h ← h + 1
 8      H[h] ← i
 9  até que i = H[0]      ▷ fechou o polígono
10  devolva (H, h)
```

Embrulho de presente

```
EMBRULHO(X, Y, n)
 1  h ← 0
 2  H[0] ← min{i ∈ [1..n] : X[i] ≤ X[j], 1 ≤ j ≤ n}
 3  repita
 4      i ← (H[h] mod h) + 1      ▷ qualquer ponto distinto de H[
 5      para j ← 1 até n faça
 6          se DIR(X, Y, H[h], i, j) então i ← j
 7      h ← h + 1
 8      H[h] ← i
 9  até que i = H[0]      ▷ fechou o polígono
10  devolva (H, h)
```

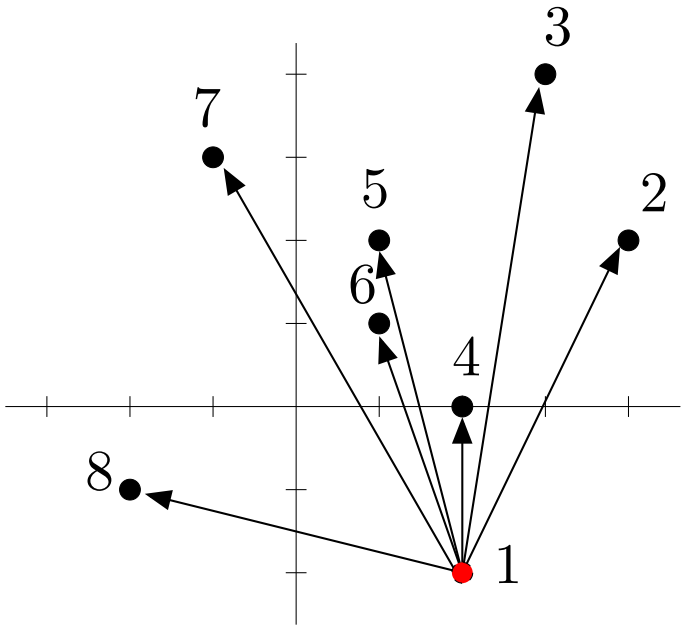
Consumo de tempo: $\Theta(nh)$,
onde h é o número de pontos no fecho convexo.

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

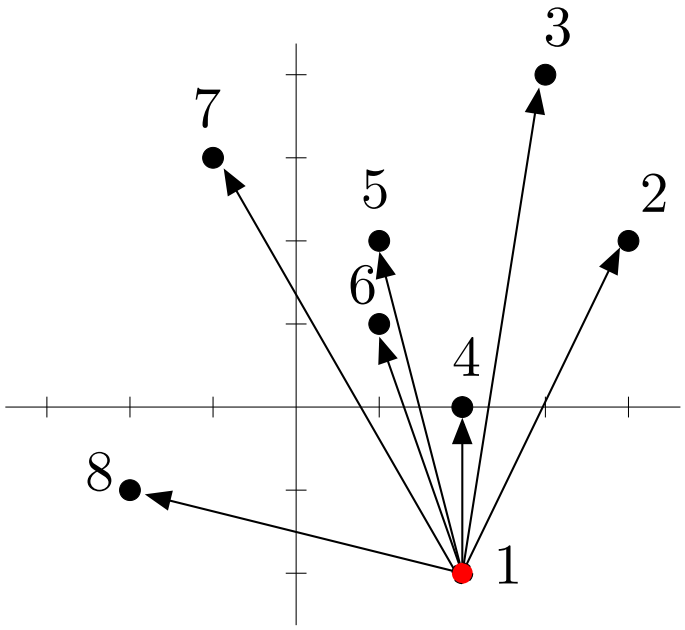
Pré-processamento do Graham

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

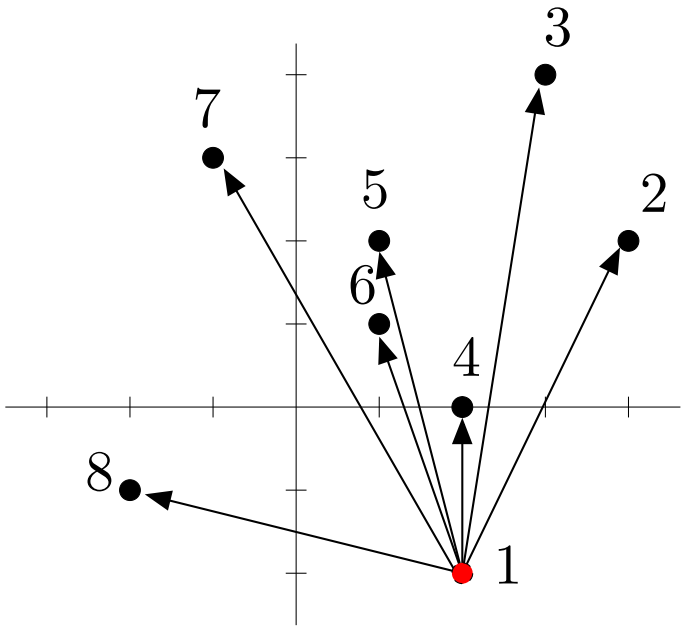
Pré-processamento do Graham

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

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

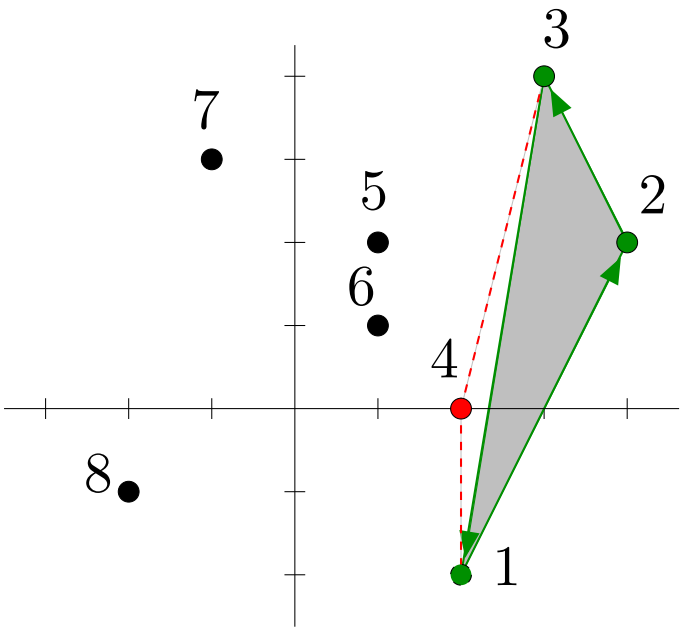
Algoritmo de Graham

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.

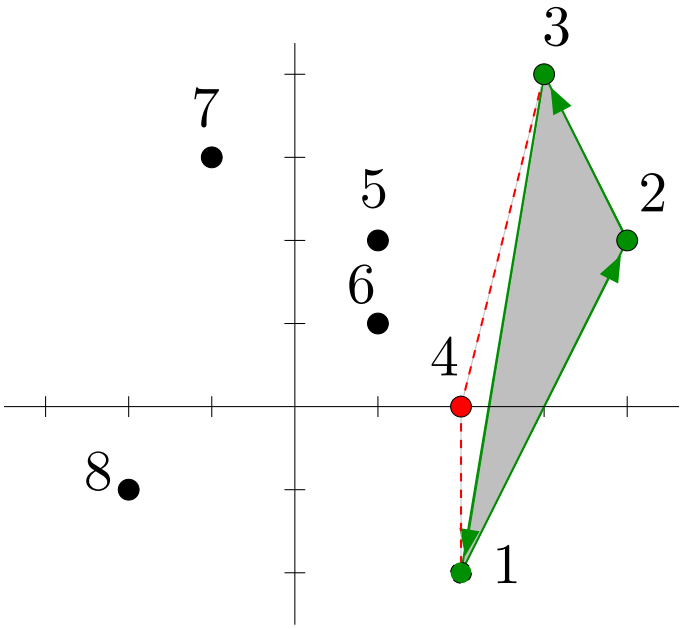


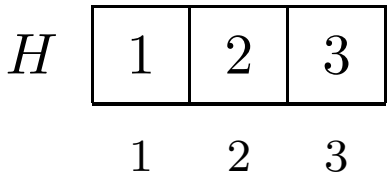
H	1	2	3
	1	2	3

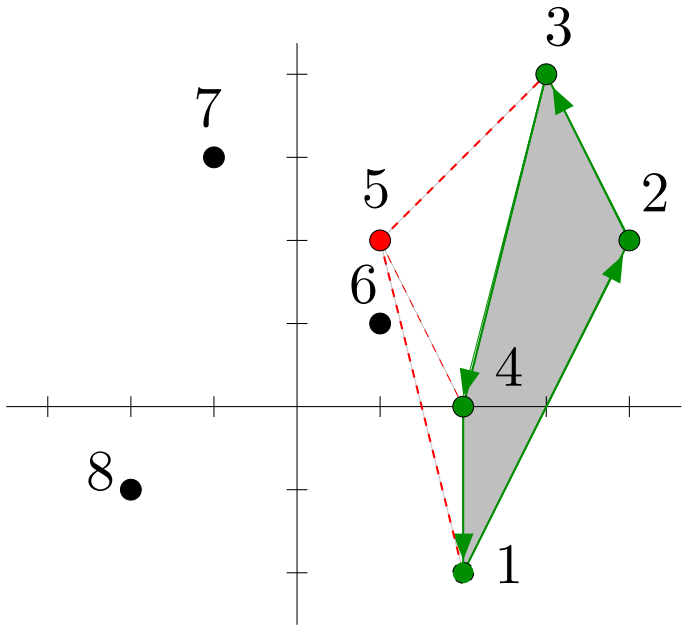
Algoritmo de Graham

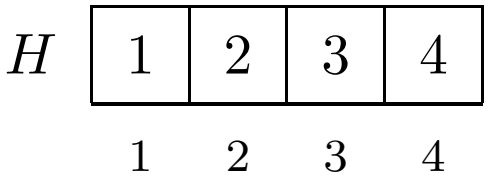
Após pré-processamento: examinar um ponto após o outro, mantendo o fecho convexo dos pontos já examinados.

Começamos com os três primeiros pontos.



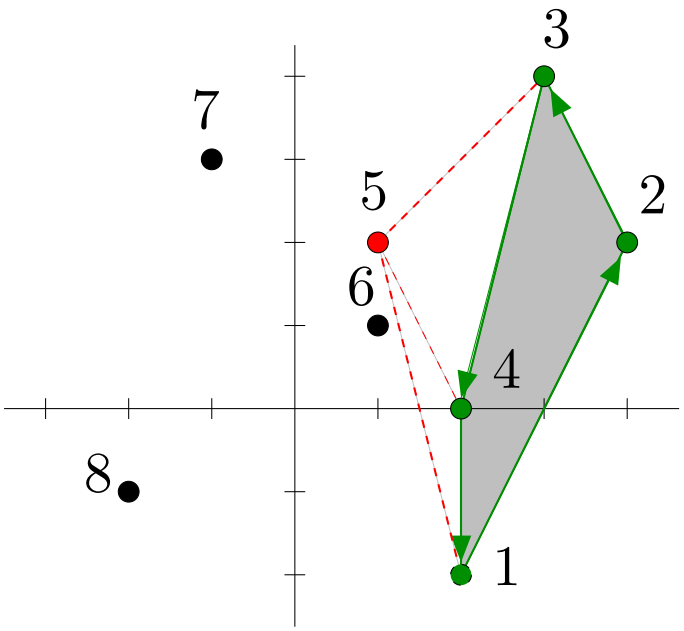




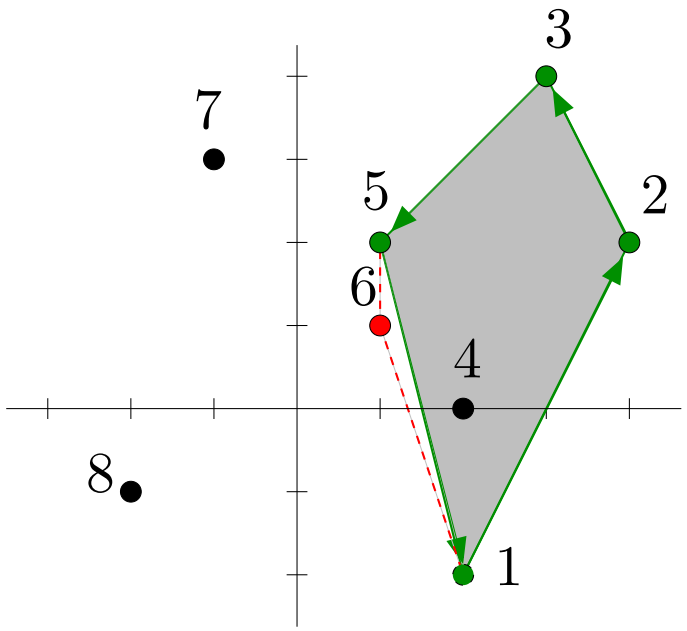


Algoritmo de Graham

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



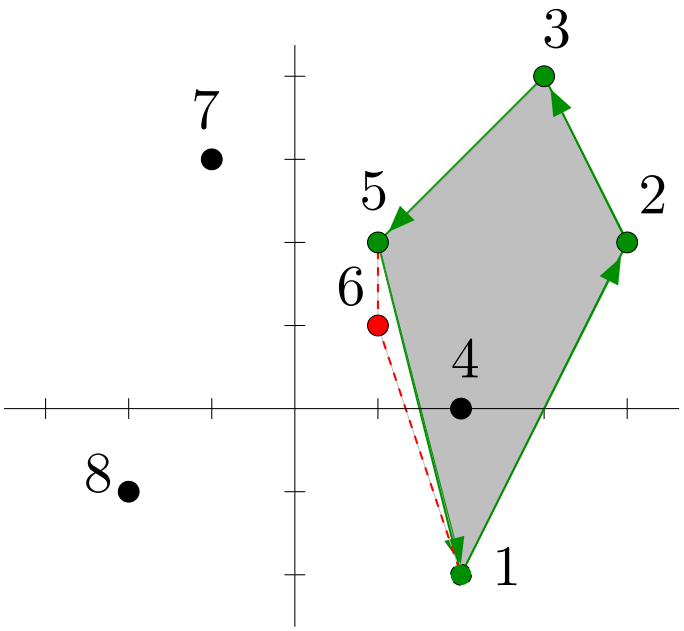
H	1	2	3	4
	1	2	3	4



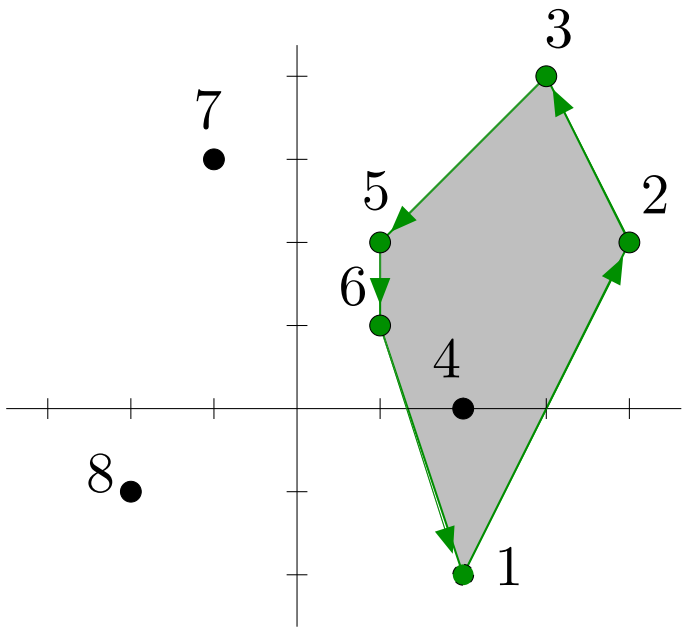
H	1	2	3	5
	1	2	3	4

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

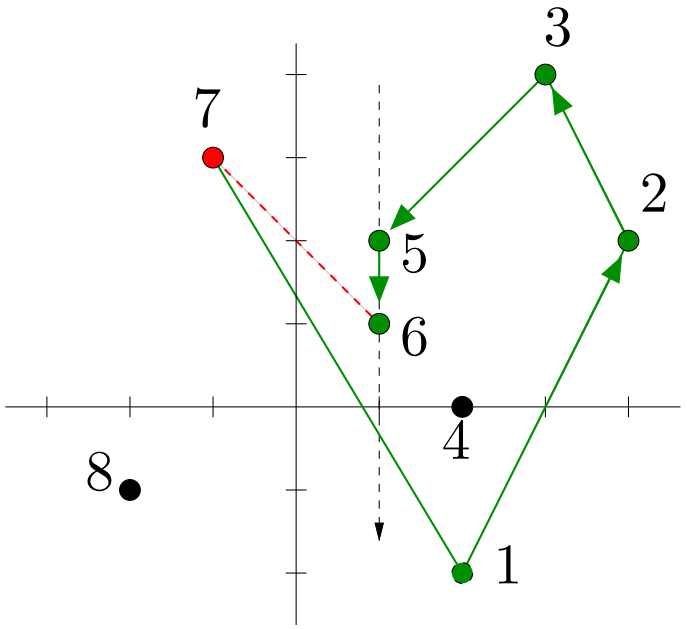
Consumo de tempo:
Pré-processamento: $\Theta(n \lg n)$
Restante: $\Theta(n)$.

Algoritmo de Graham

```
3  para k ← 4 até n faça
4      j ← h
5      enquanto DIR(X, Y, H[j−1], H[j], k) faça
6          j ← j − 1
7      h ← j + 1     H[h] ← k
```

Algoritmo de Graham

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

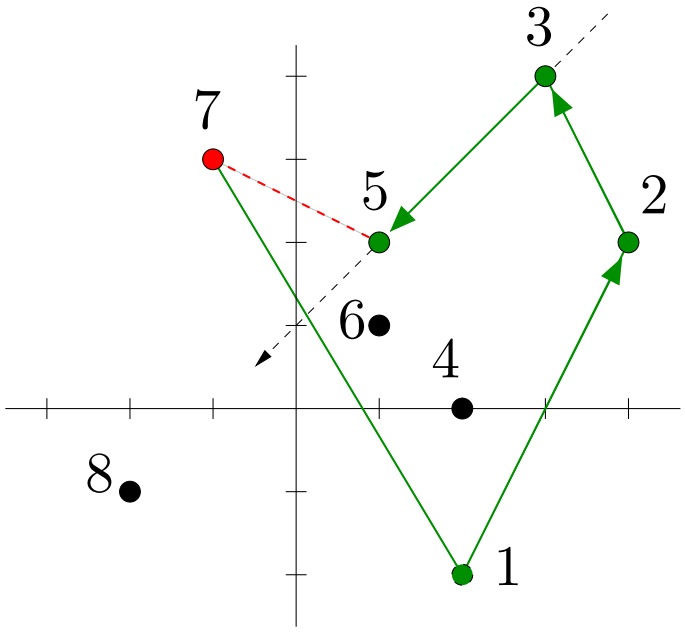


DIR$(X, Y, 5, 6, 7) =$ VERDADE

H	1	2	3	5	~~6~~
	1	2	3	4	5

Algoritmo de Graham

```
3   para k ← 4 até n faça
4       j ← h
5       enquanto DIR(X, Y, H[j−1], H[j], k) faça
6           j ← j − 1
7       h ← j + 1     H[h] ← k
```

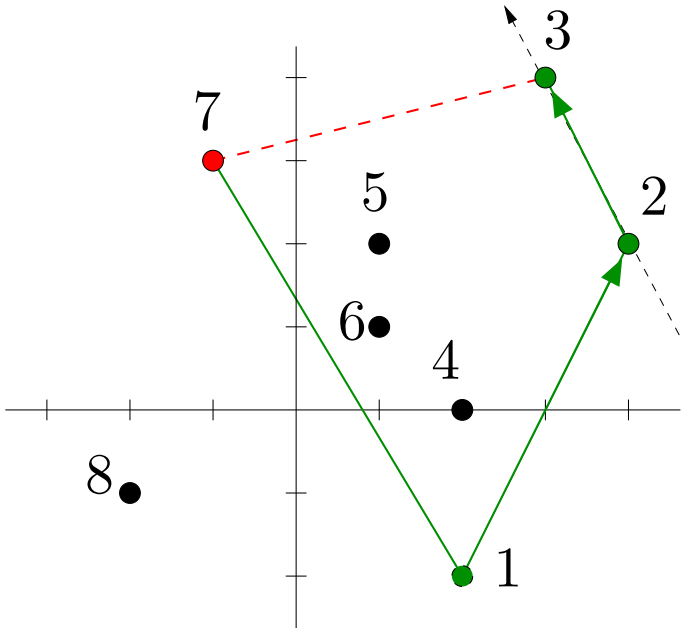


DIR$(X, Y, 3, 5, 7)$ = VERDADE

H	1	2	3	~~5~~
	1	2	3	4

Algoritmo de Graham

```
3  para k ← 4 até n faça
4      j ← h
5      enquanto DIR(X, Y, H[j−1], H[j], k) faça
6          j ← j − 1
7      h ← j + 1    H[h] ← k
```

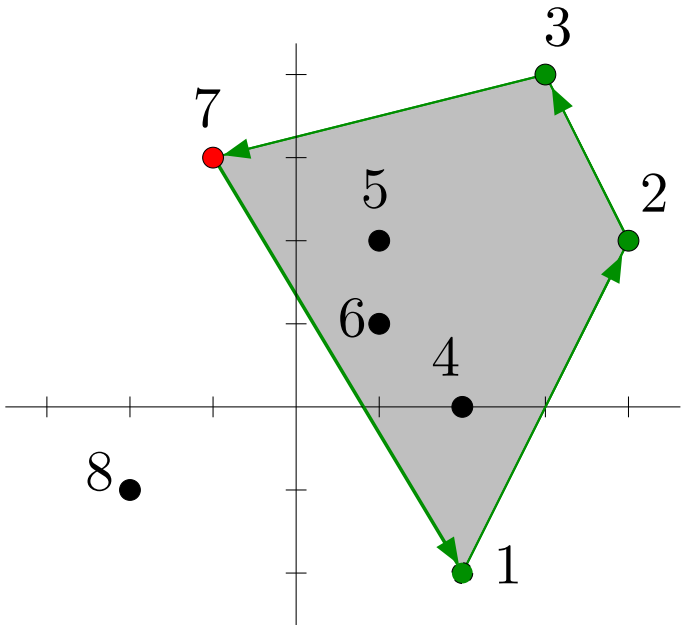


DIR$(X, Y, 2, 3, 7)$ = FALSO

H	1	2	3
	1	2	3

Algoritmo de Graham

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

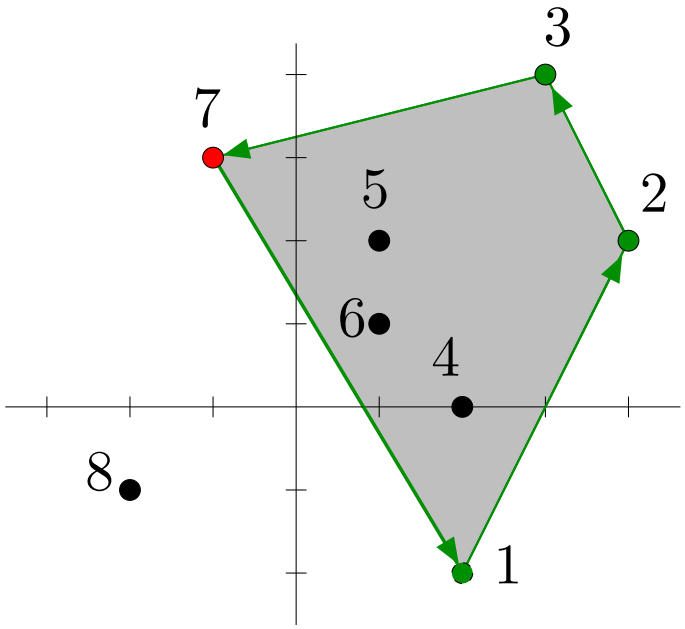


ESQ$(X, Y, 3, 2, 7) =$ FALSO

H	1	2	3	7
	1	2	3	4

Algoritmo de Graham

```
3  para k ← 4 até n faça
4      j ← h
5      enquanto DIR(X, Y, H[j−1], H[j], k) faça
6          j ← j − 1
7      h ← j + 1    H[h] ← k
```



ESQ$(X, Y, 3, 2, 7)$ = FALSO

H	1	2	3	7
	1	2	3	4

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

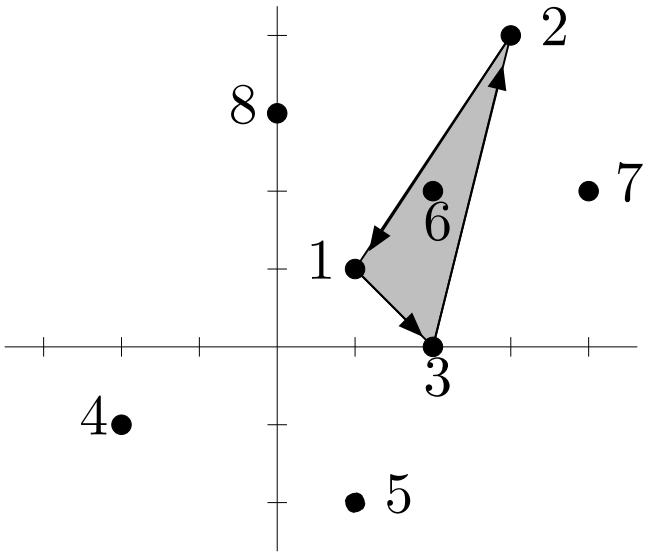
Um algoritmo incremental

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.



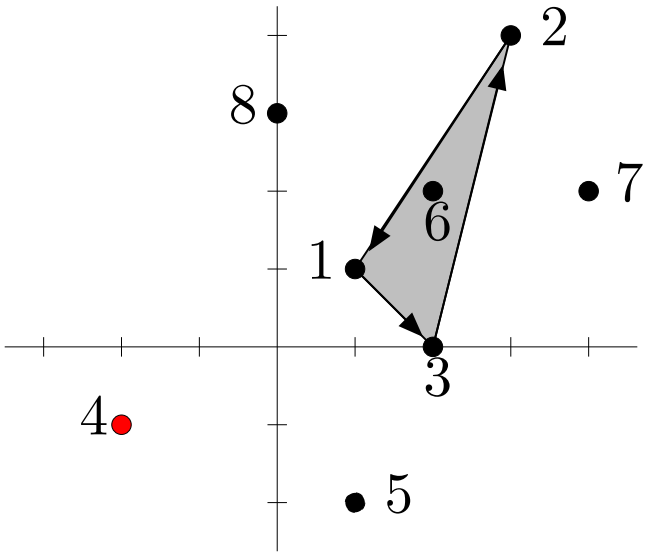
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

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

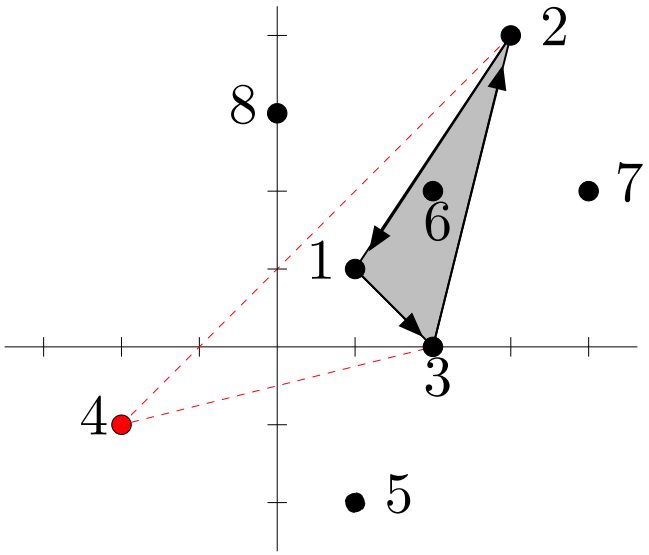
H	1	3	2
	1	2	3

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

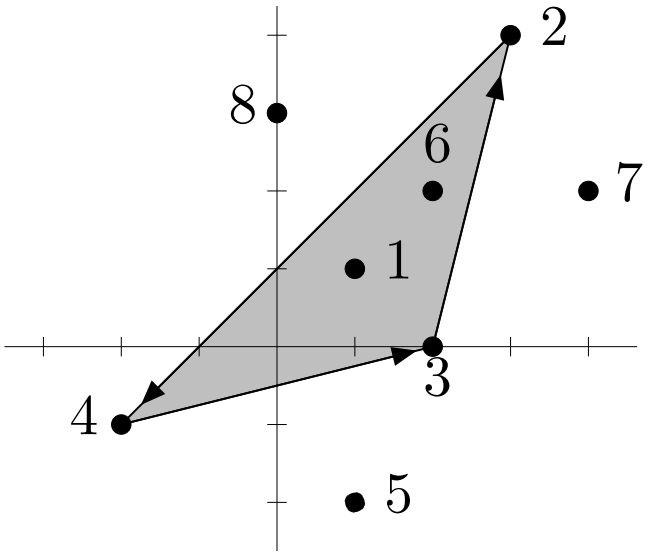
H	1	3	2
	1	2	3

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.



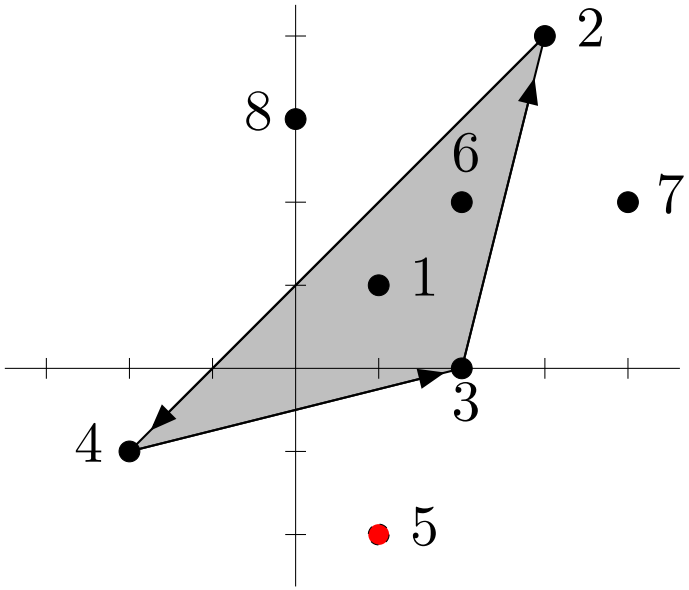
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	3	2	4
	1	2	3

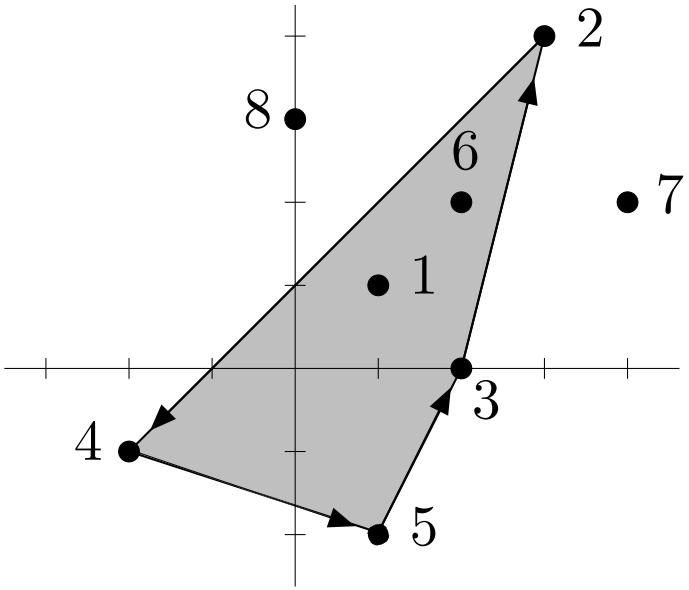
Atualizamos o fecho para incluir o ponto $(-2, -1)$.

Um algoritmo incremental

Próximas iterações...



H	3	2	4
	1	2	3

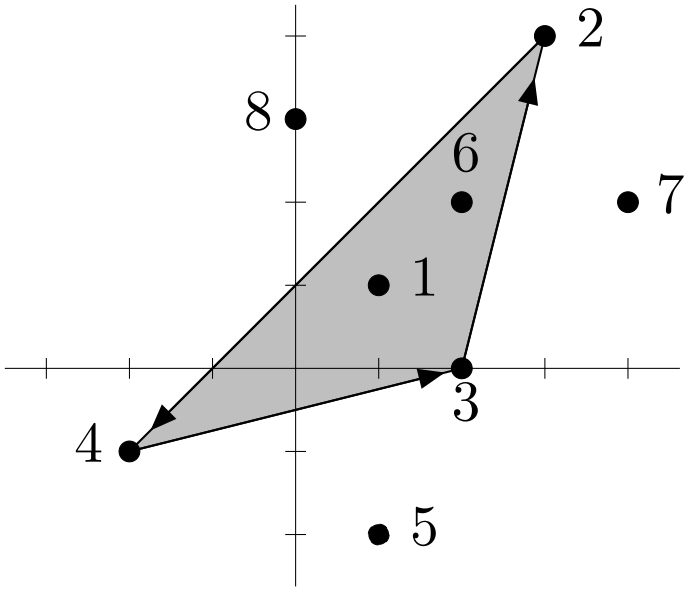


H	3	2	4	5
	1	2	3	4

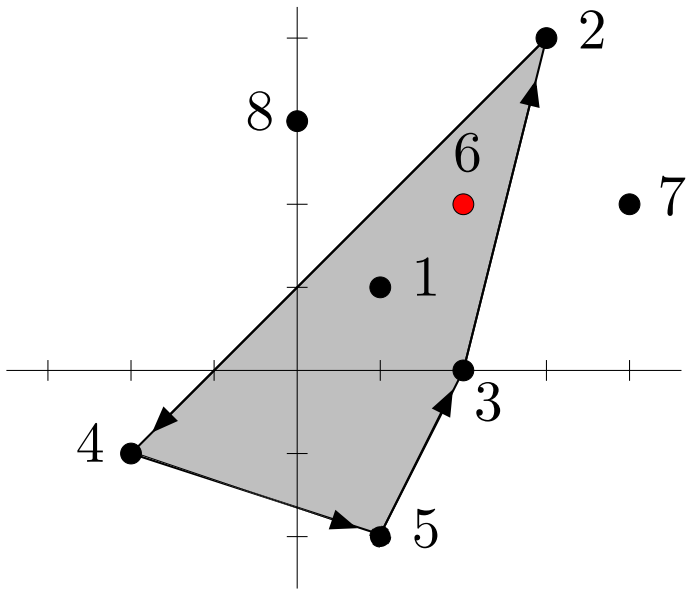
O quinto ponto, $(1, -2)$, pertence ao fecho corrente? Não.

Um algoritmo incremental

Próximas iterações...



H	3	2	4
	1	2	3

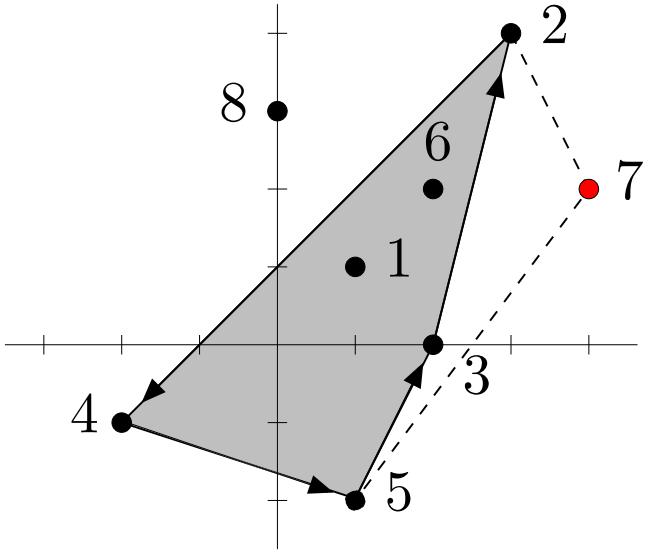


H	3	2	4	5
	1	2	3	4

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

Um algoritmo incremental

Próximas iterações...



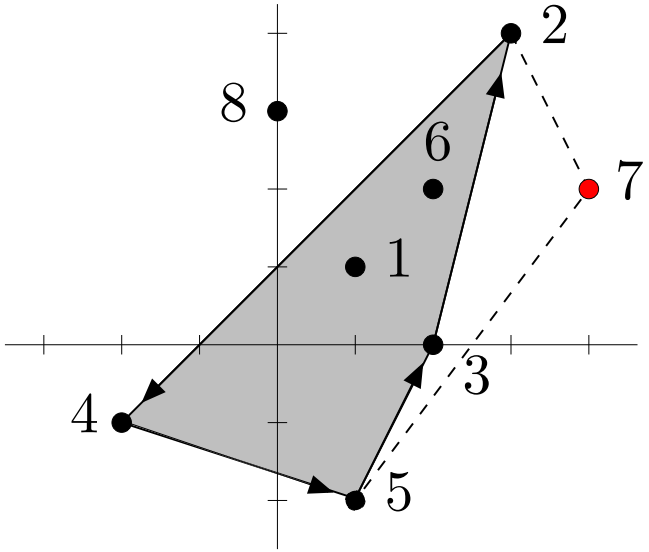
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

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 $(H, h) \leftarrow$ INSEREPONTO(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] ← 1   H[2] ← 2   H[3] ← 3   h ← 3
3      senão  H[1] ← 1   H[2] ← 3   H[3] ← 2   h ← 3
4  para k ← 4 até n faça
5      se não PERTENCE(H, h, X, Y, X[k], Y[k])
6          então (H, h) ← INSEREPONTO(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, $\mathrm{O}(h)$.

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 $(H, h) \leftarrow$ INSEREPONTO(H, h, X, Y, k)
7 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 $X[1..k], Y[1..k]$.

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)

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, $\Theta(h)$.

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

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

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

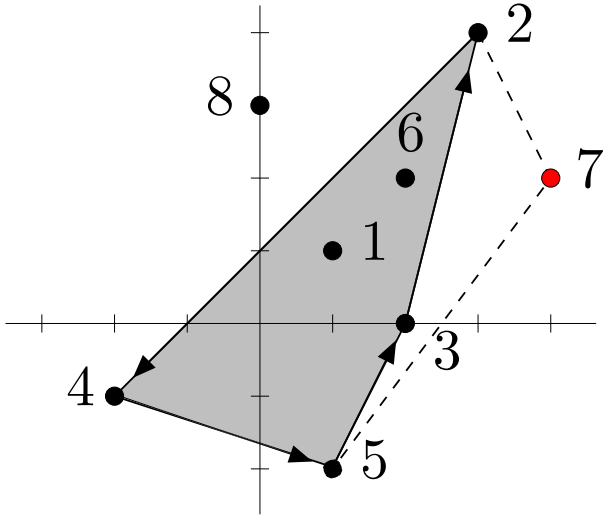
Pertence

```
PERTENCE(H, h, X, Y, x, y)
1  H[h+1] ← H[1]      ▷ sentinela
2  para i ← 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



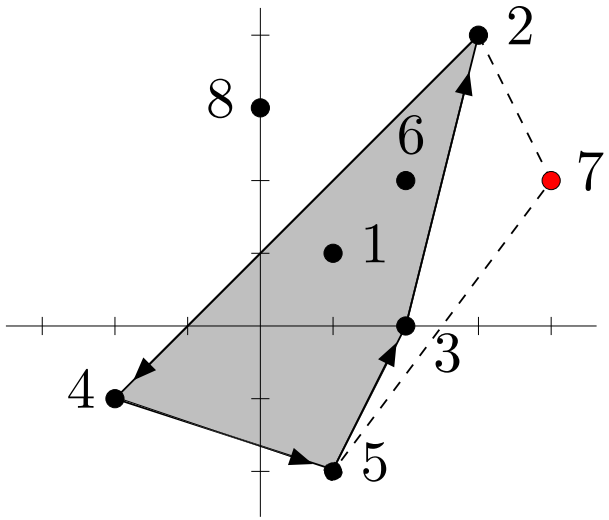
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

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

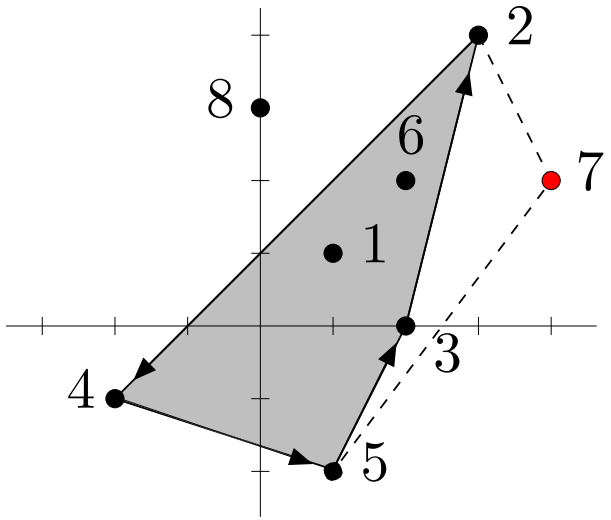
Insere Ponto



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

Que características estes pontos têm?

Insere Ponto

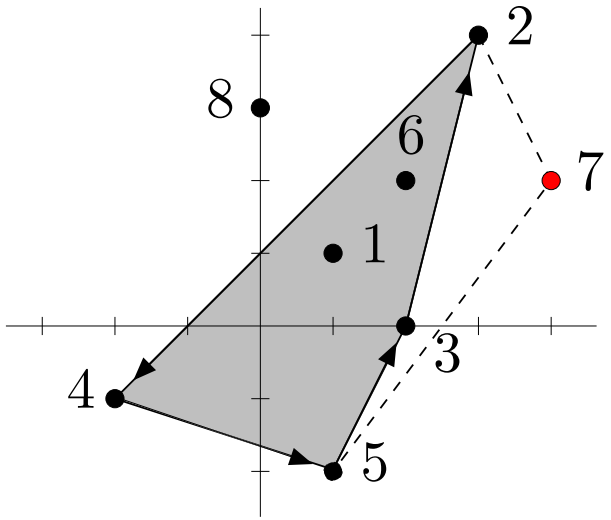


O INSEREPONTO tem que encontrar os pontos 2 e 5 acima.

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, k)

1 $H[0] \leftarrow H[h] \quad H[h{+}1] \leftarrow H[1]$ ▷ sentinelas
2 $i \leftarrow 1$
3 enquanto ESQ$(X, Y, H[i{-}1], H[i], k) =$ ESQ$(X, Y, H[i], H[i{+}1], k)$
4 $\quad i \leftarrow i + 1$
5 $j \leftarrow i + 1$
6 enquanto ESQ$(X, Y, H[j{-}1], H[j], k) =$ ESQ$(X, Y, H[j], H[j{+}1], k$
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)