Convite à Geometria Computacional

Jornadas de Atualização em Informática

Cristina G. Fernandes e **José Coelho de Pina**

Departamento de Ciência da Computação do IME-USP

`http://www.ime.usp.br/dcc/`

Bento Gonçalves, julho de 2009

Visão geral do minicurso

Aula 1:

- Introdução (Secs. 7.1 e 7.2)
- Par de pontos mais próximos (Sec. 7.3)

Aula 2:

- Fecho convexo (Sec. 7.4)

Visão geral do minicurso

Aula 1:

- Introdução (Secs. 7.1 e 7.2)
- Par de pontos mais próximos (Sec. 7.3)

Aula 2:

- Fecho convexo (Sec. 7.4)

Aula 3:

- Método da linha de varredura (Secs. 7.5 e 7.6)

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 $\alpha_i \geq 0$, 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 $\alpha_1 + \cdots + \alpha_n = 1$.

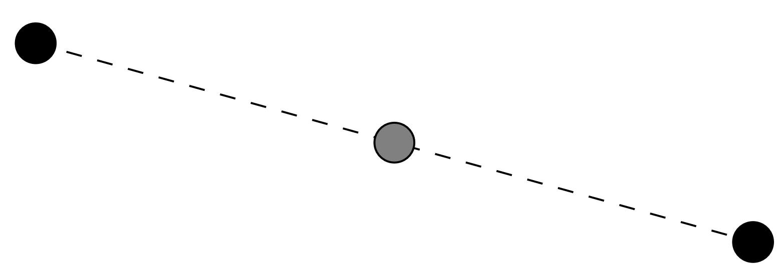
Combinação convexa

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



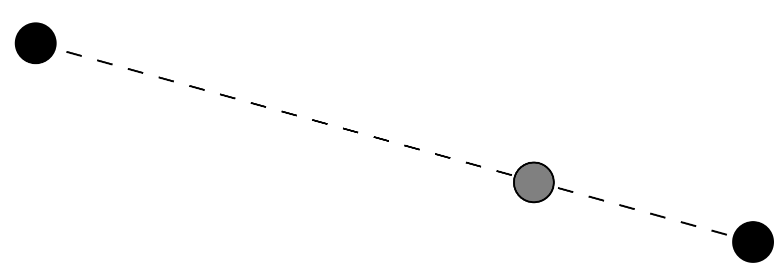
Combinação convexa

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



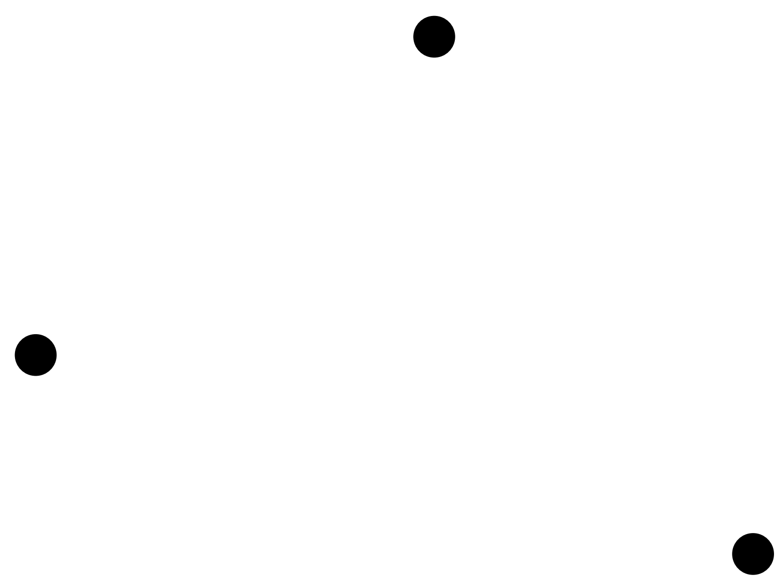
Combinação convexa

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



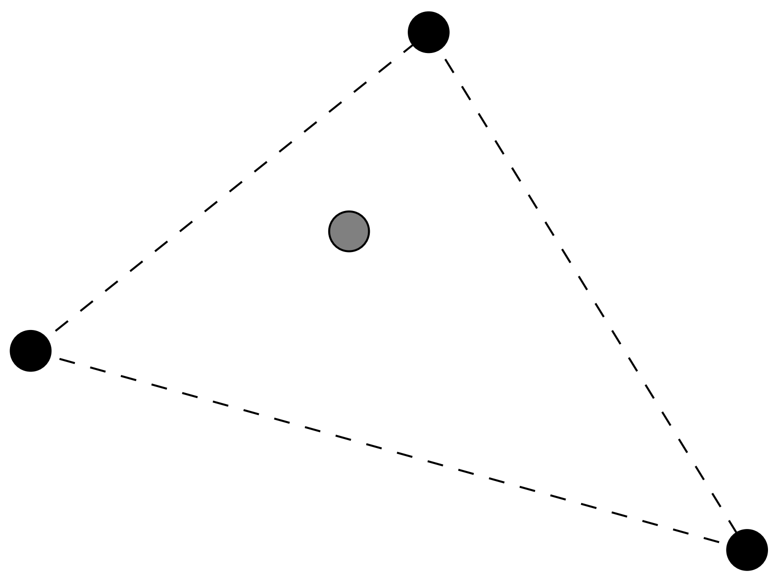
Combinação convexa

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



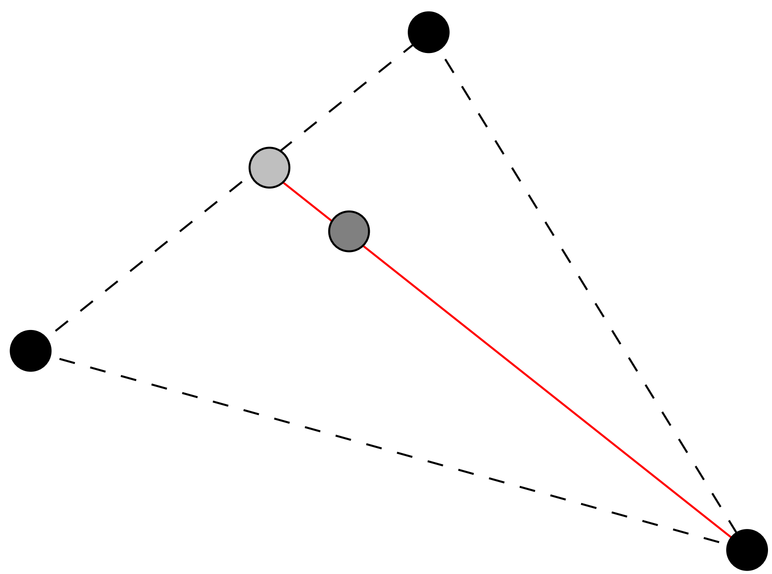
Combinação convexa

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

Combinação convexa de pontos de P: soma da forma

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

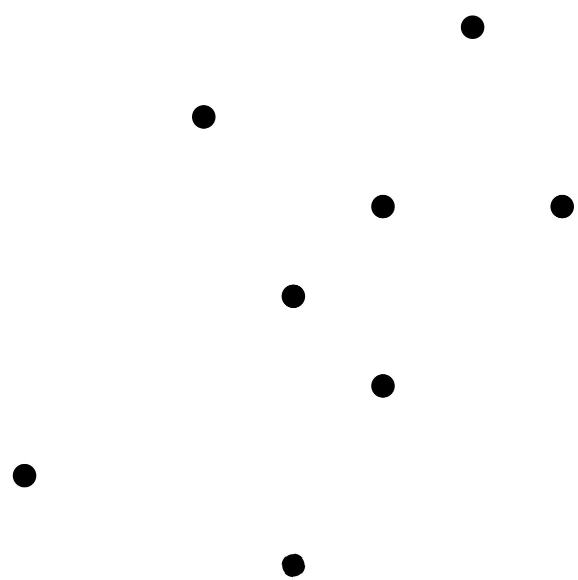
com $\alpha_i \geq 0$, para $i = 1, \ldots, n$, e $\alpha_1 + \cdots + \alpha_n = 1$.



Fecho convexo

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

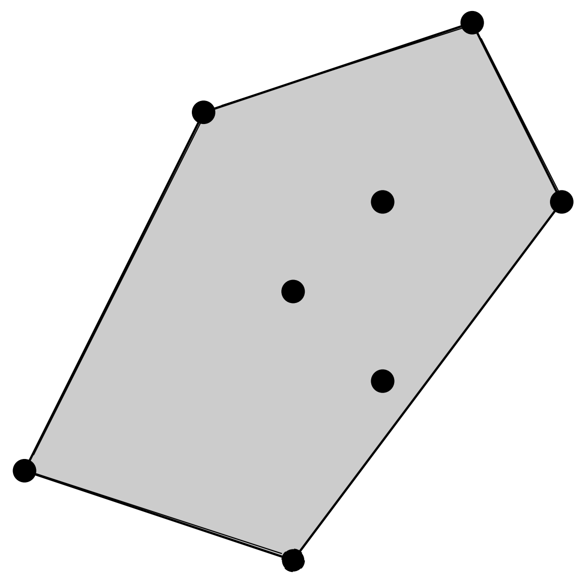
$$\mathrm{conv}(P) := \{\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]) : \alpha_1 + \cdots + \alpha_n = 1, \text{ e } \alpha_i \geq 0 \ (i = 1, \ldots, n)\}.$$



Fecho convexo

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

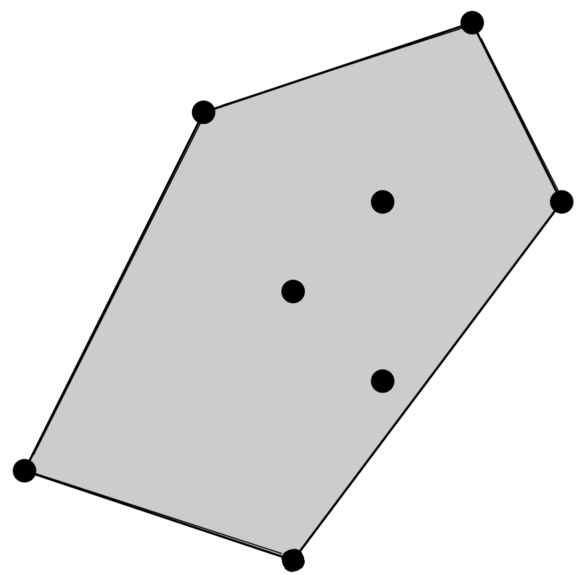
$$\mathrm{conv}(P) := \{\alpha_1(X[1], Y[1]) + \cdots + \alpha_n(X[n], Y[n]) : \alpha_1 + \cdots + \alpha_n = 1, \text{ e } \alpha_i \geq 0 \ (i = 1, \ldots, n)\}.$$



Fecho convexo

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

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



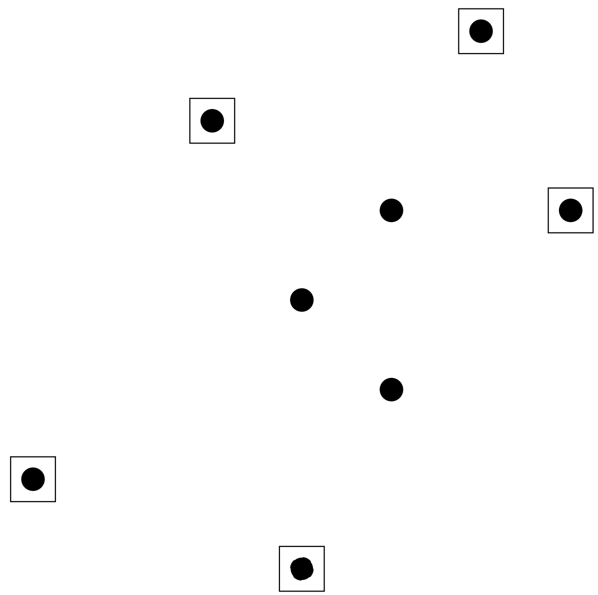
Problema: Dada uma coleção P de pontos do plano, determinar o fecho convexo de P.

Pontos extremos

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

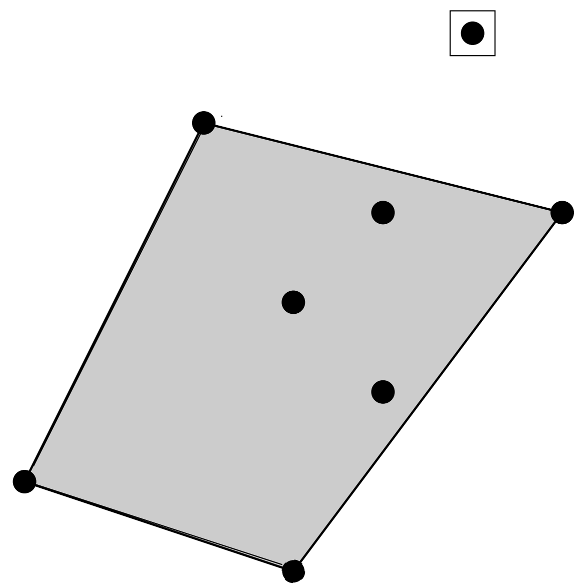
Pontos extremos

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



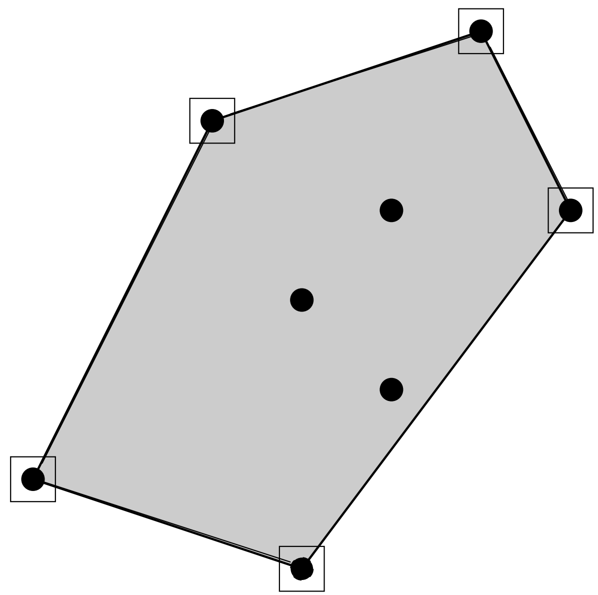
Pontos extremos

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



Pontos extremos

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



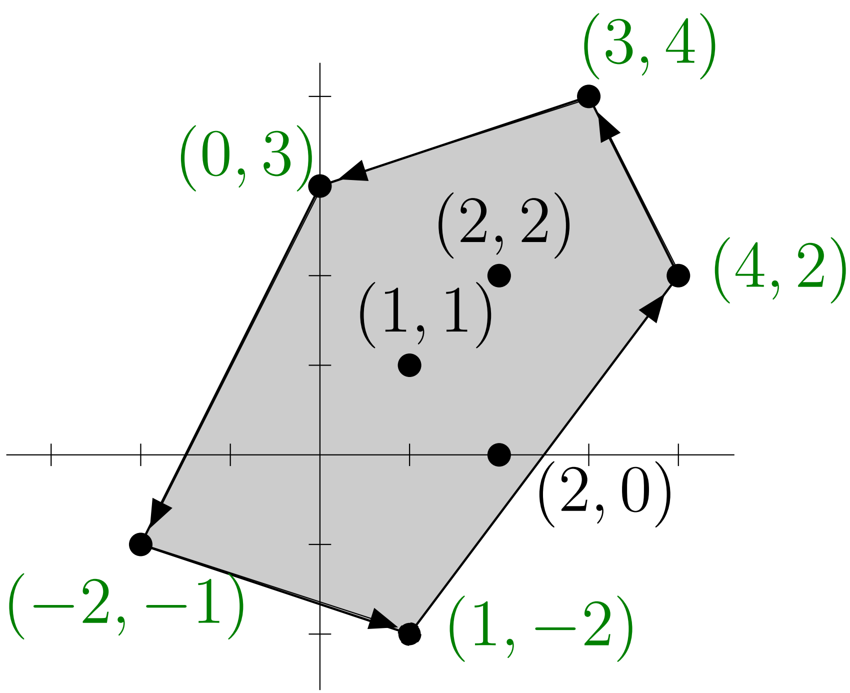
Pontos extremos de $\mathrm{conv}(P)$ são pontos extremos de P.

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).

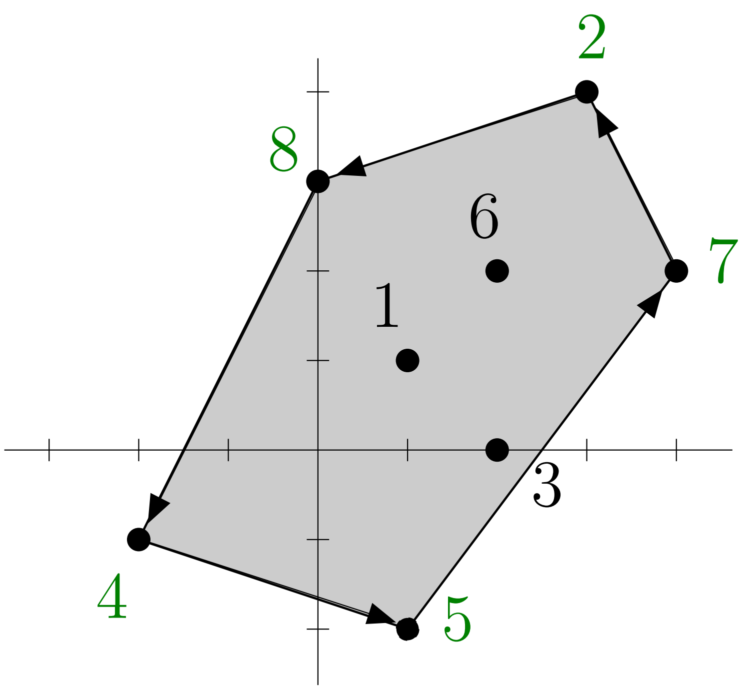


	1	2	3	4	5	6	7	8
X	1	3	2	−2	1	2	4	0
Y	1	4	0	−1	−2	2	2	3

	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).

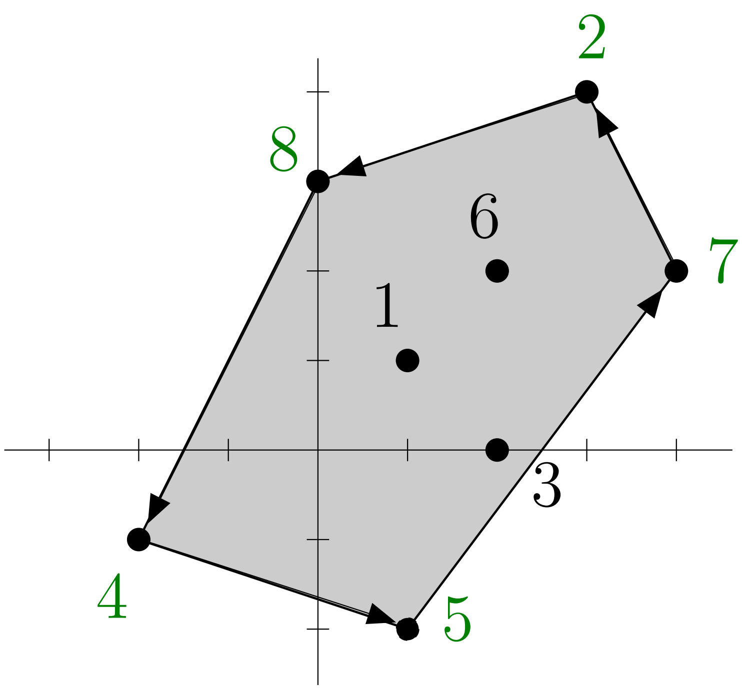


X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

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).



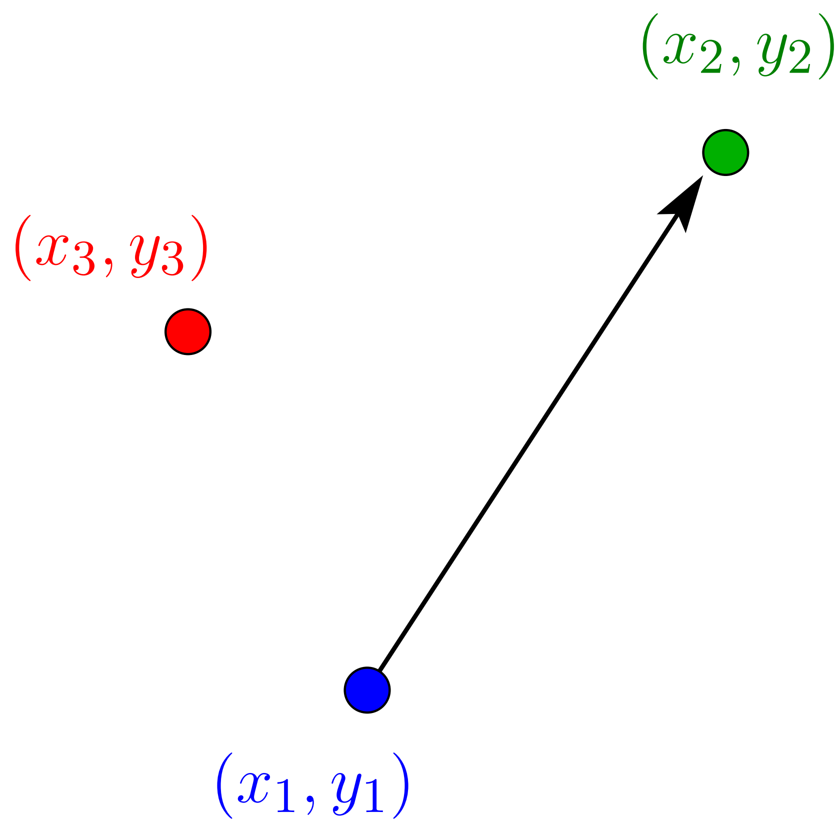
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	2	8	4	5	7
	1	2	3	4	5

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

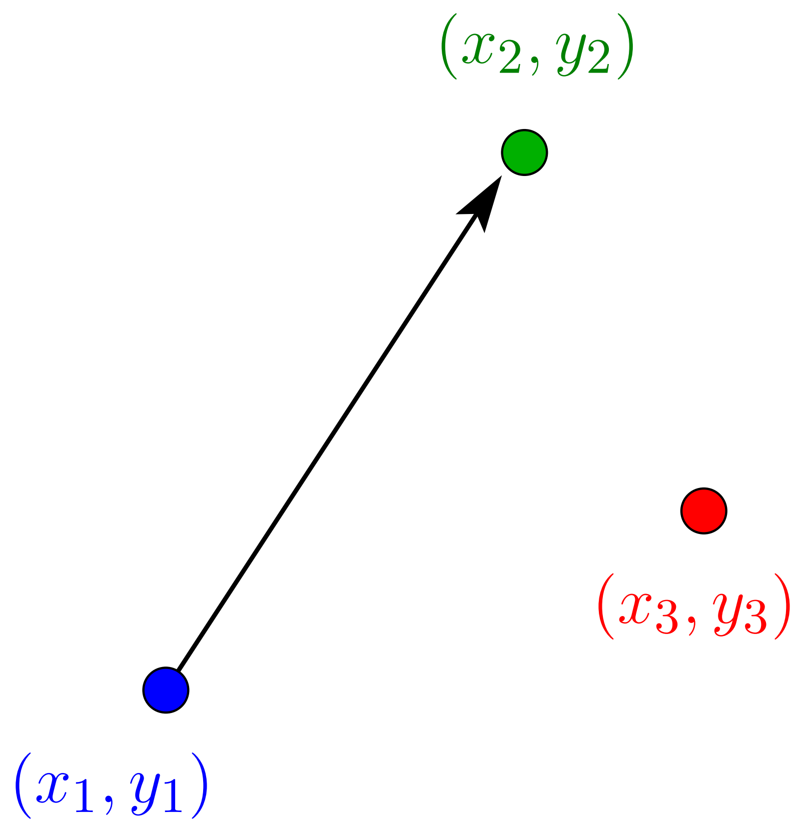
Predicados geométricos

$\textsc{Esquerda}((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \text{VERDADE}$
$\textsc{Direita}((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \text{FALSO}$



Predicados geométricos

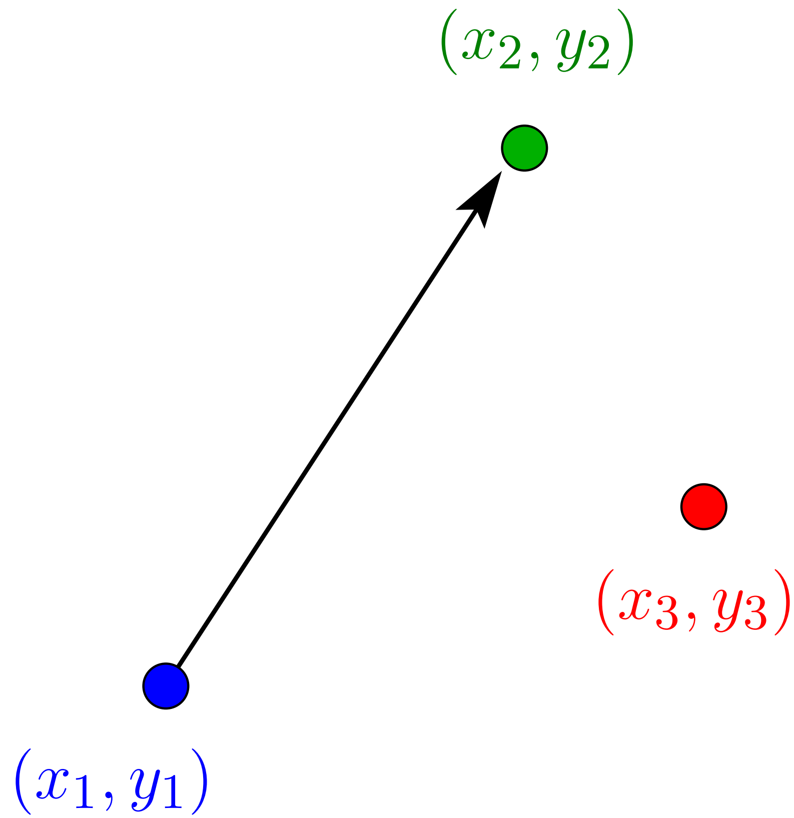
$\text{ESQUERDA}((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \text{FALSO}$
$\text{DIREITA}((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \text{VERDADE}$



Predicados geométricos

$\text{ESQUERDA}((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \text{FALSO}$

$\text{DIREITA}((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \text{VERDADE}$



Vamos supor que não há três pontos colineares.

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

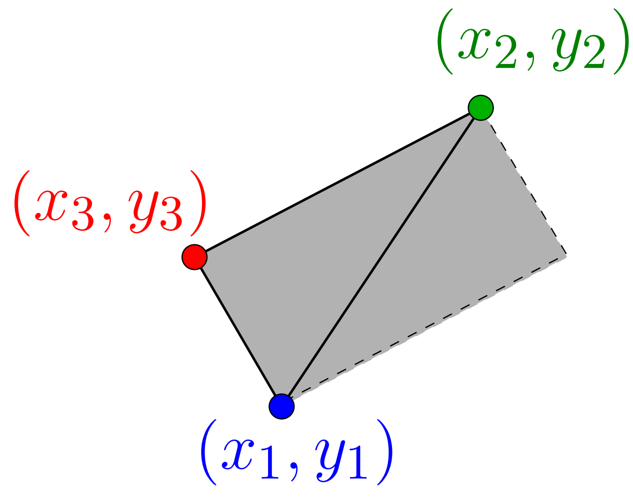
$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1).$$

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).



Abreviaturas

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

Abreviaturas

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

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

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

Abreviaturas

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

$\text{ESQ}(X, Y, i, j, k) =$
$\quad\text{ESQUERDA}((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

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.

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

Em pseudocódigo:

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

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

$\textsc{Dir}(X, Y, i, j, k)$
1 devolva $\textsc{Direita}((X[i], Y[i]), (X[j], Y[j]), (X[k], Y[k]))$

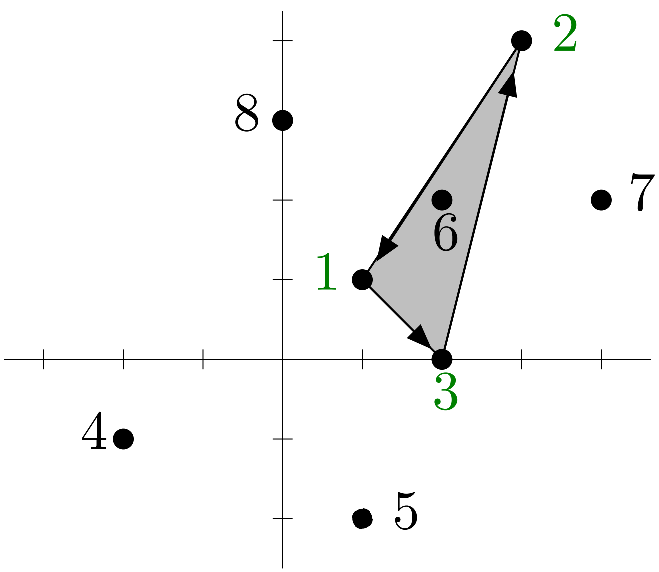
Um algoritmo incremental

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.



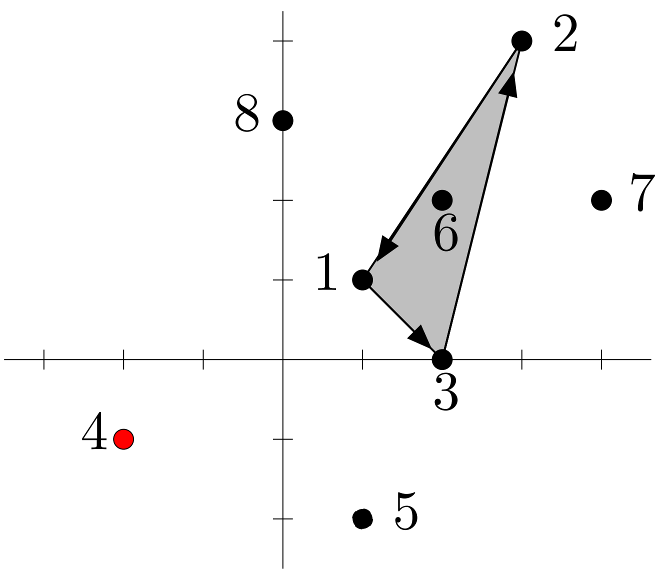
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

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.



	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

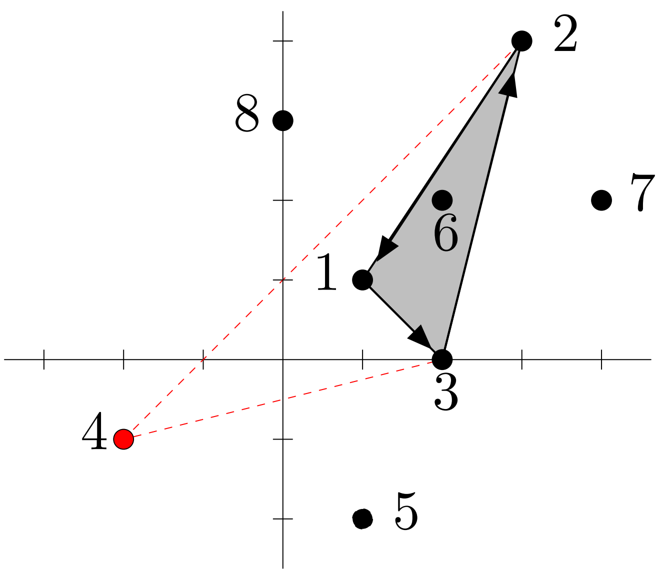
	1	2	3
H	1	3	2

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

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.



	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

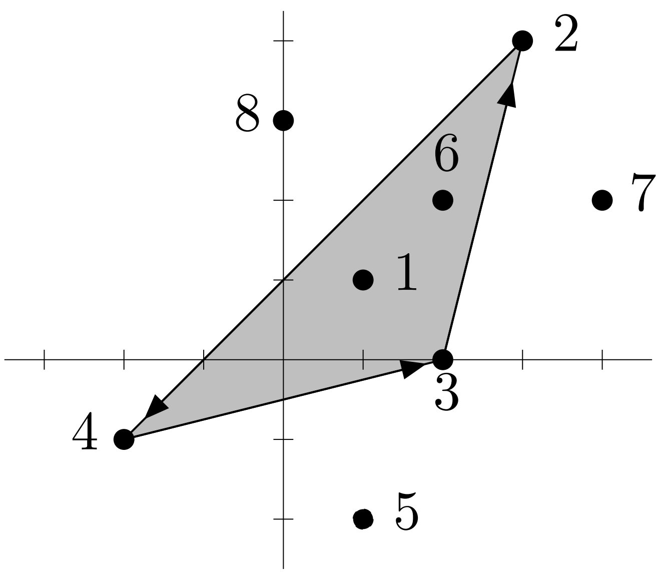
	1	2	3
H	1	3	2

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

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.



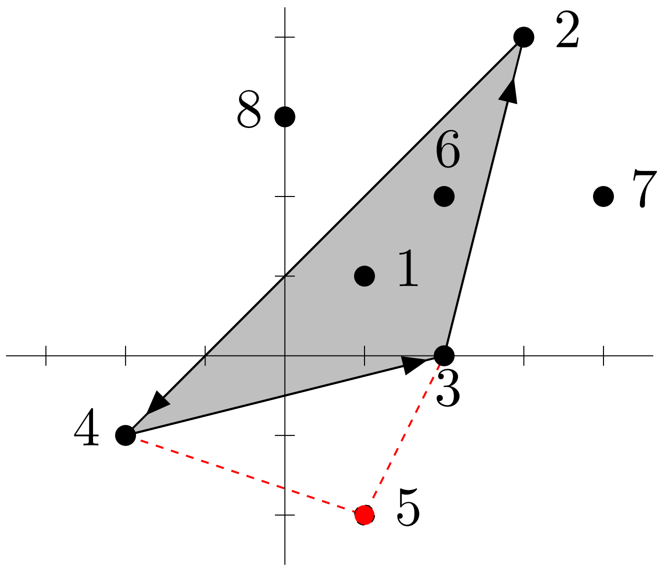
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

H	3	2	4
	1	2	3

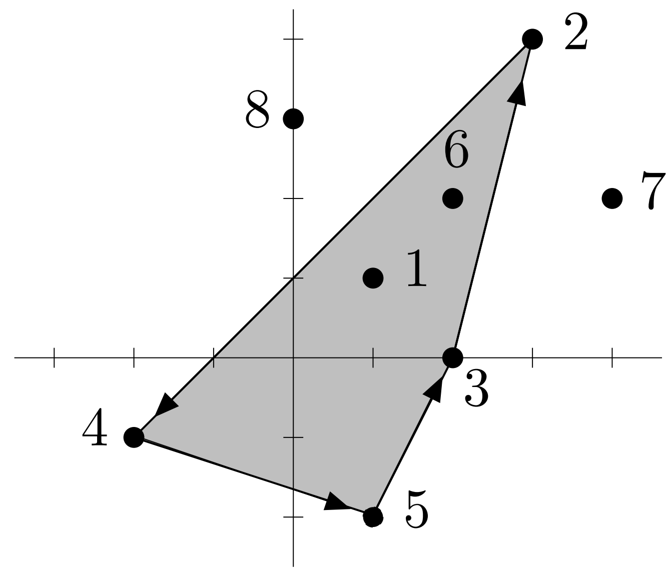
Atualizamos o fecho para incluir o ponto $(-2, -1)$.

Um algoritmo incremental

Próximas iterações...



H	3	2	4
	1	2	3

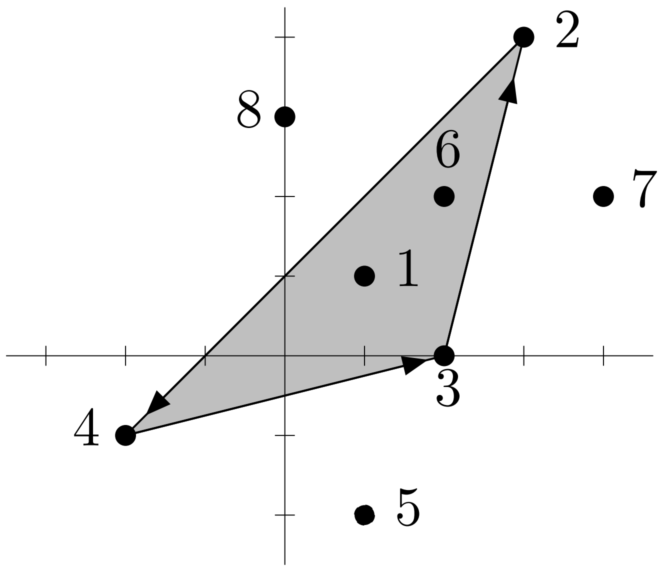


H	3	2	4	5
	1	2	3	4

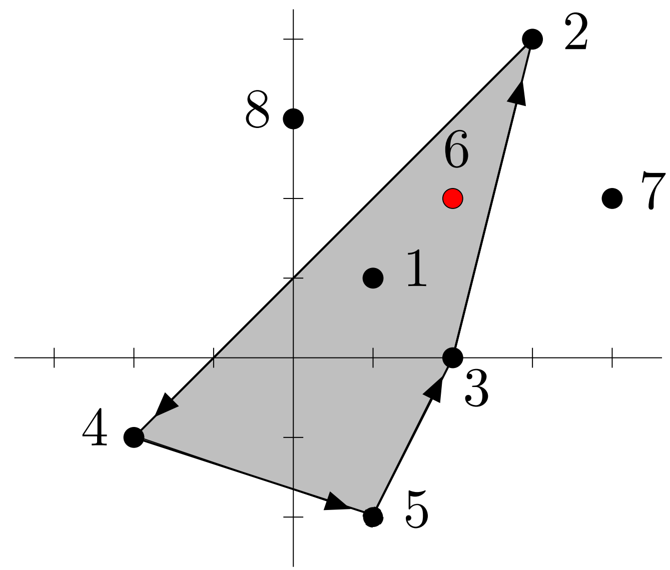
O quinto ponto, $(1, -2)$, pertence ao fecho corrente? Não.

Um algoritmo incremental

Próximas iterações...



H	3	2	4
	1	2	3

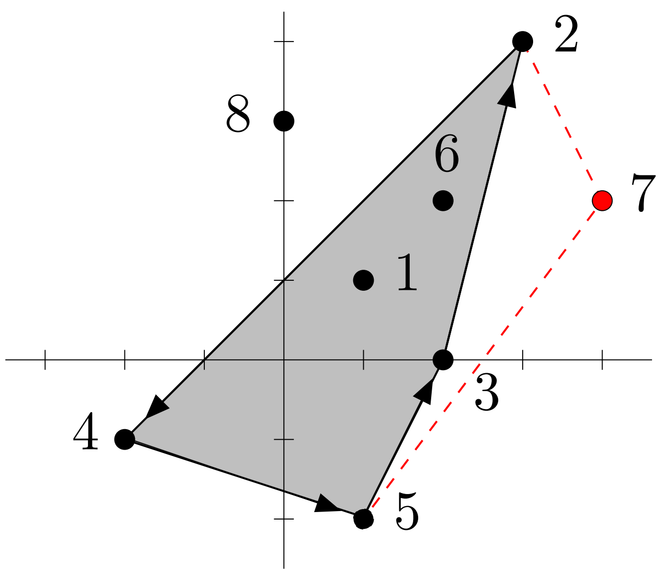


H	3	2	4	5
	1	2	3	4

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...



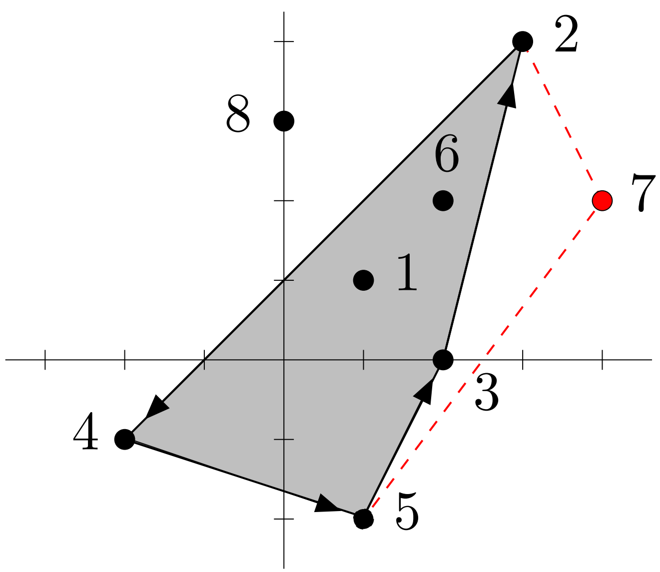
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3
	1	2	3	4	5	6	7	8

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)
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)
```

Um algoritmo incremental

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

```
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 (x, y) está no fecho convexo, dado por $H[1\,..\,h]$, da coleção $X[1\,..\,k-1], Y[1\,..\,k-1]$ 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)
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 (x, y) está no fecho convexo, dado por $H[1\,..\,h]$, da coleção $X[1\,..\,k{-}1], Y[1\,..\,k{-}1]$ de pontos, FALSO caso contrário.

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)
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)
```

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 $H[1] \leftarrow 1 \quad H[2] \leftarrow 2 \quad H[3] \leftarrow 3 \quad h \leftarrow 3$
3   senão $H[1] \leftarrow 1 \quad H[2] \leftarrow 3 \quad H[3] \leftarrow 2 \quad h \leftarrow 3$
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: $\mathrm{O}(n^2)$, pois $h \leq n$ sempre na linha 5.

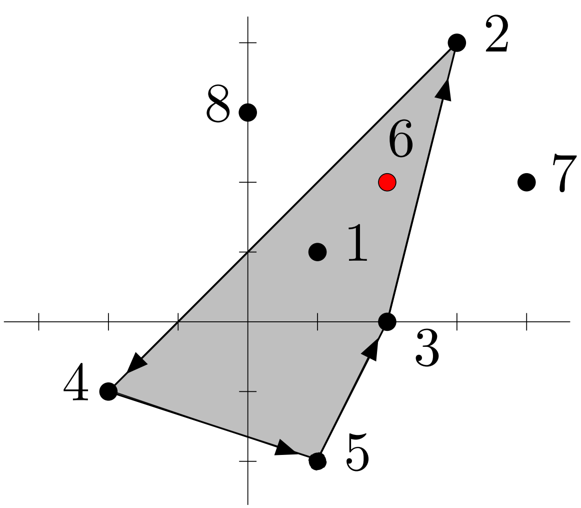
Pertence

```
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
```

Pertence

PERTENCE$(H, h, X, Y, \color{red}{x, y})$

$H[h+1] \leftarrow H[1]$ $\triangleright$ sentinela
para $i \leftarrow 1$ até h faça
  se DIREITA$((X[H[i]], Y[H[i]]), (X[H[i+1]], Y[H[i+1]]), \color{red}{(x, y)})$
    então devolva FALSO
devolva VERDADE



X	1	3	2	-2	1	$\color{red}{2}$	4	0
Y	1	4	0	-1	-2	$\color{red}{2}$	2	3
	1	2	3	4	5	6	7	8

H	3	2	4	5
	1	2	3	4

PERTENCE$(H, 4, X, Y, \color{green}{2, 2})$ = VERDADE

Pertence

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

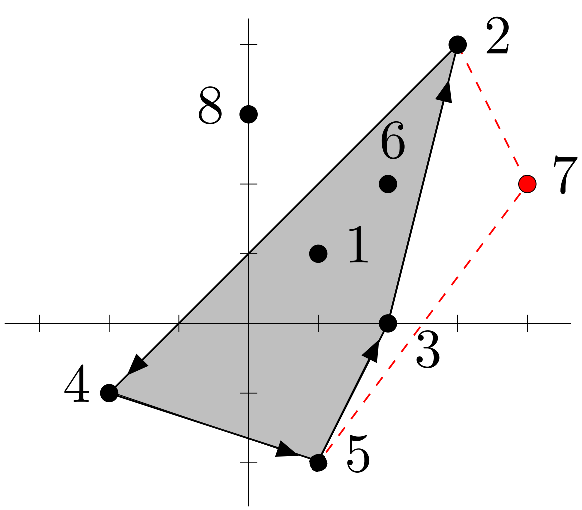
$H[h{+}1] \leftarrow H[1]$ ▷ sentinela

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



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

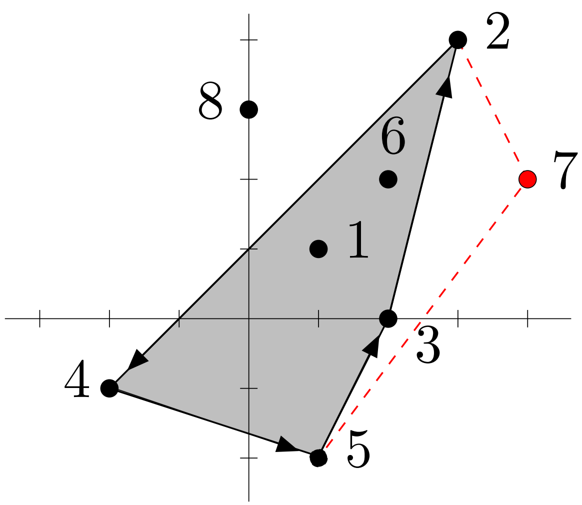
H	3	2	4	5
	1	2	3	4

PERTENCE$(H, 4, X, Y, 4, 2) =$ FALSO

Pertence

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

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

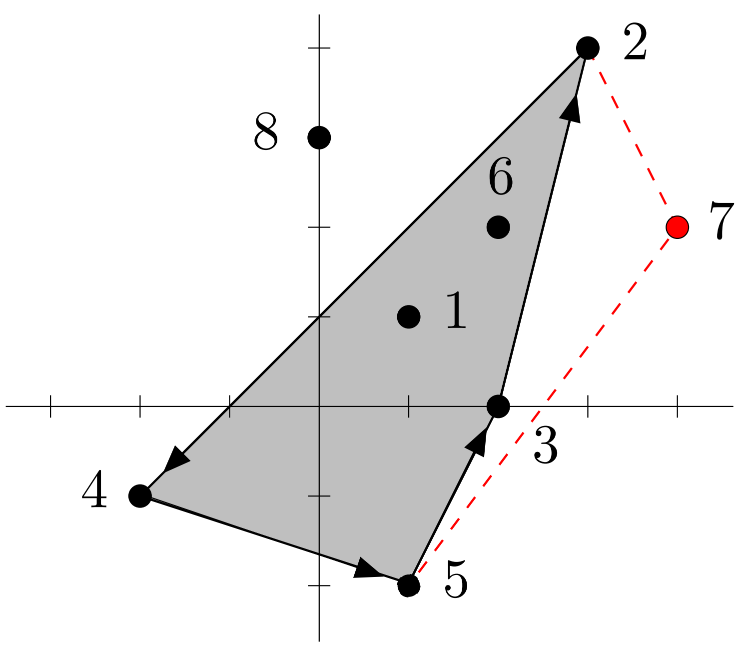
	1	2	3	4
H	3	2	4	5

Consumo de tempo: $\Theta(h)$.

Insere Ponto

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

$H[0] \leftarrow H[h] \qquad H[h{+}1] \leftarrow H[1]$ $\qquad \triangleright$ sentinelas
$i \leftarrow 1$
enquanto $\textsc{Esq}(X, Y, H[i{-}1], H[i], k) = \textsc{Esq}(X, Y, H[i], H[i{+}1], k)$ faça
$\qquad i \leftarrow i + 1$



	i			
H	3	2	4	5
	1	2	3	4

Insere Ponto

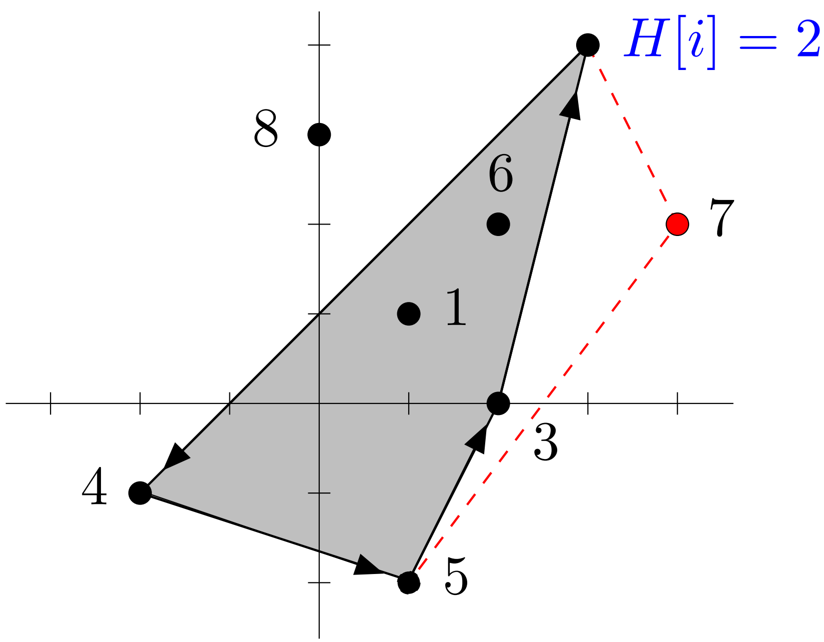
$\textsc{InserePonto}(H, h, X, Y, k)$

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

$i \leftarrow 1$

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

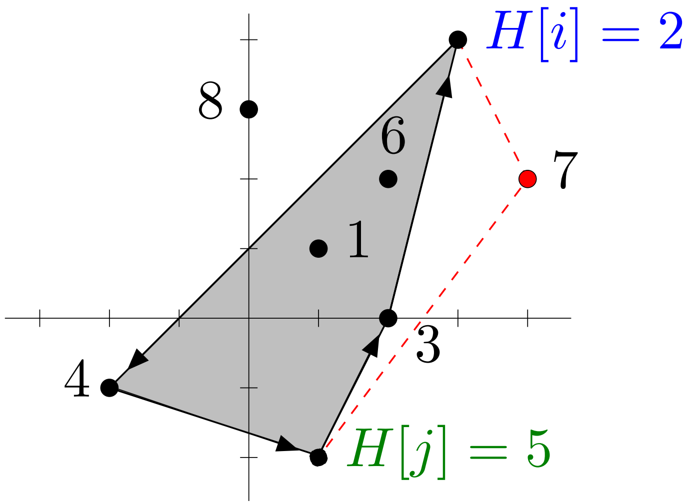
$\quad i \leftarrow i + 1$



	i			
H	3	2	4	5
	1	2	3	4

Insere Ponto

```
InserePonto(H, h, X, Y, k)
  H[0] ← H[h]     H[h+1] ← H[1]     ▷ sentinelas
  i ← 1
  enquanto Esq(X, Y, H[i−1], H[i], k) = Esq(X, Y, H[i], H[i+1], k) faça
      i ← i + 1
  j ← i + 1
  enquanto Esq(X, Y, H[j−1], H[j], k) = Esq(X, Y, H[j], H[j+1], k) faça
      j ← j + 1
```



		i		j
H	3	2	4	5
	1	2	3	4

Insere Ponto

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

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

$i \leftarrow 1$

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

$\qquad i \leftarrow i + 1$

$j \leftarrow i + 1$

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

$\qquad j \leftarrow j + 1$

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

$t \leftarrow 1$

enquanto $i \neq j$ faça

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

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

devolva (F, t)

Insere Ponto

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

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

$i \leftarrow 1$

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

$\quad\quad i \leftarrow i + 1$

$j \leftarrow i + 1$

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

$\quad\quad j \leftarrow j + 1$

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

$t \leftarrow 1$

enquanto $i \neq j$ faça

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

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

devolva (F, t)

Insere Ponto

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

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

$i \leftarrow 1$

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

    $i \leftarrow i + 1$

$j \leftarrow i + 1$

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

    $j \leftarrow j + 1$

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

$t \leftarrow 1$

enquanto $i \neq j$ faça

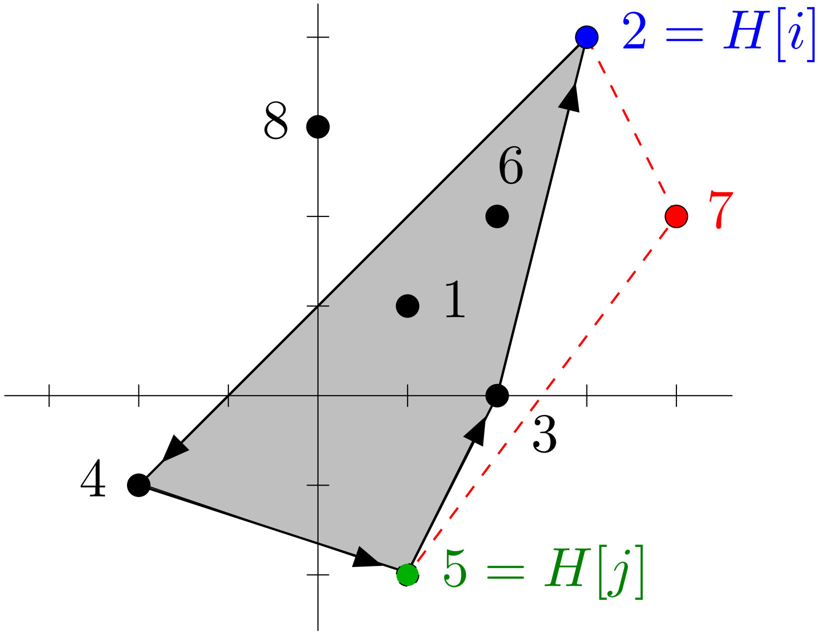
    $F[t] \leftarrow H[i] \quad t \leftarrow t + 1 \quad i \leftarrow (i \bmod h) + 1$

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

devolva (F, t)

Consumo de tempo: $\Theta(h)$.

Insere Ponto

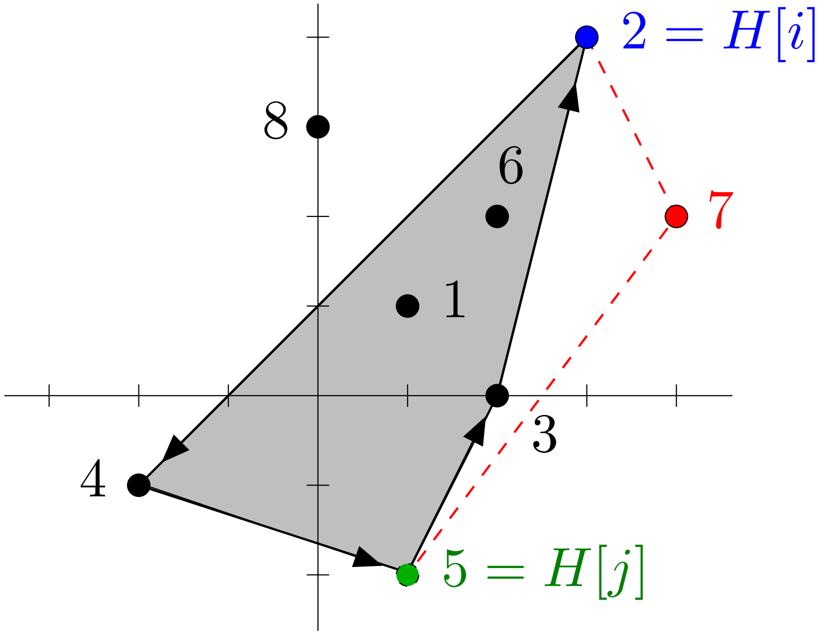


X	1	3	2	−2	1	2	4	0
Y	1	4	0	−1	−2	2	2	3
	1	2	3	4	5	6	7	8

H	3	2	4	5
	1	2	3	4

$i = 2$ e $j = 4$ na linha 9.

Insere Ponto



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

$i = 2$ e $j = 4$ na linha 9.

Ao final

F	2	4	5	7
	1	2	3	4

Um algoritmo incremental

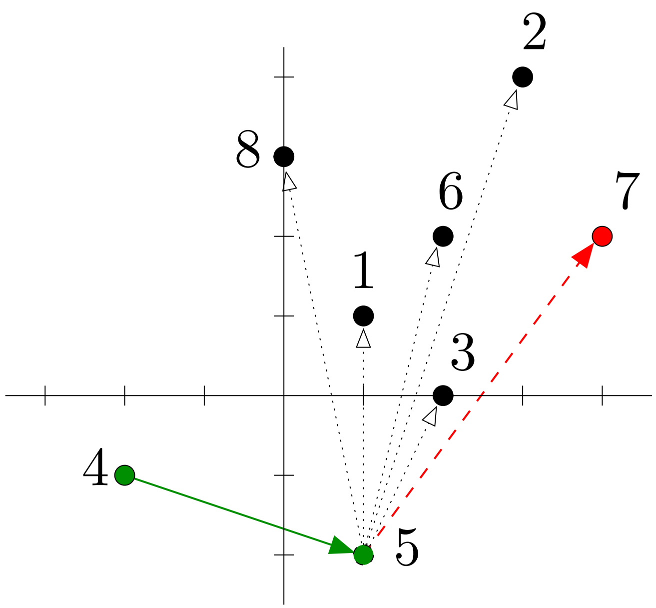
Uma paradinha para olhar uma animação!

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.



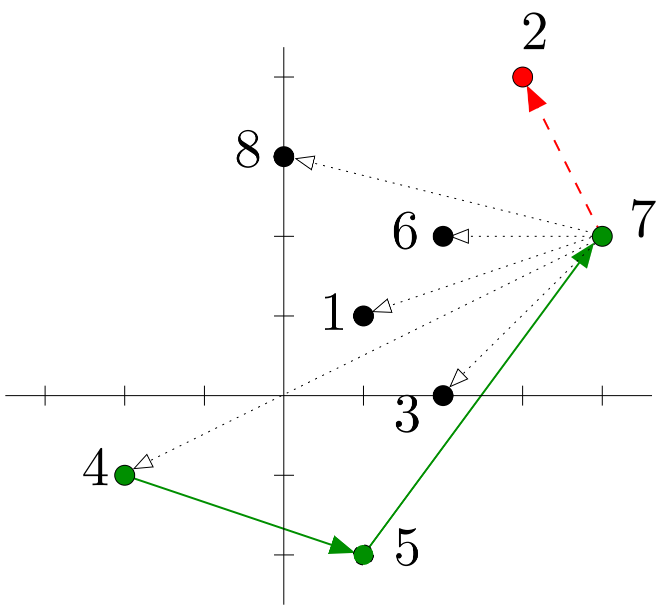
	1	2	3	4	5	6	7	8
X	1	3	2	-2	1	2	4	0
Y	1	4	0	-1	-2	2	2	3

	1	2	3	4
H	4	5	?	$\ldots$

ESQ$(X, Y, 5, 7, j) =$ VERDADE 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.



	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

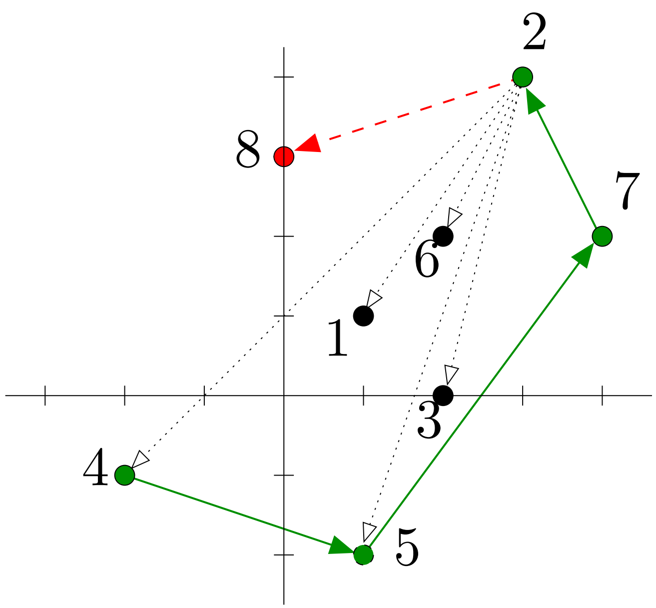
	1	2	3	4	5
H	4	5	7	?	...

ESQ$(X, Y, 5, 7, j) =$ VERDADE para $j = 1, \ldots, 8$

ESQ$(X, Y, 7, 2, j) =$ VERDADE 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	?	$\ldots$
	1	2	3	4	5	6

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

$h \leftarrow 0$

$H[0] \leftarrow \min\{i \in [1 \,..\, n] : X[i] \leq X[j], 1 \leq j \leq n\}$

repita

  $i \leftarrow (H[h]$ mod $h) + 1$ $\triangleright$ qualquer ponto distinto de $H[h]$

  para $j \leftarrow 1$ até n faça

    se DIR$(X, Y, H[h], i, j)$ então $i \leftarrow j$

  $h \leftarrow h + 1$

  $H[h] \leftarrow i$

até que $i = H[0]$ $\triangleright$ fechou o polígono

devolva (H, h)

Embrulho de presente

EMBRULHO(X, Y, n)

$h \leftarrow 0$
$H[0] \leftarrow \min\{i \in [1\,..\,n] : X[i] \leq X[j], 1 \leq j \leq n\}$
repita
    $i \leftarrow (H[h] \bmod h) + 1$ ▷ qualquer ponto distinto de $H[h]$
    para $j \leftarrow 1$ até n faça
        se DIR$(X, Y, H[h], i, j)$ então $i \leftarrow j$
    $h \leftarrow h + 1$
    $H[h] \leftarrow i$
até que $i = H[0]$ ▷ fechou o polígono
devolva (H, h)

Embrulho de presente

EMBRULHO(X, Y, n)

$h \leftarrow 0$
$H[0] \leftarrow \min\{i \in [1\,..\,n] : X[i] \leq X[j], 1 \leq j \leq n\}$
repita
    $i \leftarrow (H[h] \bmod h) + 1$ ▷ qualquer ponto distinto de $H[h]$
    para $j \leftarrow 1$ até n faça
        se DIR$(X, Y, H[h], i, j)$ então $i \leftarrow j$
    $h \leftarrow h + 1$
    $H[h] \leftarrow i$
até que $i = H[0]$ ▷ fechou o polígono
devolva (H, h)

Invariante: $H[1\,..\,h]$ contém pontos extremos da coleção, "consecutivos" no sentido anti-horário.

Embrulho de presente

EMBRULHO(X, Y, n)

$h \leftarrow 0$
$H[0] \leftarrow \min\{i \in [1 \,..\, n] : X[i] \leq X[j], 1 \leq j \leq n\}$
repita
  $i \leftarrow (H[h] \bmod h) + 1$ ▷ qualquer ponto distinto de $H[h]$
  para $j \leftarrow 1$ até n faça
    se DIR$(X, Y, H[h], i, j)$ então $i \leftarrow j$
  $h \leftarrow h + 1$
  $H[h] \leftarrow i$
até que $i = H[0]$ ▷ fechou o polígono
devolva (H, h)

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

Embrulho de presente

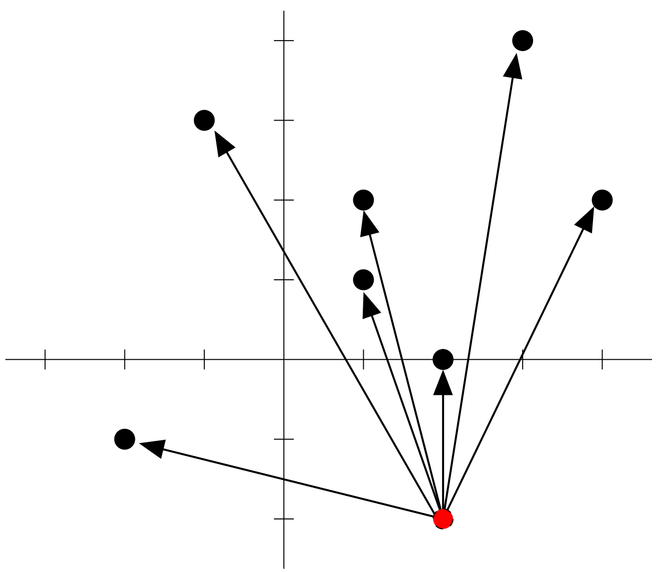
Uma paradinha para olhar uma animação!

Algoritmo de Graham

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

Algoritmo de Graham

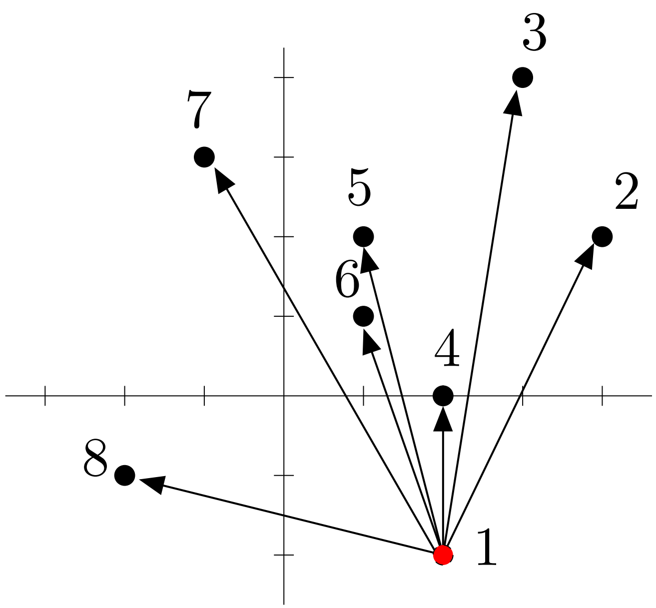
Ideia: primeiro fazemos uma “ordenação angular” dos pontos em torno do ponto de menor Y-coordenada.



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

Algoritmo de Graham

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



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

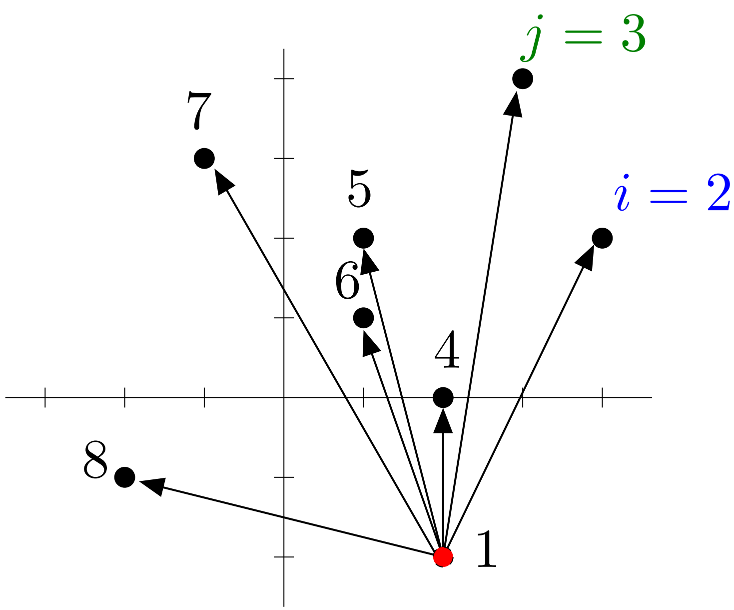
Pré-processamento do Graham

ORDENA-G(X, Y, n)

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

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

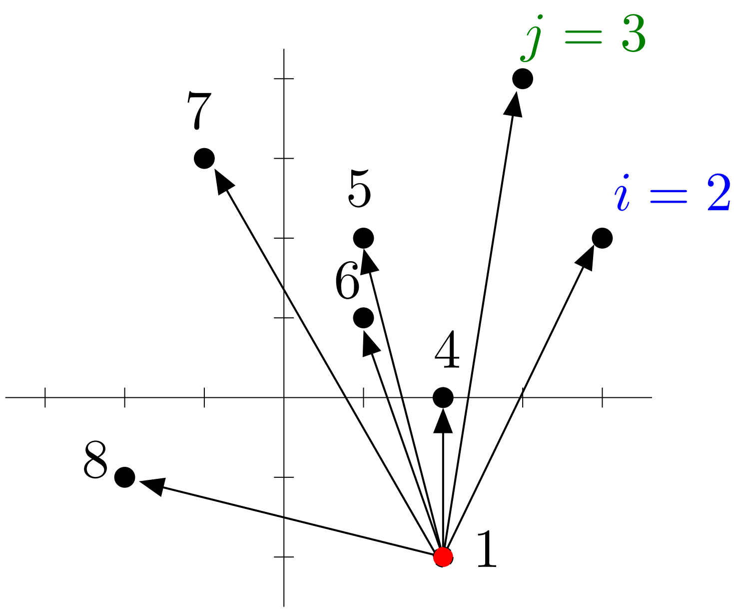
MERGESORT-G$(X, Y, 2, n)$



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

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$(X, Y, 2, n)$



	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

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

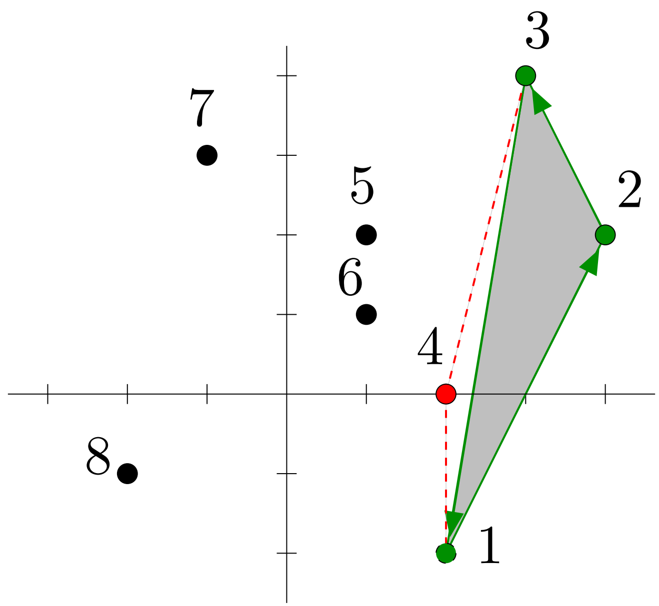
Algoritmo de Graham

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.

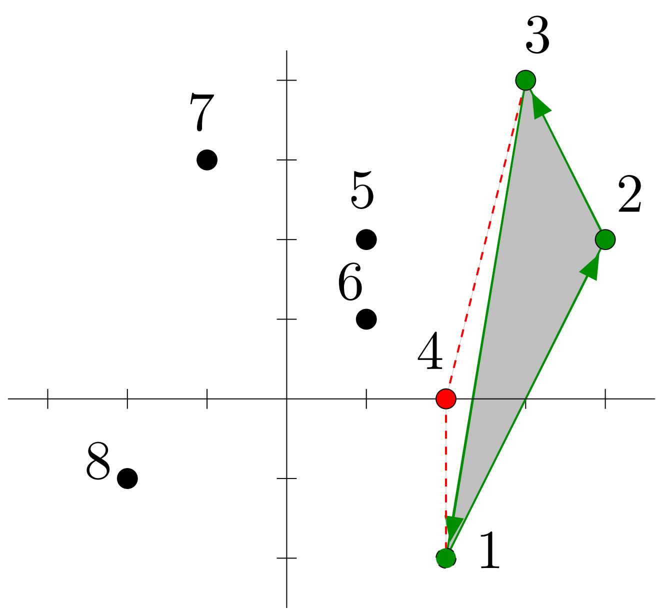


H	1	2	3
	1	2	3

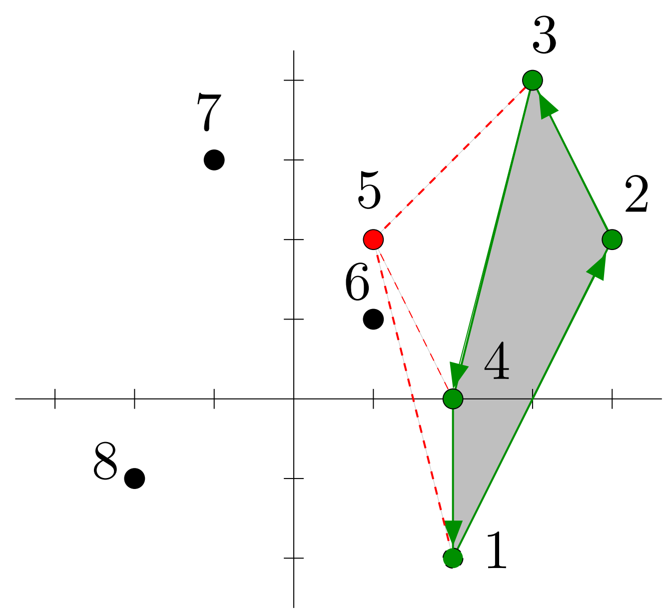
Algoritmo de Graham

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

Começamos com os três primeiros pontos.



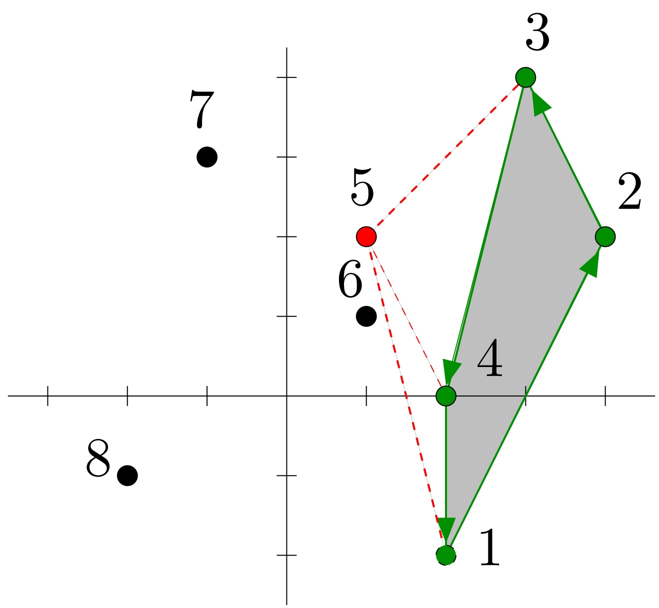
	1	2	3
H	1	2	3



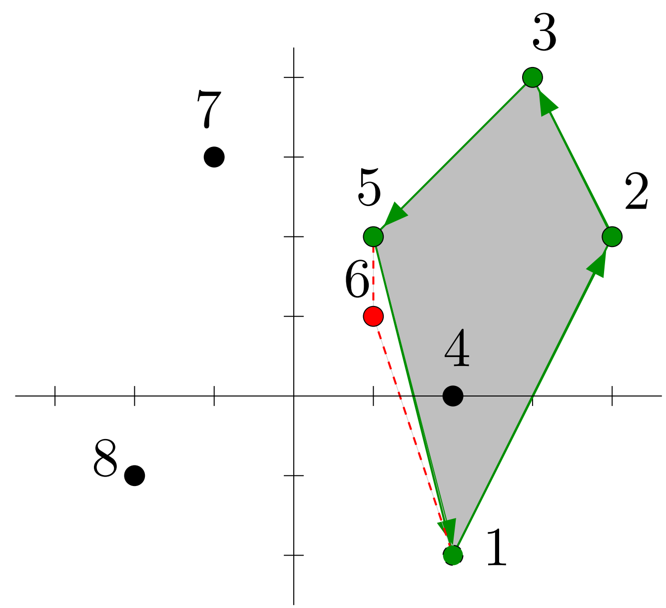
	1	2	3	4
H	1	2	3	4

Algoritmo de Graham

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



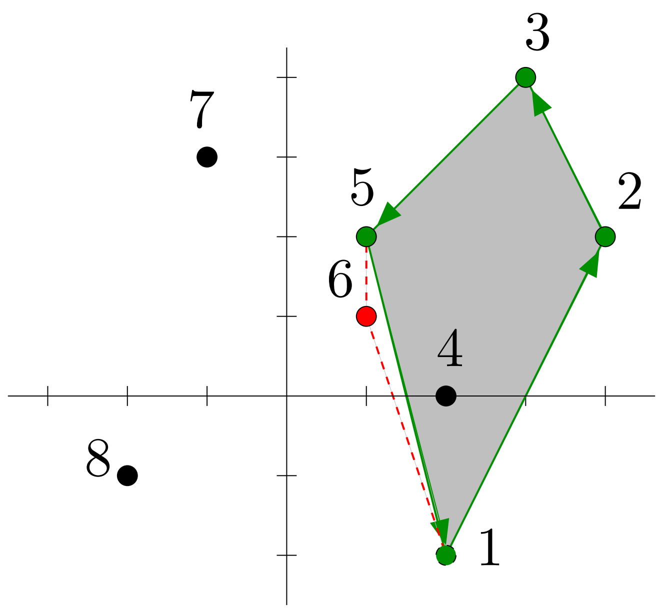
H	1	2	3	4
	1	2	3	4

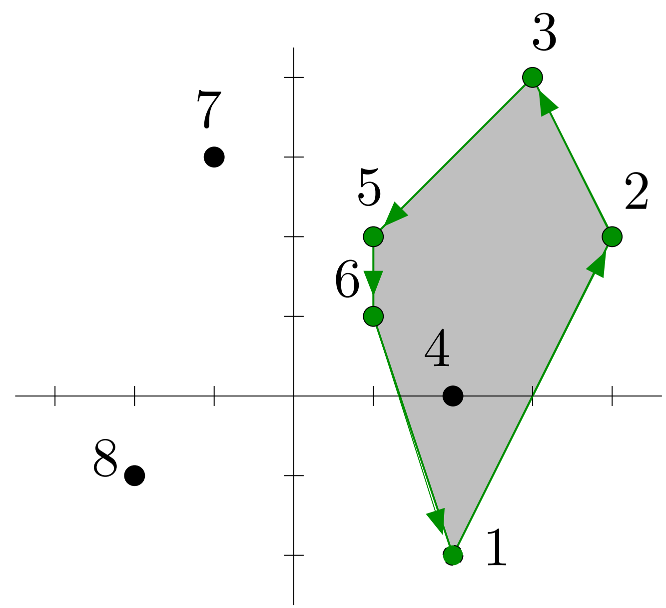


H	1	2	3	5
	1	2	3	4

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.

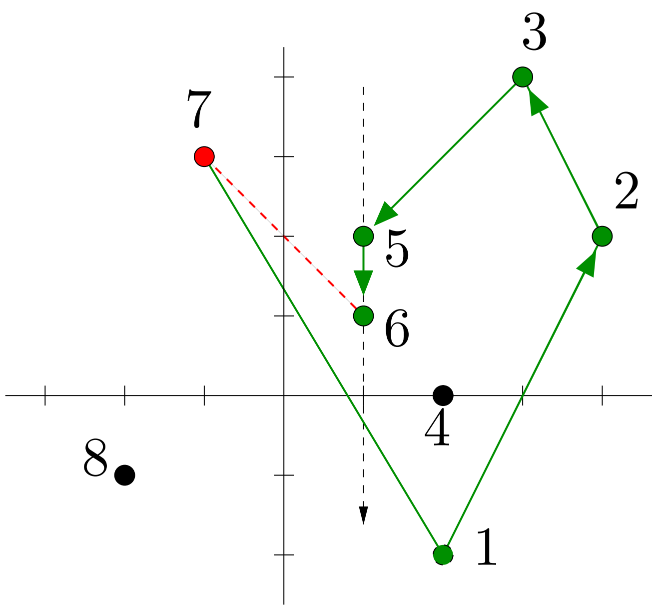
GRAHAM(X, Y, n)
  ORDENA-G(X, Y, n)
  $H[1] \leftarrow 1 \quad H[2] \leftarrow 2 \quad H[3] \leftarrow 3 \quad h \leftarrow 3$
  para $k \leftarrow 4$ até n faça
      enquanto ESQ$(X, Y, H[h], H[h-1], k)$ faça
          $h \leftarrow h - 1$
      $h \leftarrow h + 1 \quad H[h] \leftarrow k$
  devolva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

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
    enquanto $\textsc{Esq}(X, Y, H[h], H[h{-}1], k)$ faça $\triangleright\ h \geq 2$
        $h \leftarrow h - 1$
    $h \leftarrow h + 1 \quad H[h] \leftarrow k$

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
    enquanto ESQ$(X, Y, H[h], H[h{-}1], k)$ faça $\triangleright\ h \geq 2$
        $h \leftarrow h - 1$
    $h \leftarrow h + 1$ $H[h] \leftarrow k$

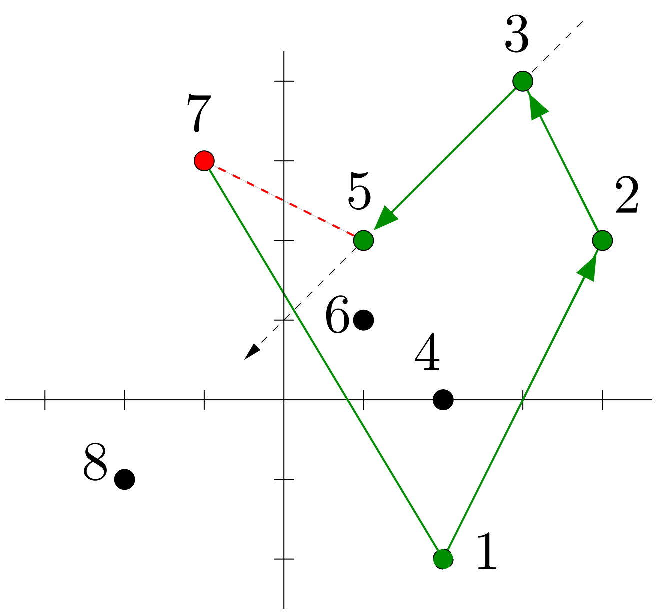


ESQ$(X, Y, 6, 5, 7) =$ VERDADE

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

Algoritmo de Graham

para $\color{red}{k}$ $\leftarrow 4$ até n faça
    enquanto ESQ$(X, Y, \color{green}{H[h]}, \color{green}{H[h-1]}, \color{red}{k})$ faça $\quad \color{blue}{\triangleright\ h \geq 2}$
        $h \leftarrow h - 1$
    $h \leftarrow h + 1 \quad \color{green}{H[h]} \leftarrow \color{red}{k}$

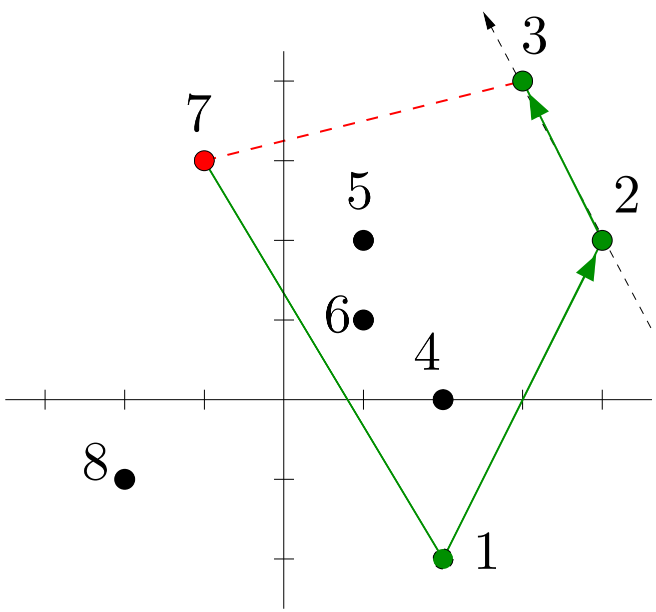


ESQ$(X, Y, 5, 3, 7) =$ VERDADE

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

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
    enquanto ESQ$(X, Y, H[h], H[h-1], k)$ faça $\triangleright\ h \geq 2$
        $h \leftarrow h - 1$
    $h \leftarrow h + 1$ $H[h] \leftarrow k$

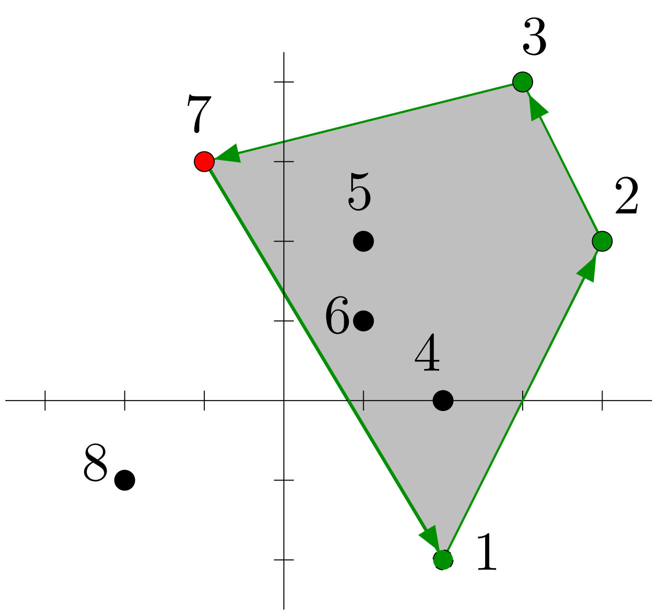


ESQ$(X, Y, 3, 2, 7) =$ FALSO

H	1	2	3
	1	2	3

Algoritmo de Graham

```
para k ← 4 até n faça
    enquanto Esq(X, Y, H[h], H[h−1], k) faça    ▷ h ≥ 2
        h ← h − 1
    h ← h + 1    H[h] ← k
```

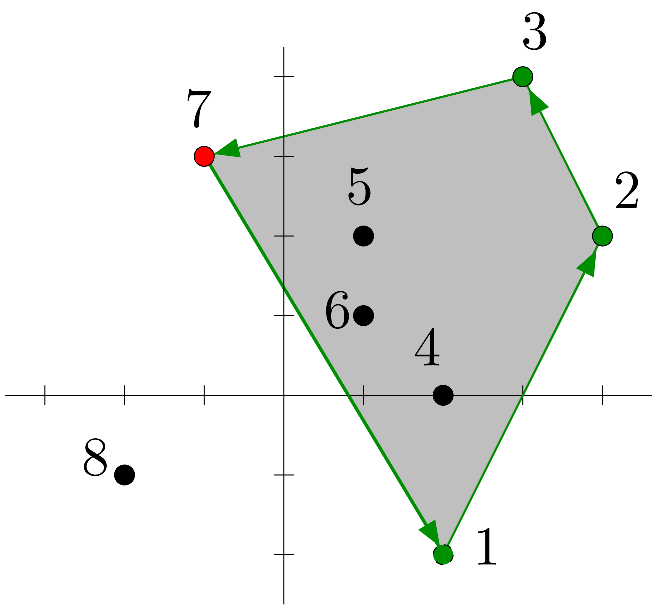


$\text{Esq}(X, Y, 3, 2, 7) = $ FALSO

H				
	1	2	3	7
	1	2	3	4

Algoritmo de Graham

para $k \leftarrow 4$ até n faça
    enquanto $\text{Esq}(X, Y, H[h], H[h-1], k)$ faça $\triangleright\ h \geq 2$
        $h \leftarrow h - 1$
    $h \leftarrow h + 1$ $H[h] \leftarrow k$



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

H	1	2	3	4
	1	2	3	7

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

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
    enquanto ESQ$(X, Y, H[h], H[h-1], k)$ faça
        $h \leftarrow h - 1$
    $h \leftarrow h + 1 \quad H[h] \leftarrow k$
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)

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
    enquanto ESQ$(X, Y, H[h], H[h-1], k)$ faça
        $h \leftarrow h - 1$
    $h \leftarrow h + 1 \quad H[h] \leftarrow k$
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   enquanto ESQ$(X, Y, H[h], H[h-1], k)$ faça

5         $h \leftarrow h - 1$

6     $h \leftarrow h + 1 \quad H[h] \leftarrow k$

7 devolva (H, h)

Consumo de tempo:

Pré-processamento: $\Theta(n \lg n)$

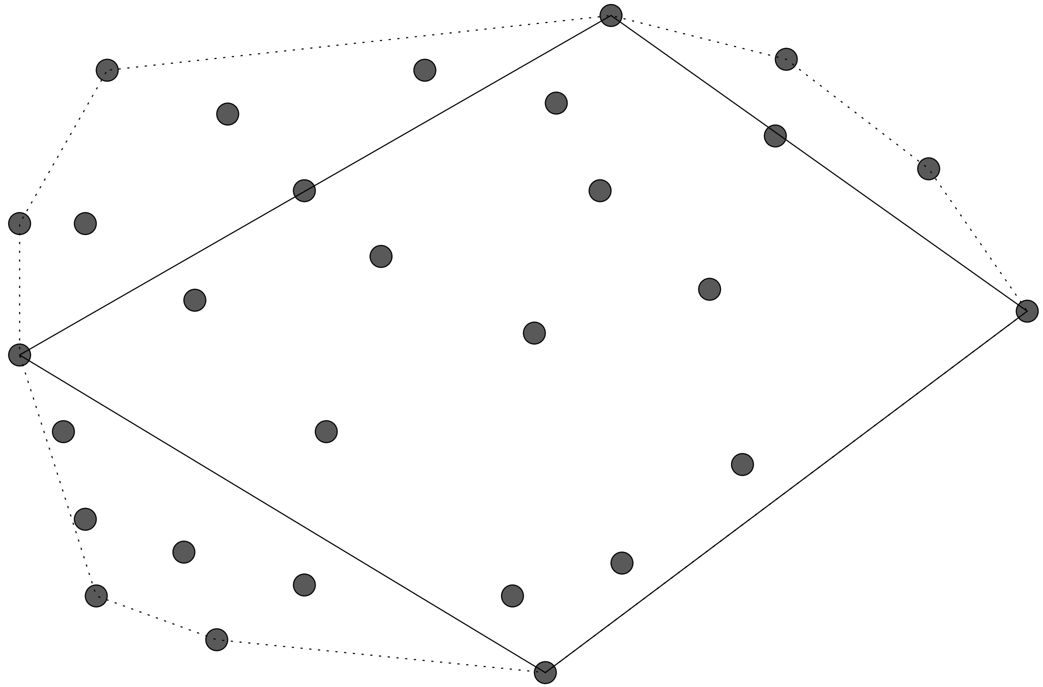
Restante: $\Theta(n)$.

Algoritmo de Graham

Uma paradinha para olhar uma animação!

Quickhull

Ideia: descartar muitos pontos que estão no interior do fecho convexo e concentrar o trabalho nos pontos que estão próximos da fronteira.



Quickhull

Com um pouco mais de detalhe...

Quickhull

Com um pouco mais de detalhe...

Pré-processamento:

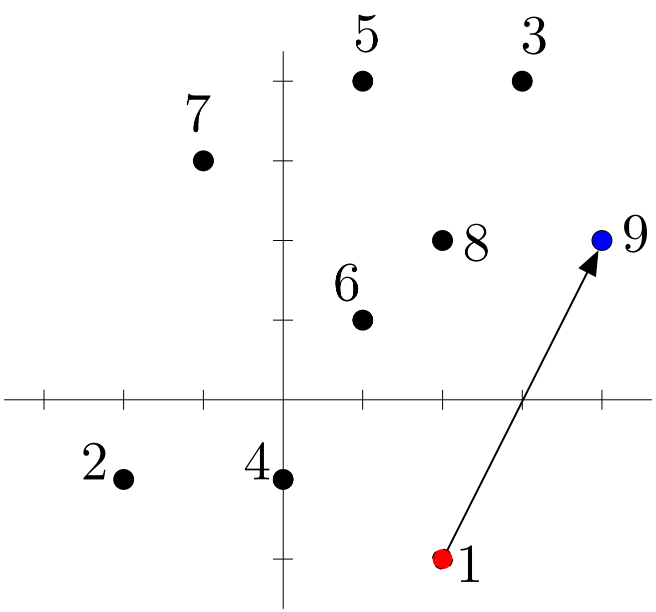
Rearrange os pontos dados de modo que o primeiro e o último sejam extremos “consecutivos” na fronteira do fecho.

Quickhull

Com um pouco mais de detalhe...

Pré-processamento:

Rearrange os pontos dados de modo que o primeiro e o último sejam extremos “consecutivos” na fronteira do fecho.

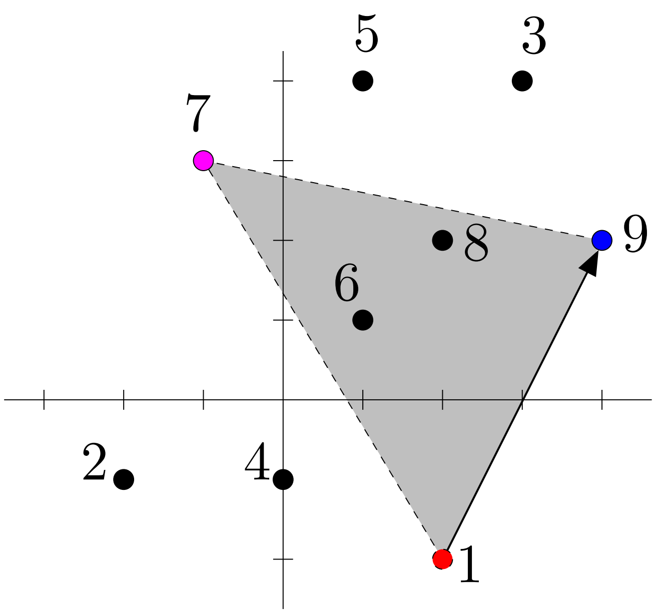


	1	2	3	4	5	6	7	8	9
X	2	-2	3	0	1	1	-1	2	4
Y	-2	-1	4	-1	4	1	3	2	2

Quickhull

Particione:

Encontre ponto extremo "oposto" a estes dois pontos.



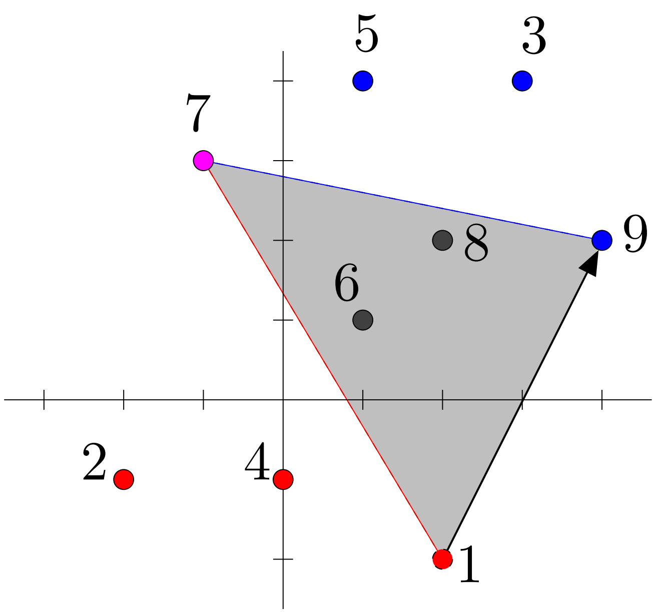
	1	2	3	4	5	6	7	8	9
X	2	-2	3	0	1	1	-1	2	4
Y	-2	-1	4	-1	4	1	3	2	2

Quickhull

Particione:

Encontre ponto extremo “oposto” a estes dois pontos.

Particione os pontos em três grupos:
os dentro do triângulo,
os “acima” do lado vermelho do triângulo,
os “acima” do lado azul do triângulo.



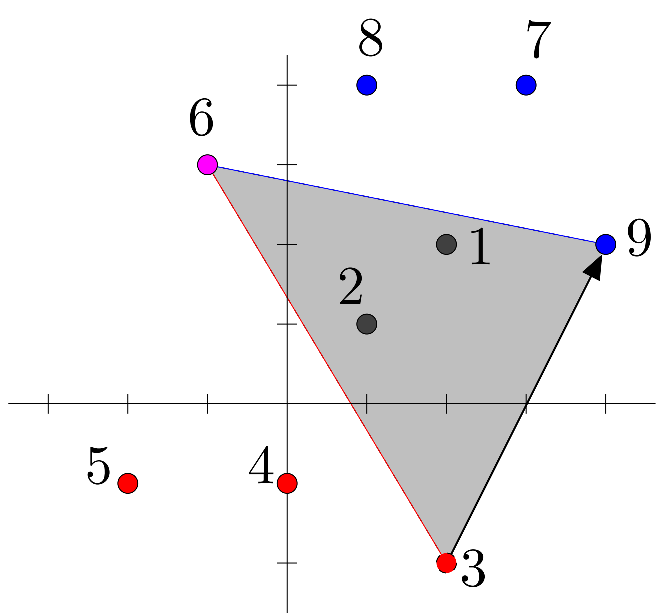
	1	2	3	4	5	6	7	8	9
X	2	−2	3	0	1	1	−1	2	4
Y	−2	−1	4	−1	4	1	3	2	2

Quickhull

Particione:

Encontre ponto extremo “oposto” a estes dois pontos.

Particione os pontos em três grupos:
os dentro do triângulo,
os “acima” do lado vermelho do triângulo,
os “acima” do lado azul do triângulo.



	1	2	3	4	5	6	7	8	9
X	2	1	2	0	−2	−1	3	1	4
Y	2	1	−2	−1	−1	3	4	4	2

Quickhull

QUICKHULL (X, Y, n)

se $n = 1$
　　então $h \leftarrow 1 \quad H[1] \leftarrow 1$
　　senão $k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j], 1 \leq j \leq n\}$
　　　　$(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$
　　　　$i \leftarrow 2$
　　　　para $j \leftarrow 3$ até n faça
　　　　　　se DIR$(X, Y, 1, i, j)$ então $i \leftarrow j$
　　　　$(X[n], Y[n]) \leftrightarrow (X[i], Y[i])$
　　　　$(H, h) \leftarrow$ QUICKHULLREC$(X, Y, 1, n, H, h)$
devolva (H, h)

Quickhull

QUICKHULL (X, Y, n)

se $n = 1$
    então $h \leftarrow 1 \quad H[1] \leftarrow 1$
    senão $k \leftarrow \min\{i \in [1\,..\,n] : Y[i] \leq Y[j], 1 \leq j \leq n\}$
        $(X[1], Y[1]) \leftrightarrow (X[k], Y[k])$
        $i \leftarrow 2$
        para $j \leftarrow 3$ até n faça
            se DIR$(X, Y, 1, i, j)$ então $i \leftarrow j$
        $(X[n], Y[n]) \leftrightarrow (X[i], Y[i])$
        $(H, h) \leftarrow$ QUICKHULLREC$(X, Y, 1, n, H, h)$
devolva (H, h)

Consumo de tempo:
$\mathrm{O}(n)$ mais o tempo do QUICKHULLREC.

Quickhull

QUICKHULLREC (X, Y, p, r, H, h)

se $p = r - 1$ $\triangleright$ há exatamente dois pontos na coleção

    então $h \leftarrow 2$ $\quad H[1] \leftarrow r$ $\quad H[2] \leftarrow p$

    senão $(p', q) \leftarrow$ PARTICIONE(X, Y, p, r)

        $(H_1, h_1) \leftarrow$ QUICKHULLREC(X, Y, q, r, H, h)

        $(H_2, h_2) \leftarrow$ QUICKHULLREC(X, Y, p', q, H, h)

        $\triangleright$ $H \leftarrow H_1 \cdot H_2$ removendo uma cópia do q

        $h \leftarrow 0$

        para $i \leftarrow 1$ até h_1 faça

            $h \leftarrow h + 1$ $\quad H[h] \leftarrow H_1[i]$

        para $i \leftarrow 2$ até h_2 faça

            $h \leftarrow h + 1$ $\quad H[h] \leftarrow H_2[i]$

devolva (H, h)

Quickhull

QuickHullRec (X, Y, p, r, H, h)

se $p = r - 1$ $\triangleright$ há exatamente dois pontos na coleção
  então $h \leftarrow 2$ $H[1] \leftarrow r$ $H[2] \leftarrow p$
  senão $(p', q) \leftarrow$ Particione(X, Y, p, r)
    $(H_1, h_1) \leftarrow$ QuickHullRec(X, Y, q, r, H, h)
    $(H_2, h_2) \leftarrow$ QuickHullRec(X, Y, p', q, H, h)
    $\triangleright$ $H \leftarrow H_1 \cdot H_2$ removendo uma cópia do q
    $h \leftarrow 0$
    para $i \leftarrow 1$ até h_1 faça
      $h \leftarrow h + 1$ $H[h] \leftarrow H_1[i]$
    para $i \leftarrow 2$ até h_2 faça
      $h \leftarrow h + 1$ $H[h] \leftarrow H_2[i]$
devolva (H, h)

Consumo de tempo:

$O(n^2)$, como o QuickSort, onde $n = r - p + 1$.

Particione

PARTICIONE (X, Y, p, r)

$q \leftarrow$ PONTOEXTREMO(X, Y, p, r)
$(X[q], Y[q]) \leftrightarrow (X[p{+}1], Y[p{+}1])$
$p' \leftarrow r \quad q \leftarrow r$
para $k \leftarrow r - 1$ decrescendo até $p + 2$ faça
    se ESQ$(X, Y, p, p{+}1, k)$
        então $p' \leftarrow p' - 1 \quad (X[p'], Y[p']) \leftrightarrow (X[k], Y[k])$
        senão se ESQ$(X, Y, p{+}1, r, k)$
            então $q \leftarrow q{-}1 \quad (X[q], Y[q]) \leftrightarrow (X[k], Y[k])$
                $p' \leftarrow p'{-}1 \quad (X[k], Y[k]) \leftrightarrow (X[p'], Y[p'$
$q \leftarrow q - 1 \quad (X[q], Y[q]) \leftrightarrow (X[p{+}1], Y[p{+}1])$
$p' \leftarrow p' - 1 \quad (X[p'], Y[p']) \leftrightarrow (X[p{+}1], Y[p{+}1])$
$p' \leftarrow p' - 1 \quad (X[p'], Y[p']) \leftrightarrow (X[p], Y[p])$
devolva (p', q)

Particione

PARTICIONE (X, Y, p, r)

```
q ← PONTOEXTREMO(X, Y, p, r)
(X[q], Y[q]) ↔ (X[p+1], Y[p+1])
p′ ← r    q ← r
para k ← r − 1 decrescendo até p + 2 faça
      se ESQ(X, Y, p, p+1, k)
            então p′ ← p′ − 1    (X[p′], Y[p′]) ↔ (X[k], Y[k])
            senão se ESQ(X, Y, p+1, r, k)
                        então q ← q−1    (X[q], Y[q]) ↔ (X[k], Y[k])
                              p′ ← p′−1   (X[k], Y[k]) ↔ (X[p′], Y[p′
q ← q − 1      (X[q], Y[q]) ↔ (X[p+1], Y[p+1])
p′ ← p′ − 1    (X[p′], Y[p′]) ↔ (X[p+1], Y[p+1])
p′ ← p′ − 1    (X[p′], Y[p′]) ↔ (X[p], Y[p])
devolva (p′, q)
```

Consumo de tempo: $\Theta(n)$ onde $n = r - p + 1$.

Quickhull

Uma paradinha para olhar uma animação!