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

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

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

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

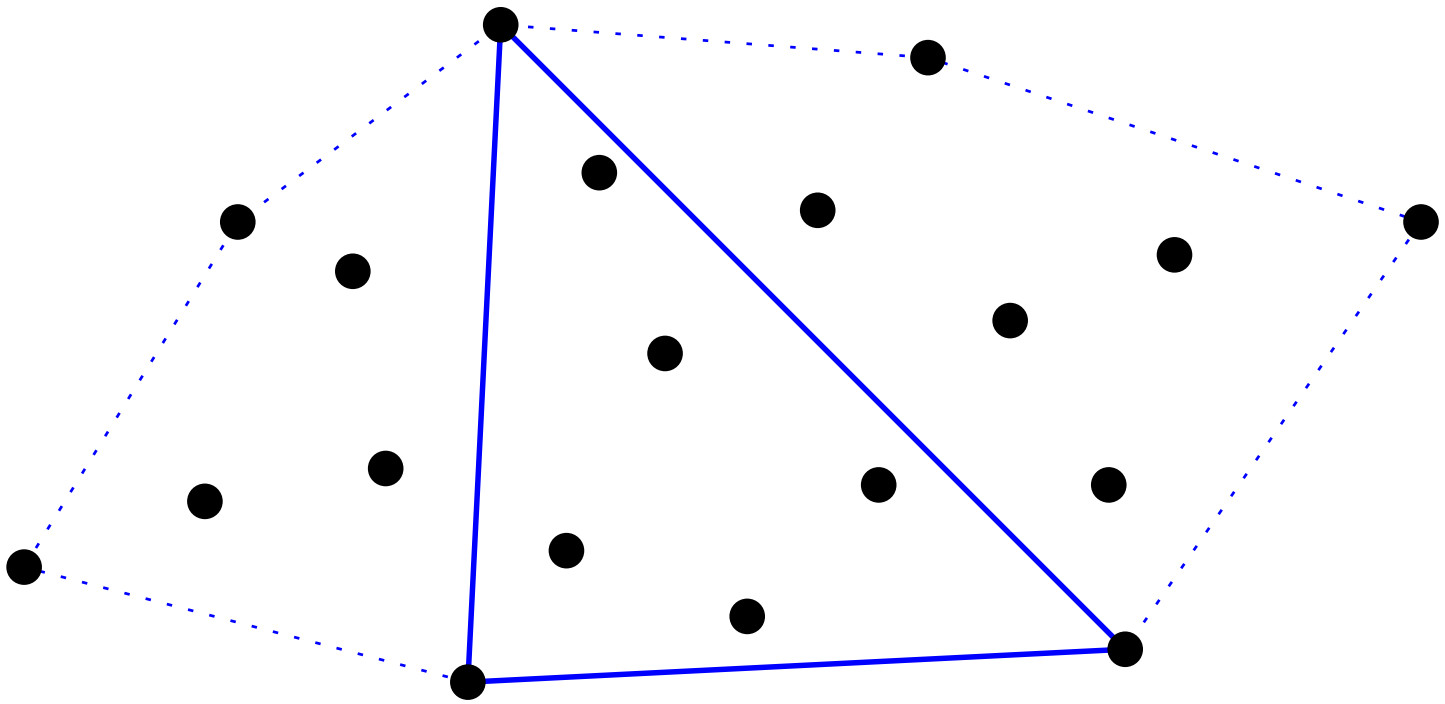
Fecho convexo de P: conjunto de combinações convexas de pontos de P, ou seja,

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

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

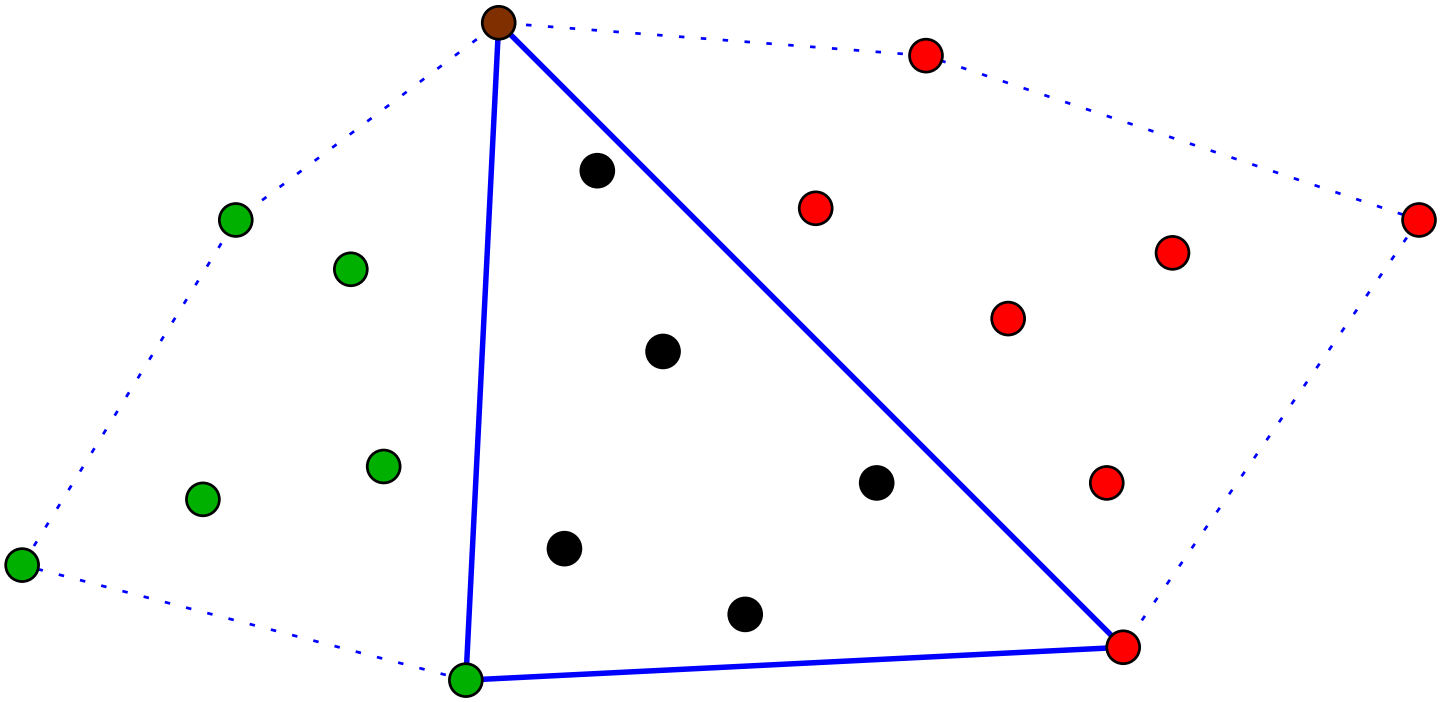
Quickhull

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



Quickhull

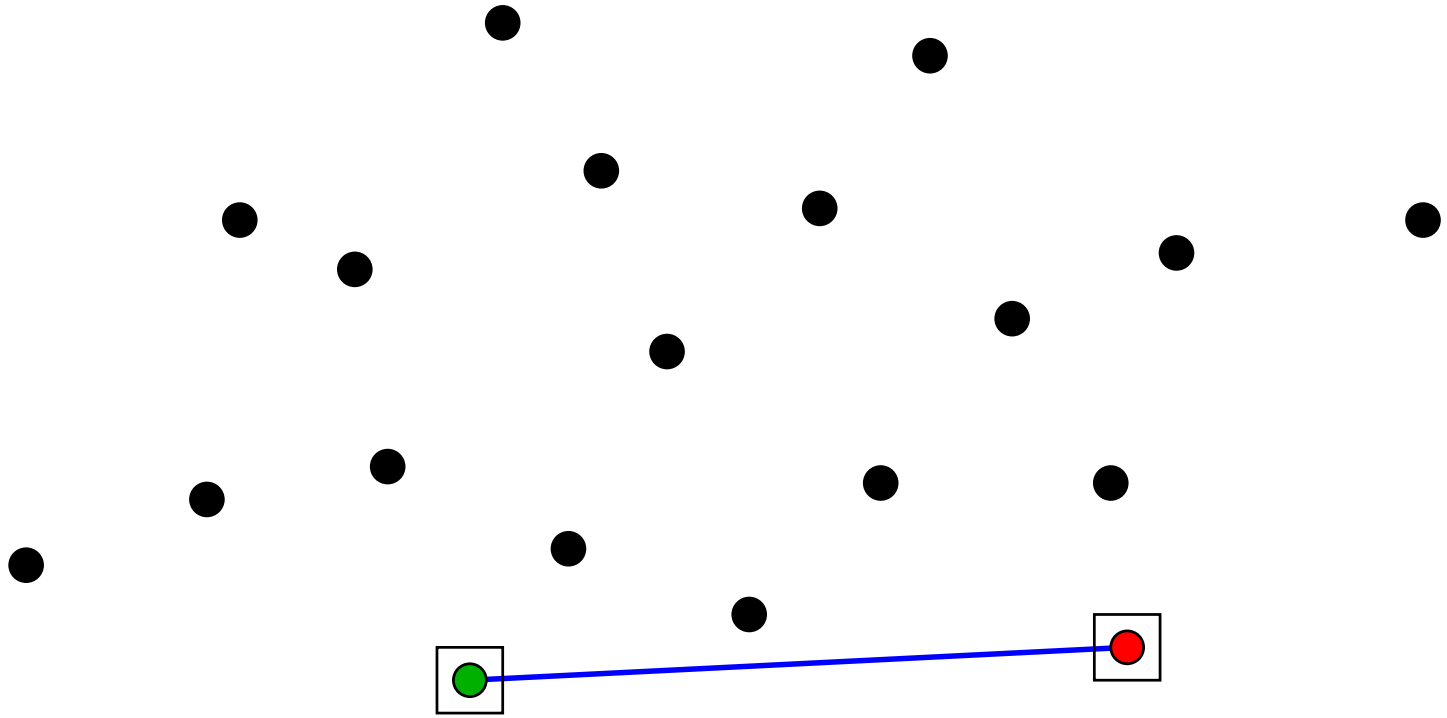
Ideia: descartar pontos que estão no interior do fecho convexo e concentrar o trabalho nos pontos que estão próximos da fronteira.



Monta coleções da esquerda e da direita
e aplica recursão nelas.

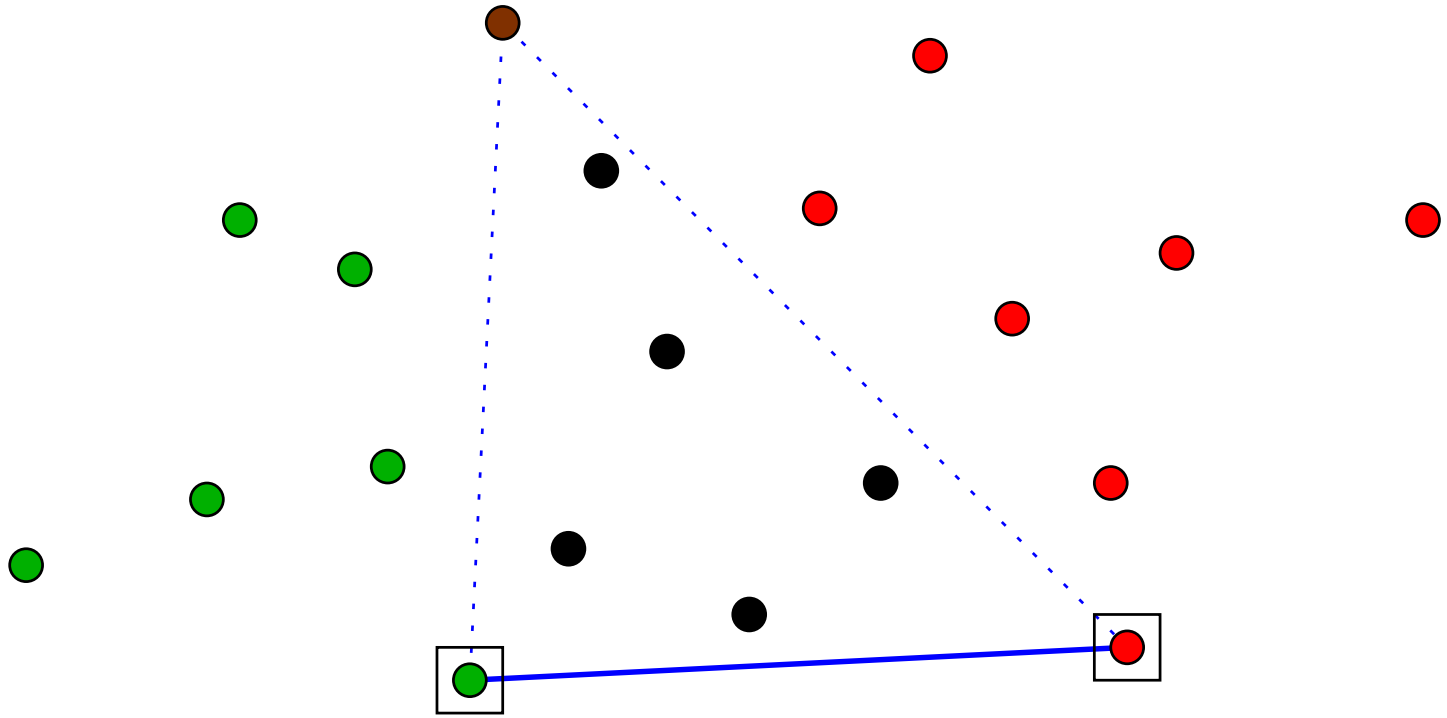
Como dividir a coleção?

Começamos com dois extremos consecutivos do fecho.



Como dividir a coleção?

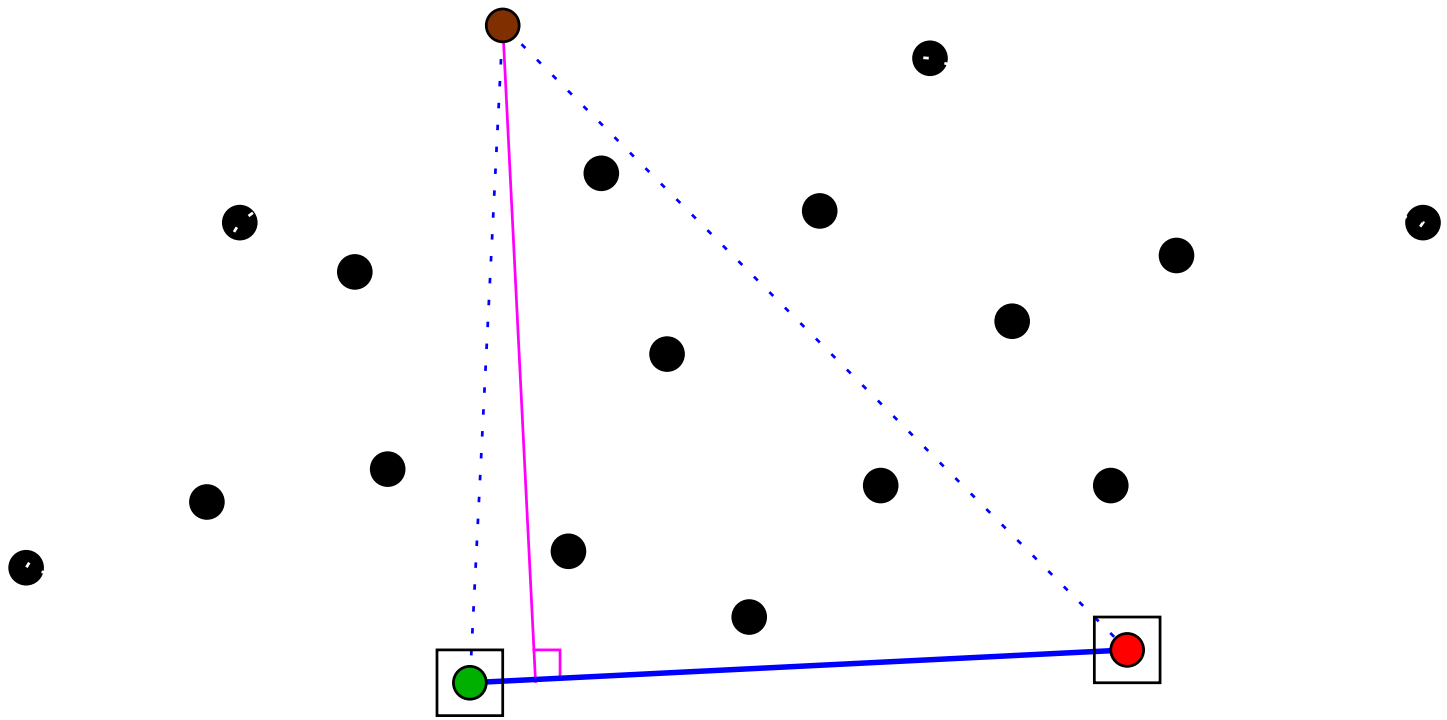
Começamos com dois extremos consecutivos do fecho.



Então precisamos encontrar o extremo marrom
e dividir em três a coleção de pontos.

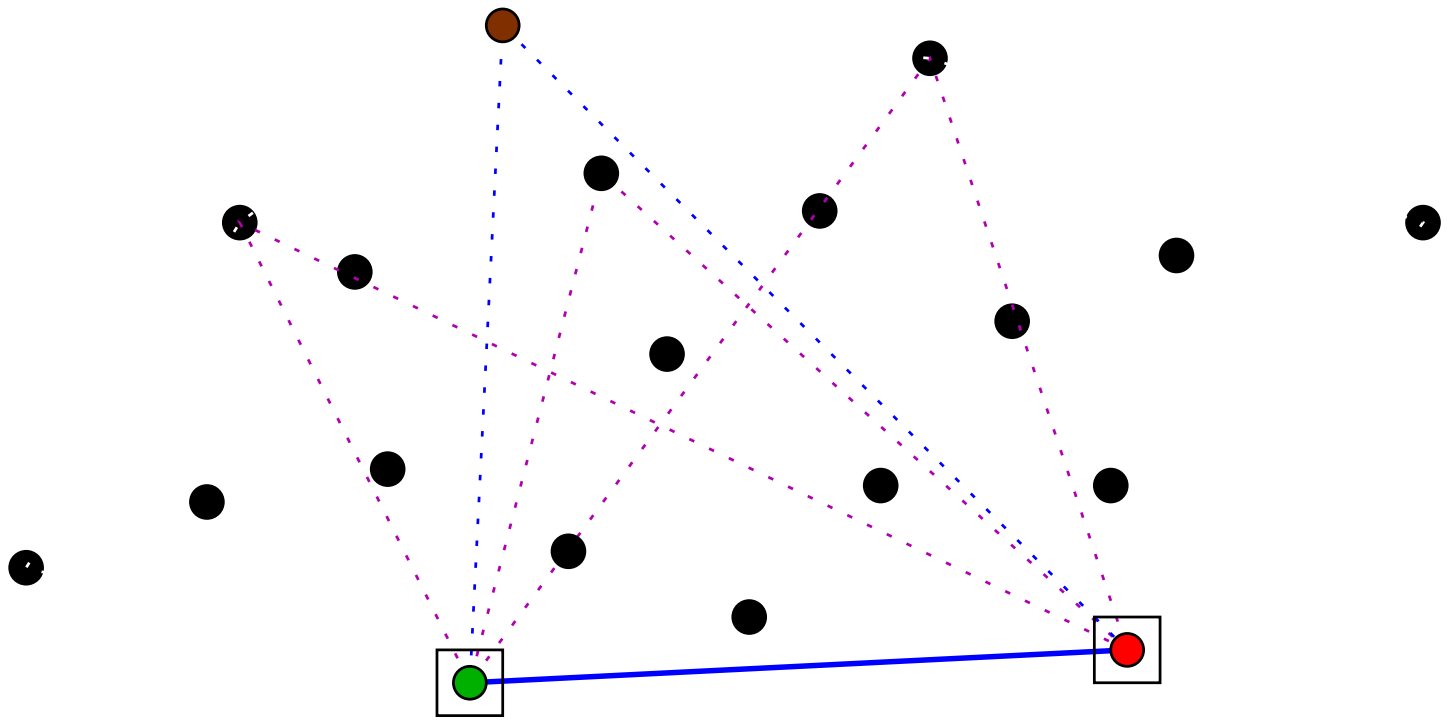
Como encontrar o ponto marrom?

O extremo marrom é o ponto mais distante da coleção em relação à reta que passa pelos dois extremos iniciais.



Como encontrar o ponto marrom?

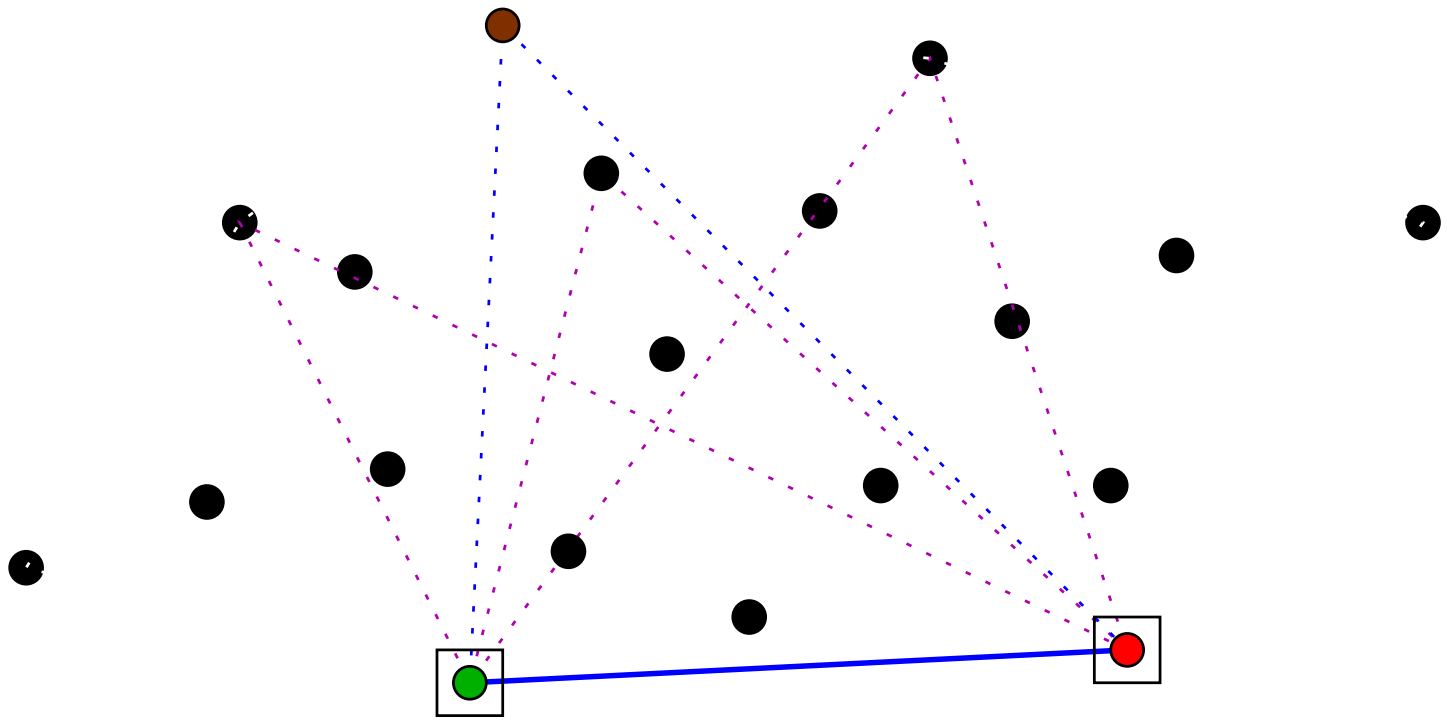
O extremo marrom é o ponto mais distante da coleção em relação à reta que passa pelos dois extremos iniciais.



Como são as áreas dos vários triângulos?
Qual tem a maior área?

Como encontrar o ponto marrom?

O extremo marrom é o ponto mais distante da coleção em relação à reta que passa pelos dois extremos iniciais.



Como são as áreas dos vários triângulos?
Qual tem a maior área?
O triângulo com terceira ponta mais distante!

O ponto marrom

$X[p\,..\,r], Y[p\,..\,r]$: coleção com ≥ 3 pontos em posição geral.

O ponto marrom

$X[p\,..\,r], Y[p\,..\,r]$: coleção com ≥ 3 pontos em posição geral.

A função abaixo devolve a área do triângulo cujos extremos são os pontos de índices i, j e k.

ÁREA (X, Y, i, j, k)
1 devolva $|\textsc{Det}(X[i], Y[i], X[j], Y[j], X[k], Y[k])|/2$

O ponto marrom

$X[p\,..\,r], Y[p\,..\,r]$: coleção com ≥ 3 pontos em posição geral.

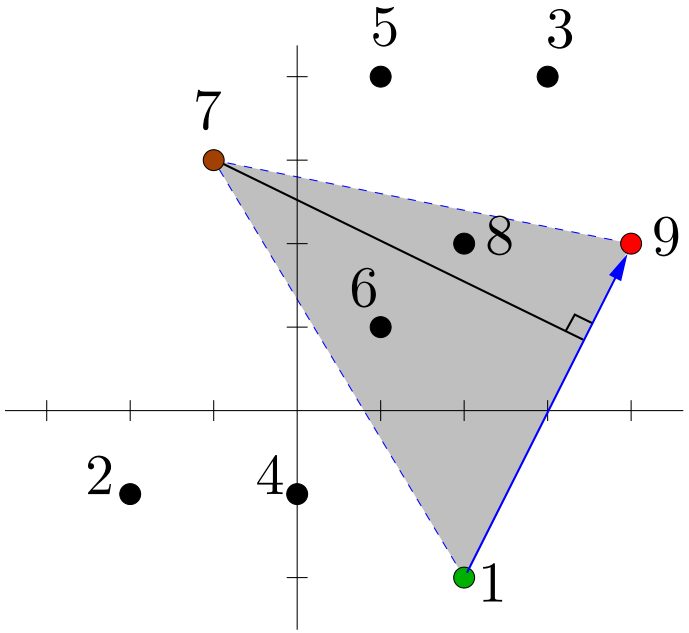
A função abaixo devolve a área do triângulo cujos extremos são os pontos de índices i, j e k.

ÁREA (X, Y, i, j, k)
1 devolva $|\text{DET}(X[i], Y[i], X[j], Y[j], X[k], Y[k])|/2$

Recebe $X[p\,..\,r], Y[p\,..\,r]$ e, usando ÁREA, devolve o índice de um ponto extremo da coleção distinto de p e r.

PONTOEXTREMO (X, Y, p, r)
1 $q \leftarrow p + 1$ $\quad maior \leftarrow$ ÁREA(X, Y, p, r, q)
2 para $i \leftarrow p + 2$ até $r - 1$ faça
3 se ÁREA$(X, Y, p, r, i) > maior$
4 então $q \leftarrow i$ $\quad maior \leftarrow$ ÁREA(X, Y, p, r, q)
5 devolva q

Exemplo

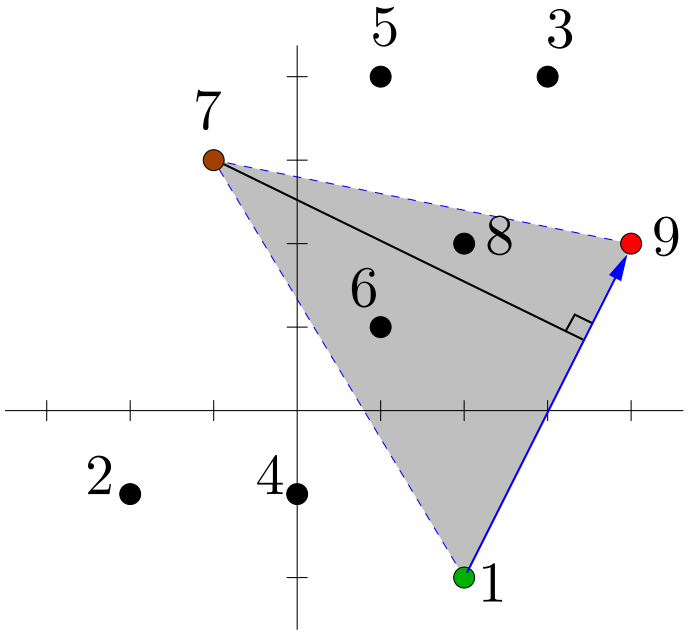


X	2	−2	3	0	1	1	−1	2	4
	1	2	3	4	5	6	7	8	9

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

$$\textsc{PontoExtremo}(X, Y, 1, 9) = 7$$

Exemplo

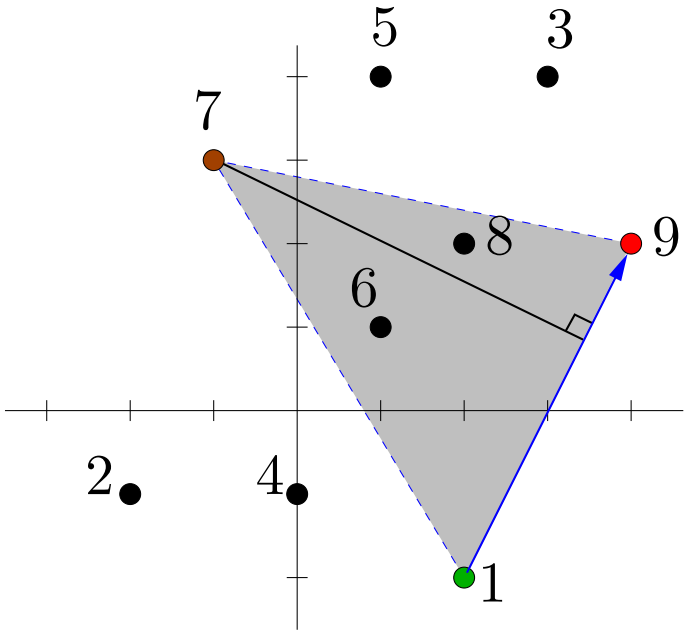


	1	2	3	4	5	6	7	8	9
X	2	−2	3	0	1	1	−1	2	4
Y	−2	−1	4	−1	4	1	3	2	2

$\textsc{PontoExtremo}(X, Y, 1, 9) = 7$

Exercício: Mostre que o algoritmo de fato devolve o índice de um ponto extremo da coleção $X[p\,..\,r], Y[p\,..\,r]$.

Exemplo



	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

PONTOEXTREMO$(X, Y, 1, 9) = 7$

Exercício: Mostre que o algoritmo de fato devolve o índice de um ponto extremo da coleção $X[p\,..\,r], Y[p\,..\,r]$.

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

Particionamento

PARTICIONE (X, Y, p, r):

Recebe coleção $X[p\,..\,r], Y[p\,..\,r]$ de pontos em posição geral, com pelo menos três pontos, tal que os pontos de índice p e r são extremos consecutivos na fronteira do fecho convexo da coleção no sentido anti-horário.

Particionamento

PARTICIONE (X, Y, p, r):

Recebe coleção $X[p\,..\,r], Y[p\,..\,r]$ de pontos em posição geral, com pelo menos três pontos, tal que os pontos de índice p e r são extremos consecutivos na fronteira do fecho convexo da coleção no sentido anti-horário.

Rearranja $X[p\,..\,r], Y[p\,..\,r]$ e devolve p', q tq $p \leq p' < q < r$ e

(i) o ponto de índice r permanece na mesma posição, enquanto que o ponto de índice p foi para a posição p',

(ii) o ponto de índice q é extremo,

(iii) $X[p\,..\,p'-1], Y[p\,..\,p'-1]$ é uma coleção de pontos interiores ao fecho convexo de $X[p\,..\,r], Y[p\,..\,r]$,

(iv) $\cdots$

Particionamento

PARTICIONE (X, Y, p, r):

Rearranja $X[p\,..\,r], Y[p\,..\,r]$ e devolve p', q tq $p \leq p' < q < r$ e

(i) o ponto de índice r permanece na mesma posição, enquanto que o ponto de índice p foi para a posição p',

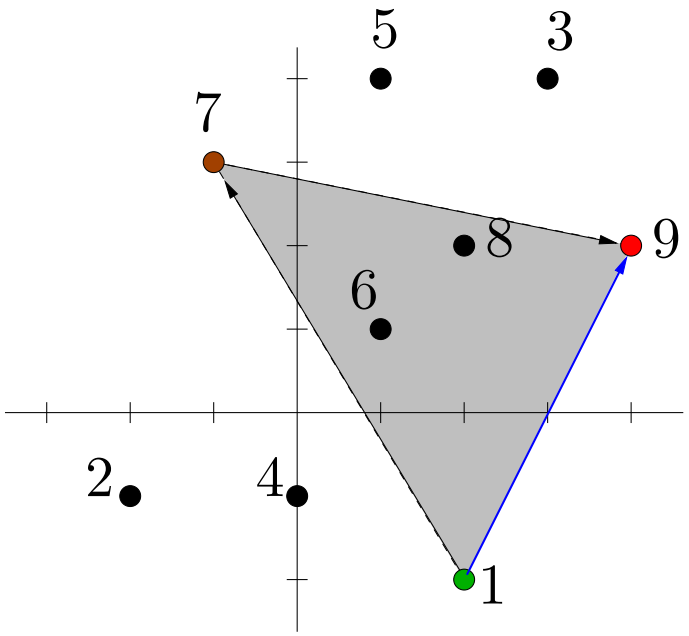
(ii) o ponto de índice q é extremo,

(iii) $X[p\,..\,p'-1], Y[p\,..\,p'-1]$ é uma coleção de pontos interiores ao fecho convexo de $X[p\,..\,r], Y[p\,..\,r]$,

(iv) $X[p'+1\,..\,q-1], Y[p'+1\,..\,q-1]$ é a coleção dos pontos que estão à esquerda da reta orientada determinada por $(X[p'], Y[p'])$ e $(X[q], Y[q])$,

(v) $X[q+1\,..\,r-1], Y[q+1\,..\,r-1]$ é a coleção dos pontos que estão à esquerda da reta orientada determinada por $(X[q], Y[q])$ e $(X[r], Y[r])$.

Particionamento

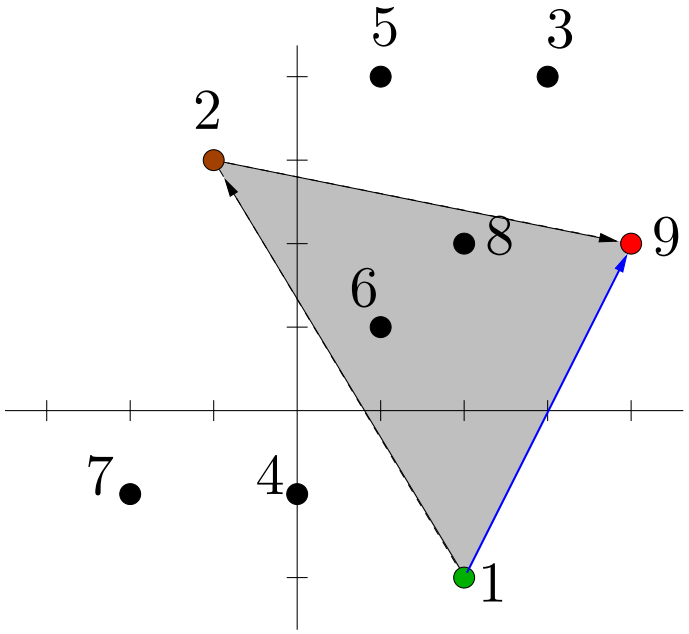


PARTICIONE$(X, Y, 1, 9)$

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

Encontra o ponto extremo indicado por q.

Particionamento



PARTICIONE$(X, Y, 1, 9)$

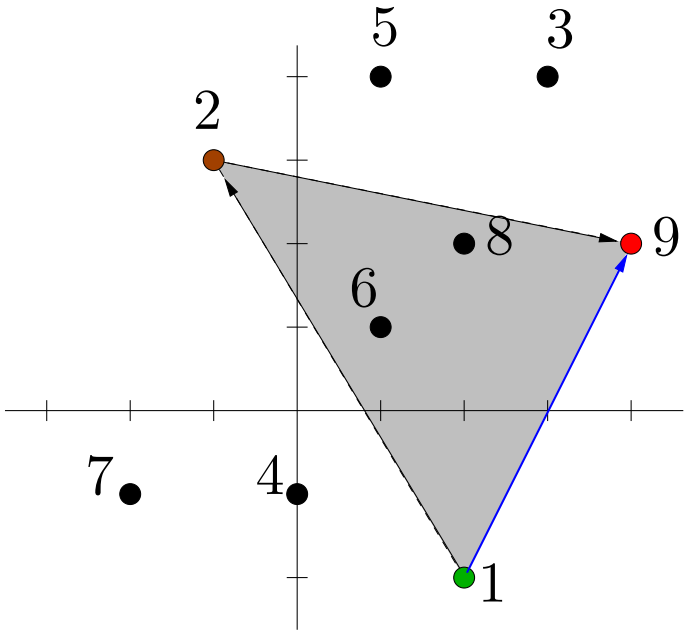
	p	$p{+}1$						k	r
X	2	-1	3	0	1	1	-2	2	4
Y	-2	3	4	-1	4	1	-1	2	2
	1	2	3	4	5	6	7	8	9

p'

q

Encontra o ponto extremo indicado por q.
Coloca esse ponto na posição $p{+}1$.

Particionamento



PARTICIONE$(X, Y, 1, 9)$

									p'
									q
	p	$p{+}1$						k	r
X	2	−1	3	0	1	1	−2	2	4
Y	−2	3	4	−1	4	1	−1	2	2
	1	2	3	4	5	6	7	8	9

Encontra o ponto extremo indicado por q.
Coloca esse ponto na posição $p{+}1$.

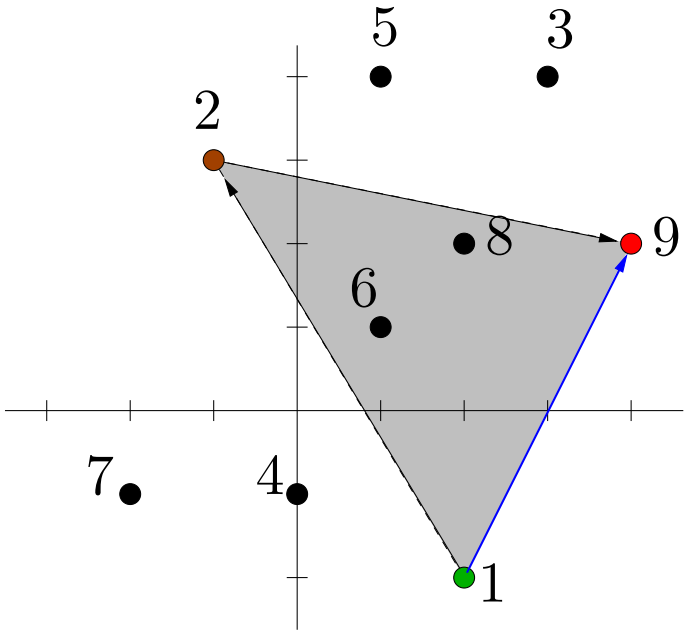
Invariante:

$X[q\,..\,r], Y[q\,..\,r]$: pontos vermelhos examinados

$X[p'\,..\,q{-}1], Y[p'\,..\,q{-}1]$: pontos verdes examinados

$X[k{+}1\,..\,p'{-}1], Y[k{+}1\,..\,p'{-}1]$: pontos interiores examinados

Particionamento



PARTICIONE$(X, Y, 1, 9)$

	p	$p{+}1$						k	r
X	2	-1	3	0	1	1	-2	2	4
Y	-2	3	4	-1	4	1	-1	2	2
	1	2	3	4	5	6	7	8	9

p'

q

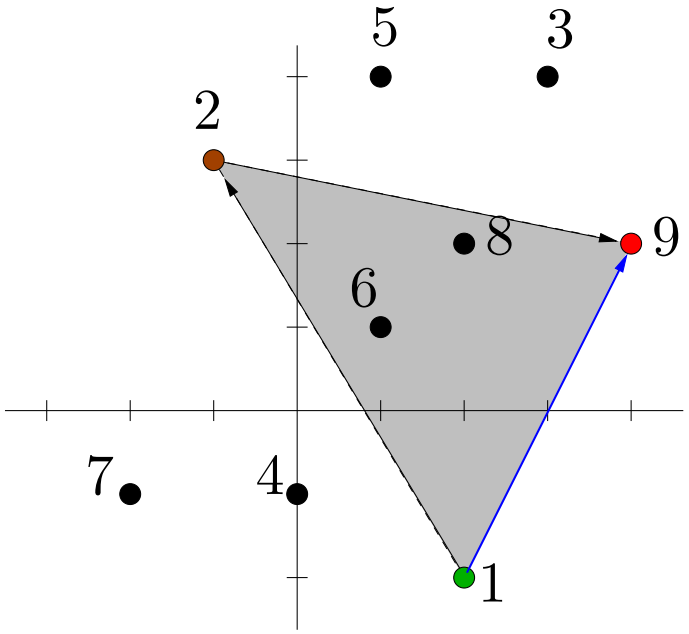
Invariante:

$X[q\,..\,r], Y[q\,..\,r]$: pontos vermelhos examinados

$X[p'\,..\,q{-}1], Y[p'\,..\,q{-}1]$: pontos verdes examinados

$X[k{+}1\,..\,p'{-}1], Y[k{+}1\,..\,p'{-}1]$: pontos interiores examinados

Particionamento



PARTICIONE$(X, Y, 1, 9)$

	p	$p{+}1$						k	r
X	2	-1	3	0	1	1	-2	2	4
Y	-2	3	4	-1	4	1	-1	2	2
	1	2	3	4	5	6	7	8	9

p' q

Invariante:

$X[q\,..\,r], Y[q\,..\,r]$: pontos vermelhos examinados

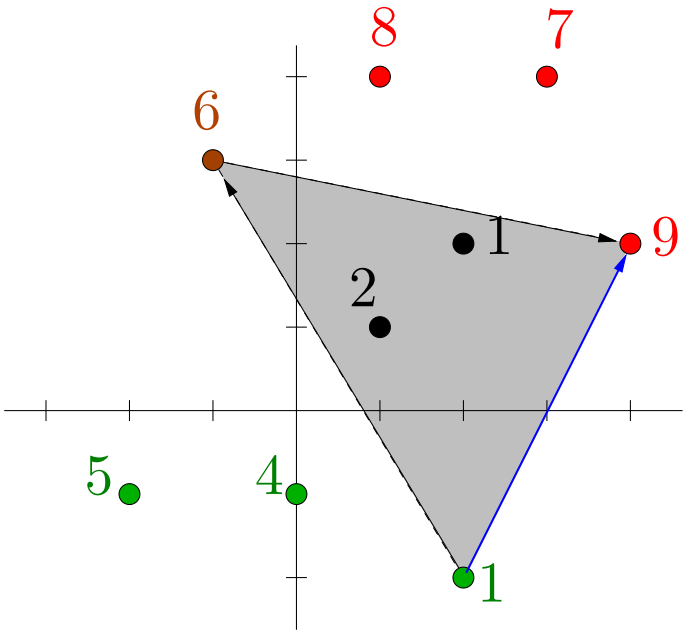
$X[p'\,..\,q{-}1], Y[p'\,..\,q{-}1]$: pontos verdes examinados

$X[k{+}1\,..\,p'{-}1], Y[k{+}1\,..\,p'{-}1]$: pontos interiores examinados

Para $k \leftarrow r{-}1$ até $p{+}1$

coloque o k-ésimo ponto na parte apropriada.

Particionamento



Ao final...

	p	$p{+}1$						k	r
X	2	−1	3	0	1	1	−2	2	4
Y	−2	3	4	−1	4	1	−1	2	2

	p		p'			q			r
X	2	1	2	0	−2	−1	3	1	4
Y	2	1	−2	−1	−1	3	4	4	2
	1	2	3	4	5	6	7	8	9

Invariante:

$X[q\,..\,r], Y[q\,..\,r]$: pontos vermelhos examinados

$X[p'\,..\,q{-}1], Y[p'\,..\,q{-}1]$: pontos verdes examinados

$X[k{+}1\,..\,p'{-}1], Y[k{+}1\,..\,p'{-}1]$: pontos interiores examinados

Particione

PARTICIONE (X, Y, p, r)

1 $q \leftarrow$ PONTOEXTREMO(X, Y, p, r)
2 $(X[p{+}1], Y[p{+}1]) \leftrightarrow (X[q], Y[q])$
3 $p' \leftarrow r \quad q \leftarrow r$
4 para $k \leftarrow r - 1$ decrescendo até $p + 2$ faça
5 se ESQ$(X, Y, p, p{+}1, k)$ ▷ verde?
6 então $p' \leftarrow p' - 1 \quad (X[p'], Y[p']) \leftrightarrow (X[k], Y[k])$
7 senão se ESQ$(X, Y, p{+}1, r, k)$ ▷ vermelho?
8 então $p' \leftarrow p' - 1 \quad q \leftarrow q - 1$
9 $(X[q], Y[q]) \leftrightarrow (X[k], Y[k])$
10 $(X[k], Y[k]) \leftrightarrow (X[p'], Y[p'])$
11 $p' \leftarrow p' - 1 \quad q \leftarrow q - 1$
12 $(X[q], Y[q]) \leftrightarrow (X[p{+}1], Y[p{+}1])$
13 se $p' \neq q$ então $(X[p'], Y[p']) \leftrightarrow (X[p{+}1], Y[p{+}1])$
14 $p' \leftarrow p' - 1 \quad (X[p'], Y[p']) \leftrightarrow (X[p], Y[p])$
15 devolva (p', q)

Particione

```
PARTICIONE (X, Y, p, r)
 1  q ← PONTOEXTREMO(X, Y, p, r)
 2  (X[p+1], Y[p+1]) ↔ (X[q], Y[q])
 3  p' ← r    q ← r
 4  para k ← r − 1 decrescendo até p + 2 faça
 5      se ESQ(X, Y, p, p+1, k)          ▷ verde?
 6          então p' ← p' − 1    (X[p'], Y[p']) ↔ (X[k], Y[k])
 7          senão se ESQ(X, Y, p+1, r, k)          ▷ vermelho?
 8                    então p' ← p' − 1    q ← q − 1
 9                          (X[q], Y[q]) ↔ (X[k], Y[k])
10                          (X[k], Y[k]) ↔ (X[p'], Y[p'])
11  p' ← p' − 1    q ← q − 1
12  (X[q], Y[q]) ↔ (X[p+1], Y[p+1])
13  se p' ≠ q então (X[p'], Y[p']) ↔ (X[p+1], Y[p+1])
14  p' ← p' − 1    (X[p'], Y[p']) ↔ (X[p], Y[p])
15  devolva (p', q)
```

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

Quickhull

```
QUICKHULL (X, Y, n)
 1  se n = 1
 2      então h ← 1    H[1] ← 1
 3      senão k ← min{i ∈ [1..n] : Y[i] ≤ Y[j], 1 ≤ j ≤ n}
 4            (X[1], Y[1]) ↔ (X[k], Y[k])
 5            i ← 2
 6            para j ← 3 até n faça
 7                se DIR(X, Y, 1, i, j) então i ← j
 8            (X[n], Y[n]) ↔ (X[i], Y[i])
 9            (H, h) ← QUICKHULLREC(X, Y, 1, n)
10  devolva (H, h)
```

Quickhull

```
QUICKHULL (X, Y, n)
 1  se n = 1
 2      então h ← 1    H[1] ← 1
 3      senão k ← min{i ∈ [1..n] : Y[i] ≤ Y[j], 1 ≤ j ≤ n}
 4            (X[1], Y[1]) ↔ (X[k], Y[k])
 5            i ← 2
 6            para j ← 3 até n faça
 7                se DIR(X, Y, 1, i, j) então i ← j
 8            (X[n], Y[n]) ↔ (X[i], Y[i])
 9            (H, h) ← QUICKHULLREC(X, Y, 1, n)
10  devolva (H, h)
```

Consumo de tempo: $\Theta(n) + T(n)$,
onde $n = r - p + 1$ e $T(n)$ é o tempo consumido por QUICKHULLREC$(X, Y, 1, n)$.

Miolo recursivo do Quickhull

```
QUICKHULLREC (X, Y, p, r)
 1  se p = r − 1      ▷ há exatamente dois pontos na coleção
 2       então h ← 2     H[1] ← r     H[2] ← p
 3       senão (p′, q) ← PARTICIONE(X, Y, p, r)
 4             (H, h) ← QUICKHULLREC(X, Y, q, r)
 5             (H′, h′) ← QUICKHULLREC(X, Y, p′, q)
               ▷ H ← H · H′ removendo uma cópia do q
 6             para i ← 2 até h′ faça
 7                  h ← h + 1     H[h] ← H′[i]
 8  devolva (H, h)
```

Miolo recursivo do Quickhull

```
QUICKHULLREC (X, Y, p, r)
 1  se p = r − 1      ▷ há exatamente dois pontos na coleção
 2       então h ← 2     H[1] ← r     H[2] ← p
 3       senão (p′, q) ← PARTICIONE(X, Y, p, r)
 4             (H, h) ← QUICKHULLREC(X, Y, q, r)
 5             (H′, h′) ← QUICKHULLREC(X, Y, p′, q)
               ▷ H ← H · H′ removendo uma cópia do q
 6             para i ← 2 até h′ faça
 7                  h ← h + 1     H[h] ← H′[i]
 8  devolva (H, h)
```

Consumo de tempo: $T(n) = T(n_1) + T(n_2) + \Theta(n)$,
onde $n = r - p + 1$, $n_d = r - q + 1$ e $n_e = r - p' + 1$.

Miolo recursivo do Quickhull

```
QUICKHULLREC (X, Y, p, r)
 1  se p = r − 1      ▷ há exatamente dois pontos na coleção
 2       então h ← 2     H[1] ← r     H[2] ← p
 3       senão (p′, q) ← PARTICIONE(X, Y, p, r)
 4             (H, h) ← QUICKHULLREC(X, Y, q, r)
 5             (H′, h′) ← QUICKHULLREC(X, Y, p′, q)
               ▷ H ← H · H′ removendo uma cópia do q
 6             para i ← 2 até h′ faça
 7                  h ← h + 1     H[h] ← H′[i]
 8  devolva (H, h)
```

Consumo de tempo: $T(n) = T(n_1) + T(n_2) + \Theta(n)$,
onde $n = r - p + 1$, $n_d = r - q + 1$ e $n_e = r - p' + 1$.

Observe que $n_d + n_e \leq n$.
Com isso, podemos mostrar que $T(n) = \mathrm{O}(n^2)$.

Casos degenerados

Como tratar os casos degenerados nos quatro algoritmos vistos para fecho convexo?

Casos degenerados

Como tratar os casos degenerados nos quatro algoritmos vistos para fecho convexo?

- embrulho de presente
- Graham
- incremental
- quickhull

Fecho convexo: um resumo

Algoritmo	Consumo no pior caso
EMBRULHO	$\mathrm{O}(nh)$
INCREMENTAL	$\mathrm{O}(n^2)$
GRAHAM	$\mathrm{O}(n \lg n)$
QUICKHULL	$\mathrm{O}(nh)$
MERGEHULL	$\mathrm{O}(n \lg n)$

Fecho convexo: um resumo

Algoritmo	Consumo no pior caso
EMBRULHO	$\mathrm{O}(nh)$
INCREMENTAL	$\mathrm{O}(n^2)$
GRAHAM	$\mathrm{O}(n \lg n)$
QUICKHULL	$\mathrm{O}(nh)$
MERGEHULL	$\mathrm{O}(n \lg n)$

Cota inferior para o fecho convexo:
não existe algoritmo para encontrar o fecho convexo que no pior caso consuma $\mathrm{o}(n \lg h)$.

Fecho convexo: um resumo

Algoritmo	Consumo no pior caso
EMBRULHO	$\mathrm{O}(nh)$
INCREMENTAL	$\mathrm{O}(n^2)$
GRAHAM	$\mathrm{O}(n \lg n)$
QUICKHULL	$\mathrm{O}(nh)$
MERGEHULL	$\mathrm{O}(n \lg n)$

Cota inferior para o fecho convexo:
não existe algoritmo para encontrar o fecho convexo que no pior caso consuma $\mathrm{o}(n \lg h)$.

Na aula vimos uma redução do problema da ordenação para o fecho convexo no plano que implica um resultado um pouco mais fraco que este.