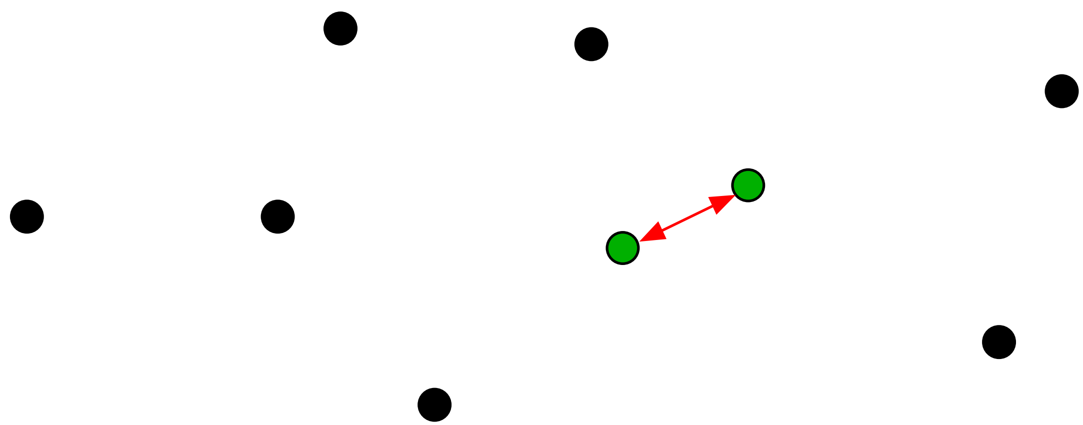
Par de pontos mais próximos

Problema: Dados n pontos no plano,
determinar dois deles que estão à distância mínima.

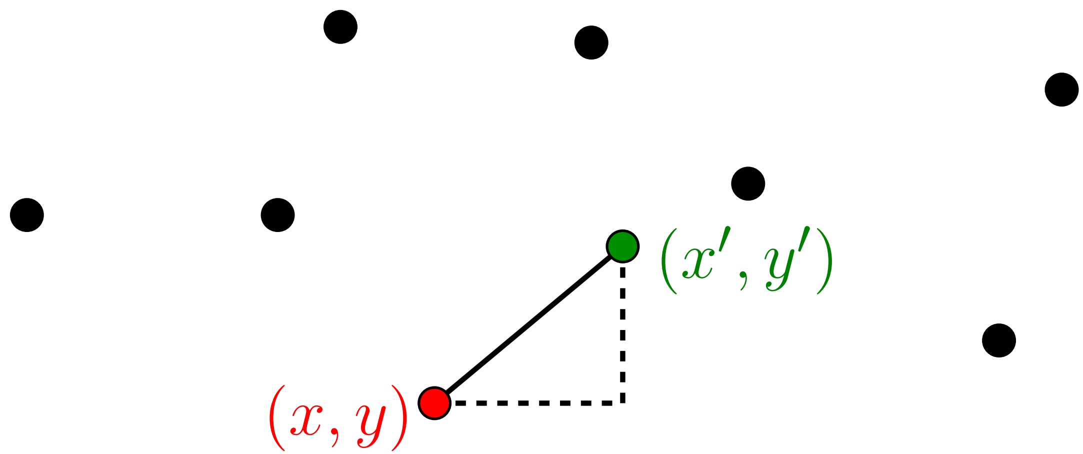
Par de pontos mais próximos

Problema: Dados n pontos no plano,
determinar dois deles que estão à distância mínima.



Par de pontos mais próximos

Problema: Dados n pontos no plano,
determinar dois deles que estão à distância mínima.



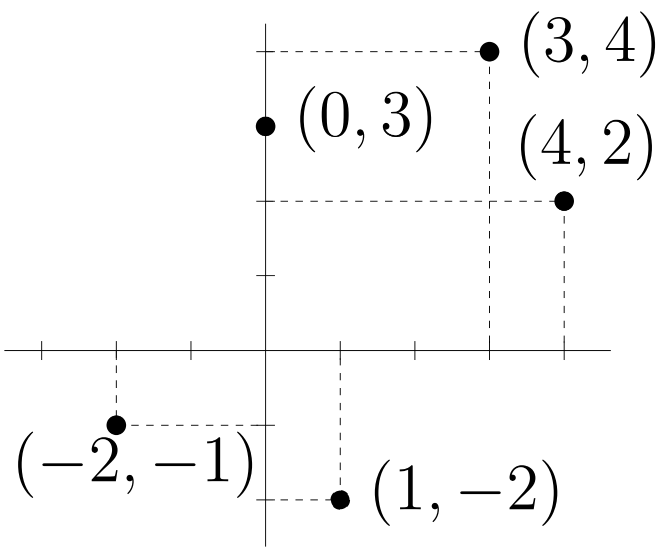
Lembre-se que, para dois pontos (x, y) e (x', y') no plano,

$$\textsc{Dist}(x, y, x', y') = \sqrt{(x - x')^2 + (y - y')^2}.$$

Par de pontos mais próximos

Problema: Dados n pontos no plano,
determinar dois deles que estão à distância mínima.

Entrada: coleção de n pontos representada por vetores $X[1\,..\,n]$ e $Y[1\,..\,n]$.

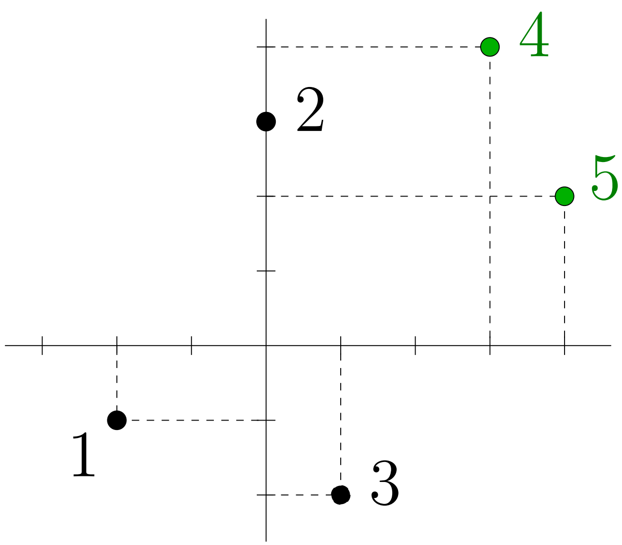


X	-2	0	1	3	4
Y	-1	3	-2	4	2
	1	2	3	4	5

Par de pontos mais próximos

Problema: Dados n pontos no plano,
determinar dois deles que estão à distância mínima.

Entrada: coleção de n pontos representada por
vetores $X[1\,..\,n]$ e $Y[1\,..\,n]$.



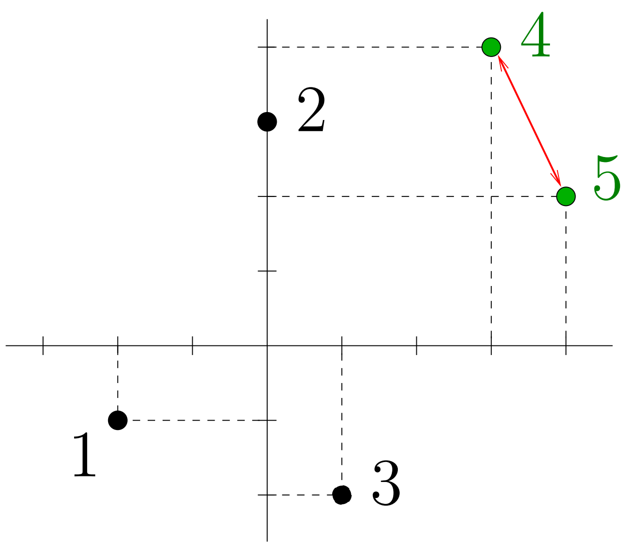
	1	2	3	4	5
X	-2	0	1	3	4
Y	-1	3	-2	4	2

Saída: índices i e j indicando
dois pontos à distância mínima.

Par de pontos mais próximos

Problema: Dados n pontos no plano, determinar dois deles que estão à distância mínima.

Entrada: coleção de n pontos representada por vetores $X[1\,..\,n]$ e $Y[1\,..\,n]$.



X	-2	0	1	3	4
Y	-1	3	-2	4	2
	1	2	3	4	5

Saída: menor distância entre dois pontos da coleção.

Par de pontos mais próximos

Problema: Dados n pontos no plano,
determinar dois deles que estão à distância mínima.

Entrada: vetores $X[1\,..\,n]$ e $Y[1\,..\,n]$
Saída: menor distância entre dois pontos da coleção

Par de pontos mais próximos

Problema: Dados n pontos no plano,
determinar dois deles que estão à distância mínima.

Entrada: vetores $X[1\,..\,n]$ e $Y[1\,..\,n]$
Saída: menor distância entre dois pontos da coleção

Primeira solução: algoritmo quadrático,
que testa todos os pares de pontos.

Par de pontos mais próximos

Problema: Dados n pontos no plano, determinar dois deles que estão à distância mínima.

Entrada: vetores $X[1\,.\,.\,n]$ e $Y[1\,.\,.\,n]$
Saída: menor distância entre dois pontos da coleção

Primeira solução: algoritmo quadrático, que testa todos os pares de pontos.

ELEMENTAR(X, Y, n)
$d \leftarrow +\infty$
para $i \leftarrow 2$ até n faça
    para $j \leftarrow 1$ até $i - 1$ faça
        se DIST$(X[i], Y[i], X[j], Y[j]) < d$
            então $d \leftarrow$ DIST$(X[i], Y[i], X[j], Y[j])$
devolva d

Algoritmo elementar

ELEMENTAR(X, Y, n)
$d \leftarrow +\infty$
para $i \leftarrow 2$ até n faça
    para $j \leftarrow 1$ até $i - 1$ faça
        se DIST$(X[i], Y[i], X[j], Y[j]) < d$
            então $d \leftarrow$ DIST$(X[i], Y[i], X[j], Y[j])$
devolva d

Algoritmo elementar

ELEMENTAR(X, Y, n)

$d \leftarrow +\infty$
para $i \leftarrow 2$ até n faça
    para $j \leftarrow 1$ até $i - 1$ faça
        se DIST$(X[i], Y[i], X[j], Y[j]) < d$
            então $d \leftarrow$ DIST$(X[i], Y[i], X[j], Y[j])$
devolva d

Invariante: d é a menor distância entre os pontos da coleção $X[1\,..\,i-1], Y[1\,..\,i-1]$.

Algoritmo elementar

ELEMENTAR(X, Y, n)
1 $d \leftarrow +\infty$
2 para $i \leftarrow 2$ até n faça
3 para $j \leftarrow 1$ até $i-1$ faça
4 se DIST$(X[i], Y[i], X[j], Y[j]) < d$
5 então $d \leftarrow$ DIST$(X[i], Y[i], X[j], Y[j])$
6 devolva d

Invariante: d é a menor distância entre os pontos da coleção $X[1 \mathinner{.\,.} i-1], Y[1 \mathinner{.\,.} i-1]$.

Consumo de tempo: linha 4 é executada

$$\sum_{i=2}^{n}(i-1) \;=\; \sum_{i=1}^{n-1} i \;=\; \frac{n(n-1)}{2} \;=\; \Theta(n^2).$$

Algoritmo elementar

ELEMENTAR(X, Y, n)

1 $d \leftarrow +\infty$
2 para $i \leftarrow 2$ até n faça
3   para $j \leftarrow 1$ até $i-1$ faça
4   se DIST$(X[i], Y[i], X[j], Y[j]) < d$
5   então $d \leftarrow$ DIST$(X[i], Y[i], X[j], Y[j])$
6 devolva d

Invariante: d é a menor distância entre os pontos da coleção $X[1\,..\,i-1], Y[1\,..\,i-1]$.

Consumo de tempo: linha 4 é executada

$$\sum_{i=2}^{n}(i-1) \;=\; \sum_{i=1}^{n-1} i \;=\; \frac{n(n-1)}{2} \;=\; \Theta(n^2).$$

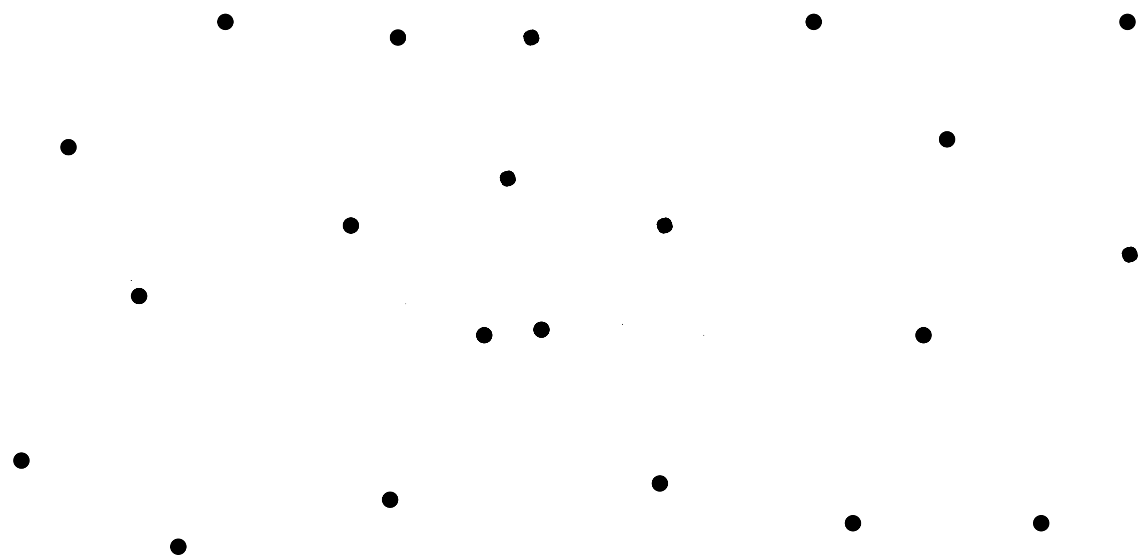
É possível projetar um algoritmo mais eficiente que este?

Par mais próximo no plano

Usemos divisão e conquista!

Par mais próximo no plano

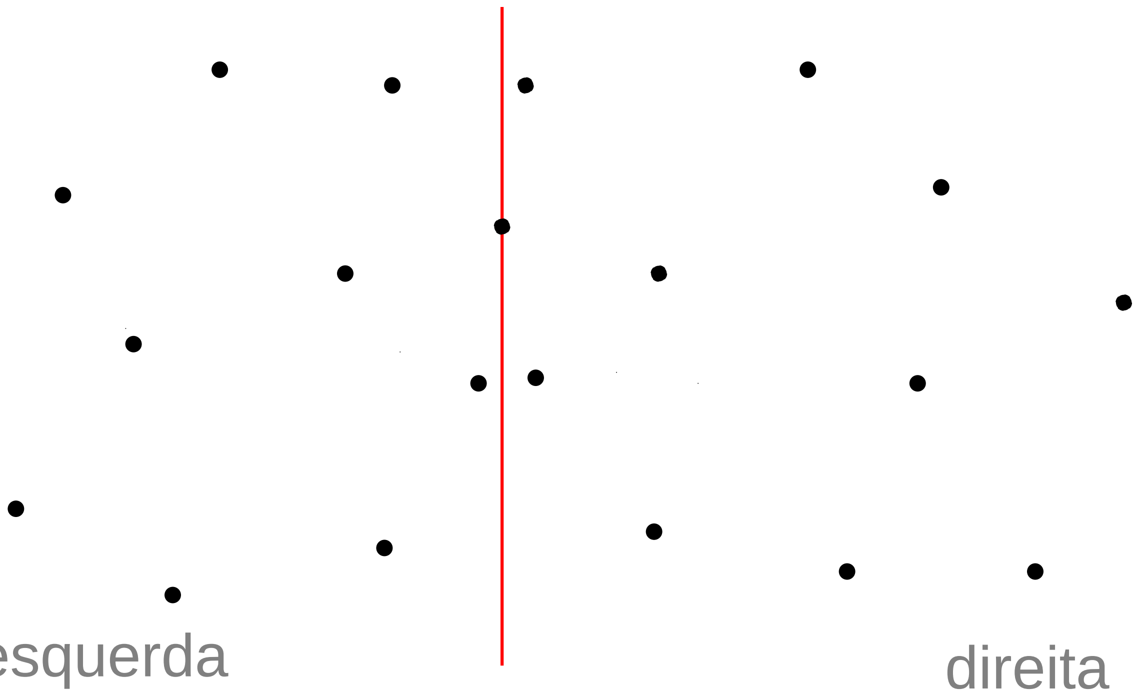
Usemos divisão e conquista!



Par mais próximo no plano

Usemos divisão e conquista!

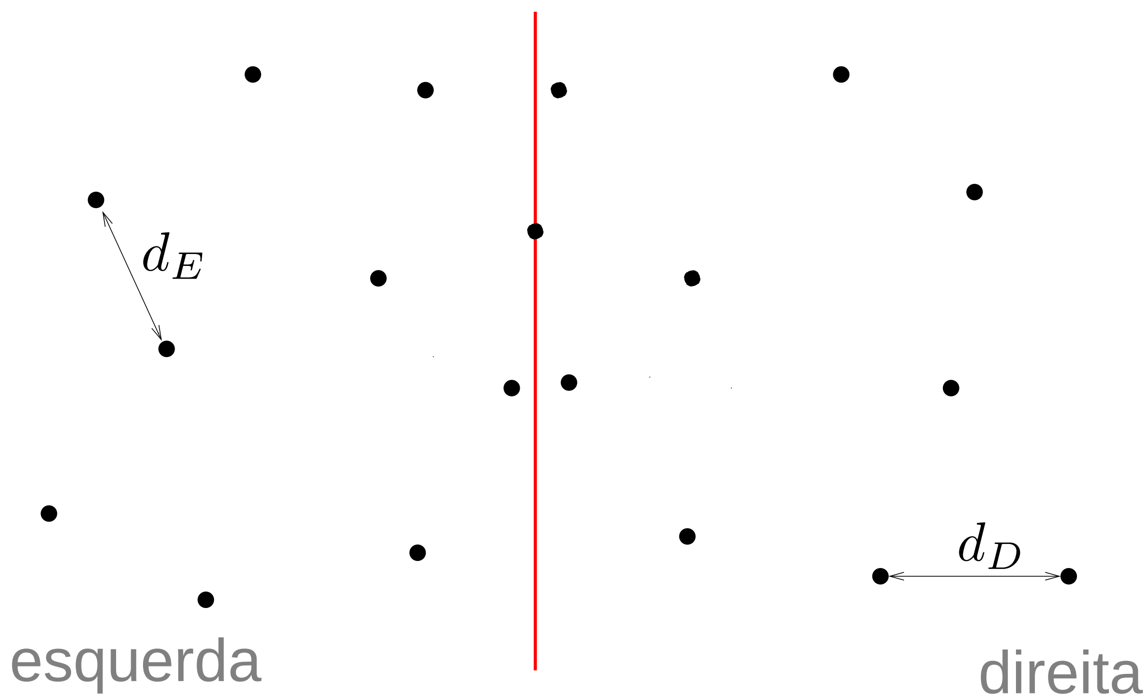
Divide...



Par mais próximo no plano

Usemos divisão e conquista!

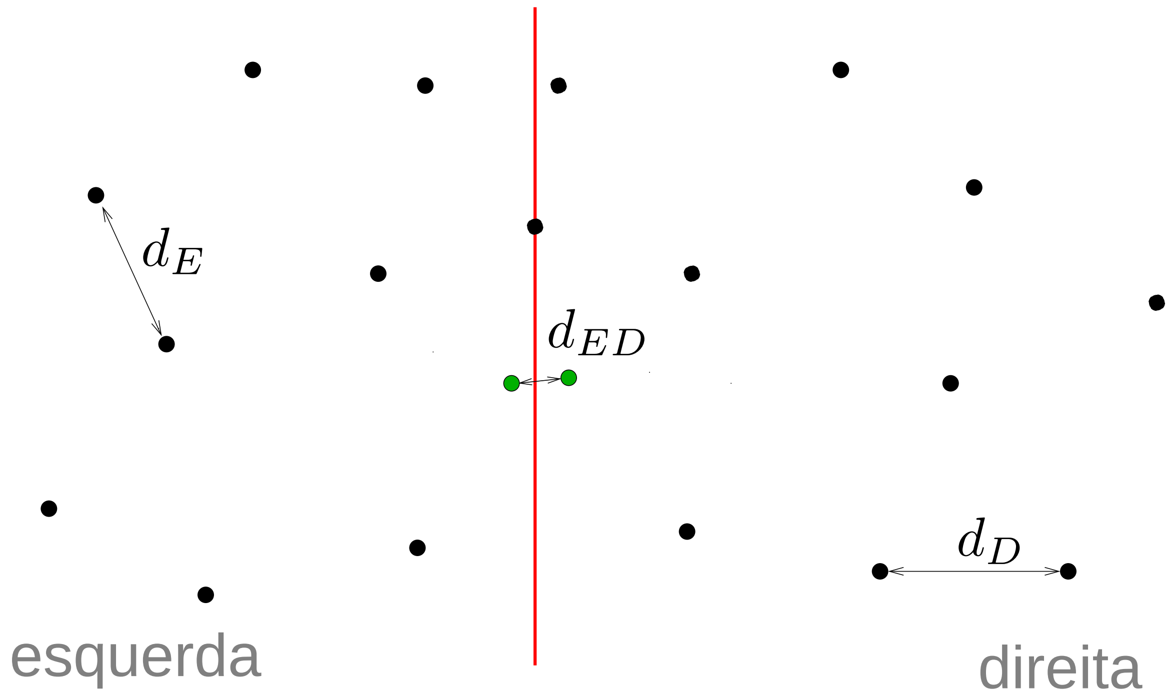
Divide... Conquista...



Par mais próximo no plano

Usemos divisão e conquista!

Divide... Conquista... Combina...



Algoritmo de Shamos e Hoey

Pré-processamento: ordenar os pontos pela X-coordenada

Algoritmo de Shamos e Hoey

Pré-processamento: ordenar os pontos pela X-coordenada

DISTÂNCIA-SH(X, Y, n)
1 MERGESORT$(X, Y, 1, n)$
2 devolva DISTÂNCIAREC-SH $(X, Y, 1, n)$

Algoritmo de Shamos e Hoey

Pré-processamento: ordenar os pontos pela X-coordenada

DISTÂNCIA-SH(X, Y, n)
1 MERGESORT$(X, Y, 1, n)$
2 devolva DISTÂNCIAREC-SH $(X, Y, 1, n)$

Consumo de tempo:
$\mathrm{O}(n \lg n)$ mais o tempo do DISTÂNCIAREC-SH.

Divisão e conquista

DISTÂNCIAREC-SH (X, Y, p, r)

Dividir: $X[p \,..\, q], Y[p \,..\, q]$ (esquerda)
$X[q{+}1 \,..\, r], Y[q{+}1 \,..\, r]$ (direita)
onde $q := \lfloor (p+r)/2 \rfloor$.

Divisão e conquista

DISTÂNCIAREC-SH (X, Y, p, r)

Dividir: $X[p \,..\, q], Y[p \,..\, q]$ (esquerda)
$X[q{+}1 \,..\, r], Y[q{+}1 \,..\, r]$ (direita)
onde $q := \lfloor (p + r)/2 \rfloor$.

Conquistar: Determine, recursivamente, a menor distância d_E entre dois pontos da esquerda e a menor distância d_D entre dois pontos da direita.

Divisão e conquista

DISTÂNCIAREC-SH (X, Y, p, r)

Dividir: $X[p \mathinner{..} q], Y[p \mathinner{..} q]$ (esquerda)
$X[q{+}1 \mathinner{..} r], Y[q{+}1 \mathinner{..} r]$ (direita)
onde $q := \lfloor (p+r)/2 \rfloor$.

Conquistar: Determine, recursivamente, a menor distância d_E entre dois pontos da esquerda e a menor distância d_D entre dois pontos da direita.

Combinar: Devolva o mínimo entre d_E, d_D e a menor distância d_{ED} entre um ponto da esquerda e um ponto da direita.

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, p, r) ▷ Divisão e conquista

se $r \leq p + 2$
    então ▷ resolva o problema diretamente
    senão $q \leftarrow \lfloor (p + r)/2 \rfloor$
        $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, p, q)
        $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, q{+}1, r)$
        devolva COMBINE (X, Y, p, r, d_E, d_D)

Algoritmo de Shamos e Hoey

DistânciaRec-SH (X, Y, p, r) $\triangleright$ Divisão e conquista

```
se r ≤ p + 2
    então ▷ resolva o problema diretamente
    senão q ← ⌊(p + r)/2⌋
          d_E ← DistânciaRec-SH (X, Y, p, q)
          d_D ← DistânciaRec-SH (X, Y, q+1, r)
          devolva Combine (X, Y, p, r, d_E, d_D)
```

Suponha que Combine é linear.

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, p, r) ▷ Divisão e conquista

se $r \leq p + 2$
  então ▷ resolva o problema diretamente
  senão $q \leftarrow \lfloor (p + r)/2 \rfloor$
    $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, p, q)
    $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, q{+}1, r)$
    devolva COMBINE (X, Y, p, r, d_E, d_D)

Suponha que COMBINE é linear.

Consumo de tempo do DISTÂNCIAREC-SH:

$$T(n) \;=\; T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

onde $n = r - p + 1$.

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, p, r) ▷ Divisão e conquista

se $r \leq p + 2$

  então ▷ resolva o problema diretamente

  senão $q \leftarrow \lfloor (p + r)/2 \rfloor$

    $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, p, q)

    $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, q{+}1, r)$

    devolva COMBINE (X, Y, p, r, d_E, d_D)

Suponha que COMBINE é linear.

Consumo de tempo:

$$T(n) \;=\; T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

onde $n = r - p + 1$. Quanto vale $T(n)$?

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, p, r) $\quad \triangleright$ Divisão e conquista

se $r \leq p + 2$
    então $\triangleright$ resolva o problema diretamente
    senão $q \leftarrow \lfloor (p+r)/2 \rfloor$
        $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, p, q)
        $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, q{+}1, r)$
        devolva COMBINE (X, Y, p, r, d_E, d_D)

Suponha que COMBINE é linear.

Consumo de tempo:

$$T(n) \;=\; T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

onde $n = r - p + 1$. Quanto vale $T(n)$? $T(n) = \mathrm{O}(n \lg n)$.

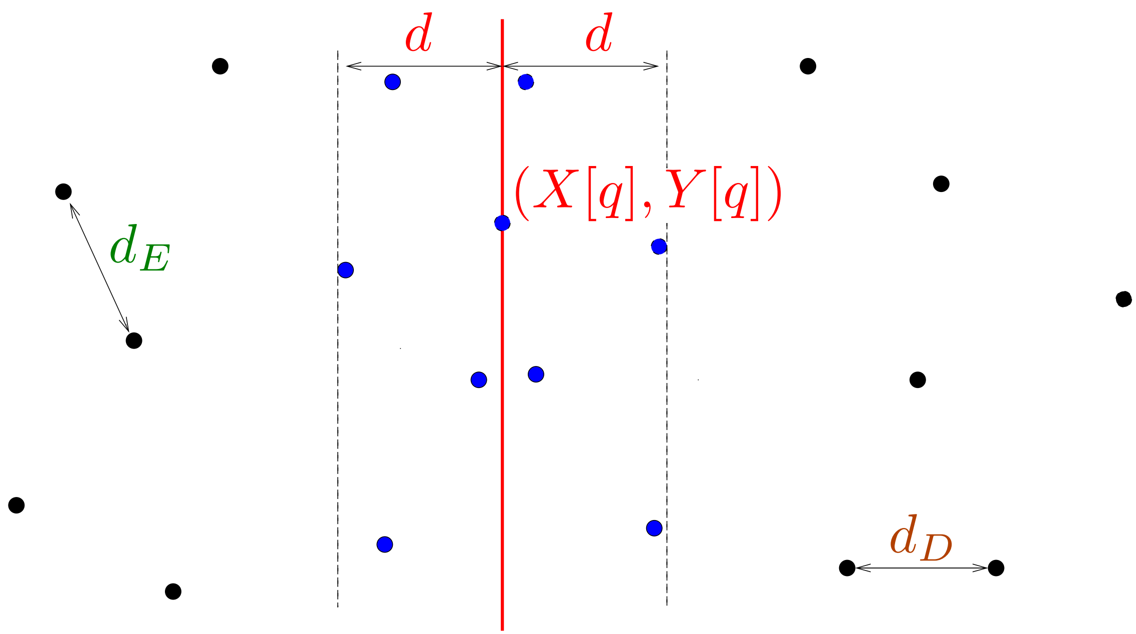
Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

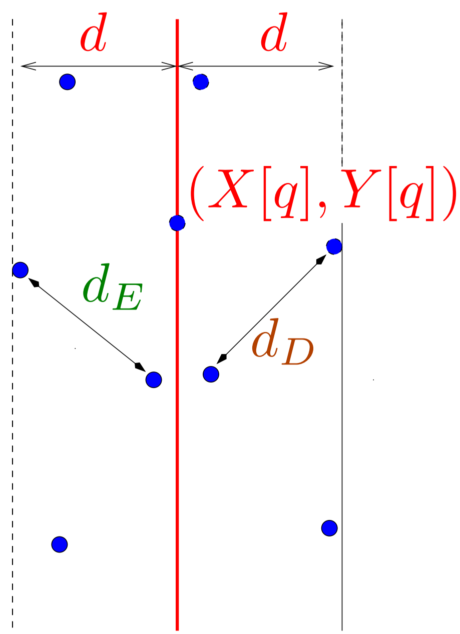
COMBINE precisa considerar apenas pontos que estão a uma distância menor que $d = \min\{d_E, d_D\}$ da reta vertical $x = X[q]$.



Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

COMBINE precisa considerar apenas pontos que estão a uma distância menor que $d = \min\{d_E, d_D\}$ da reta vertical $x = X[q]$.



Infelizmente todos os pontos podem estar nesta faixa...

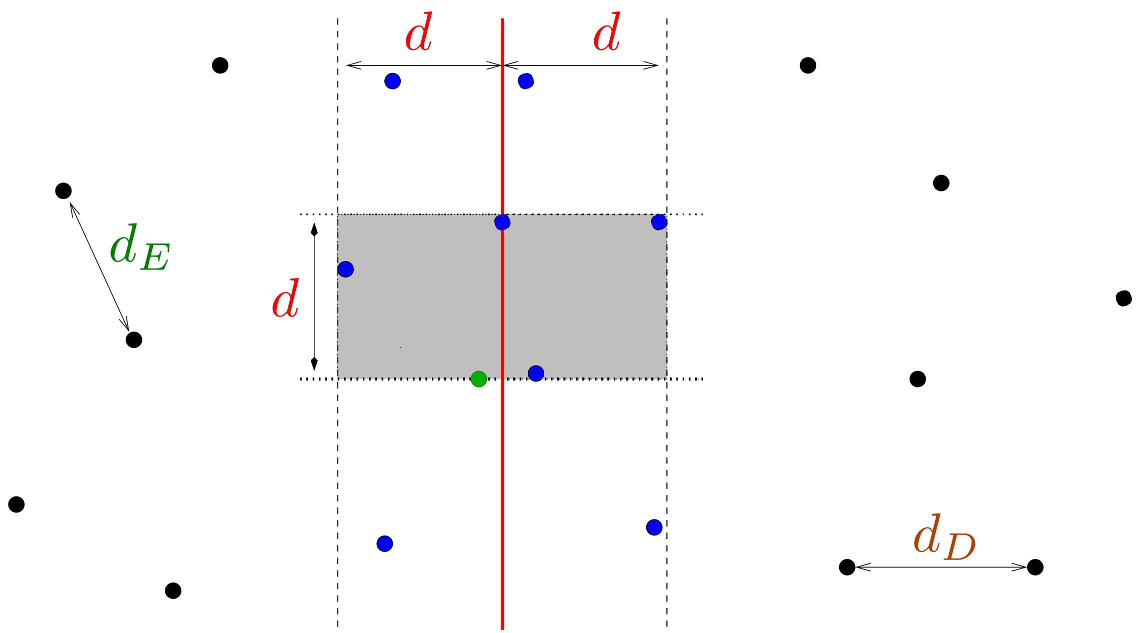
Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

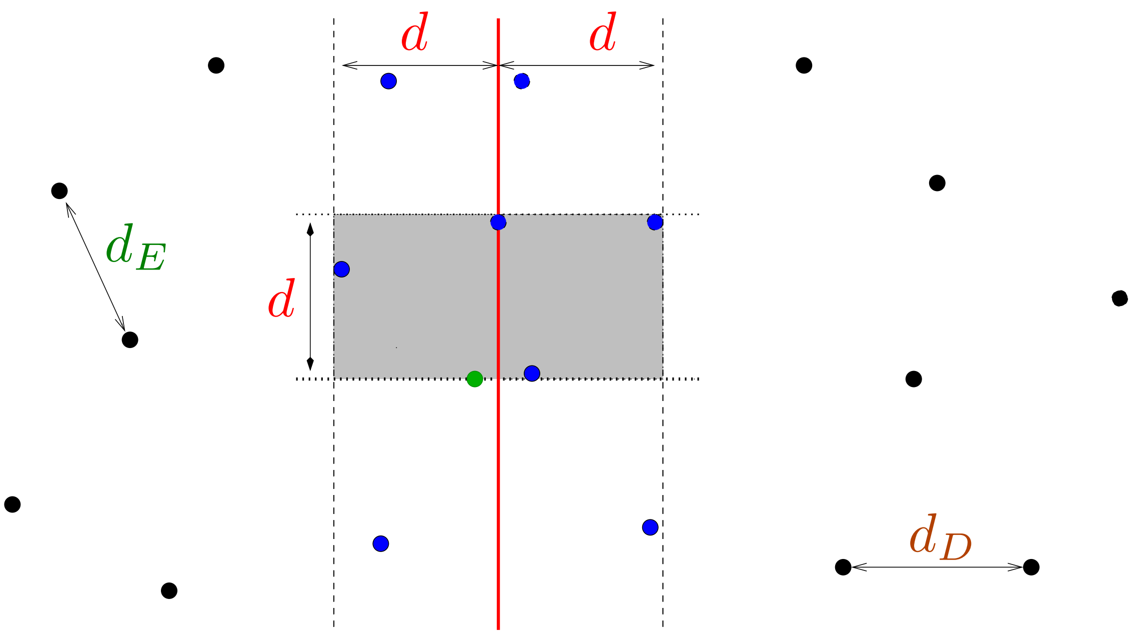
Ideia...



Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

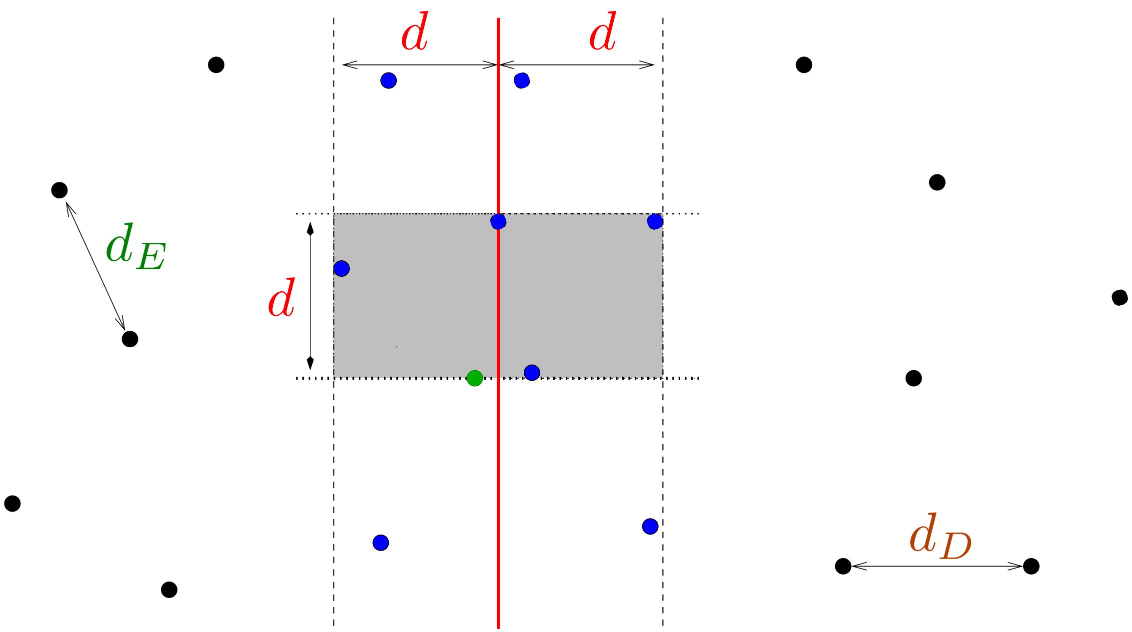
Ideia...



Para cada ponto na faixa, olhamos apenas para pontos da faixa que tenham Y-coordenada no máximo d mais que este ponto.

Algoritmo de Shamos e Hoey

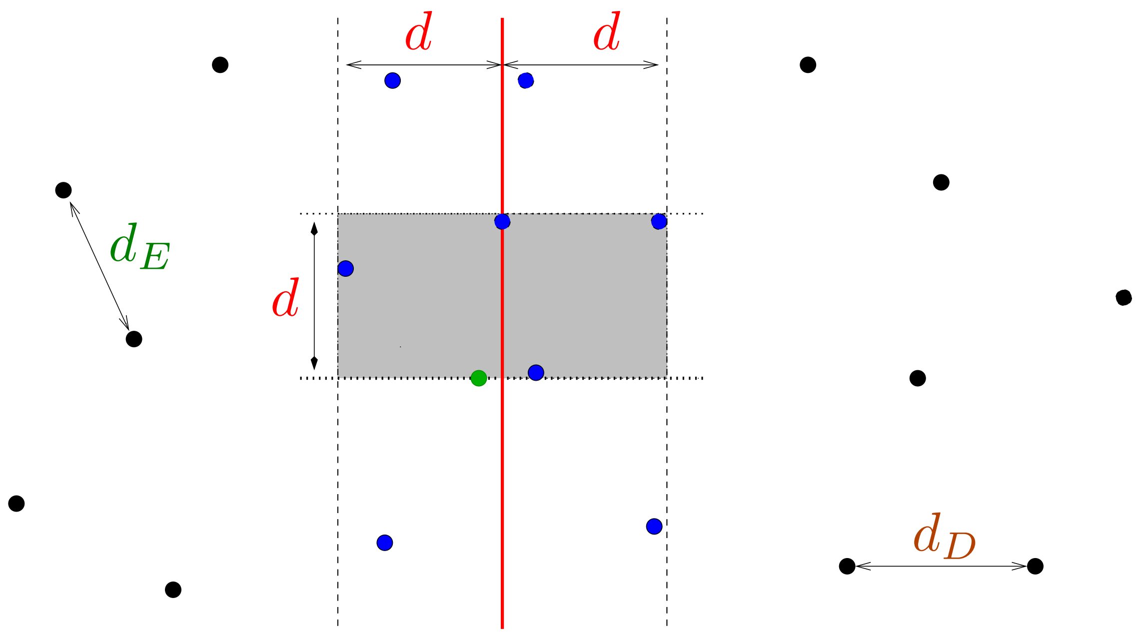
Como fazer o COMBINE linear?



Quantos pontos assim há?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

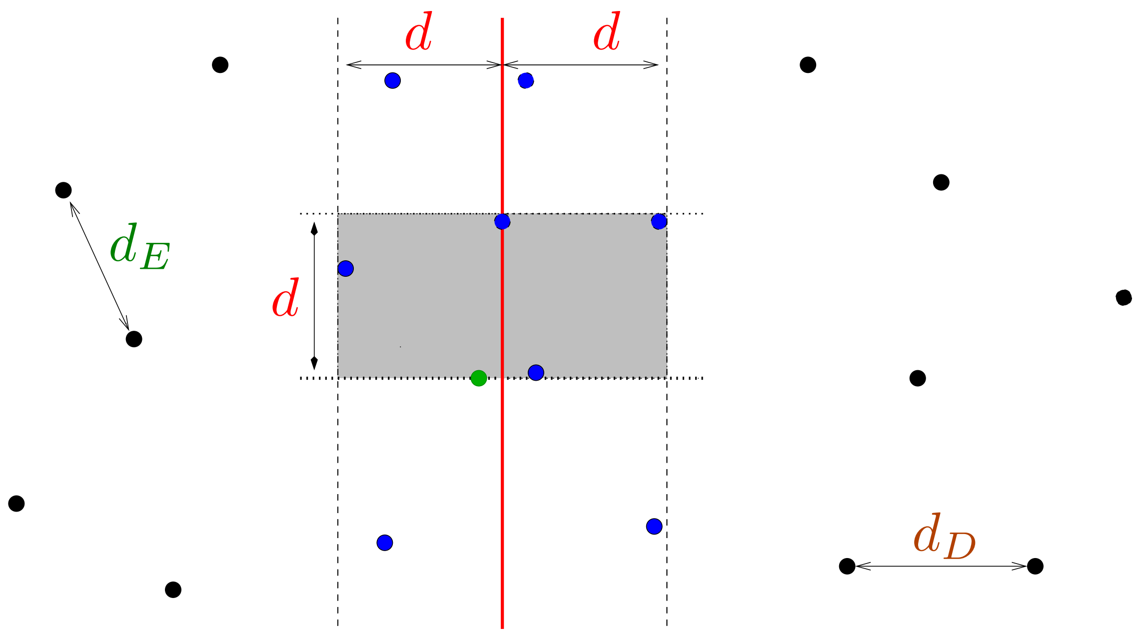


Quantos pontos assim há?

Em cada um dos dois quadrados de lado d,
há no máximo 4 pontos porque $d \leq d_E$ e $d \leq d_D$.

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?



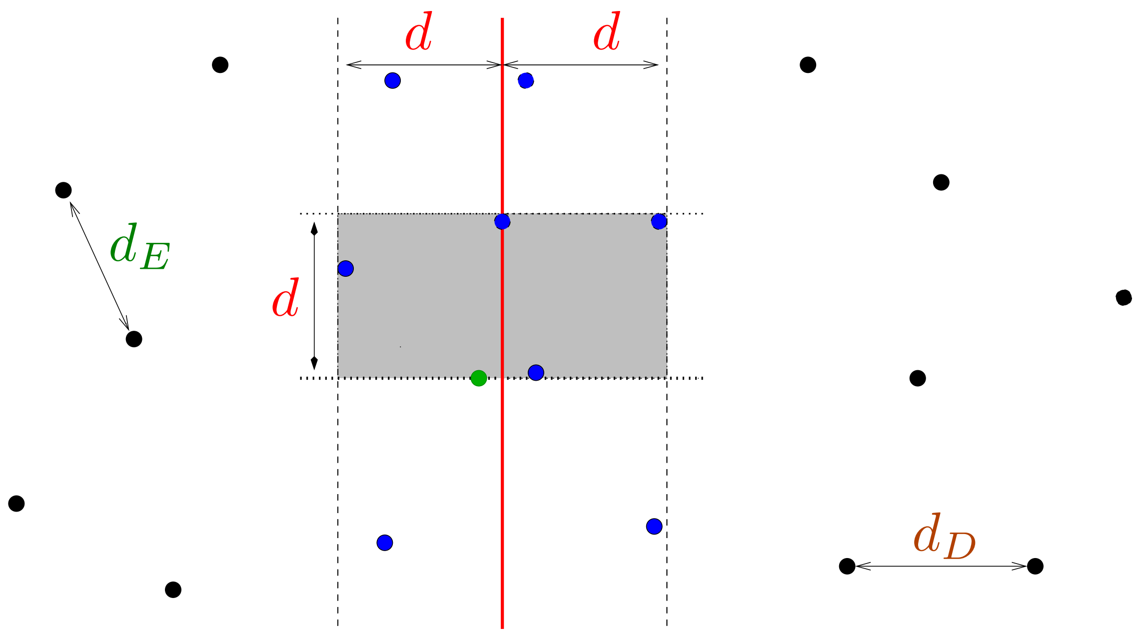
Quantos pontos assim há?

Em cada um dos dois quadrados de lado d,
há no máximo 4 pontos porque $d \leq d_E$ e $d \leq d_D$.

Logo há não mais que 7 pontos assim (excluindo o ponto).

Algoritmo de Shamos e Hoey

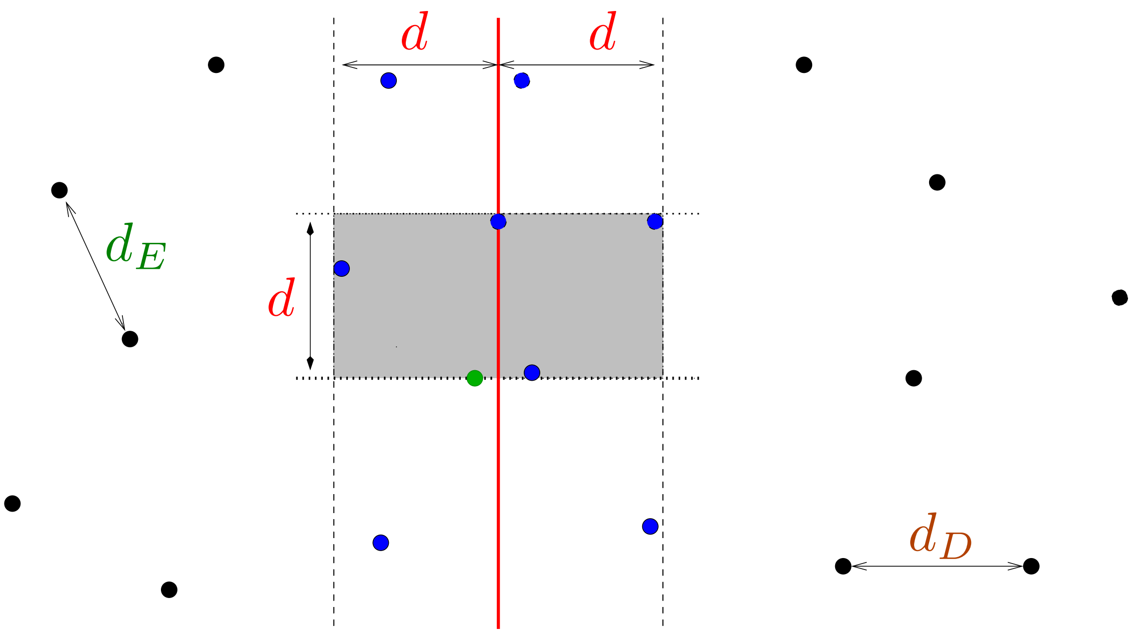
Como fazer o COMBINE linear?



Mas como ter acesso rápido a estes pontos?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?



Mas como ter acesso rápido a estes pontos?

Alteraremos o pré-processamento, para ter acesso aos pontos ordenados pelas suas Y-coordenadas também.

Algoritmo de Shamos e Hoey

Pré-processamento: ordenar os pontos pela X-coordenada e ordenar (indiretamente) os pontos pela Y-coordenada

Algoritmo de Shamos e Hoey

Pré-processamento: ordenar os pontos pela X-coordenada e ordenar (indiretamente) os pontos pela Y-coordenada

DISTÂNCIA-SH(X, Y, n)
MERGESORT$(X, Y, 1, n)$
para $i \leftarrow 1$ até n faça
    $a[i] \leftarrow i$
MERGESORTIND$(Y, 1, n, a)$ $\triangleright$ ordenação indireta
devolva DISTÂNCIAREC-SH $(X, Y, a, 1, n)$

Algoritmo de Shamos e Hoey

Pré-processamento: ordenar os pontos pela X-coordenada e ordenar (indiretamente) os pontos pela Y-coordenada

DISTÂNCIA-SH(X, Y, n)

```
MERGESORT(X, Y, 1, n)
para i ← 1 até n faça
      a[i] ← i
MERGESORTIND(Y, 1, n, a)      ▷ ordenação indireta
devolva DISTÂNCIAREC-SH (X, Y, a, 1, n)
```

Consumo de tempo:
De novo, $O(n \lg n)$ mais o tempo do DISTÂNCIAREC-SH.

Divisão e conquista

DISTÂNCIAREC-SH (X, Y, a, p, r)

Dividir: Seja $q := \lfloor (p + r)/2 \rfloor$. Obtenha um vetor $b[p \, . \, . \, r]$ tal que $X[p \, . \, . \, q], Y[p \, . \, . \, q]$, $b[p \, . \, . \, q]$ seja uma representação ordenada dos pontos mais à esquerda e $X[q{+}1 \, . \, . \, r], Y[q{+}1 \, . \, . \, r]$, $b[q{+}1 \, . \, . \, r]$, uma representação ordenada dos pontos mais à direita.

Conquistar: Determine, recursivamente, a menor distância d_E entre dois pontos da esquerda e a menor distância d_D entre dois pontos da direita.

Combinar: Devolva o mínimo entre d_E, d_D e a menor distância d_{ED} entre um ponto da esquerda e um ponto da direita.

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, a, p, r) ▷ Divisão e conquista

se $r \leq p + 2$

  então ▷ resolva o problema diretamente

  senão $q \leftarrow \lfloor (p+r)/2 \rfloor$

    $b \leftarrow$ DIVIDA (X, Y, a, p, r)

    $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, b, p, q)

    $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, b, q+1, r)$

    devolva COMBINE $(X, Y, a, p, r, d_E, d_D)$

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, a, p, r) $\triangleright$ Divisão e conquista

se $r \leq p + 2$
    então $\triangleright$ resolva o problema diretamente
    senão $q \leftarrow \lfloor (p + r)/2 \rfloor$
        $b \leftarrow$ DIVIDA (X, Y, a, p, r)
        $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, b, p, q)
        $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, b, q + 1, r)$
        devolva COMBINE $(X, Y, a, p, r, d_E, d_D)$

DIVIDA e COMBINE são algoritmos lineares.

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, a, p, r) $\triangleright$ Divisão e conquista

```
se r ≤ p + 2
    então ▷ resolva o problema diretamente
    senão q ← ⌊(p + r)/2⌋
        b ← DIVIDA (X, Y, a, p, r)
        d_E ← DISTÂNCIAREC-SH (X, Y, b, p, q)
        d_D ← DISTÂNCIAREC-SH (X, Y, b, q + 1, r)
        devolva COMBINE (X, Y, a, p, r, d_E, d_D)
```

DIVIDA e COMBINE são algoritmos lineares.

Consumo de tempo:

$$T(n) \ = \ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \mathrm{O}(n)$$

onde $n = r - p + 1$.

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, a, p, r) $\triangleright$ Divisão e conquista

se $r \le p + 2$
    então $\triangleright$ resolva o problema diretamente
    senão $q \leftarrow \lfloor (p + r)/2 \rfloor$
        $b \leftarrow$ DIVIDA (X, Y, a, p, r)
        $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, b, p, q)
        $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, b, q + 1, r)$
        devolva COMBINE $(X, Y, a, p, r, d_E, d_D)$

DIVIDA e COMBINE são algoritmos lineares.

Consumo de tempo:

$$T(n) \; = \; T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \mathrm{O}(n)$$

onde $n = r - p + 1$. Como antes, $T(n) = \mathrm{O}(n \lg n)$.

Algoritmo de Shamos e Hoey

DIVIDA (X, Y, a, p, r)

$q \leftarrow \lfloor (p+r)/2 \rfloor$
$i \leftarrow p-1 \quad j \leftarrow q$
para $k \leftarrow p$ até r faça
    se $a[k] \leq q$ $\quad \triangleright$ $(X[a[k]], Y[a[k]])$ está à esquerda da reta $x = X[q]$?
        então $i \leftarrow i+1$
            $b[i] \leftarrow a[k]$
        senão $j \leftarrow j+1$
            $b[j] \leftarrow a[k]$
devolva b

Algoritmo de Shamos e Hoey

DIVIDA (X, Y, a, p, r)

$q \leftarrow \lfloor (p+r)/2 \rfloor$
$i \leftarrow p-1 \quad j \leftarrow q$
para $k \leftarrow p$ até r faça
  se $a[k] \leq q$
    então $i \leftarrow i+1$
      $b[i] \leftarrow a[k]$
    senão $j \leftarrow j+1$
      $b[j] \leftarrow a[k]$
devolva b

X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5
b					

$q = 3 \qquad X[q] = 1$

Algoritmo de Shamos e Hoey

DIVIDA (X, Y, a, p, r)

```
q ← ⌊(p + r)/2⌋
i ← p − 1     j ← q
para k ← p até r faça
      se a[k] ≤ q
            então  i ← i + 1
                   b[i] ← a[k]
            senão  j ← j + 1
                   b[j] ← a[k]
devolva b
```

	k				
X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5

b	3				

$q = 3 \qquad X[q] = 1$

Algoritmo de Shamos e Hoey

DIVIDA (X, Y, a, p, r)

$q \leftarrow \lfloor (p + r)/2 \rfloor$
$i \leftarrow p - 1 \quad j \leftarrow q$
para $k \leftarrow p$ até r faça
    se $a[k] \leq q$
        então $i \leftarrow i + 1$
            $b[i] \leftarrow a[k]$
        senão $j \leftarrow j + 1$
            $b[j] \leftarrow a[k]$
devolva b

		k			
X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5
b	3	1			

$q = 3 \qquad X[q] = 1$

Algoritmo de Shamos e Hoey

DIVIDA (X, Y, a, p, r)

$q \leftarrow \lfloor (p+r)/2 \rfloor$

$i \leftarrow p-1 \quad j \leftarrow q$

para $k \leftarrow p$ até r faça

    se $a[k] \leq q$

        então $i \leftarrow i+1$

            $b[i] \leftarrow a[k]$

        senão $j \leftarrow j+1$

            $b[j] \leftarrow a[k]$

devolva b

			k		
X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5
b	3	1		5	

$q = 3 \qquad X[q] = 1$

Algoritmo de Shamos e Hoey

```
DIVIDA (X, Y, a, p, r)
    q ← ⌊(p + r)/2⌋
    i ← p − 1     j ← q
    para k ← p até r faça
        se a[k] ≤ q
            então  i ← i + 1
                   b[i] ← a[k]
            senão  j ← j + 1
                   b[j] ← a[k]
    devolva b
```

X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5
b	3	1	2	5	4

$q = 3 \qquad X[q] = 1$

Algoritmo de Shamos e Hoey

DIVIDA (X, Y, a, p, r)

```
q ← ⌊(p + r)/2⌋
i ← p − 1     j ← q
para k ← p até r faça
    se a[k] ≤ q
        então  i ← i + 1
               b[i] ← a[k]
        senão  j ← j + 1
               b[j] ← a[k]
devolva b
```

X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5
b	3	1	2	5	4

$q = 3 \qquad X[q] = 1$

Consumo de tempo:

É fácil ver que o consumo é $\Theta(n)$ onde $n = r - p + 1$.

Algoritmo de Shamos e Hoey

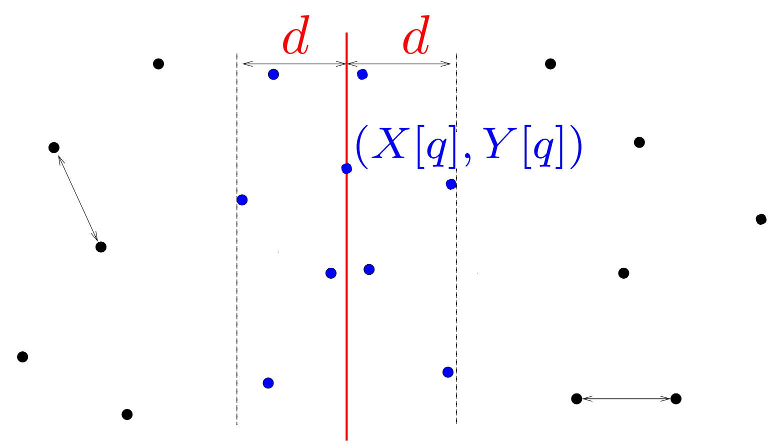
A rotina abaixo identifica os pontos que estão na faixa, ordenados pela Y-coordenada.

Algoritmo de Shamos e Hoey

A rotina abaixo identifica os pontos que estão na faixa, ordenados pela Y-coordenada.

CANDIDATOS (X, a, p, r, d)

$q \leftarrow \lfloor (p+r)/2 \rfloor$
$t \leftarrow 0$
para $k \leftarrow p$ até r faça
    se $|X[a[k]] - X[q]| < d$
        então $t \leftarrow t + 1$
            $f[t] \leftarrow a[k]$
devolva (f, t)



Algoritmo de Shamos e Hoey

A rotina abaixo identifica os pontos que estão na faixa, ordenados pela Y-coordenada.

CANDIDATOS (X, a, p, r, d)

```
q ← ⌊(p + r)/2⌋
t ← 0
para k ← p até r faça
    se |X[a[k]] − X[q]| < d
        então t ← t + 1
              f[t] ← a[k]
devolva (f, t)
```

X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5
f					

$X[3] = 1 \qquad d = \sqrt{5}$

Algoritmo de Shamos e Hoey

A rotina abaixo identifica os pontos que estão na faixa, ordenados pela Y-coordenada.

CANDIDATOS (X, a, p, r, d)

$q \leftarrow \lfloor (p+r)/2 \rfloor$
$t \leftarrow 0$
para $k \leftarrow p$ até r faça
  se $|X[a[k]] - X[q]| < d$
    então $t \leftarrow t+1$
      $f[t] \leftarrow a[k]$
devolva (f, t)

X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5

f	3	2	4

$X[3] = 1 \qquad d = \sqrt{5}$

Algoritmo de Shamos e Hoey

A rotina abaixo identifica os pontos que estão na faixa, ordenados pela Y-coordenada.

CANDIDATOS (X, a, p, r, d)

$q \leftarrow \lfloor (p+r)/2 \rfloor$
$t \leftarrow 0$
para $k \leftarrow p$ até r faça
    se $|X[a[k]] - X[q]| < d$
        então $t \leftarrow t + 1$
            $f[t] \leftarrow a[k]$
devolva (f, t)

X	-2	0	1	3	4
Y	-1	3	-2	4	2
a	3	1	5	2	4
	1	2	3	4	5

f	3	2	4

$X[3] = 1$ $\quad$ $d = \sqrt{5}$

Consumo de tempo:

É fácil ver que o consumo é $\Theta(n)$ onde $n = r - p + 1$.

Algoritmo de Shamos e Hoey

COMBINE $(X, Y, a, p, r, d_E, d_D)$

$d \leftarrow \min\{d_E, d_D\}$

$(f, t) \leftarrow$ CANDIDATOS (X, a, p, r, d) $\triangleright$ pontos na faixa

para $i \leftarrow 1$ até $t-1$ faça

    para $j \leftarrow i+1$ até $\min\{i+7, t\}$ faça $\triangleright$ ≤ 7 próximos

        $d' \leftarrow$ DIST$(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])$

        se $d' < d$

            então $d \leftarrow d'$

devolva d

Algoritmo de Shamos e Hoey

COMBINE $(X, Y, a, p, r, d_E, d_D)$

$d \leftarrow \min\{d_E, d_D\}$
$(f, t) \leftarrow$ CANDIDATOS (X, a, p, r, d) ▷ pontos na faixa
para $i \leftarrow 1$ até $t - 1$ faça
    para $j \leftarrow i + 1$ até $\min\{i + 7, t\}$ faça ▷ $\leq$ 7 próximos
        $d' \leftarrow \text{DIST}(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])$
        se $d' < d$
            então $d \leftarrow d'$
devolva d

Consumo de tempo:

É fácil ver que o consumo é $\Theta(n)$ onde $n = r - p + 1$.

Algoritmo de Shamos e Hoey

COMBINE $(X, Y, a, p, r, d_E, d_D)$

$d \leftarrow \min\{d_E, d_D\}$
$(f, t) \leftarrow$ CANDIDATOS (X, a, p, r, d) ▷ pontos na faixa
para $i \leftarrow 1$ até $t - 1$ faça
    $j \leftarrow i + 1$
    enquanto $j \leq t$ e $Y[f[j]] - Y[f[i]] < d$ faça ▷ ≤ 7 próximos
        $d' \leftarrow$ DIST$(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])$
        se $d' < d$
            então $d \leftarrow d'$
        $j \leftarrow j + 1$
devolva d

Algoritmo de Shamos e Hoey

COMBINE $(X, Y, a, p, r, d_E, d_D)$

$d \leftarrow \min\{d_E, d_D\}$
$(f, t) \leftarrow$ CANDIDATOS (X, a, p, r, d) $\triangleright$ pontos na faixa
para $i \leftarrow 1$ até $t - 1$ faça
    $j \leftarrow i + 1$
    enquanto $j \leq t$ e $Y[f[j]] - Y[f[i]] < d$ faça $\triangleright \leq 7$ próximos
        $d' \leftarrow$ DIST$(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])$
        se $d' < d$
            então $d \leftarrow d'$
        $j \leftarrow j + 1$
devolva d

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

Finalizando...

Exercício: Adapte os algoritmos vistos nesta aula para que devolvam dois pontos da coleção que estejam à distância mínima (em vez de devolver a distância apenas).

Finalizando...

Exercício: Adapte os algoritmos vistos nesta aula para que devolvam dois pontos da coleção que estejam à distância mínima (em vez de devolver a distância apenas).

Exercício: O algoritmo de divisão e conquista funciona se trocarmos o 7 na implementação com o comando **para** por 6? E se trocarmos o 6 por 5?

Finalizando...

Exercício: Adapte os algoritmos vistos nesta aula para que devolvam dois pontos da coleção que estejam à distância mínima (em vez de devolver a distância apenas).

Exercício: O algoritmo de divisão e conquista funciona se trocarmos o 7 na implementação com o comando **para** por 6? E se trocarmos o 6 por 5?

Exercício: Modifique os algoritmos para que não usem o vetor a. (Para tanto, no COMBINE, adicionalmente rearranje os vetores X e Y para que saiam ordenados pela Y-coordenada. Ou seja, na entrada da recursão os pontos estão ordenados por X-coordenada e na saída, estão ordenados por Y-coordenada.)

Finalizando...

Exercício: Adapte os algoritmos vistos nesta aula para que devolvam dois pontos da coleção que estejam à distância mínima (em vez de devolver a distância apenas).

Exercício: O algoritmo de divisão e conquista funciona se trocarmos o 7 na implementação com o comando **para** por 6? E se trocarmos o 6 por 5?

Exercício: Modifique os algoritmos para que não usem o vetor a. (Para tanto, no COMBINE, adicionalmente rearranje os vetores X e Y para que saiam ordenados pela Y-coordenada. Ou seja, na entrada da recursão os pontos estão ordenados por X-coordenada e na saída, estão ordenados por Y-coordenada.)

Material coberto nas duas primeiras aulas:
Secs 7.1 – 7.3 do livro das JAI.