Geometria Computacional

Cristina G. Fernandes

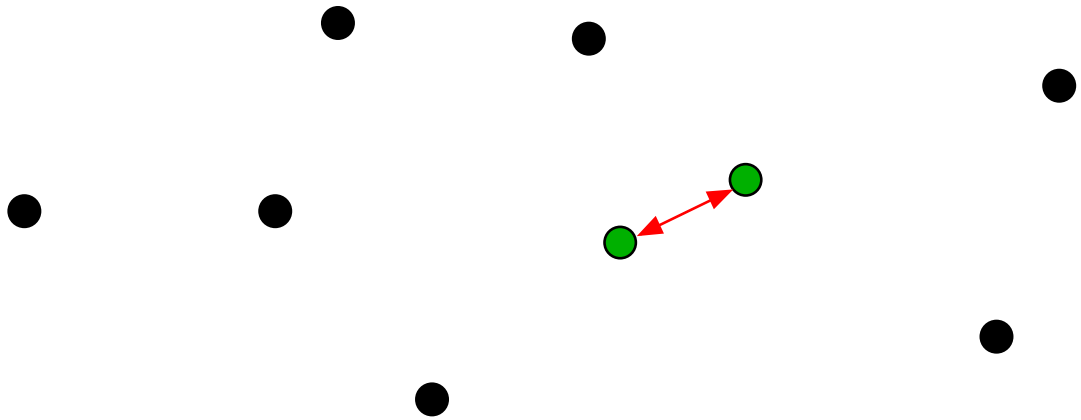
Departamento de Ciência da Computação do IME-USP

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

segundo semestre de 2011

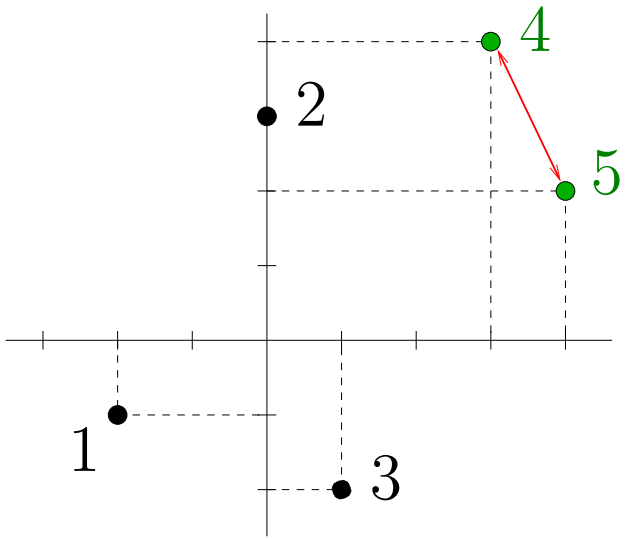
Par de pontos mais próximos

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.



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

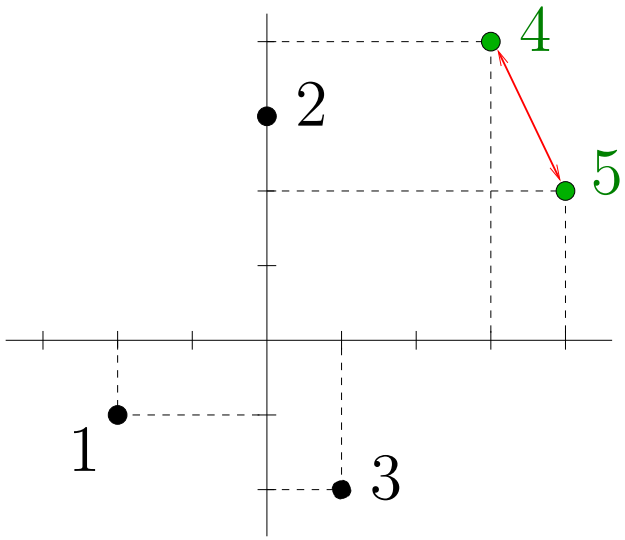
Entrada: coleção de n pontos representada por vetores $X[1\,..\,n]$ e $Y[1\,..\,n]$.

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.



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

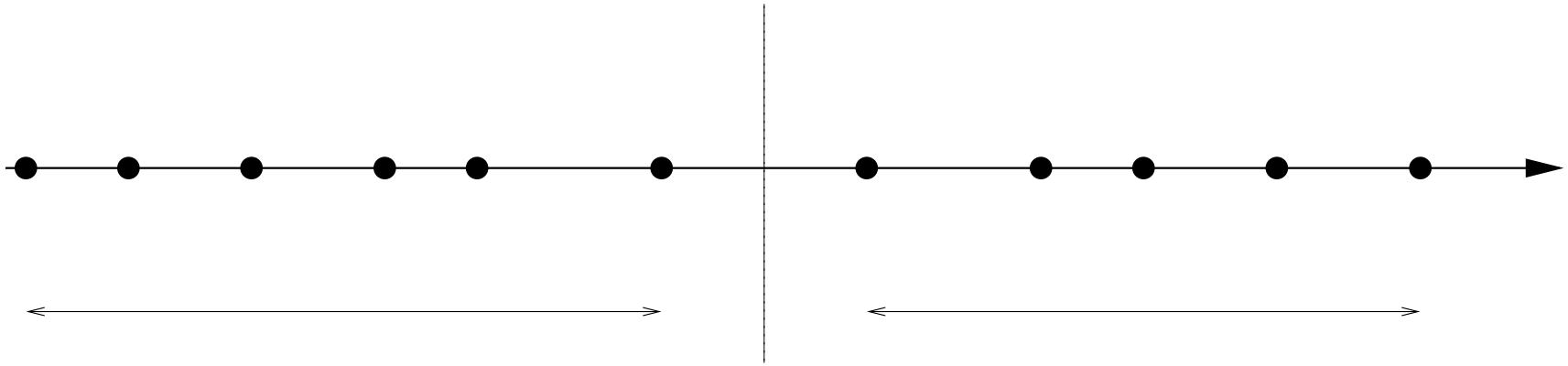
Entrada: coleção de n pontos representada por vetores $X[1\,..\,n]$ e $Y[1\,..\,n]$.

Saída: menor distância entre dois pontos da coleção.

Par de pontos mais próximos na reta

Algoritmo de divisão e conquista:

Divisão: dividir a instância do problema em instâncias menores do problema.

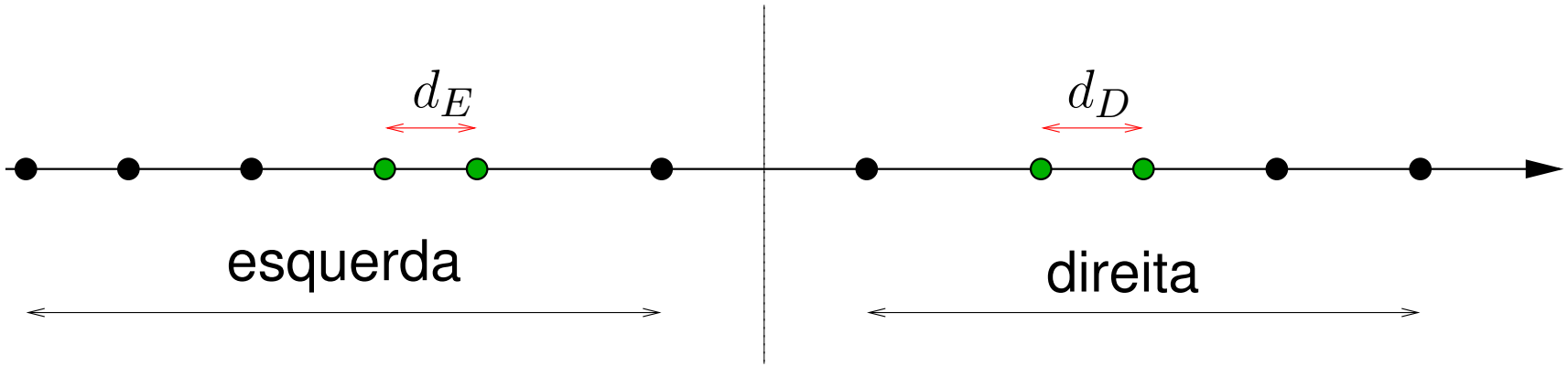


Par de pontos mais próximos na reta

Algoritmo de divisão e conquista:

Divisão: dividir a instância do problema em instâncias menores do problema.

Conquista: resolver o problema nas instâncias menores recursivamente (ou diretamente, se elas forem pequenas o suficiente).



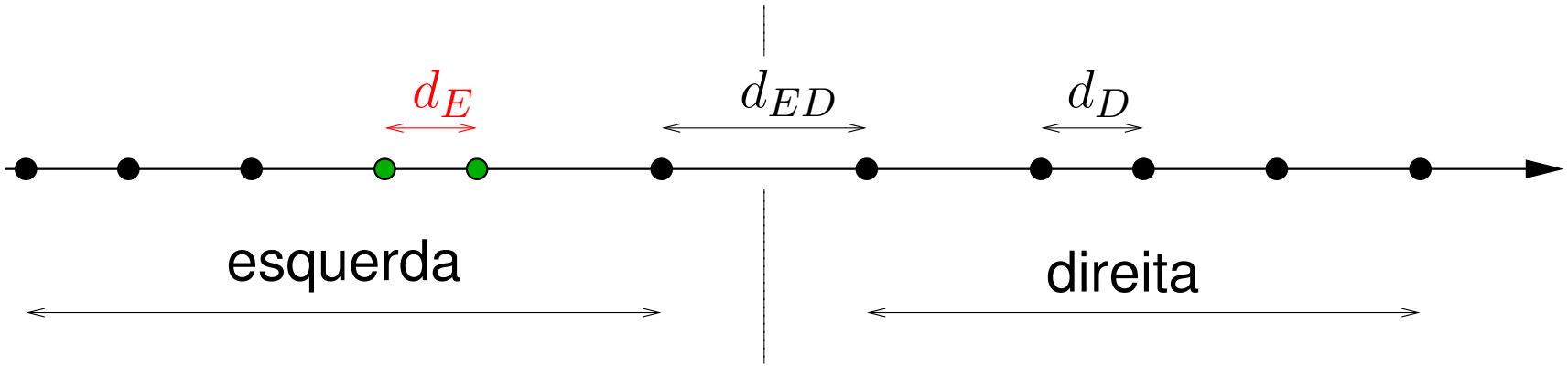
Par de pontos mais próximos na reta

Algoritmo de divisão e conquista:

Divisão: dividir a instância do problema em instâncias menores do problema.

Conquista: resolver o problema nas instâncias menores recursivamente (ou diretamente, se elas forem pequenas o suficiente).

Combinação: combinar as soluções das instâncias menores para gerar uma solução da instância original.



Par mais próximo na reta

```
DistânciaReta(X, n)
1  MergeSort(X, 1, n)
2  devolva DistânciaRetaRec (X, 1, n)

DistânciaRetaRec (X, p, r)      ▷ Divisão e conquista
1  se r ≤ p + 1
2      então  se r = p
3                 então devolva +∞
4                 senão devolva X[r] − X[p]
5      senão q ← ⌊(p + r)/2⌋
6            d_E ← DistânciaRetaRec(X, p, q)
7            d_D ← DistânciaRetaRec(X, q + 1, r)
8            d ← min{d_E, d_D, X[q+1] − X[q]}
9            devolva d
```

Par mais próximo na reta

```
DistânciaReta(X, n)
1  MergeSort(X, 1, n)
2  devolva DistânciaRetaRec (X, 1, n)

DistânciaRetaRec (X, p, r)      ▷ Divisão e conquista
1  se r ≤ p + 1
2      então  se r = p
3                 então devolva +∞
4                 senão devolva X[r] − X[p]
5      senão q ← ⌊(p + r)/2⌋
6            d_E ← DistânciaRetaRec(X, p, q)
7            d_D ← DistânciaRetaRec(X, q + 1, r)
8            d ← min{d_E, d_D, X[q+1] − X[q]}
9            devolva d
```

Consumo de tempo: $\mathrm{O}(n \lg n) + \Theta(n)$.

Par mais próximo no plano

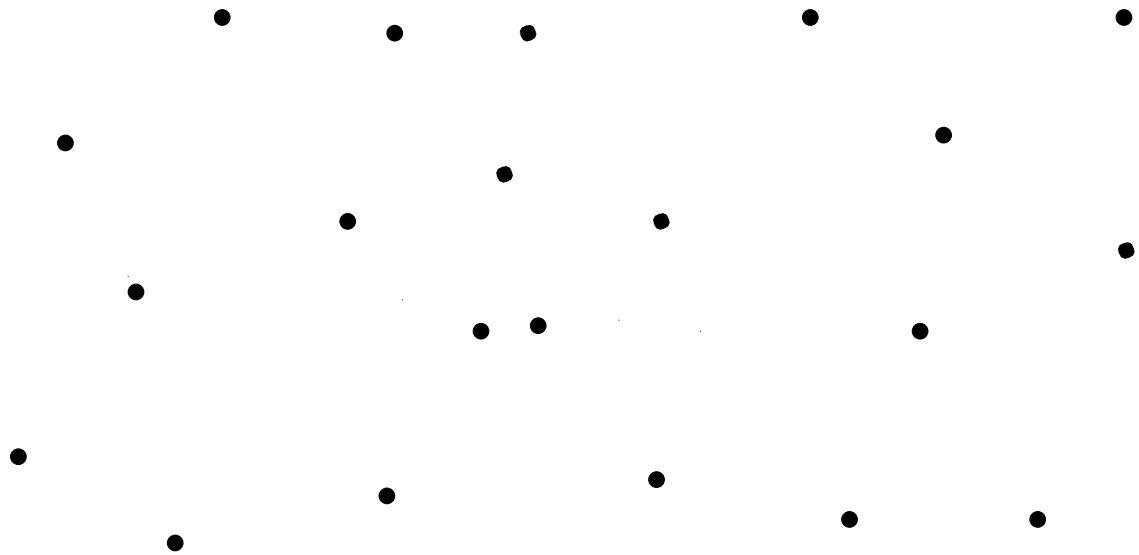
Obtivemos um algoritmo $\mathrm{O}(n \lg n)$ para pontos na reta.

Como generalizar essa ideia para o plano?

Par mais próximo no plano

Obtivemos um algoritmo $\mathrm{O}(n \lg n)$ para pontos na reta.

Como generalizar essa ideia para o plano?

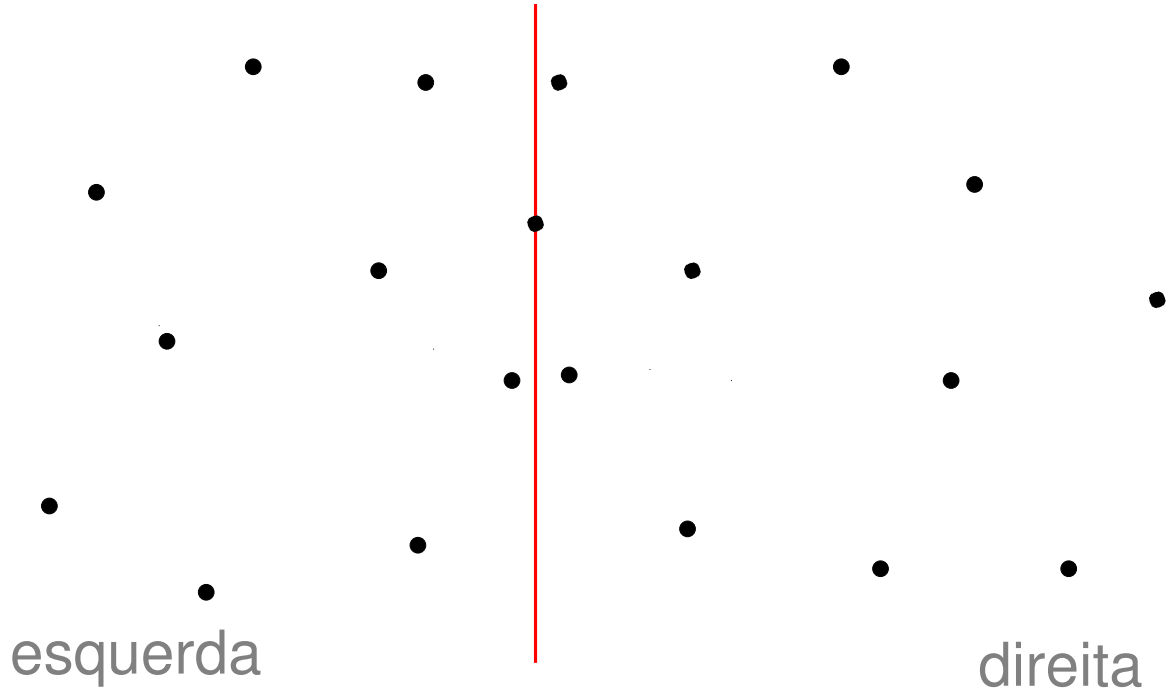


Par mais próximo no plano

Obtivemos um algoritmo $\mathrm{O}(n \lg n)$ para pontos na reta.

Como generalizar essa ideia para o plano?

Divide...

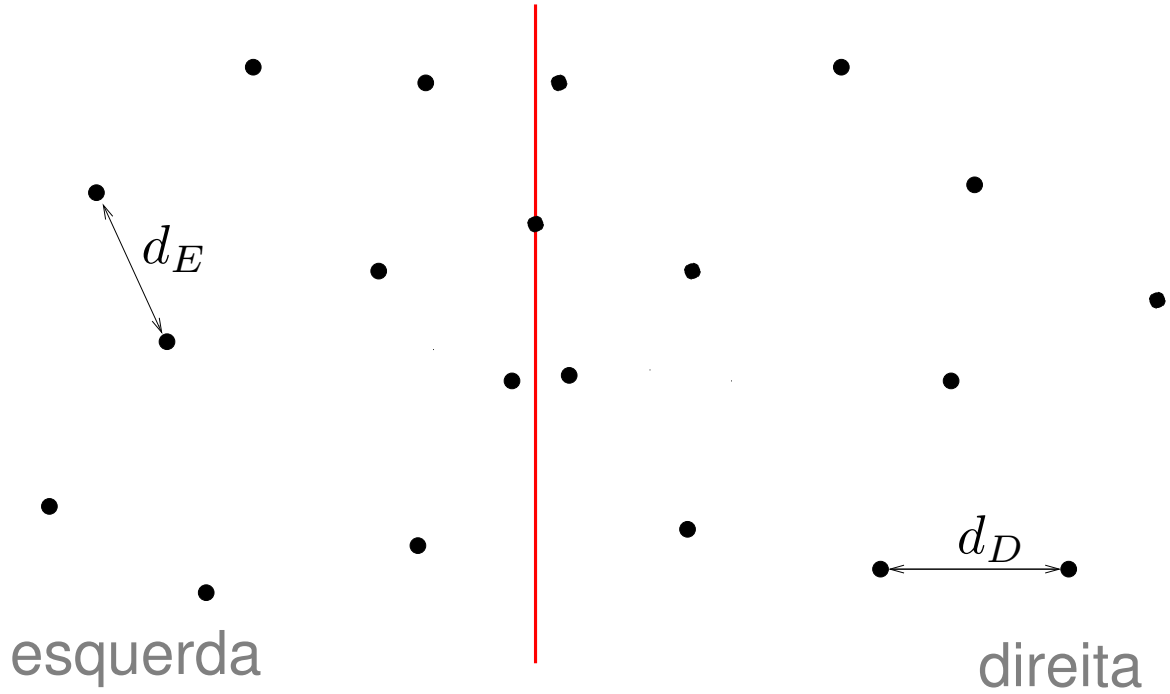


Par mais próximo no plano

Obtivemos um algoritmo $\mathrm{O}(n \lg n)$ para pontos na reta.

Como generalizar essa ideia para o plano?

Divide... Conquista...

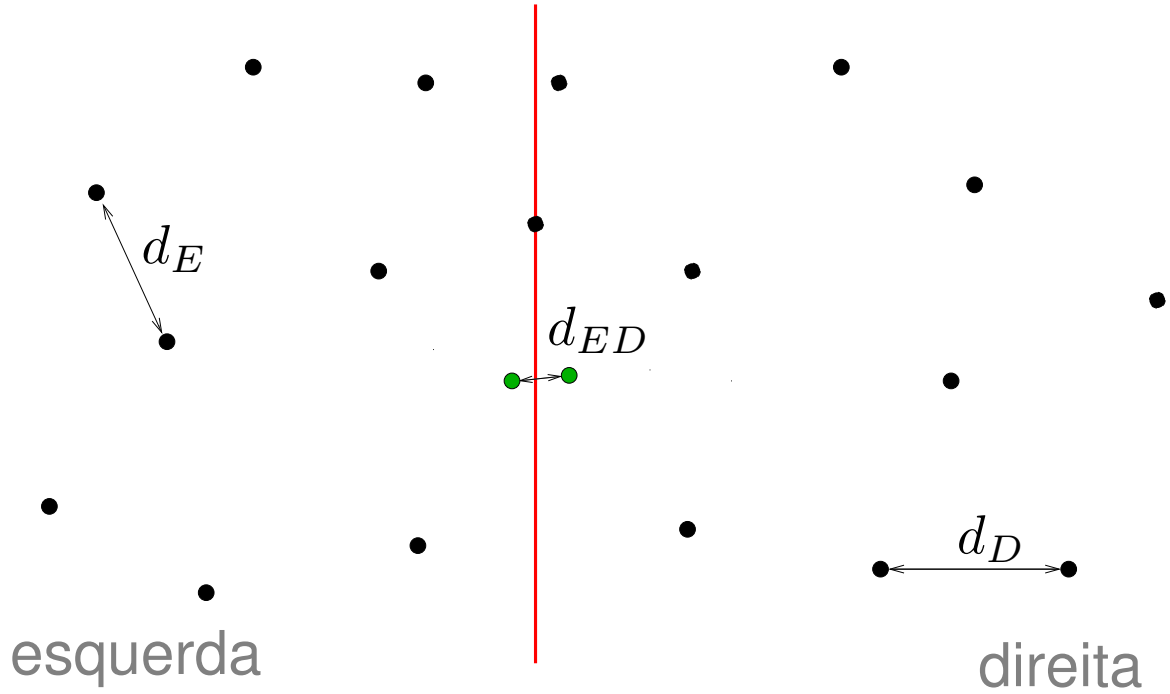


Par mais próximo no plano

Obtivemos um algoritmo $\mathrm{O}(n \lg n)$ para pontos na reta.

Como generalizar essa ideia para o plano?

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: $X[p\,..\,q], Y[p\,..\,q]$ (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.

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

Algoritmo de Shamos e Hoey

```
DISTÂNCIAREC-SH (X, Y, p, r)        ▷ Divisão e conquista
1  se r ≤ p + 2
2      então  ▷ resolva o problema diretamente
3      senão  q ← ⌊(p + r)/2⌋
4             d_E ← DISTÂNCIAREC-SH (X, Y, p, q)
5             d_D ← DISTÂNCIAREC-SH (X, Y, q+1, r)
6             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

1 se $r \leq p + 2$
2 então ▷ resolva o problema diretamente
3 senão $q \leftarrow \lfloor (p + r)/2 \rfloor$
4 $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, p, q)
5 $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, q{+}1, r)$
6 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 ≤ 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.

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) ▷ 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.

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 $T(n)$? $T(n) = \mathrm{O}(n \lg n)$.

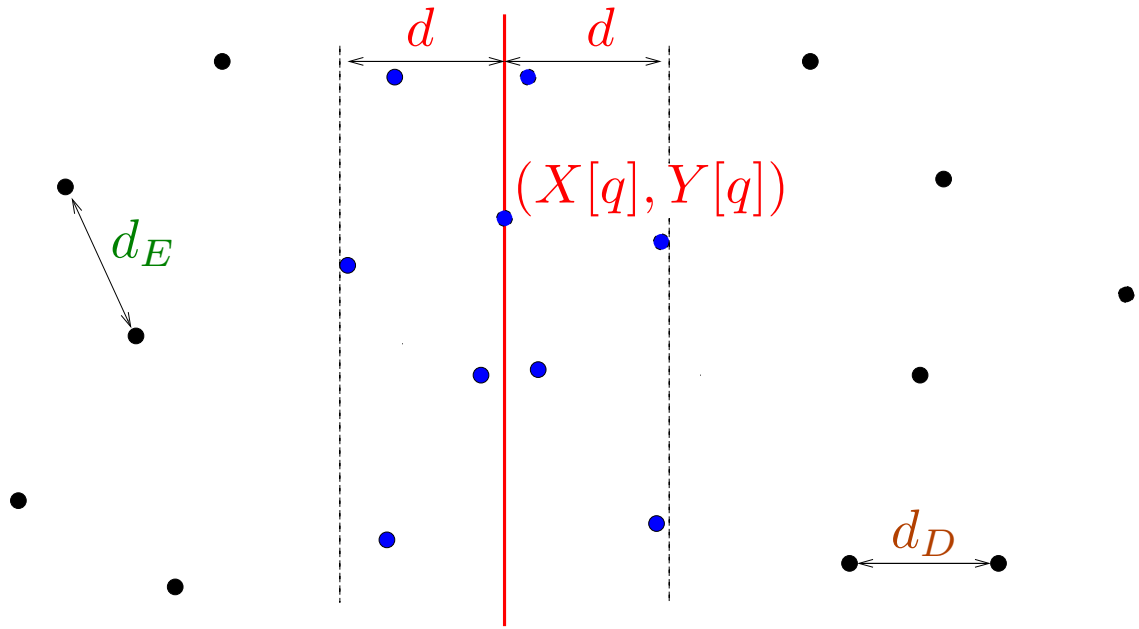
Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

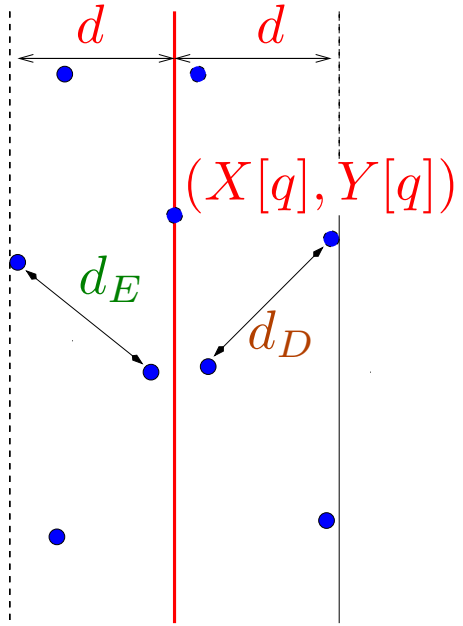
COMBINE precisa considerar apenas pontos que estão a uma distância menor que $d = \min\{d_E, d_D\}$ da reta vertical $x = X[q]$.



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...

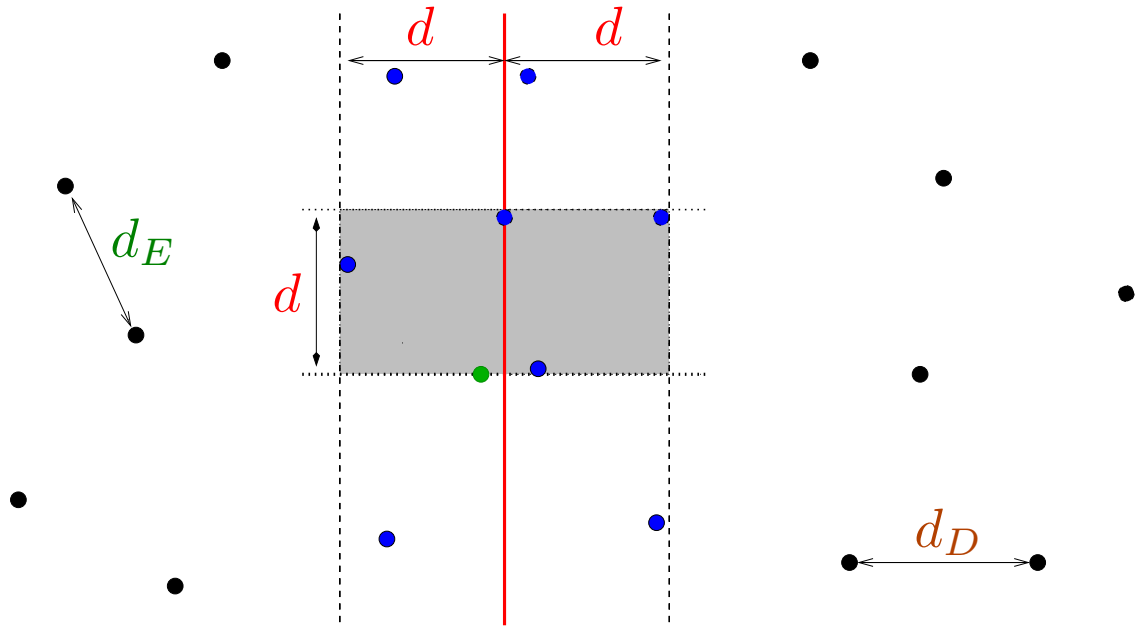
Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

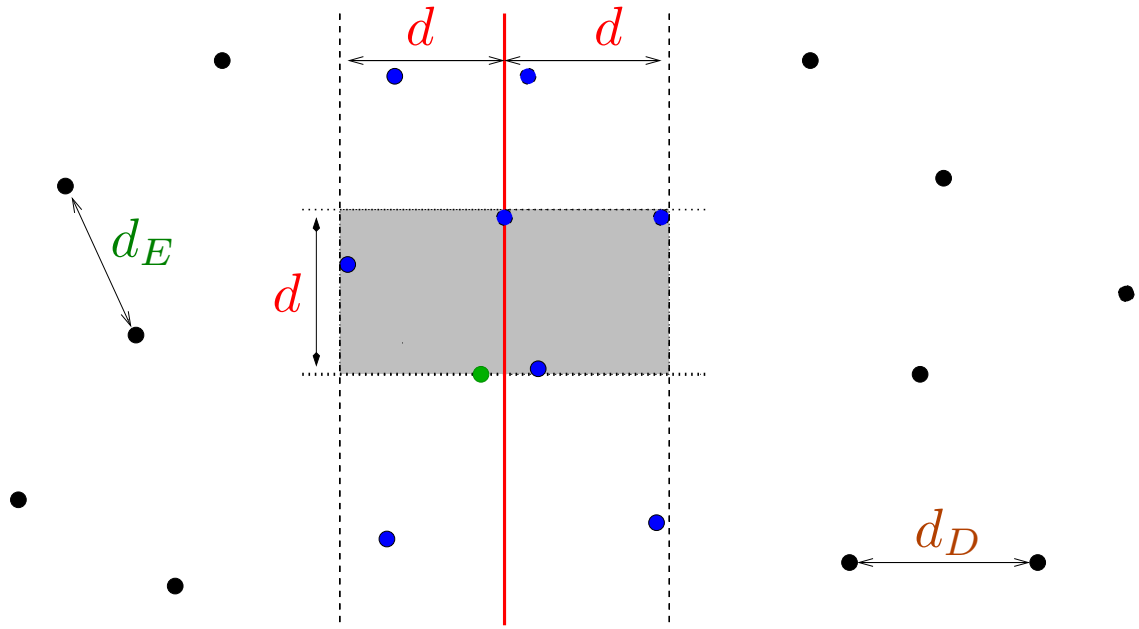
Ideia...



Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

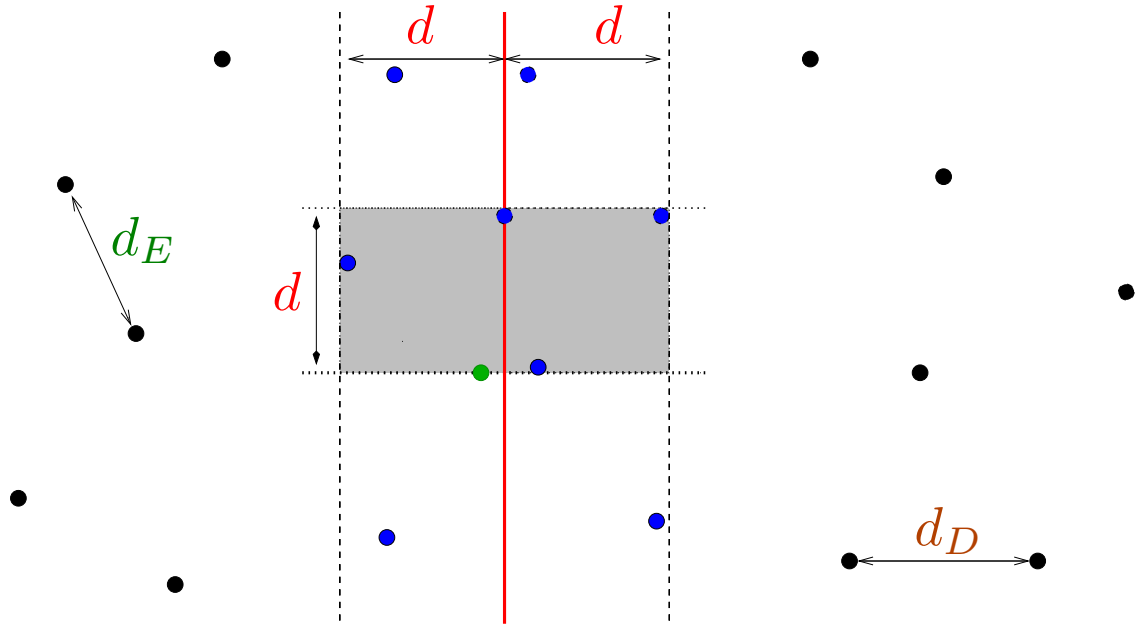
Ideia...



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

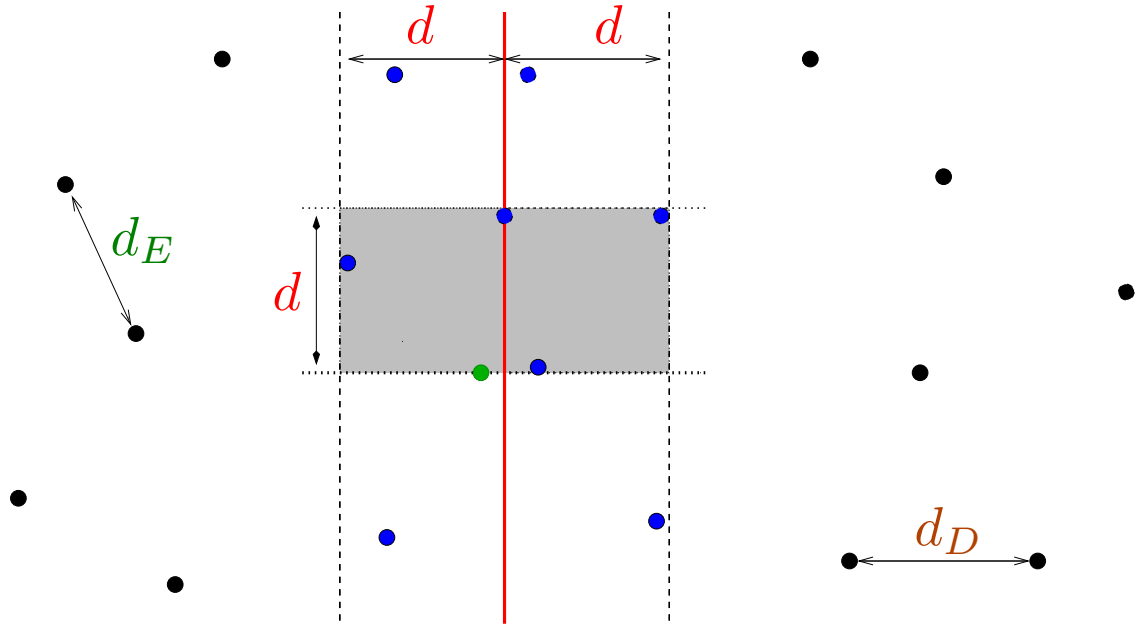
Como fazer o COMBINE linear?



Quantos pontos assim há?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

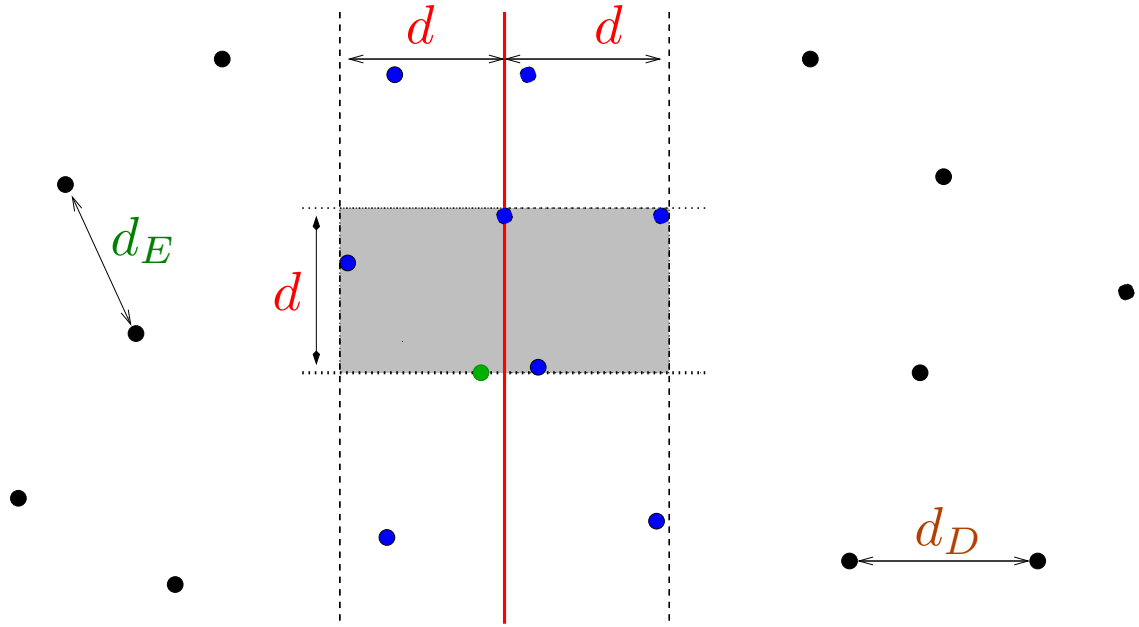


Quantos pontos assim há?

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?



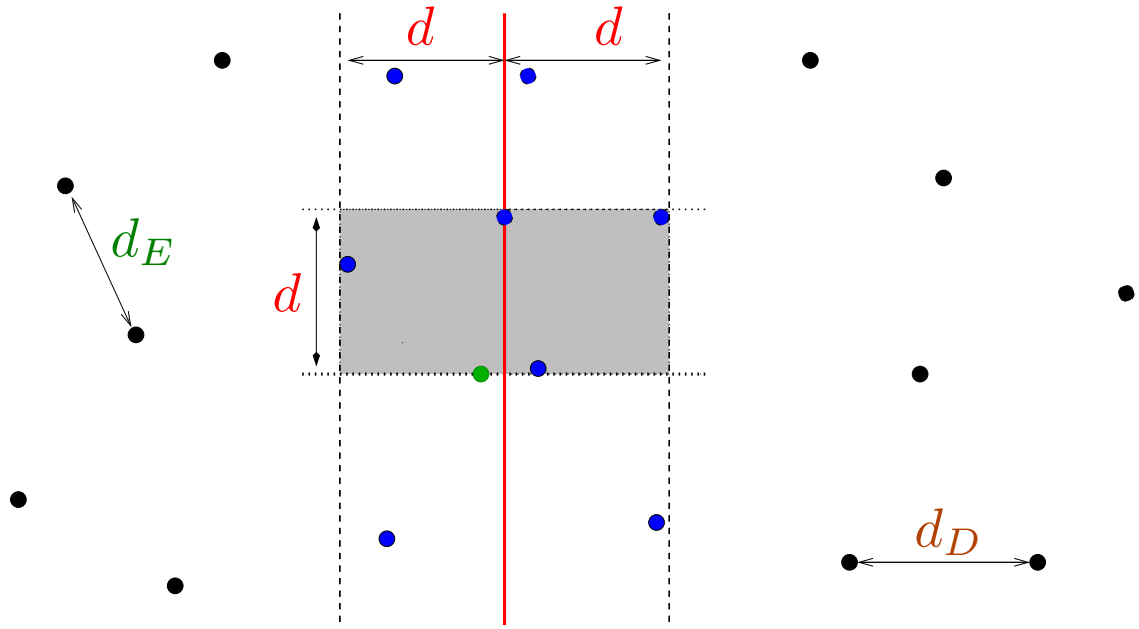
Quantos pontos assim há?

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

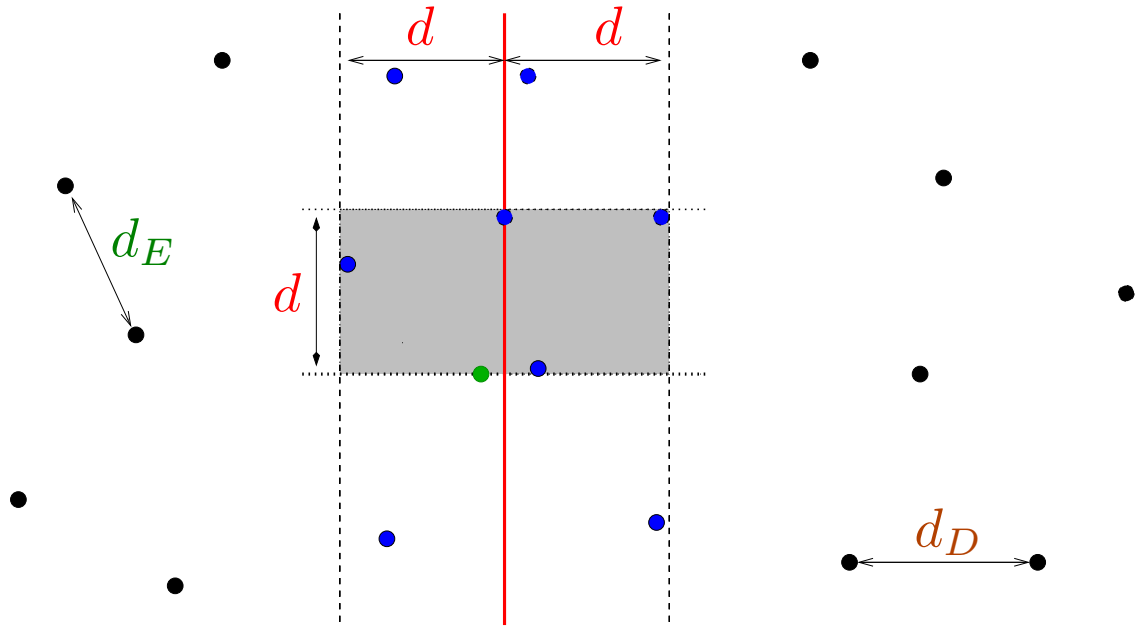
Como fazer o COMBINE linear?



Mas como ter acesso rápido a estes pontos?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

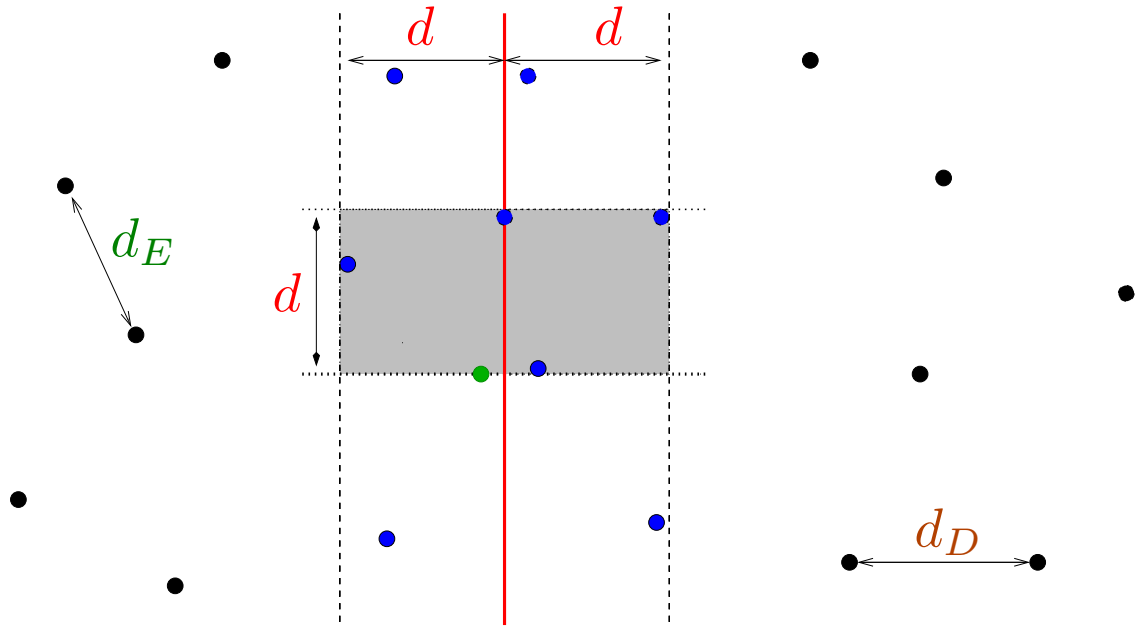


Mas como ter acesso rápido a estes pontos?

No COMBINE, precisamos ter acesso aos pontos ordenados pelas suas Y-coordenadas.

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?



Mas como ter acesso rápido a estes pontos?

No COMBINE, precisamos ter acesso aos pontos ordenados pelas suas Y-coordenadas.

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

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)

```
1  MERGESORT(X, Y, 1, n)
2  para i ← 1 até n faça
3      a[i] ← i
4  MERGESORTIND(Y, 1, n, a)        ▷ ordenação indireta
5  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)
1 MERGESORT$(X, Y, 1, n)$
2 para $i \leftarrow 1$ até n faça
3 $\quad a[i] \leftarrow i$
4 MERGESORTIND$(Y, 1, n, a)$ $\quad \triangleright$ ordenação indireta
5 devolva DISTÂNCIAREC-SH $(X, Y, a, 1, n)$

Consumo de tempo:

De novo, $\mathrm{O}(n \lg n)$ 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
1  se r ≤ p + 2
2      então  ▷ resolva o problema diretamente
3      senão  q ← ⌊(p + r)/2⌋
4             b ← DIVIDA (X, Y, a, p, r)
5             d_E ← DISTÂNCIAREC-SH (X, Y, b, p, q)
6             d_D ← DISTÂNCIAREC-SH (X, Y, b, q + 1, r)
7             devolva COMBINE (X, Y, a, p, r, d_E, d_D)
```

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, a, p, r) ▷ 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.

Algoritmo de Shamos e Hoey

DISTÂNCIAREC-SH (X, Y, a, p, r) ▷ 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) ▷ 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 $n = r - p + 1$. Como antes, $T(n) = \mathrm{O}(n \lg n)$.

Algoritmo de Shamos e Hoey

```
DIVIDA (X, Y, a, p, r)
1  q ← ⌊(p + r)/2⌋
2  i ← p − 1     j ← q
3  para k ← p até r faça
4      se a[k] ≤ q     ▷ (X[a[k]], Y[a[k]]) está à esquerda da reta x = X[q]?
5          então  i ← i + 1
6                 b[i] ← a[k]
7          senão  j ← j + 1
8                 b[j] ← a[k]
9  devolva b
```

Algoritmo de Shamos e Hoey

```
DIVIDA (X, Y, a, p, r)
1  q ← ⌊(p + r)/2⌋
2  i ← p − 1     j ← q
3  para k ← p até r faça
4       se a[k] ≤ q
5            então  i ← i + 1
6                   b[i] ← a[k]
7            senão  j ← j + 1
8                   b[j] ← a[k]
9  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					

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

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				

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

Algoritmo de Shamos e Hoey

```
DIVIDA (X, Y, a, p, r)
1  q ← ⌊(p + r)/2⌋
2  i ← p − 1     j ← q
3  para k ← p até r faça
4       se a[k] ≤ q
5            então  i ← i + 1
6                   b[i] ← a[k]
7            senão  j ← j + 1
8                   b[j] ← a[k]
9  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			

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

Algoritmo de Shamos e Hoey

```
DIVIDA (X, Y, a, p, r)
1  q ← ⌊(p + r)/2⌋
2  i ← p − 1     j ← q
3  para k ← p até r faça
4       se a[k] ≤ q
5            então  i ← i + 1
6                   b[i] ← a[k]
7            senão  j ← j + 1
8                   b[j] ← a[k]
9  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	

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

Algoritmo de Shamos e Hoey

```
DIVIDA (X, Y, a, p, r)
1  q ← ⌊(p + r)/2⌋
2  i ← p − 1     j ← q
3  para k ← p até r faça
4       se a[k] ≤ q
5            então  i ← i + 1
6                   b[i] ← a[k]
7            senão  j ← j + 1
8                   b[j] ← a[k]
9  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

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

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

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

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

Consumo de tempo:

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

Algoritmo de Shamos e Hoey

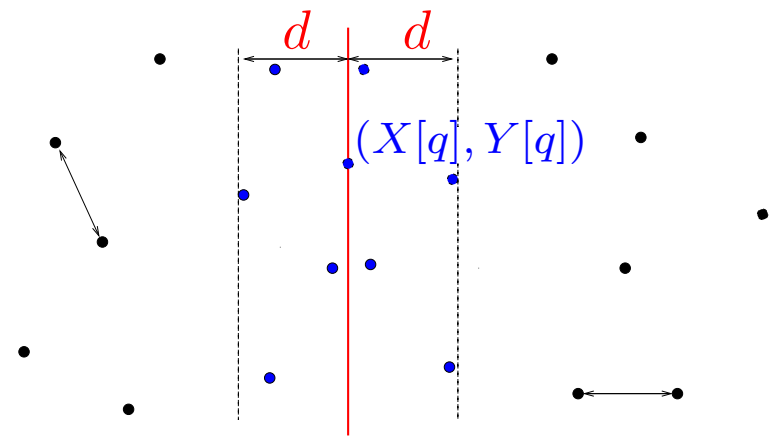
A rotina abaixo identifica os pontos que estão na faixa, ordenados pela Y-coordenada.

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 ← ⌊(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)
```



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					

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

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 ← ⌊(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	3	2	4

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

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 ← ⌊(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	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)
1  d ← min{d_E, d_D}
2  (f, t) ← CANDIDATOS (X, a, p, r, d)        ▷ pontos na faixa
3  para i ← 1 até t − 1 faça
4      para j ← i + 1 até min{i + 7, t} faça      ▷ ≤ 7 próximos
5          d′ ← DIST(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])
6          se d′ < d
7              então d ← d′
8  devolva d
```

Algoritmo de Shamos e Hoey

```
COMBINE (X, Y, a, p, r, d_E, d_D)
1  d ← min{d_E, d_D}
2  (f, t) ← CANDIDATOS (X, a, p, r, d)      ▷ pontos na faixa
3  para i ← 1 até t − 1 faça
4      para j ← i + 1 até min{i + 7, t} faça      ▷ ≤ 7 próximos
5          d′ ← DIST(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])
6          se d′ < d
7              então d ← d′
8  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 ← min{d_E, d_D}
(f, t) ← CANDIDATOS (X, a, p, r, d)        ▷ pontos na faixa
para i ← 1 até t − 1 faça
    j ← i + 1
    enquanto j ≤ t e Y[f[j]] − Y[f[i]] < d faça        ▷ ≤ 7 próximo
        d′ ← DIST(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])
        se d′ < d
            então d ← d′
        j ← j + 1
devolva d
```

Algoritmo de Shamos e Hoey

```
COMBINE (X, Y, a, p, r, d_E, d_D)
d ← min{d_E, d_D}
(f, t) ← CANDIDATOS (X, a, p, r, d)        ▷ pontos na faixa
para i ← 1 até t − 1 faça
    j ← i + 1
    enquanto j ≤ t e Y[f[j]] − Y[f[i]] < d faça        ▷ ≤ 7 próximo
        d′ ← DIST(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])
        se d′ < d
            então d ← d′
        j ← j + 1
devolva d
```

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

Um jeito mais bonito!

Recebe uma coleção de pontos dada por $X[p\,..\,r]$ e $Y[p\,..\,r]$, com $X[p] \leq X[p+1] \leq \cdots \leq X[r]$.

Um jeito mais bonito!

Recebe uma coleção de pontos dada por $X[p\,..\,r]$ e $Y[p\,..\,r]$, com $X[p] \leq X[p+1] \leq \cdots \leq X[r]$.

Devolve $X[p\,..\,r]$ e $Y[p\,..\,r]$ rearranjados de modo que $Y[p] \leq Y[p+1] \leq \cdots \leq Y[r]$ e retorna a menor distância entre dois pontos da coleção.

Um jeito mais bonito!

Recebe uma coleção de pontos dada por $X[p\,..\,r]$ e $Y[p\,..\,r]$, com $X[p] \leq X[p+1] \leq \cdots \leq X[r]$.

Devolve $X[p\,..\,r]$ e $Y[p\,..\,r]$ rearranjados de modo que $Y[p] \leq Y[p+1] \leq \cdots \leq Y[r]$ e retorna a menor distância entre dois pontos da coleção.

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

```
se r ≤ p + 2
    então ▷ resolva o problema diretamente
    senão q ← ⌊(p + r)/2⌋        xf ← X[q]
          d_E ← DISTÂNCIAREC-SH (X, Y, p, q)
          d_D ← DISTÂNCIAREC-SH (X, Y, q+1, r)
          INTERCALE (Y, X, p, q, r)    ▷ ordena por Y
          devolva COMBINE (X, Y, p, r, d_E, d_D, xf)
```

Algoritmo de Shamos e Hoey

Recebe $X[p\,..\,r]$ e $Y[p\,..\,r]$, com $X[p] \le X[p+1] \le \cdots \le X[r]$.

Devolve $X[p\,..\,r]$ e $Y[p\,..\,r]$ rearranjados de modo que $Y[p] \le Y[p+1] \le \cdots \le Y[r]$ e retorna a menor distância entre dois pontos da coleção.

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

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

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, $T(n) = \mathrm{O}(n \lg n)$.

Quase finalizando...

Exercício: Adapte o algoritmo COMBINE para a versão mais bonita do algoritmo.

Quase finalizando...

Exercício: Adapte o algoritmo COMBINE para a versão mais bonita do algoritmo.

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).

Quase finalizando...

Exercício: Adapte o algoritmo COMBINE para a versão mais bonita do algoritmo.

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?

Quase finalizando...

Exercício: Adapte o algoritmo COMBINE para a versão mais bonita do algoritmo.

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?

Tarefa 1: UVA 10245 - The Closest Pair Problem

Quase finalizando...

Exercício: Adapte o algoritmo COMBINE para a versão mais bonita do algoritmo.

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?

Tarefa 1: UVA 10245 - The Closest Pair Problem

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

Agora, às animações!