

Geometria Computacional

Cristina G. Fernandes

Departamento de Ciência da Computação do IME-USP

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

segundo semestre de 2014

Introdução

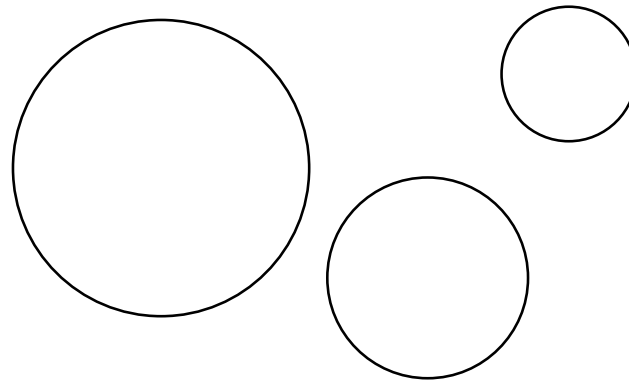
Antiguidade:

- construções geométricas de Euclides (régua e compasso)

Introdução

Antiguidade:

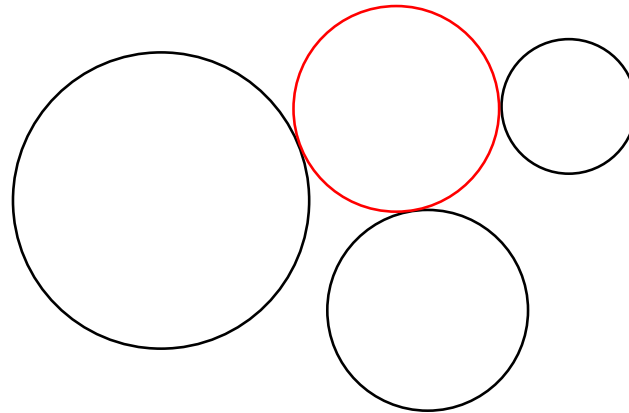
- construções geométricas de Euclides (régua e compasso)
- problema de Apollonius (cerca de 200 aC)



Introdução

Antiguidade:

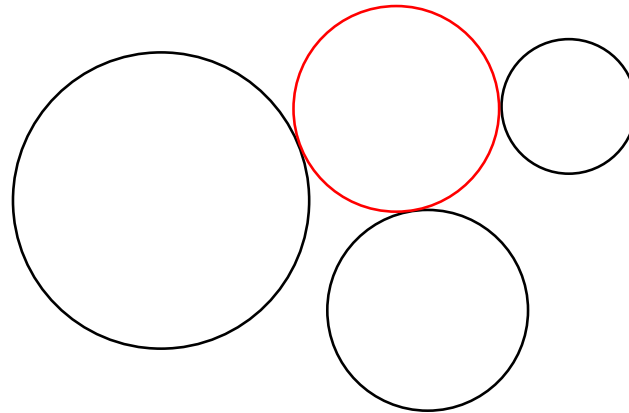
- construções geométricas de Euclides (régua e compasso)
- problema de Apollonius (cerca de 200 aC)



Introdução

Antiguidade:

- construções geométricas de Euclides (régua e compasso)
- problema de Apollonius (cerca de 200 aC)

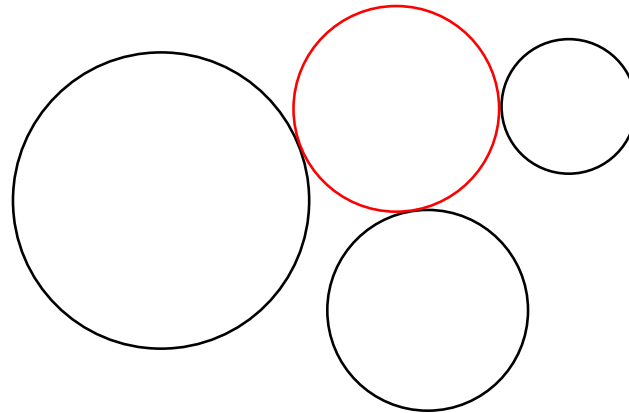


Solução de Euclides: 508 operações “elementares”

Introdução

Antiguidade:

- construções geométricas de Euclides (régua e compasso)
- problema de Apollonius (cerca de 200 aC)



Solução de Euclides: 508 operações “elementares”

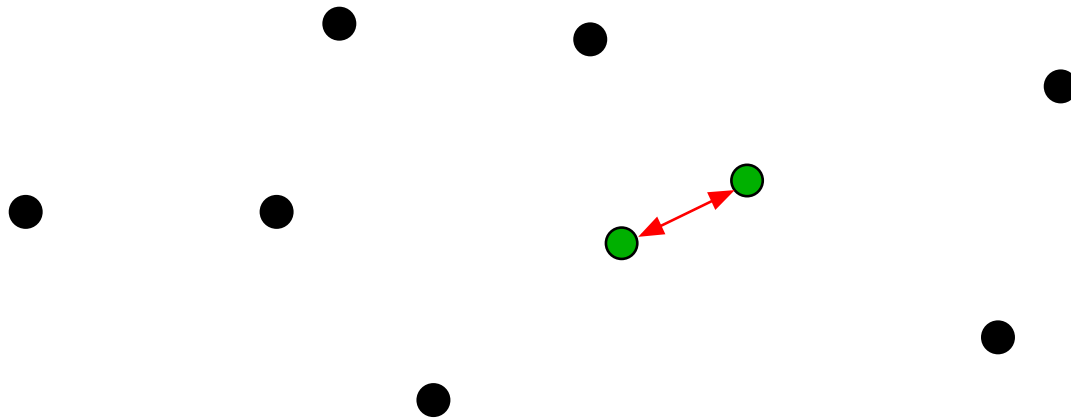
Solução de Lemoine (1902): menos de 200 operações

Par de pontos mais próximos

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

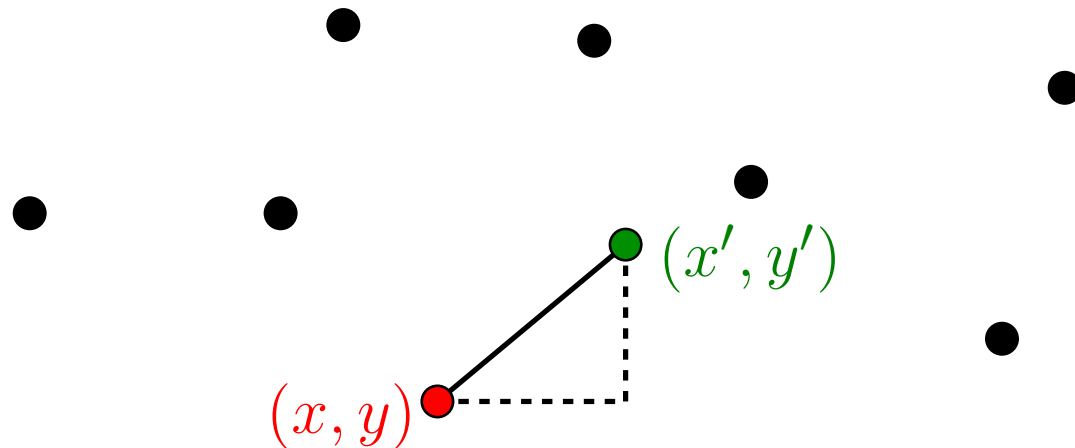
Par de pontos mais próximos

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.



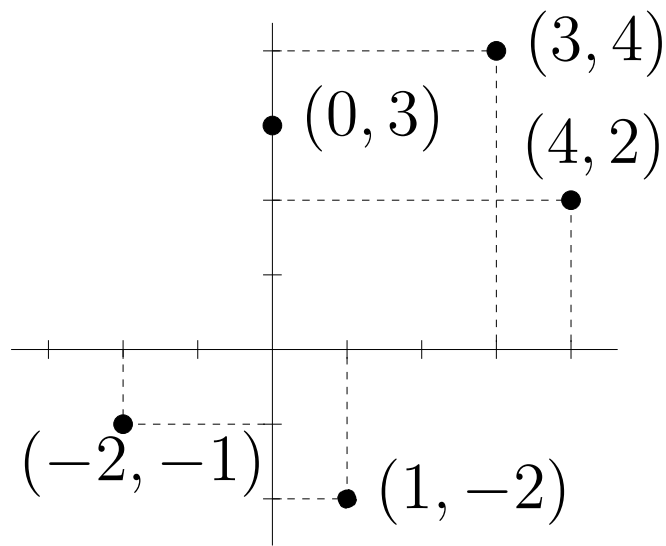
Lembre-se que, para dois pontos (x, y) e (x', y') no plano,

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

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 $Y[1..n]$.

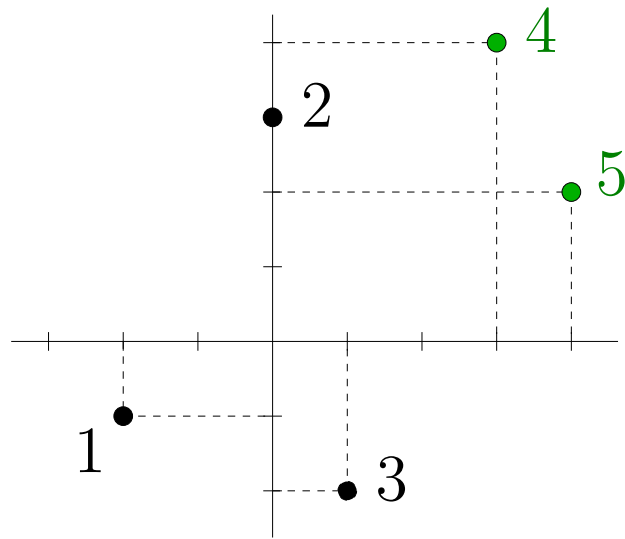


| | | | | | |
|-----|----|---|----|---|---|
| X | -2 | 0 | 1 | 3 | 4 |
| Y | -1 | 3 | -2 | 4 | 2 |
| | 1 | 2 | 3 | 4 | 5 |

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 $Y[1..n]$.



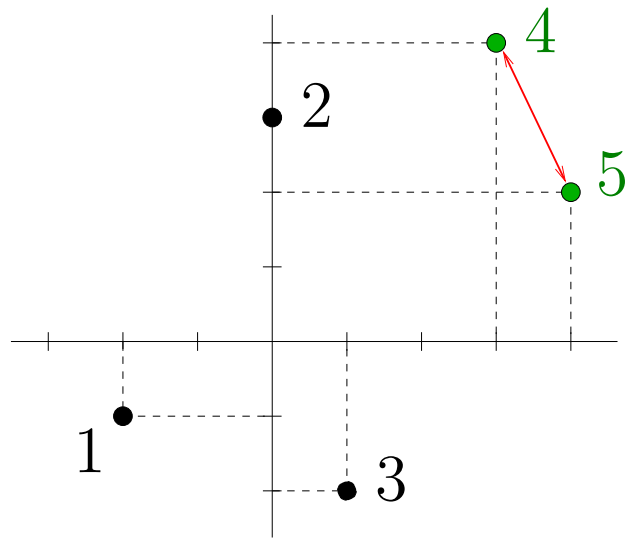
| | | | | | |
|-----|----|---|----|---|---|
| X | -2 | 0 | 1 | 3 | 4 |
| Y | -1 | 3 | -2 | 4 | 2 |
| | 1 | 2 | 3 | 4 | 5 |

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)

1 $d \leftarrow +\infty$

2 para $i \leftarrow 2$ até n faça

3 para $j \leftarrow 1$ até $i - 1$ faça

4 se $\text{DIST}(X[i], Y[i], X[j], Y[j]) < d$

5 então $d \leftarrow \text{DIST}(X[i], Y[i], X[j], Y[j])$

6 devolva d

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 $\text{DIST}(X[i], Y[i], X[j], Y[j]) < d$

5 então $d \leftarrow \text{DIST}(X[i], Y[i], X[j], Y[j])$

6 devolva d

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 $\text{DIST}(X[i], Y[i], X[j], Y[j]) < d$

5 então $d \leftarrow \text{DIST}(X[i], Y[i], X[j], Y[j])$

6 devolva d

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

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 $\text{DIST}(X[i], Y[i], X[j], Y[j]) < d$
- 5 então $d \leftarrow \text{DIST}(X[i], Y[i], X[j], Y[j])$
- 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

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

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 $\text{DIST}(X[i], Y[i], X[j], Y[j]) < d$

5 então $d \leftarrow \text{DIST}(X[i], Y[i], X[j], Y[j])$

6 devolva d

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

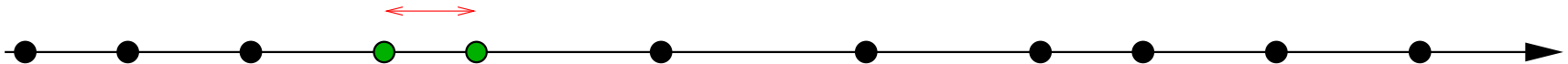
Consumo de tempo: linha 4 é executada

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

É possível projetar um algoritmo mais eficiente que este?

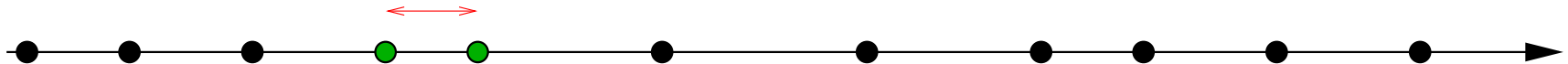
Par mais próximo na reta

Problema: Dados n pontos numa reta, determinar dois deles que estão à distância mínima.



Par mais próximo na reta

Problema: Dados n pontos numa reta, determinar dois deles que estão à distância mínima.

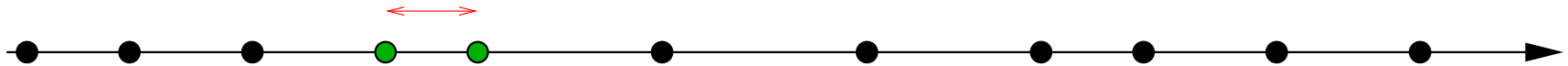


Primeira solução: ordene os pontos, e encontre os dois consecutivos mais próximos.

Tempo consumido: $O(n \lg n)$.

Par mais próximo na reta

Problema: Dados n pontos numa reta, determinar dois deles que estão à distância mínima.



Primeira solução: ordene os pontos, e encontre os dois consecutivos mais próximos.

Tempo consumido: $O(n \lg n)$.

Problema com essa solução:

não sei como generalizá-la para o plano...

Divisão e conquista

Esse paradigma envolve os seguintes passos:

Divisão e conquista

Esse paradigma envolve os seguintes passos:

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

Divisão e conquista

Esse paradigma envolve os seguintes passos:

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).

Divisão e conquista

Esse paradigma envolve os seguintes passos:

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.

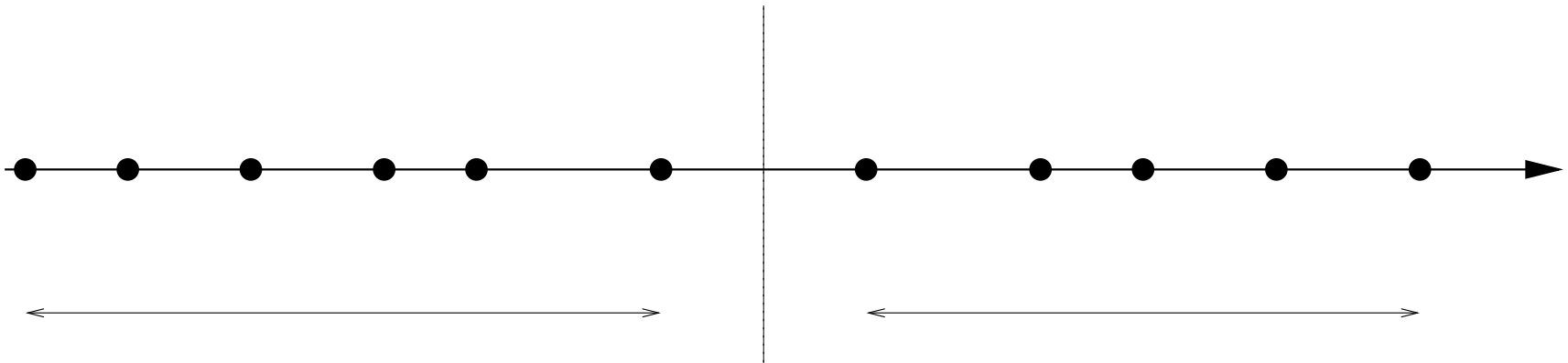
Divisão e conquista

Esse paradigma envolve os seguintes passos:

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.



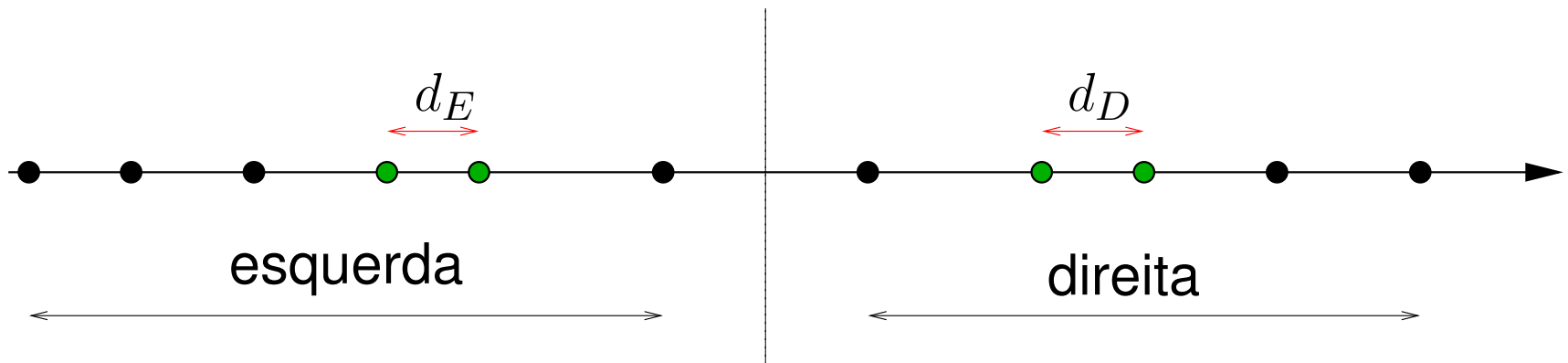
Divisão e conquista

Esse paradigma envolve os seguintes passos:

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.



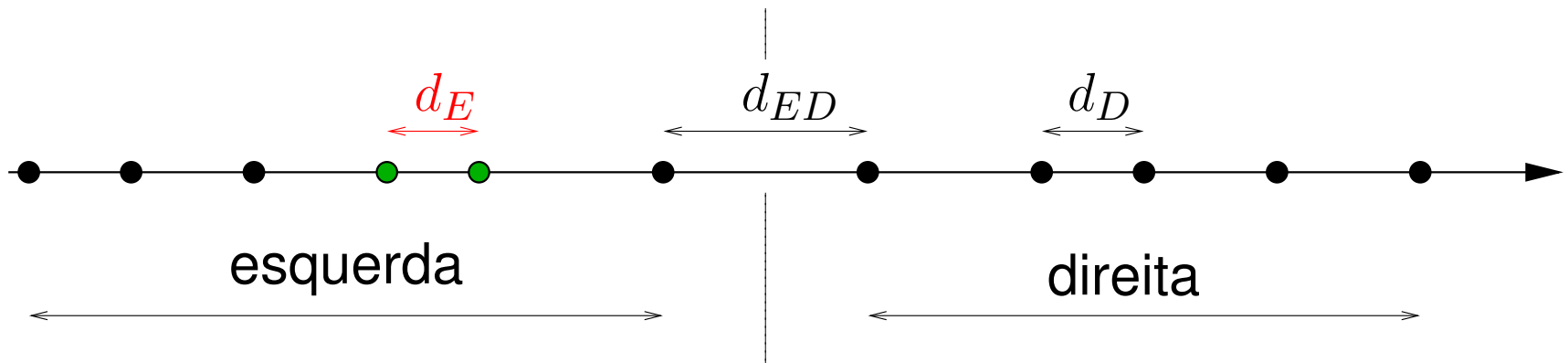
Divisão e conquista

Esse paradigma envolve os seguintes passos:

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

Pré-processamento: ordenar os pontos.

Par mais próximo na reta

Pré-processamento: ordenar os pontos.

$\text{DISTÂNCIARETA}(X, n)$

1 MERGESORT($X, 1, n$)

2 devolva $\text{DISTÂNCIARETA REC}(X, 1, n)$

Par mais próximo na reta

Pré-processamento: ordenar os pontos.

$\text{DISTÂNCIARETA}(X, n)$

1 MERGESORT($X, 1, n$)

2 devolva $\text{DISTÂNCIARETA REC}(X, 1, n)$

DISTÂNCIARETA REC : divisão e conquista.

Par mais próximo na reta

Pré-processamento: ordenar os pontos.

$\text{DISTÂNCIARETA}(X, n)$

1 MERGESORT($X, 1, n$)

2 devolva $\text{DISTÂNCIARETAREC}(X, 1, n)$

DISTÂNCIARETAREC : divisão e conquista.

Tempo consumido pelo DISTÂNCIARETA :

$O(n \lg n)$ mais o tempo do DISTÂNCIARETAREC .

Par mais próximo na reta

DISTÂNCIARETAREC (X, p, r) ▷ Divisão e conquista

```
1  se  $r \leq p + 1$ 
2    então se  $r = p$ 
3        então devolva  $+\infty$ 
4        senão devolva  $X[r] - X[p]$ 
5  senão  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
6         $d_E \leftarrow$  DISTÂNCIARETAREC( $X, p, q$ )
7         $d_D \leftarrow$  DISTÂNCIARETAREC( $X, q + 1, r$ )
8         $d \leftarrow \min\{d_E, d_D, X[q+1] - X[q]\}$ 
9  devolva  $d$ 
```

Par mais próximo na reta

DISTÂNCIARETAREC (X, p, r) ▷ Divisão e conquista

```
1  se  $r \leq p + 1$ 
2    então se  $r = p$ 
3        então devolva  $+\infty$ 
4        senão devolva  $X[r] - X[p]$ 
5  senão  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
6         $d_E \leftarrow$  DISTÂNCIARETAREC( $X, p, q$ )
7         $d_D \leftarrow$  DISTÂNCIARETAREC( $X, q + 1, r$ )
8         $d \leftarrow \min\{d_E, d_D, X[q+1] - X[q]\}$ 
9  devolva  $d$ 
```

Consumo de tempo:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(1)$$

onde $n = r - p + 1$.

Par mais próximo na reta

DISTÂNCIARETAREC (X, p, r) ▷ Divisão e conquista

```
1  se  $r \leq p + 1$ 
2    então se  $r = p$ 
3        então devolva  $+\infty$ 
4        senão devolva  $X[r] - X[p]$ 
5  senão  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
6         $d_E \leftarrow \text{DISTÂNCIARETAREC}(X, p, q)$ 
7         $d_D \leftarrow \text{DISTÂNCIARETAREC}(X, q + 1, r)$ 
8         $d \leftarrow \min\{d_E, d_D, X[q+1] - X[q]\}$ 
9  devolva  $d$ 
```

Consumo de tempo:

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(1)$$

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

Par mais próximo na reta

DISTÂNCIARETAREC (X, p, r) ▷ Divisão e conquista

```
1  se  $r \leq p + 1$ 
2    então se  $r = p$ 
3        então devolva  $+\infty$ 
4        senão devolva  $X[r] - X[p]$ 
5  senão  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
6         $d_E \leftarrow$  DISTÂNCIARETAREC( $X, p, q$ )
7         $d_D \leftarrow$  DISTÂNCIARETAREC( $X, q + 1, r$ )
8         $d \leftarrow \min\{d_E, d_D, X[q+1] - X[q]\}$ 
9  devolva  $d$ 
```

Consumo de tempo:

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(1)$$

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

Par mais próximo na reta

Voltando...

$\text{DIST\^A}NCIA\text{RETA}(X, n)$

1 $\text{MERGESORT}(X, 1, n)$

2 devolva $\text{DIST\^A}NCIA\text{RETA}\text{REC}(X, 1, n)$

Par mais próximo na reta

Voltando...

$\text{DIST\^A}NCIARETA(X, n)$

1 $\text{MERGESORT}(X, 1, n)$

2 devolva $\text{DIST\^A}NCIARETAREC(X, 1, n)$

MERGESORT consome tempo $O(n \lg n)$.

$\text{DIST\^A}NCIARETAREC$ consome tempo $\Theta(n)$.

Par mais próximo na reta

Voltando...

$\text{DISTÂNCIARETA}(X, n)$

1 $\text{MERGESORT}(X, 1, n)$

2 devolva $\text{DISTÂNCIARETA REC}(X, 1, n)$

MERGESORT consome tempo $O(n \lg n)$.

DISTÂNCIARETA REC consome tempo $\Theta(n)$.

Tempo consumido pelo DISTÂNCIARETA : $O(n \lg n)$.

Par mais próximo no plano

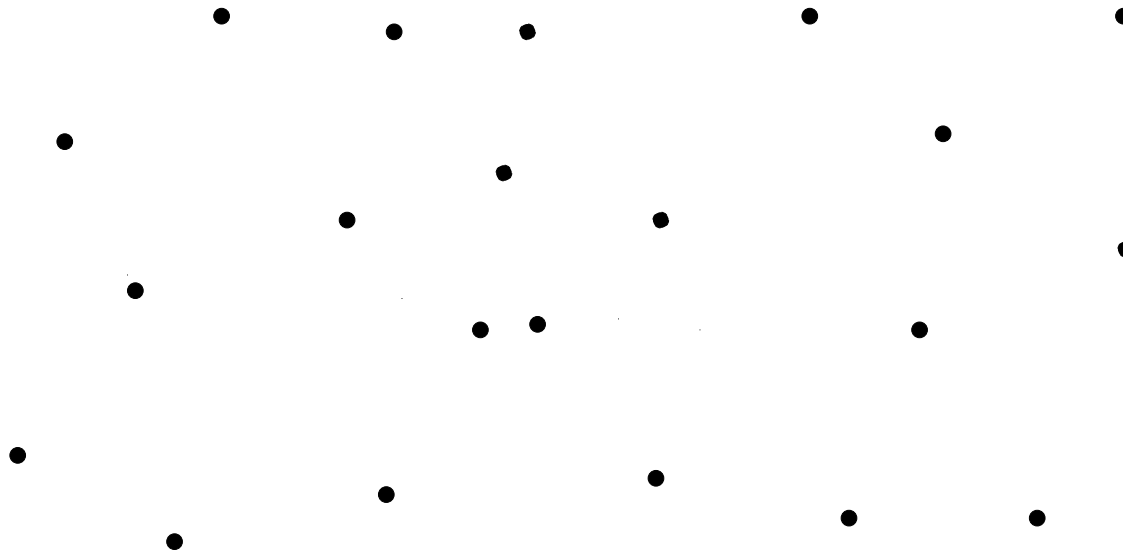
Obtivemos um algoritmo $O(n \lg n)$ para pontos na reta.

Como generalizar essa ideia para o plano?

Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

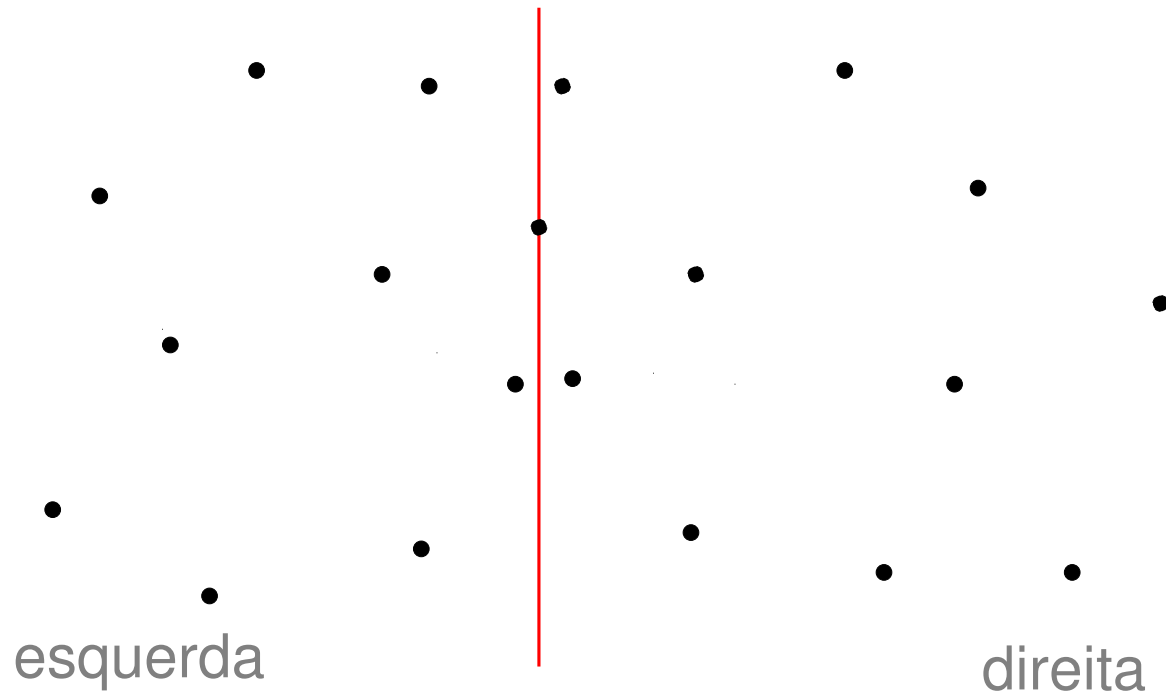


Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

Divide...

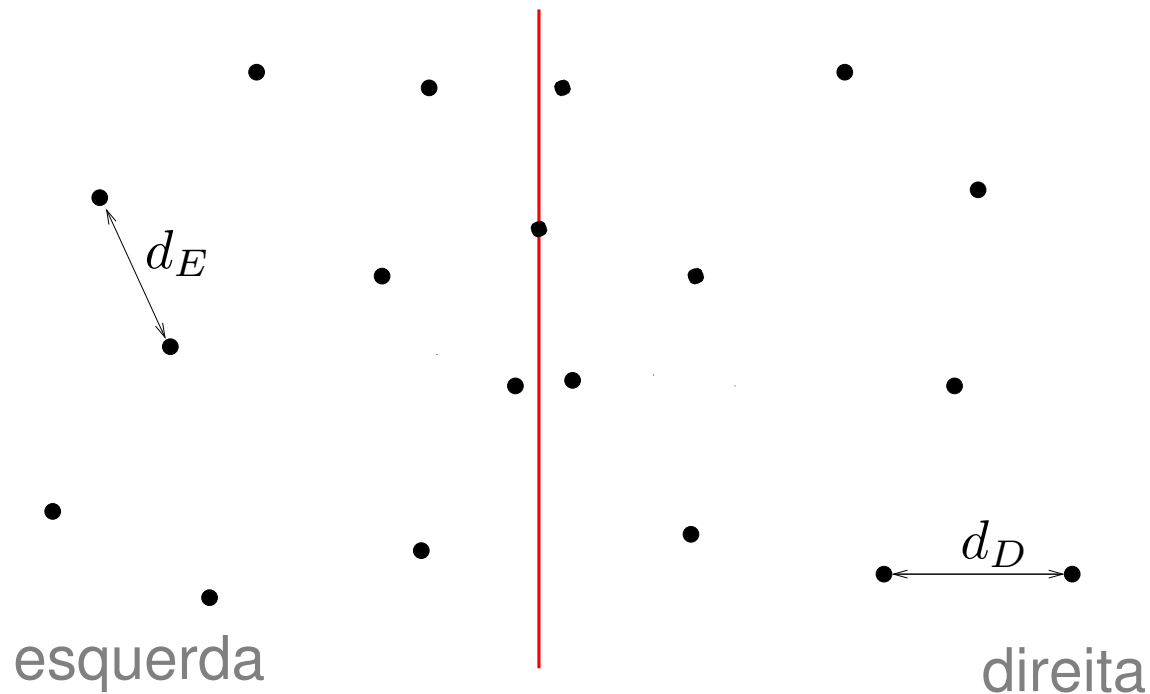


Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

Divide... Conquista...

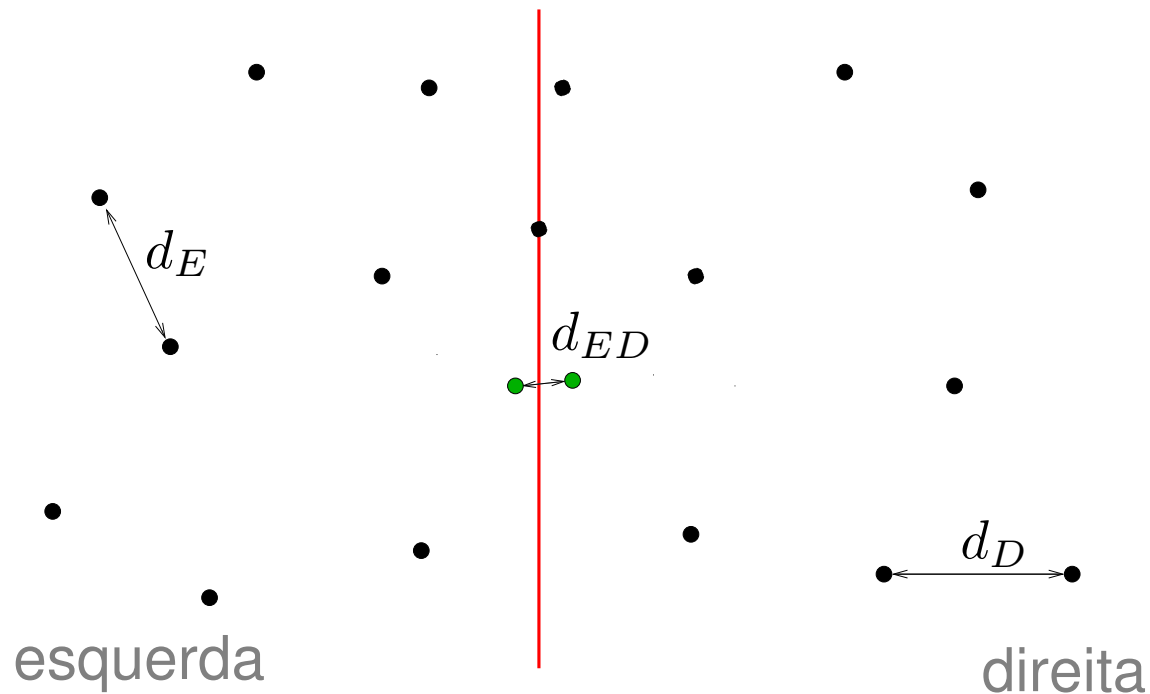


Par mais próximo no plano

Obtivemos um algoritmo $O(n \lg n)$ 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

$\text{DIST\^A}NCIA\text{-SH}(X, Y, n)$

1 MERGESORT($X, Y, 1, n$)

2 devolva $\text{DIST\^A}NCIA\text{REC-SH}(X, Y, 1, n)$

Consumo de tempo:

$O(n \lg n)$ mais o tempo do $\text{DIST\^A}NCIA\text{REC-SH}$.

Divisão e conquista

DISTÂNCIA REC-SH (X, Y, p, r)

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

Divisão e conquista

DISTÂNCIA REC-SH (X, Y, p, r)

Dividir: $X[p \dots q], Y[p \dots q]$ (esquerda)
 $X[q+1 \dots r], Y[q+1 \dots r]$ (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

DISTÂNCIA REC-SH (X, Y, p, r)

Dividir: $X[p \dots q], Y[p \dots q]$ (esquerda)
 $X[q+1 \dots r], Y[q+1 \dots r]$ (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 $r \leq p + 2$

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)

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.

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:

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

onde $n = r - p + 1$.

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:

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

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

Algoritmo de Shamos e Hoey

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

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:

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

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

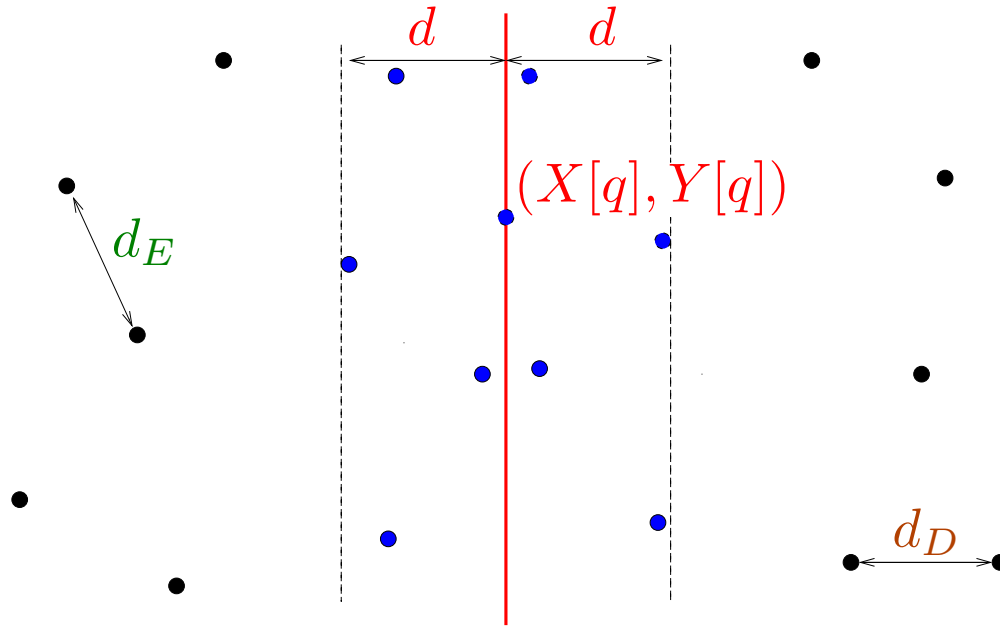
Algoritmo de Shamos e Hoey

Como fazer o **COMBINE** linear?

Algoritmo de Shamos e Hoey

Como fazer o **COMBINE** linear?

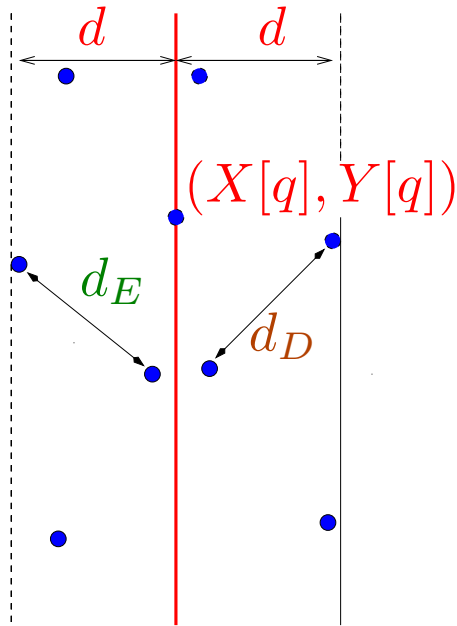
COMBINE precisa considerar apenas pontos que estão a uma distância menor que $d = \min\{d_E, d_D\}$ da reta vertical $x = X[q]$.



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...

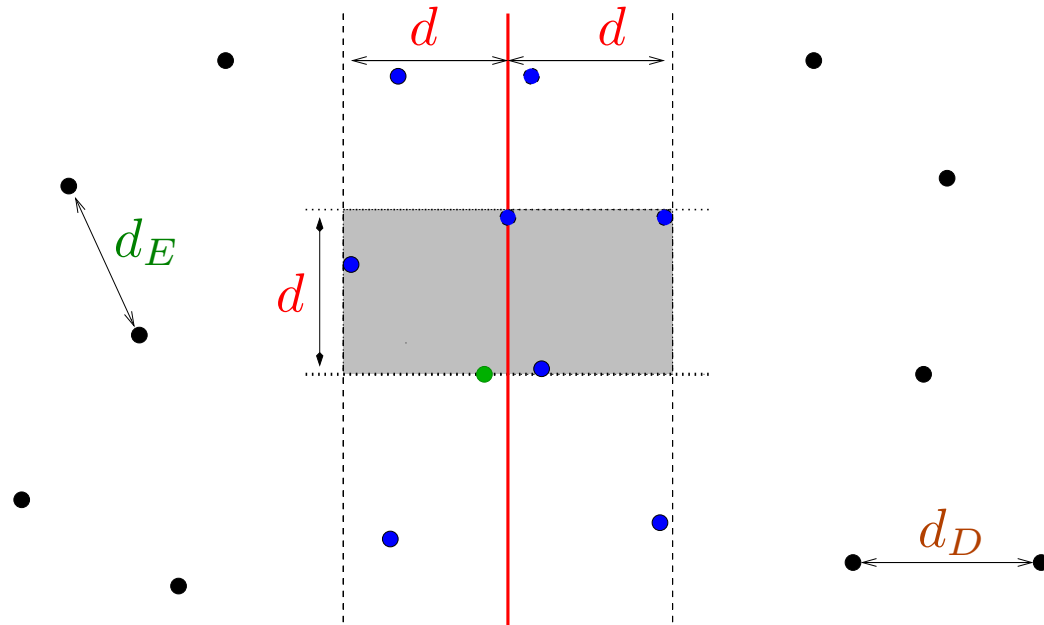
Algoritmo de Shamos e Hoey

Como fazer o **COMBINE** linear?

Algoritmo de Shamos e Hoey

Como fazer o **COMBINE** linear?

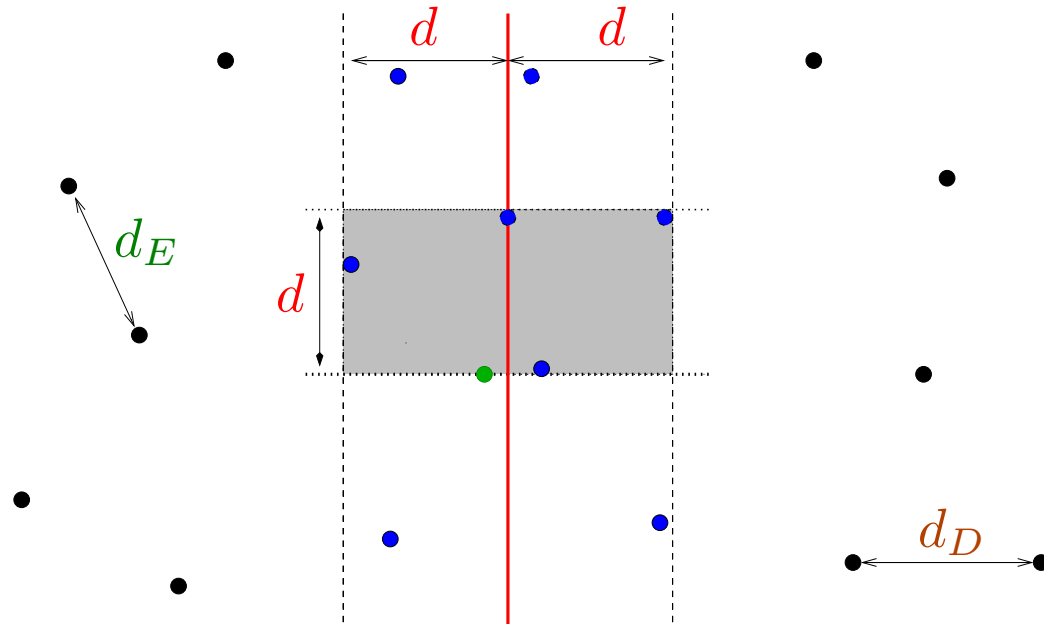
Ideia...



Algoritmo de Shamos e Hoey

Como fazer o **COMBINE** linear?

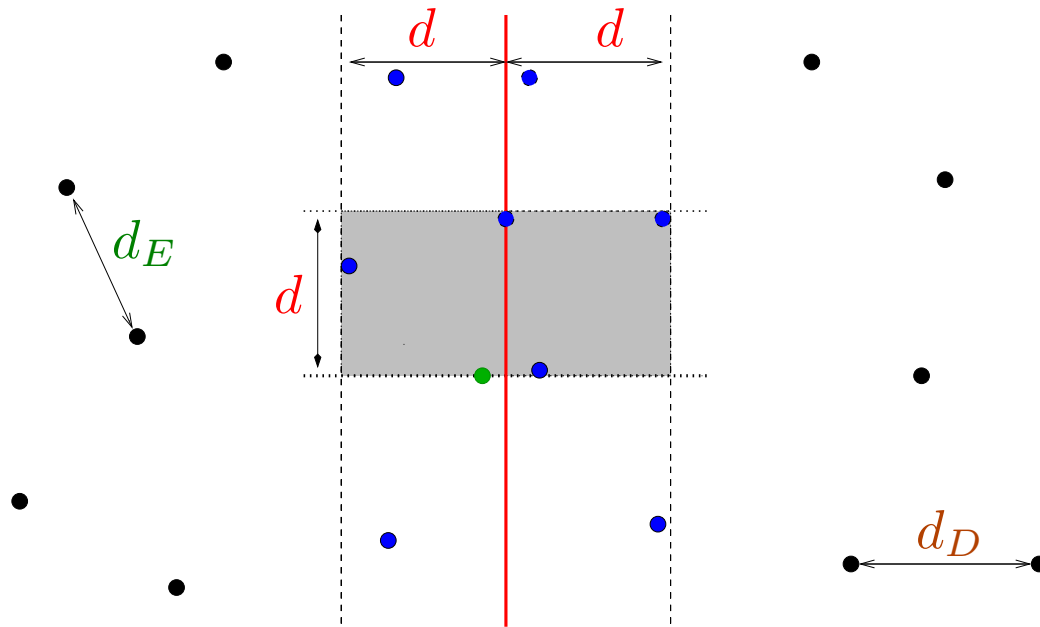
Ideia...



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

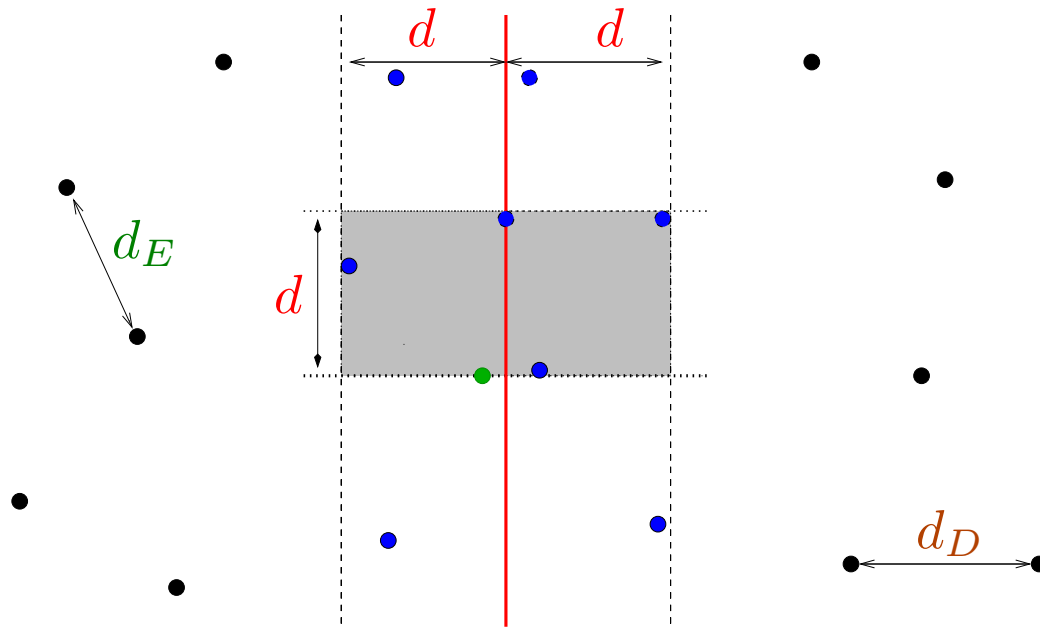
Como fazer o **COMBINE** linear?



Quantos pontos assim há?

Algoritmo de Shamos e Hoey

Como fazer o **COMBINE** linear?

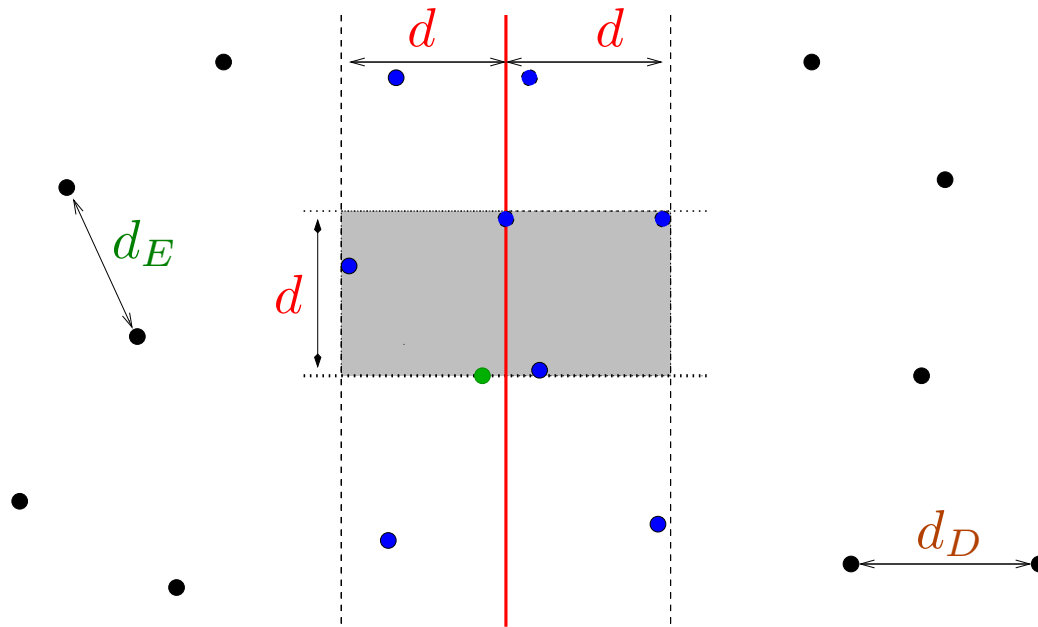


Quantos pontos assim há?

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?



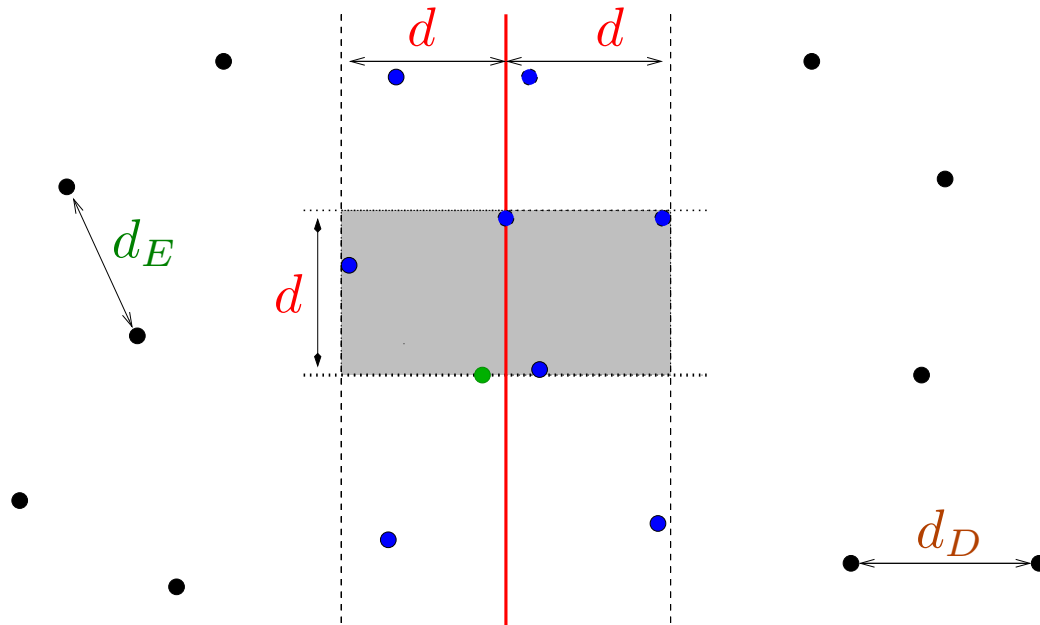
Quantos pontos assim há?

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

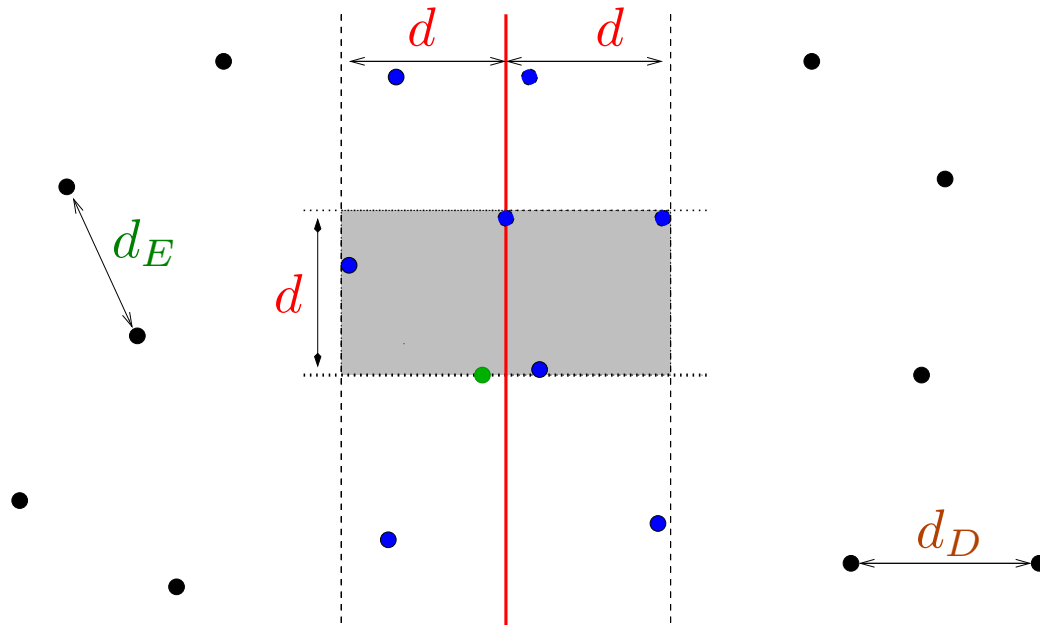
Como fazer o **COMBINE** linear?



Mas como ter acesso rápido a estes pontos?

Algoritmo de Shamos e Hoey

Como fazer o **COMBINE** linear?

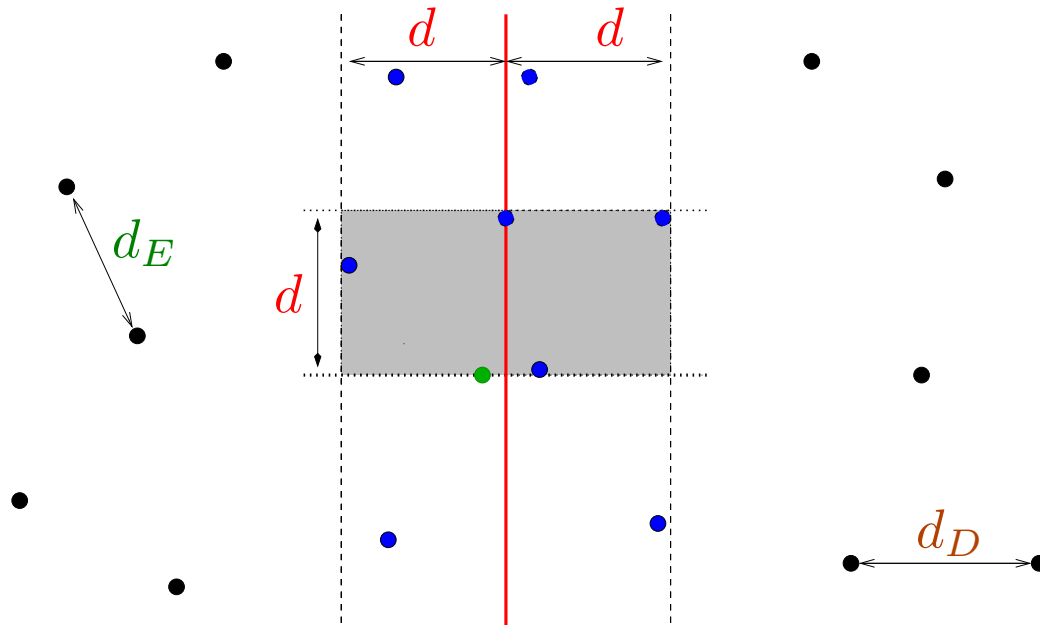


Mas como ter acesso rápido a estes pontos?

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 $i \leftarrow 1$ até n faça

3 $a[i] \leftarrow i$

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 $a[i] \leftarrow i$

4 MERGESORTIND($Y, 1, n, a$) ▷ ordenação indireta

5 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

DISTÂNCIA REC-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 $r \leq p + 2$

2 então ▷ resolva o problema diretamente

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

4 $b \leftarrow \text{DIVIDA}(X, Y, a, p, r)$

5 $d_E \leftarrow \text{DISTÂNCIAREC-SH}(X, Y, b, p, q)$

6 $d_D \leftarrow \text{DISTÂNCIAREC-SH}(X, Y, b, q + 1, r)$

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

1 se $r \leq p + 2$

2 então ▷ resolva o problema diretamente

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

4 $b \leftarrow$ **DIVIDA** (X, Y, a, p, r)

5 $d_E \leftarrow$ **DISTÂNCIAREC-SH** (X, Y, b, p, q)

6 $d_D \leftarrow$ **DISTÂNCIAREC-SH** ($X, Y, b, q + 1, r$)

7 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

1 se $r \leq p + 2$

2 então \triangleright resolva o problema diretamente

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

4 $b \leftarrow$ **DIVIDA** (X, Y, a, p, r)

5 $d_E \leftarrow$ **DISTÂNCIAREC-SH** (X, Y, b, p, q)

6 $d_D \leftarrow$ **DISTÂNCIAREC-SH** ($X, Y, b, q + 1, r$)

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

DIVIDA e **COMBINE** são algoritmos lineares.

Consumo de tempo:

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

onde $n = r - p + 1$.

Algoritmo de Shamos e Hoey

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

1 se $r \leq p + 2$

2 então \triangleright resolva o problema diretamente

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

4 $b \leftarrow$ **DIVIDA** (X, Y, a, p, r)

5 $d_E \leftarrow$ **DISTÂNCIAREC-SH** (X, Y, b, p, q)

6 $d_D \leftarrow$ **DISTÂNCIAREC-SH** ($X, Y, b, q + 1, r$)

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

DIVIDA e **COMBINE** são algoritmos lineares.

Consumo de tempo:

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

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

Algoritmo de Shamos e Hoey

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

1 $q \leftarrow \lfloor (p + r) / 2 \rfloor$

2 $i \leftarrow p - 1$ $j \leftarrow q$

3 para $k \leftarrow p$ até r faça

4 se $a[k] \leq q$ $\triangleright (X[a[k]], Y[a[k]])$ está à esquerda da reta $x = X[q]$?

5 então $i \leftarrow i + 1$

6 $b[i] \leftarrow a[k]$

7 senão $j \leftarrow j + 1$

8 $b[j] \leftarrow a[k]$

9 devolva b

Algoritmo de Shamos e Hoey

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

```
1  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2  $i \leftarrow p - 1$     $j \leftarrow q$ 
3 para  $k \leftarrow p$  até  $r$  faça
4     se  $a[k] \leq q$ 
5         então  $i \leftarrow i + 1$ 
6              $b[i] \leftarrow a[k]$ 
7     senão  $j \leftarrow j + 1$ 
8          $b[j] \leftarrow a[k]$ 
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 | | | | | |

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

Algoritmo de Shamos e Hoey

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

```
1  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2  $i \leftarrow p - 1$      $j \leftarrow q$ 
3 para  $k \leftarrow p$  até  $r$  faça
4     se  $a[k] \leq q$ 
5         então  $i \leftarrow i + 1$ 
6              $b[i] \leftarrow a[k]$ 
7     senão  $j \leftarrow j + 1$ 
8          $b[j] \leftarrow a[k]$ 
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 | | | | |

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

Algoritmo de Shamos e Hoey

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

```
1   $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2   $i \leftarrow p - 1$     $j \leftarrow q$ 
3  para  $k \leftarrow p$  até  $r$  faça
4      se  $a[k] \leq q$ 
5          então  $i \leftarrow i + 1$ 
6               $b[i] \leftarrow a[k]$ 
7      senão  $j \leftarrow j + 1$ 
8               $b[j] \leftarrow a[k]$ 
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 | | | |

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

Algoritmo de Shamos e Hoey

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

```
1   $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2   $i \leftarrow p - 1$     $j \leftarrow q$ 
3  para  $k \leftarrow p$  até  $r$  faça
4      se  $a[k] \leq q$ 
5          então  $i \leftarrow i + 1$ 
6               $b[i] \leftarrow a[k]$ 
7      senão  $j \leftarrow j + 1$ 
8               $b[j] \leftarrow a[k]$ 
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 | |

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

Algoritmo de Shamos e Hoey

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

```
1   $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2   $i \leftarrow p - 1$     $j \leftarrow q$ 
3  para  $k \leftarrow p$  até  $r$  faça
4      se  $a[k] \leq q$ 
5          então  $i \leftarrow i + 1$ 
6               $b[i] \leftarrow a[k]$ 
7      senão  $j \leftarrow j + 1$ 
8               $b[j] \leftarrow a[k]$ 
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 |

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

Algoritmo de Shamos e Hoey

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

```
1   $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2   $i \leftarrow p - 1$     $j \leftarrow q$ 
3  para  $k \leftarrow p$  até  $r$  faça
4      se  $a[k] \leq q$ 
5          então  $i \leftarrow i + 1$ 
6               $b[i] \leftarrow a[k]$ 
7      senão  $j \leftarrow j + 1$ 
8               $b[j] \leftarrow a[k]$ 
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 |

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

Consumo de tempo:

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

Algoritmo de Shamos e Hoey

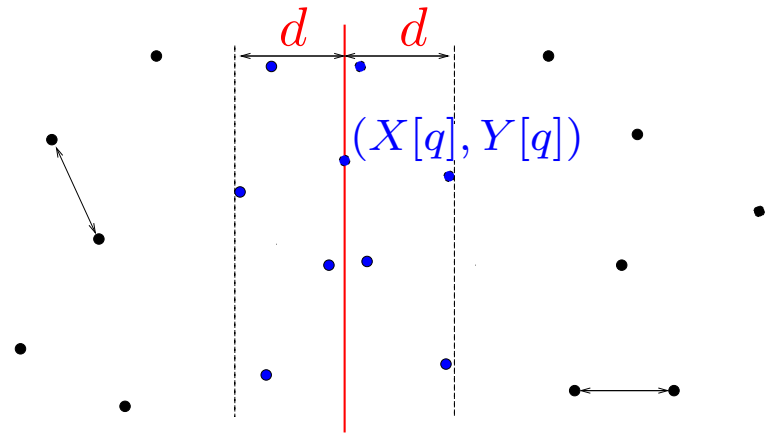
A rotina abaixo identifica os pontos que estão na faixa, ordenados pela Y -coordenada.

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)

- 1 $q \leftarrow \lfloor (p + r) / 2 \rfloor$
- 2 $t \leftarrow 0$
- 3 para $k \leftarrow p$ até r faça
- 4 se $|X[a[k]] - X[q]| < d$
- 5 então $t \leftarrow t + 1$
- 6 $f[t] \leftarrow a[k]$
- 7 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)

```
1  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2  $t \leftarrow 0$ 
3 para  $k \leftarrow p$  até  $r$  faça
4     se  $|X[a[k]] - X[q]| < d$ 
5         então  $t \leftarrow t + 1$ 
6          $f[t] \leftarrow a[k]$ 
7 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)

```
1  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2  $t \leftarrow 0$ 
3 para  $k \leftarrow p$  até  $r$  faça
4     se  $|X[a[k]] - X[q]| < d$ 
5         então  $t \leftarrow t + 1$ 
6              $f[t] \leftarrow a[k]$ 
7 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)

```
1  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
2  $t \leftarrow 0$ 
3 para  $k \leftarrow p$  até  $r$  faça
4     se  $|X[a[k]] - X[q]| < d$ 
5         então  $t \leftarrow t + 1$ 
6              $f[t] \leftarrow a[k]$ 
7 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 $d \leftarrow \min\{d_E, d_D\}$

2 $(f, t) \leftarrow \text{CANDIDATOS}(X, a, p, r, d)$ \triangleright pontos na faixa

3 para $i \leftarrow 1$ até $t - 1$ faça

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

5 $d' \leftarrow \text{DIST}(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])$

6 se $d' < d$

7 então $d \leftarrow d'$

8 devolva d

Algoritmo de Shamos e Hoey

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

1 $d \leftarrow \min\{d_E, d_D\}$

2 $(f, t) \leftarrow \text{CANDIDATOS}(X, a, p, r, d)$ \triangleright pontos na faixa

3 para $i \leftarrow 1$ até $t - 1$ faça

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

5 $d' \leftarrow \text{DIST}(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])$

6 se $d' < d$

7 então $d \leftarrow d'$

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

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

Algoritmo de Shamos e Hoey

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

```
1   $d \leftarrow \min\{d_E, d_D\}$ 
2   $(f, t) \leftarrow \text{CANDIDATOS}(X, a, p, r, d)$   $\triangleright$  pontos na faixa
3  para  $i \leftarrow 1$  até  $t - 1$  faça
4       $j \leftarrow i + 1$ 
5      enquanto  $j \leq t$  e  $Y[f[j]] - Y[f[i]] < d$  faça  $\triangleright \leq 7$  próximos
6           $d' \leftarrow \text{DIST}(X[f[i]], Y[f[i]], X[f[j]], Y[f[j]])$ 
7          se  $d' < d$ 
8              então  $d \leftarrow d'$ 
9               $j \leftarrow j + 1$ 
10 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 $X[p] \leq X[p+1] \leq \dots \leq X[r]$.

Um jeito mais bonito!

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

Devolve $X[p..r]$ e $Y[p..r]$ rearranjados de modo que $Y[p] \leq Y[p+1] \leq \dots \leq Y[r]$ 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 $X[p] \leq X[p+1] \leq \dots \leq X[r]$.

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

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 **INTERCALE** (Y, X, p, q, r) ▷ ordena por Y
- 7 devolva **COMBINE** (X, Y, p, r, d_E, d_D)

Algoritmo de Shamos e Hoey

Recebe $X[p..r]$ e $Y[p..r]$, com $X[p] \leq X[p+1] \leq \dots \leq X[r]$.

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

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          INTERCALE ( $Y, X, p, q, r$ )
7          devolva COMBINE ( $X, Y, p, r, d_E, d_D$ )
```

Consumo de tempo: $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$,
onde $n = r - p + 1$. Como antes, $T(n) = O(n \lg n)$.

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!**