

Tópicos de Análise de Algoritmos

Parte destes slides são adaptações de slides
do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

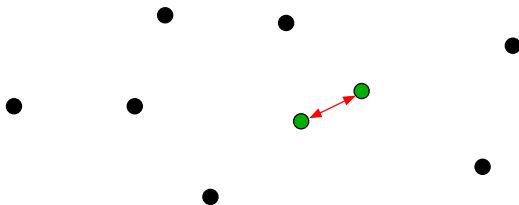
Divisão e conquista

Exemplo 1: Número de inversões de uma permutação
(problema 2-4 do CLRS; veja também sec 5.3 do KT)

Exemplo 2: Par de pontos mais próximos
(sec 33.4 do CLRS; veja também sec 5.4 do KT)

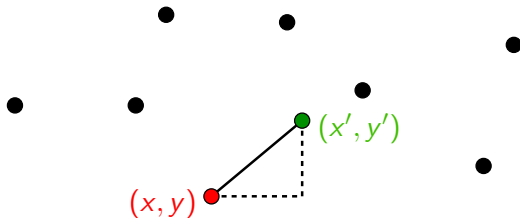
Par de pontos mais próximos

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.



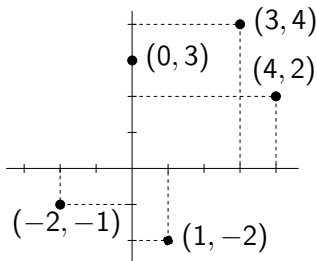
Lembre-se que, para dois pontos (x, y) e (x', y') no plano,

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

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 $Y[1..n]$.

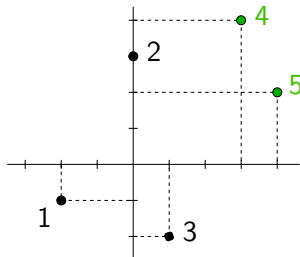


X	-2	0	1	3	4
Y	-1	3	-2	4	2
	1	2	3	4	5

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 $Y[1..n]$.



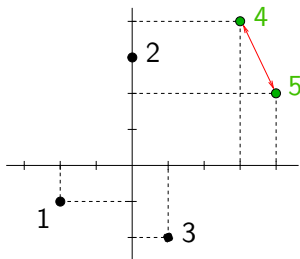
X	-2	0	1	3	4
Y	-1	3	-2	4	2
	1	2	3	4	5

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 $X[1 \dots i - 1], Y[1 \dots i - 1]$.

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.

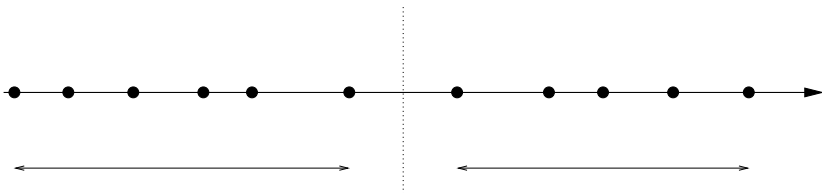
Divisão e conquista

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.



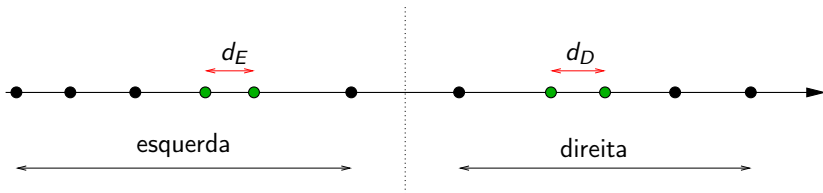
Divisão e conquista

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.



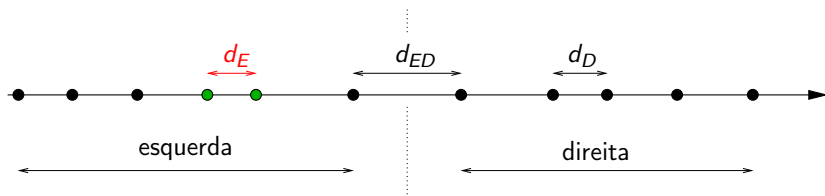
Divisão e conquista

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.

DistânciaReta(X, n)

- 1 MERGESORT($X, 1, n$)
- 2 devolva DistânciaRetaRec($X, 1, n$)

Par mais próximo na reta

Pré-processamento: ordenar os pontos.

$\text{DistânciaReta}(X, n)$

- 1 **MERGESORT** $(X, 1, n)$
- 2 devolva **DistânciaRetaRec** $(X, 1, n)$

DistânciaRetaRec: divisão e conquista.

Par mais próximo na reta

Pré-processamento: ordenar os pontos.

$\text{DistânciaReta}(X, n)$

- 1 **MERGESORT** $(X, 1, n)$
- 2 devolva **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       $d_E \leftarrow \text{DistânciaRetaRec}(X, p, q)$ 
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  $d$ 
```

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 $n = r - p + 1$.

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       $d_E \leftarrow \text{DistânciaRetaRec}(X, p, q)$ 
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 $T(n)$? $T(n) = \Theta(n)$.

Par mais próximo na reta

Voltando...

DistânciaReta(X, n)

- 1 MERGESORT($X, 1, n$)
- 2 devolva DistânciaRetaRec ($X, 1, n$)

Par mais próximo na reta

Voltando...

DistânciaReta(X, n)

- 1 MERGESORT($X, 1, n$)
- 2 devolva DistânciaRetaRec ($X, 1, n$)

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

DistânciaRetaRec consome tempo $\Theta(n)$.

Par mais próximo na reta

Voltando...

DistânciaReta(X, n)

- 1 MERGESORT($X, 1, n$)
- 2 devolva DistânciaRetaRec ($X, 1, n$)

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

DistânciaRetaRec consome tempo $\Theta(n)$.

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

Par mais próximo no plano

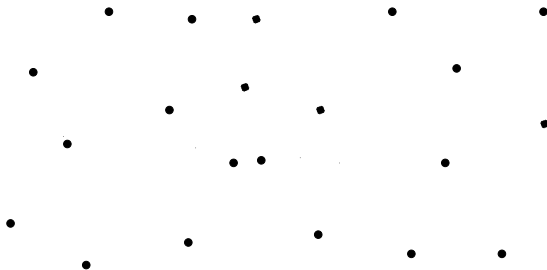
Obtivemos um algoritmo $O(n \lg n)$ para pontos na reta.

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?

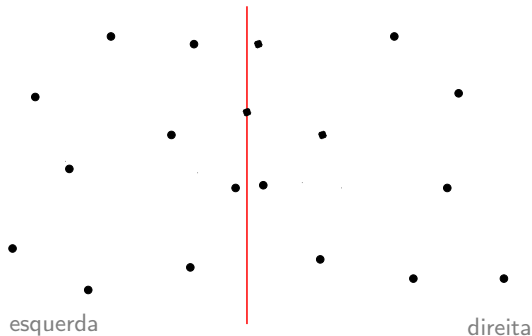


Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

Divide...

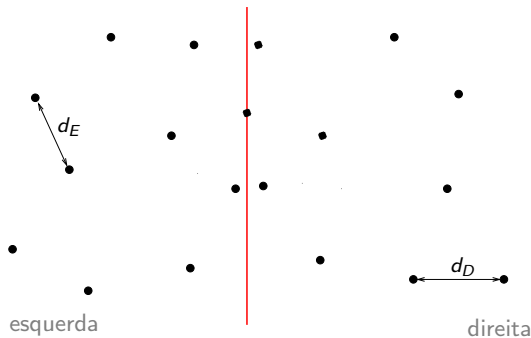


Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

Divide... Conquista...

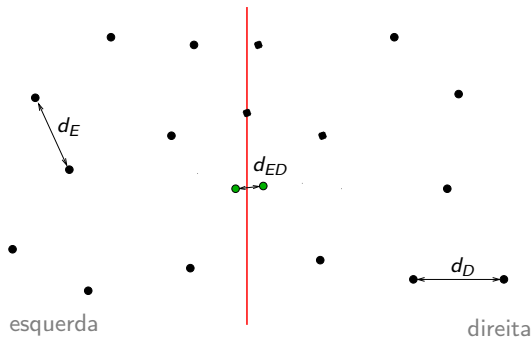


Par mais próximo no plano

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

Distância-SH(X, Y, n)

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

Consumo de tempo:

$O(n \lg n)$ mais o tempo do DistânciaRec-SH.

Divisão e conquista

DistânciaRec-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

DistânciaRec-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

DistânciaRec-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 $d_E \leftarrow \text{DistânciaRec-SH} (X, Y, p, q)$
- 5 $d_D \leftarrow \text{DistânciaRec-SH} (X, Y, q+1, r)$
- 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 $d_E \leftarrow \text{DistânciaRec-SH} (X, Y, p, q)$
- 5 $d_D \leftarrow \text{DistânciaRec-SH} (X, Y, q+1, r)$
- 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           $d_E \leftarrow \text{DistânciaRec-SH} (X, Y, p, q)$ 
5           $d_D \leftarrow \text{DistânciaRec-SH} (X, Y, q+1, r)$ 
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 do DistânciaRec-SH:

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

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 $d_E \leftarrow \text{DistânciaRec-SH}(X, Y, p, q)$
- 5 $d_D \leftarrow \text{DistânciaRec-SH}(X, Y, q+1, r)$
- 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 $n = r - p + 1$. Quanto vale $T(n)$? $T(n) = O(n \lg n)$.

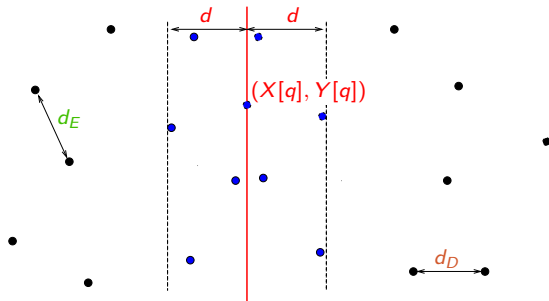
Algoritmo de Shamos e Hoey

Como fazer o **Combine** linear?

Algoritmo de Shamos e Hoey

Como fazer o **Combine** linear?

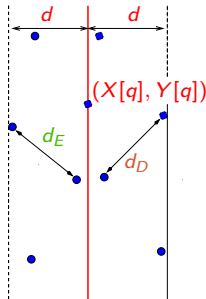
Combine precisa considerar apenas pontos que estão a uma distância menor que $d = \min\{d_E, d_D\}$ da reta vertical $x = X[q]$.



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...

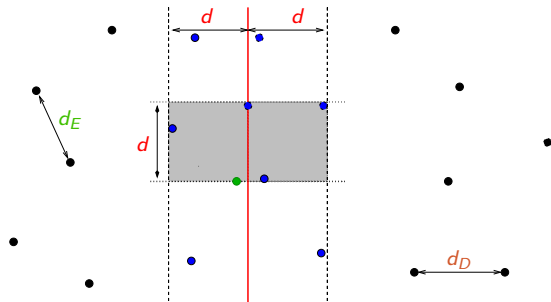
Algoritmo de Shamos e Hoey

Como fazer o **Combine** linear?

Algoritmo de Shamos e Hoey

Como fazer o **Combine** linear?

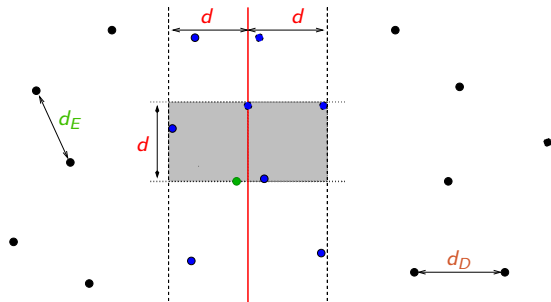
Ideia...



Algoritmo de Shamos e Hoey

Como fazer o **Combine** linear?

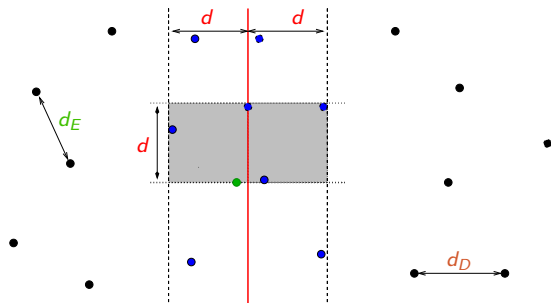
Ideia...



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

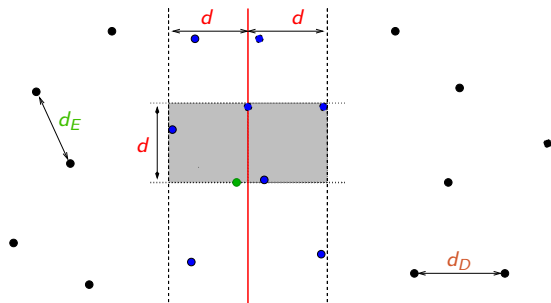
Como fazer o **Combine** linear?



Quantos pontos assim há?

Algoritmo de Shamos e Hoey

Como fazer o **Combine** linear?

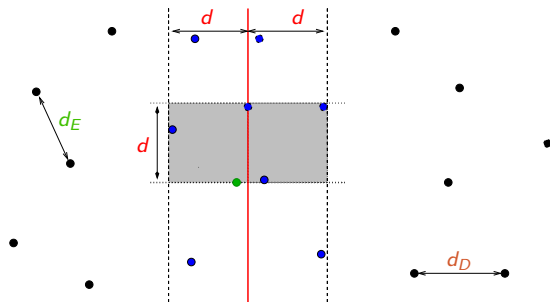


Quantos pontos assim há?

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?



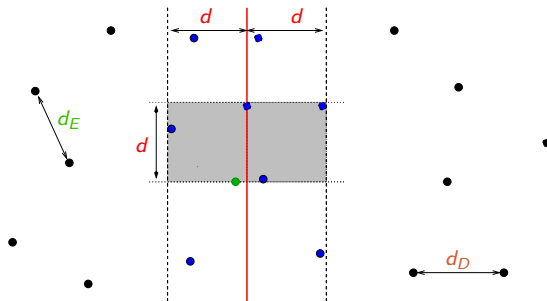
Quantos pontos assim há?

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

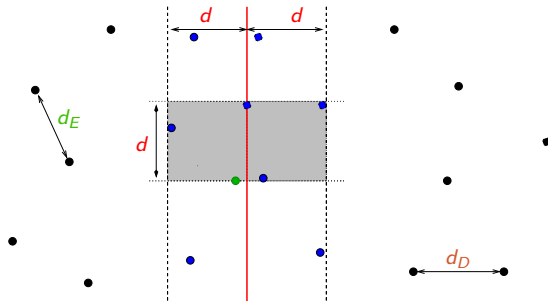
Como fazer o **Combine** linear?



Mas como ter acesso rápido a estes pontos?

Algoritmo de Shamos e Hoey

Como fazer o **Combine** linear?

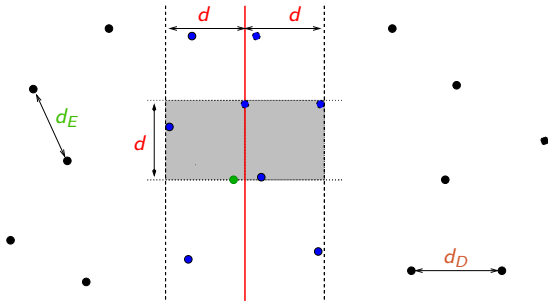


Mas como ter acesso rápido a estes pontos?

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.

Façamos o **DistânciaRec-SH** devolver os pontos ordenados pelas suas Y-coordenadas!

Divisão e conquista

DistânciaRec-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,
e reorganize os pontos para que estejam ordenados
por suas Y -coordenadas.

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) $\triangleright$ Divisão e conquista

- 1 se $r \leq p + 2$
- 2 então $\triangleright$ resolva o problema diretamente
- 3 senão $q \leftarrow \lfloor (p + r)/2 \rfloor$ $x_f \leftarrow X[q]$
- 4 $d_E \leftarrow \text{DistânciaRec-SH}(X, Y, p, q)$
- 5 $d_D \leftarrow \text{DistânciaRec-SH}(X, Y, q + 1, r)$
- 6 **Intercala**(Y, X, p, q, r)
- 7 devolva **Combine**($X, Y, p, r, d_E, d_D, x_f$)

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  $q \leftarrow \lfloor (p + r)/2 \rfloor$      $x_f \leftarrow X[q]$ 
4          $d_E \leftarrow \text{DistânciaRec-SH}(X, Y, p, q)$ 
5          $d_D \leftarrow \text{DistânciaRec-SH}(X, Y, q + 1, r)$ 
6         Intercala ( $Y, X, p, q, r$ )
7         devolva Combine ( $X, Y, p, r, d_E, d_D, x_f$ )
```

Não se esqueça de ordenar na base!

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  $q \leftarrow \lfloor (p + r)/2 \rfloor$      $x_f \leftarrow X[q]$ 
4         $d_E \leftarrow \text{DistânciaRec-SH}(X, Y, p, q)$ 
5         $d_D \leftarrow \text{DistânciaRec-SH}(X, Y, q + 1, r)$ 
6        Intercala ( $Y, X, p, q, r$ )
7        devolva Combine ( $X, Y, p, r, d_E, d_D, x_f$ )
```

Intercala e **Combine** são algoritmos lineares.

Algoritmo de Shamos e Hoey

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

```
1 se  $r \leq p + 2$ 
2   então  $\triangleright$  resolva o problema diretamente
3   senão  $q \leftarrow \lfloor (p + r)/2 \rfloor$     $x_f \leftarrow X[q]$ 
4          $d_E \leftarrow \text{DistânciaRec-SH}(X, Y, p, q)$ 
5          $d_D \leftarrow \text{DistânciaRec-SH}(X, Y, q + 1, r)$ 
6         Intercala ( $Y, X, p, q, r$ )
7         devolva Combine ( $X, Y, p, r, d_E, d_D, x_f$ )
```

Intercala 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, p, r) $\triangleright$ Divisão e conquista

- 1 se $r \leq p + 2$
- 2 então $\triangleright$ resolva o problema diretamente
- 3 senão $q \leftarrow \lfloor (p + r)/2 \rfloor$ $x_f \leftarrow X[q]$
- 4 $d_E \leftarrow \text{DistânciaRec-SH}(X, Y, p, q)$
- 5 $d_D \leftarrow \text{DistânciaRec-SH}(X, Y, q + 1, r)$
- 6 $\text{Intercala}(Y, X, p, q, r)$
- 7 devolva $\text{Combine}(X, Y, p, r, d_E, d_D, x_f)$

Intercala 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

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, p, r, d, x_f)

- 1 $t \leftarrow 0$
- 2 para $k \leftarrow p$ até r faça
- 3 se $|X[k] - x_f| < d$
- 4 então $t \leftarrow t + 1$
- 5 $f[t] \leftarrow k$
- 6 devolva (f, t)

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

$$x_f = 1 \quad d = \sqrt{5}$$

Algoritmo de Shamos e Hoey

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

Candidatos (X, p, r, d, x_f)

- 1 $t \leftarrow 0$
- 2 para $k \leftarrow p$ até r faça
- 3 se $|X[k] - x_f| < d$
- 4 então $t \leftarrow t + 1$
- 5 $f[t] \leftarrow k$
- 6 devolva (f, t)

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

$x_f = 1$ $d = \sqrt{5}$

Algoritmo de Shamos e Hoey

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

Candidatos (X, p, r, d, x_f)

```
1   $t \leftarrow 0$ 
2  para  $k \leftarrow p$  até  $r$  faça
3      se  $|X[k] - x_f| < d$ 
4          então  $t \leftarrow t + 1$ 
5               $f[t] \leftarrow k$ 
6  devolva  $(f, t)$ 
```

X	1	-2	4	0	3
Y	-2	-1	2	3	4
	1	2	3	4	5
f	1	4	5		
	$x_f = 1$		$d = \sqrt{5}$		

Consumo de tempo:

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

Algoritmo de Shamos e Hoey

Combine ($X, Y, p, r, d_E, d_D, x_f$)

- 1 $d \leftarrow \min\{d_E, d_D\}$
- 2 $(f, t) \leftarrow \text{Candidatos}(X, p, r, d, x_f)$ $\triangleright$ 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 ($X, Y, p, r, d_E, d_D, x_f$)

- 1 $d \leftarrow \min\{d_E, d_D\}$
- 2 $(f, t) \leftarrow \text{Candidatos}(X, p, r, d, x_f)$ $\triangleright$ 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 ($X, Y, p, r, d_E, d_D, x_f$)

- 1 $d \leftarrow \min\{d_E, d_D\}$
- 2 $(f, t) \leftarrow \text{Candidatos}(X, p, r, d, x_f)$ $\triangleright$ 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 ($X, Y, p, r, d_E, d_D, x_f$)

- 1 $d \leftarrow \min\{d_E, d_D\}$
- 2 $(f, t) \leftarrow \text{Candidatos}(X, p, r, d, x_f)$ $\triangleright$ 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 $n = r - p + 1$.

Exercícios

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).

Exercícios

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?

Exercícios

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 de implementação:

UVA 10245 – The Closest Pair Problem

<http://uva.onlinejudge.org/external/102/10245.pdf>

Exercícios

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 de implementação:

UVA 10245 – The Closest Pair Problem

<http://uva.onlinejudge.org/external/102/10245.pdf>

Agora, vamos ver uma **animação** deste algoritmo!