Geometria Computacional

Cristina G. Fernandes

Departamento de Ciência da Computação do IME-USP

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

segundo semestre de 2011

Introdução

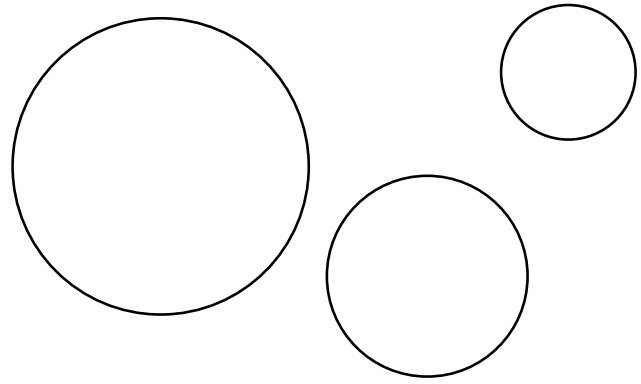
Antiguidade:

- construções geométricas de Euclides (régua e compasso)

Introdução

Antiguidade:

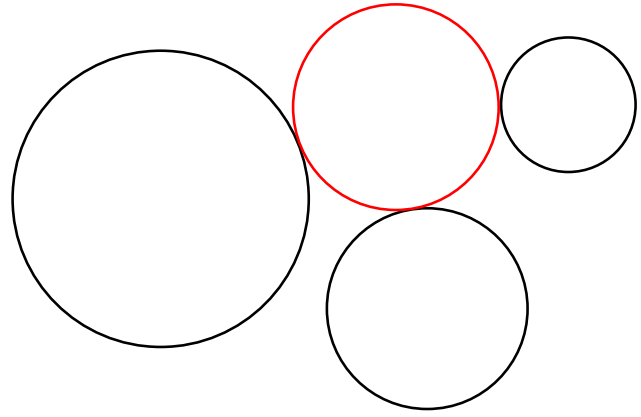
- construções geométricas de Euclides (régua e compasso)
- problema de Apollonius (cerca de 200 aC)



Introdução

Antiguidade:

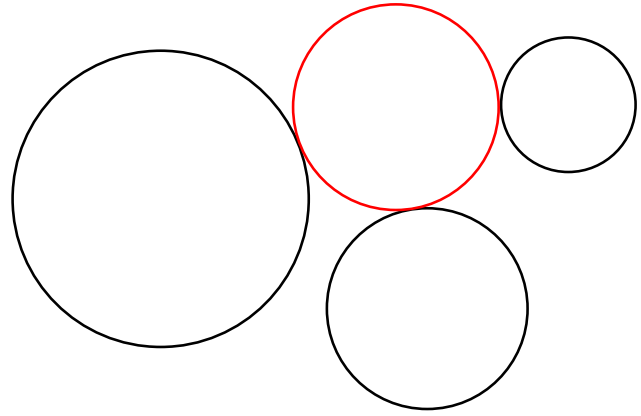
- construções geométricas de Euclides (régua e compasso)
- problema de Apollonius (cerca de 200 aC)



Introdução

Antiguidade:

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

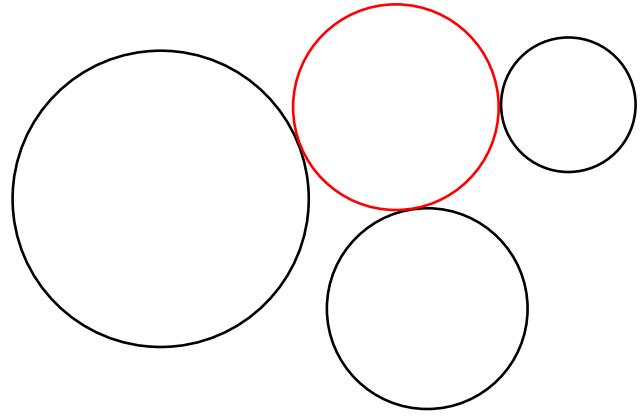


Solução de Euclides: 508 operações “elementares”

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

Modelo de computação

Algoritmo:
sequência finita de instruções que resolve um problema.

Modelo de computação

Algoritmo:
sequência finita de instruções que resolve um problema.

Modelo de computação: descrição abstrata de um computador que será usado para executar um algoritmo.

Modelo de computação

Algoritmo:
sequência finita de instruções que resolve um problema.

Modelo de computação: descrição abstrata de um computador que será usado para executar um algoritmo.

- operações elementares (aritméticas, comparações, etc)
- critério para medir consumo de tempo

Modelo de computação

Algoritmo:
sequência finita de instruções que resolve um problema.

Modelo de computação: descrição abstrata de um computador que será usado para executar um algoritmo.

- operações elementares (aritméticas, comparações, etc)
- critério para medir consumo de tempo

Compromisso entre realidade e tratabilidade matemática.

Modelo de computação

Real RAM (real random access machine)
com custo uniforme:

- manipula números reais arbitrários
- operações com reais custam 1 (mesmo raiz quadrada)

Modelo de computação

Real RAM (real random access machine)
com custo uniforme:

- manipula números reais arbitrários
- operações com reais custam 1 (mesmo raiz quadrada)

Análise de algoritmos:

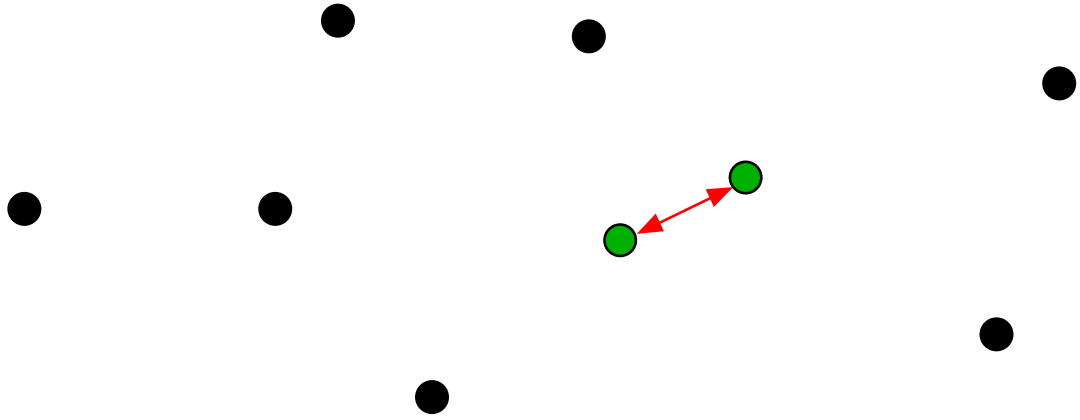
- notação assintótica
- técnicas básicas

Par de pontos mais próximos

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

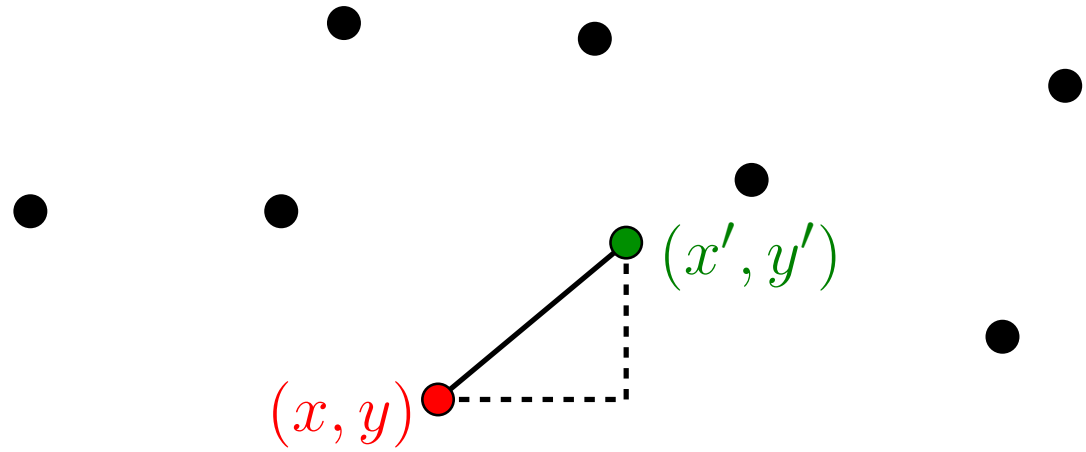
Par de pontos mais próximos

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



Par de pontos mais próximos

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



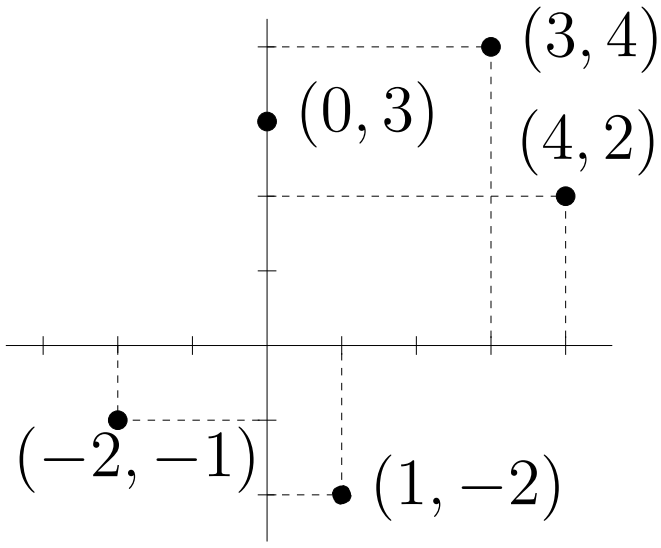
Lembre-se que, para dois pontos (x, y) e (x', y') no plano,

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

Par de pontos mais próximos

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

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

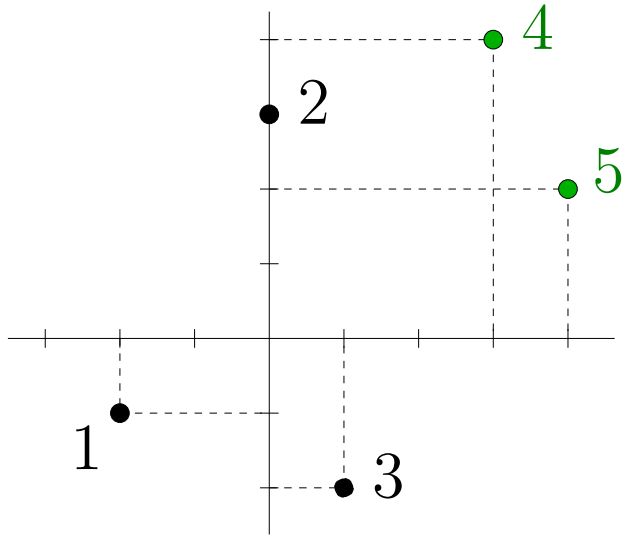


X	-2	0	1	3	4
Y	-1	3	-2	4	2
	1	2	3	4	5

Par de pontos mais próximos

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

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



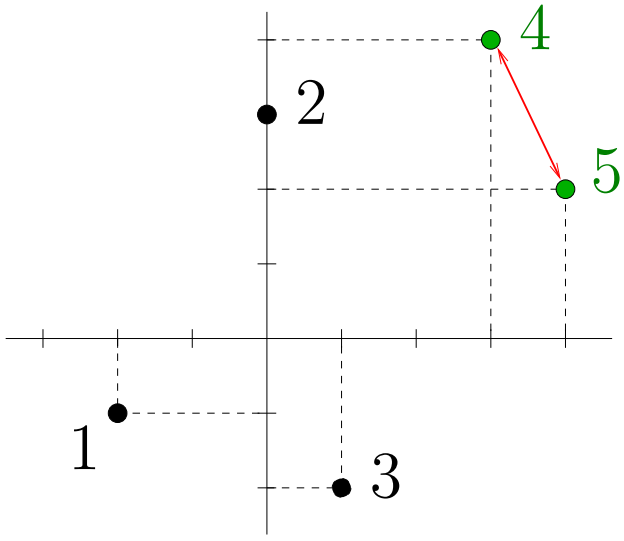
X	-2	0	1	3	4
Y	-1	3	-2	4	2
	1	2	3	4	5

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 ← +∞
 2  para i ← 2 até n faça
 3       para j ← 1 até i − 1 faça
 4            se DIST(X[i], Y[i], X[j], Y[j]) < d
 5                 então d ← DIST(X[i], Y[i], X[j], Y[j])
 6  devolva d
```

Algoritmo elementar

```
ELEMENTAR(X, Y, n)
 1  d ← +∞
 2  para i ← 2 até n faça
 3       para j ← 1 até i − 1 faça
 4            se DIST(X[i], Y[i], X[j], Y[j]) < d
 5                 então d ← DIST(X[i], Y[i], X[j], Y[j])
 6  devolva d
```

Algoritmo elementar

ELEMENTAR(X, Y, n)

```
1  d ← +∞
2  para i ← 2 até n faça
3      para j ← 1 até i − 1 faça
4          se DIST(X[i], Y[i], X[j], Y[j]) < d
5              então d ← DIST(X[i], Y[i], X[j], Y[j])
6  devolva d
```

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

Algoritmo elementar

ELEMENTAR(X, Y, n)

```
1  d ← +∞
2  para i ← 2 até n faça
3      para j ← 1 até i − 1 faça
4          se DIST(X[i], Y[i], X[j], Y[j]) < d
5              então d ← 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 ← +∞
 2  para i ← 2 até n faça
 3       para j ← 1 até i − 1 faça
 4            se DIST(X[i], Y[i], X[j], Y[j]) < d
 5                 então d ← DIST(X[i], Y[i], X[j], Y[j])
 6  devolva d
```

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

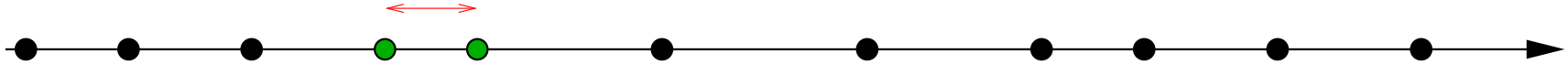
Consumo de tempo: linha 4 é executada

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

É possível projetar um algoritmo mais eficiente que este?

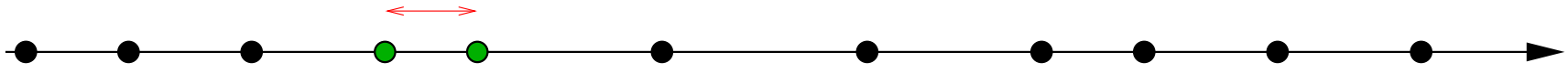
Par mais próximo na reta

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.

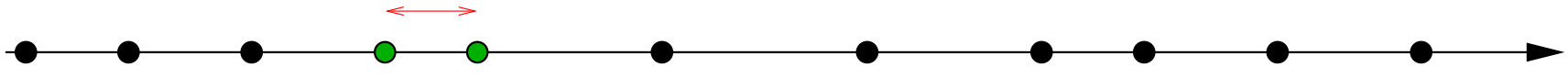


Primeira solução: ordene os pontos, e encontre os dois consecutivos mais próximos.

Tempo consumido: $\mathrm{O}(n \lg n)$.

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: $\mathrm{O}(n \lg n)$.

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.

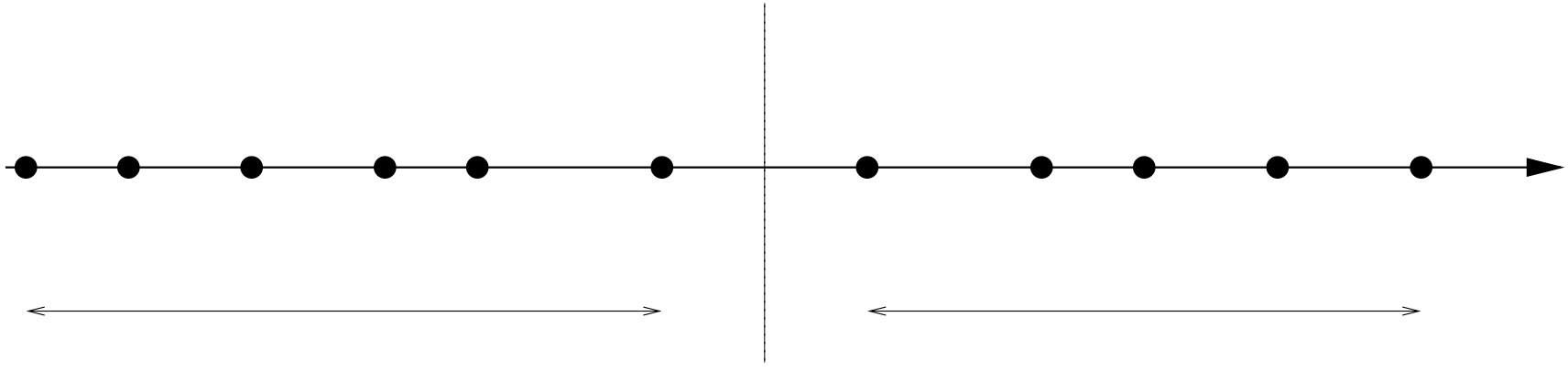
Divisão e conquista

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.



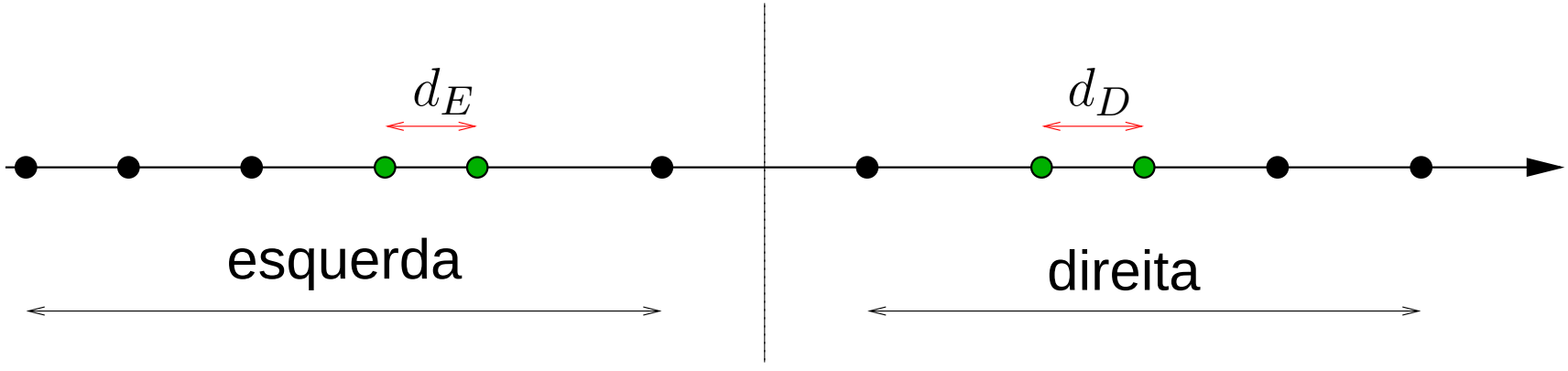
Divisão e conquista

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.



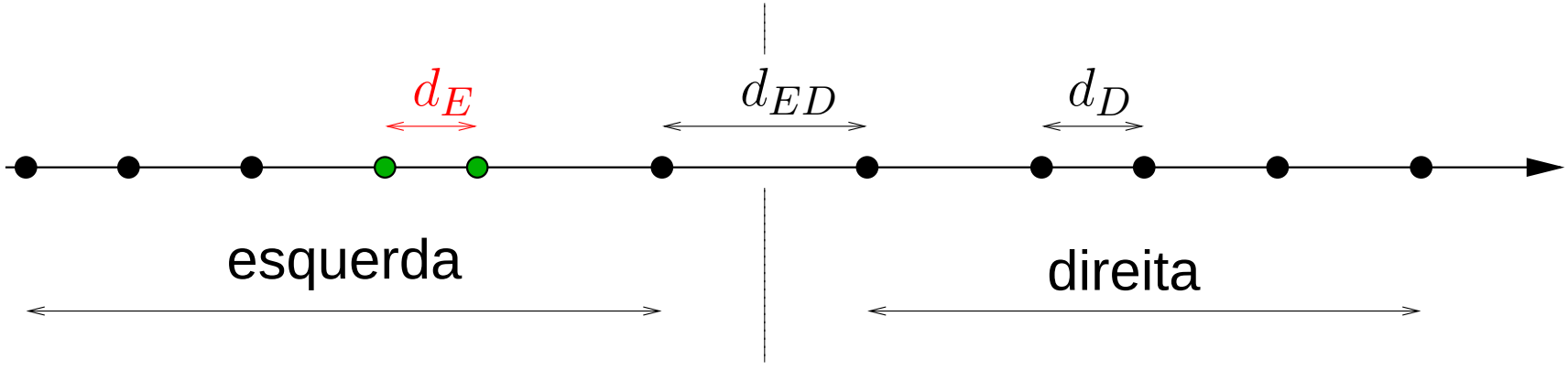
Divisão e conquista

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.

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.

DISTÂNCIARETA(X, n)
1 MERGESORT$(X, 1, n)$
2 devolva DISTÂNCIARETAREC $(X, 1, n)$

DISTÂNCIARETAREC: divisão e conquista.

Tempo consumido pelo DISTÂNCIARETA:

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

Par mais próximo na reta

```
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

```
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:

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

onde $n = r - p + 1$.

Par mais próximo na reta

```
DistânciaRetaRec (X, p, r)        ▷ Divisão e conquista
1  se r ≤ 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:

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

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

Par mais próximo na reta

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

```
1  se r ≤ 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:

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

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 $\mathrm{O}(n \lg n)$.

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 $\mathrm{O}(n \lg n)$.

DISTÂNCIARETAREC consome tempo $\Theta(n)$.

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

Par mais próximo no plano

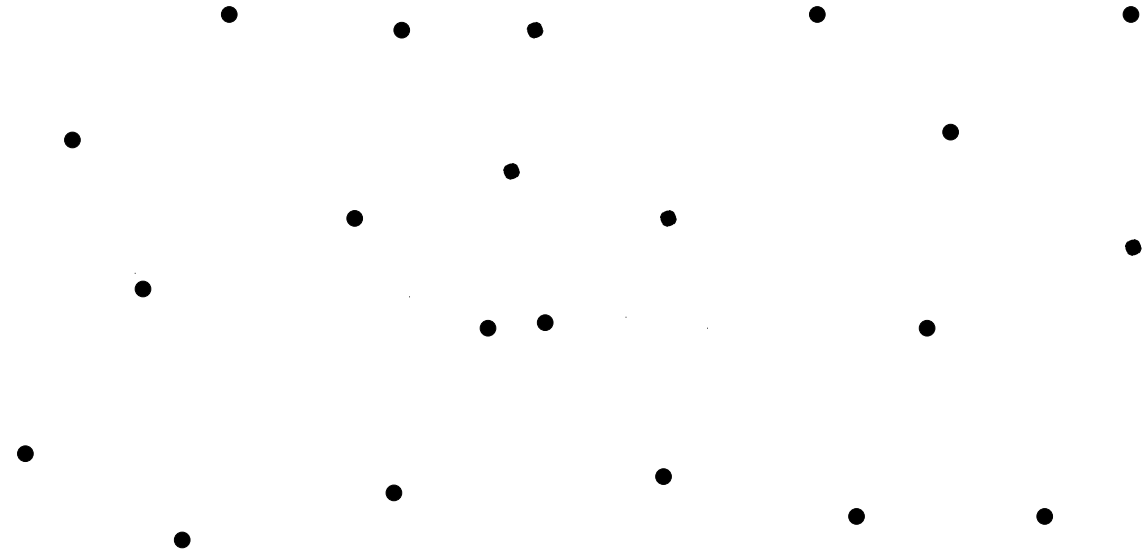
Obtivemos um algoritmo $\mathrm{O}(n \lg n)$ para pontos na reta.

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?

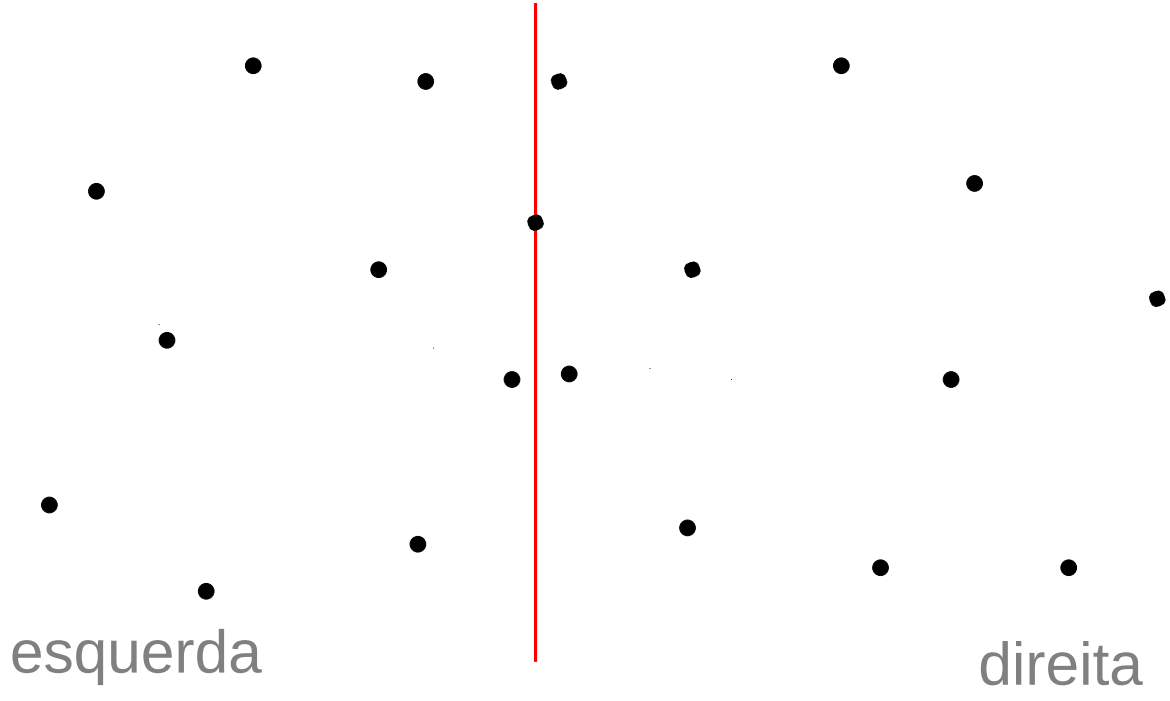


Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

Divide...

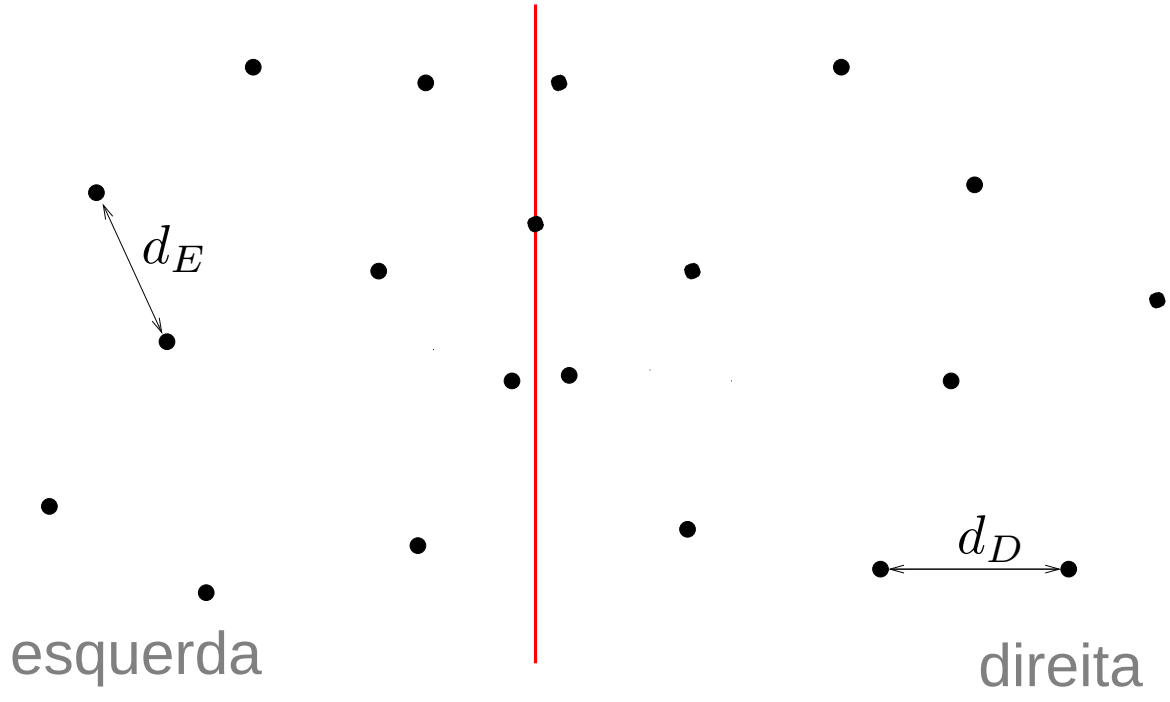


Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

Divide... Conquista...

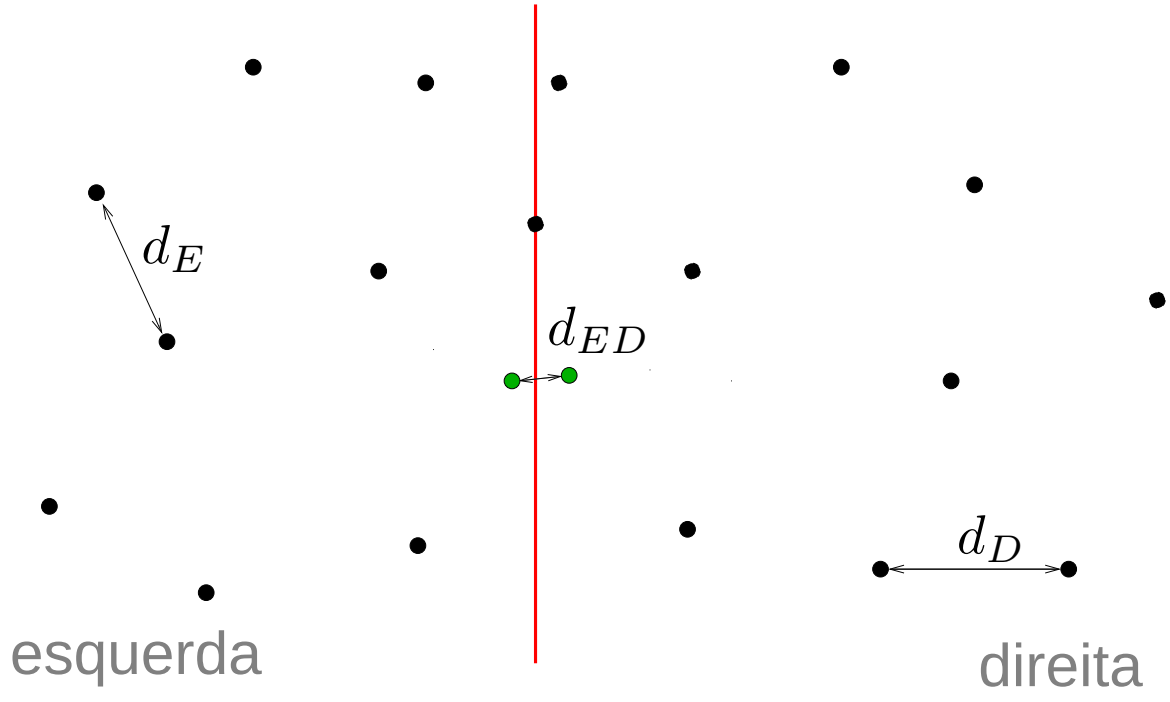


Par mais próximo no plano

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 $r \leq p + 2$
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 ≤ 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 ≤ 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.

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

```
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.

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

```
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.

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

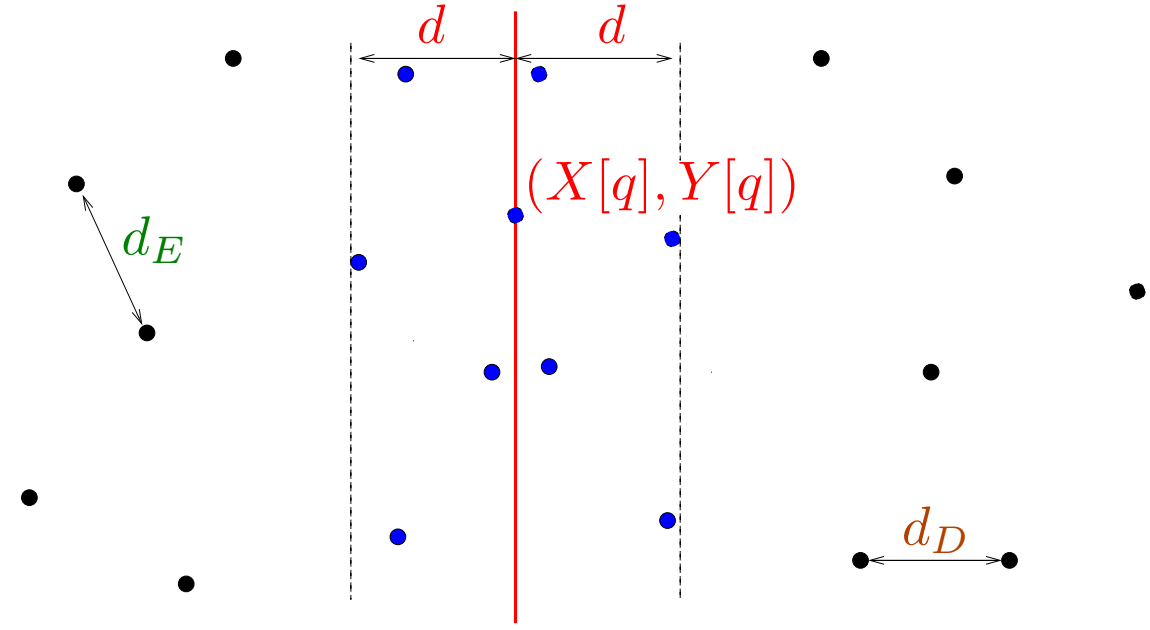
Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

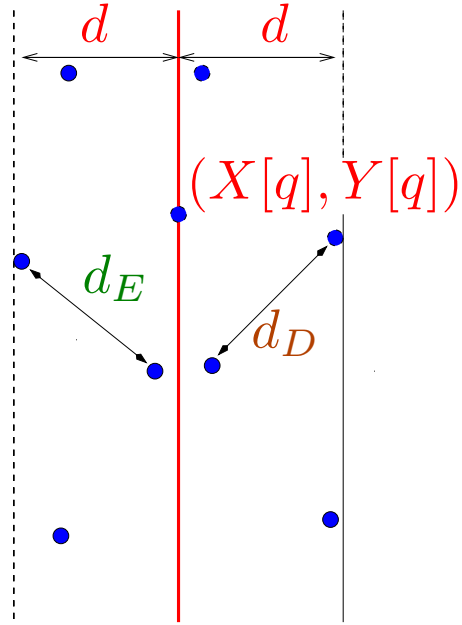
COMBINE precisa considerar apenas pontos que estão a uma distância menor que $d = \min\{d_E, d_D\}$ da reta vertical $x = X[q]$.



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...

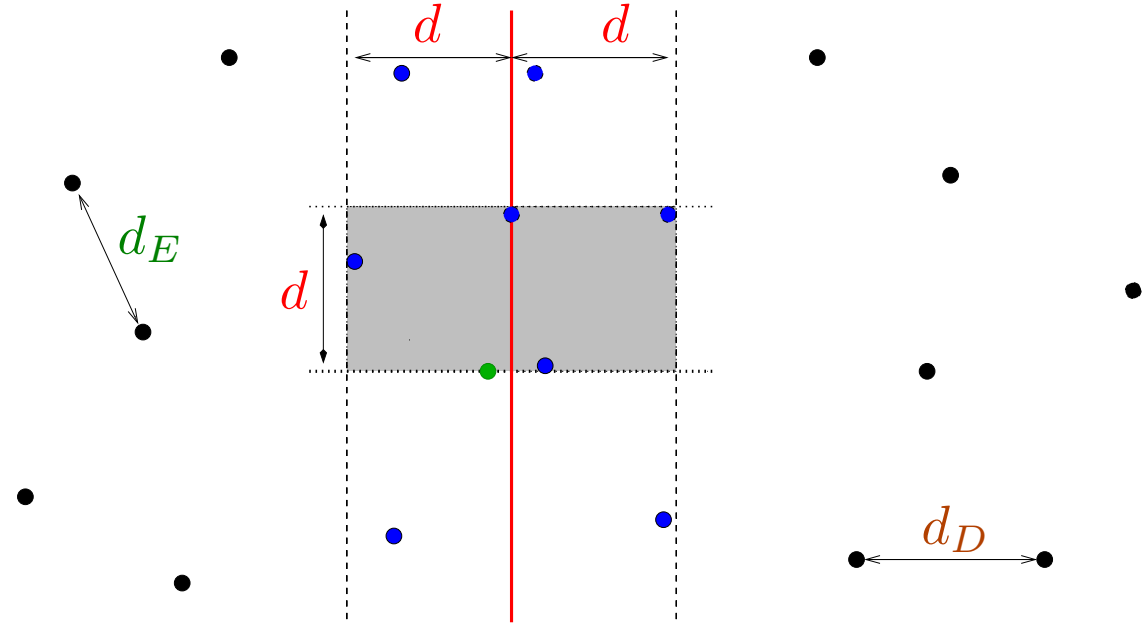
Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

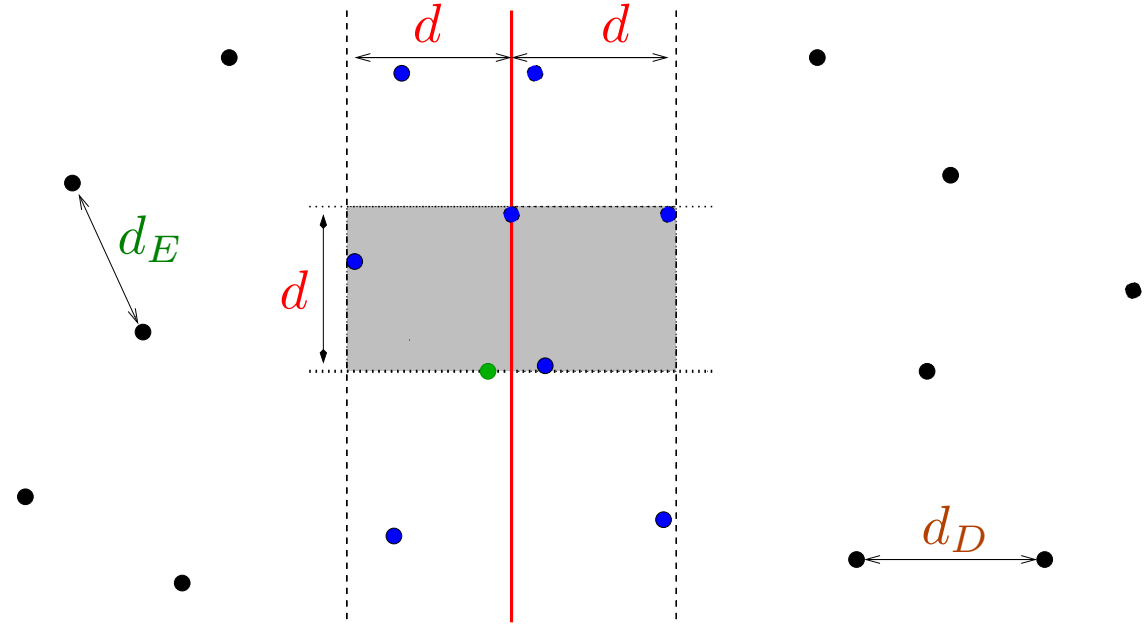
Ideia...



Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

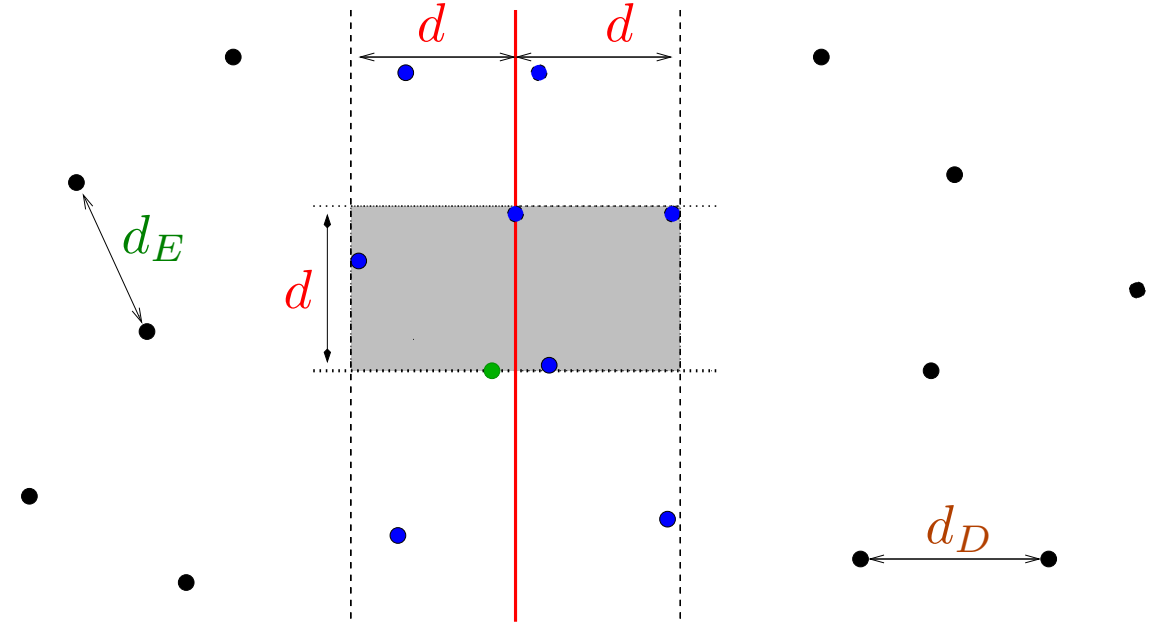
Ideia...



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

Algoritmo de Shamos e Hoey

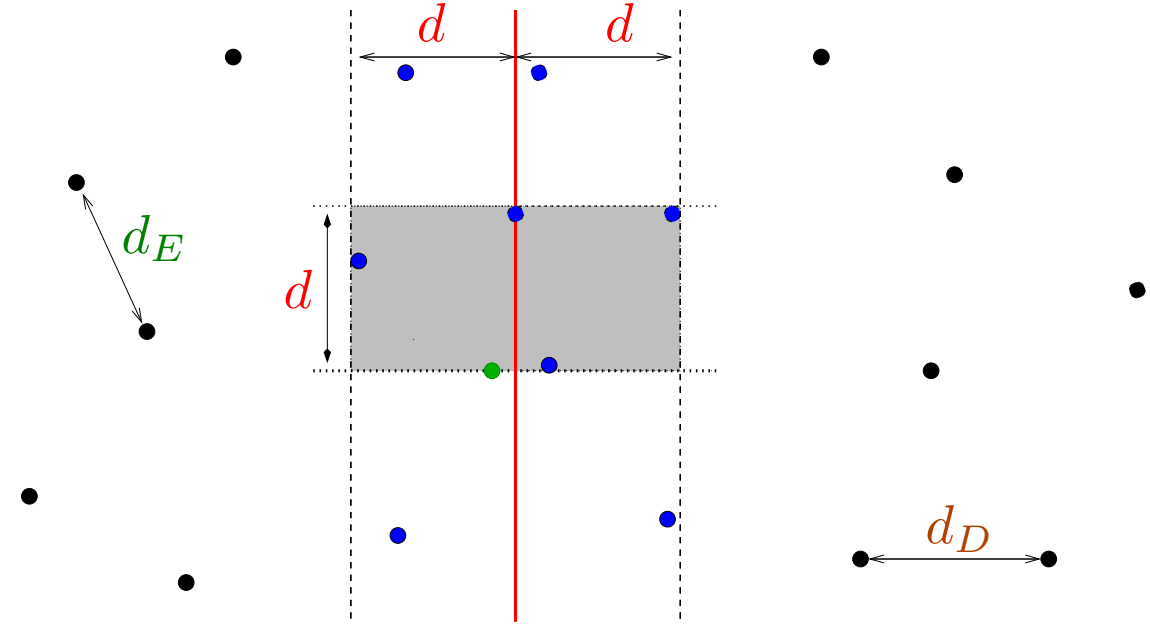
Como fazer o COMBINE linear?



Quantos pontos assim há?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

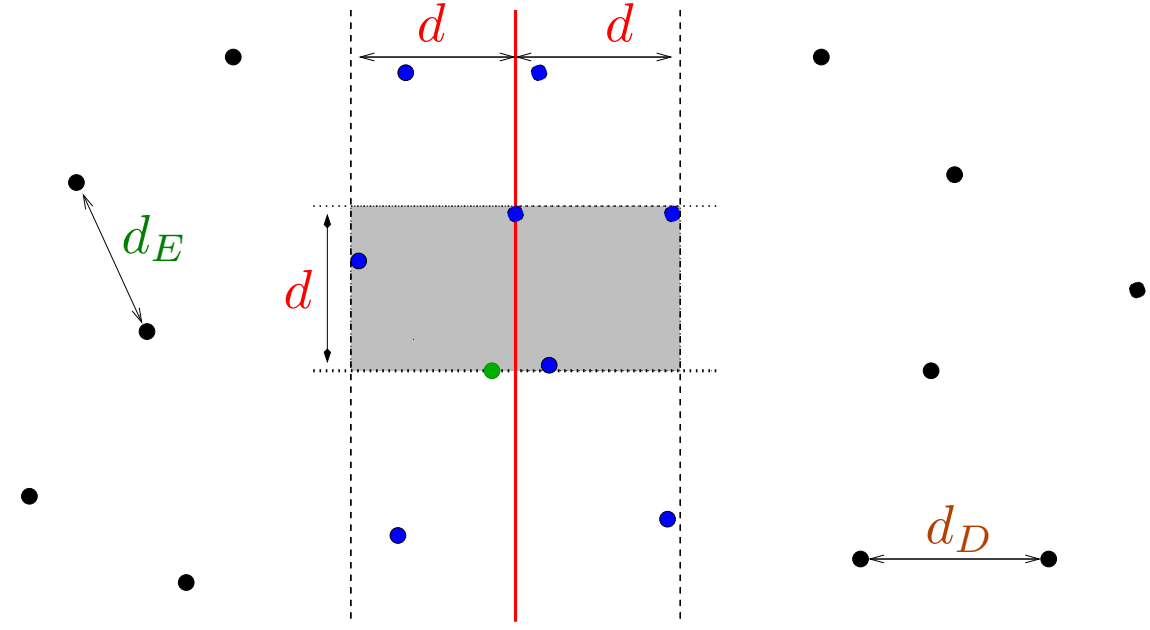


Quantos pontos assim há?

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

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?



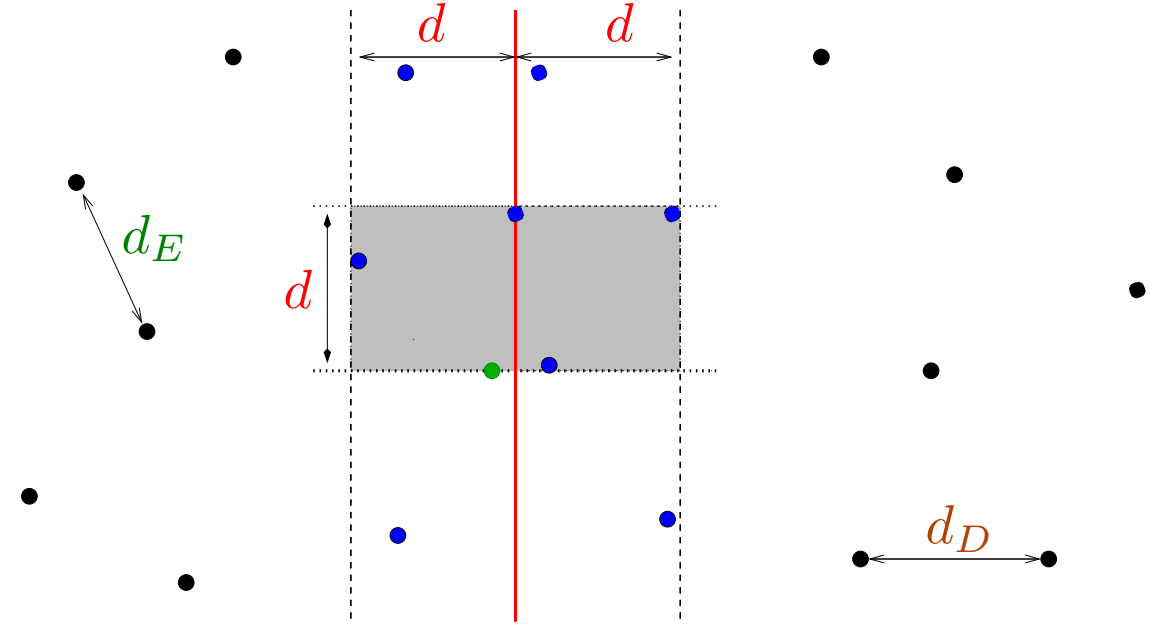
Quantos pontos assim há?

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

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

Algoritmo de Shamos e Hoey

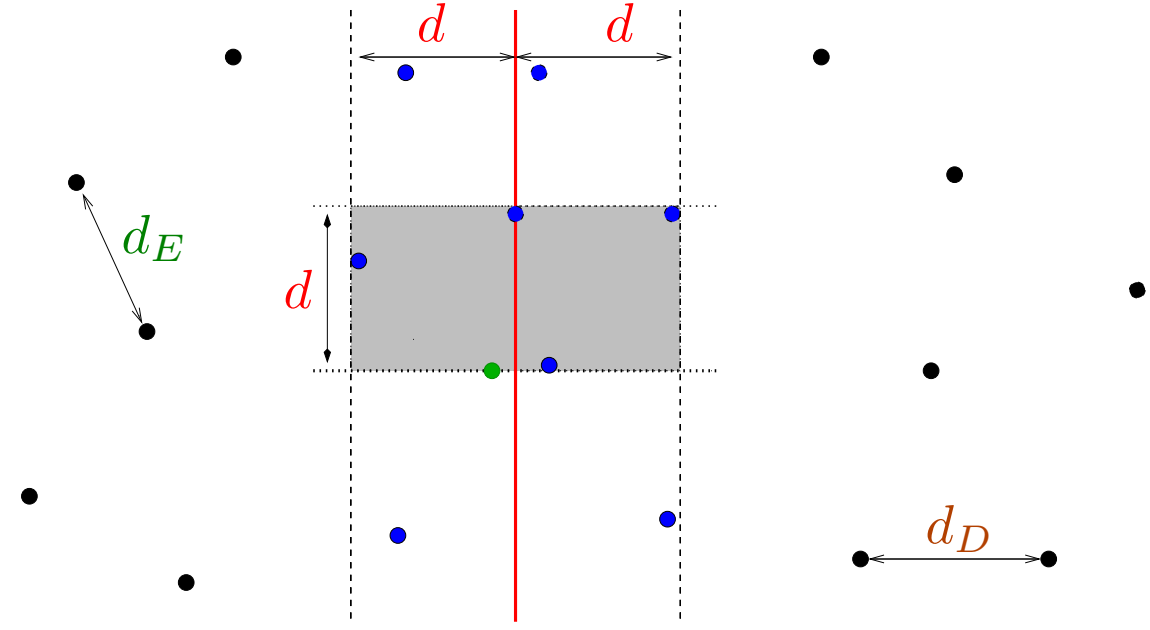
Como fazer o COMBINE linear?



Mas como ter acesso rápido a estes pontos?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?



Mas como ter acesso rápido a estes pontos?

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