Convite à Geometria Computacional

Jornadas de Atualização em Informática

Cristina G. Fernandes e **José Coelho de Pina**

Departamento de Ciência da Computação do IME-USP

http://www.ime.usp.br/dcc/

Bento Gonçalves, julho de 2009

Visão geral do minicurso

Aula 1:

- Introdução (Secs. 7.1 e 7.2)
- Par de pontos mais próximos (Sec. 7.3)

Visão geral do minicurso

Aula 1:

- Introdução (Secs. 7.1 e 7.2)
- Par de pontos mais próximos (Sec. 7.3)

Aula 2:

- Fecho convexo (Sec. 7.4)

Visão geral do minicurso

Aula 1:

- Introdução (Secs. 7.1 e 7.2)
- Par de pontos mais próximos (Sec. 7.3)

Aula 2:

- Fecho convexo (Sec. 7.4)

Aula 3:

- Método da linha de varredura (Secs. 7.5 e 7.6)

Introdução

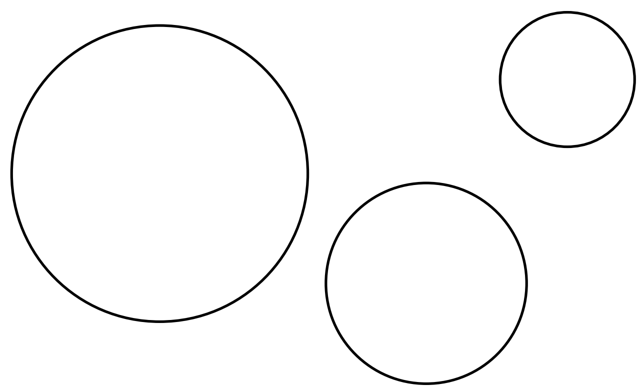
Antiguidade:

- construções geométricas de Euclides (régua e compasso)

Introdução

Antiguidade:

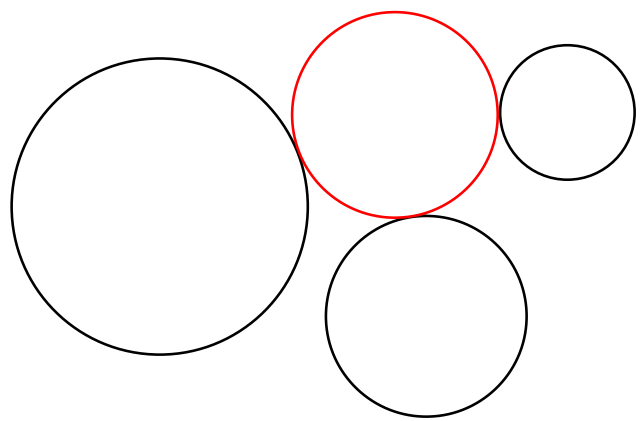
- construções geométricas de Euclides (régua e compasso)
- problema de Apollonius (cerca de 200 aC)



Introdução

Antiguidade:

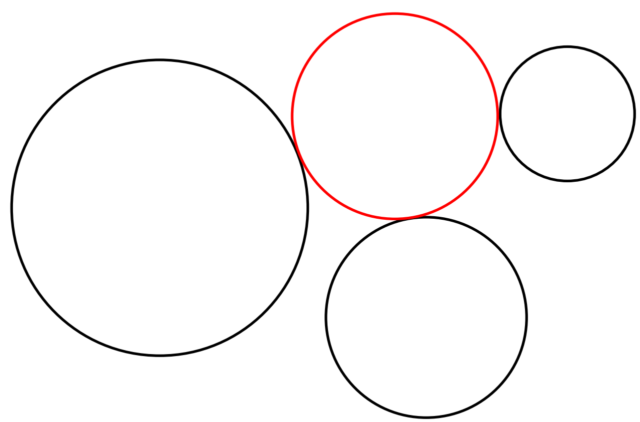
- construções geométricas de Euclides (régua e compasso)
- problema de Apollonius (cerca de 200 aC)



Introdução

Antiguidade:

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

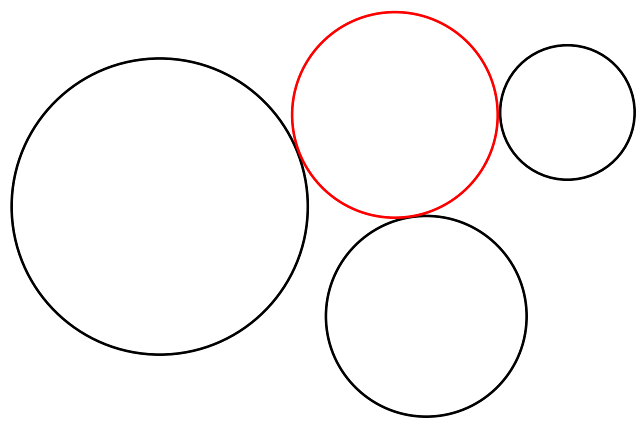


Solução de Euclides: 508 operações “elementares”

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

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

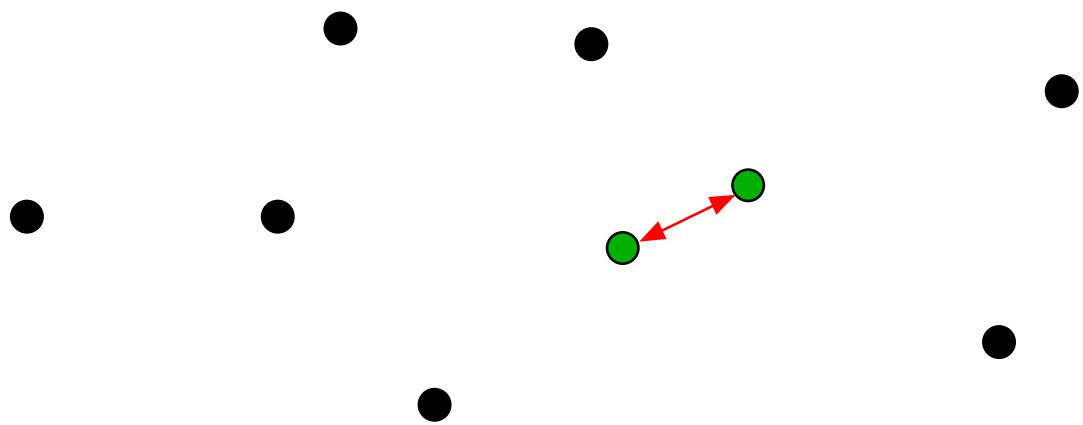
Capítulo 1 *Análise de algoritmos*
por Paulo Feofiloff

Par de pontos mais próximos

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

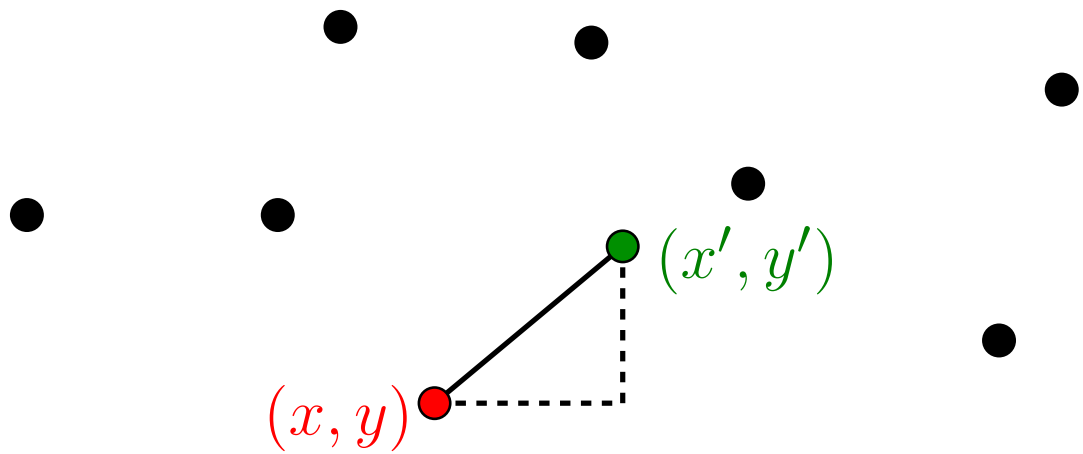
Par de pontos mais próximos

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.



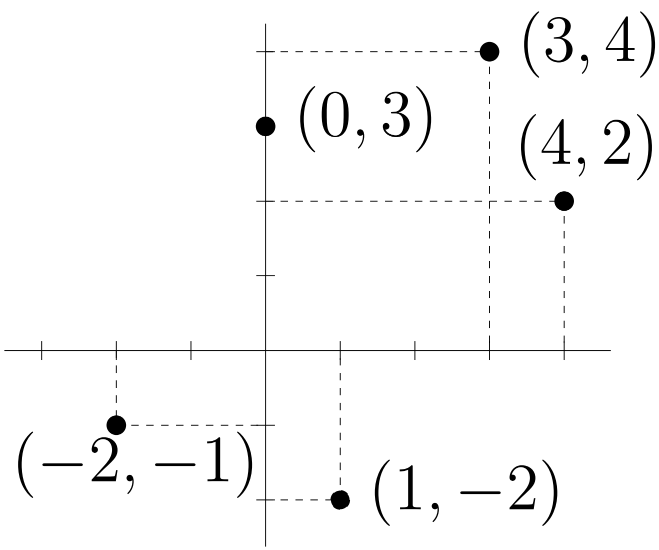
Lembre-se que, para dois pontos (x, y) e (x', y') no plano,

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

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 $Y[1\,.\,.\,n]$.

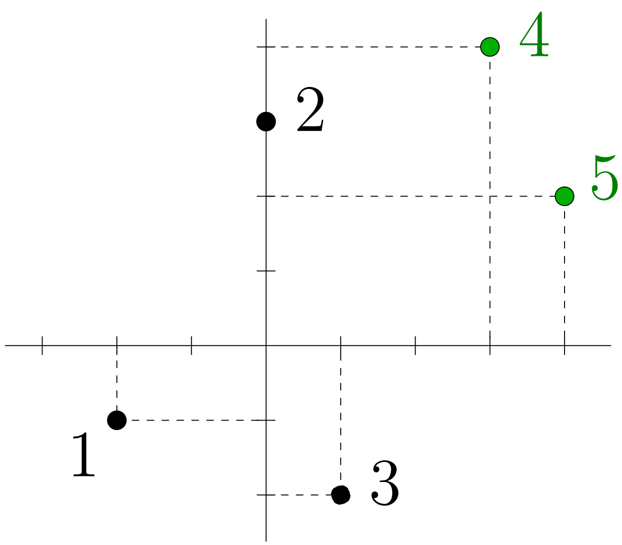


X	-2	0	1	3	4
Y	-1	3	-2	4	2
	1	2	3	4	5

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 $Y[1\,.\,.\,n]$.



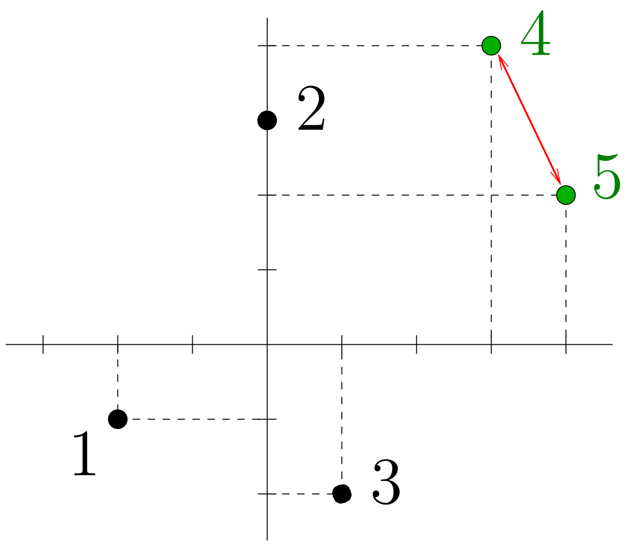
X	-2	0	1	3	4
Y	-1	3	-2	4	2
	1	2	3	4	5

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

Algoritmo elementar

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

Algoritmo elementar

ELEMENTAR(X, Y, n)

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

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

Algoritmo elementar

ELEMENTAR(X, Y, n)

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

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

Consumo de tempo: linha 4 é executada

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

Algoritmo elementar

```
ELEMENTAR(X, Y, n)
1  d ← +∞
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 $X[1\,..\,i-1], Y[1\,..\,i-1]$.

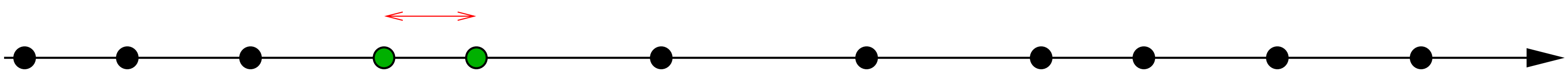
Consumo de tempo: linha 4 é executada

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

É possível projetar um algoritmo mais eficiente que este?

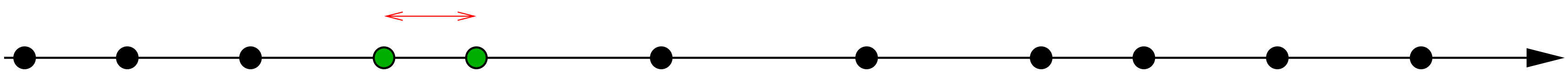
Par mais próximo na reta

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.

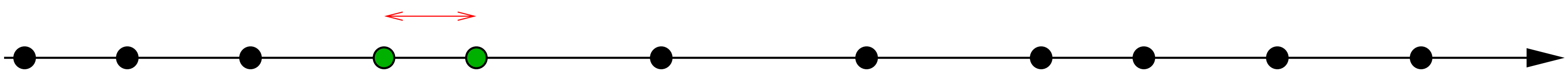


Primeira solução: ordene os pontos, e encontre os dois consecutivos mais próximos.

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

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.

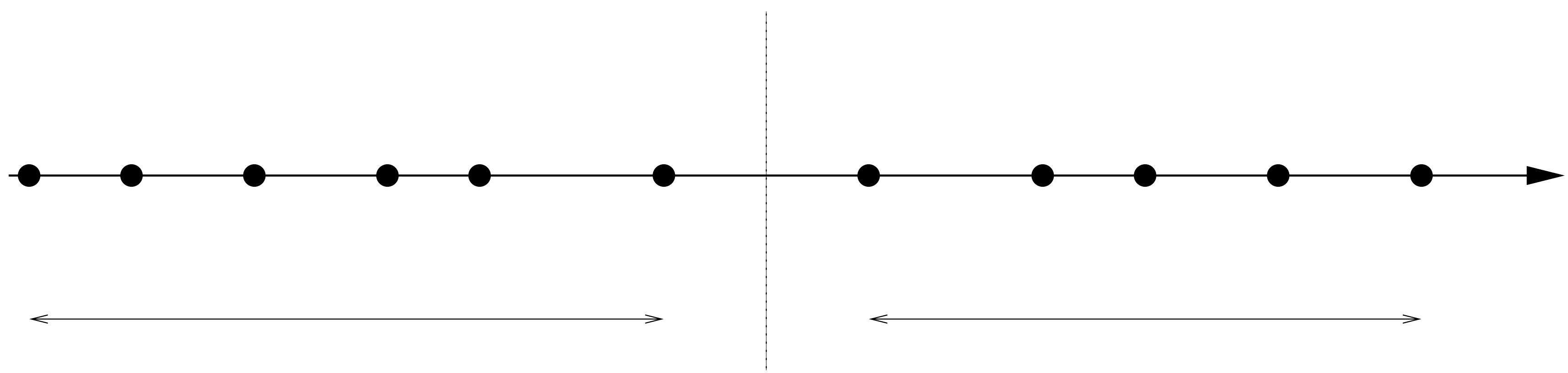
Divisão e conquista

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.



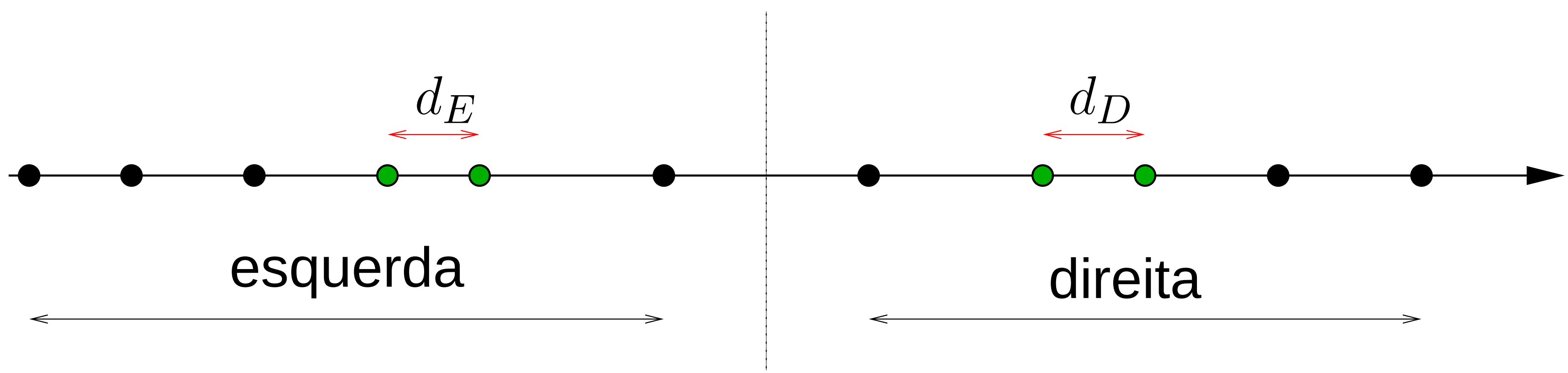
Divisão e conquista

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.



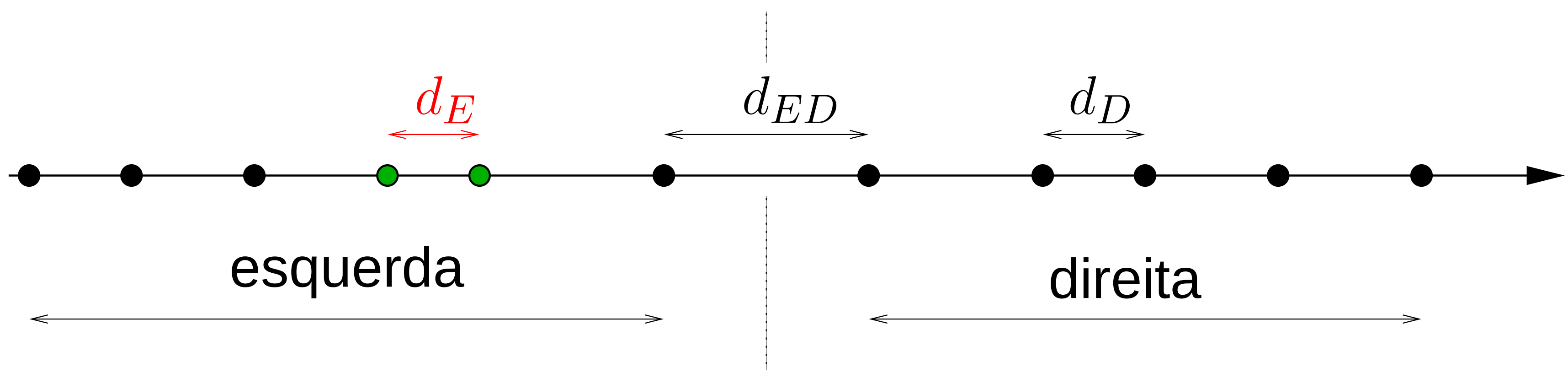
Divisão e conquista

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) $\triangleright$ Divisão e conquista

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

Par mais próximo na reta

DistânciaRetaRec (X, p, r) $\triangleright$ Divisão e conquista

se $r \leq p + 1$
    então se $r = p$
        então devolva $+\infty$
        senão devolva $X[r] - X[p]$
    senão $q \leftarrow \lfloor (p + r)/2 \rfloor$
        $d_E \leftarrow$ DistânciaRetaRec(X, p, q)
        $d_D \leftarrow$ DistânciaRetaRec$(X, q + 1, r)$
        $d \leftarrow \min\{d_E, d_D, X[q{+}1] - X[q]\}$
        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) $\triangleright$ Divisão e conquista

se $r \leq p + 1$
  então se $r = p$
      então devolva $+\infty$
      senão devolva $X[r] - X[p]$
  senão $q \leftarrow \lfloor (p + r)/2 \rfloor$
      $d_E \leftarrow$ DISTÂNCIARETAREC(X, p, q)
      $d_D \leftarrow$ DISTÂNCIARETAREC$(X, q + 1, r)$
      $d \leftarrow \min\{d_E, d_D, X[q{+}1] - X[q]\}$
      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) $\triangleright$ Divisão e conquista

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

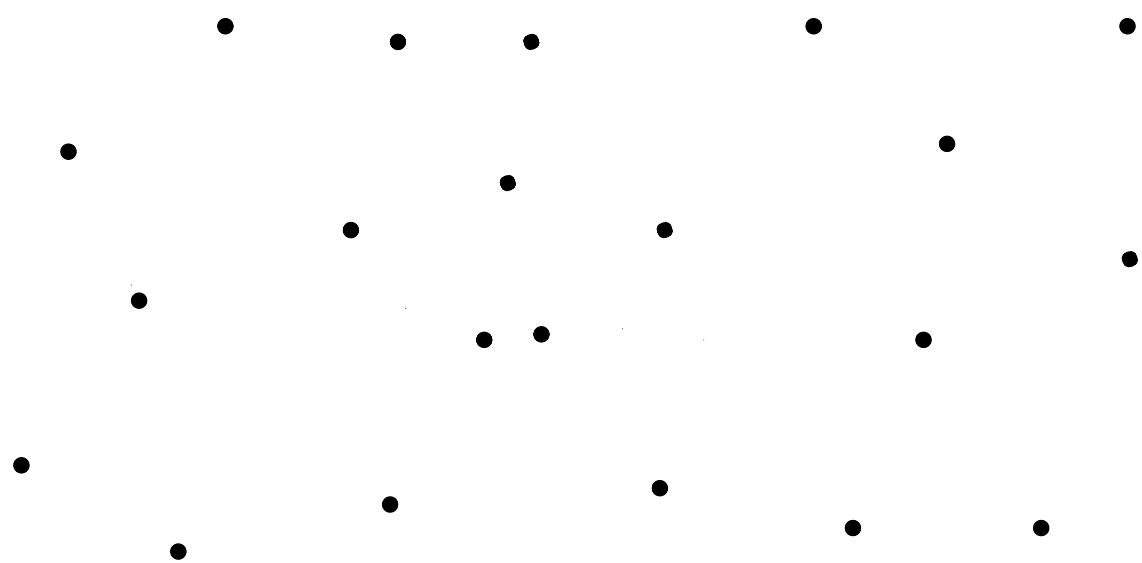
Obtivemos um algoritmo $\mathrm{O}(n \lg n)$ para pontos na reta.

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?

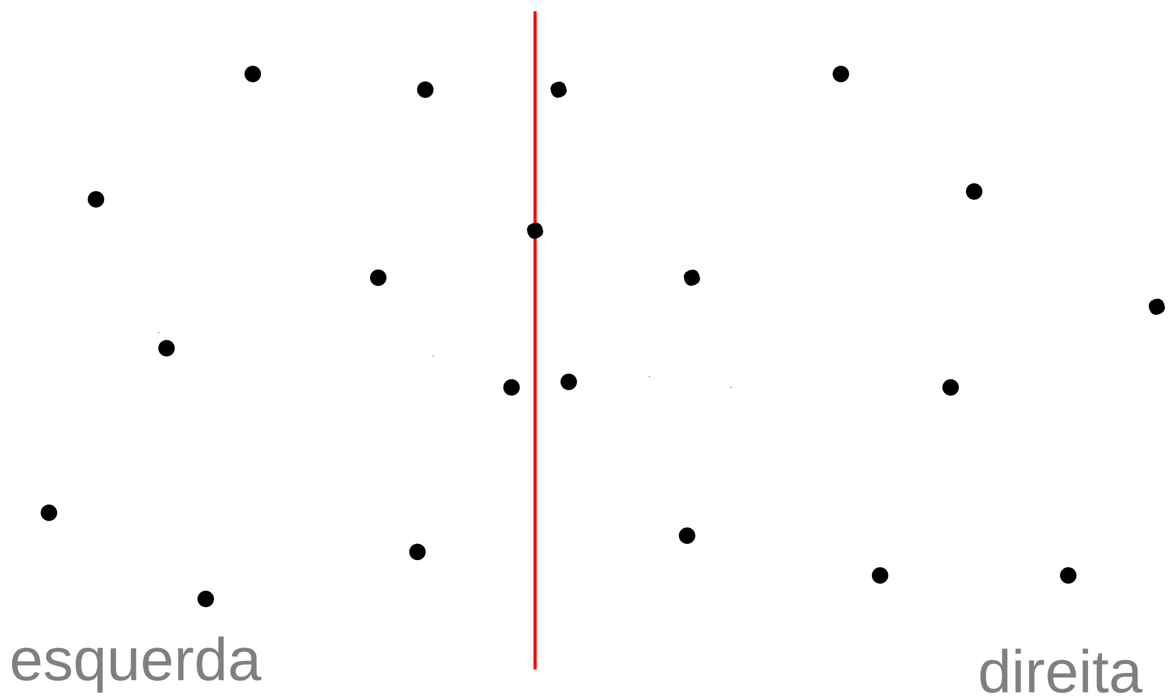


Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

Divide...

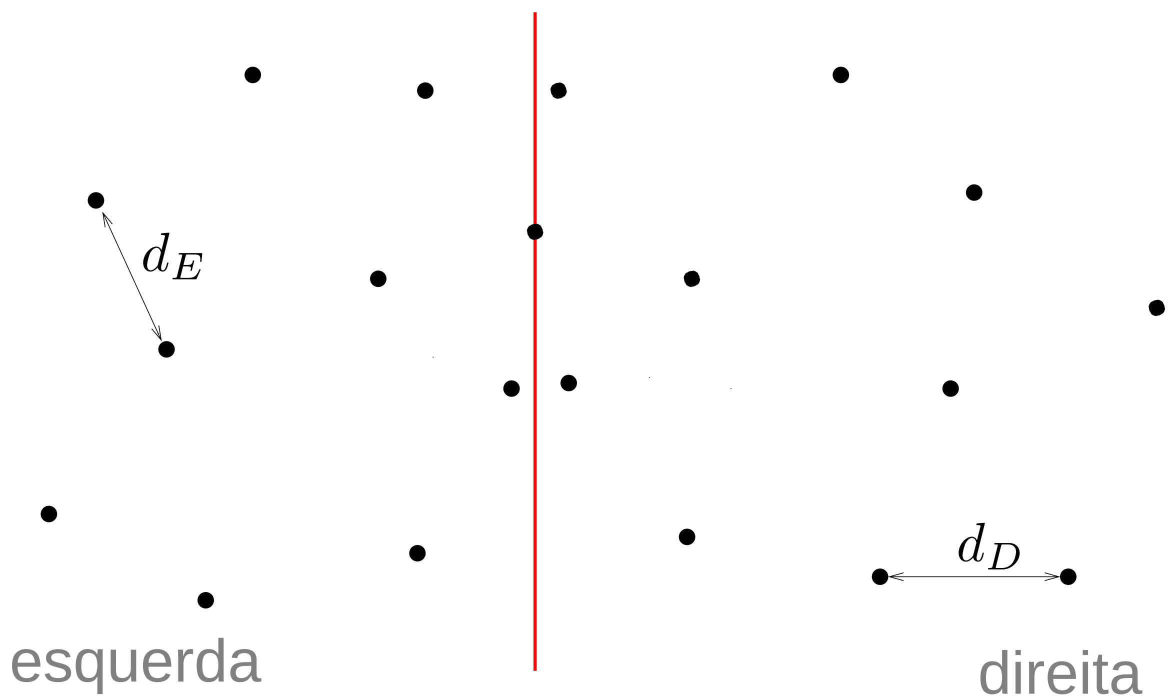


Par mais próximo no plano

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

Como generalizar essa ideia para o plano?

Divide... Conquista...

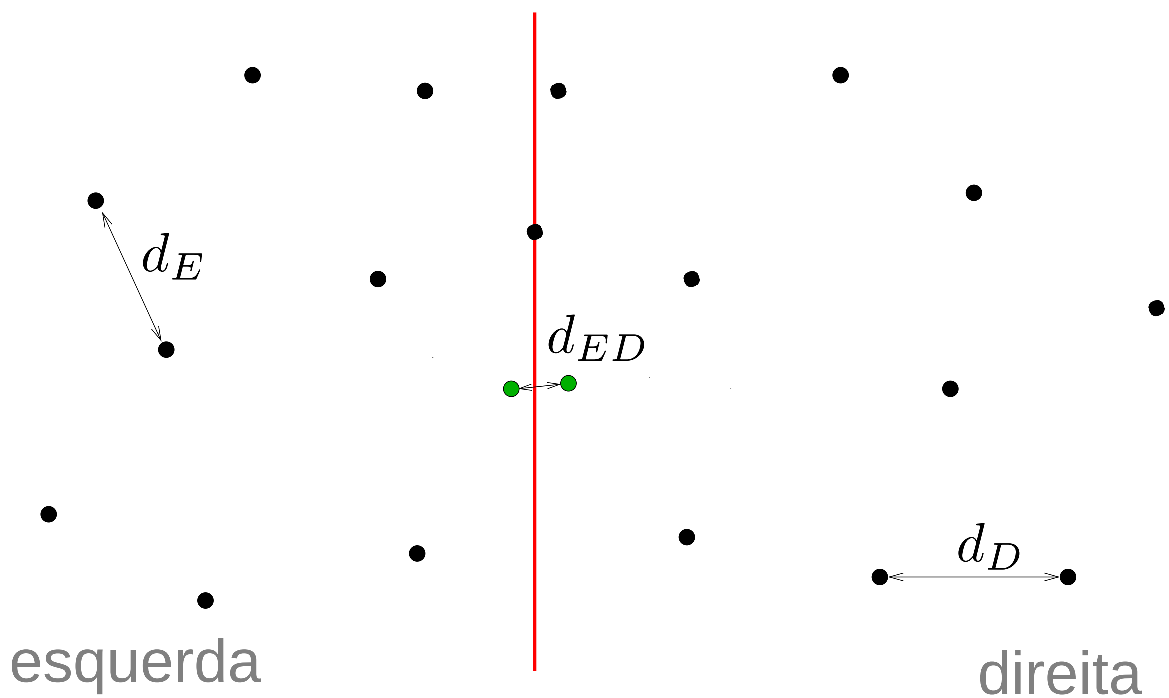


Par mais próximo no plano

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) $\triangleright$ Divisão e conquista

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

Algoritmo de Shamos e Hoey

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

se $r \leq p + 2$

  então $\triangleright$ resolva o problema diretamente

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

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

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

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

Suponha que Combine é linear.

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 $q \leftarrow \lfloor (p+r)/2 \rfloor$
4 $d_E \leftarrow$ DISTÂNCIAREC-SH (X, Y, p, q)
5 $d_D \leftarrow$ DISTÂNCIAREC-SH $(X, Y, q+1, r)$
6 devolva COMBINE (X, Y, p, r, d_E, d_D)

Suponha que COMBINE é linear.

Consumo de tempo do DISTÂNCIAREC-SH:

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

onde $n = r - p + 1$.

Algoritmo de Shamos e Hoey

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

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

Suponha que COMBINE é linear.

Consumo de tempo:

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

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

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

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

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

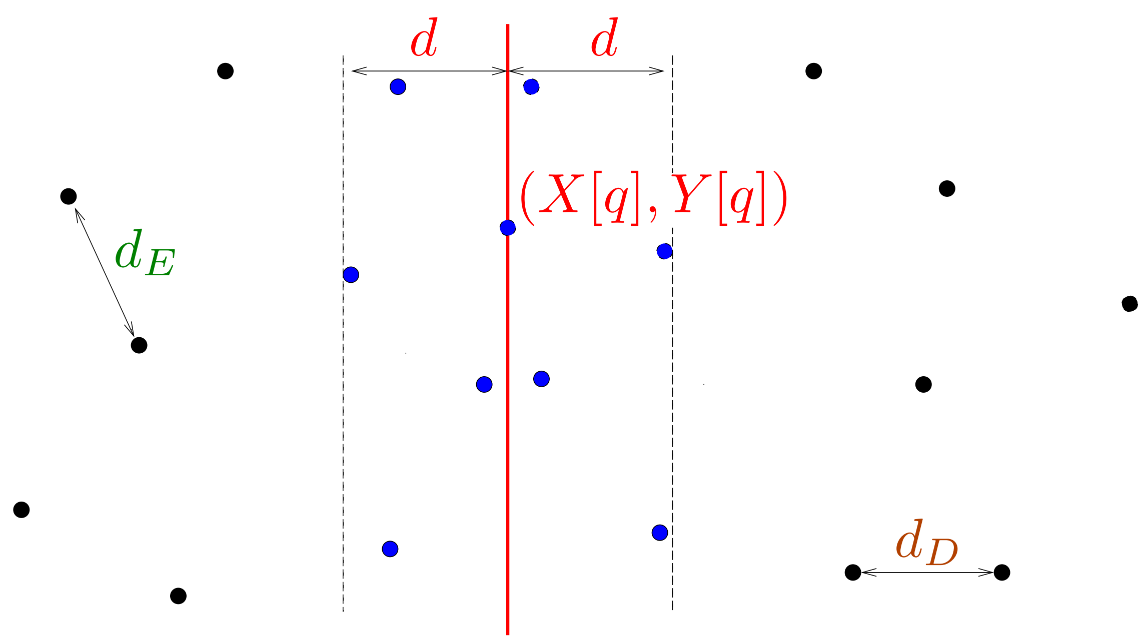
Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

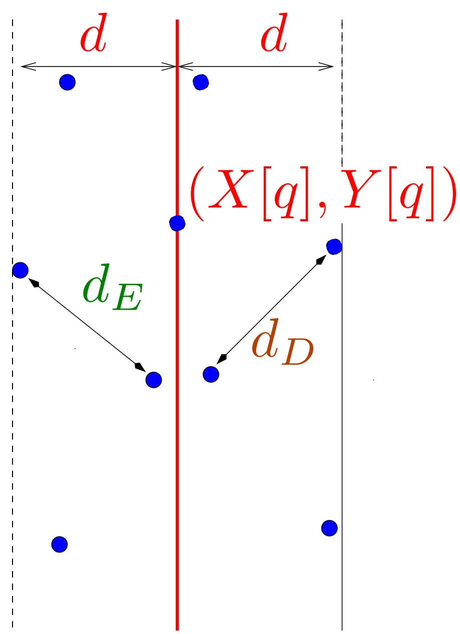
COMBINE precisa considerar apenas pontos que estão a uma distância menor que $d = \min\{d_E, d_D\}$ da reta vertical $x = X[q]$.



Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

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



Infelizmente todos os pontos podem estar nesta faixa...

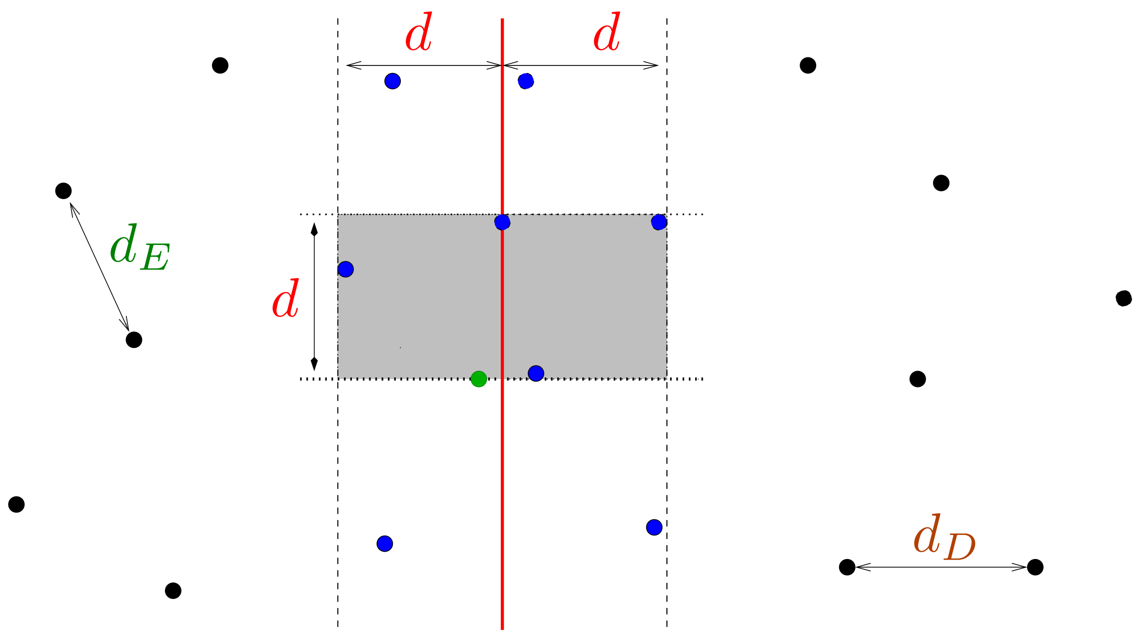
Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

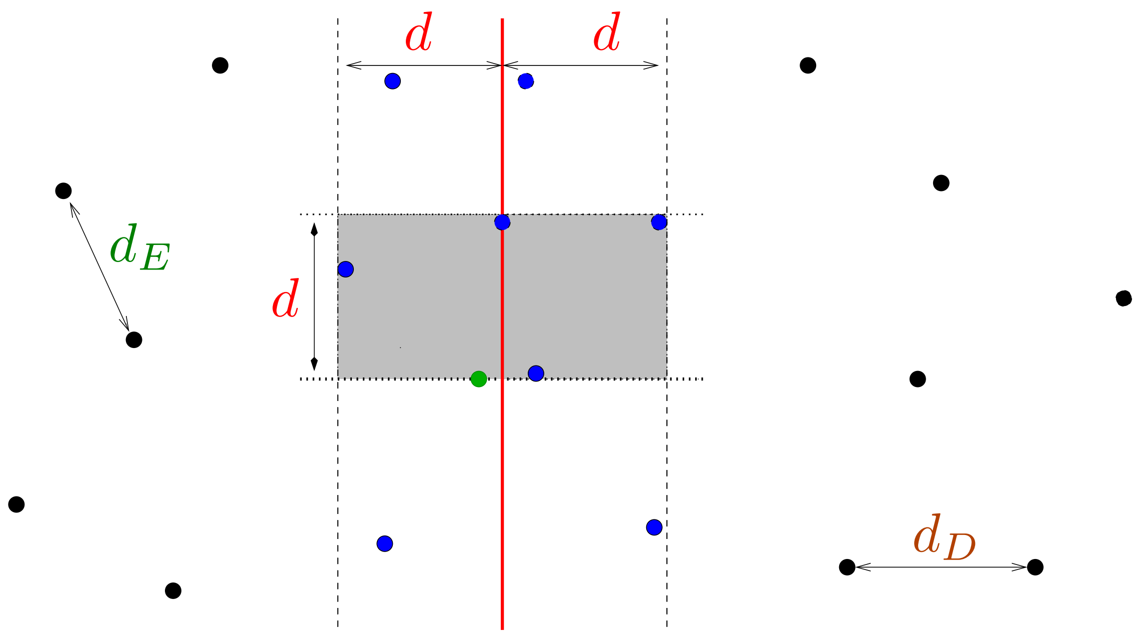
Ideia...



Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

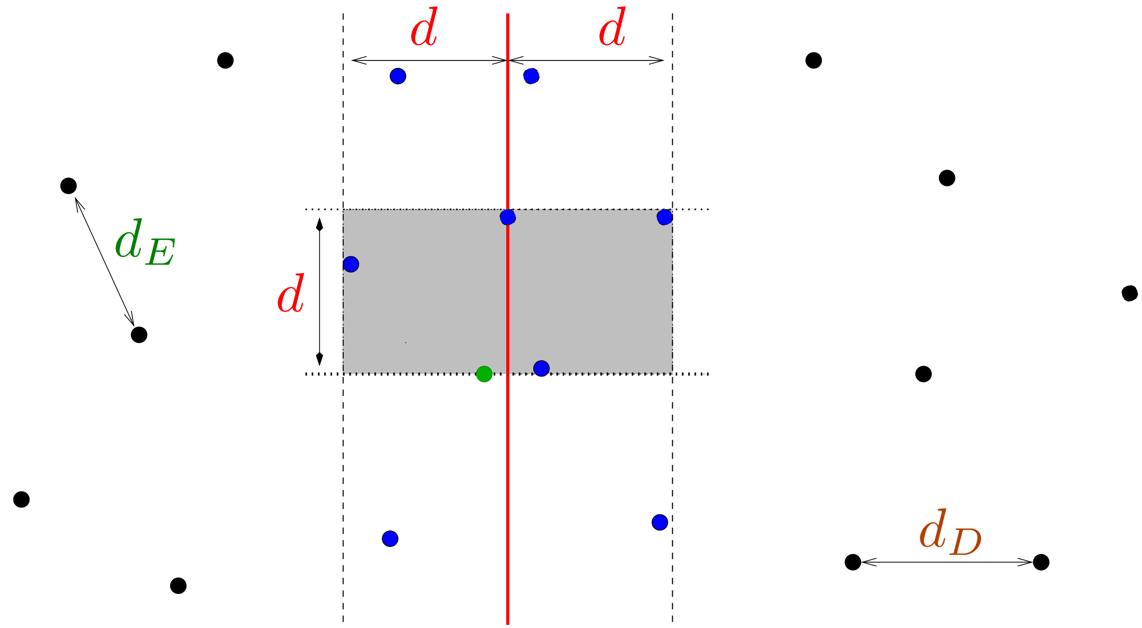
Ideia...



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

Algoritmo de Shamos e Hoey

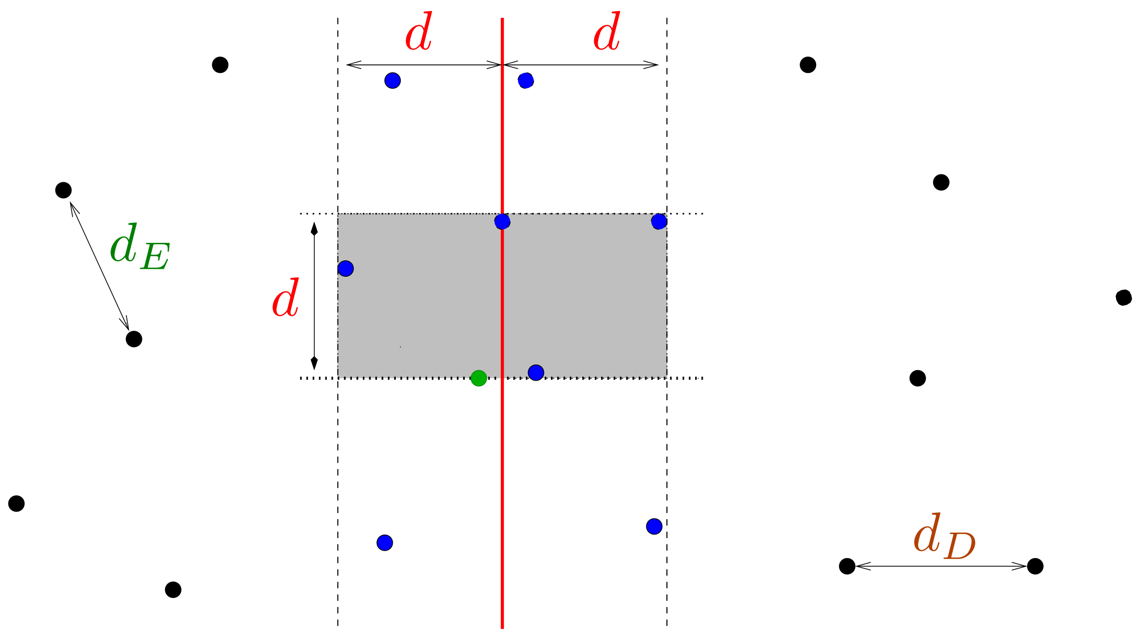
Como fazer o COMBINE linear?



Quantos pontos assim há?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?

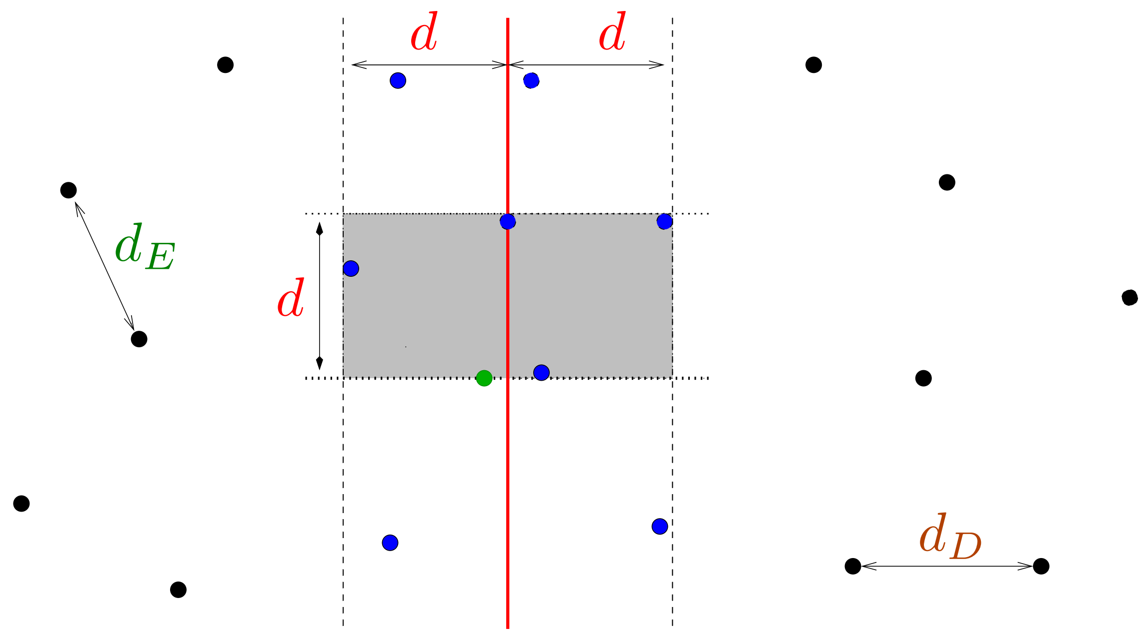


Quantos pontos assim há?

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

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?



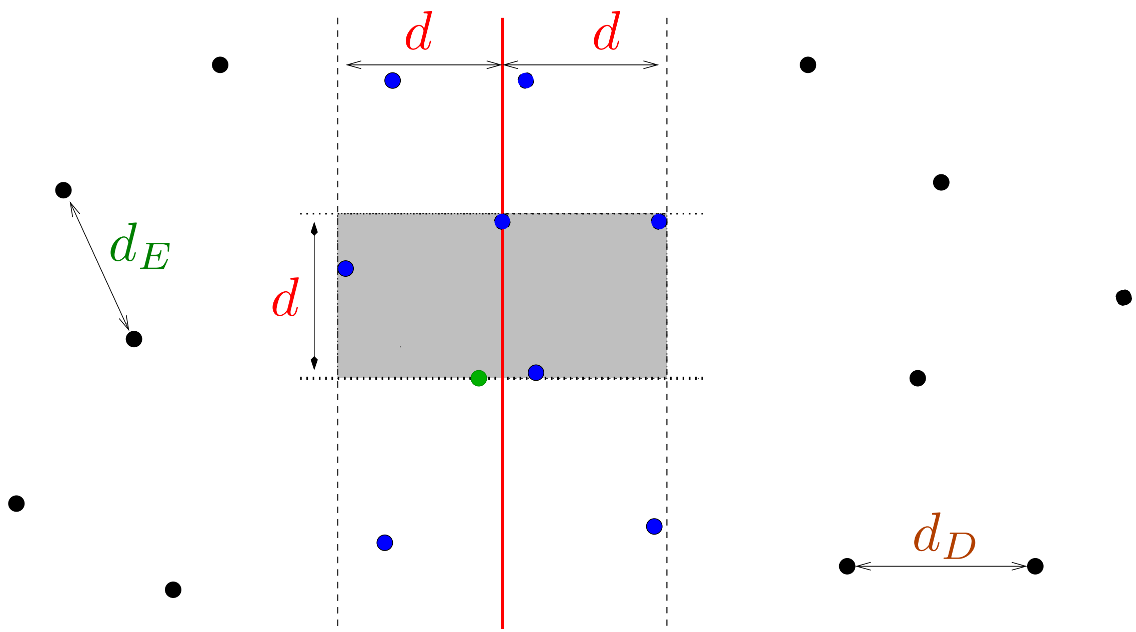
Quantos pontos assim há?

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

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

Algoritmo de Shamos e Hoey

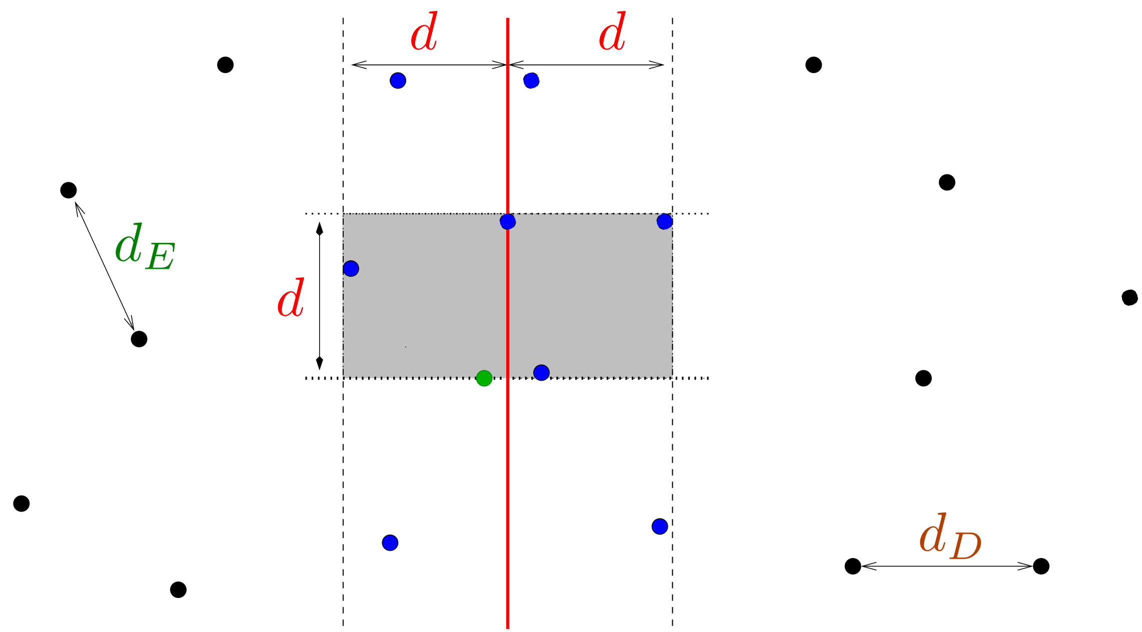
Como fazer o COMBINE linear?



Mas como ter acesso rápido a estes pontos?

Algoritmo de Shamos e Hoey

Como fazer o COMBINE linear?



Mas como ter acesso rápido a estes pontos?

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

Algoritmo de Shamos e Hoey

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

Algoritmo de Shamos e Hoey

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

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

Algoritmo de Shamos e Hoey

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

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

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

Divisão e conquista

DISTÂNCIAREC-SH (X, Y, a, p, r)

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

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

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

Algoritmo de Shamos e Hoey

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

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

Algoritmo de Shamos e Hoey

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

se $r \leq p + 2$

  então $\triangleright$ resolva o problema diretamente

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

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

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

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

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

DIVIDA e COMBINE são algoritmos lineares.

Algoritmo de Shamos e Hoey

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

DIVIDA e COMBINE são algoritmos lineares.

Consumo de tempo:

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

onde $n = r - p + 1$.

Algoritmo de Shamos e Hoey

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

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

DIVIDA e COMBINE são algoritmos lineares.

Consumo de tempo:

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

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

Algoritmo de Shamos e Hoey

Suponha que na coleção
não há dois pontos com a mesma X-coordenada.

Algoritmo de Shamos e Hoey

Suponha que na coleção
não há dois pontos com a mesma X-coordenada.

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

Algoritmo de Shamos e Hoey

Suponha que na coleção
não há dois pontos com a mesma X-coordenada.

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

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

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

Algoritmo de Shamos e Hoey

Suponha que na coleção
não há dois pontos com a mesma X-coordenada.

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

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

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

Algoritmo de Shamos e Hoey

Suponha que na coleção
não há dois pontos com a mesma X-coordenada.

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

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

	1	2	3	4	5
b	3	1			

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

Algoritmo de Shamos e Hoey

Suponha que na coleção
não há dois pontos com a mesma X-coordenada.

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

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

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

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

Algoritmo de Shamos e Hoey

Suponha que na coleção
não há dois pontos com a mesma X-coordenada.

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

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

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

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

Algoritmo de Shamos e Hoey

Suponha que na coleção
não há dois pontos com a mesma X-coordenada.

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

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

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

Consumo de tempo:

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

Algoritmo de Shamos e Hoey

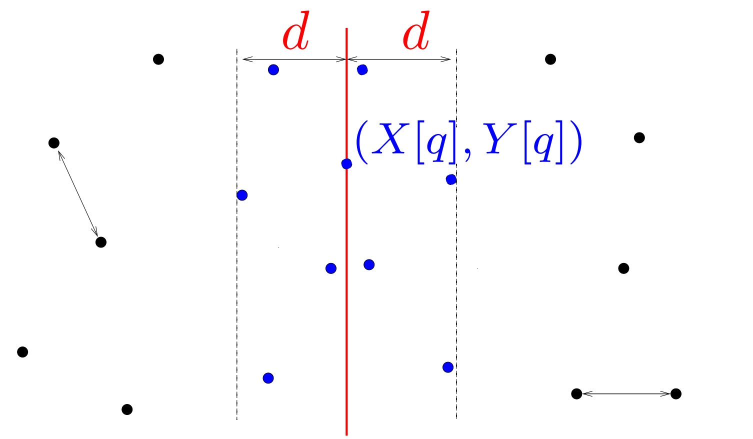
A rotina abaixo identifica os pontos que estão na faixa, ordenados pela Y-coordenada.

Algoritmo de Shamos e Hoey

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

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

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



Algoritmo de Shamos e Hoey

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

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

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

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

f					

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

Algoritmo de Shamos e Hoey

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

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

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

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

	1	2	3
f	3	2	4

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

Algoritmo de Shamos e Hoey

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

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

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

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

f	3	2	4

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

Consumo de tempo:

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

Algoritmo de Shamos e Hoey

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

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

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

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

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

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

    se $d' < d$

      então $d \leftarrow d'$

devolva d

Algoritmo de Shamos e Hoey

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

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

Consumo de tempo:

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

Algoritmo de Shamos e Hoey

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

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

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

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

  $j \leftarrow i + 1$

  enquanto $j \leq t$ e $Y[f[j]] - Y[f[i]] < d$ faça $\triangleright \leq 7$ próximos

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

    se $d' < d$

      então $d \leftarrow d'$

    $j \leftarrow j + 1$

devolva d

Algoritmo de Shamos e Hoey

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

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

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

Quase finalizando...

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

Quase finalizando...

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

Material coberto nesta aula: Secs 7.1 – 7.3.

Quase finalizando...

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

Material coberto nesta aula: Secs 7.1 – 7.3.

Agora, às animações!