

Geometria Computacional

Cristina G. Fernandes

Departamento de Ciência da Computação do IME-USP

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

segundo semestre de 2011

Interseção de segmentos

Uma coleção de segmentos do plano é dada por dois vetores $e[1..n], d[1..n]$ de pontos.

Interseção de segmentos

Uma coleção de segmentos do plano é dada por dois vetores $e[1..n], d[1..n]$ de pontos.

A coordenada do ponto $e[i]$ é $(e_X[i], e_Y[i])$.

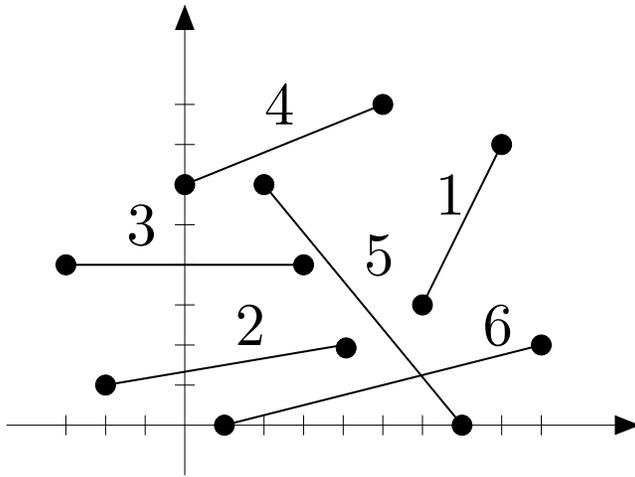
A coordenada do ponto $d[i]$ é $(d_X[i], d_Y[i])$.

Interseção de segmentos

Uma coleção de segmentos do plano é dada por dois vetores $e[1..n], d[1..n]$ de pontos.

A coordenada do ponto $e[i]$ é $(e_X[i], e_Y[i])$.

A coordenada do ponto $d[i]$ é $(d_X[i], d_Y[i])$.

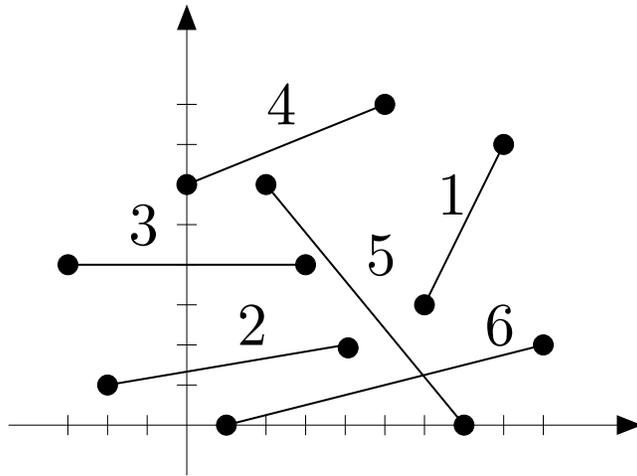


e_X	6	-2	-3	0	2	1
e_Y	3	1	4	6	6	0
	1	2	3	4	5	6

d_X	8	4	3	5	7	9
d_Y	7	2	4	8	0	2
	1	2	3	4	5	6

Interseção de segmentos

Problema: Dada uma coleção de segmentos no plano, decidir se existem dois segmentos na coleção que se intersectam.

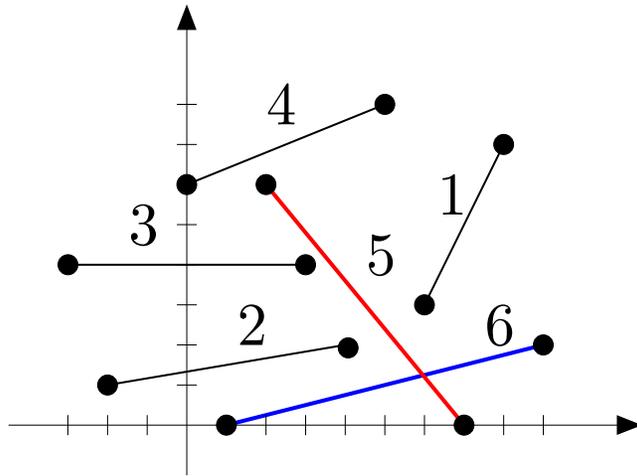


e_X	6	-2	-3	0	2	1
e_Y	3	1	4	6	6	0
	1	2	3	4	5	6

d_X	8	4	3	5	7	9
d_Y	7	2	4	8	0	2
	1	2	3	4	5	6

Interseção de segmentos

Problema: Dada uma coleção de segmentos no plano, decidir se existem dois segmentos na coleção que se intersectam.

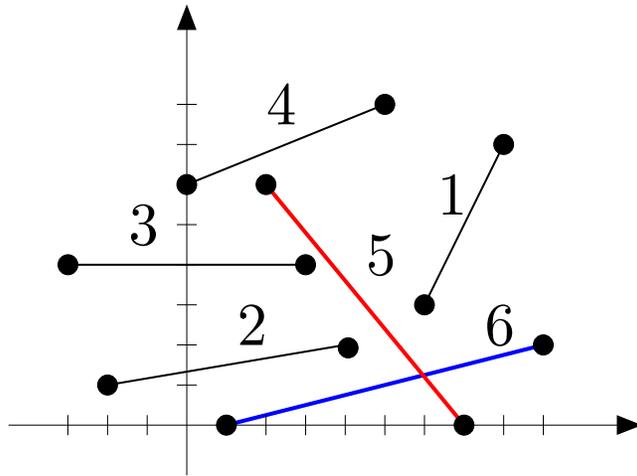


e_X	6	-2	-3	0	2	1
e_Y	3	1	4	6	6	0
	1	2	3	4	5	6

d_X	8	4	3	5	7	9
d_Y	7	2	4	8	0	2
	1	2	3	4	5	6

Interseção de segmentos

Problema: Dada uma coleção de segmentos no plano, decidir se existem dois segmentos na coleção que se intersectam.



e_X	6	-2	-3	0	2	1
e_Y	3	1	4	6	6	0
	1	2	3	4	5	6

d_X	8	4	3	5	7	9
d_Y	7	2	4	8	0	2
	1	2	3	4	5	6

Resposta: sim, existem dois segmentos com interseção.

Interseção de dois segmentos

Das aulas passadas...

Interseção entre ab e cd

INTERSECTA(a, b, c, d)

- 1 se **IntersectaProp**(a, b, c, d)
- 2 então devolva VERDADE
- 3 devolva **Entre**(a, b, c) ou **Entre**(a, b, d)
ou **Entre**(c, d, a) ou **Entre**(c, d, b)

Interseção de dois segmentos

Das aulas passadas...

Interseção entre ab e cd

INTERSECTA(a, b, c, d)

- 1 se **IntersectaProp**(a, b, c, d)
- 2 então devolva VERDADE
- 3 devolva **Entre**(a, b, c) ou **Entre**(a, b, d)
ou **Entre**(c, d, a) ou **Entre**(c, d, b)

Abreviatura:

INTER(e, d, i, j)

- 1 devolva INTERSECTA($e[i], d[i], e[j], d[j]$)

Interseção de segmentos

Solução quadrática:

IntersectaQuad(e, d, n)

- 1 para $i \leftarrow 1$ até $n-1$ faça
- 2 para $j \leftarrow i+1$ até n faça
- 3 se INTER (e, d, i, j)
- 4 então devolva VERDADE
- 5 devolva FALSO

Interseção de segmentos

Solução quadrática:

IntersectaQuad(e, d, n)

- 1 para $i \leftarrow 1$ até $n-1$ faça
- 2 para $j \leftarrow i+1$ até n faça
- 3 se INTER (e, d, i, j)
- 4 então devolva VERDADE
- 5 devolva FALSO

Consumo de tempo: $\Theta(n^2)$.

Interseção de segmentos

Solução quadrática:

IntersectaQuad(e, d, n)

- 1 para $i \leftarrow 1$ até $n-1$ faça
- 2 para $j \leftarrow i+1$ até n faça
- 3 se INTER (e, d, i, j)
- 4 então devolva VERDADE
- 5 devolva FALSO

Consumo de tempo: $\Theta(n^2)$.

Conseguimos fazer melhor que isso?

Interseção de intervalos

Este é o caso **na reta**.

Interseção de intervalos

Este é o caso **na reta**.

Um segmento na reta é um **intervalo**.

Interseção de intervalos

Este é o caso **na reta**.

Um segmento na reta é um **intervalo**.

Os vetores $e_X[1..n]$ e $d_X[1..n]$ representam os intervalos $[e_X[1]..d_X[1]], \dots, [e_X[n]..d_X[n]]$.

Interseção de intervalos

Este é o caso **na reta**.

Um segmento na reta é um **intervalo**.

Os vetores $e_X[1..n]$ e $d_X[1..n]$ representam os intervalos $[e_X[1]..d_X[1]], \dots, [e_X[n]..d_X[n]]$.

Se **ordenarmos os pontos extremos dos intervalos**, é fácil decidir se há interseção ou não, percorrendo os pontos na ordem obtida.

Interseção de intervalos

Este é o caso **na reta**.

Um segmento na reta é um **intervalo**.

Os vetores $e_X[1..n]$ e $d_X[1..n]$ representam os intervalos $[e_X[1]..d_X[1]], \dots, [e_X[n]..d_X[n]]$.

Se **ordenarmos os pontos extremos dos intervalos**, é fácil decidir se há interseção ou não, percorrendo os pontos na ordem obtida.

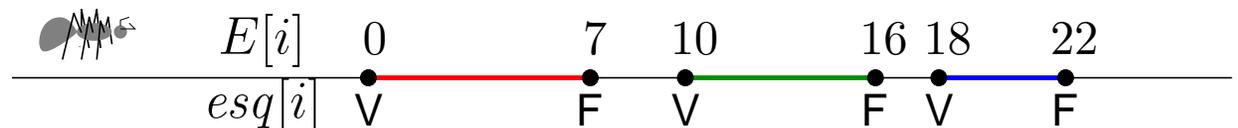
Basta **contar quantos intervalos estão “abertos”**. Se houver mais do que um aberto num momento, há interseção.

Interseção de intervalos

VARREDURA(e, d, n)

- 1 para $i \leftarrow 1$ até n faça ▷ para cada intervalo marca
- 2 $E[i] \leftarrow e_X[i]$ $esq[i] \leftarrow$ VERDADE ▷ extremo esquerdo
- 3 $E[i + n] \leftarrow d_X[i]$ $esq[i + n] \leftarrow$ FALSO ▷ extremo direito
- 4 MERGESORT($E, esq, 1, 2n$) ▷ ordena os extremos

e_X	10	0	18
d_X	16	7	22
	1	2	3



Interseção de intervalos

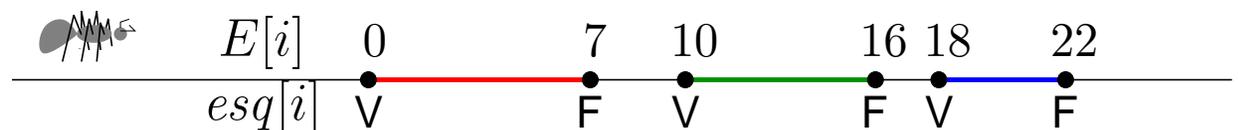
VARREDURA(e, d, n)

```

1  para  $i \leftarrow 1$  até  $n$  faça           ▷ para cada intervalo marca
2       $E[i] \leftarrow e_X[i]$        $esq[i] \leftarrow$  VERDADE   ▷ extremo esquerdo
3       $E[i + n] \leftarrow d_X[i]$    $esq[i + n] \leftarrow$  FALSO  ▷ extremo direito
4  MERGESORT( $E, esq, 1, 2n$ ) ▷ ordena os extremos
5   $cont \leftarrow 0$     $resp \leftarrow$  FALSO
6  para  $p \leftarrow 1$  até  $2n$  faça       ▷ para cada ponto extremo
7      se  $esq[p]$                                ▷ se extremo esquerdo
8          então  $cont \leftarrow cont + 1$ 
9              se  $cont = 2$  então  $resp \leftarrow$  VERDADE
10         senão  $cont \leftarrow cont - 1$ 
11 devolva  $resp$ 

```

e_X	10	0	18
d_X	16	7	22
	1	2	3



Interseção de intervalos

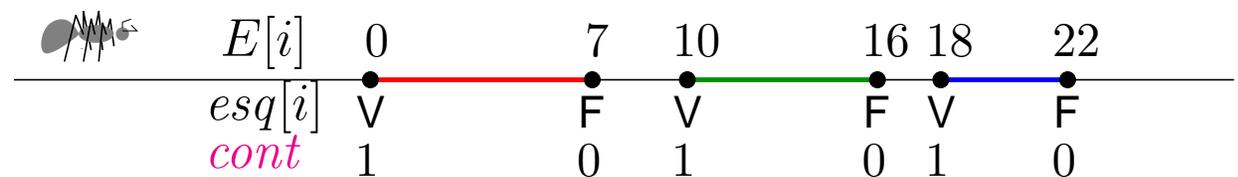
VARREDURA(e, d, n)

```

1  para  $i \leftarrow 1$  até  $n$  faça           ▷ para cada intervalo marca
2     $E[i] \leftarrow e_X[i]$        $esq[i] \leftarrow$  VERDADE   ▷ extremo esquerdo
3     $E[i + n] \leftarrow d_X[i]$    $esq[i + n] \leftarrow$  FALSO  ▷ extremo direito
4  MERGESORT( $E, esq, 1, 2n$ ) ▷ ordena os extremos
5   $cont \leftarrow 0$     $resp \leftarrow$  FALSO
6  para  $p \leftarrow 1$  até  $2n$  faça       ▷ para cada ponto extremo
7    se  $esq[p]$                                ▷ se extremo esquerdo
8      então  $cont \leftarrow cont + 1$ 
9      se  $cont = 2$  então  $resp \leftarrow$  VERDADE
10     senão  $cont \leftarrow cont - 1$ 
11 devolva  $resp$ 

```

e_X	10	0	18
d_X	16	7	22
	1	2	3



VARREDURA($e_X, d_X, 3$) = FALSO

Interseção de intervalos

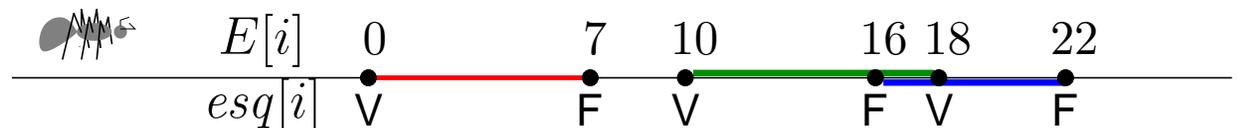
VARREDURA(e, d, n)

```

1  para  $i \leftarrow 1$  até  $n$  faça           ▷ para cada intervalo marca
2     $E[i] \leftarrow e_X[i]$        $esq[i] \leftarrow$  VERDADE   ▷ extremo esquerdo
3     $E[i + n] \leftarrow d_X[i]$   $esq[i + n] \leftarrow$  FALSO  ▷ extremo direito
4  MERGESORT( $E, esq, 1, 2n$ ) ▷ ordena os extremos
5   $cont \leftarrow 0$     $resp \leftarrow$  FALSO
6  para  $p \leftarrow 1$  até  $2n$  faça       ▷ para cada ponto extremo
7    se  $esq[p]$                                ▷ se extremo esquerdo
8      então  $cont \leftarrow cont + 1$ 
9      se  $cont = 2$  então  $resp \leftarrow$  VERDADE
10     senão  $cont \leftarrow cont - 1$ 
11 devolva  $resp$ 

```

e_X	10	0	16
d_X	18	7	22
	1	2	3



Interseção de intervalos

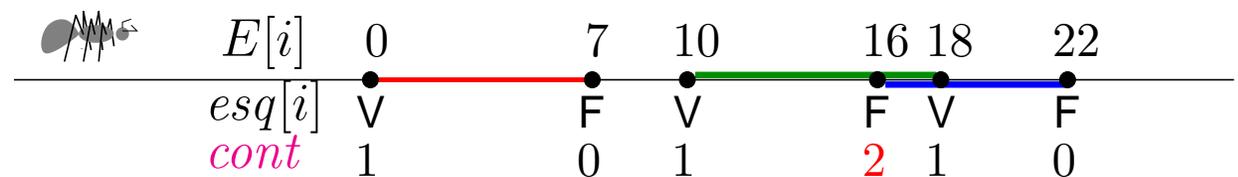
VARREDURA(e, d, n)

```

1  para  $i \leftarrow 1$  até  $n$  faça           ▷ para cada intervalo marca
2       $E[i] \leftarrow e_X[i]$        $esq[i] \leftarrow$  VERDADE   ▷ extremo esquerdo
3       $E[i + n] \leftarrow d_X[i]$    $esq[i + n] \leftarrow$  FALSO  ▷ extremo direito
4  MERGESORT( $E, esq, 1, 2n$ ) ▷ ordena os extremos
5   $cont \leftarrow 0$     $resp \leftarrow$  FALSO
6  para  $p \leftarrow 1$  até  $2n$  faça       ▷ para cada ponto extremo
7      se  $esq[p]$                                ▷ se extremo esquerdo
8          então  $cont \leftarrow cont + 1$ 
9              se  $cont = 2$  então  $resp \leftarrow$  VERDADE
10         senão  $cont \leftarrow cont - 1$ 
11 devolva  $resp$ 

```

e_X	10	0	16
d_X	18	7	22
	1	2	3



VARREDURA($e_X, d_X, 3$) = VERDADE

Interseção de intervalos

VARREDURA(e, d, n)

```
1  para  $i \leftarrow 1$  até  $n$  faça           ▷ para cada intervalo marca
2     $E[i] \leftarrow e_X[i]$        $esq[i] \leftarrow$  VERDADE   ▷ extremo esquerdo
3     $E[i + n] \leftarrow d_X[i]$   $esq[i + n] \leftarrow$  FALSO  ▷ extremo direito
4  MERGESORT( $E, esq, 1, 2n$ ) ▷ ordena os extremos
5   $cont \leftarrow 0$     $resp \leftarrow$  FALSO
6  para  $p \leftarrow 1$  até  $2n$  faça       ▷ para cada ponto extremo
7    se  $esq[p]$                                ▷ se extremo esquerdo
8      então  $cont \leftarrow cont + 1$ 
9          se  $cont = 2$  então  $resp \leftarrow$  VERDADE
10     senão  $cont \leftarrow cont - 1$ 
11  devolva  $resp$ 
```

Consumo de tempo: $\Theta(n \lg n)$.

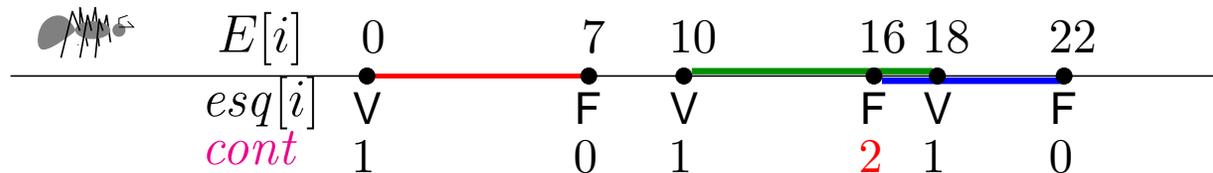
Método da linha de varredura

Ideia: reduzir um problema estático bidimensional a um problema dinâmico unidimensional

Método da linha de varredura

Ideia: reduzir um problema estático bidimensional a um problema dinâmico unidimensional

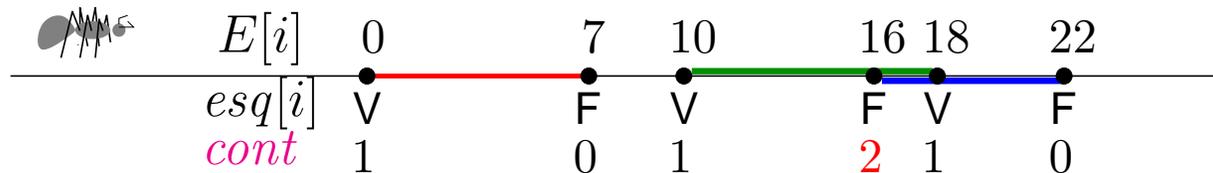
Uma **linha imaginária** move-se da esquerda para a direita.



Método da linha de varredura

Ideia: reduzir um problema estático bidimensional a um problema dinâmico unidimensional

Uma **linha imaginária** move-se da esquerda para a direita.

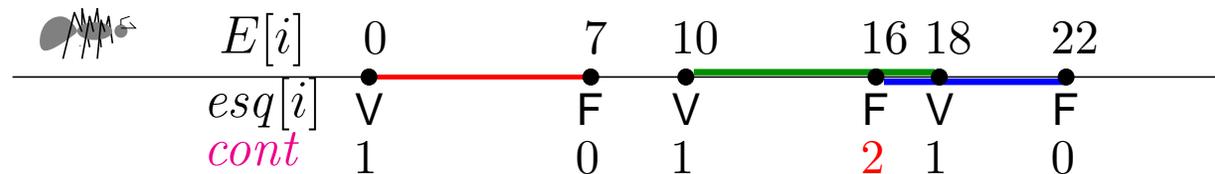


À medida que ela move,
o problema restrito à esquerda dela é resolvido.

Método da linha de varredura

Ideia: reduzir um problema estático bidimensional a um problema dinâmico unidimensional

Uma **linha imaginária** move-se da esquerda para a direita.



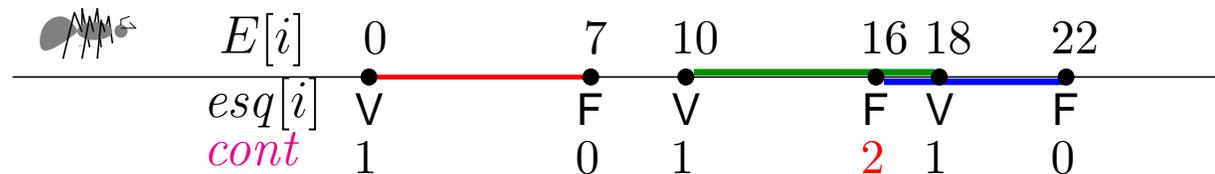
À medida que ela move,
o **problema restrito à esquerda dela é resolvido.**

Informação necessária para estender a solução parcial é mantida numa **descrição combinatória da linha.**

Método da linha de varredura

Ideia: reduzir um problema estático bidimensional a um problema dinâmico unidimensional

Uma **linha imaginária** move-se da esquerda para a direita.



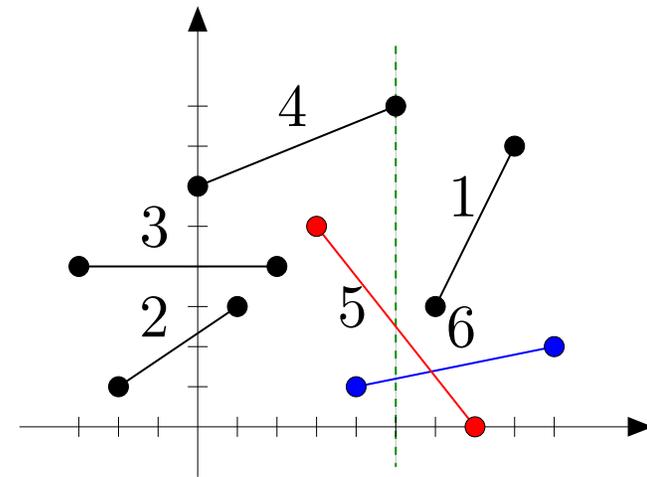
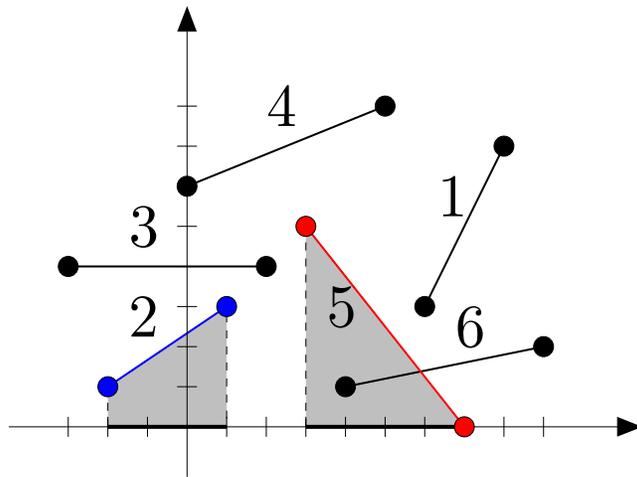
À medida que ela move,
o **problema restrito à esquerda dela é resolvido.**

Informação necessária para estender a solução parcial é mantida numa **descrição combinatória da linha.**

Muda apenas em posições chaves: os **pontos eventos.**

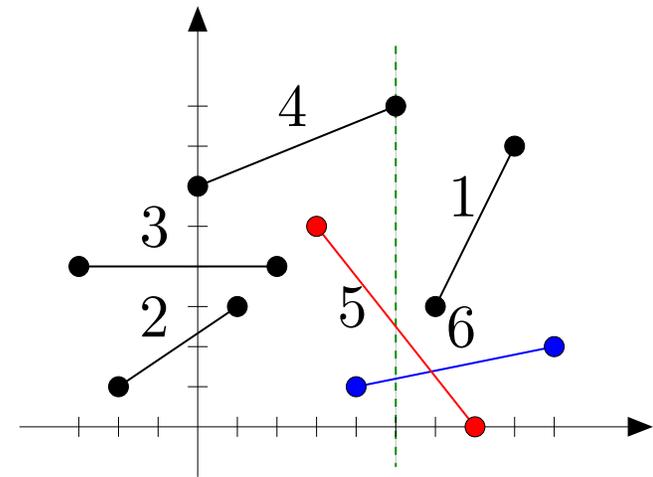
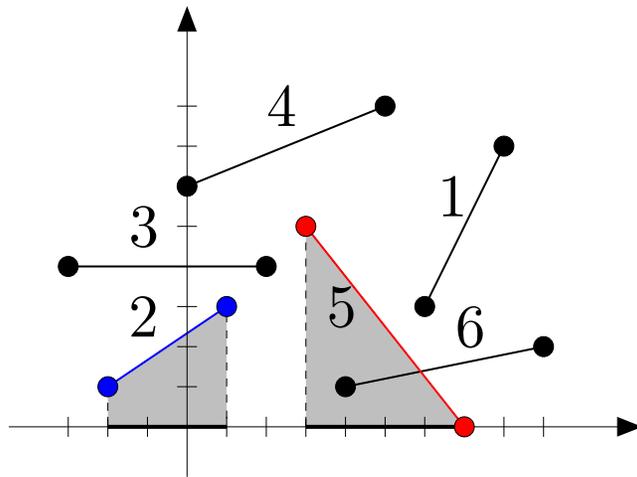
Algoritmo de Shamos e Hoey

Ideia: Dois segmentos cuja projeção no eixo X sejam disjuntas não se intersectam.



Algoritmo de Shamos e Hoey

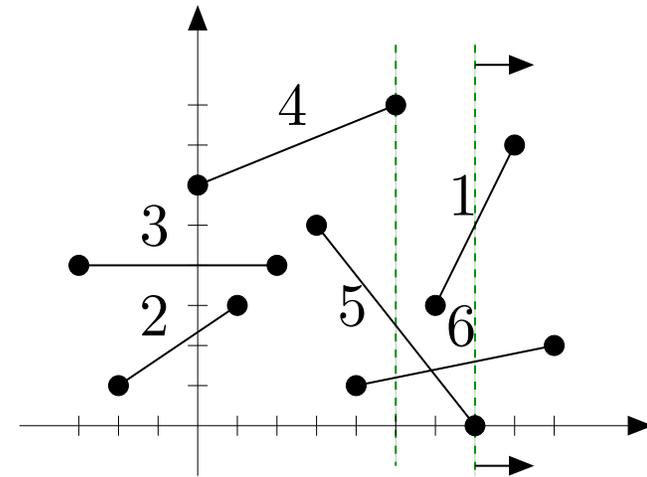
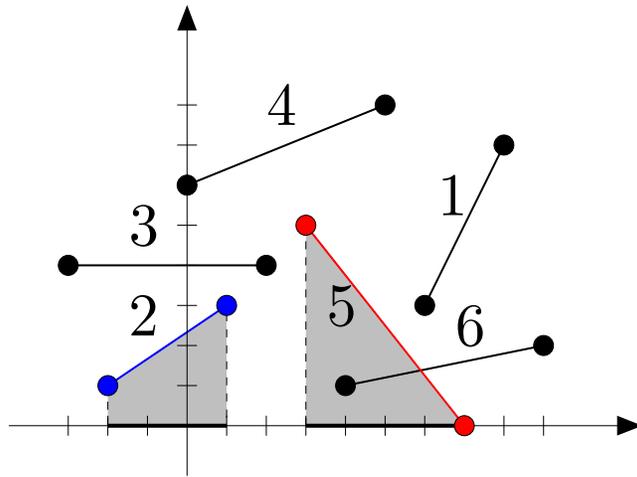
Ideia: Dois segmentos cuja projeção no eixo X sejam disjuntas não se intersectam.



Se a projeção no eixo X de dois segmentos tem interseção, então há uma **linha vertical** que intersecta ambos.

Algoritmo de Shamos e Hoey

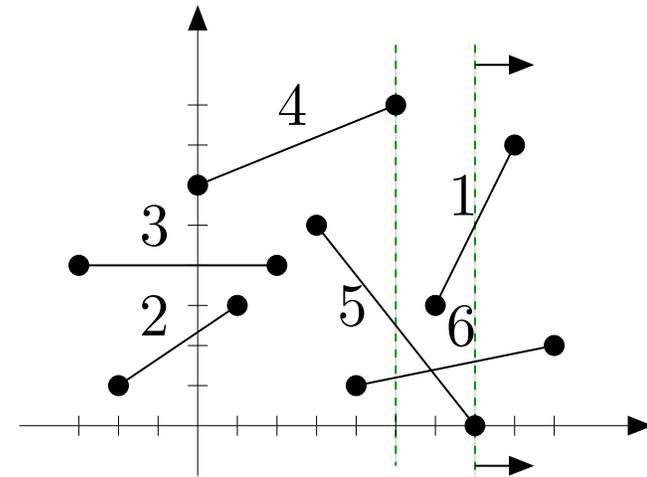
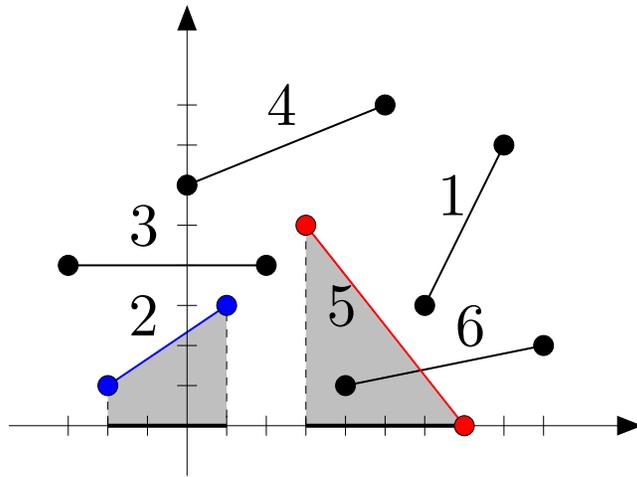
Ideia: Dois segmentos cuja projeção no eixo X sejam disjuntas não se intersectam.



Imagine esta **linha vertical** varrendo o plano da esquerda para a direita...

Algoritmo de Shamos e Hoey

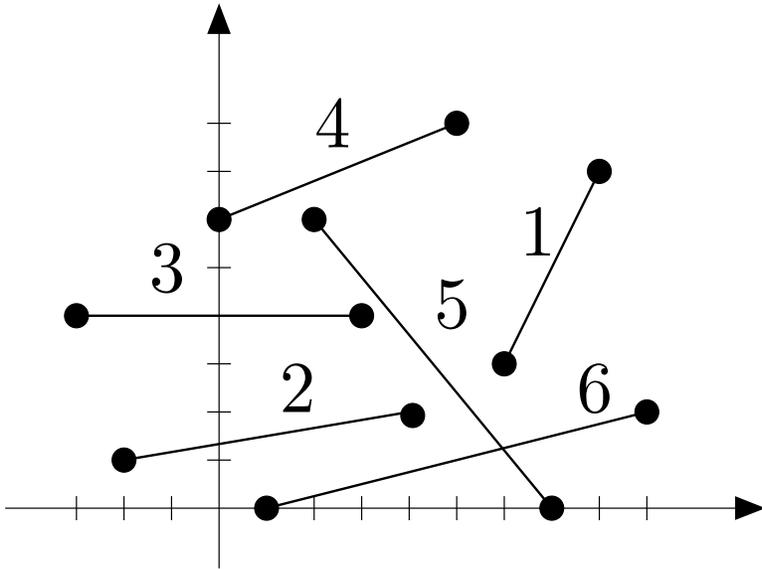
Ideia: Dois segmentos cuja projeção no eixo X sejam disjuntas não se intersectam.



Imagine esta **linha vertical** varrendo o plano da esquerda para a direita...

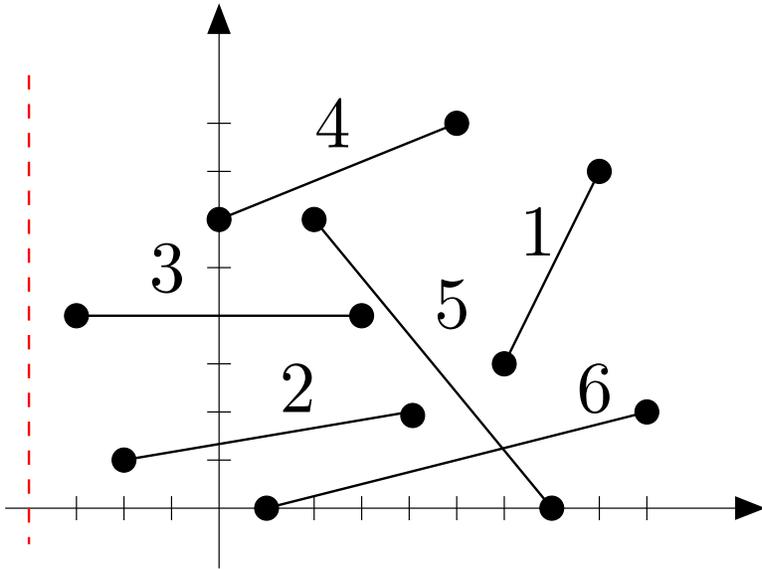
Enquanto a **linha** varre o plano, mantemos os segmentos intersectados por ela na **descrição combinatória da linha**.

Descrição combinatória da linha



$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

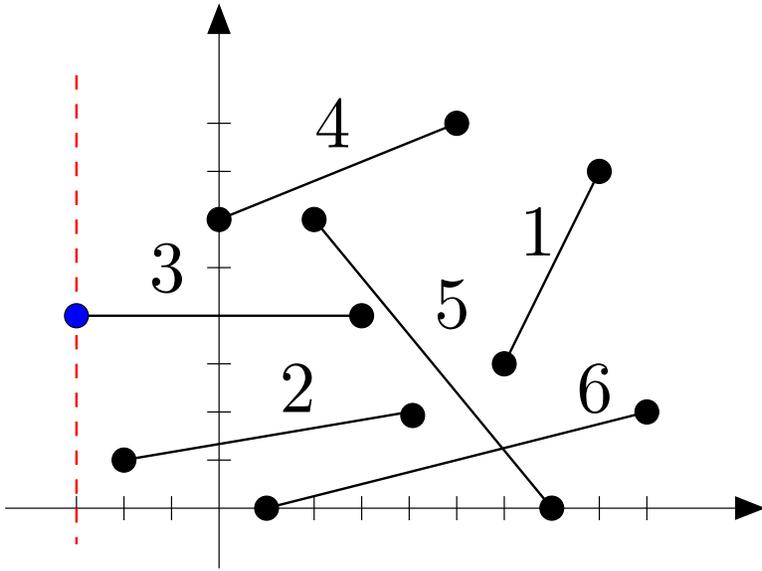
Descrição combinatória da linha



Alterações ocorrem
nos **extremos dos segmentos.**

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{3, 4\}$
$2 < x < 3$	$\{4\}$
$3 \leq x < 4$	$\{4, 5\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

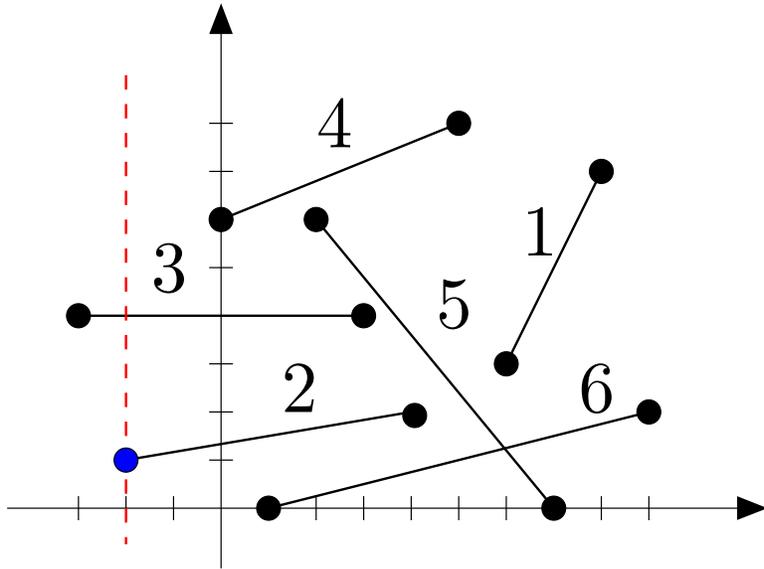


Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

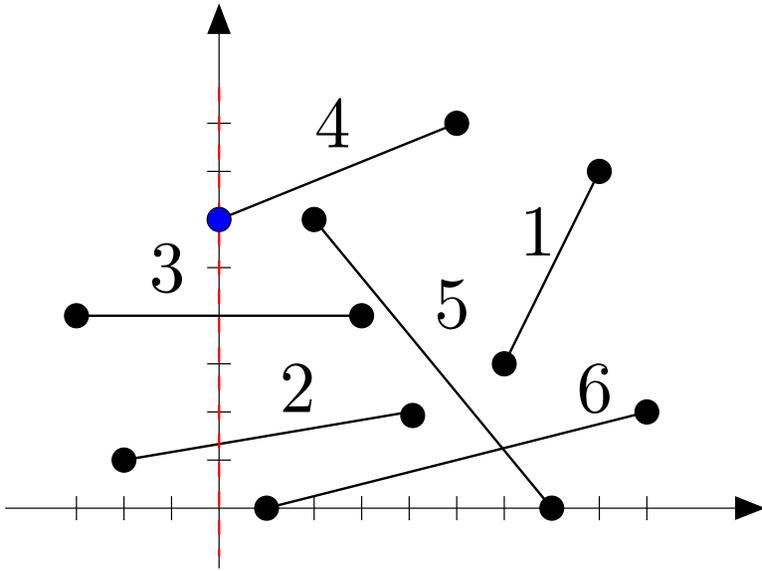


$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Descrição combinatória da linha

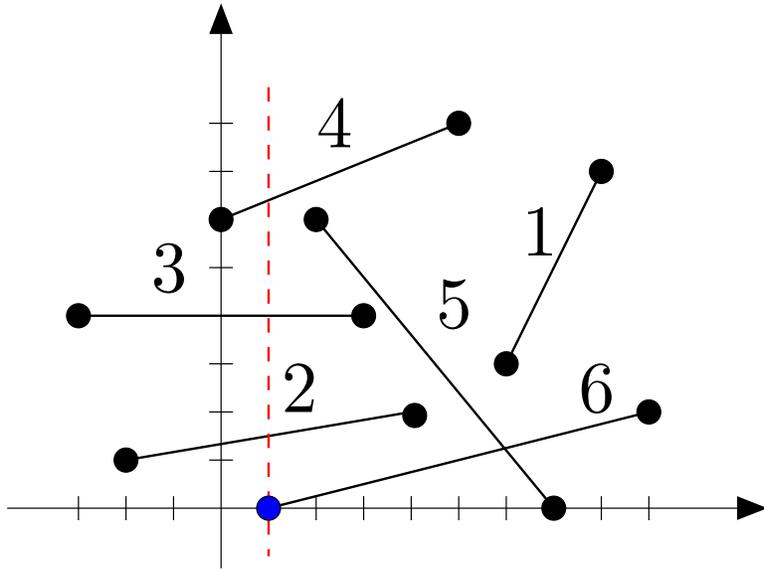


Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

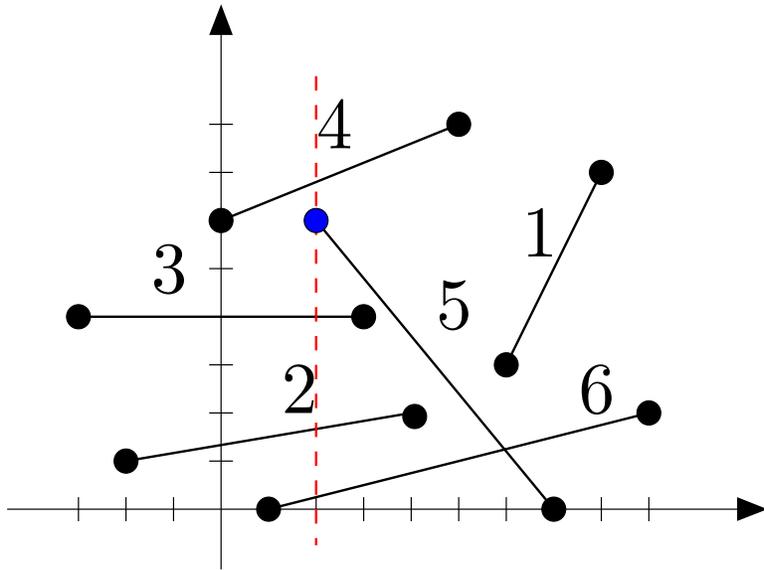


Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

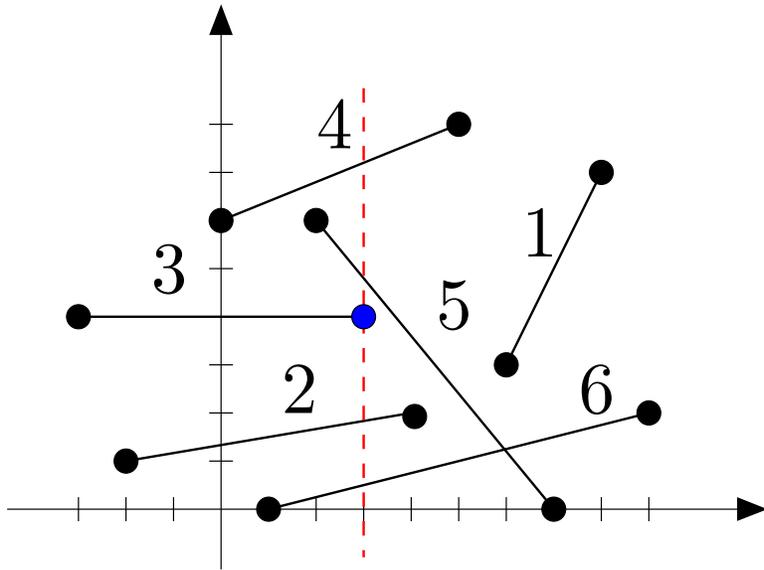


Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

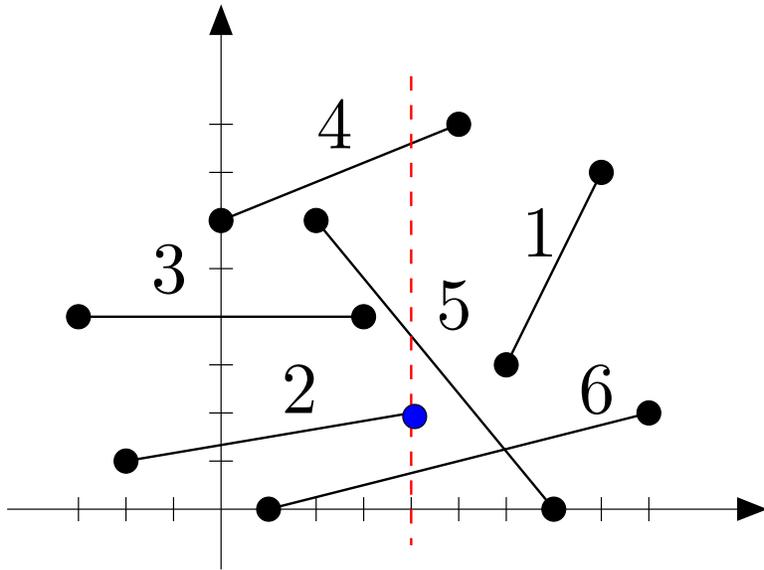


Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

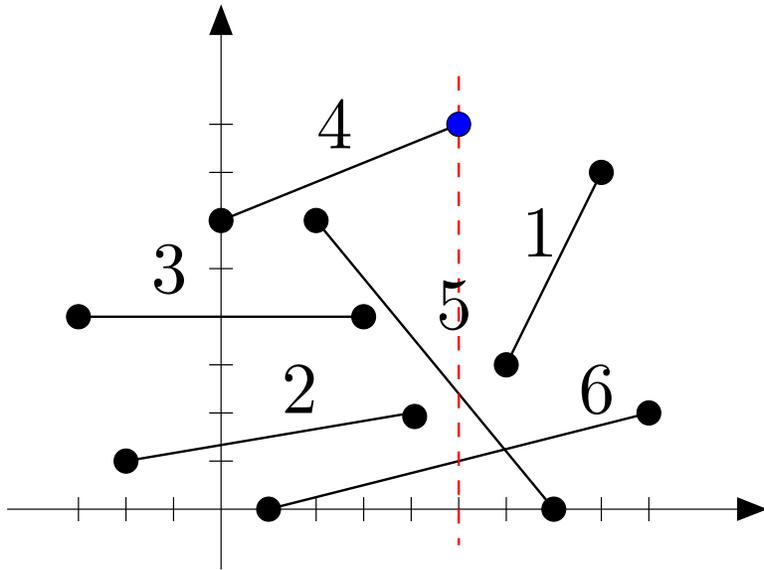


Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

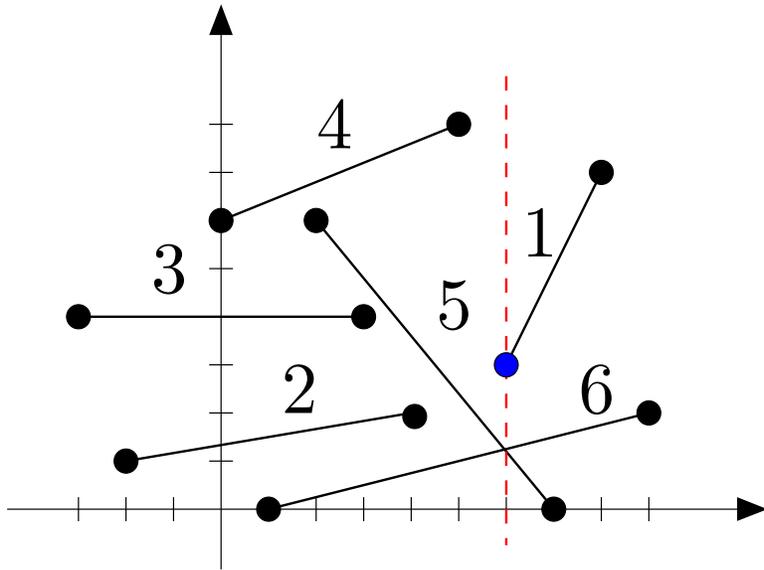


Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

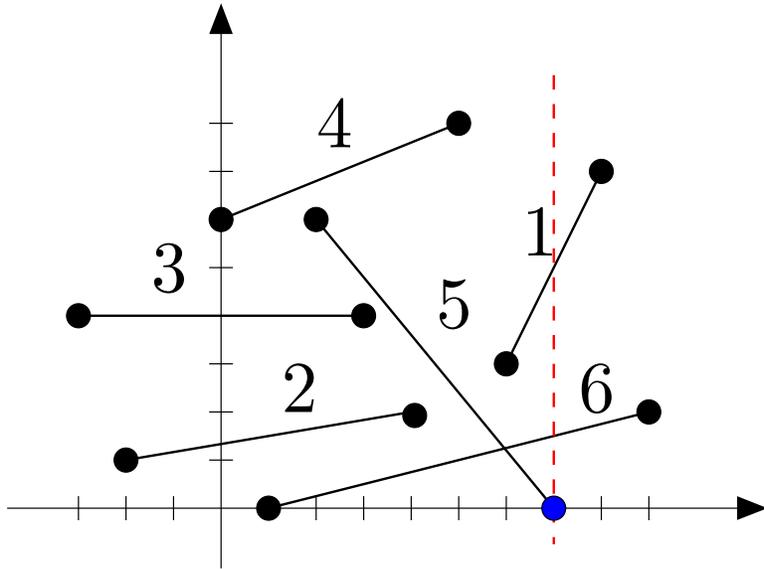


$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Descrição combinatória da linha

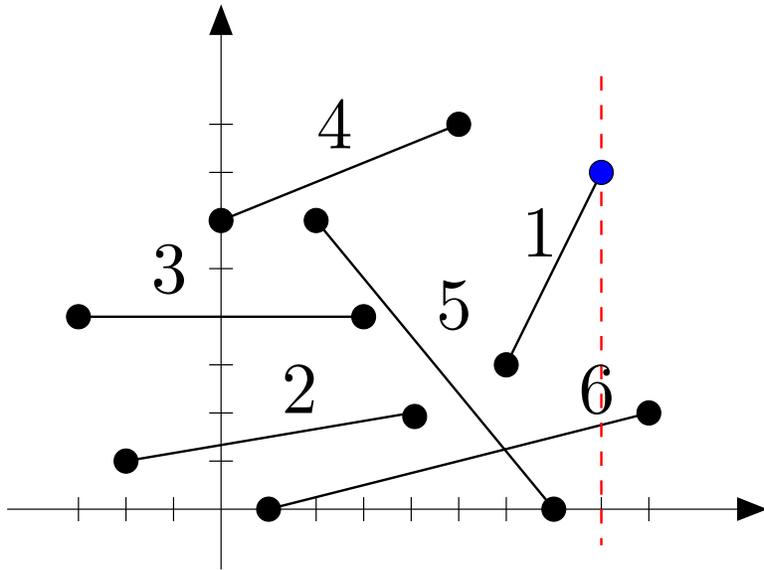


Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

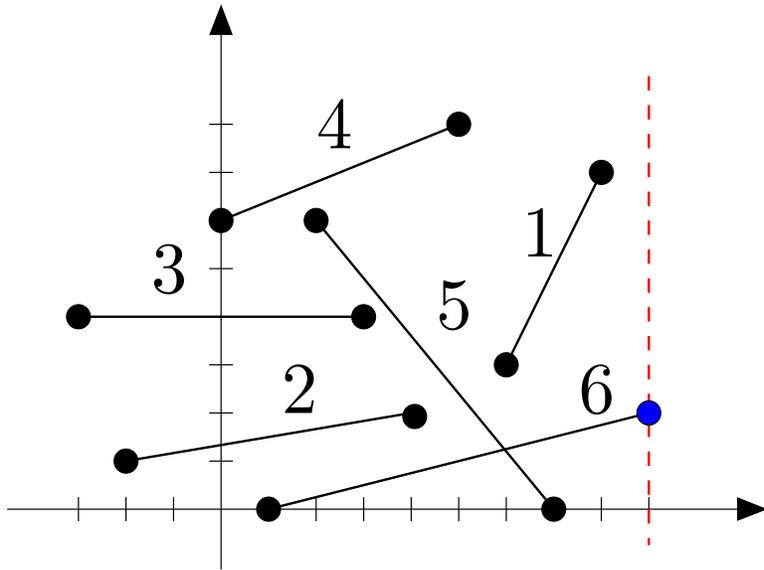


Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

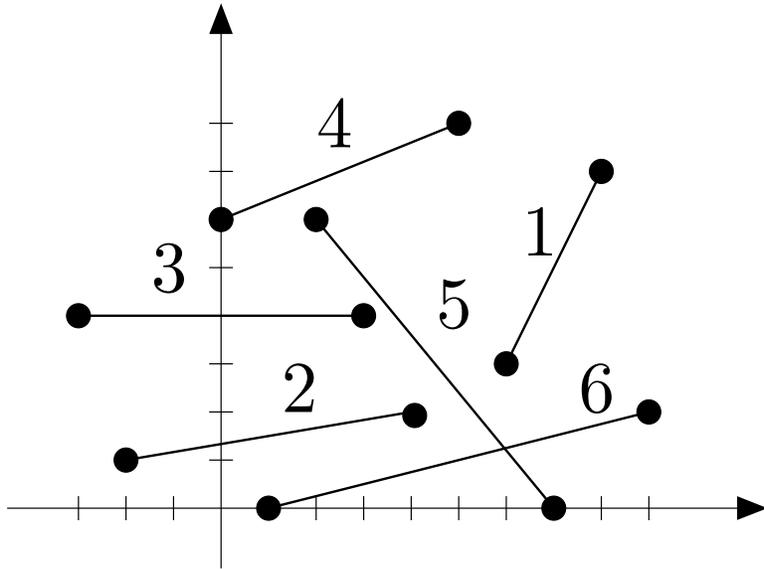


Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

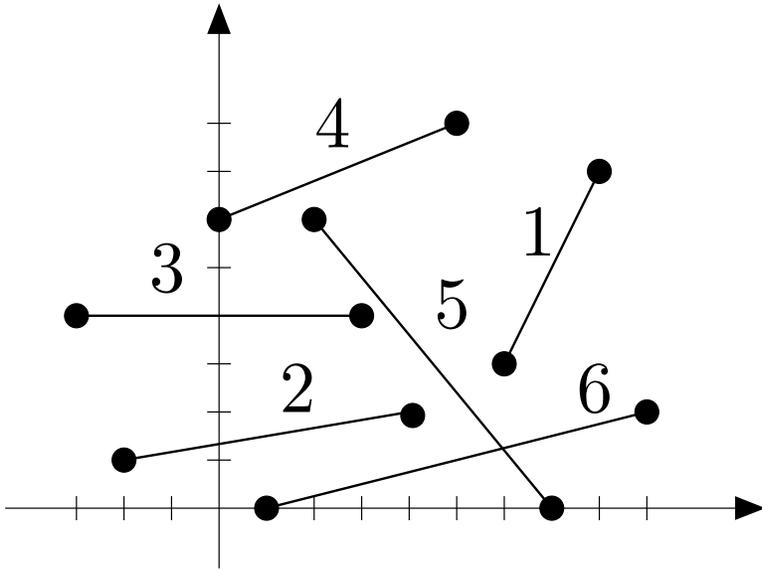
Descrição combinatória da linha



Como guardar
um destes conjuntos?

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha



Como guardar
um destes conjuntos?

Que operações ele sofre?

$x < -3$	\emptyset
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$	$\{1, 6\}$
$8 < x \leq 9$	$\{6\}$
$9 < x$	\emptyset

Descrição combinatória da linha

O conjunto dos segmentos na linha sofre **inserções** e **remoções**.

Descrição combinatória da linha

O conjunto dos segmentos na linha sofre **inserções** e **remoções**.

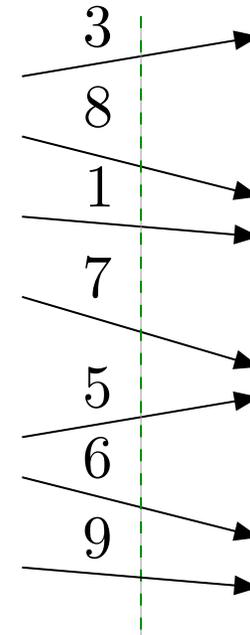
Como a linha vai nos ajudar a detectar interseção?

Descrição combinatória da linha

O conjunto dos segmentos na linha sofre **inserções** e **remoções**.

Como a linha vai nos ajudar a detectar interseção?

Ideia: testar interseção apenas entre segmentos “vizinhos na linha”.



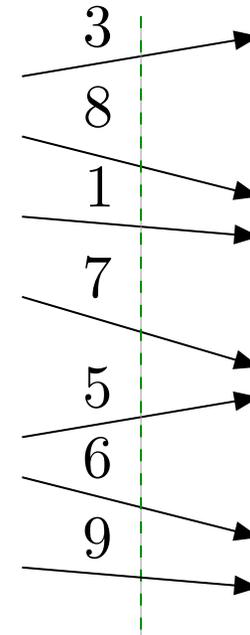
Descrição combinatória da linha

O conjunto dos segmentos na linha sofre **inserções** e **remoções**.

Como a linha vai nos ajudar a detectar interseção?

Ideia: testar interseção apenas entre segmentos “vizinhos na linha”.

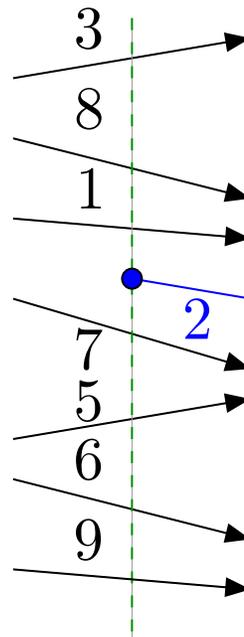
Para isso, mantemos os segmentos na linha **ordenados**.



Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a **linha**.

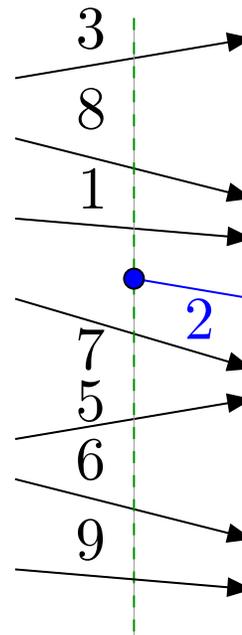
3 < 8 < 1 < 2 < 7 < 5 < 6 < 9



Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a **linha**.

3 < 8 < 1 < 2 < 7 < 5 < 6 < 9

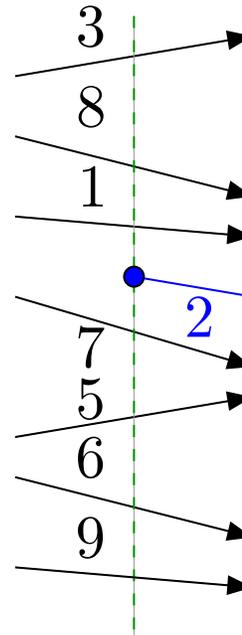


Ao **inserimos** um segmento, testamos a interseção dele com seu **predecessor** e com seu **sucessor** na ordem.

Descrição combinatória da linha

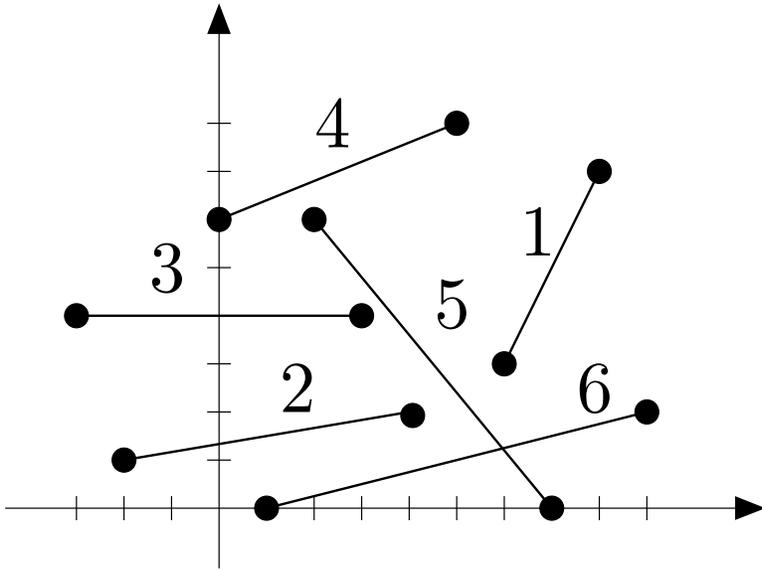
Os segmentos ficam na ordem em que intersectam a **linha**.

3 < 8 < 1 < 2 < 7 < 5 < 6 < 9



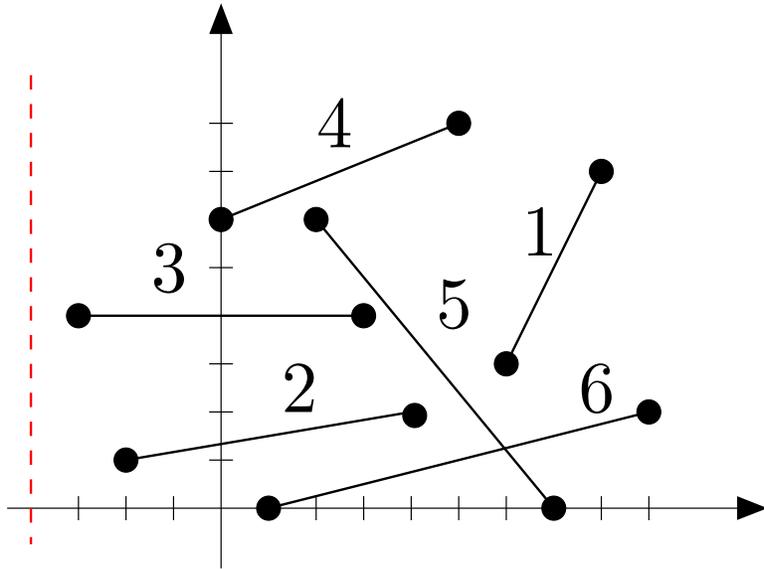
Ao **removermos** um segmento, testamos a interseção de seu **predecessor** e com seu **sucessor** na ordem.

Algoritmo de Shamos e Hoey



$-\infty$	
-3	3
-2	3 \prec 2
0	4 \prec 3 \prec 2
1	4 \prec 3 \prec 2 \prec 6
2	4 \prec 5 \prec 3 \prec 2 \prec 6
3	4 \prec 5 \prec 2 \prec 6
4	4 \prec 5 \prec 6
5	...
6	
7	
8	

Algoritmo de Shamos e Hoey



$-\infty$

-3

3

-2

3 \prec 2

0

4 \prec 3 \prec 2

1

4 \prec 3 \prec 2 \prec 6

2

4 \prec 5 \prec 3 \prec 2 \prec 6

3

4 \prec 5 \prec 2 \prec 6

4

4 \prec 5 \prec 6

5

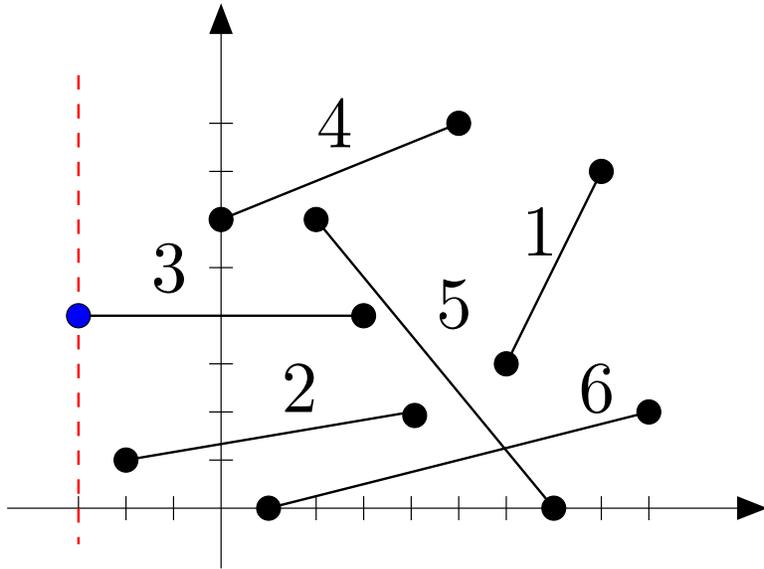
6

7

8

Alterações ocorrem nos **extremos dos segmentos**.

Algoritmo de Shamos e Hoey

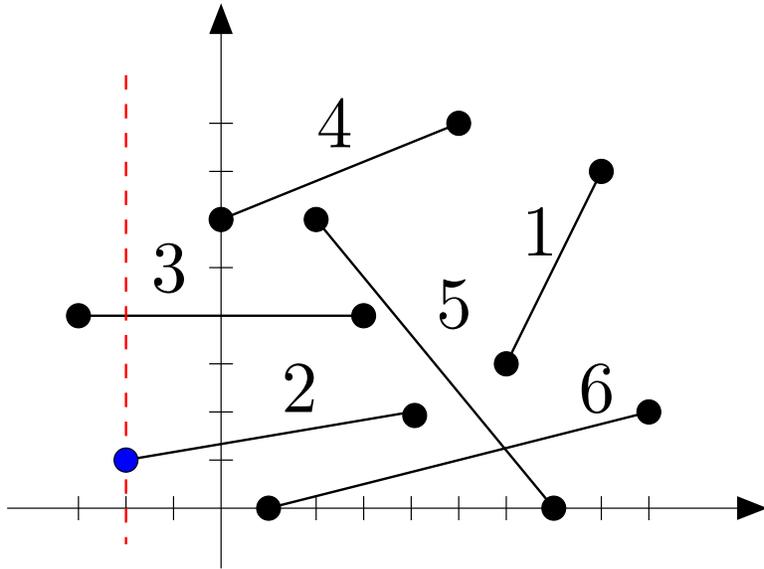


$-\infty$	
-3	3
-2	3 \prec 2
0	4 \prec 3 \prec 2
1	4 \prec 3 \prec 2 \prec 6
2	4 \prec 5 \prec 3 \prec 2 \prec 6
3	4 \prec 5 \prec 2 \prec 6
4	4 \prec 5 \prec 6
5	
6	
7	
8	

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Algoritmo de Shamos e Hoey

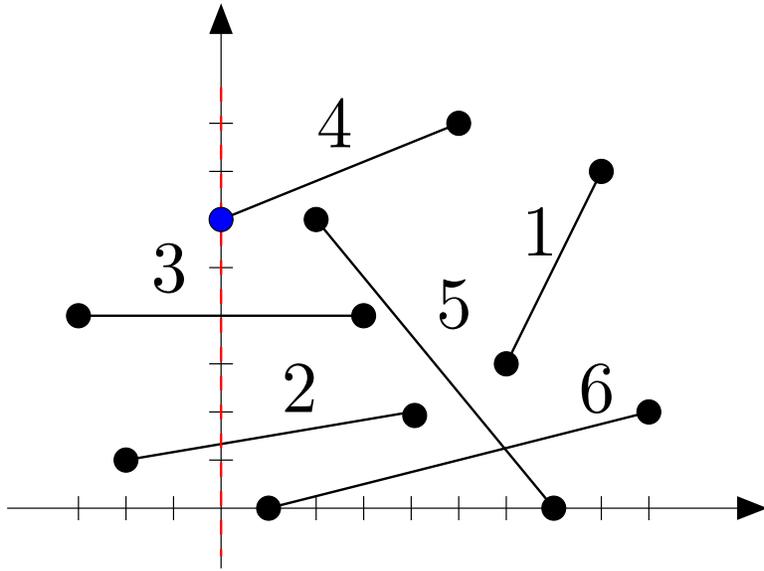


$-\infty$	
-3	3
-2	3 < 2
0	4 < 3 < 2
1	4 < 3 < 2 < 6
2	4 < 5 < 3 < 2 < 6
3	4 < 5 < 2 < 6
4	4 < 5 < 6
5	5 < 6
6	...
7	
8	

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Algoritmo de Shamos e Hoey

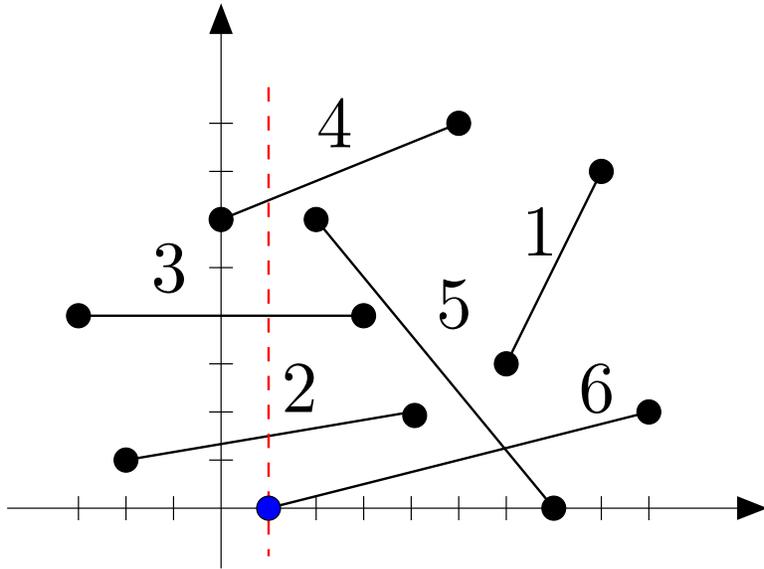


$-\infty$	
-3	3
-2	3 \prec 2
0	4 \prec 3 \prec 2
1	4 \prec 3 \prec 2 \prec 6
2	4 \prec 5 \prec 3 \prec 2 \prec 6
3	4 \prec 5 \prec 2 \prec 6
4	4 \prec 5 \prec 6
5	
6	
7	
8	

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Algoritmo de Shamos e Hoey

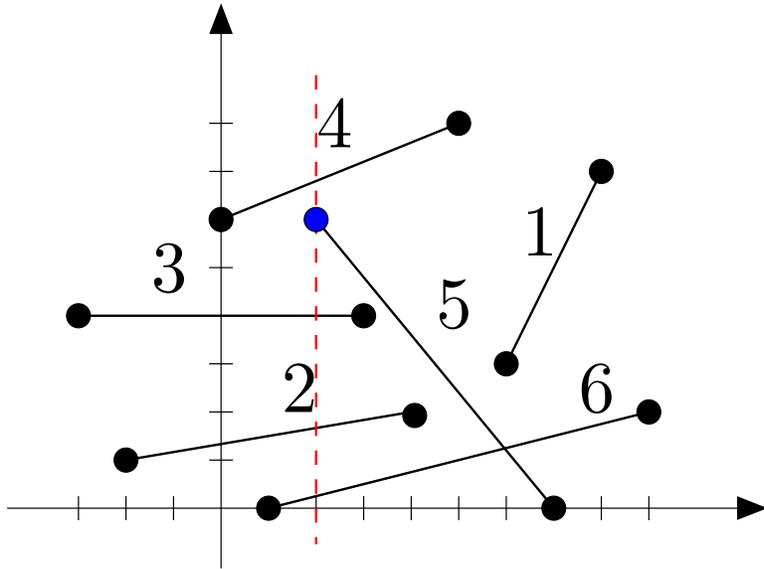


$-\infty$	
-3	3
-2	3 \prec 2
0	4 \prec 3 \prec 2
1	4 \prec 3 \prec 2 \prec 6
2	4 \prec 5 \prec 3 \prec 2 \prec 6
3	4 \prec 5 \prec 2 \prec 6
4	4 \prec 5 \prec 6
5	
6	
7	
8	

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Algoritmo de Shamos e Hoey

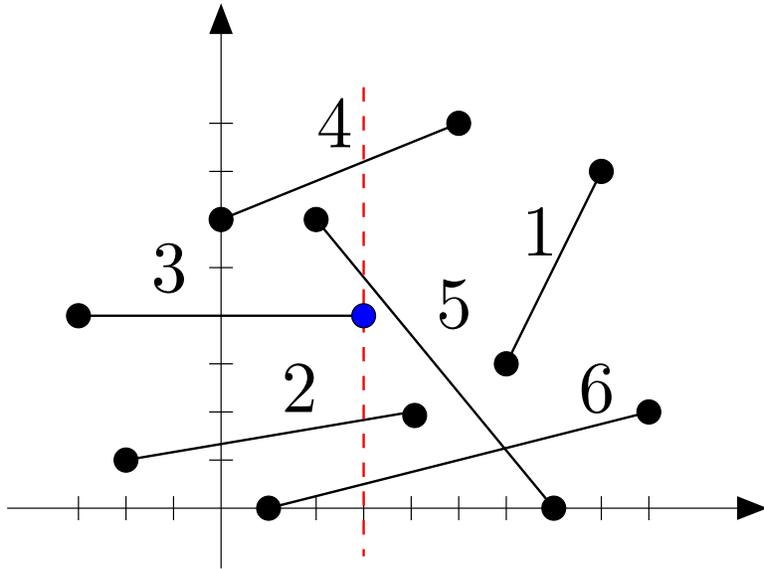


$-\infty$	
-3	3
-2	3 < 2
0	4 < 3 < 2
1	4 < 3 < 2 < 6
2	4 < 5 < 3 < 2 < 6
3	4 < 5 < 2 < 6
4	4 < 5 < 6
5	
6	
7	
8	

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Algoritmo de Shamos e Hoey

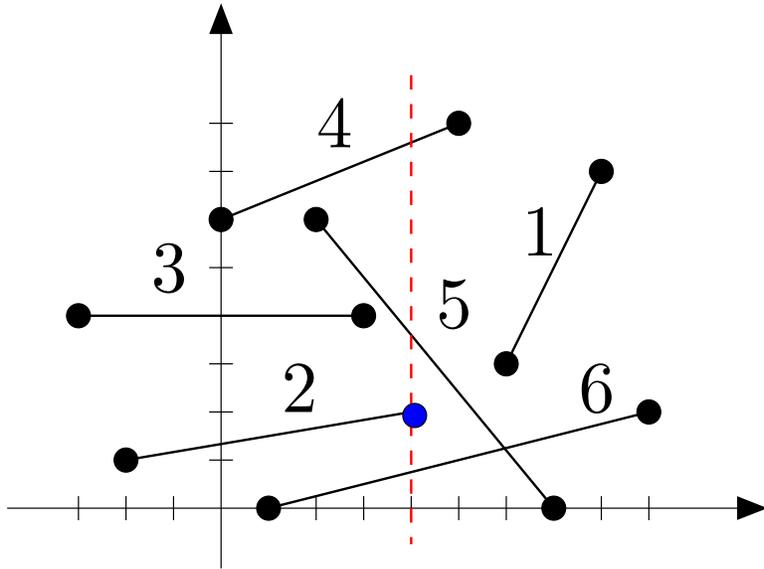


$-\infty$	
-3	3
-2	3 < 2
0	4 < 3 < 2
1	4 < 3 < 2 < 6
2	4 < 5 < 3 < 2 < 6
3	4 < 5 < 2 < 6
4	4 < 5 < 6
5	
6	
7	
8	

Alterações ocorrem
nos **extremos dos segmentos**.

Estes são
os **pontos eventos**.

Algoritmo de Shamos e Hoey

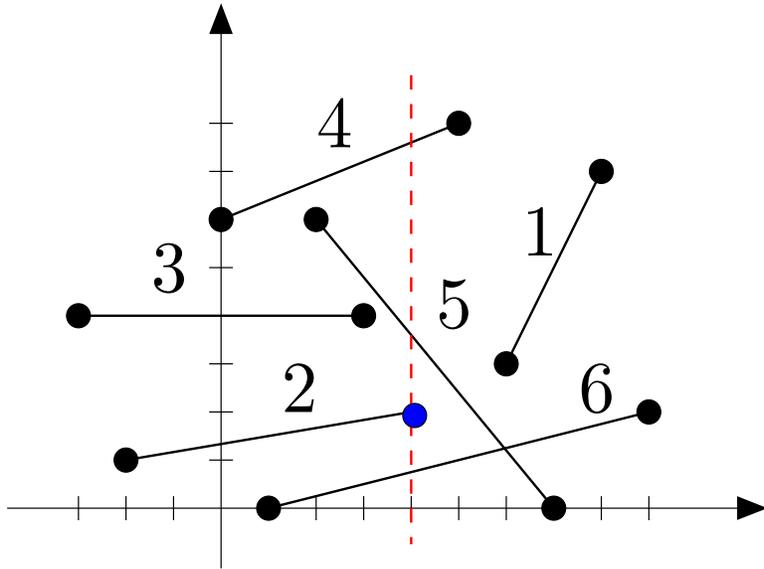


$-\infty$	
-3	3
-2	3 < 2
0	4 < 3 < 2
1	4 < 3 < 2 < 6
2	4 < 5 < 3 < 2 < 6
3	4 < 5 < 2 < 6
4	4 < 5 < 6
5	
6	
7	
8	

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Algoritmo de Shamos e Hoey



$-\infty$	
-3	3
-2	3 < 2
0	4 < 3 < 2
1	4 < 3 < 2 < 6
2	4 < 5 < 3 < 2 < 6
3	4 < 5 < 2 < 6
4	4 < 5 < 6
5	
6	
7	
8	

Alterações ocorrem nos **extremos dos segmentos**.

Estes são os **pontos eventos**.

Encontrou uma interseção!

Descrição combinatória da linha

Como guardar esse conjunto ordenado?

Descrição combinatória da linha

Como guardar esse conjunto ordenado?

Efetuiremos **inserções**, **remoções**, **predecessor** e **sucessor** neste conjunto.

Descrição combinatória da linha

Como guardar esse conjunto ordenado?

Efetuiremos **inserções**, **remoções**, **predecessor** e **sucessor** neste conjunto.

Por isso, boas escolhas de EDs são:
uma **árvore de busca binária balanceada (ABBB)**
ou uma **skip lists**.

Descrição combinatória da linha

Como guardar esse conjunto ordenado?

Efetuiremos **inserções**, **remoções**, **predecessor** e **sucessor** neste conjunto.

Por isso, boas escolhas de EDs são:
uma **árvore de busca binária balanceada (ABBB)**
ou uma **skip lists**.

Numa **ABBB**,
custo de pior caso por operação é $O(\lg m)$,
onde m é o número de elementos armazenados.

Numa **skip list**,
custo esperado por operação é $O(\lg m)$.

Ordem usada

Se a linha de varredura é a **reta** $x = t$,
a ordem nos segmentos é \prec_t , definida a seguir.

Ordem usada

Se a linha de varredura é a **reta** $x = t$,
a ordem nos segmentos é \prec_t , definida a seguir.

Considere dois segmentos de índices i e j
intersectados pela linha.

Ordem usada

Se a linha de varredura é a **reta** $x = t$,
a ordem nos segmentos é \prec_t , definida a seguir.

Considere dois segmentos de índices i e j
intersectados pela linha.

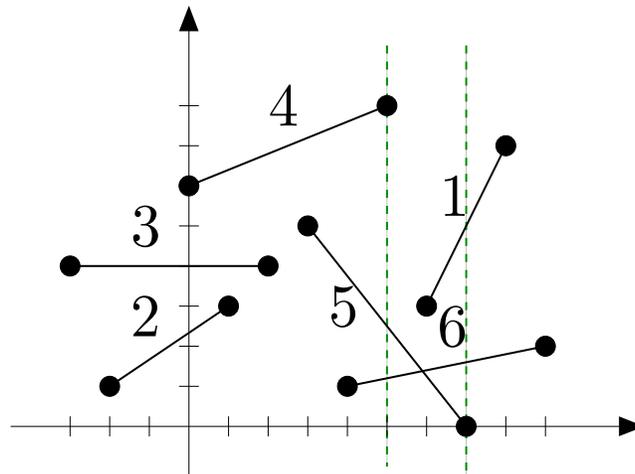
$i \prec_t j$ se a interseção da linha com i
fica acima da interseção com j .

Ordem usada

Se a linha de varredura é a **reta** $x = t$, a ordem nos segmentos é \prec_t , definida a seguir.

Considere dois segmentos de índices i e j intersectados pela linha.

$i \prec_t j$ se a interseção da linha com i fica acima da interseção com j .



Exemplo: $4 \prec_5 5 \prec_5 6$ e $1 \prec_7 6 \prec_7 5$.

Árvore de busca binária balanceada

Rotinas de manipulação de uma ABBB:

CRIE(T): cria uma ABBB T vazia;

INSIRA(T, i): insere i na ABBB T , usando \prec_t com $t = e_X[i]$;

REMOVA(T, i): remove i de T , usando \prec_t com $t = d_X[i]$;

PREDECESSOR(T, x, y): devolve o predecessor em T de um segmento que passa pelo ponto (x, y) , usando \prec_x ;

SUCCESSOR(T, x, y): devolve o sucessor em T de um segmento que passa pelo ponto (x, y) , usando \prec_x .

Árvore de busca binária balanceada

Rotinas de manipulação de uma ABBB:

CRIE(T): cria uma ABBB T vazia;

INSIRA(T, i): insere i na ABBB T , usando \prec_t com $t = e_X[i]$;

REMOVA(T, i): remove i de T , usando \prec_t com $t = d_X[i]$;

PREDECESSOR(T, x, y): devolve o predecessor em T de um segmento que passa pelo ponto (x, y) , usando \prec_x ;

SUCCESSOR(T, x, y): devolve o sucessor em T de um segmento que passa pelo ponto (x, y) , usando \prec_x .

O consumo de tempo de cada uma destas rotinas é $O(\lg m)$, onde m é o número de elementos em T .

Árvore de busca binária balanceada

Mais detalhes sobre, por exemplo, o $\text{INSIRA}(T, i)$:
insere i na ABBB T , usando \prec_t com $t = e_X[i]$.

Árvore de busca binária balanceada

Mais detalhes sobre, por exemplo, o $\text{INSIRA}(T, i)$:
insere i na ABBB T , usando \prec_t com $t = e_X[i]$.

n , e e d também são parâmetros (ou são globais).

Árvore de busca binária balanceada

Mais detalhes sobre, por exemplo, o $INSIRA(T, i)$:
insere i na ABBB T , usando \prec_t com $t = e_X[i]$.

n , e e d também são parâmetros (ou são globais).

$segmento(q)$: índice do segmento armazenado no nó q de T .

Árvore de busca binária balanceada

Mais detalhes sobre, por exemplo, o $INSIRA(T, i)$:
insere i na ABDB T , usando \prec_t com $t = e_X[i]$.

n , e e d também são parâmetros (ou são globais).

$segmento(q)$: índice do segmento armazenado no nó q de T .

A inserção começa com uma **busca a partir de T** .

Na busca, vamos para a esquerda de T se
o ponto $e[i]$ estiver à esquerda de $segmento(T)$.
Do contrário, vamos para a direita.

Árvore de busca binária balanceada

Mais detalhes sobre, por exemplo, o $\text{INSIRA}(T, i)$:
insere i na ABBB T , usando \prec_t com $t = e_X[i]$.

n , e e d também são parâmetros (ou são globais).

$\text{segmento}(q)$: índice do segmento armazenado no nó q de T .

A inserção começa com uma **busca a partir de T** .

Na busca, vamos para a esquerda de T se
o ponto $e[i]$ estiver à esquerda de $\text{segmento}(T)$.
Do contrário, vamos para a direita.

Exercício: Modifique a rotina de inserção em ABBB
rubro-negra para que funcione para este algoritmo.

Algoritmo de Shamos e Hoey

Entrada: coleção $e[1..n], d[1..n]$ de segmentos.

Algoritmo de Shamos e Hoey

Entrada: coleção $e[1..n], d[1..n]$ de segmentos.

Saída: VERDADE se há dois segmentos na coleção que se intersectam, e FALSO caso contrário.

Algoritmo de Shamos e Hoey

Entrada: coleção $e[1..n], d[1..n]$ de segmentos.

Saída: VERDADE se há dois segmentos na coleção que se intersectam, e FALSO caso contrário.

Hipótese simplificadora:

Não há dois pontos extremos com a mesma X -coordenada.

Em particular, não há segmentos verticais,

nem dois segmentos com extremos coincidentes.

Montagem da fila de eventos

FILA DE EVENTOS:

recebe $e[1..n]$ e $d[1..n]$ com extremos dos segmentos

Montagem da fila de eventos

FILADEEVENTOS:

recebe $e[1..n]$ e $d[1..n]$ com extremos dos segmentos

troca $e[i]$ por $d[i]$ para todo i tal que $e_X[i] < d_X[i]$

($e[i]$: extremo esquerdo do segmento i e $d[i]$ o direito)

Montagem da fila de eventos

FILADEEVENTOS:

recebe $e[1..n]$ e $d[1..n]$ com extremos dos segmentos

troca $e[i]$ por $d[i]$ para todo i tal que $e_X[i] < d_X[i]$
($e[i]$: extremo esquerdo do segmento i e $d[i]$ o direito)

devolve

$E[1..2n]$: pontos de $e[1..n]$ e $d[1..n]$
ordenados pelas suas coordenadas X

Montagem da fila de eventos

FILADEEVENTOS:

recebe $e[1..n]$ e $d[1..n]$ com extremos dos segmentos

troca $e[i]$ por $d[i]$ para todo i tal que $e_X[i] < d_X[i]$
($e[i]$: extremo esquerdo do segmento i e $d[i]$ o direito)

devolve

$E[1..2n]$: pontos de $e[1..n]$ e $d[1..n]$
ordenados pelas suas coordenadas X

$segm[1..2n]$:

$segm[p]$: índice do segmento do qual $E[p]$ é extremo

Montagem da fila de eventos

FILADEEVENTOS:

recebe $e[1..n]$ e $d[1..n]$ com extremos dos segmentos

troca $e[i]$ por $d[i]$ para todo i tal que $e_X[i] < d_X[i]$
($e[i]$: extremo esquerdo do segmento i e $d[i]$ o direito)

devolve

$E[1..2n]$: pontos de $e[1..n]$ e $d[1..n]$
ordenados pelas suas coordenadas X

$segm[1..2n]$:

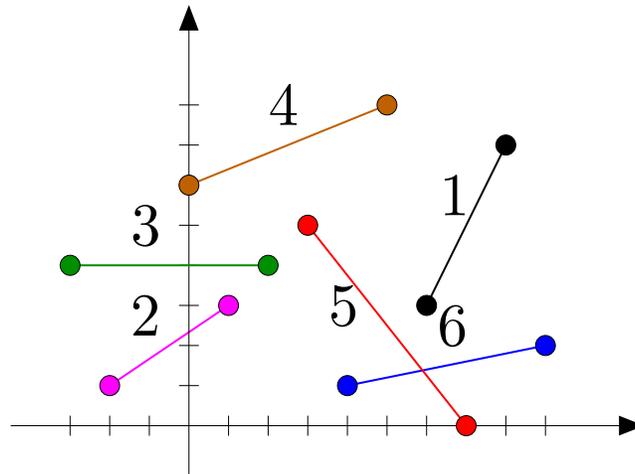
$segm[p]$: índice do segmento do qual $E[p]$ é extremo

$esq[1..2n]$:

$esq[p]$: VERDADE se $E[p]$ é extremo esquerdo de $segm[p]$

FALSO caso contrário.

Fila de eventos



E_X	-3	-2	0	1	2	3	4	5	6	7	8	9
E_Y	4	1	6	3	4	5	1	8	3	0	7	2
	1	2	3	4	5	6	7	8	9	10	11	12

$segm$	3	2	4	2	3	5	6	4	1	5	1	6
esq	V	V	V	F	F	V	V	F	V	F	F	F
	1	2	3	4	5	6	7	8	9	10	11	12

Processamento de ponto evento

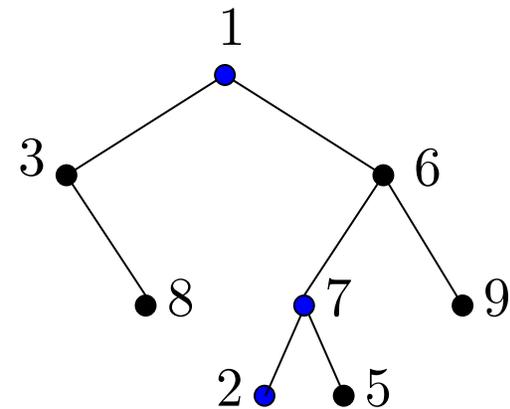
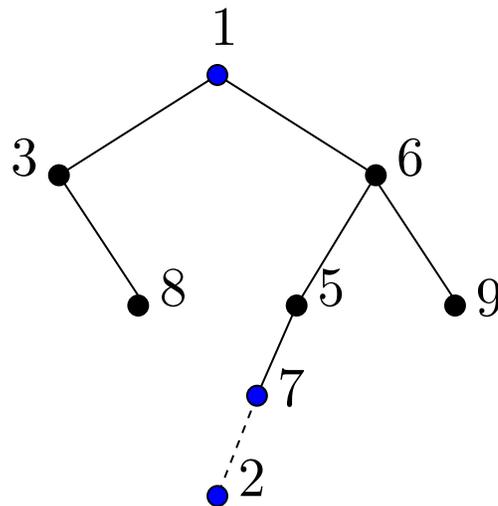
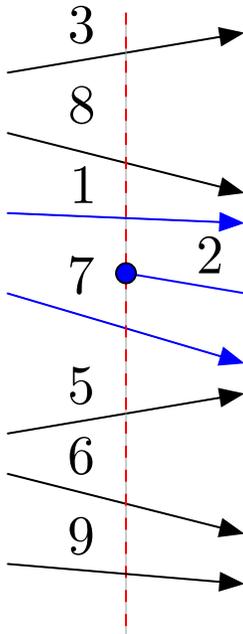
Dois tipos:

- **começo de segmento:** inclui o novo segmento na ABBB e verifica interseção com **seus dois novos “vizinhos”**.

Processamento de ponto evento

Dois tipos:

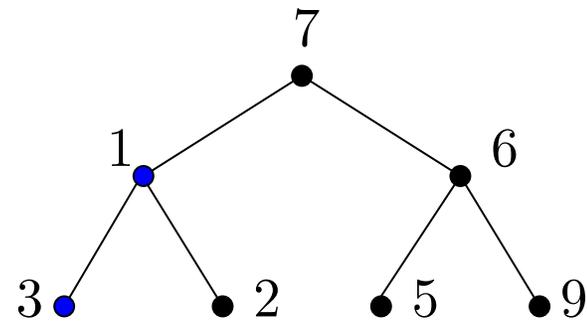
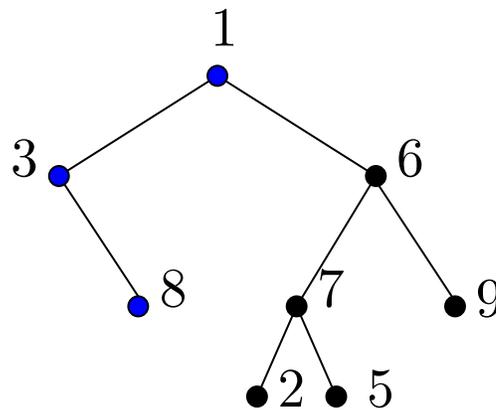
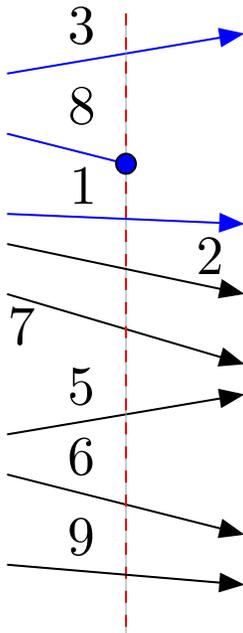
- **começo de segmento:** inclui o novo segmento na ABBB e verifica interseção com seus dois novos “vizinhos”.



Processamento de ponto evento

Dois tipos:

- **começo de segmento:** inclui o novo segmento na ABBB e verifica interseção com **seus dois novos “vizinhos”**.
- **fim de segmento:** remove o segmento da ABBB e verifica interseção entre **seus dois ex-vizinhos**.



Processamento de ponto evento

Dois tipos:

- **começo de segmento:** inclui o novo segmento na ABBB e verifica interseção com **seus dois novos “vizinhos”**.
- **fim de segmento:** remove o segmento da ABBB e verifica interseção entre **seus dois ex-vizinhos**.

Invariante: verificamos interseção entre quaisquer dois segmentos vizinhos na ABBB.

Processamento de ponto evento

Dois tipos:

- **começo de segmento:** inclui o novo segmento na ABBB e verifica interseção com **seus dois novos “vizinhos”**.
- **fim de segmento:** remove o segmento da ABBB e verifica interseção entre **seus dois ex-vizinhos**.

Invariante: verificamos interseção entre quaisquer dois segmentos vizinhos na ABBB.

Correção: se há dois segmentos que se intersectam, em algum momento, os dois serão vizinhos na ABBB.

Algoritmo de Shamos e Hoey

INTERSEÇÃO-SH(e, d, n)

```
1  ( $E, segm, esq$ )  $\leftarrow$  FILADEEVENTOS( $e, d, n$ )
2  CRIE( $T$ )
3  para  $p \leftarrow 1$  até  $2n$  faça
4       $i \leftarrow segm[p]$ 
5       $pred \leftarrow$  PREDECESSOR( $T, E_X[p], E_Y[p]$ )
6       $suc \leftarrow$  SUCESSOR( $T, E_X[p], E_Y[p]$ )
7      se  $esq[p]$ 
8          então INSIRA( $T, i$ )
9              se ( $pred \neq_{NIL}$  e INTER( $e, d, i, pred$ ))
10                 ou ( $suc \neq_{NIL}$  e INTER( $e, d, i, suc$ ))
11                     então devolva VERDADE
12             senão REMOVA( $T, i$ )
13                 se  $pred \neq_{NIL}$  e  $suc \neq_{NIL}$  e INTER( $e, d, pred, suc$ )
14                     então devolva VERDADE
15 devolva FALSO
```

Consumo de tempo

O algoritmo executa $2n$ iterações.

Cada iteração faz uma chamada a PREDECESSOR, uma a SUCESSOR, e uma a INSIRA ou a REMOVA.

Na ABBB, em qualquer momento, há $O(n)$ segmentos.

Assim, cada uma destas operações consome tempo $O(\lg n)$.

As demais operações efetuadas em uma iteração consomem tempo $O(1)$ (mesmo as chamadas a INTER).

Logo o consumo de tempo por iteração é $O(\lg n)$, e o algoritmo de Shamos e Hoey consome tempo $O(n \lg n)$.

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Você consegue projetar um algoritmo que consuma tempo $O(n \lg n)$ para este problema?

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Você consegue projetar um algoritmo que consuma tempo $O(n \lg n)$ para este problema?

No máximo, quantos pares teremos que imprimir?