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)

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)

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

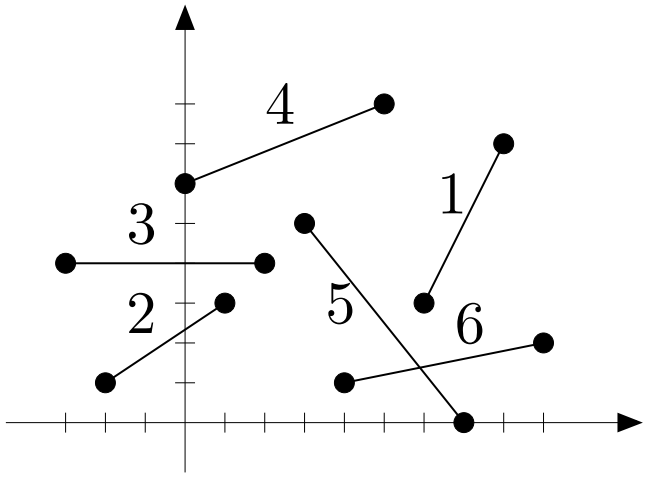
Uma coleção de segmentos do plano é dada por dois vetores $e[1\,..\,n], d[1\,..\,n]$ de pontos.

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 $(d_X[i], d_Y[i])$.

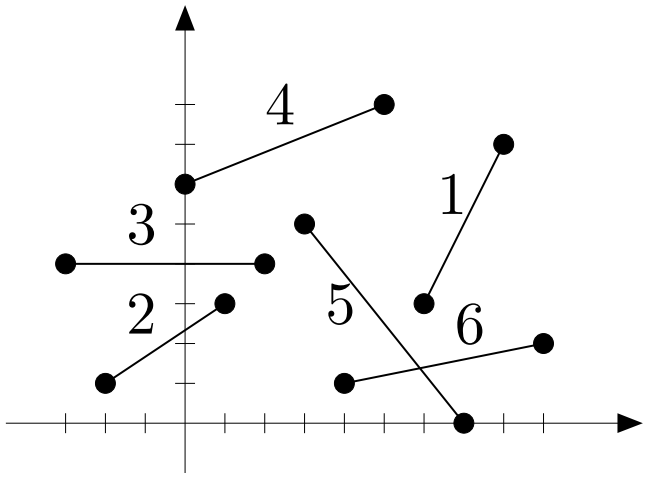


	1	2	3	4	5	6
e_X	6	−2	−3	0	3	4
e_Y	3	1	4	6	5	1

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

Interseção de segmentos

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

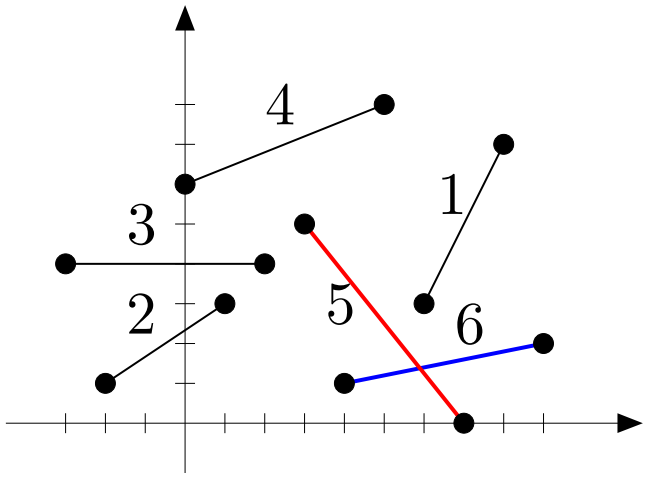


e_X	6	-2	-3	0	3	4
e_Y	3	1	4	6	5	1
	1	2	3	4	5	6

d_X	8	1	2	5	7	9
d_Y	7	3	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.

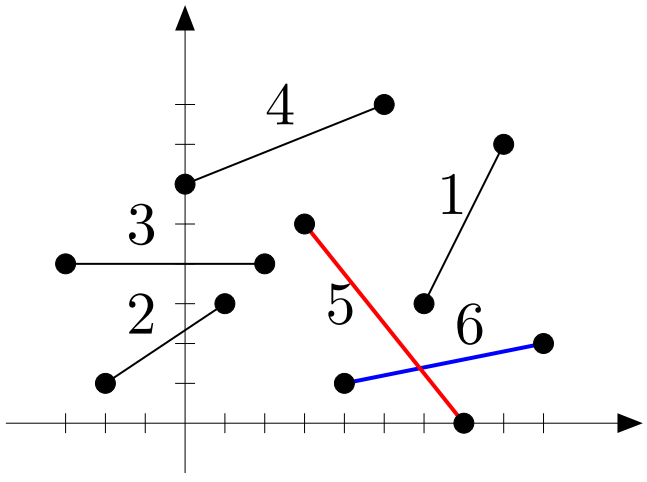


e_X	6	-2	-3	0	3	4
e_Y	3	1	4	6	5	1
	1	2	3	4	5	6

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

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

Resposta: sim, existem dois intervalos com interseção.

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 $[e_X[1]\,..\,d_X[1]], \ldots, [e_X[n]\,..\,d_X[n]]$.

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 $[e_X[1]..d_X[1]], \ldots, [e_X[n]..d_X[n]]$.

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 $[e_X[1]..d_X[1]], \ldots, [e_X[n]..d_X[n]]$.

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)

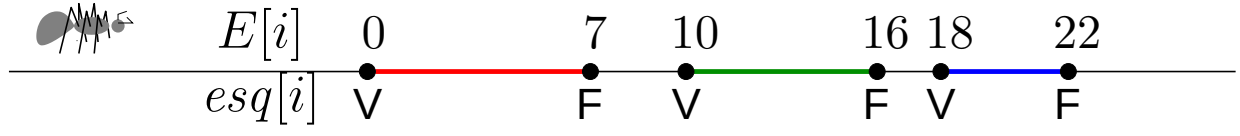
1 para $i \leftarrow 1$ até n faça ▷ para cada intervalo marca

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

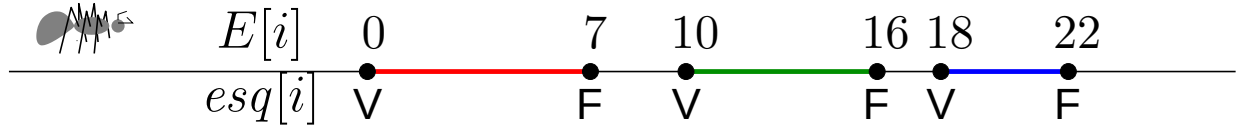
e_X	10	0	18
d_X	16	7	22
	1	2	3



Interseção de intervalos

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

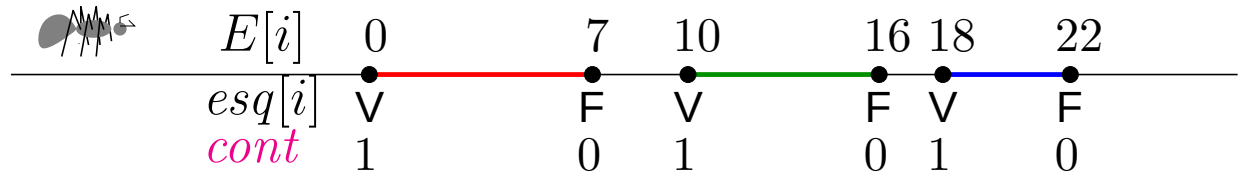
e_X	10	0	18
d_X	16	7	22
	1	2	3



Interseção de intervalos

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

e_X	10	0	18
d_X	16	7	22
	1	2	3

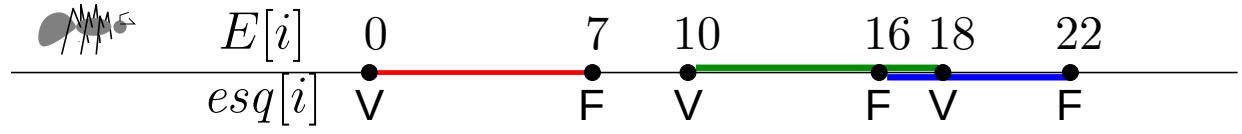


VARREDURA$(e_X, d_X, 3)$ = FALSO

Interseção de intervalos

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

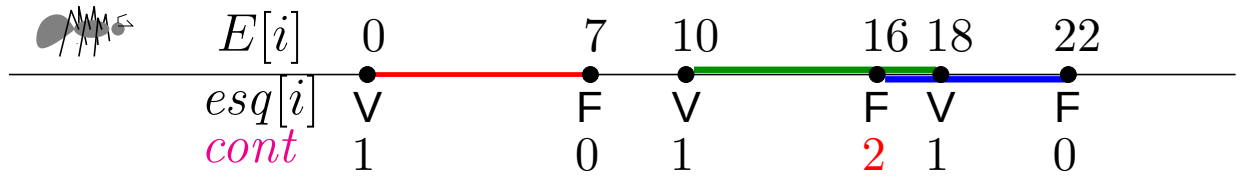
e_X	10	0	18
d_X	16	7	22
	1	2	3



Interseção de intervalos

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

	1	2	3
e_X	10	0	18
d_X	16	7	22



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

Interseção de intervalos

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

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

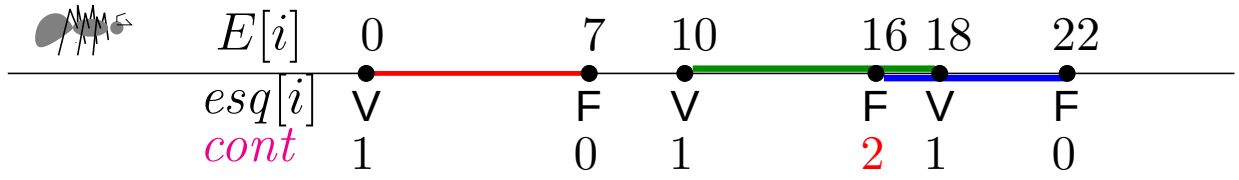
Método da linha de varredura

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

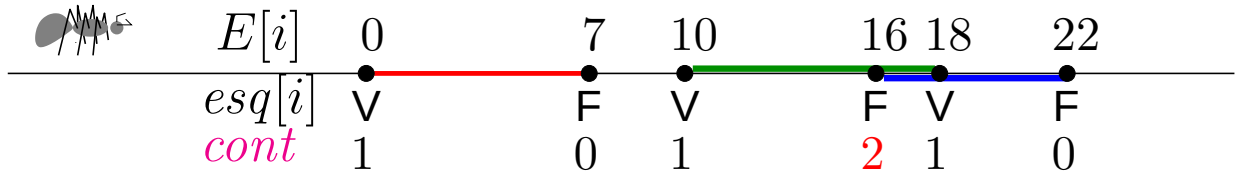
Uma linha imaginária move-se da esquerda para a direita.



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.

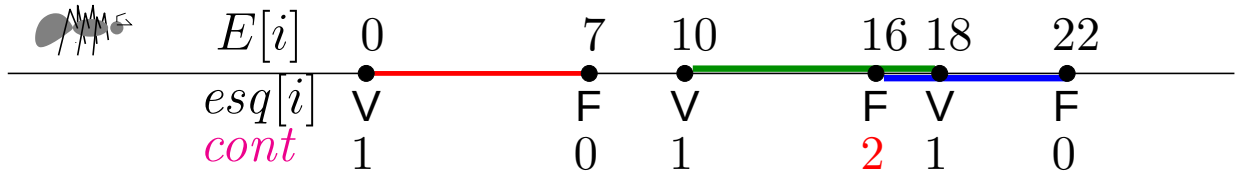


À medida que ela move,
o problema restrito à esquerda dela é resolvido.

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.



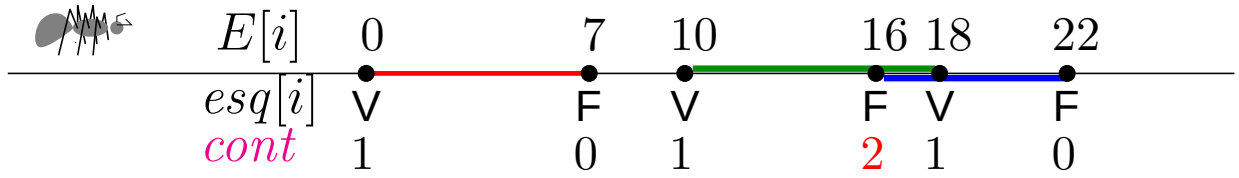
À medida que ela move,
o problema restrito à esquerda dela é resolvido.

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.

Tratamento dos pontos eventos

Pontos eventos são mantidos em uma fila.

- Atualizar a fila:
  remover o ponto evento corrente, junto com qualquer outro que tenha se tornado obsoleto e, eventualmente, inserir novos pontos eventos na fila;

Tratamento dos pontos eventos

Pontos eventos são mantidos em uma fila.

- Atualizar a fila:
  remover o ponto evento corrente, junto com qualquer outro que tenha se tornado obsoleto e, eventualmente, inserir novos pontos eventos na fila;

- Atualizar a linha:
  atualizar a descrição combinatória da linha de varredura para que esta represente a situação atual;

Tratamento dos pontos eventos

Pontos eventos são mantidos em uma fila.

- Atualizar a fila:
  remover o ponto evento corrente, junto com qualquer outro que tenha se tornado obsoleto e, eventualmente, inserir novos pontos eventos na fila;
- Atualizar a linha:
  atualizar a descrição combinatória da linha de varredura para que esta represente a situação atual;
- Resolver o problema:
  estender a solução corrente.

Interseção de dois segmentos

INTERSECTA:
Recebe dois segmentos e devolve VERDADE se os segmentos se intersectam, e FALSO caso contrário.

Interseção de dois segmentos

INTERSECTA$(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$
1 se ESQUERDA$((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq$
ESQUERDA$((x_1, y_1), (x_2, y_2), (x_4, y_4))$
2 e ESQUERDA$((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq$
ESQUERDA$((x_3, y_3), (x_4, y_4), (x_2, y_2))$
3 então devolva VERDADE
4 senão devolva FALSO

Interseção de dois segmentos

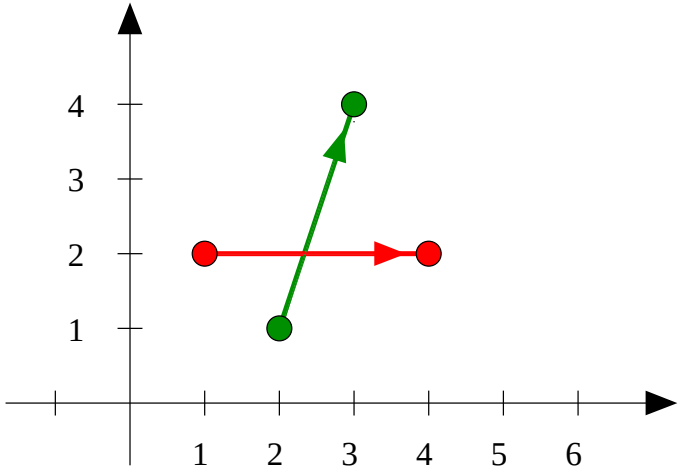
INTERSECTA$(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$

1 se ESQUERDA$((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq$
ESQUERDA$((x_1, y_1), (x_2, y_2), (x_4, y_4))$

2 e ESQUERDA$((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq$
ESQUERDA$((x_3, y_3), (x_4, y_4), (x_2, y_2))$

3 então devolva VERDADE

4 senão devolva FALSO



ESQUERDA$((1, 2), (4, 2), (2, 1)) =$ FALSO

ESQUERDA$((1, 2), (4, 2), (3, 4)) =$ VERDADE

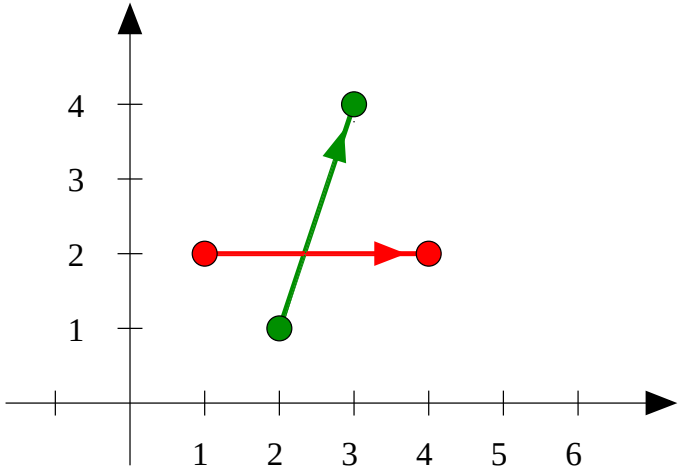
ESQUERDA$((2, 1), (3, 4), (1, 2)) =$ VERDADE

ESQUERDA$((2, 1), (3, 4), (4, 2)) =$ FALSO

Interseção de dois segmentos

INTERSECTA$(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$

1 se ESQUERDA$((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq$
ESQUERDA$((x_1, y_1), (x_2, y_2), (x_4, y_4))$
2 e ESQUERDA$((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq$
ESQUERDA$((x_3, y_3), (x_4, y_4), (x_2, y_2))$
3 então devolva VERDADE
4 senão devolva FALSO



ESQUERDA$((1, 2), (4, 2), (2, 1)) =$ FALSO
ESQUERDA$((1, 2), (4, 2), (3, 4)) =$ VERDADE
ESQUERDA$((2, 1), (3, 4), (1, 2)) =$ VERDADE
ESQUERDA$((2, 1), (3, 4), (4, 2)) =$ FALSO

INTERSECTA$((1, 2), (4, 2), (2, 1), (3, 4)) =$ VERDADE

Interseção de dois segmentos

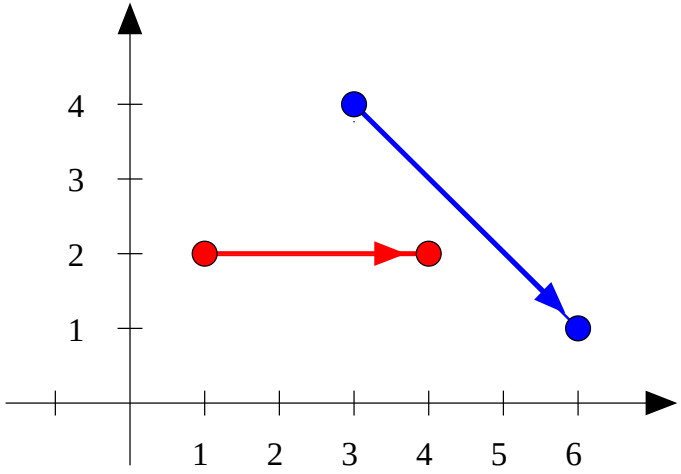
INTERSECTA$(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$

1 se ESQUERDA$((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq$
ESQUERDA$((x_1, y_1), (x_2, y_2), (x_4, y_4))$

2 e ESQUERDA$((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq$
ESQUERDA$((x_3, y_3), (x_4, y_4), (x_2, y_2))$

3 então devolva VERDADE

4 senão devolva FALSO



ESQUERDA$((3,4),(6,1),(1,2)) =$ FALSO

ESQUERDA$((3,4),(6,1),(4,2)) =$ FALSO

Interseção de dois segmentos

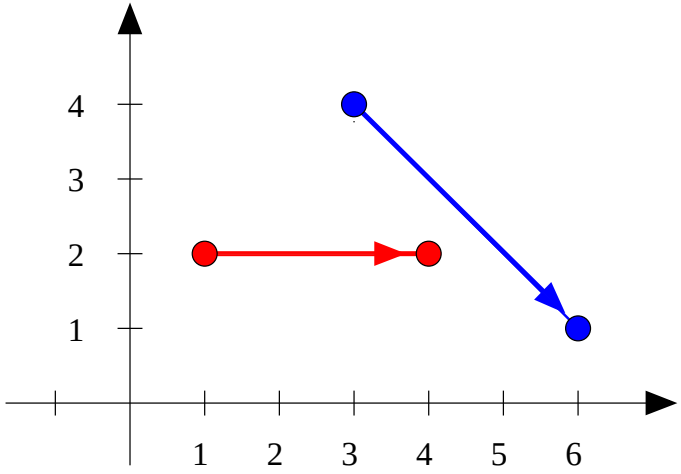
INTERSECTA$(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$

1 se ESQUERDA$((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq$
ESQUERDA$((x_1, y_1), (x_2, y_2), (x_4, y_4))$

2 e ESQUERDA$((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq$
ESQUERDA$((x_3, y_3), (x_4, y_4), (x_2, y_2))$

3 então devolva VERDADE

4 senão devolva FALSO



ESQUERDA$((3, 4), (6, 1), (1, 2)) =$ FALSO

ESQUERDA$((3, 4), (6, 1), (4, 2)) =$ FALSO

INTERSECTA$((1, 2), (4, 2), (3, 4), (6, 1)) =$ FALSO

Interseção de dois segmentos

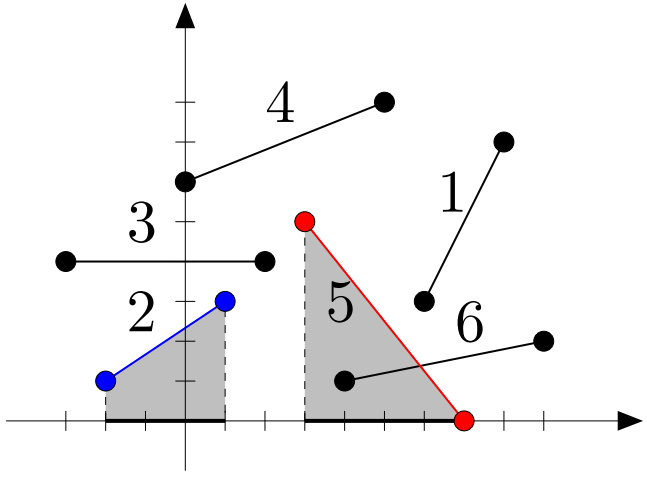
INTERSECTA$(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$
1 se ESQUERDA$((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq$
ESQUERDA$((x_1, y_1), (x_2, y_2), (x_4, y_4))$
2 e ESQUERDA$((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq$
ESQUERDA$((x_3, y_3), (x_4, y_4), (x_2, y_2))$
3 então devolva VERDADE
4 senão devolva FALSO

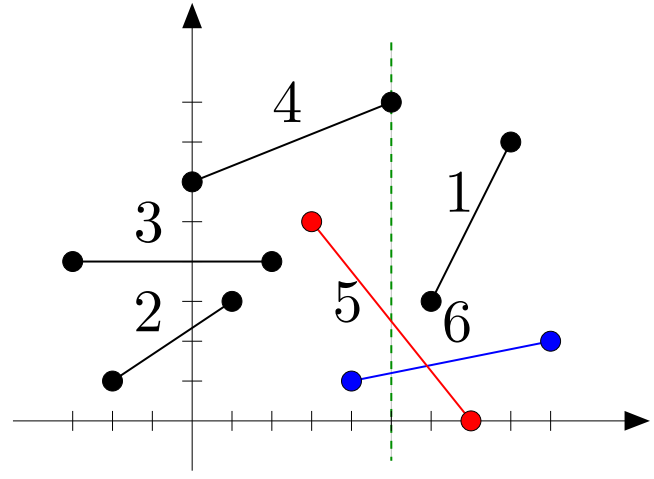
Abreviatura:

INTER(e, d, i, j)
1 devolva INTERSECTA$(e_X[i], e_Y[i], d_X[i], d_Y[i],$
$e_X[j], e_Y[j], d_X[j], d_Y[j])$

Algoritmo de Shamos e Hoey

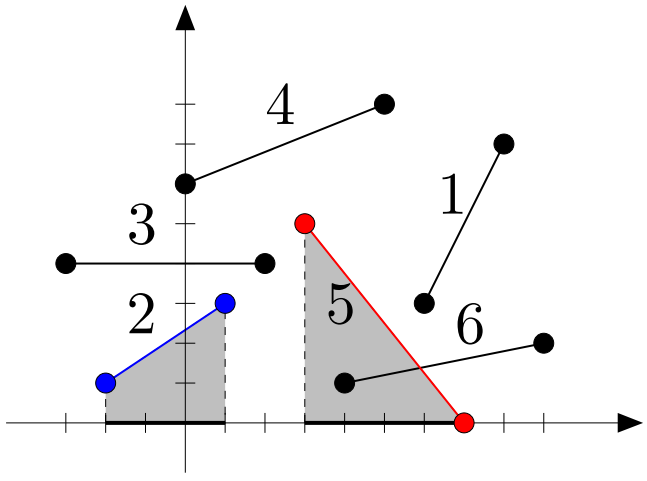
Ideia: Dois segmentos cuja projeção no eixo X sejam disjuntas não se intersectam.

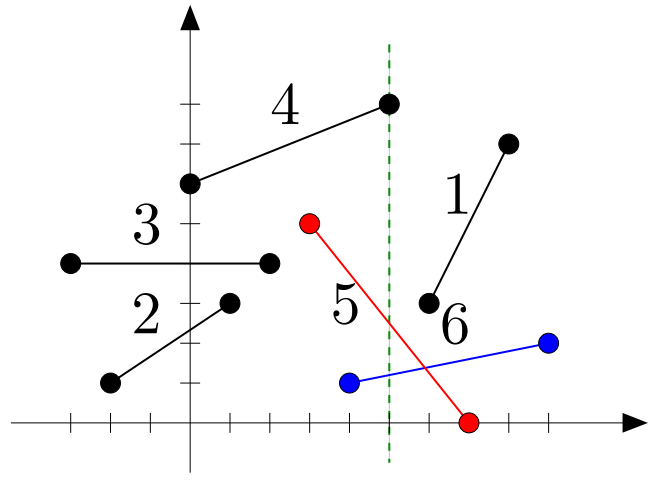




Algoritmo de Shamos e Hoey

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

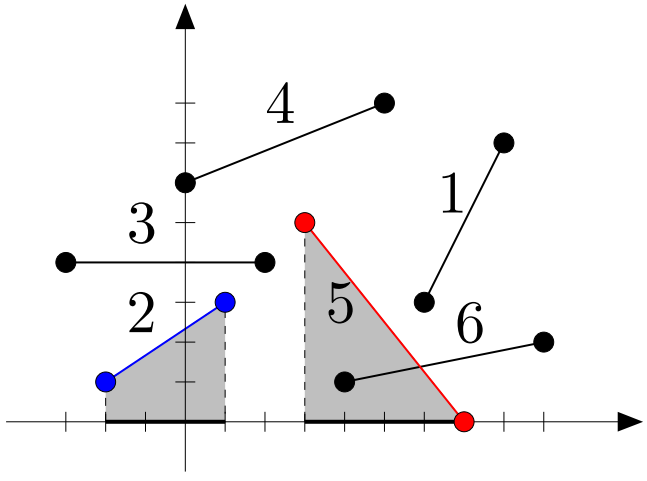


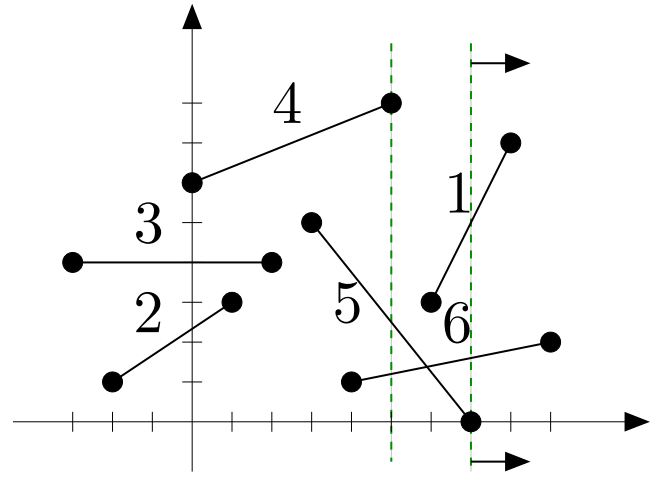


Se a projeção no eixo X de dois segmentos tem interseção, então há uma linha vertical que intersecta ambos.

Algoritmo de Shamos e Hoey

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

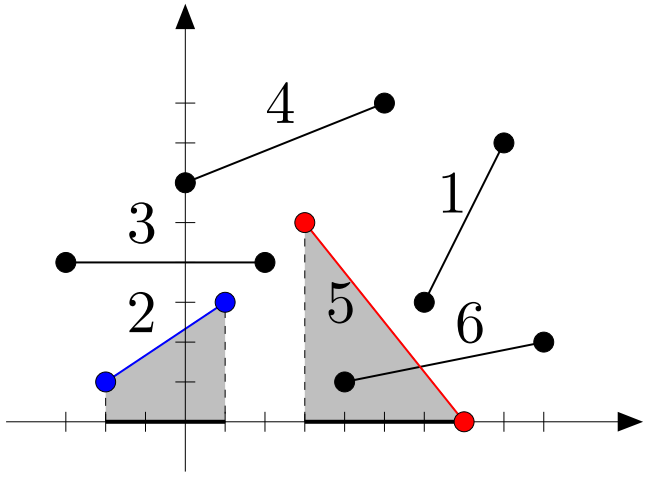


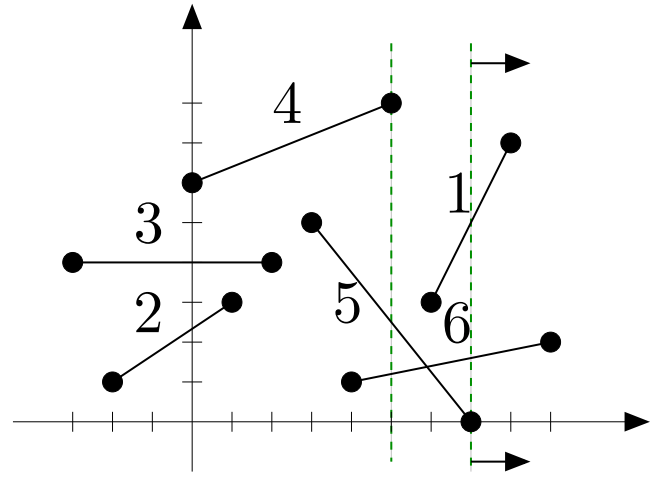


Imagine esta linha vertical varrendo o plano da esquerda para a direita...

Algoritmo de Shamos e Hoey

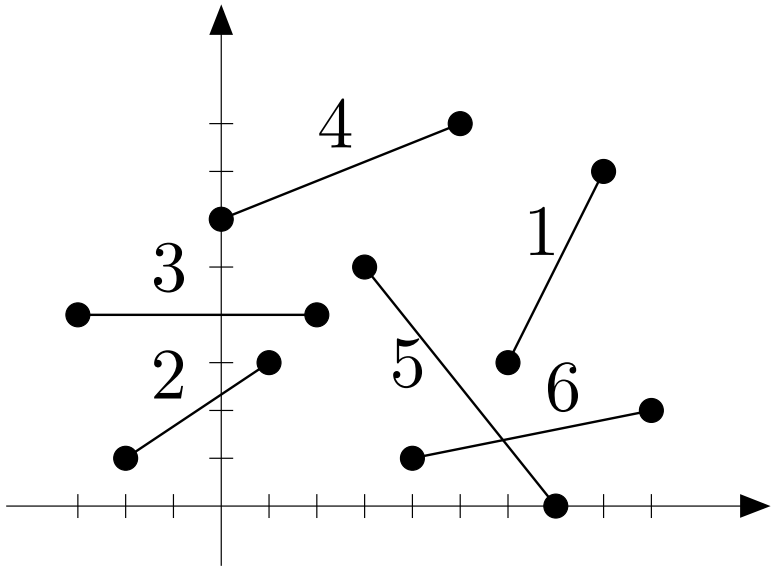
Ideia: Dois segmentos cuja projeção no eixo X sejam disjuntas não se intersectam.





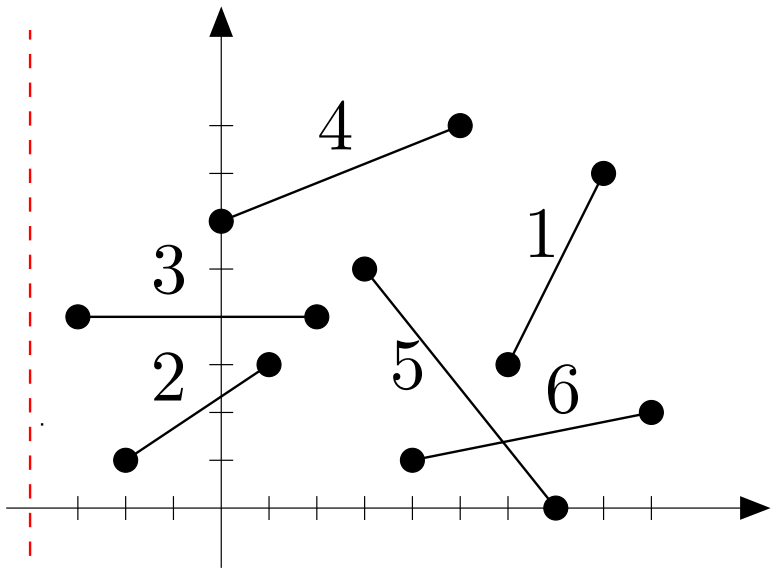
Imagine esta linha vertical varrendo o plano da esquerda para a direita...
Enquanto a linha varre o plano, mantemos os segmentos intersectados por ela na descrição combinatória da linha.

Descrição combinatória da linha



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

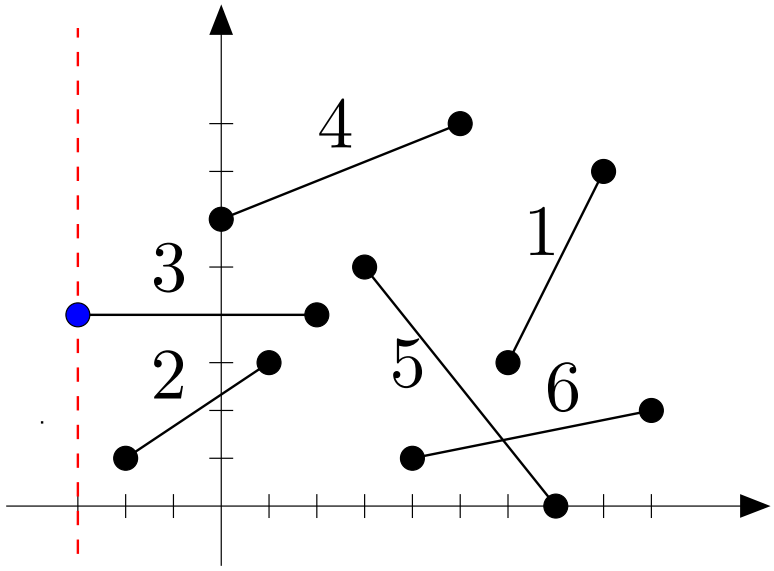
Descrição combinatória da linha



Alterações ocorrem
nos extremos dos segmentos.

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

Descrição combinatória da linha

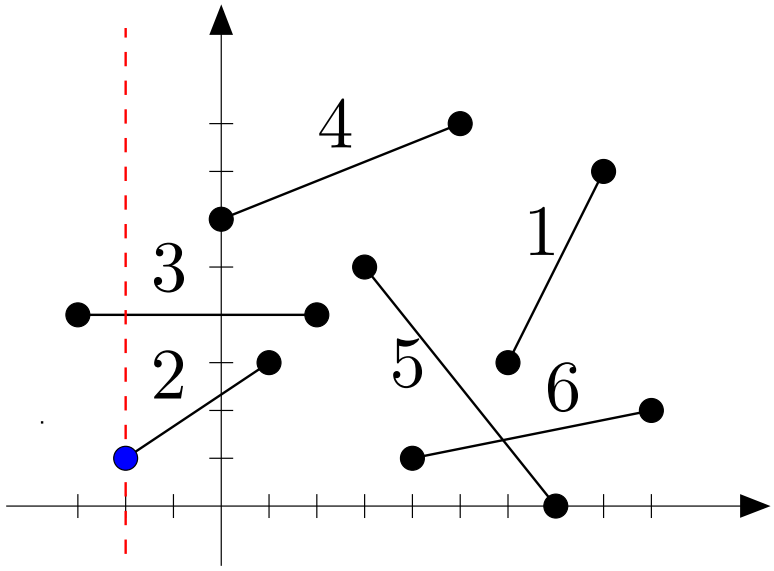


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

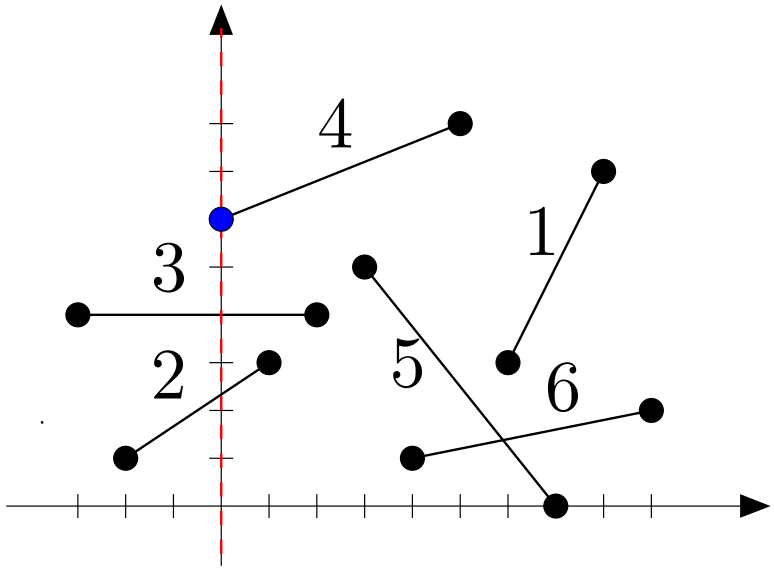


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

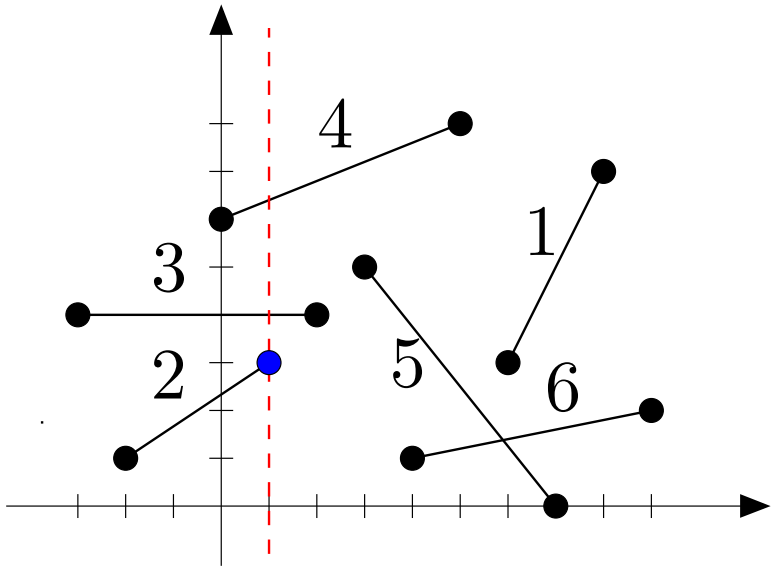


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

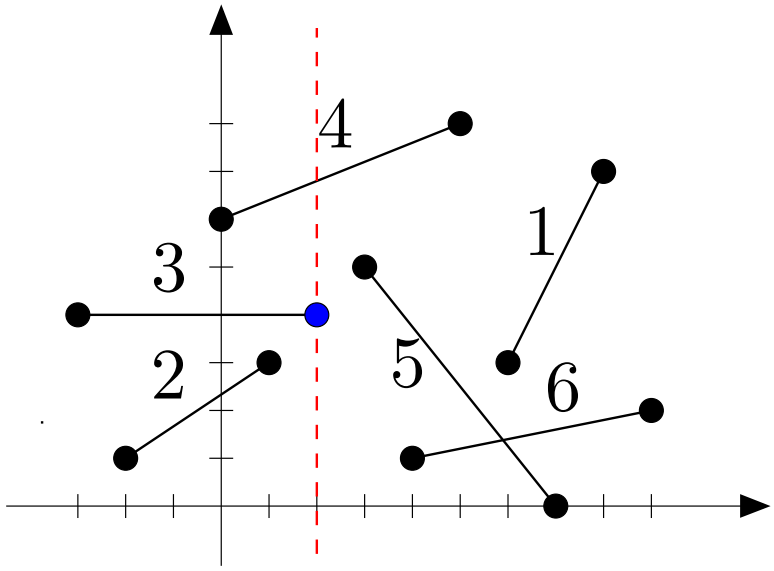


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

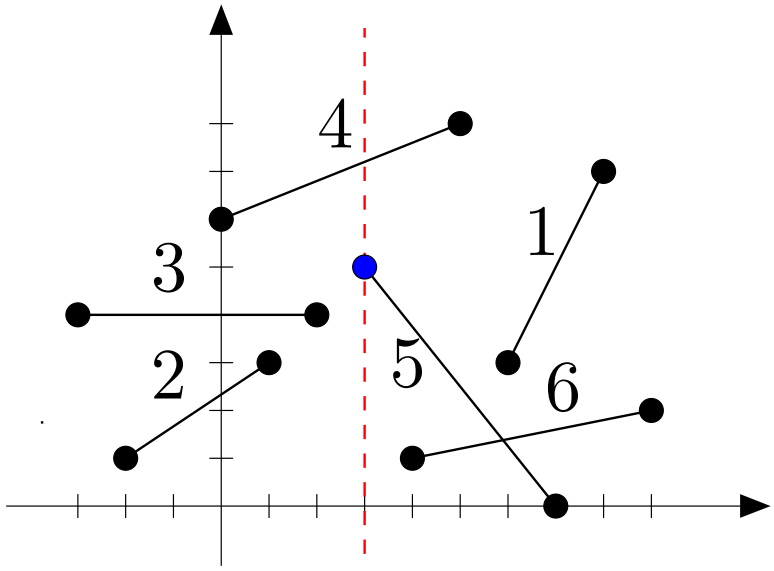


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

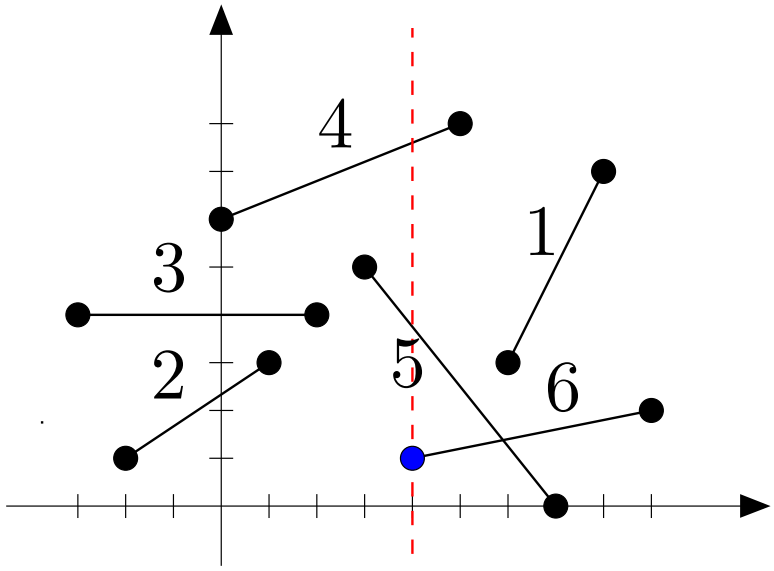


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

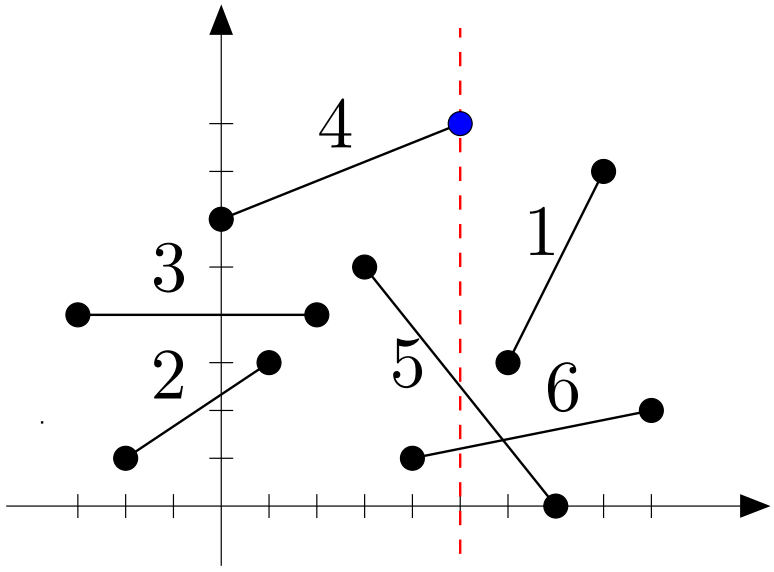


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

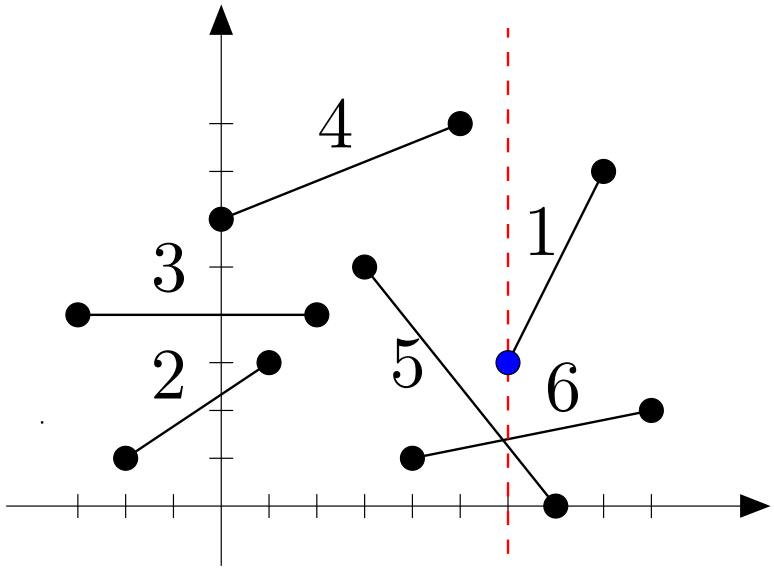


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

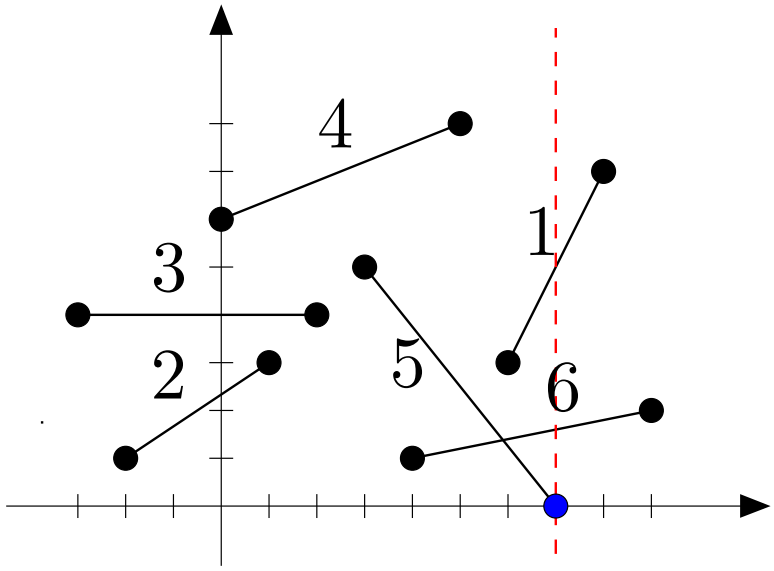


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

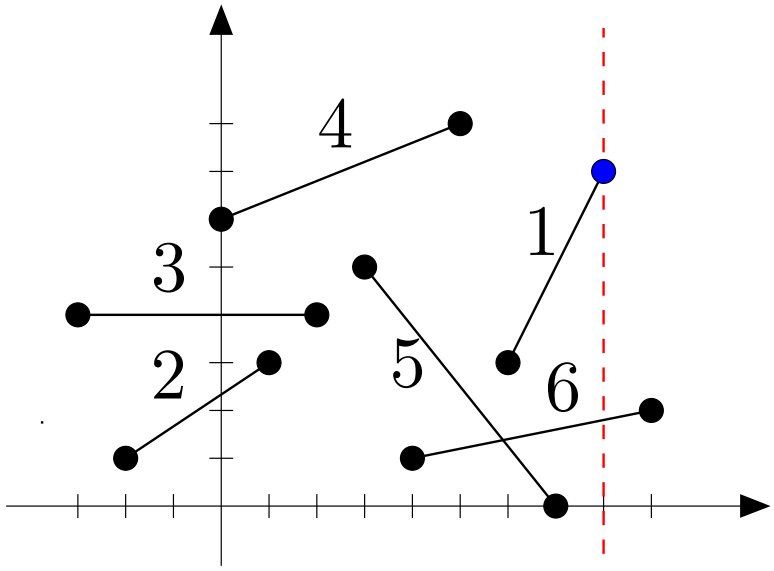


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

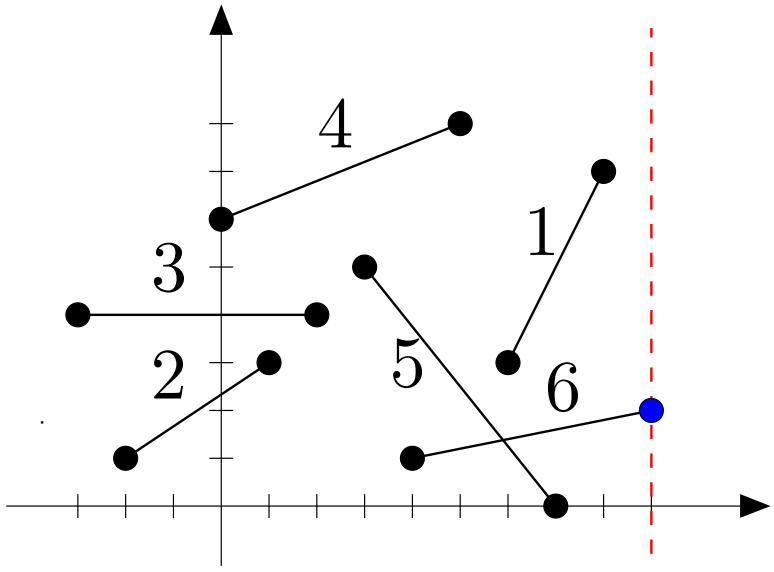


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

Descrição combinatória da linha

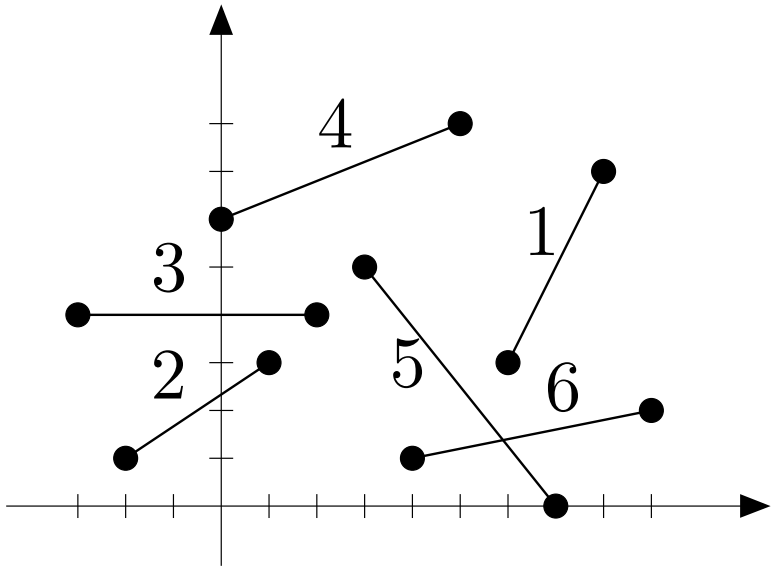


Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

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

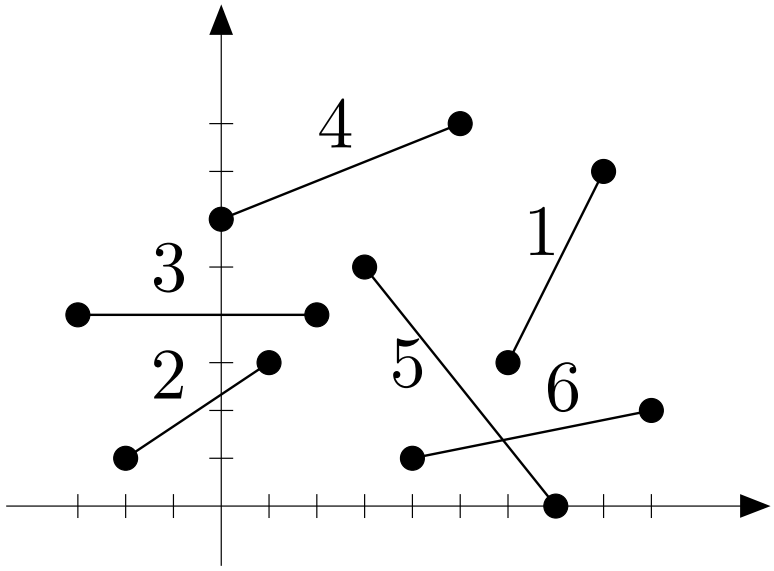
Descrição combinatória da linha



Como guardar
um destes conjuntos?

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

Descrição combinatória da linha



Como guardar
um destes conjuntos?

Que operações ele sofre?

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

Descrição combinatória da linha

O conjunto dos segmentos na linha sofre inserções e remoções.

Descrição combinatória da linha

O conjunto dos segmentos na linha sofre inserções e remoções.

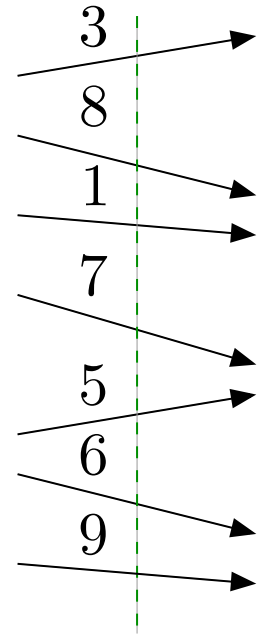
Como a linha vai nos ajudar a detectar interseção?

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".



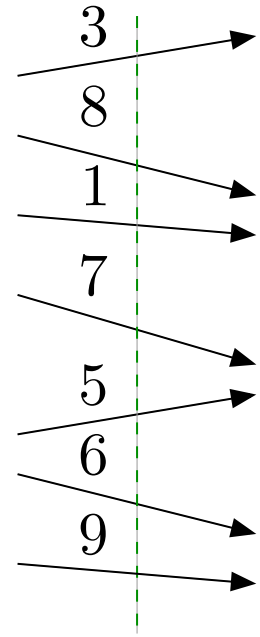
Descrição combinatória da linha

O conjunto dos segmentos na linha sofre inserções e remoções.

Como a linha vai nos ajudar a detectar interseção?

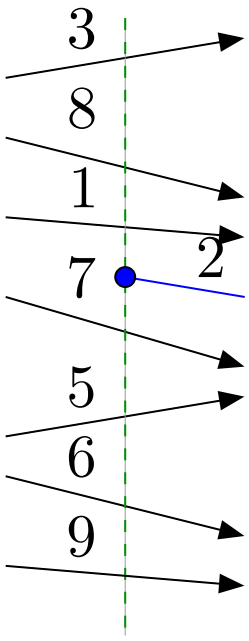
Ideia: testar interseção apenas entre segmentos "vizinhos na linha".

Para isso, mantemos os segmentos na linha ordenados.

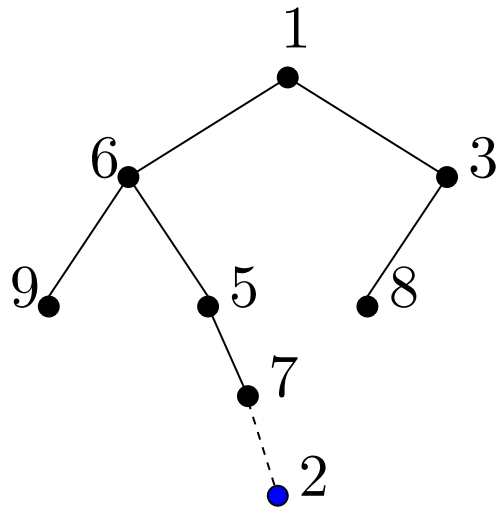


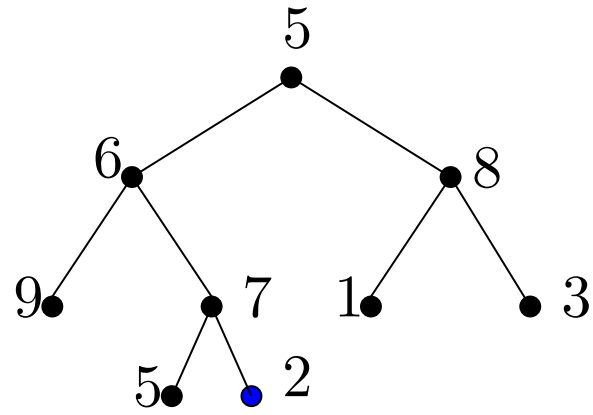
Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a linha.



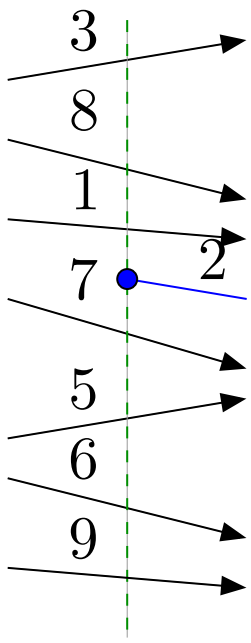
$$9 \prec 6 \prec 5 \prec 7 \prec 2 \prec 1 \prec 8 \prec 3$$



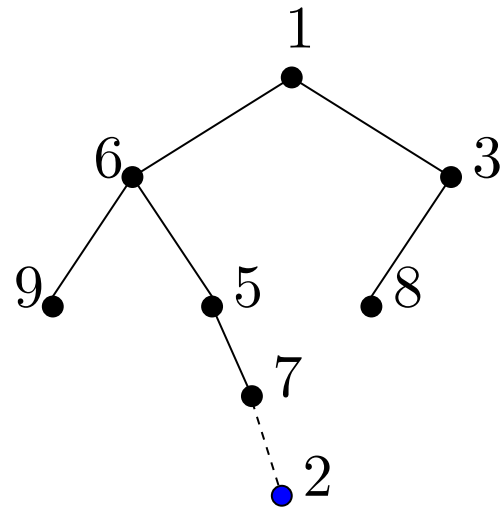


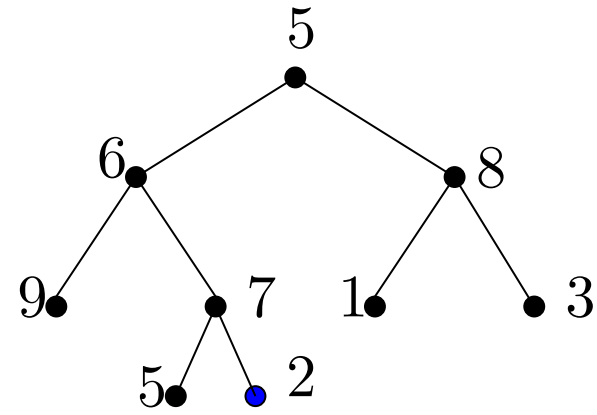
Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a linha.



$$9 \prec 6 \prec 5 \prec 7 \prec 2 \prec 1 \prec 8 \prec 3$$





Usamos uma árvore de busca binária balanceada (ABBB) para representar a descrição combinatória da linha.

Ordem usada

Se a linha de varredura é a reta $x = t$,
a ordem nos segmento é $\prec_t$.

Ordem usada

Se a linha de varredura é a reta $x = t$,
a ordem nos segmento é $\prec_t$.

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 segmento é $\prec_t$.

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

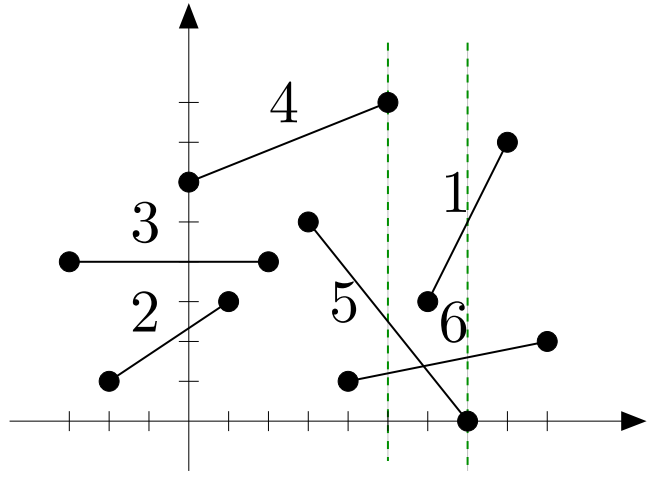
$i \prec_t j$ se a interseção da linha com i
fica abaixo da interseção com j.

Ordem usada

Se a linha de varredura é a reta $x = t$,
a ordem nos segmento é $\prec_t$.

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

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



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

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 da ABBB 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 $\prec_x$;
- SUCESSOR(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 da ABBB 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 $\prec_x$;

SUCESSOR(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 $\mathrm{O}(\lg m)$, onde m é o número de elementos em T.

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.

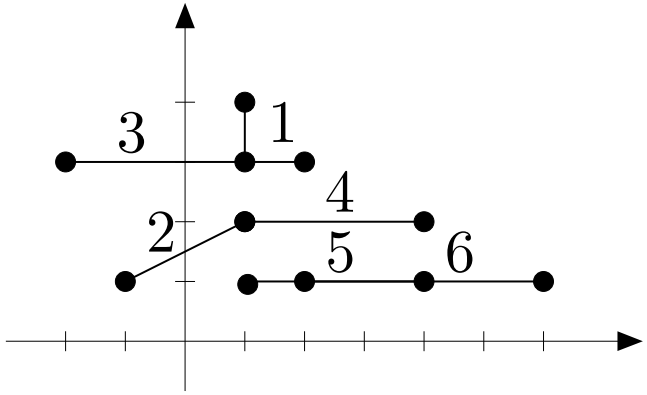
Saída: VERDADE se há dois segmentos na coleção que se intersectam, e FALSO caso contrário.

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.

Simplificação: não há dois pontos extremos com a mesma coordenada X e não há dois segmentos que se intersectam em mais do que um ponto, como abaixo.



e_X	1	−1	−2	1	1	2
e_Y	3	1	3	2	1	1
	1	2	3	4	5	6

d_X	1	1	2	4	4	6
d_Y	4	2	3	2	1	1
	1	2	3	4	5	6

Montagem da fila de eventos

FILADEEVENTOS:
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)

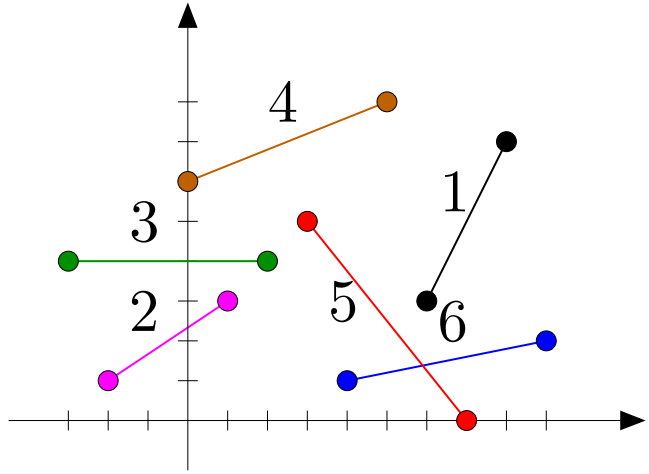
devolve

$E[1..2n]$: 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

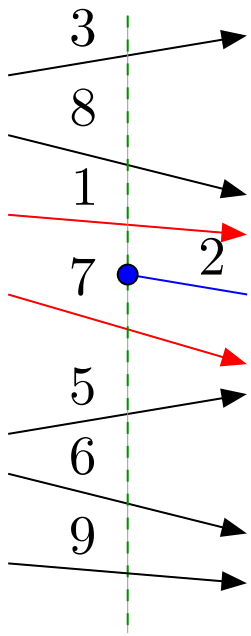
Dois tipos:

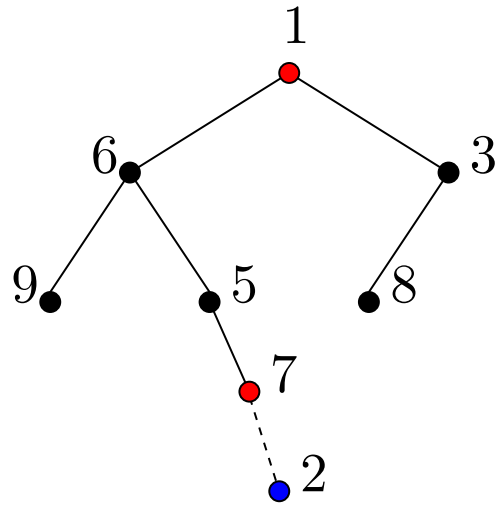
- começo de segmento: inclui o novo segmento na ABBB e verifica interseção com seus dois novos “vizinhos”.

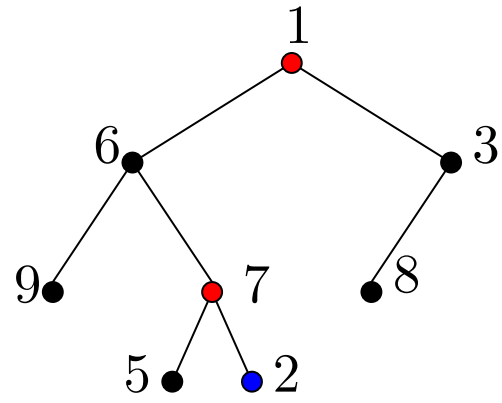
Processamento de ponto evento

Dois tipos:

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



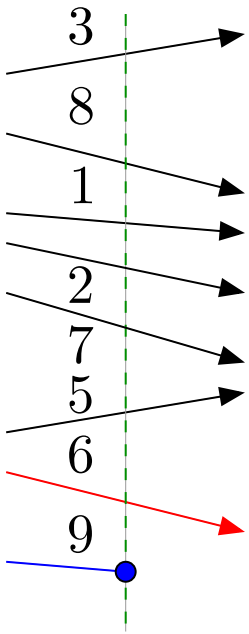


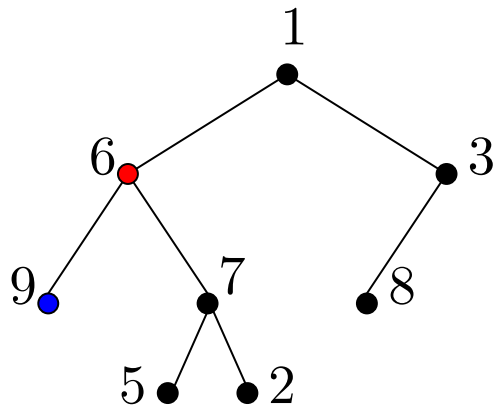


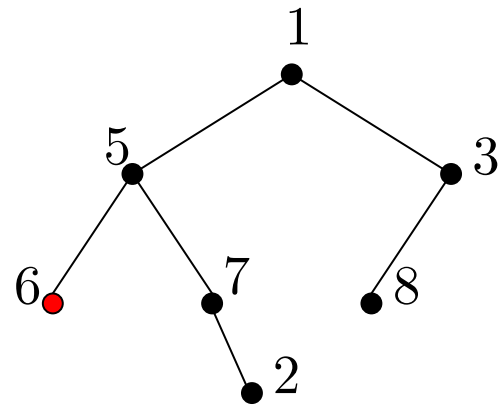
Processamento de ponto evento

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

Algoritmo de Shamos e Hoey

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

Consumo de tempo: $O(n \lg n)$

Algoritmo de Shamos e Hoey

E para encontrar todos os pontos de interseção?

Algoritmo de Shamos e Hoey

E para encontrar todos os pontos de interseção?

A fila de eventos seria dinâmica:
pontos de interseção seriam eventos também!

Algoritmo de Shamos e Hoey

E para encontrar todos os pontos de interseção?

A fila de eventos seria dinâmica:
pontos de interseção seriam eventos também!

Como processar um ponto evento que é uma interseção?

Algoritmo de Shamos e Hoey

E para encontrar todos os pontos de interseção?

A fila de eventos seria dinâmica:
pontos de interseção seriam eventos também!

Como processar um ponto evento que é uma interseção?

Tal ponto faz dois segmentos trocarem de ordem na ABBB.

Algoritmo de Shamos e Hoey

E para encontrar todos os pontos de interseção?

A fila de eventos seria dinâmica:
pontos de interseção seriam eventos também!

Como processar um ponto evento que é uma interseção?

Tal ponto faz dois segmentos trocarem de ordem na ABBB.

Temos que verificar interseção entre
cada um deles e seu “novo vizinho”.

Algoritmo de Shamos e Hoey

E para encontrar todos os pontos de interseção?

A fila de eventos seria dinâmica:
pontos de interseção seriam eventos também!

Como processar um ponto evento que é uma interseção?

Tal ponto faz dois segmentos trocarem de ordem na ABBB.

Temos que verificar interseção entre
cada um deles e seu “novo vizinho”.

Vamos ver uma animação!

Obrigada!!!