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

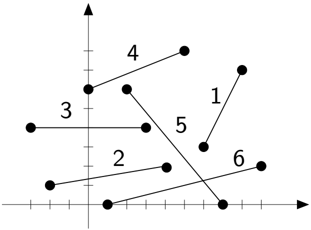
Uma coleção de segmentos do plano é dada por dois vetores $e[1\,..\,n], d[1\,..\,n]$ de pontos.

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

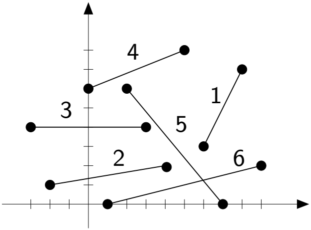


	1	2	3	4	5	6
e_X	6	-2	-3	0	2	1
e_Y	3	1	4	6	6	0

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

Interseção de segmentos

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

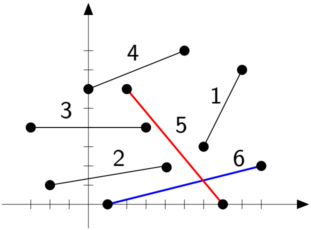


	1	2	3	4	5	6
e_X	6	-2	-3	0	2	1
e_Y	3	1	4	6	6	0

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

Interseção de segmentos

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



	1	2	3	4	5	6
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

Resposta: sim, existem dois segmentos com interseção.

Interseção de dois segmentos

Da aula passada...

Interseção entre *ab* e *cd*

```
Intersecta(a, b, c, d)
  se IntersectaProp(a, b, c, d)
      então devolva verdade
  devolva Entre(a, b, c) ou Entre(a, b, d)
          ou Entre(c, d, a) ou Entre(c, d, b)
```

Interseção de dois segmentos

Da aula passada...

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)

para $i \leftarrow 1$ até $n-1$ faça
    para $j \leftarrow i+1$ até n faça
        se Inter (e, d, i, j)
            então devolva verdade
devolva falso

Interseção de segmentos

Solução quadrática:

```
IntersectaQuad(e, d, n)
para i ← 1 até n−1 faça
    para j ← i+1 até n faça
        se Inter (e, d, i, j)
                então devolva verdade
devolva falso
```

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

Interseção de segmentos

Solução quadrática:

```
IntersectaQuad(e, d, n)
para i ← 1 até n−1 faça
    para j ← i+1 até n faça
        se Inter (e, d, i, j)
            então devolva verdade
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 $[e_X[1]\,..\,d_X[1]], \ldots, [e_X[n]\,..\,d_X[n]]$.

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.

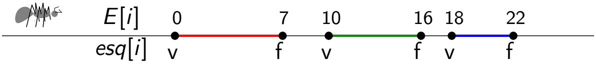
Basta contar quantos intervalos estão “abertos”.
Se houver mais do que um aberto num momento, há interseção.

Interseção de intervalos

Varredura(e, d, n)

para $i \leftarrow 1$ até n faça ▷ para cada intervalo marca
    $E[i] \leftarrow e_X[i]$ $esq[i] \leftarrow$ verdade ▷ extremo esquerdo
    $E[i+n] \leftarrow d_X[i]$ $esq[i+n] \leftarrow$ falso ▷ extremo direito
MergeSort$(E, esq, 1, 2n)$ ▷ ordena os extremos

e_X	10	0	18
d_X	16	7	22
	1	2	3

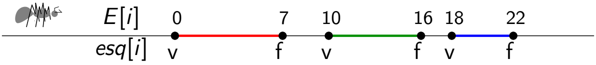


Interseção de intervalos

Varredura(e, d, n)

para $i \leftarrow 1$ até n faça ▷ para cada intervalo marca
  $E[i] \leftarrow e_X[i]$ $esq[i] \leftarrow$ verdade ▷ extremo esquerdo
  $E[i+n] \leftarrow d_X[i]$ $esq[i+n] \leftarrow$ falso ▷ extremo direito
MergeSort($E, esq, 1, 2n$) ▷ ordena os extremos
$cont \leftarrow 0$ $resp \leftarrow$ falso
para $p \leftarrow 1$ até $2n$ faça ▷ para cada ponto extremo
  se $esq[p]$ ▷ se extremo esquerdo
    então $cont \leftarrow cont+1$
      se $cont = 2$ então $resp \leftarrow$ verdade
    senão $cont \leftarrow cont-1$
devolva $resp$

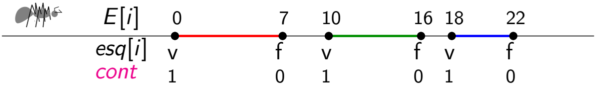
e_X	10	0	18
d_X	16	7	22
	1	2	3



Interseção de intervalos

```
Varredura(e, d, n)
para i ← 1 até n faça                          ▷ para cada intervalo marca
    E[i] ← e_X[i]          esq[i] ← verdade     ▷ extremo esquerdo
    E[i + n] ← d_X[i]      esq[i + n] ← falso   ▷ extremo direito
MergeSort(E, esq, 1, 2n)                       ▷ ordena os extremos
cont ← 0      resp ← falso
para p ← 1 até 2n faça                         ▷ para cada ponto extremo
    se esq[p]                                  ▷ se extremo esquerdo
        então cont ← cont+1
              se cont = 2 então resp ← verdade
        senão cont ← cont−1
devolva resp
```

e_X	10	0	18
d_X	16	7	22
	1	2	3



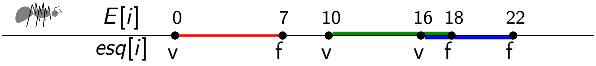
$\text{Varredura}(e_X, d_X, 3) = \text{falso}$

Interseção de intervalos

Varredura(e, d, n)

```
para i ← 1 até n faça                          ▷ para cada intervalo marca
    E[i] ← e_X[i]        esq[i] ← verdade       ▷ extremo esquerdo
    E[i + n] ← d_X[i]    esq[i + n] ← falso     ▷ extremo direito
MergeSort(E, esq, 1, 2n)                       ▷ ordena os extremos
cont ← 0      resp ← falso
para p ← 1 até 2n faça                         ▷ para cada ponto extremo
    se esq[p]                                  ▷ se extremo esquerdo
        então cont ← cont+1
              se cont = 2 então resp ← verdade
        senão cont ← cont−1
devolva resp
```

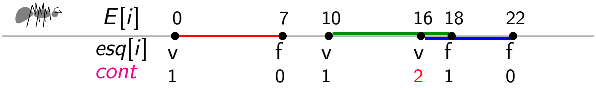
	1	2	3
e_X	10	0	16
d_X	18	7	22



Interseção de intervalos

```
Varredura(e, d, n)
para i ← 1 até n faça                       ▷ para cada intervalo marca
    E[i] ← e_X[i]         esq[i] ← verdade      ▷ extremo esquerdo
    E[i + n] ← d_X[i]     esq[i + n] ← falso    ▷ extremo direito
MergeSort(E, esq, 1, 2n)                    ▷ ordena os extremos
cont ← 0      resp ← falso
para p ← 1 até 2n faça                      ▷ para cada ponto extremo
    se esq[p]                                   ▷ se extremo esquerdo
        então cont ← cont+1
              se cont = 2 então resp ← verdade
        senão cont ← cont−1
devolva resp
```

	1	2	3
e_X	10	0	16
d_X	18	7	22



Varredura(e_X, d_X, 3) = verdade

Interseção de intervalos

```
Varredura(e, d, n)
para i ← 1 até n faça                    ▷ para cada intervalo marca
    E[i] ← e_X[i]        esq[i] ← verdade        ▷ extremo esquerdo
    E[i + n] ← d_X[i]    esq[i + n] ← falso      ▷ extremo direito
MergeSort(E, esq, 1, 2n)                 ▷ ordena os extremos
cont ← 0     resp ← falso
para p ← 1 até 2n faça                   ▷ para cada ponto extremo
    se esq[p]                                ▷ se extremo esquerdo
        então cont ← cont+1
                  se cont = 2 então resp ← verdade
        senão cont ← cont−1
devolva resp
```

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

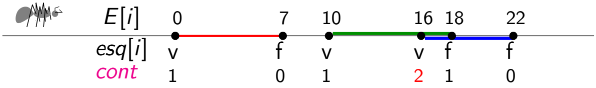
Método da linha de varredura

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

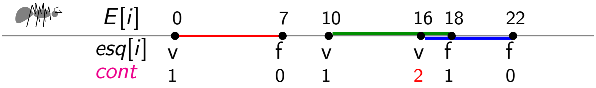
Uma linha imaginária move-se da esquerda para a direita.



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.

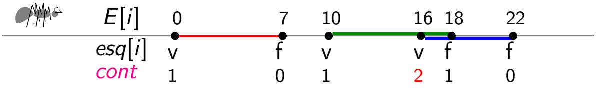


À medida que ela move,
o problema restrito à esquerda dela é resolvido.

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.



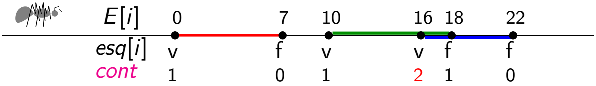
À medida que ela move,
o problema restrito à esquerda dela é resolvido.

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.



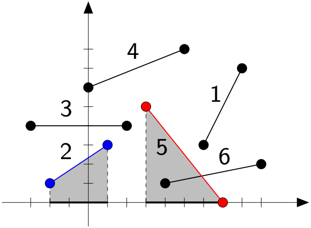
À medida que ela move,
o problema restrito à esquerda dela é resolvido.

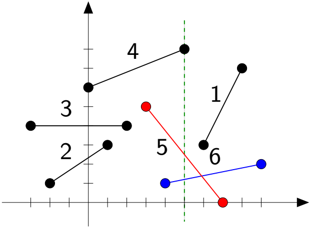
Informação necessária para estender a solução parcial
é mantida numa descrição combinatória da linha.

Muda apenas em posições chaves: os pontos eventos.

Algoritmo de Shamos e Hoey

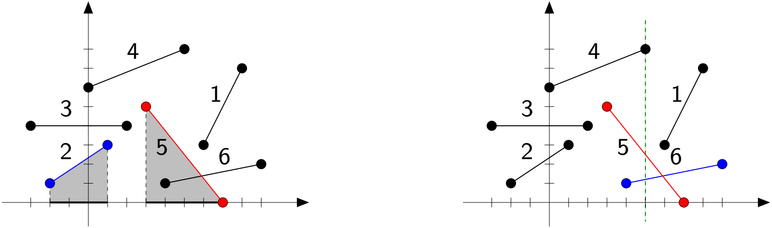
Ideia: Dois segmentos cujas projeções no eixo X sejam disjuntas não se intersectam.





Algoritmo de Shamos e Hoey

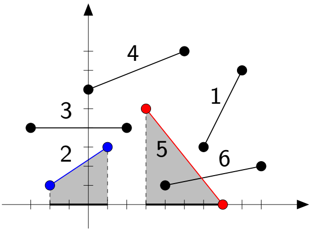
Ideia: Dois segmentos cujas projeções no eixo X sejam disjuntas não se intersectam.

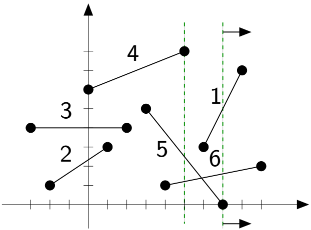


Se as projeções no eixo X de dois segmentos têm interseção, então há uma linha vertical que intersecta ambos.

Algoritmo de Shamos e Hoey

Ideia: Dois segmentos cujas projeções no eixo X sejam disjuntas não se intersectam.

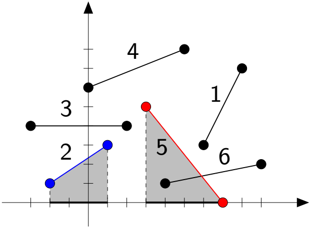


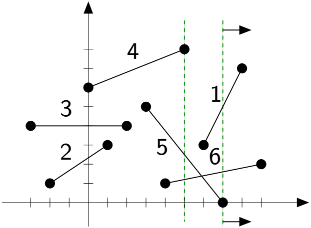


Imagine esta linha vertical varrendo o plano da esquerda para a direita...

Algoritmo de Shamos e Hoey

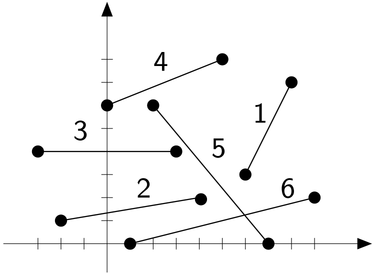
Ideia: Dois segmentos cujas projeções no eixo X sejam disjuntas não se intersectam.





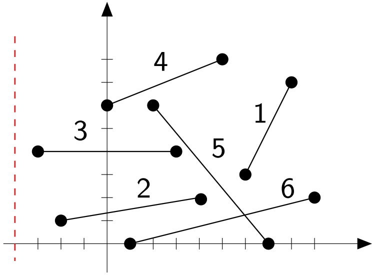
Imagine esta linha vertical varrendo o plano
da esquerda para a direita...
Enquanto a linha varre o plano, mantemos os segmentos intersectados por ela na descrição combinatória da linha.

Descrição combinatória da linha



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

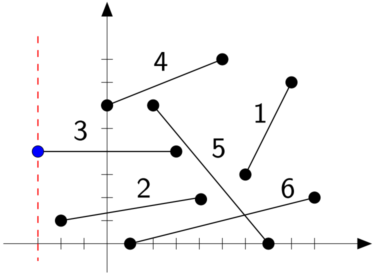
Descrição combinatória da linha



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

Alterações ocorrem
nos extremos dos segmentos.

Descrição combinatória da linha

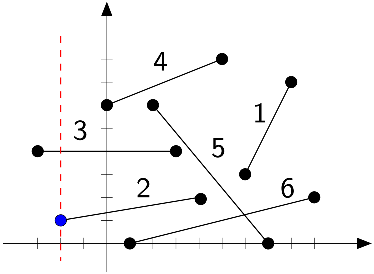


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

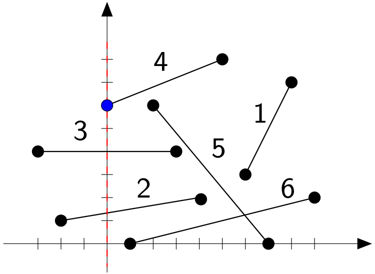


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

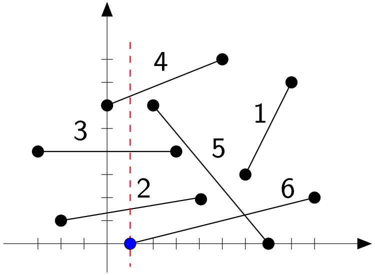


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

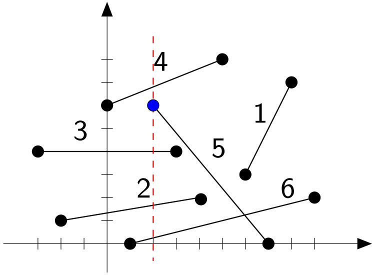


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

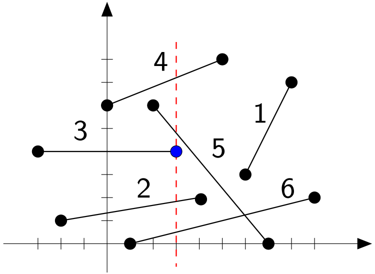


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

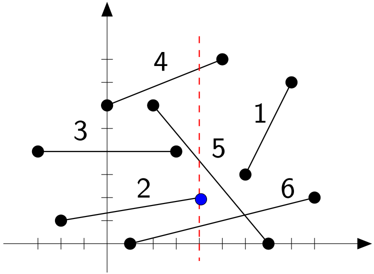


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

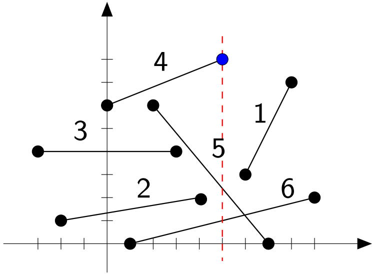


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

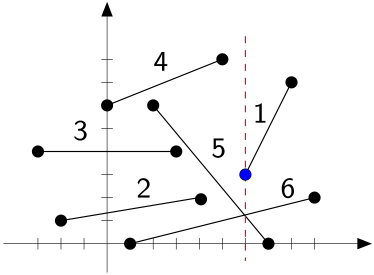


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

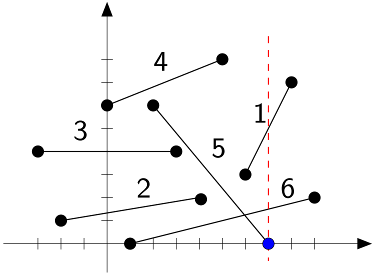


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

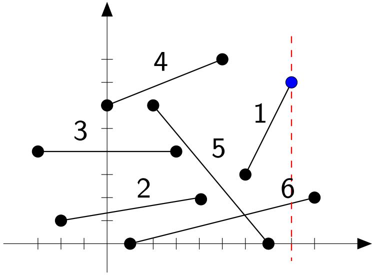


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

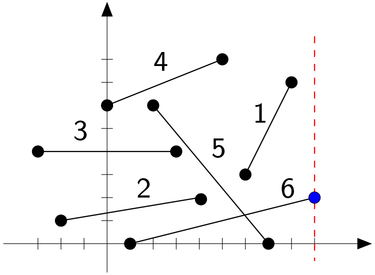


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

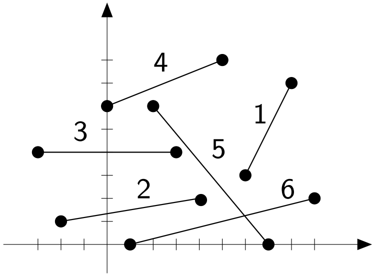


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

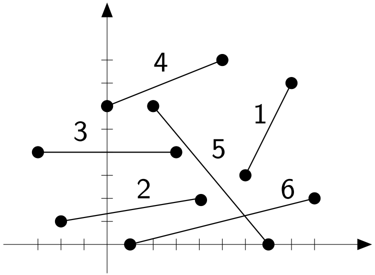
Descrição combinatória da linha



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

Como guardar
um destes conjuntos?

Descrição combinatória da linha



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

Como guardar
um destes conjuntos?

Que operações ele sofre?

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.

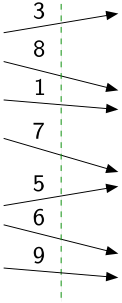
Como a linha vai nos ajudar a detectar interseção?

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".



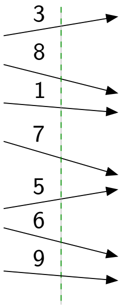
Descrição combinatória da linha

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.



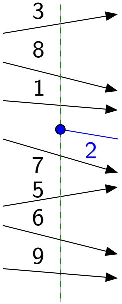
Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a linha.

Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a linha.

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

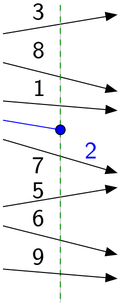


Ao inserimos um segmento, testamos a interseção dele com seu predecessor e com seu sucessor na ordem.

Descrição combinatória da linha

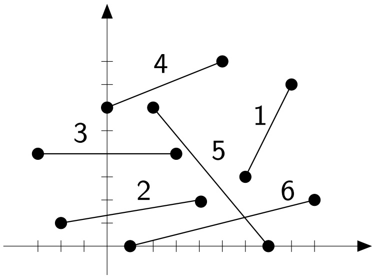
Os segmentos ficam na ordem em que intersectam a linha.

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



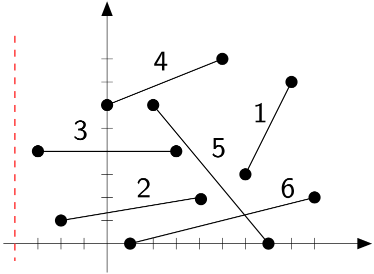
Ao removermos um segmento, testamos a interseção de seu predecessor e com seu sucessor na ordem.

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	$\cdots$
6	
7	
8	

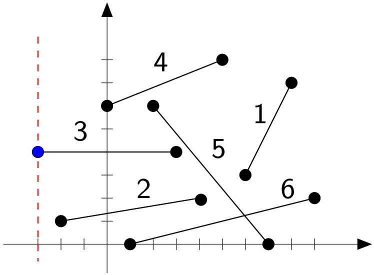
Algoritmo de Shamos e Hoey



Alterações ocorrem
nos extremos dos segmentos.

$-\infty$	
-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

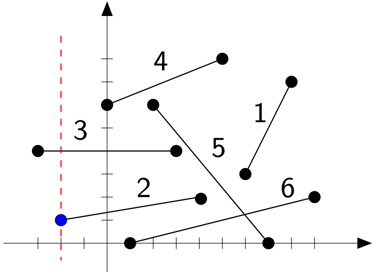


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

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

Algoritmo de Shamos e Hoey

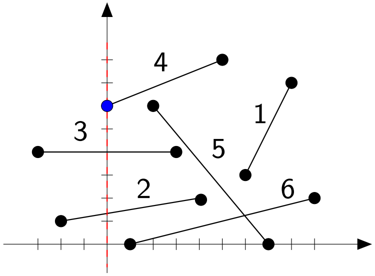


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

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

Algoritmo de Shamos e Hoey

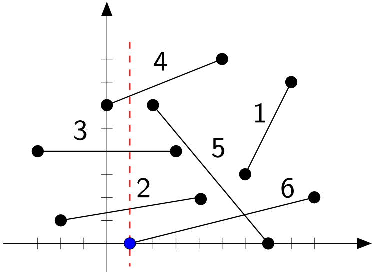


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

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

Algoritmo de Shamos e Hoey

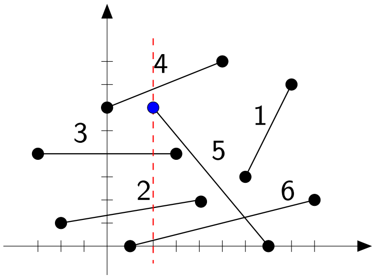


$-\infty$	
-3	3
-2	$2 \prec 3$
0	$2 \prec 3 \prec 4$
1	$6 \prec 2 \prec 3 \prec 4$
2	$6 \prec 2 \prec 3 \prec 5 \prec 4$
3	$6 \prec 2 \prec 5 \prec 4$
4	$6 \prec 5 \prec 4$
5	
6	
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Algoritmo de Shamos e Hoey

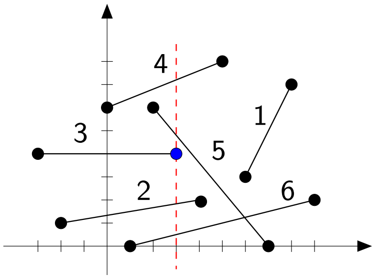


$-\infty$	
-3	3
-2	$2 \prec 3$
0	$2 \prec 3 \prec 4$
1	$6 \prec 2 \prec 3 \prec 4$
2	$6 \prec 2 \prec 3 \prec 5 \prec 4$
3	$6 \prec 2 \prec 5 \prec 4$
4	$6 \prec 5 \prec 4$
5	
6	
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Algoritmo de Shamos e Hoey

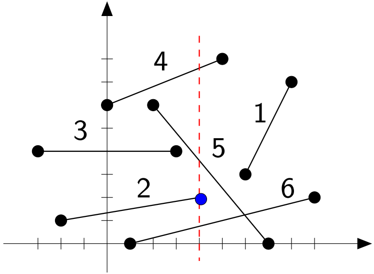


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

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

Algoritmo de Shamos e Hoey

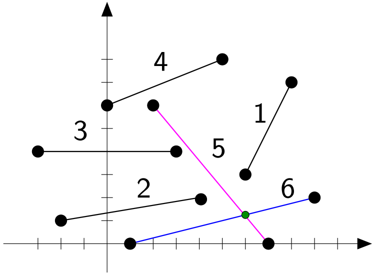


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

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

Algoritmo de Shamos e Hoey



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

Efetuaremos inserções, remoções, predecessor e sucessor neste conjunto.

Descrição combinatória da linha

Como guardar esse conjunto ordenado?

Efetuaremos inserções, remoções,
predecessor e sucessor neste conjunto.

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

Descrição combinatória da linha

Como guardar esse conjunto ordenado?

Efetuaremos inserções, remoções,
predecessor e sucessor neste conjunto.

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

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

Numa treap ou skip list,
custo esperado por operação é $\mathrm{O}(\lg m)$.

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

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 X-coordenadas

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 X-coordenadas

$segm[1..2n]$:
$segm[p]$: 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)

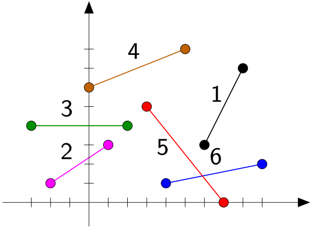
devolve

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

$segm[1\,..\,2n]$:
$segm[p]$: 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

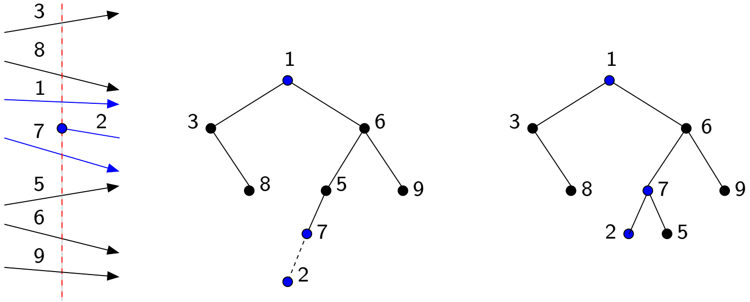
Dois tipos:

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

Processamento de ponto evento

Dois tipos:

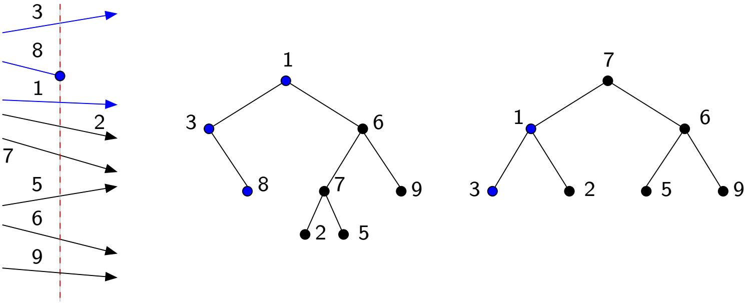
- começo de segmento: inclui o novo segmento na ABBB e verifica interseção com seus dois novos "vizinhos".



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.



(As árvores estão com esquerda/direita trocadas.)

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)
(E, segm, esq) ← FilaDeEventos(e, d, n)
Crie(T)
para p ← 1 até 2n faça
    i ← segm[p]
    pred ← Predecessor(T, E_X[p], E_Y[p])
    suc ← Sucessor(T, E_X[p], E_Y[p])
    se esq[p]
        então Insere(T, i)
              se (pred ≠ NIL e Inter(e, d, i, pred))
              ou (suc ≠ NIL e Inter(e, d, i, suc))
                  então devolva verdade
        senão Remove(T, i)
              se pred ≠ NIL e suc ≠ NIL e Inter(e, d, pred, suc)
                  então devolva verdade
devolva falso
```

Consumo de tempo

O algoritmo executa $2n$ iterações.

Cada iteração faz uma chamada a Predecessor,
uma a Sucessor, e uma a Insere ou a Remove.

Na ABBB, em qualquer momento, há $\mathrm{O}(n)$ segmentos.

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

As demais operações efetuadas em uma iteração
consomem tempo $\mathrm{O}(1)$ (mesmo as chamadas a Inter).

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

Casos degenerados

Como se livrar da hipótese simplificadora?

Lembre-se:

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

Em particular, não há segmentos verticais,
nem dois segmentos com extremos coincidentes.

Casos degenerados

Como se livrar da hipótese simplificadora?

Lembre-se:

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

Em particular, não há segmentos verticais,
nem dois segmentos com extremos coincidentes.

Ordene pontos com mesma X-coordenada
colocando extremos esquerdos antes dos direitos.

Para segmentos verticais,
chame de esquerdo um extremo arbitrário, e o outro de direito.

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 $\mathrm{O}(n \lg n)$ 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 $\mathrm{O}(n \lg n)$ para este problema?

No máximo, quantos pares teremos que imprimir?