Geometria Computacional

Cristina G. Fernandes
Departamento de Ciência da Computação do IME-USP
`http://www.ime.usp.br/~cris/`

Escola de Verão da Maratona de Programação

janeiro de 2020

Descrição da aula

Parte I: Algoritmos para fecho convexo

Parte II: Algoritmos de linha de varredura

Parte III: Animação dos algoritmos

Interseção de segmentos

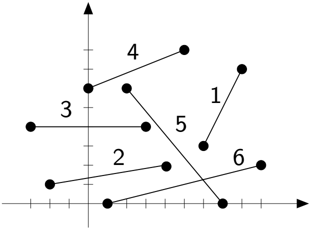
Uma coleção de segmentos do plano é dada por dois vetores $e[1\,..\,n], d[1\,..\,n]$ de pontos.

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

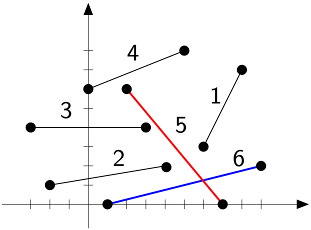


e_X	6	-2	-3	0	3	4
e_Y	3	1	4	6	5	1
	1	2	3	4	5	6

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.

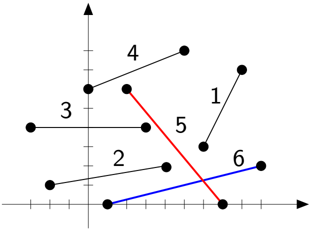


e_X	6	-2	-3	0	3	4
e_Y	3	1	4	6	5	1
	1	2	3	4	5	6

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 segmentos com interseção.

Predicados geométricos

Intersecta:

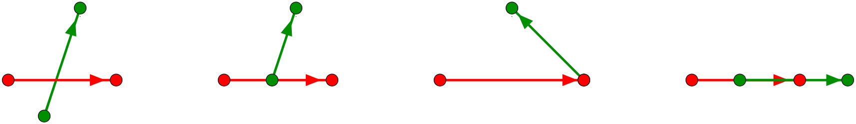
Recebe dois segmentos e devolve verdade se
os segmentos se intersectam, e falso caso contrário.

Predicados geométricos

Intersecta:

Recebe dois segmentos e devolve verdade se
os segmentos se intersectam, e falso caso contrário.

Em geral, os extremos de cada segmento devem estar
em lados opostos do outro segmento, mas há casos especiais.

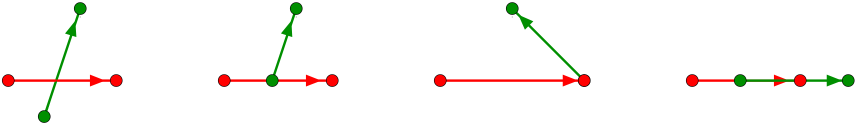


Predicados geométricos

Intersecta:

Recebe dois segmentos e devolve verdade se
os segmentos se intersectam, e falso caso contrário.

Em geral, os extremos de cada segmento devem estar
em lados opostos do outro segmento, mas há casos especiais.



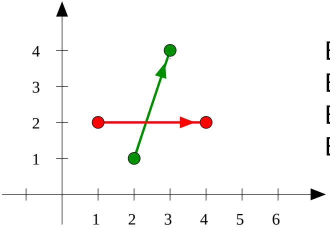
Hipótese simplificadora:
não há três extremos dos segmentos colineares.

Neste caso, é fácil detectar interseção de dois segmentos.

Interseção para pontos em posição geral

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

se Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq$ Esquerda$((x_1, y_1), (x_2, y_2), (x_4, y_4))$
    e Esquerda$((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq$ Esquerda$((x_3, y_3), (x_4, y_4), (x_2, y_2))$
        então devolva verdade
        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 para pontos em posição geral

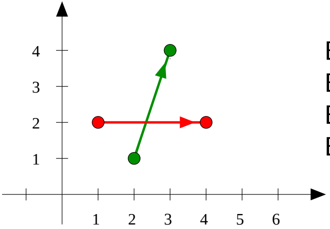
$\text{Intersecta}((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4))$

1 se $\text{Esquerda}((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq \text{Esquerda}((x_1, y_1), (x_2, y_2), (x_4, y_4))$

e $\text{Esquerda}((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq \text{Esquerda}((x_3, y_3), (x_4, y_4), (x_2, y_2))$

2 então devolva verdade

3 senão devolva falso



$\text{Esquerda}((1, 2), (4, 2), (2, 1)) = \text{falso}$

$\text{Esquerda}((1, 2), (4, 2), (3, 4)) = \text{verdade}$

$\text{Esquerda}((2, 1), (3, 4), (1, 2)) = \text{verdade}$

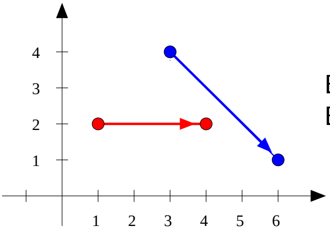
$\text{Esquerda}((2, 1), (3, 4), (4, 2)) = \text{falso}$

$\text{Intersecta}((1, 2), (4, 2), (2, 1), (3, 4)) = \text{verdade}$

Interseção para pontos em posição geral

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

se Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3)) \neq$ Esquerda$((x_1, y_1), (x_2, y_2), (x_4, y_4))$
 e Esquerda$((x_3, y_3), (x_4, y_4), (x_1, y_1)) \neq$ Esquerda$((x_3, y_3), (x_4, y_4), (x_2, y_2))$
 então devolva verdade
 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 para pontos em posição geral

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

```
se Esquerda((x1, y1), (x2, y2), (x3, y3)) ≠ Esquerda((x1, y1), (x2, y2), (x4, y4))
   e Esquerda((x3, y3), (x4, y4), (x1, y1)) ≠ Esquerda((x3, y3), (x4, y4), (x2, y2))
      então devolva verdade
      senão devolva falso
```

Abreviatura:

Inter(e, d, i, j)

```
devolva Intersecta(e[i], d[i], e[j], d[j])
```

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
```

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

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

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

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

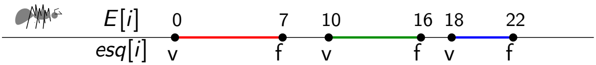
Se ordenarmos os pontos extremos dos intervalos,
é fácil decidir se há interseção ou não,
percorrendo os pontos na ordem obtida.

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

Interseção de intervalos em $O(n \lg n)$

```
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
```

	1	2	3
e_X	10	0	18
d_X	16	7	22

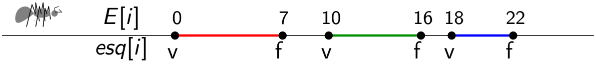


Interseção de intervalos em $O(n \lg n)$

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	18
d_X	16	7	22

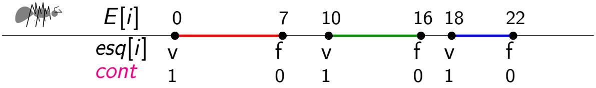


Interseção de intervalos em $O(n \lg n)$

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	18
d_X	16	7	22

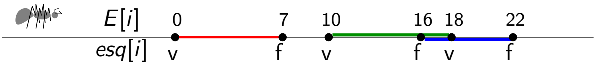


Varredura($e_X, d_X, 3$) = falso

Interseção de intervalos em O($n \lg n$)

```
Varredura(e, d, n)
para i ← 1 até n faça                      ▷ para cada intervalo marca
    E[i] ← eX[i]          esq[i] ← verdade        ▷ extremo esquerdo
    E[i + n] ← dX[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	16
d_X	18	7	22
	1	2	3



Varredura(e_X, d_X, 3) = verdade

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

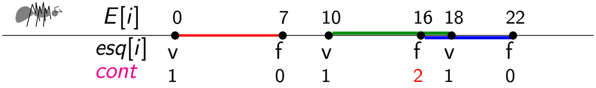
Uma linha imaginária move-se da esquerda para a direita.

$E[i]$	0	7	10	16	18	22
$esq[i]$	v	f	v	f	v	f
cont	1	0	1	2	1	0

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.

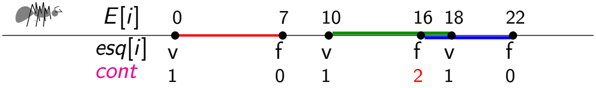


À medida que ela move,
o problema restrito à esquerda dela é resolvido.

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.



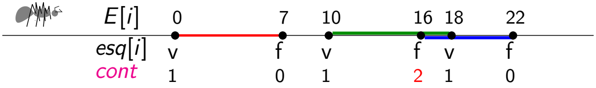
À medida que ela move,
o problema restrito à esquerda dela é resolvido.

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.



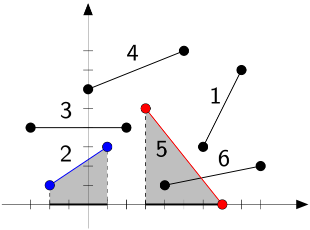
À medida que ela move,
o problema restrito à esquerda dela é resolvido.

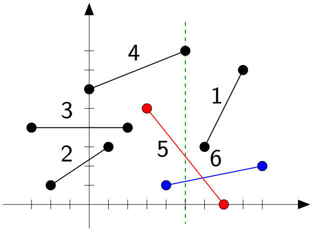
Informação necessária para estender a solução parcial
é mantida numa descrição combinatória da linha.

Muda apenas em posições chaves: os pontos eventos.

Algoritmo de Shamos e Hoey

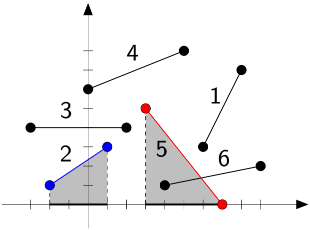
Ideia: Dois segmentos cuja projeção no eixo X sejam disjuntas não se intersectam.

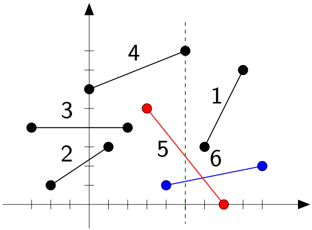




Algoritmo de Shamos e Hoey

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

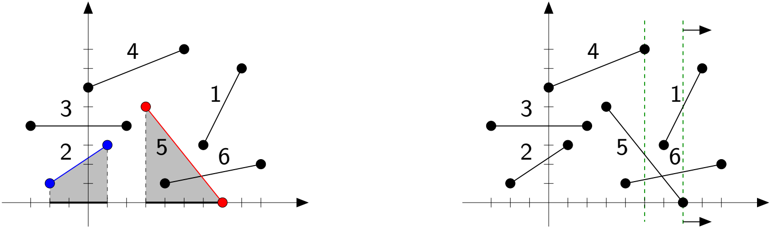




Se a projeção no eixo X de dois segmentos tem interseção, então há uma linha vertical que intersecta ambos.

Algoritmo de Shamos e Hoey

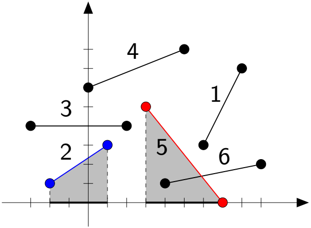
Ideia: Dois segmentos cuja projeção no eixo X sejam disjuntas não se intersectam.

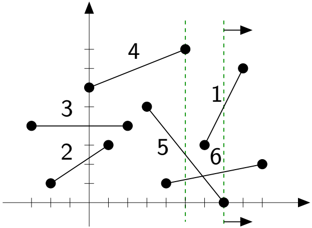


Imagine esta linha vertical varrendo o plano da esquerda para a direita...

Algoritmo de Shamos e Hoey

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

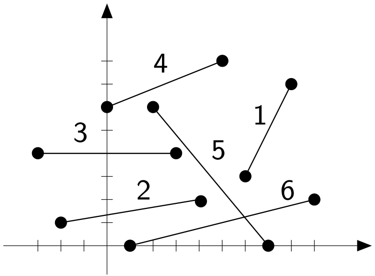




Imagine esta linha vertical varrendo o plano
da esquerda para a direita...

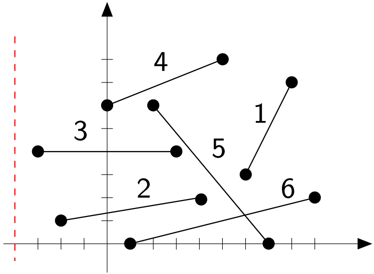
Enquanto a linha varre o plano, mantemos os segmentos intersectados por ela na descrição combinatória da linha.

Descrição combinatória da linha



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

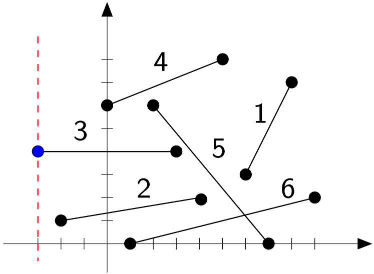
Descrição combinatória da linha



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

Alterações ocorrem
nos extremos dos segmentos.

Descrição combinatória da linha

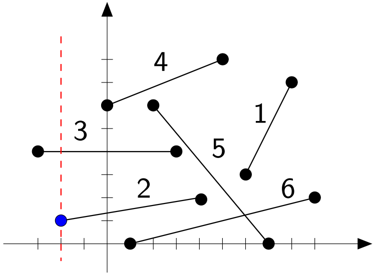


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

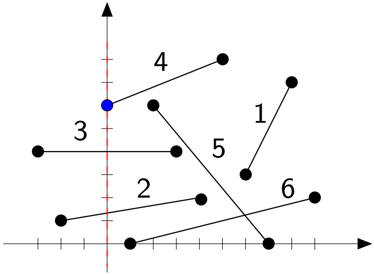


Alterações ocorrem
nos extremos dos segmentos.

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

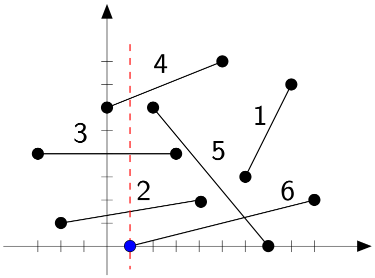


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

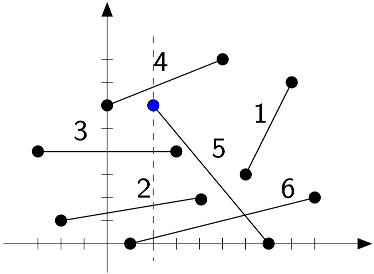


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

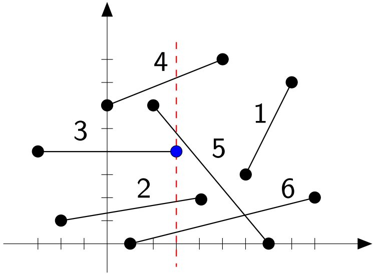


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

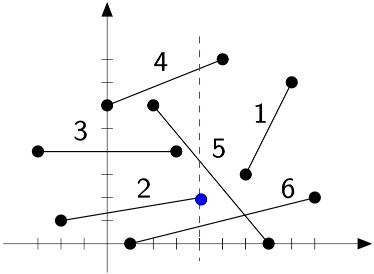


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

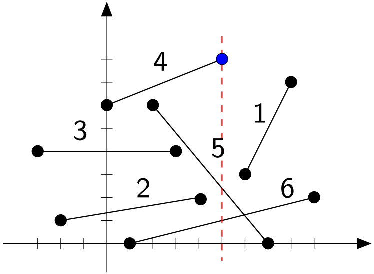


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

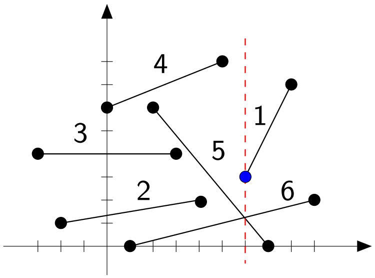


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

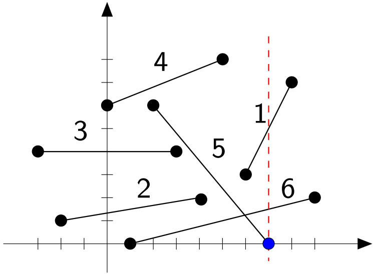


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

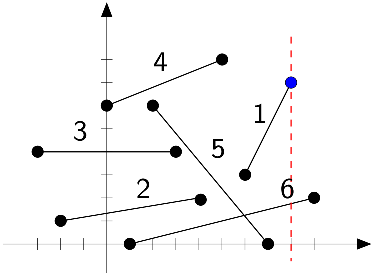


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

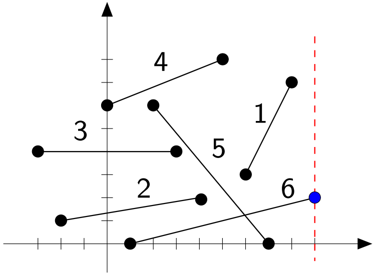


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2, 3\}$
$0 \leq x < 1$	$\{2, 3, 4\}$
$1 \leq x \leq 2$	$\{2, 3, 4, 6\}$
$2 < x < 3$	$\{2, 3, 4, 5, 6\}$
$3 \leq x < 4$	$\{2, 4, 5, 6\}$
$4 \leq x \leq 5$	$\{4, 5, 6\}$
$5 < x < 6$	$\{5, 6\}$
$6 \leq x \leq 7$	$\{1, 5, 6\}$
$7 < x \leq 8$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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Descrição combinatória da linha

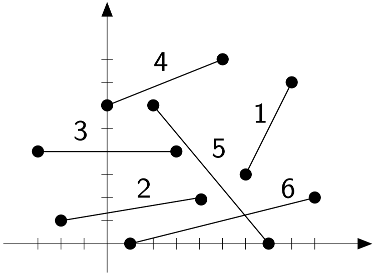


$x < -3$	$\emptyset$
$-3 \leq x < -2$	$\{3\}$
$-2 \leq x < 0$	$\{2,3\}$
$0 \leq x < 1$	$\{2,3,4\}$
$1 \leq x \leq 2$	$\{2,3,4,6\}$
$2 < x < 3$	$\{2,3,4,5,6\}$
$3 \leq x < 4$	$\{2,4,5,6\}$
$4 \leq x \leq 5$	$\{4,5,6\}$
$5 < x < 6$	$\{5,6\}$
$6 \leq x \leq 7$	$\{1,5,6\}$
$7 < x \leq 8$	$\{1,6\}$
$8 < x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

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

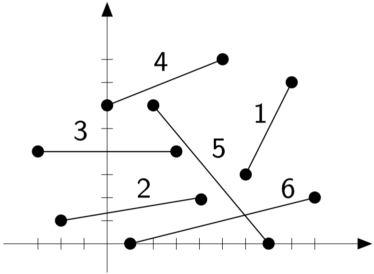
Descrição combinatória da linha



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

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.

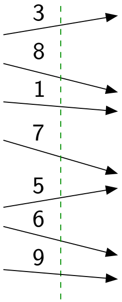
Como a linha vai nos ajudar a detectar interseção?

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".



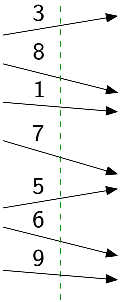
Descrição combinatória da linha

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.



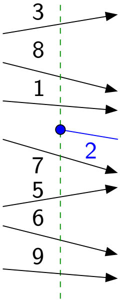
Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a linha.

Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a linha.

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

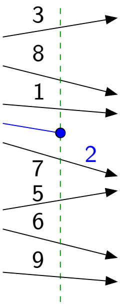


Ao inserimos um segmento, testamos a interseção dele com seu predecessor e com seu sucessor na ordem.

Descrição combinatória da linha

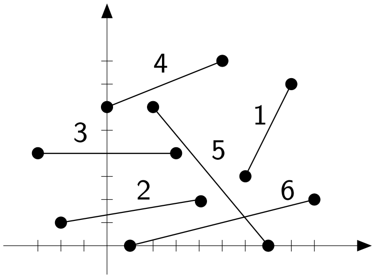
Os segmentos ficam na ordem em que intersectam a linha.

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



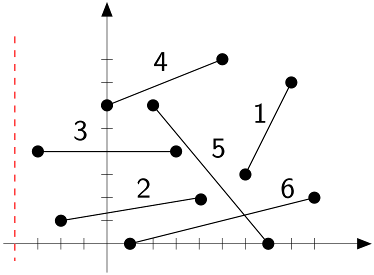
Ao removermos um segmento, testamos a interseção
de seu predecessor e com seu sucessor na ordem.

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	

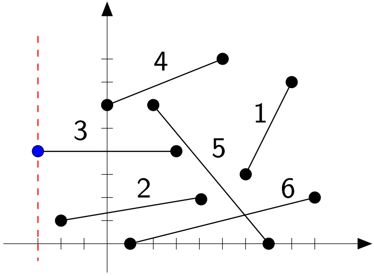
Algoritmo de Shamos e Hoey



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

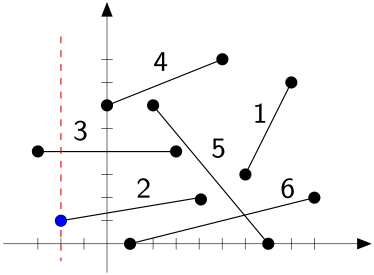


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

$-\infty$	
-3	3
-2	$3 \prec 2$
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

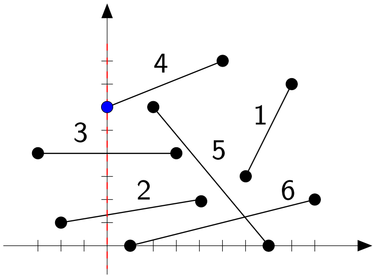


$-\infty$	
-3	3
-2	$3 \prec 2$
0	$4 \prec 3 \prec 2$
1	$4 \prec 3 \prec 2 \prec 6$
2	$4 \prec 5 \prec 3 \prec 2 \prec 6$
3	$4 \prec 5 \prec 2 \prec 6$
4	$4 \prec 5 \prec 6$
5	$5 \prec 6$
6	$\cdots$
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Algoritmo de Shamos e Hoey

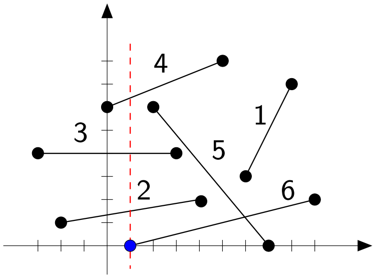


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

$-\infty$	
-3	3
-2	$3 \prec 2$
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

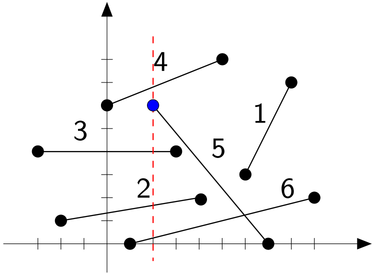


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

$-\infty$	
-3	3
-2	$3 \prec 2$
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

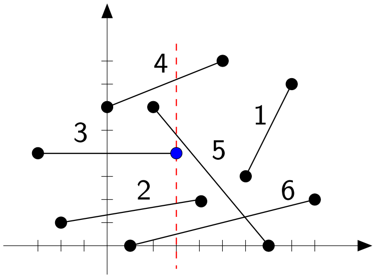


$-\infty$	
-3	3
-2	$3 \prec 2$
0	$4 \prec 3 \prec 2$
1	$4 \prec 3 \prec 2 \prec 6$
2	$4 \prec 5 \prec 3 \prec 2 \prec 6$
3	$4 \prec 5 \prec 2 \prec 6$
4	$4 \prec 5 \prec 6$
5	
6	
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

Algoritmo de Shamos e Hoey

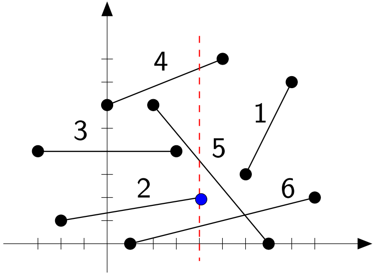


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

$-\infty$	
-3	3
-2	$3 \prec 2$
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

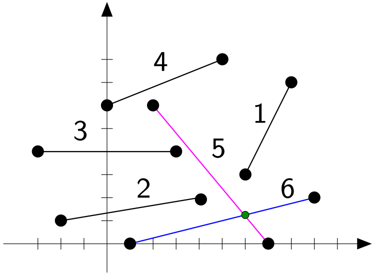


Alterações ocorrem
nos extremos dos segmentos.

Estes são os pontos eventos.

$-\infty$	
-3	3
-2	$3 \prec 2$
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.

Encontrou uma interseção!

$-\infty$	
-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	

Descrição combinatória da linha

Como guardar esse conjunto ordenado de segmentos?

Descrição combinatória da linha

Como guardar esse conjunto ordenado de segmentos?

Efetuaremos inserções, remoções, predecessor e sucessor neste conjunto.

Descrição combinatória da linha

Como guardar esse conjunto ordenado de segmentos?

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.

Descrição combinatória da linha

Como guardar esse conjunto ordenado de segmentos?

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.

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

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

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 três extremos dos segmentos colineares.

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 três extremos dos segmentos colineares.

Hipótese simplificadora extra:
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 $e[1\mathinner{\ldotp\ldotp}n]$ e $d[1\mathinner{\ldotp\ldotp}n]$ 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)

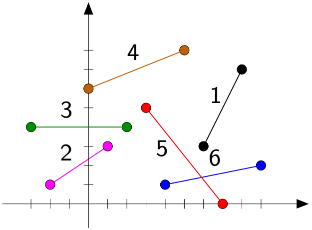
devolve

$E[1\mathinner{\ldotp\ldotp}2n]$: pontos de $e[1\mathinner{\ldotp\ldotp}n]$ e $d[1\mathinner{\ldotp\ldotp}n]$
ordenados pelas suas X-coordenadas

$segm[1\mathinner{\ldotp\ldotp}2n]$:
$segm[p]$: índice do segmento do qual $E[p]$ é extremo

$esq[1\mathinner{\ldotp\ldotp}2n]$:
$esq[p]$: 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

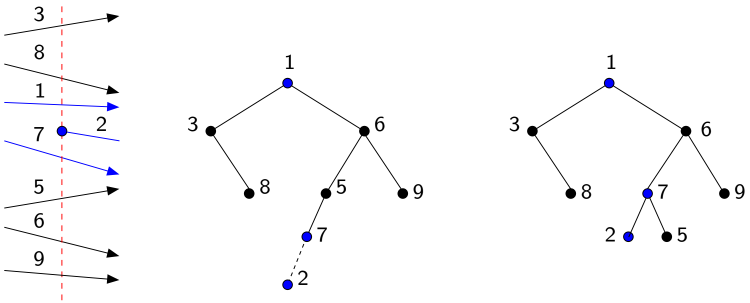
Dois tipos:

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

Processamento de ponto evento

Dois tipos:

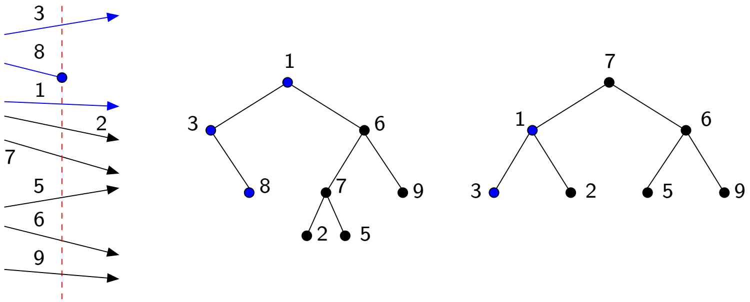
- começo de segmento: inclui o novo segmento na ABB e verifica interseção com seus dois novos "vizinhos".



Processamento de ponto evento

Dois tipos:

- começo de segmento: inclui o novo segmento na ABB e verifica interseção com seus dois novos "vizinhos".
- fim de segmento: remove o segmento da ABB e verifica interseção entre seus dois ex-vizinhos.



Processamento de ponto evento

Dois tipos:

- começo de segmento: inclui o novo segmento na ABB e verifica interseção com seus dois novos "vizinhos".
- fim de segmento: remove o segmento da ABB e verifica interseção entre seus dois ex-vizinhos.

Invariante: verificamos interseção entre quaiquer dois segmentos vizinhos na ABB.

Processamento de ponto evento

Dois tipos:

- começo de segmento: inclui o novo segmento na ABB e verifica interseção com seus dois novos "vizinhos".
- fim de segmento: remove o segmento da ABB e verifica interseção entre seus dois ex-vizinhos.

Invariante: verificamos interseção entre quaiquer dois segmentos vizinhos na ABB.

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

Algoritmo de Shamos e Hoey

Interseção-SH(e, d, n)
($E, segm, esq$) $\leftarrow$ FilaDeEventos(e, d, n)
Crie(T) $\triangleright$ cria a ABB vazia

Algoritmo de Shamos e Hoey

Interseção-SH(e, d, n)

$(E, segm, esq) \leftarrow$ FilaDeEventos(e, d, n)
Crie(T) $\triangleright$ cria a ABB vazia
para $p \leftarrow 1$ até $2n$ faça
  $i \leftarrow segm[p]$
  $pred \leftarrow$ Predecessor($T, E_X[p], E_Y[p]$)
  $suc \leftarrow$ Sucessor($T, E_X[p], E_Y[p]$)

Algoritmo de Shamos e Hoey

```
Interseção-SH(e, d, n)
  (E, segm, esq) ← FilaDeEventos(e, d, n)
  Crie(T)          ▷ cria a ABB vazia
  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
```

Algoritmo de Shamos e Hoey

Interseção-SH(e, d, n)

$(E, segm, esq) \leftarrow$ FilaDeEventos(e, d, n)
Crie(T) $\triangleright$ cria a ABB vazia
para $p \leftarrow 1$ até $2n$ faça
  $i \leftarrow segm[p]$
  $pred \leftarrow$ Predecessor$(T, E_X[p], E_Y[p])$
  $suc \leftarrow$ Sucessor$(T, E_X[p], E_Y[p])$
  se $esq[p]$
    então Insere(T, i)
      se $(pred \neq$ NIL e Inter$(e, d, i, pred))$
      ou $(suc \neq$ NIL e Inter$(e, d, i, suc))$
        então devolva verdade
    senão Remove(T, i)
      se $pred \neq$ NIL e $suc \neq$ 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 ABB, 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 $\mathrm{O}(n \lg n)$.

Interseção de segmentos

Problema: Dados n segmentos,
determinar se dois deles se intersectam.

Interseção de segmentos

Problema: Dados n segmentos,
determinar se dois deles se intersectam.

Até aqui: algoritmo $\mathrm{O}(n \lg n)$ para esse problema.

Interseção de segmentos

Problema: Dados n segmentos,
determinar se dois deles se intersectam.

Até aqui: algoritmo $\mathrm{O}(n \lg n)$ para esse problema.

Hipóteses simplificadoras:

Não há três extremos dos segmentos colineares.

Não há dois pontos extremos com mesma X-coordenada.

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

Interseção de segmentos

Problema: Dados n segmentos,
determinar se dois deles se intersectam.

Até aqui: algoritmo $\mathrm{O}(n \lg n)$ para esse problema.

Hipóteses simplificadoras:

Não há três extremos dos segmentos colineares.

Não há dois pontos extremos com mesma X-coordenada.

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

Como tratar destes casos degenerados?

Teste de interseção incluindo casos degenerados

Teste para posição geral corresponde à interseção própria.

Interseção de dois segmentos 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)

Teste de interseção incluindo casos degenerados

Teste para posição geral corresponde à interseção própria.

Interseção de dois segmentos 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)

Entre(a, b, c): verdade se são colineares e c está entre a e b.

Pré-processamento incluindo casos degenerados

Pontos extremos com mesma X-coordenada:

Pré-processamento incluindo casos degenerados

Pontos extremos com mesma X-coordenada:

Pré-processamento:
ordene os extremos dos segmentos por X-coordenada.

Se existir um segmento vertical,
considere o extremo inferior como esquerdo, colocando-o no vetor e, e o superior como direito, colocando-o no vetor d.

Pré-processamento incluindo casos degenerados

Pontos extremos com mesma X-coordenada:

Pré-processamento:
ordene os extremos dos segmentos por X-coordenada.

Se existir um segmento vertical,
considere o extremo inferior como esquerdo, colocando-o no vetor e, e o superior como direito, colocando-o no vetor d.

Se houver extremos repetidos de segmentos distintos, há interseção.

Pré-processamento incluindo casos degenerados

Pontos extremos com mesma X-coordenada:

Pré-processamento:
ordene os extremos dos segmentos por X-coordenada.

Se existir um segmento vertical,
considere o extremo inferior como esquerdo, colocando-o no vetor e, e o superior como direito, colocando-o no vetor d.

Se houver extremos repetidos de segmentos distintos, há interseção.

Extremos-Ordenados(n, S):
ordena os extremos dos n segmentos em S e já dá a resposta se houver repetição de extremos de segmentos distintos.

Detecção de interseção

Detecta-Interseção(n, S)

$E \leftarrow$ Extremos-Ordenados(n, S)
$T \leftarrow \emptyset$ $\qquad \triangleright$ ABBB ou treap
para cada $p \in E$ **faça**
    $s \leftarrow segmento(p)$
    $pred \leftarrow$ Predecessor(T, s) $\qquad suc \leftarrow$ Sucessor(T, s)
    se p é extremo esquerdo de s
        então Insere(T, s)
            se $(pred \neq \textsc{nil}$ **e** Intersecta$(s, pred))$
            ou $(suc \neq \textsc{nil}$ **e** Intersecta$(s, suc))$
                então devolva verdade
        senão Remove(T, s)
            se $pred$ **e** $suc \neq \textsc{nil}$ **e** Intersecta$(pred, suc)$
                então devolva verdade
devolva falso

Inserção em ABB

InsiraRec (T, x)

```
se T = NIL
    então devolva NovaCélula(x, NIL, NIL)
se x < info(T)          ▷ Vamos alterar aqui!
    então esq(T) ← InsiraRec(esq(T), x)
    senão dir(T) ← InsiraRec(dir(T), x)
devolva T
```

Inserção em ABB

InsiraRec (T, x)

se $T = \textsc{nil}$
  então devolva NovaCélula$(x, \textsc{nil}, \textsc{nil})$
se $x < info(T)$ $\triangleright$ Vamos alterar aqui!
  então $esq(T) \leftarrow$ InsiraRec$(esq(T), x)$
  senão $dir(T) \leftarrow$ InsiraRec$(dir(T), x)$
devolva T

InsiraRec (T, e, d, i)

se $T = \textsc{nil}$
  então devolva NovaCélula$(i, \textsc{nil}, \textsc{nil})$
se Esquerda$(e[segmento(T)], d[segmento(T)], e[i])$
  então $esq(T) \leftarrow$ InsiraRec$(esq(T), i)$
  senão $dir(T) \leftarrow$ InsiraRec$(dir(T), i)$
devolva T

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?

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?

Algoritmos sensíveis à saída (*output sensitive*).

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.

Como adaptar o algoritmo de Shamos e Hoey?

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.

Como adaptar o algoritmo de Shamos e Hoey?

Novo tipo de ponto evento: as interseções.

Como tratá-las?

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.

Como adaptar o algoritmo de Shamos e Hoey?

Novo tipo de ponto evento: as interseções.

Como tratá-las?

Ao detectar cada uma, além de imprimi-la,
a colocamos na fila de eventos (que é agora dinâmica).

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.

Como adaptar o algoritmo de Shamos e Hoey?

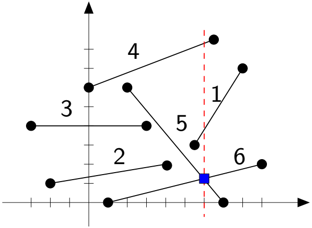
Novo tipo de ponto evento: as interseções.

Como tratá-las?

Ao detectar cada uma, além de imprimi-la,
a colocamos na fila de eventos (que é agora dinâmica).

Ao processar um ponto evento que é uma interseção, deve-se inverter a ordem dos segmentos que se intersectam neste ponto.

Ponto evento: interseção



Antes do ponto evento: $4 \prec 1 \prec 5 \prec 6$

Depois do ponto evento: $4 \prec 1 \prec 6 \prec 5$

Algoritmo de Bentley e Ottmann

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

Algoritmo de Bentley e Ottmann

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

Saída: todos os pares de segmentos da coleção que se intersectam.

Algoritmo de Bentley e Ottmann

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

Saída: todos os pares de segmentos da coleção que se intersectam.

Hipótese simplificadora:

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

Em particular, não há interseção com mesma X-coordenada
que outra, ou com algum extremo de segmento.

Algoritmo de Bentley e Ottmann

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

Saída: todos os pares de segmentos da coleção que se intersectam.

Hipótese simplificadora:

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

Em particular, não há interseção com mesma X-coordenada que outra, ou com algum extremo de segmento.

Não há interseções múltiplas, ou seja, não há um ponto em mais do que dois segmentos da coleção.

Fila de eventos

Agora ela é dinâmica: sofre inserções
(e, como antes, remoções).

Que ED usar para a fila de eventos?

Fila de eventos

Agora ela é dinâmica: sofre inserções
(e, como antes, remoções).

Que ED usar para a fila de eventos?

ABB com ordem dada pelas X-coordenadas dos pontos.

Fila de eventos

Agora ela é dinâmica: sofre inserções
(e, como antes, remoções).

Que ED usar para a fila de eventos?

ABB com ordem dada pelas X-coordenadas dos pontos.

A fila começa com os extremos dos intervalos.

A cada iteração, removemos um evento da fila para processá-lo.

Fila de eventos

Agora ela é dinâmica: sofre inserções
(e, como antes, remoções).

Que ED usar para a fila de eventos?

ABB com ordem dada pelas X-coordenadas dos pontos.

A fila começa com os extremos dos intervalos.

A cada iteração, removemos um evento da fila para processá-lo.

Ao detectar uma interseção,
inserimos tal ponto na fila de eventos.

Quantos elementos estão na fila no pior caso?

Versão simplificada

Hipótese simplificadora: não há pontos extremos repetidos, e as interseções são em pontos internos dos segmentos.

Versão simplificada

Hipótese simplificadora: não há pontos extremos repetidos, e as interseções são em pontos internos dos segmentos.

Extremos-Ordenados(n, S):
ordena os extremos dos n segmentos em S.

Versão simplificada

Hipótese simplificadora: não há pontos extremos repetidos, e as interseções são em pontos internos dos segmentos.

Extremos-Ordenados(n, S):
ordena os extremos dos n segmentos em S.

Acha-Interseções(n, S)
$Q \leftarrow$ Extremos(n, S) $\quad \triangleright$ inicializa a ABB Q com os extremos
$T \leftarrow \emptyset$
enquanto não Vazia(Q) **faça**
    $p \leftarrow$ Extrai-Min(Q)
    Trata-Evento(p)

Versão simplificada

Hipótese simplificadora: não há pontos extremos repetidos,
e as interseções são em pontos internos dos segmentos.

Extremos-Ordenados(n, S):
ordena os extremos dos n segmentos em S.

Acha-Interseções(n, S)
$Q \leftarrow$ Extremos(n, S) ▷ inicializa a ABB Q com os extremos
$T \leftarrow \emptyset$
enquanto não Vazia(Q) **faça**
    $p \leftarrow$ Extrai-Min(Q)
    Trata-Evento(p)

Notação: Para dois pontos eventos p e q,
escrevemos $p \prec q$ se $p_x < q_x$ ou $(p_x = q_x$ e $p_y < q_y)$

Versão simplificada

```
Trata-Evento(p)
se p é extremo esquerdo de um segmento s
   então Insere(T, s)
         pred ← Predecessor(T, s)
         suc ← Sucessor(T, s)
         se pred ≠ NIL e Intersecta(s, pred)
            então Verifica-Novo-Evento(p, Q, s, pred)
         se suc ≠ NIL e Intersecta(s, suc)
            então Verifica-Novo-Evento(p, Q, s, suc)
```

Versão simplificada

```
Trata-Evento(p)
se p é extremo esquerdo de um segmento s
    então Insere(T, s)
            pred ← Predecessor(T, s)
            suc ← Sucessor(T, s)
            se pred ≠ NIL e Intersecta(s, pred)
                então Verifica-Novo-Evento(p, Q, s, pred)
            se suc ≠ NIL e Intersecta(s, suc)
                então Verifica-Novo-Evento(p, Q, s, suc)
```

```
Verifica-Novo-Evento(p, Q, s1, s2)
q ← Ponto-de-Interseção(s1, s2)
se q ≻ p e não Pertence(Q, q)
    então Insere(Q, q)
            imprima q
```

Versão simplificada

```
Trata-Evento(p)
se p é extremo esquerdo de um segmento s
   então Insere(T, s)
         pred ← Predecessor(T, s)
         suc ← Sucessor(T, s)
         se pred ≠ NIL e Intersecta(s, pred)
            então Verifica-Novo-Evento(p, Q, s, pred)
         se suc ≠ NIL e Intersecta(s, suc)
            então Verifica-Novo-Evento(p, Q, s, suc)
se p é extremo direito de um segmento s
   então Remove(T, s)
         pred ← Predecessor(T, s)
         suc ← Sucessor(T, s)
         se pred e suc ≠ NIL e Intersecta(pred, suc)
            então Verifica-Novo-Evento(p, Q, suc, pred)
```

Versão simplificada

```
Trata-Evento(p)
...
se p é ponto de interseção
   então sejam s e s′ os segmentos em T que contém p
         pred ← Predecessor(T, s)
         suc ← Sucessor(T, s′)
         Remove(T, s)   Remove(T, s′)
         ▷ insere s e s′ na ordem inversa
         Insere(T, s′)   Insere(T, s)
         se pred ≠ NIL e Intersecta(pred, s′)
            então Verifica-Novo-Evento(p, Q, pred, s′)
         se suc ≠ NIL e Intersecta(s, suc)
            então Verifica-Novo-Evento(p, Q, s, suc)
```

Consumo de tempo

Seja i o número de interseções.

O algoritmo executa $2n + i$ iterações.

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

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

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

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

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

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

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

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

Cada iteração faz uma chamada a Extrai-Min e, eventualmente, uma a Insere na ABBB Q.

Na ABBB Q, em qq momento, há $\mathrm{O}(n + i) = \mathrm{O}(n^2)$ pontos.

Assim, cada operação consome tempo $\mathrm{O}(\lg n^2) = \mathrm{O}(\lg n)$.

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

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

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

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

Cada iteração faz uma chamada a Extrai-Min e, eventualmente, uma a Insere na ABBB Q.

Na ABBB Q, em qq momento, há $\mathrm{O}(n + i) = \mathrm{O}(n^2)$ pontos.

Assim, cada operação consome tempo $\mathrm{O}(\lg n^2) = \mathrm{O}(\lg n)$.

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

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

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

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

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

Cada iteração faz uma chamada a Extrai-Min e, eventualmente, uma a Insere na ABBB Q.

Na ABBB Q, em qq momento, há $\mathrm{O}(n + i) = \mathrm{O}(n^2)$ pontos.

Assim, cada operação consome tempo $\mathrm{O}(\lg n^2) = \mathrm{O}(\lg n)$.

O consumo de tempo por iteração é $\mathrm{O}(\lg n)$, e
o algoritmo de Bentley e Ottmann consome tempo $\mathrm{O}((n + i) \lg n)$.

Versão completa

O que fazer com os casos que excluímos?

Versão completa

O que fazer com os casos que excluímos?

Alterações:

- Q conterá os pontos eventos, sem repetições.
- Ponto evento extremo: tem a lista dos segmentos que têm esse ponto como extremo.
- impressão apenas no momento do processamento do ponto.

Versão completa

O que fazer com os casos que excluímos?

Alterações:

- Q conterá os pontos eventos, sem repetições.
- Ponto evento extremo: tem a lista dos segmentos que têm esse ponto como extremo.
- impressão apenas no momento do processamento do ponto.

Ao processar um ponto evento,
determinam-se todos os segmentos que o contém
(pela lista do ponto e/ou pelos segmentos em T).

Versão completa

O que fazer com os casos que excluímos?

Alterações:

- Q conterá os pontos eventos, sem repetições.
- Ponto evento extremo: tem a lista dos segmentos que têm esse ponto como extremo.
- impressão apenas no momento do processamento do ponto.

Ao processar um ponto evento,
determinam-se todos os segmentos que o contém
(pela lista do ponto e/ou pelos segmentos em T).

Se mais de um segmento o contém, imprimimos o ponto.

Versão completa

O que fazer com os casos que excluímos?

Alterações:

- Q conterá os pontos eventos, sem repetições.
- Ponto evento extremo: tem a lista dos segmentos que têm esse ponto como extremo.
- impressão apenas no momento do processamento do ponto.

Ao processar um ponto evento,
determinam-se todos os segmentos que o contém
(pela lista do ponto e/ou pelos segmentos em T).

Se mais de um segmento o contém, imprimimos o ponto.

Atualiza-se T.

Atualização de T

Se o ponto evento é um extremo, faz-se como antes:

Atualização de T

Se o ponto evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Atualização de T

Se o ponto evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Primeiro trata-se de extremos esquerdos.

Atualização de T

Se o ponto evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Primeiro trata-se de extremos esquerdos.

Se o ponto evento é uma interseção

Atualização de T

Se o ponto evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Primeiro trata-se de extremos esquerdos.

Se o ponto evento é uma interseção

- remove-se de T todos os segmentos que o contém no interior.
- estes são incluídos novamente na ordem inversa.

Atualização de T

Se o ponto evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Primeiro trata-se de extremos esquerdos.

Se o ponto evento é uma interseção

- remove-se de T todos os segmentos que o contém no interior.
- estes são incluídos novamente na ordem inversa.

Os dois casos podem acontecer ao mesmo tempo...
Isso está detalhado no livro de de Berg e outros, capítulo 2.

Resumo do método da linha de varredura

Pensar na varredura:
horizontal, vertical, em diagonal, angular?

Resumo do método da linha de varredura

Pensar na varredura:
horizontal, vertical, em diagonal, angular?

Decidir quem são os eventos, onde a linha muda:
é um conjunto pré-determinado ou dinâmico?

Resumo do método da linha de varredura

Pensar na varredura:
horizontal, vertical, em diagonal, angular?

Decidir quem são os eventos, onde a linha muda:
é um conjunto pré-determinado ou dinâmico?

O que é guardado na ED da linha de varredura?
Em que ordem?

Resumo do método da linha de varredura

Pensar na varredura:
horizontal, vertical, em diagonal, angular?

Decidir quem são os eventos, onde a linha muda:
é um conjunto pré-determinado ou dinâmico?

O que é guardado na ED da linha de varredura?
Em que ordem?

Como tratar cada evento?

Perguntas???

Perguntas???

Agora vamos ver as animações?

Perguntas???

Agora vamos ver as animações?

Obrigada!!!!

Referências

Se você quiser aprender mais sobre geometria computacional, sugiro os seguintes textos:

- C.G. Fernandes e J.C. de Pina, *Um convite à Geometria Computacional*, `https://www.ime.usp.br/~cris/jai2009/`
  Texto preparado para acompanhar um minicurso ministrado nas Jornadas de Atualização em Informática, em 2009.

- M. de Berg, M. van Kreveld, M. Overmars, e O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer, Segunda Edição, 2000.
  Livro sensacional, que cobre uma ampla variedade de tópicos em geometria computacional.

Comentários extras sobre dois problemas do contest

Enclosure:

Calcule o fecho convexo de todas as árvores: os vértices deste fecho, na ordem em que aparecem, são os candidatos a serem a árvore que você está procurando. Isso, em particular, já vai lhe dar os pontos a serem testados na ordem que queríamos.

November Rain:

Primeiro, com o método da linha de varredura, determine quais telhados recebem chuva diretamente do céu, quanto cada um recebe assim, e em qual telhado ele despeja sua água. Inicialmente cada segmento recebeu a água que caiu nele diretamente do céu. Segundo, percorra as pontas mais baixas dos segmentos de cima para baixo. Ao passar pela ponta de baixo de cada segmento, a quantidade de água que caiu sobre ele estará calculada, e você deve somar esta quantidade de água à quantidade de água do segmento onde este despeja sua água, que está mais para baixo deste.