Geometria Computacional

Cristina G. Fernandes

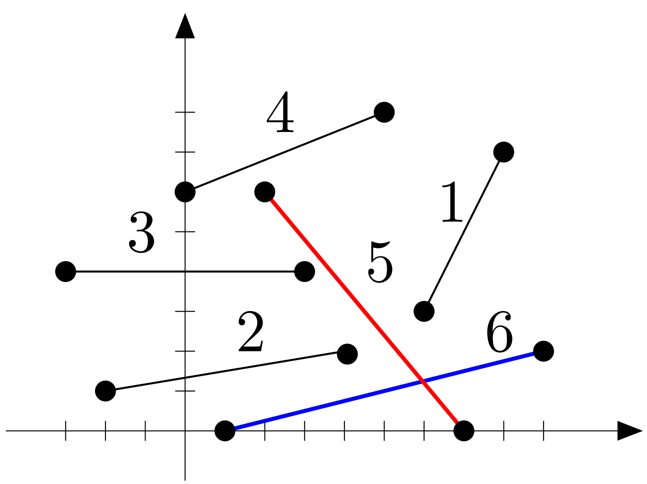
Departamento de Ciência da Computação do IME-USP

`http://www.ime.usp.br/~cris/`

segundo semestre de 2009

Detecção de interseção

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.

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.

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.

Muda apenas em posições chaves: os pontos eventos.

Tratamento dos pontos eventos

Pontos eventos são mantidos em uma fila.

- Atualizar a fila:
  remover o ponto evento corrente, junto com alguns outros que tenham se tornado obsoletos e, eventualmente, inserir novos pontos eventos na fila;

Tratamento dos pontos eventos

Pontos eventos são mantidos em uma fila.

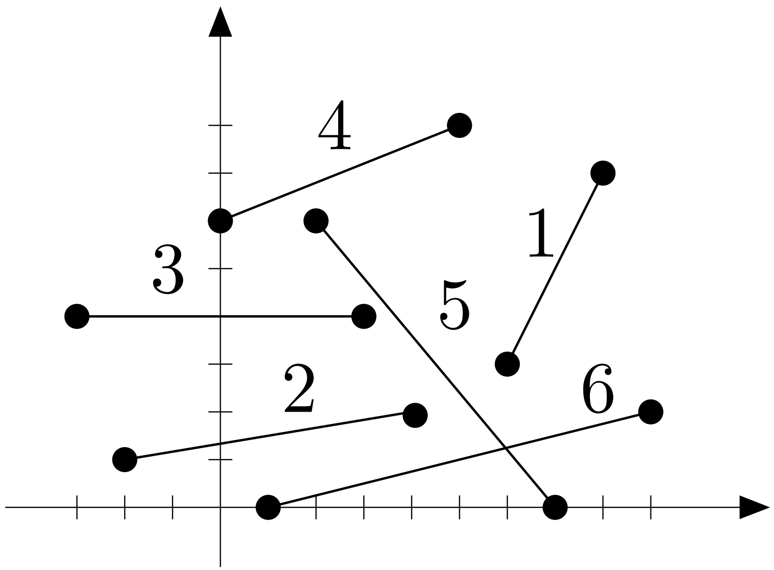
- Atualizar a fila:
  remover o ponto evento corrente, junto com alguns outros que tenham se tornado obsoletos e, eventualmente, inserir novos pontos eventos na fila;
- Atualizar a linha:
  atualizar a descrição combinatória da linha de varredura para que esta represente a situação atual;

Tratamento dos pontos eventos

Pontos eventos são mantidos em uma fila.

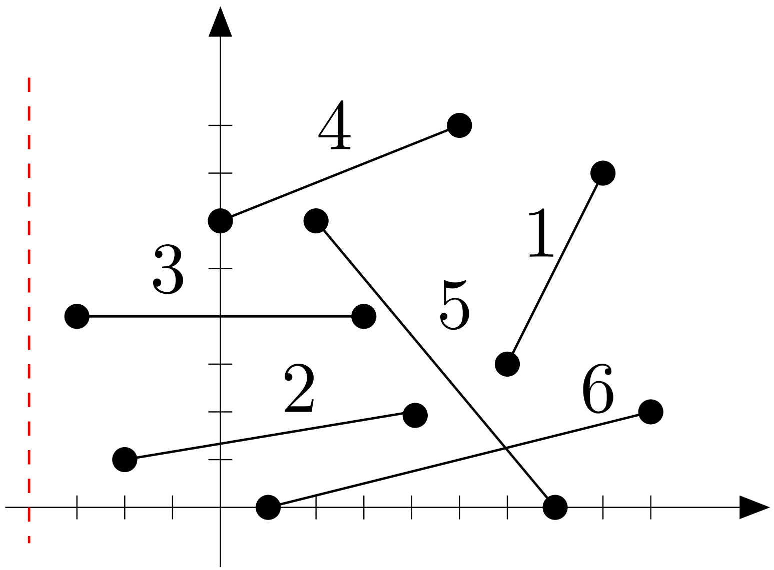
- Atualizar a fila:
  remover o ponto evento corrente, junto com alguns outros que tenham se tornado obsoletos e, eventualmente, inserir novos pontos eventos na fila;
- Atualizar a linha:
  atualizar a descrição combinatória da linha de varredura para que esta represente a situação atual;
- Resolver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

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	

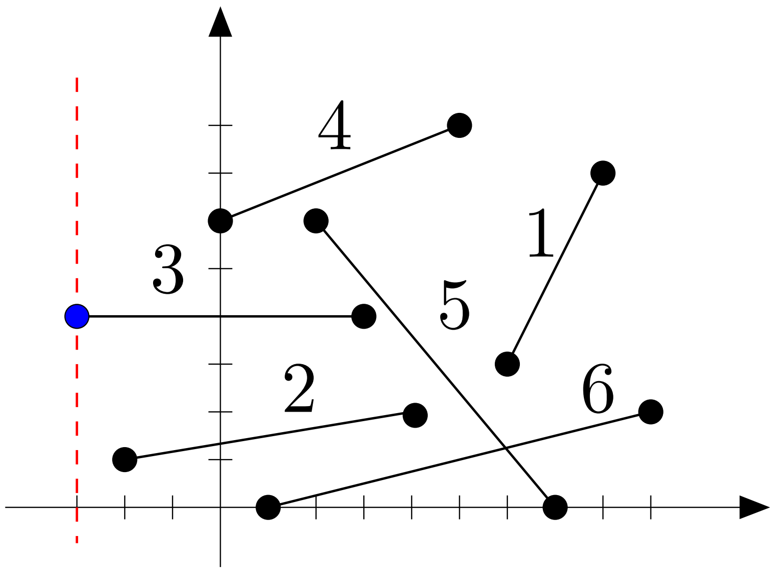
Algoritmo de Shamos e Hoey



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

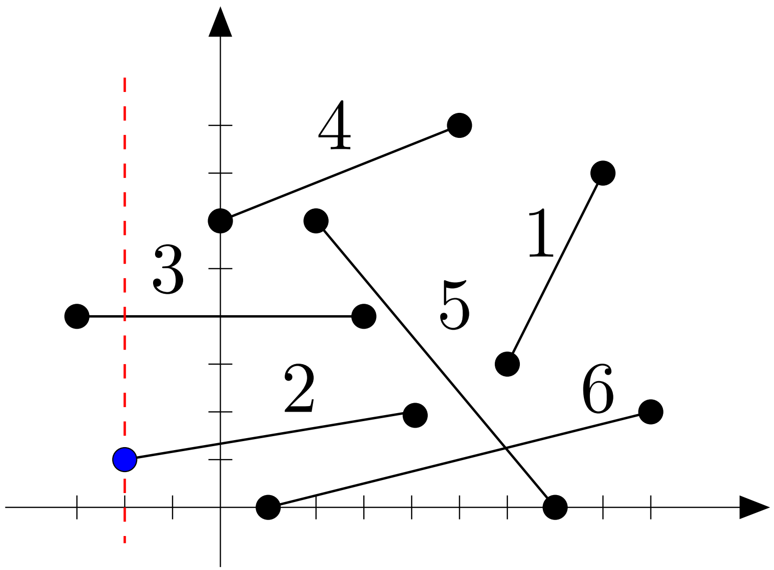


$-\infty$	
-3	3
-2	$3 \prec 2$
0	$4 \prec 3 \prec 2$
1	$4 \prec 3 \prec 2 \prec 6$
2	$4 \prec 5 \prec 3 \prec 2 \prec 6$
3	$4 \prec 5 \prec 2 \prec 6$
4	$4 \prec 5 \prec 6$
5	
6	
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

Algoritmo de Shamos e Hoey

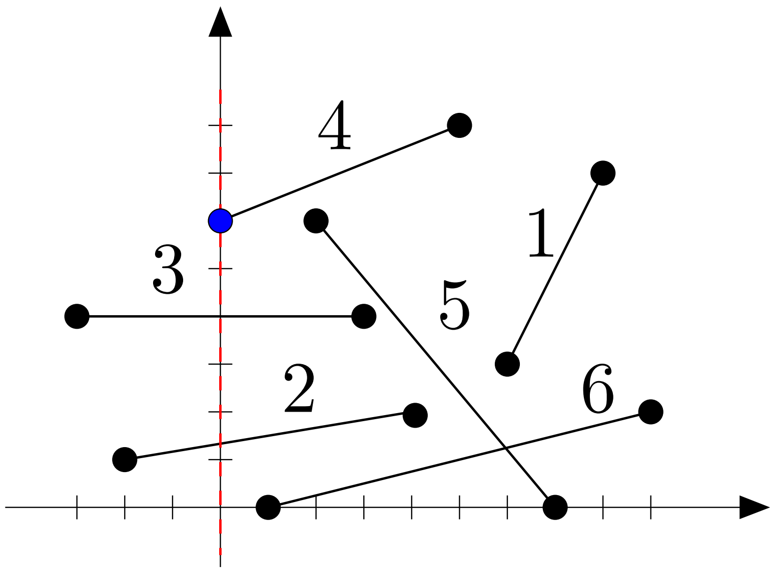


Alterações ocorrem
nos extremos dos segmentos.

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	$5 \prec 6$
6	$\cdots$
7	
8	

Algoritmo de Shamos e Hoey

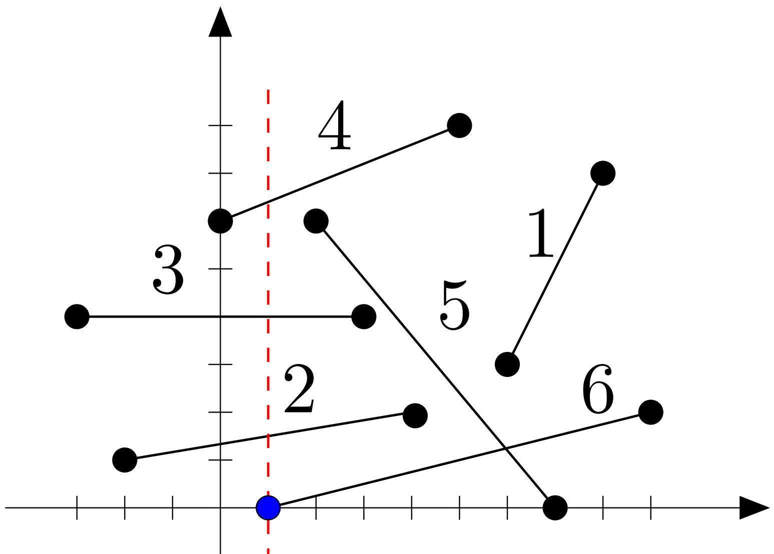


$-\infty$	
-3	3
-2	$3 \prec 2$
0	$4 \prec 3 \prec 2$
1	$4 \prec 3 \prec 2 \prec 6$
2	$4 \prec 5 \prec 3 \prec 2 \prec 6$
3	$4 \prec 5 \prec 2 \prec 6$
4	$4 \prec 5 \prec 6$
5	
6	
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

Algoritmo de Shamos e Hoey

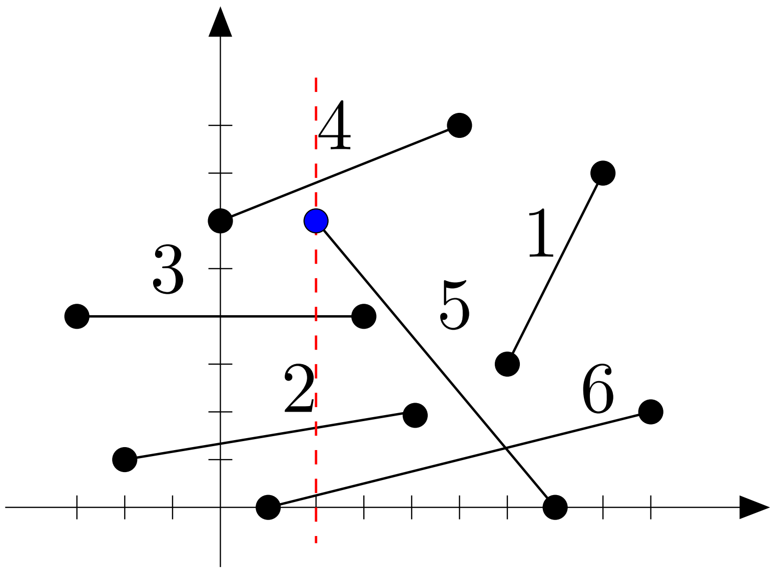


$-\infty$	
-3	3
-2	$3 \prec 2$
0	$4 \prec 3 \prec 2$
1	$4 \prec 3 \prec 2 \prec 6$
2	$4 \prec 5 \prec 3 \prec 2 \prec 6$
3	$4 \prec 5 \prec 2 \prec 6$
4	$4 \prec 5 \prec 6$
5	
6	
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

Algoritmo de Shamos e Hoey

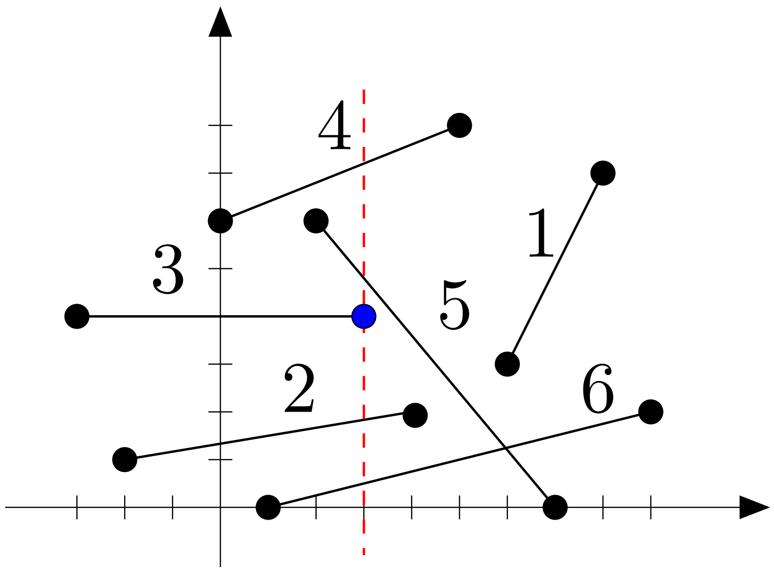


$-\infty$	
-3	3
-2	$3 \prec 2$
0	$4 \prec 3 \prec 2$
1	$4 \prec 3 \prec 2 \prec 6$
2	$4 \prec 5 \prec 3 \prec 2 \prec 6$
3	$4 \prec 5 \prec 2 \prec 6$
4	$4 \prec 5 \prec 6$
5	
6	
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

Algoritmo de Shamos e Hoey



$-\infty$

$-3 \qquad 3$

$-2 \qquad 3 \prec 2$

$0 \qquad 4 \prec 3 \prec 2$

$1 \qquad 4 \prec 3 \prec 2 \prec 6$

$2 \qquad 4 \prec 5 \prec 3 \prec 2 \prec 6$

$3 \qquad 4 \prec 5 \prec 2 \prec 6$

$4 \qquad 4 \prec 5 \prec 6$

5

6

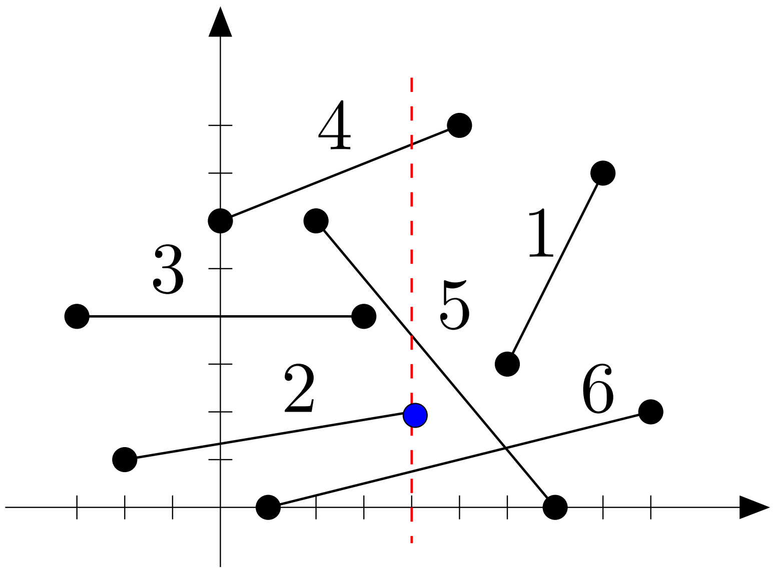
7

8

Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

Algoritmo de Shamos e Hoey

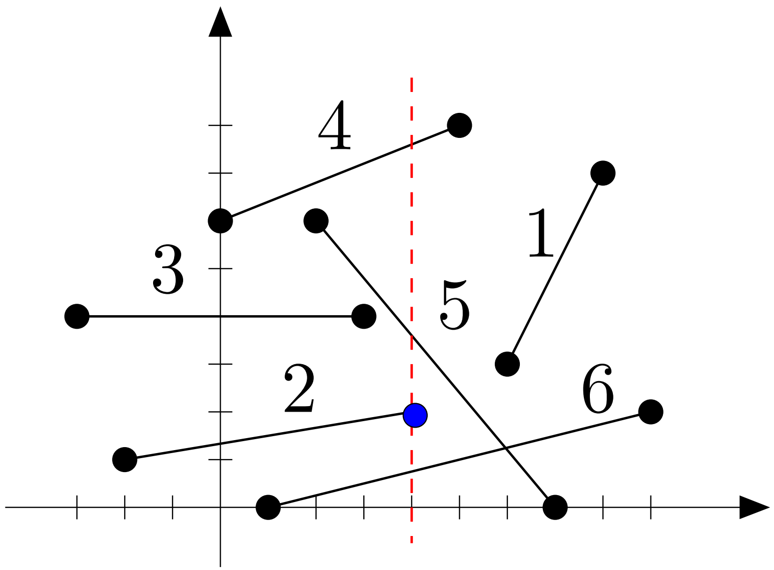


$-\infty$	
-3	3
-2	$3 \prec 2$
0	$4 \prec 3 \prec 2$
1	$4 \prec 3 \prec 2 \prec 6$
2	$4 \prec 5 \prec 3 \prec 2 \prec 6$
3	$4 \prec 5 \prec 2 \prec 6$
4	$4 \prec 5 \prec 6$
5	
6	
7	
8	

Alterações ocorrem
nos extremos dos segmentos.

Estes são
os pontos eventos.

Algoritmo de Shamos e Hoey



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

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 de busca binária balanceada (ABBB)
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 de busca binária balanceada (ABBB)
ou uma skip lists.

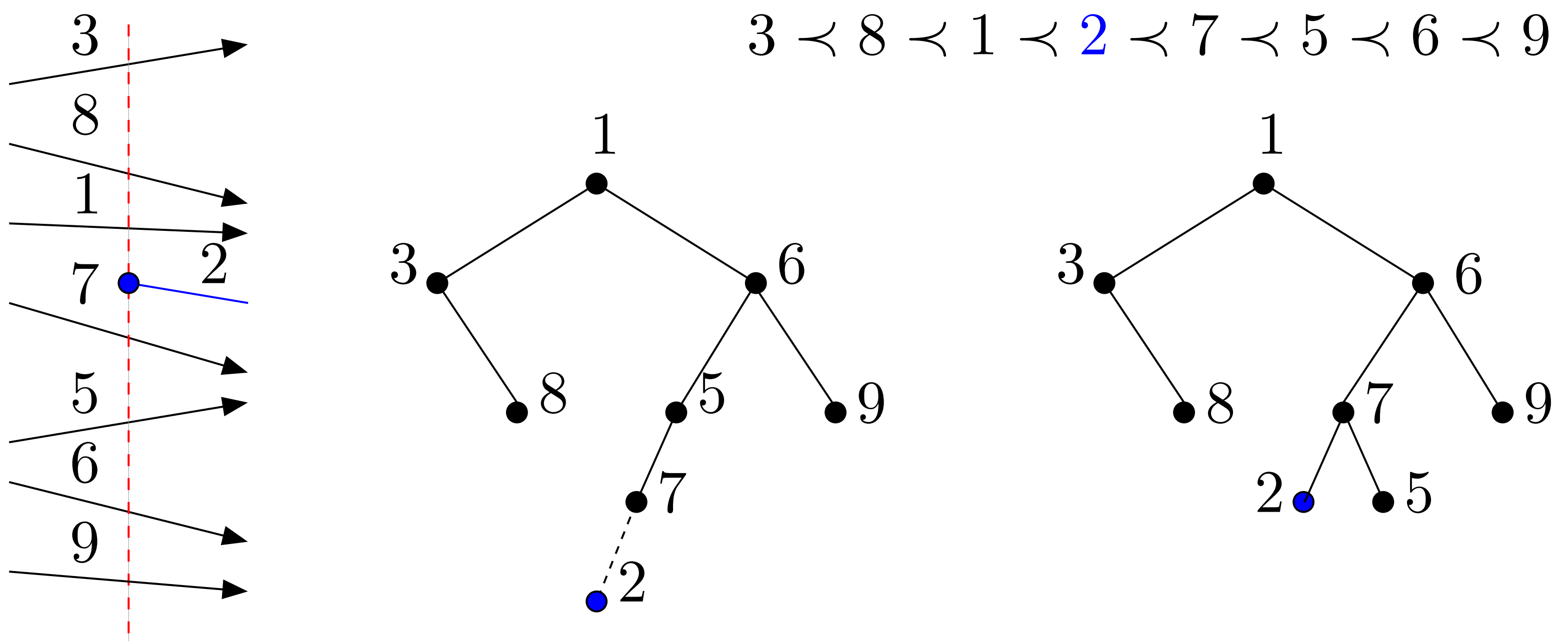
Numa ABBB,
custo de pior caso por operação é $\mathrm{O}(\lg m)$,
onde m é o número de elementos armazenados.

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

Descrição combinatória da linha

Os segmentos ficam na ordem em que intersectam a linha.

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



Exemplo de inserção usando uma ABBB
para a descrição combinatória da linha.

Ordem usada

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

Ordem usada

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

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

Ordem usada

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

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

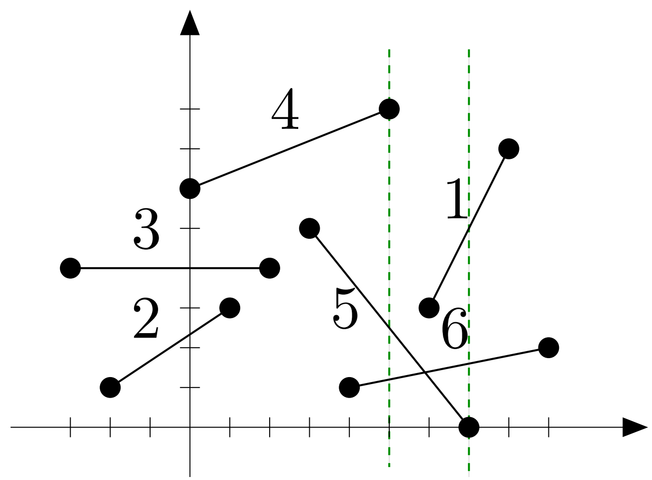
$i \prec_t j$ se a interseção da linha com i
fica acima da interseção com j.

Ordem usada

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

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

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



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

Árvore de busca binária balanceada

Rotinas de manipulação de uma ABBB:

CRIE(T): cria uma ABBB T vazia;

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

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

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

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

Árvore de busca binária balanceada

Rotinas de manipulação de uma ABBB:

- **CRIE**(T): cria uma ABBB T vazia;
- **INSIRA**(T, i): insere i na ABBB T, usando $\prec_t$ com $t = e_X[i]$;
- **REMOVA**(T, i): remove i de T, usando $\prec_t$ com $t = d_X[i]$;
- **PREDECESSOR**(T, x, y): devolve o predecessor em T de um segmento que passa pelo ponto (x, y), usando $\prec_x$;
- **SUCESSOR**(T, x, y): devolve o sucessor em T de um segmento que passa pelo ponto (x, y), usando $\prec_x$.

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

Árvore de busca binária balanceada

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

Árvore de busca binária balanceada

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

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

Árvore de busca binária balanceada

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

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

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

Árvore de busca binária balanceada

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

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

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

A inserção começa com uma busca a partir de T.

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

Árvore de busca binária balanceada

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

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

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

A inserção começa com uma busca a partir de T.

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

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

Inserção em ABB rubro-negra

Esta é a inserção para árvores 2-3-4:

```
INSIRAREC (T, x)
se T = NIL
   então q ← NOVACÉLULA(x, NIL, NIL, RUBRO)
         devolva q
se RUBRO(esq(T)) e RUBRO(esq(T))
   então TROQUECORES(T)
se x < info(T)
   então esq(T) ← INSIRAREC(esq(T), x)
   senão dir(T) ← INSIRAREC(dir(T), x)
se RUBRO(dir(T)) e NEGRO(esq(T))
   então T ← ROTACIONEESQ(T)
se RUBRO(esq(T)) e RUBRO(esq(esq(T)))
   então T ← ROTACIONEDIR(T)
devolva T
```

Algoritmo de Shamos e Hoey

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

Algoritmo de Shamos e Hoey

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

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

Algoritmo de Shamos e Hoey

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

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

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 $e[1 \mathinner{..} n]$ e $d[1 \mathinner{..} n]$ com extremos dos segmentos

Montagem da fila de eventos

FILADEEVENTOS:
recebe $e[1\,.\,.\,n]$ e $d[1\,.\,.\,n]$ com extremos dos segmentos

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

Montagem da fila de eventos

FILADEEVENTOS:
recebe $e[1\,.\,.\,n]$ e $d[1\,.\,.\,n]$ com extremos dos segmentos

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

devolve

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

Montagem da fila de eventos

FILADEEVENTOS:

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

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

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

devolve

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

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

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

Montagem da fila de eventos

FILADEEVENTOS:
recebe $e[1\,..\,n]$ e $d[1\,..\,n]$ com extremos dos segmentos

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

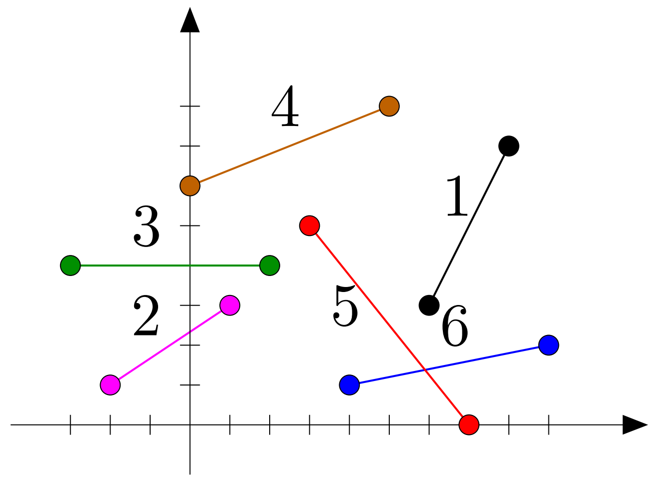
devolve

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

$segm[1\,..\,2n]$:
$segm[p]$: índice do segmento do qual $E[p]$ é extremo

$esq[1\,..\,2n]$:
$esq[p]$: VERDADE se $E[p]$ é extremo esquerdo de $segm[p]$
esq FALSO caso contrário.

Fila de eventos



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

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

Processamento de ponto evento

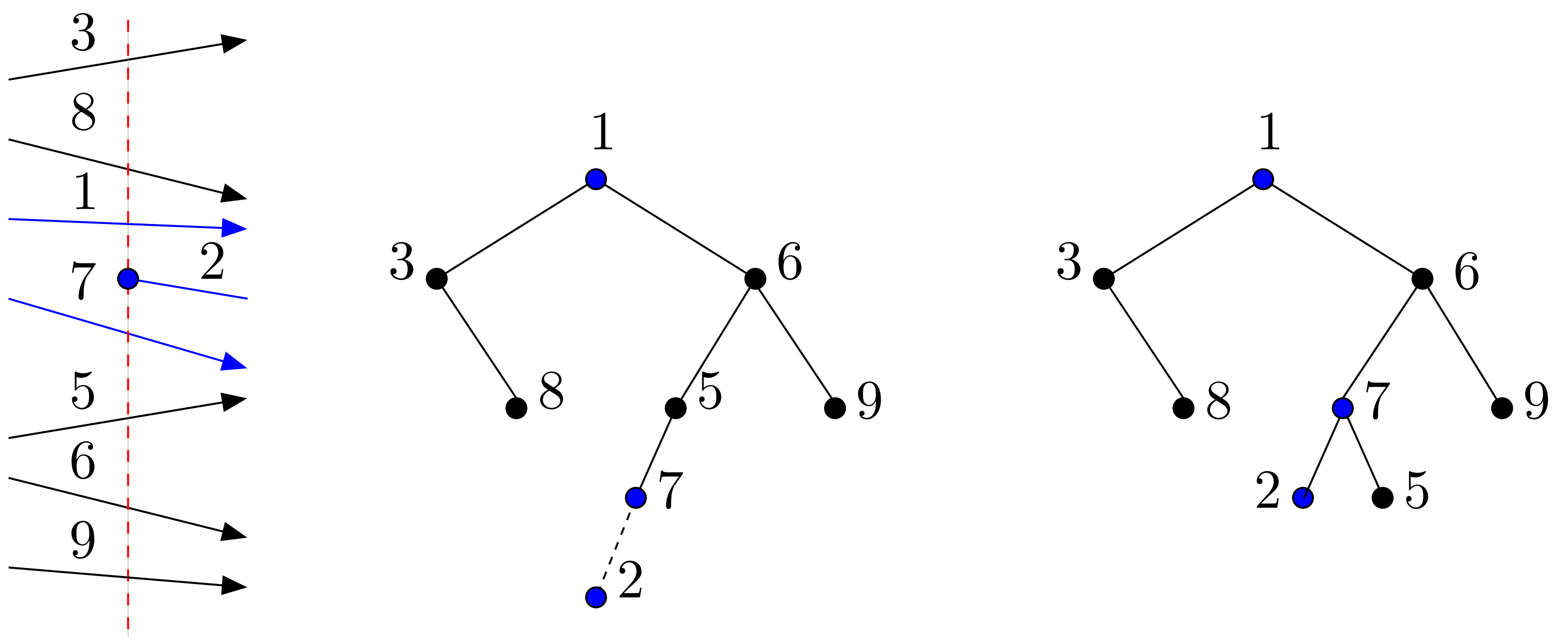
Dois tipos:

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

Processamento de ponto evento

Dois tipos:

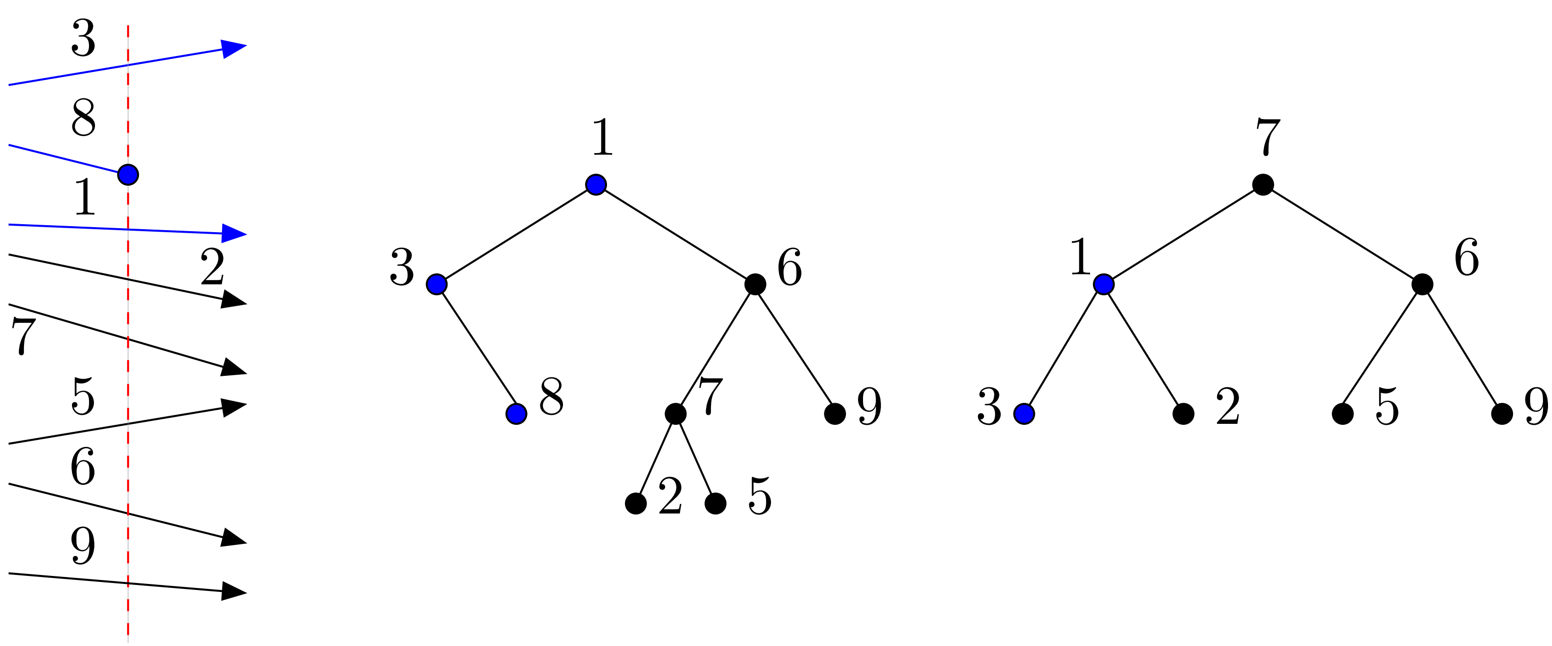
- começo de segmento: inclui o novo segmento na ABBB e verifica interseção com seus dois novos “vizinhos”.



Processamento de ponto evento

Dois tipos:

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



Processamento de ponto evento

Dois tipos:

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

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

Processamento de ponto evento

Dois tipos:

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

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

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

Algoritmo de Shamos e Hoey

INTERSEÇÃO-SH(e, d, n)

$(E, segm, esq) \leftarrow$ FILADEEVENTOS(e, d, n)
CRIE(T)
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 INSIRA(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 REMOVA(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 INSIRA ou a REMOVA.

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

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

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

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

Todas as interseções de segmentos

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

Todas as interseções de segmentos

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

Você consegue projetar um algoritmo
que consuma tempo $\mathrm{O}(n \lg n)$ 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?

Na aula que vem...

- algoritmos sensíveis à saída (*output sensitive*)
- algoritmo de Bentley e Ottmann
- fila de eventos dinâmica: uma interseção é um ponto evento também!
- como tratar esse tipo de ponto evento?
- o que muda na EC da linha?
- como tratar os casos especiais excluídos por nossas hipóteses simplificadoras?
- uma animação!
- projetos, algumas possibilidades