# Estruturas de Dados e Algoritmos: Viagem no Tempo e no Espaço

Cristina G. Fernandes
Universidade de São Paulo

Seminários de Pós-Graduação em Computação da UFABC

17 de julho de 2019

# Estruturas de dados e algoritmos

Tópicos que abordaremos:

- Viagem no tempo: estruturas de dados temporais
- Viagem no espaço: linha de varredura em geometria computacional

# Estruturas de dados temporais

Em EDs usuais, não temos acesso a versões anteriores da ED:

# Estruturas de dados temporais

Em EDs usuais, não temos acesso a versões anteriores da ED:

Ao fazer uma atualização,
o estado anterior da ED em geral se perde.

# Estruturas de dados temporais

Em EDs usuais, não temos acesso a versões anteriores da ED:

Ao fazer uma atualização,
o estado anterior da ED em geral se perde.

EDs temporais:

Dão acesso a todas as versões anteriores da ED.

# Estruturas de dados temporais

Em EDs usuais, não temos acesso a versões anteriores da ED:

Ao fazer uma atualização,
o estado anterior da ED em geral se perde.

EDs temporais:

Dão acesso a todas as versões anteriores da ED.

## Dois tipos

- EDs persistentes

# Estruturas de dados temporais

Em EDs usuais, não temos acesso a versões anteriores da ED:

Ao fazer uma atualização,
o estado anterior da ED em geral se perde.

EDs temporais:

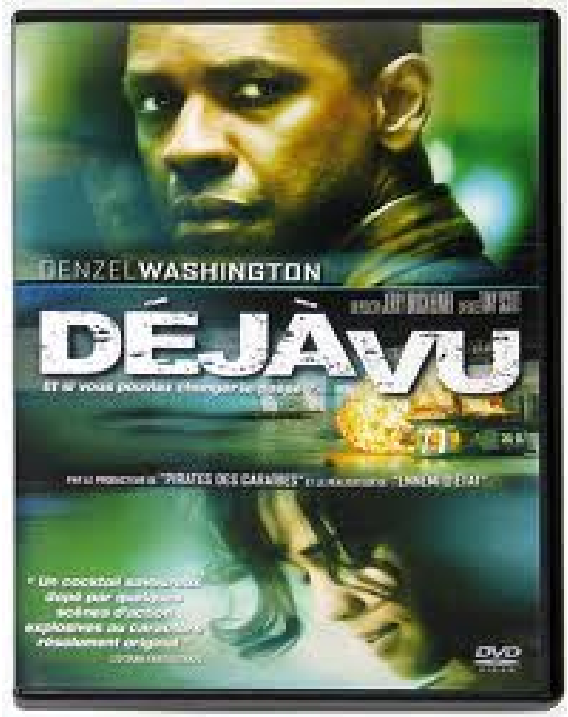
Dão acesso a todas as versões anteriores da ED.

**Dois tipos**

- EDs persistentes
- EDs retroativas

# Persistência

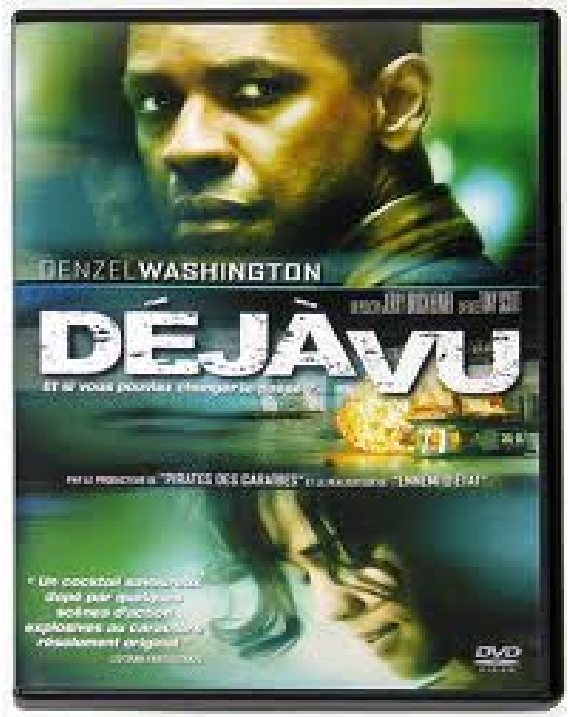
Como no filme **Déjà Vu**:
linhas do tempo paralelas.



# Persistência

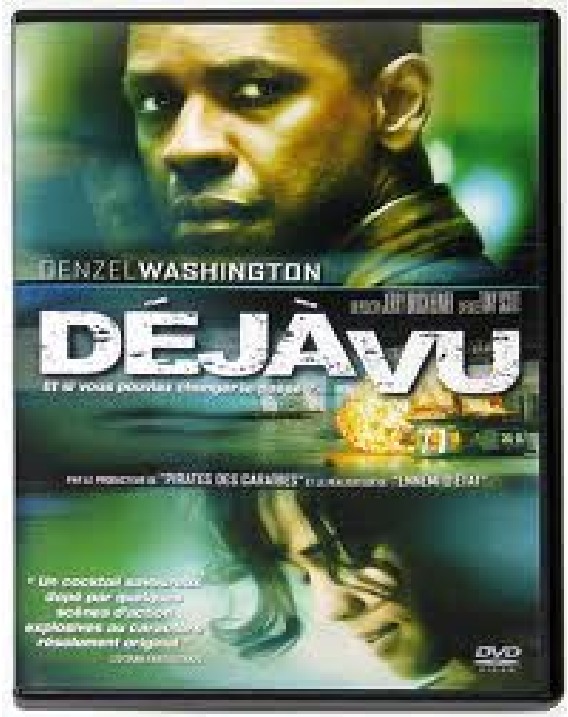
Como no filme **Déjà Vu**:
linhas do tempo paralelas.

Alterações no passado causam
aparecimento de nova cópia da ED
a partir daquele ponto.



# Persistência

Como no filme **Déjà Vu**:
linhas do tempo paralelas.



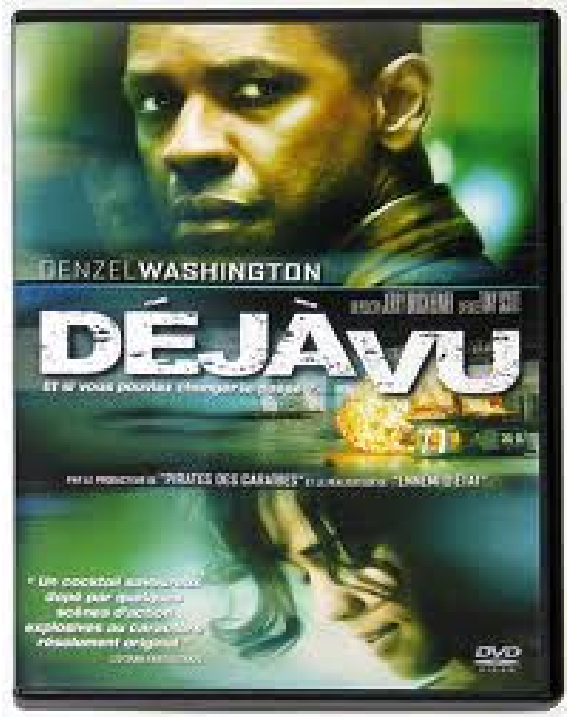

Alterações no passado causam
aparecimento de nova cópia da ED
a partir daquele ponto.

Persistência parcial:

- somente consultas nas versões passadas
- atualizações apenas na versão corrente

# Persistência

Como no filme **Déjà Vu**:
linhas do tempo paralelas.




Alterações no passado causam
aparecimento de nova cópia da ED
a partir daquele ponto.

Persistência parcial:

- somente consultas nas versões passadas
- atualizações apenas na versão corrente
- linha de tempo única

# Exemplo: pilha persistente

## Operações

- push (empilhe)
- pop (desempilhe)
- top (topo) - consulta

# Exemplo: pilha persistente

## Operações

- push (empilhe)
- pop (desempilhe)
- top (topo) - consulta

Todas custam tempo constante na implementação tradicional.

# Exemplo: pilha persistente

## Operações

- push (empilhe)
- pop (desempilhe)
- top (topo) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar (eficientemente) uma versão persistente?

# Exemplo: pilha persistente

## Operações

- push (empilhe)
- pop (desempilhe)
- top (topo) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar (eficientemente) uma versão persistente?

- cada modificação (push/pop) gera uma nova versão da pilha

# Exemplo: pilha persistente

## Operações

- push (empilhe)
- pop (desempilhe)
- top (topo) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar (eficientemente) uma versão persistente?

- cada modificação (push/pop) gera uma nova versão da pilha
- implementação usando lista ligada

# Exemplo: pilha persistente

## Operações

- push (empilhe)
- pop (desempilhe)
- top (topo) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar (eficientemente) uma versão persistente?

- cada modificação (push/pop) gera uma nova versão da pilha
- implementação usando lista ligada
- $p_i$: indica o topo da $i$-ésima versão da pilha

# Exemplo: pilha persistente

## Operações

- push (empilhe)
- pop (desempilhe)
- top (topo) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar (eficientemente) uma versão persistente?

- cada modificação (push/pop) gera uma nova versão da pilha
- implementação usando lista ligada
- $p_i$: indica o topo da $i$-ésima versão da pilha

Resultado: cada operação vai custar tempo constante

# Exemplo: pilha persistente

## Operações

- push (empilhe)
- pop (desempilhe)
- top (topo) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar (eficientemente) uma versão persistente?

- cada modificação (push/pop) gera uma nova versão da pilha
- implementação usando lista ligada
- $p_i$: indica o topo da $i$-ésima versão da pilha

Resultado: cada operação vai custar tempo constante e
o espaço é proporcional ao número de operações de modificação.

# Exemplo: pilha persistente

**Operações:**

$p_0 = \text{Stack}()$

**Pilhas:**

# Exemplo: pilha persistente

**Operações:**

$p_0 = \text{Stack}()$

**Pilhas:**

$p_0$ :

# Exemplo: pilha persistente

**Operações:**

$p_0 = \text{Stack}()$

$p_1 = \text{Push}(p_0, 5)$

**Pilhas:**

$p_0$ :

# Exemplo: pilha persistente

**Operações:**

$p_0 = \text{Stack}()$

$p_1 = \text{Push}(p_0, 5)$

**Pilhas:**

$p_0$ :

$p_1 : 5$

# Exemplo: pilha persistente

**Operações:**

$p_0 = \text{Stack}()$

$p_1 = \text{Push}(p_0, 5)$

$p_2 = \text{Push}(p_1, 7)$

**Pilhas:**

$p_0$ :

$p_1$ : 5

# Exemplo: pilha persistente

**Operações:**

$p_0 = \text{Stack}()$

$p_1 = \text{Push}(p_0, 5)$

$p_2 = \text{Push}(p_1, 7)$

**Pilhas:**

$p_0$ :

$p_1$ : 5

$p_2$ : 5 7

# Exemplo: pilha persistente

**Operações:**

$p_0 = \text{Stack}()$

$p_1 = \text{Push}(p_0, 5)$

$p_2 = \text{Push}(p_1, 7)$

$\text{Top}(p_2)$

**Pilhas:**

$p_0$ :

$p_1$ : 5

$p_2$ : 5 7

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |
| $\text{Top}(p_3)$ | |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |
| $\text{Top}(p_3)$ | **6** |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |
| $\text{Top}(p_3)$ | **6** |
| $p_5 = \text{Push}(p_4, 9)$ | |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |
| $\text{Top}(p_3)$ | **6** |
| $p_5 = \text{Push}(p_4, 9)$ | $p_5$ : 5 9 |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |
| $\text{Top}(p_3)$ | **6** |
| $p_5 = \text{Push}(p_4, 9)$ | $p_5$ : 5 9 |
| $\text{Top}(p_4)$ | |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |
| $\text{Top}(p_3)$ | **6** |
| $p_5 = \text{Push}(p_4, 9)$ | $p_5$ : 5 9 |
| $\text{Top}(p_4)$ | **5** |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |
| $\text{Top}(p_3)$ | **6** |
| $p_5 = \text{Push}(p_4, 9)$ | $p_5$ : 5 9 |
| $\text{Top}(p_4)$ | **5** |
| $p_6 = \text{Push}(p_0, 5)$ | |

# Exemplo: pilha persistente

| **Operações:** | **Pilhas:** |
|---|---|
| $p_0 = \text{Stack}()$ | $p_0$ : |
| $p_1 = \text{Push}(p_0, 5)$ | $p_1$ : 5 |
| $p_2 = \text{Push}(p_1, 7)$ | $p_2$ : 5 7 |
| $\text{Top}(p_2)$ | **7** |
| $p_3 = \text{Push}(p_2, 6)$ | $p_3$ : 5 7 6 |
| $p_4 = \text{Pop}(p_2)$ | $p_4$ : 5 |
| $\text{Top}(p_3)$ | **6** |
| $p_5 = \text{Push}(p_4, 9)$ | $p_5$ : 5 9 |
| $\text{Top}(p_4)$ | **5** |
| $p_6 = \text{Push}(p_0, 5)$ | $p_6$ : 5 |

# Exemplo: pilha persistente

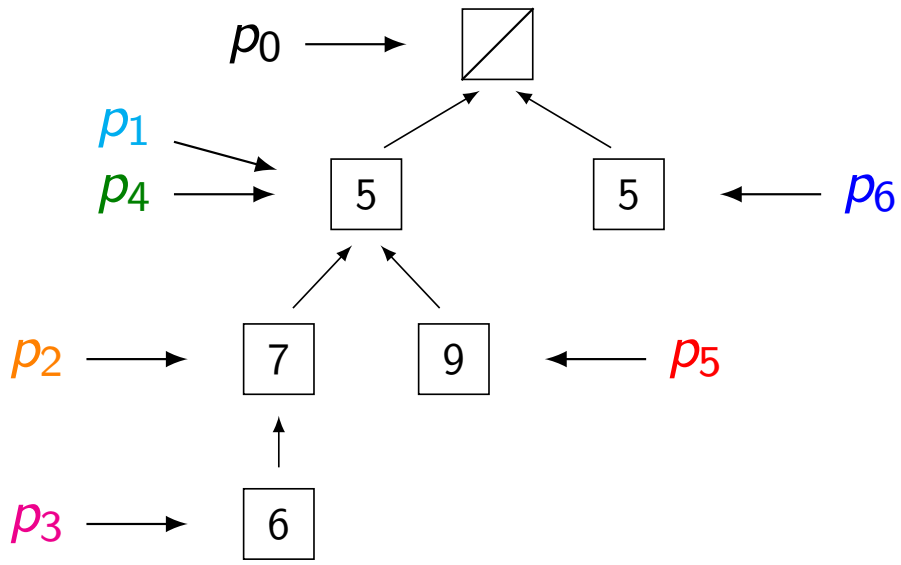
**Operações:**

$p_0 = \text{Stack}()$
$p_1 = \text{Push}(p_0, 5)$
$p_2 = \text{Push}(p_1, 7)$
$\text{Top}(p_2)$
$p_3 = \text{Push}(p_2, 6)$
$p_4 = \text{Pop}(p_2)$
$\text{Top}(p_3)$
$p_5 = \text{Push}(p_4, 9)$
$\text{Top}(p_4)$
$p_6 = \text{Push}(p_0, 5)$

**Pilhas:**

$p_0$ :
$p_1$ : 5
$p_2$ : 5 7
7
$p_3$ : 5 7 6
$p_4$ : 5
6
$p_5$ : 5 9
5
$p_6$ : 5

**Implementação:**



# Exemplo: pilha persistente

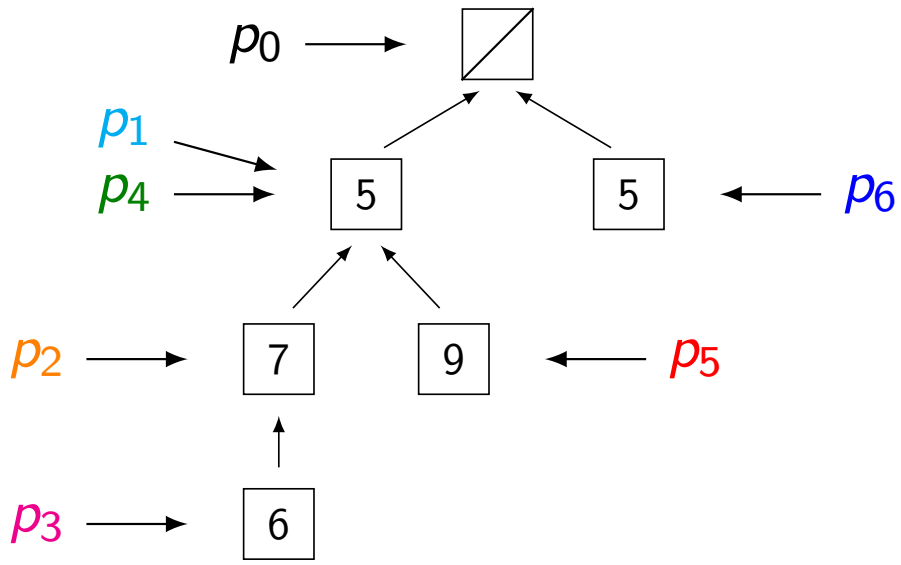
**Operações:**

$p_0 = \text{Stack}()$
$p_1 = \text{Push}(p_0, 5)$
$p_2 = \text{Push}(p_1, 7)$
$\text{Top}(p_2)$
$p_3 = \text{Push}(p_2, 6)$
$p_4 = \text{Pop}(p_2)$
$\text{Top}(p_3)$
$p_5 = \text{Push}(p_4, 9)$
$\text{Top}(p_4)$
$p_6 = \text{Push}(p_0, 5)$

**Pilhas:**

$p_0$ :
$p_1$ : 5
$p_2$ : 5 7
7
$p_3$ : 5 7 6
$p_4$ : 5
6
$p_5$ : 5 9
5
$p_6$ : 5

**Implementação:**




Espaço: número de pushs e pops mais número de pushs.

# ABB parcialmente persistente

Consultas a qualquer versão; alterações apenas na versão corrente.

# ABB parcialmente persistente

Consultas a qualquer versão; alterações apenas na versão corrente.

Como implementar?

Técnica: node copying

# ABB parcialmente persistente

Consultas a qualquer versão; alterações apenas na versão corrente.

Como implementar?

Técnica: node copying

Copie todos os nós da raiz até o nó inserido/removido.

# ABB parcialmente persistente

Consultas a qualquer versão; alterações apenas na versão corrente.

Como implementar?

Técnica: node copying

Copie todos os nós da raiz até o nó inserido/removido.

Podemos usar uma ABB balanceada.

# ABB parcialmente persistente

Consultas a qualquer versão; alterações apenas na versão corrente.

Como implementar?

Técnica: node copying

Copie todos os nós da raiz até o nó inserido/removido.

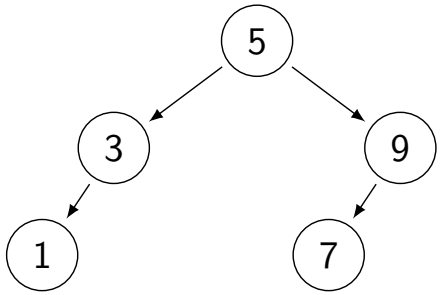
Podemos usar uma ABB balanceada.

Espaço consumido: $\mathrm{O}(m \lg n)$,
onde $m$ é o número de operações
e $n$ o número máximo de nós na ABB.

Tempo por operação: $\mathrm{O}(\lg n)$.

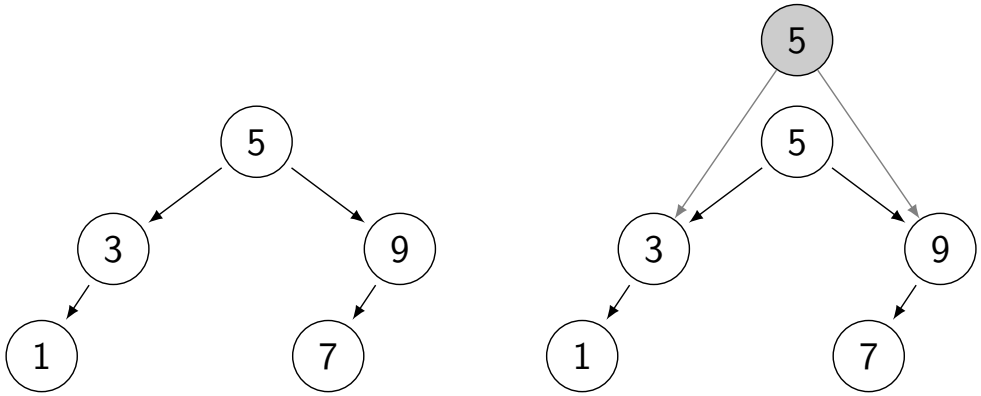
# Node copying aplicado a uma ABB

Adição de um nó com chave 4 na ABB.



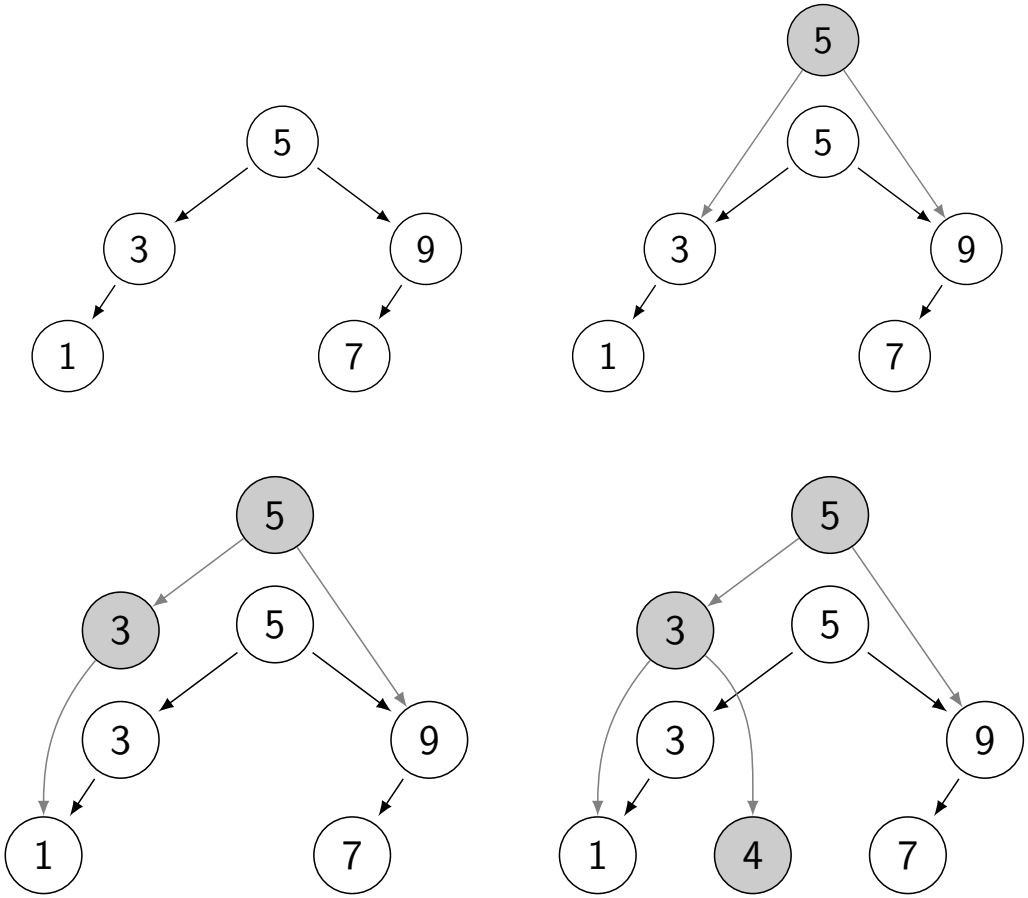
# Node copying aplicado a uma ABB

Adição de um nó com chave 4 na ABB.



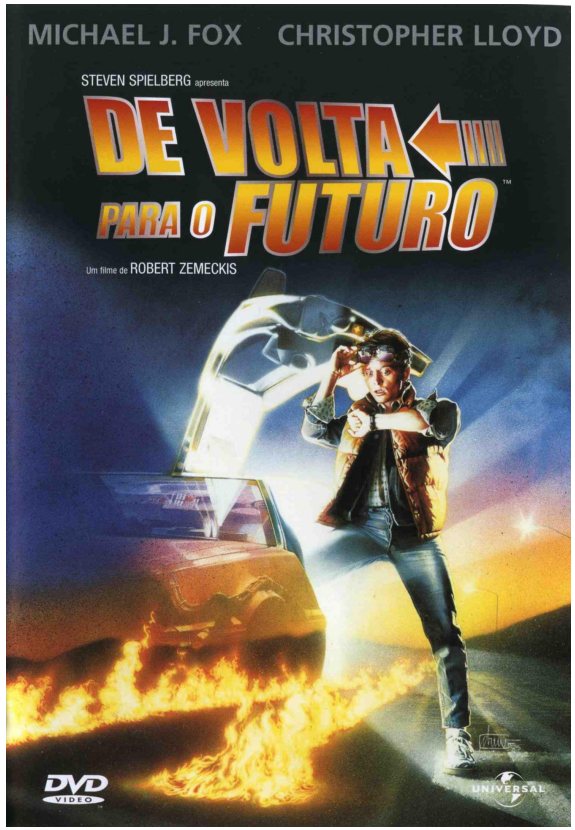
# Node copying aplicado a uma ABB

Adição de um nó com chave 4 na ABB.



# Retroatividade

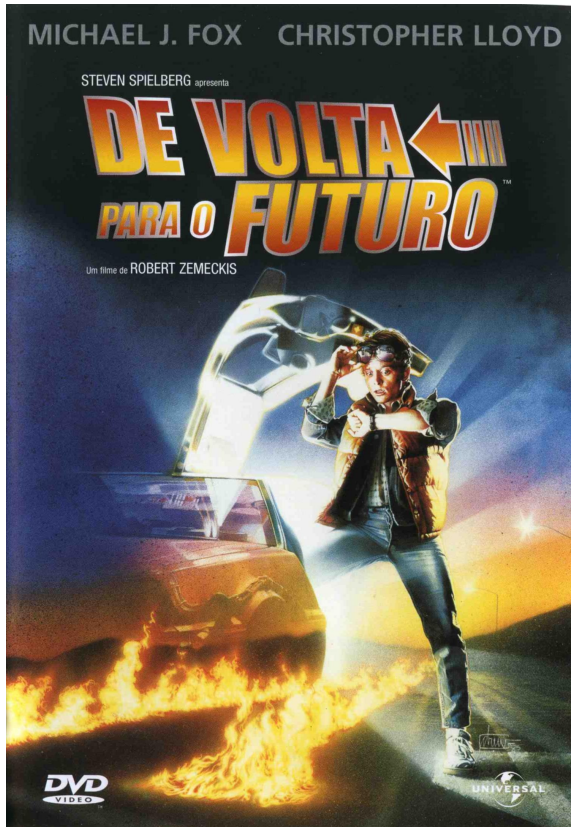
Como no filme
**De Volta para o Futuro**:
linha do tempo única.



# Retroatividade

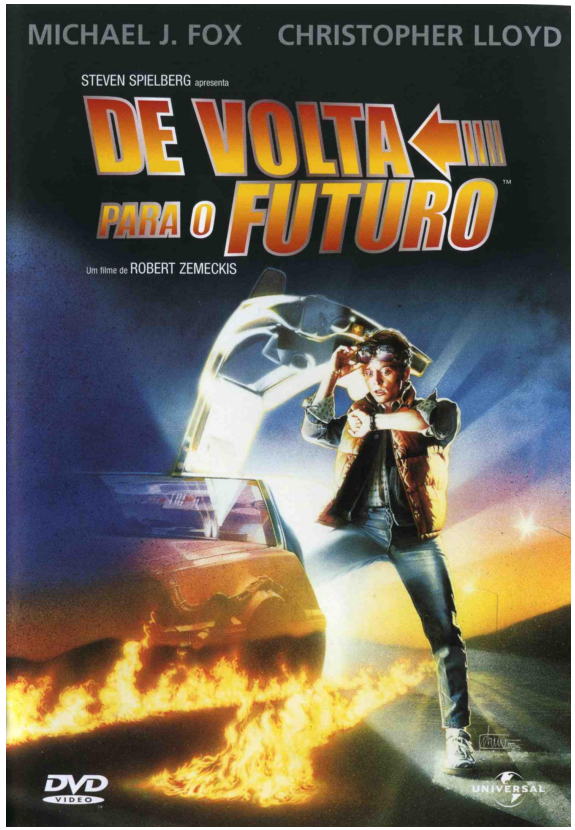
Como no filme
**De Volta para o Futuro**:
linha do tempo única.

Alterações no passado se
propagam pela linha do tempo.



# Retroatividade

Como no filme
**De Volta para o Futuro**:
linha do tempo única.




Alterações no passado se
propagam pela linha do tempo.

Retroatividade parcial:

- somente atualizações nas versões passadas
- consultas apenas na versão corrente

# Exemplo: fila retroativa

## Operações

- enqueue (enfile)
- dequeue (desenfile)
- first (primeiro) - consulta

# Exemplo: fila retroativa

## Operações

- enqueue (enfile)
- dequeue (desenfile)
- first (primeiro) - consulta

Todas custam tempo constante na implementação tradicional.

# Exemplo: fila retroativa

## Operações

- enqueue (enfile)
- dequeue (desenfile)
- first (primeiro) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar uma versão retroativa?

# Exemplo: fila retroativa

## Operações

- enqueue (enfile)
- dequeue (desenfile)
- first (primeiro) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar uma versão retroativa?

Cada operação está associada a um instante.

# Exemplo: fila retroativa

## Operações

- enqueue (enfile)
- dequeue (desenfile)
- first (primeiro) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar uma versão retroativa?

Cada operação está associada a um instante.

First(t): Se ocorreram $d$ dequeues até o instante $t$,
então o elemento do $d + 1$ enqueue é o primeiro (first) no instante $t$.

# Exemplo: fila retroativa

## Operações

- enqueue (enfile)
- dequeue (desenfile)
- first (primeiro) - consulta

Todas custam tempo constante na implementação tradicional.

Como implementar uma versão retroativa?

Cada operação está associada a um instante.

First(t): Se ocorreram $d$ dequeues até o instante $t$,
então o elemento do $d + 1$ enqueue é o primeiro (first) no instante $t$.

Enqueue/dequeue: Ordem dos enqueues/dequeue
pelo instante importa para futuras consultas.

# Exemplo: fila retroativa

**Operações:**

Queue()

**Fila corrente:**

$Q$ :

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
| --- | --- |
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q$ : $A$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q$ : $A$ |
| Enqueue(20, $B$) | $Q$ : $A$ $B$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q$ : $A$ |
| Enqueue(20, $B$) | $Q$ : $A$ $B$ |
| Enqueue(30, $C$) | $Q$ : $A$ $B$ $C$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q$ : $A$ |
| Enqueue(20, $B$) | $Q$ : $A$ $B$ |
| Enqueue(30, $C$) | $Q$ : $A$ $B$ $C$ |
| Enqueue(15, $D$) | |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q$ : $A$ |
| Enqueue(20, $B$) | $Q$ : $A$ $B$ |
| Enqueue(30, $C$) | $Q$ : $A$ $B$ $C$ |
| Enqueue(15, $D$) | $Q$ : $A$ $D$ $B$ $C$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q$ : $A$ |
| Enqueue(20, $B$) | $Q$ : $A$ $B$ |
| Enqueue(30, $C$) | $Q$ : $A$ $B$ $C$ |
| Enqueue(15, $D$) | $Q$ : $A$ $D$ $B$ $C$ |
| Dequeue(25) | $Q$ : ~~$A$~~ $D$ $B$ $C$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q$ : $A$ |
| Enqueue(20, $B$) | $Q$ : $A$ $B$ |
| Enqueue(30, $C$) | $Q$ : $A$ $B$ $C$ |
| Enqueue(15, $D$) | $Q$ : $A$ $D$ $B$ $C$ |
| Dequeue(25) | $Q$ : ~~$A$~~ $D$ $B$ $C$ |
| Dequeue(35) | $Q$ : ~~$A$~~ ~~$D$~~ $B$ $C$ |
| First(22) | |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q$ : $A$ |
| Enqueue(20, $B$) | $Q$ : $A$ $B$ |
| Enqueue(30, $C$) | $Q$ : $A$ $B$ $C$ |
| Enqueue(15, $D$) | $Q$ : $A$ $D$ $B$ $C$ |
| Dequeue(25) | $Q$ : ~~$A$~~ $D$ $B$ $C$ |
| Dequeue(35) | $Q$ : ~~$A$~~ ~~$D$~~ $B$ $C$ |
| First(22) | $A$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q$ : |
| Enqueue(10, $A$) | $Q : A$ |
| Enqueue(20, $B$) | $Q : A\ B$ |
| Enqueue(30, $C$) | $Q : A\ B\ C$ |
| Enqueue(15, $D$) | $Q : A\ D\ B\ C$ |
| Dequeue(25) | $Q : \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q : \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |
| First(32) | |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |
| First(32) | $A$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |
| First(32) | $A$ |
| Delete(25) | |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |
| First(32) | $A$ |
| Delete(25) | $Q: \cancel{E}\ A\ D\ B\ C$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |
| First(32) | $A$ |
| Delete(25) | $Q: \cancel{E}\ A\ D\ B\ C$ |
| First(32) | |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |
| First(32) | $A$ |
| Delete(25) | $Q: \cancel{E}\ A\ D\ B\ C$ |
| First(32) | $E$ |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |
| First(32) | $A$ |
| Delete(25) | $Q: \cancel{E}\ A\ D\ B\ C$ |
| First(32) | $E$ |
| Delete(10) | |

# Exemplo: fila retroativa

| **Operações:** | **Fila corrente:** |
|---|---|
| Queue() | $Q:$ |
| Enqueue(10, $A$) | $Q: A$ |
| Enqueue(20, $B$) | $Q: A\ B$ |
| Enqueue(30, $C$) | $Q: A\ B\ C$ |
| Enqueue(15, $D$) | $Q: A\ D\ B\ C$ |
| Dequeue(25) | $Q: \cancel{A}\ D\ B\ C$ |
| Dequeue(35) | $Q: \cancel{A}\ \cancel{D}\ B\ C$ |
| First(22) | $A$ |
| First(32) | $D$ |
| Enqueue(5, $E$) | $Q: \cancel{E}\ \cancel{A}\ D\ B\ C$ |
| First(32) | $A$ |
| Delete(25) | $Q: \cancel{E}\ A\ D\ B\ C$ |
| First(32) | $E$ |
| Delete(10) | $Q: \cancel{E}\ D\ B\ C$ |

# Exemplo: fila retroativa

First(t): Se ocorreram $d$ dequeus até o instante $t$,
então o elemento do $d + 1$ enqueue é o primeiro (first) no instante $t$.

# Exemplo: fila retroativa

First(t): Se ocorreram $d$ dequeus até o instante $t$,
então o elemento do $d + 1$ enqueue é o primeiro (first) no instante $t$.

Vamos manter

- enqueues ordenados pelo instante
- dequeues ordenados pelo instante

# Exemplo: fila retroativa

First(t): Se ocorreram $d$ dequeus até o instante $t$,
então o elemento do $d+1$ enqueue é o primeiro (first) no instante $t$.

Vamos manter

- enqueues ordenados pelo instante
- dequeues ordenados pelo instante

**Nessas listas ordenadas, precisamos**

- inserir e remover operações

# Exemplo: fila retroativa

First(t): Se ocorreram $d$ dequeus até o instante $t$,
então o elemento do $d+1$ enqueue é o primeiro (first) no instante $t$.

Vamos manter

- enqueues ordenados pelo instante
- dequeues ordenados pelo instante

**Nessas listas ordenadas, precisamos**

- inserir e remover operações
- dado $t$, determinar quantos dequeues há até o instante $t$

# Exemplo: fila retroativa

First(t): Se ocorreram $d$ dequeus até o instante $t$,
então o elemento do $d+1$ enqueue é o primeiro (first) no instante $t$.

Vamos manter

- enqueues ordenados pelo instante
- dequeues ordenados pelo instante

**Nessas listas ordenadas, precisamos**

- inserir e remover operações
- dado $t$, determinar quantos dequeues há até o instante $t$
- dado $k$, determinar o $k$-ésimo enqueue por instante

# Exemplo: fila retroativa

First(t): Se ocorreram $d$ dequeus até o instante $t$,
então o elemento do $d+1$ enqueue é o primeiro (first) no instante $t$.

Vamos manter

- enqueues ordenados pelo instante
- dequeues ordenados pelo instante

**Nessas listas ordenadas, precisamos**

- inserir e remover operações
- dado $t$, determinar quantos dequeues há até o instante $t$
- dado $k$, determinar o $k$-ésimo enqueue por instante

Que ED usar para dar suporte a estas operações?

# Qual ED usar?

**Queremos manter uma lista ordenada e dar suporte a**

(a) inserção e remoção

(b) número de elementos menores ou iguais a um dado $t$

(c) $k$-ésimo elemento da lista para um dado $k$

ABBB: árvore de busca binária balanceada!

# Qual ED usar?

**Queremos manter uma lista ordenada e dar suporte a**

(a) inserção e remoção

(b) número de elementos menores ou iguais a um dado $t$

(c) $k$-ésimo elemento da lista para um dado $k$

ABBB: árvore de busca binária balanceada!

- ABBB com os enqueues ordenados pelo instante
- ABBB com os dequeues ordenados pelo instante

# Qual ED usar?

**Queremos manter uma lista ordenada e dar suporte a**

(a) inserção e remoção

(b) número de elementos menores ou iguais a um dado $t$

(c) $k$-ésimo elemento da lista para um dado $k$

ABBB: árvore de busca binária balanceada!

- ABBB com os enqueues ordenados pelo instante
- ABBB com os dequeues ordenados pelo instante

Assim (a) é fácil.

# Qual ED usar?

**Queremos manter uma lista ordenada e dar suporte a**

(a) inserção e remoção

(b) número de elementos menores ou iguais a um dado $t$

(c) $k$-ésimo elemento da lista para um dado $k$

ABBB: árvore de busca binária balanceada!

- ABBB com os enqueues ordenados pelo instante
- ABBB com os dequeues ordenados pelo instante

Assim (a) é fácil. Como fazer (b) e (c)?

# Qual ED usar?

**Queremos manter uma lista ordenada e dar suporte a**

(a) inserção e remoção

(b) número de elementos menores ou iguais a um dado $t$

(c) $k$-ésimo elemento da lista para um dado $k$

ABBB: árvore de busca binária balanceada!

- ABBB com os enqueues ordenados pelo instante
- ABBB com os dequeues ordenados pelo instante

Assim (a) é fácil. Como fazer (b) e (c)?

- operações enqueue e dequeue nas folhas
- instante máximo da subárvore nos nós internos
- contador de folhas nos nós internos

# Qual ED usar?

**Queremos manter uma lista ordenada e dar suporte a**

(a) inserção e remoção

(b) número de elementos menores ou iguais a um dado $t$

(c) $k$-ésimo elemento da lista para um dado $k$

ABBB: árvore de busca binária balanceada!

- ABBB com os enqueues ordenados pelo instante
- ABBB com os dequeues ordenados pelo instante

Assim (a) é fácil. Como fazer (b) e (c)?

- operações enqueue e dequeue nas folhas
- instante máximo da subárvore nos nós internos
- contador de folhas nos nós internos

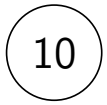
Custo por operação: $\mathrm{O}(\lg n)$

# Exemplo: fila retroativa

**Operações:**

Queue()
Enqueue(10, A)

**ABBB dos Enqueues:**

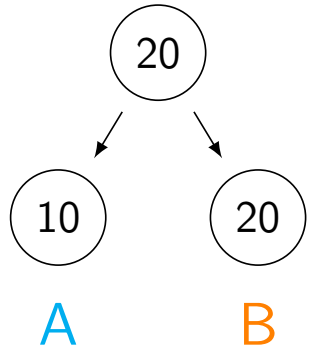


# Exemplo: fila retroativa

**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)

**ABBB dos Enqueues:**

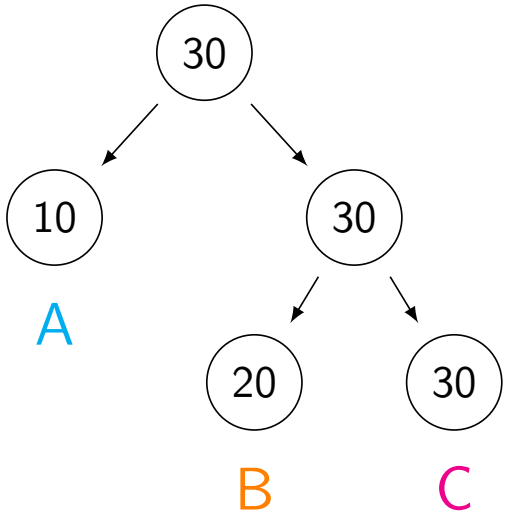


# Exemplo: fila retroativa

**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)

**ABBB dos Enqueues:**

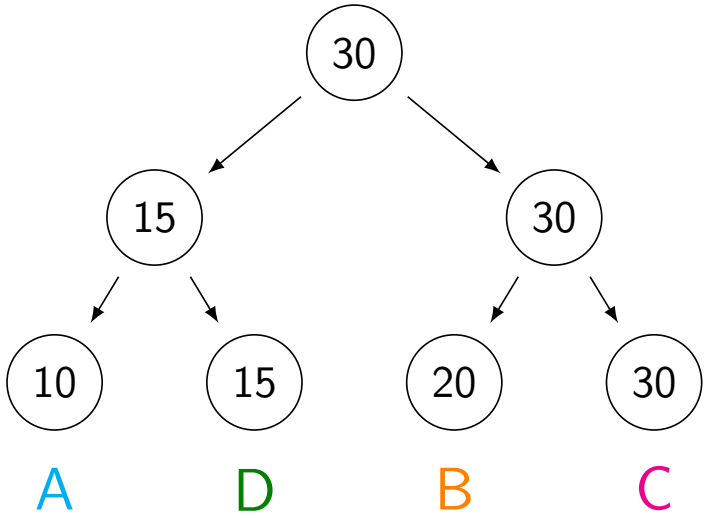


# Exemplo: fila retroativa

**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)

**ABBB dos Enqueues:**

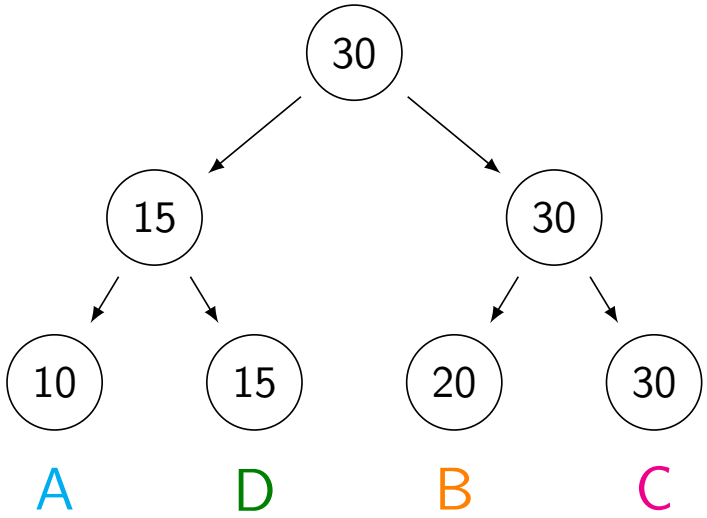


# Exemplo: fila retroativa

**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)

**ABBB dos Enqueues:**



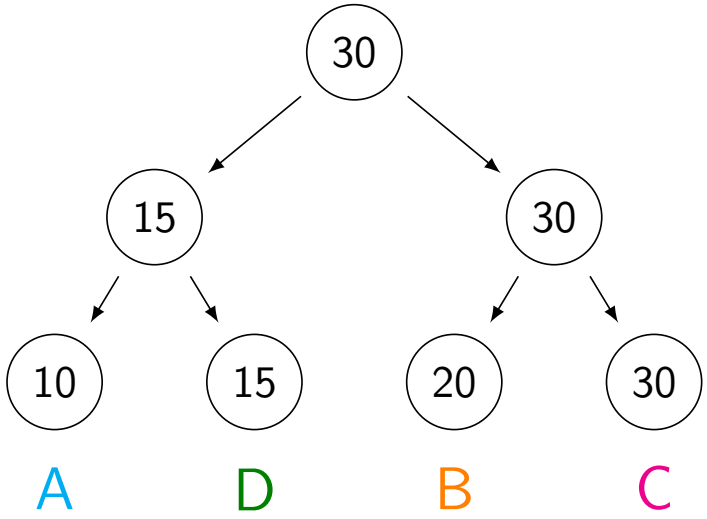

**ABBB dos Dequeues:**

# Exemplo: fila retroativa

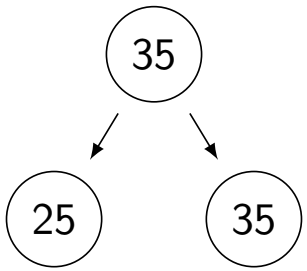
**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)

**ABBB dos Enqueues:**




**ABBB dos Dequeues:**

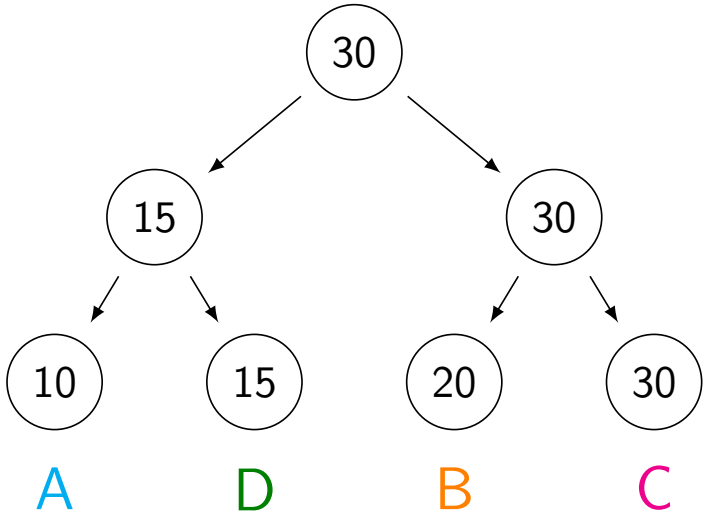


# Exemplo: fila retroativa

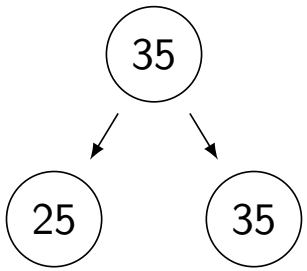
**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22)

**ABBB dos Enqueues:**




**ABBB dos Dequeues:**

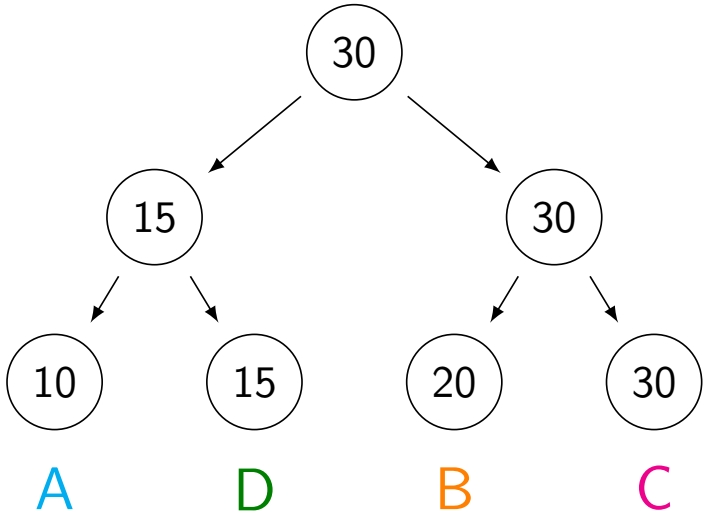


# Exemplo: fila retroativa

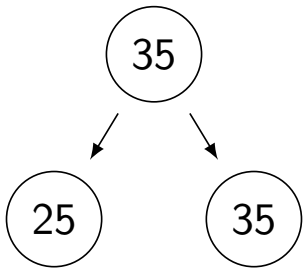
**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22) : A
First(32)

**ABBB dos Enqueues:**




**ABBB dos Dequeues:**

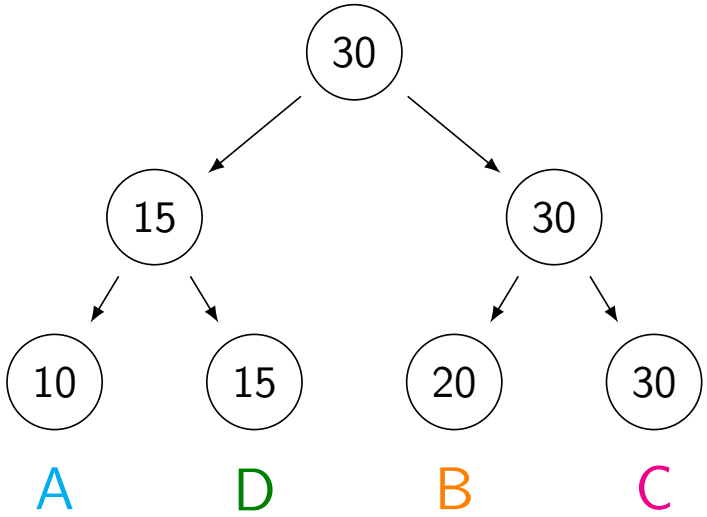


# Exemplo: fila retroativa

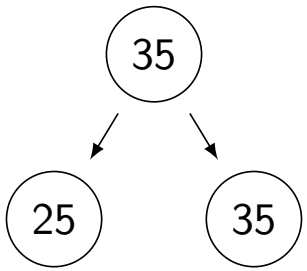
**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22) : A
First(32) : D

**ABBB dos Enqueues:**




**ABBB dos Dequeues:**

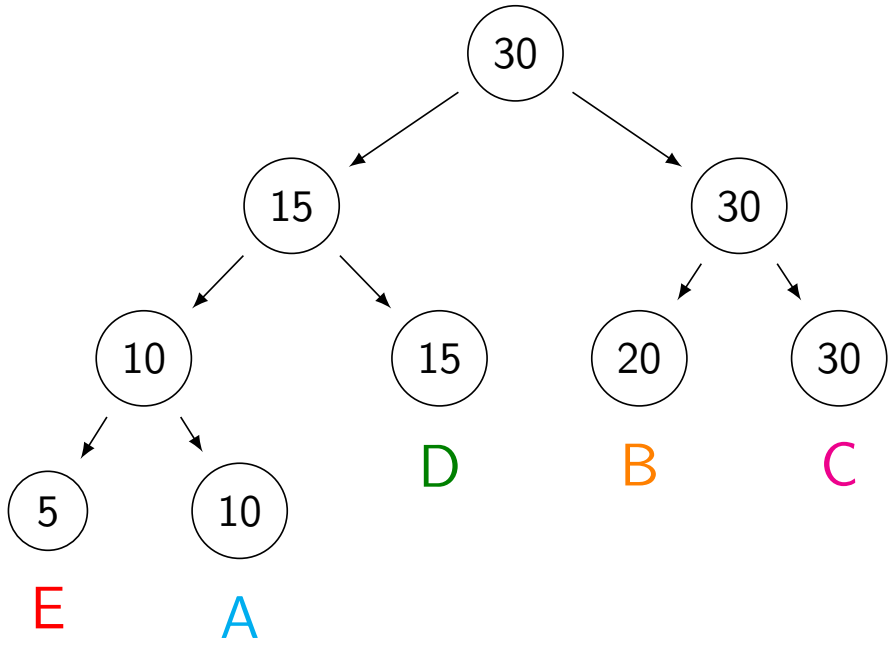


# Exemplo: fila retroativa

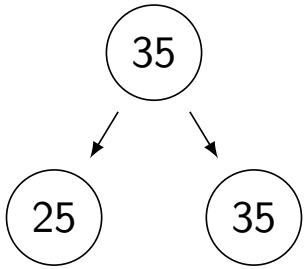
**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22) : A
First(32) : D
Enqueue(5, E)

**ABBB dos Enqueues:**




**ABBB dos Dequeues:**

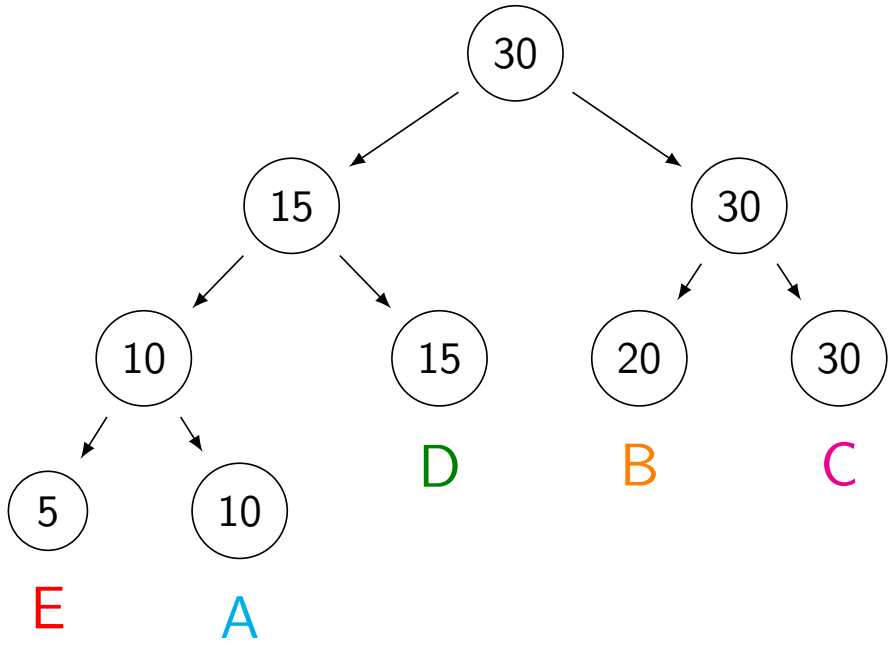


# Exemplo: fila retroativa

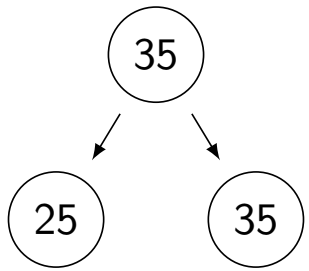
**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22) : A
First(32) : D
Enqueue(5, E)
First(32)

**ABBB dos Enqueues:**




**ABBB dos Dequeues:**

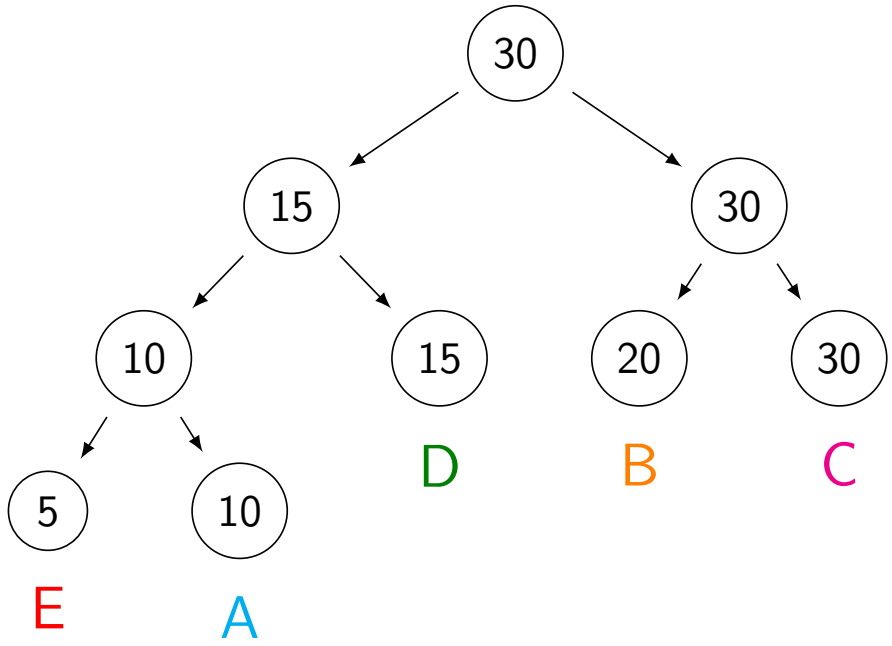


# Exemplo: fila retroativa

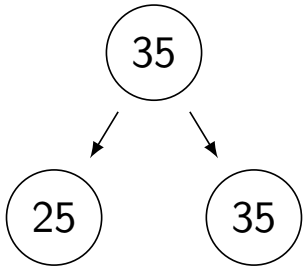
**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22) : A
First(32) : D
Enqueue(5, E)
First(32) : A

**ABBB dos Enqueues:**




**ABBB dos Dequeues:**

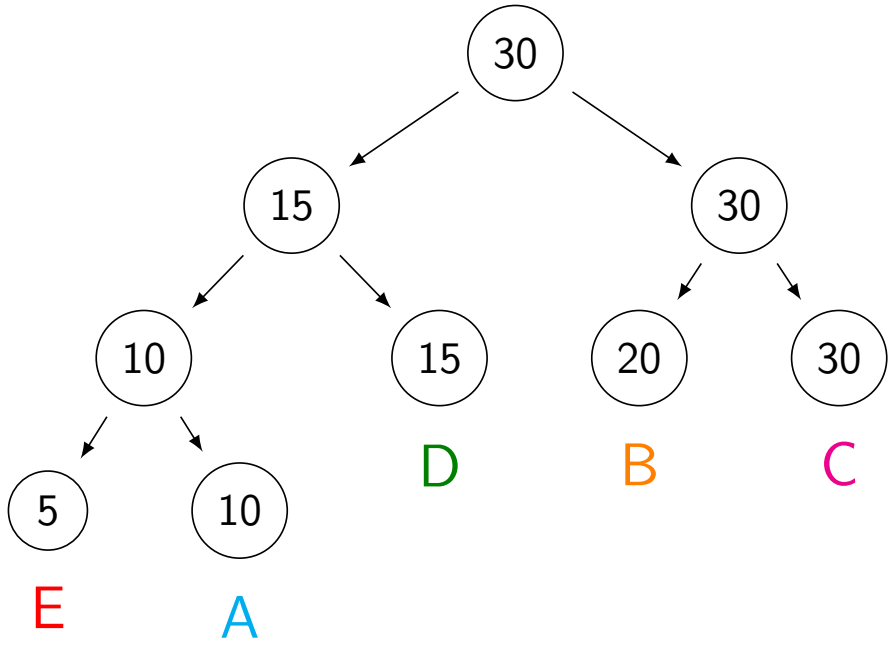


# Exemplo: fila retroativa

**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22) : A
First(32) : D
Enqueue(5, E)
First(32) : A
Delete(25)

**ABBB dos Enqueues:**



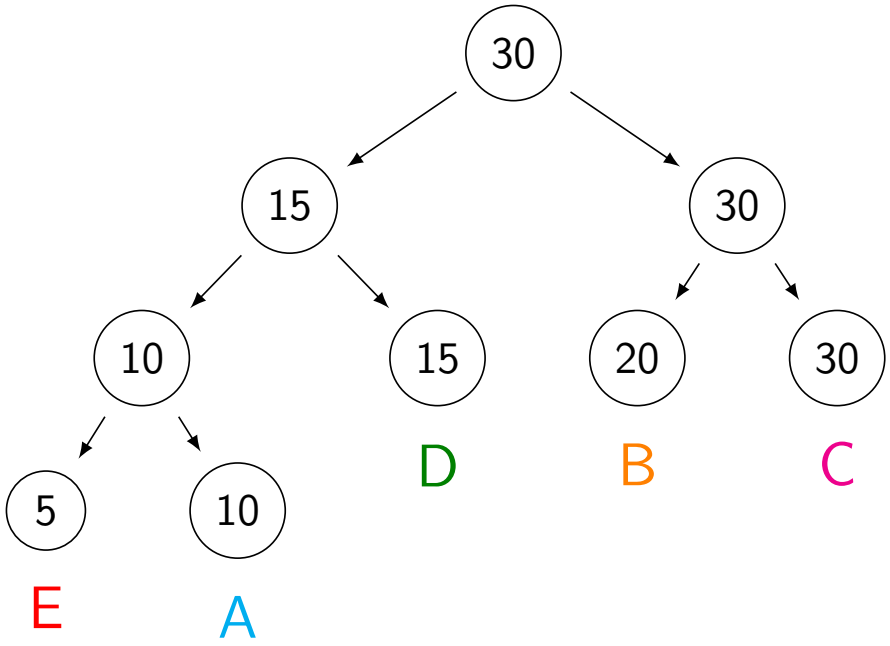

**ABBB dos Dequeues:**

# Exemplo: fila retroativa

**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22) : A
First(32) : D
Enqueue(5, E)
First(32) : A
Delete(25)
First(32)

**ABBB dos Enqueues:**



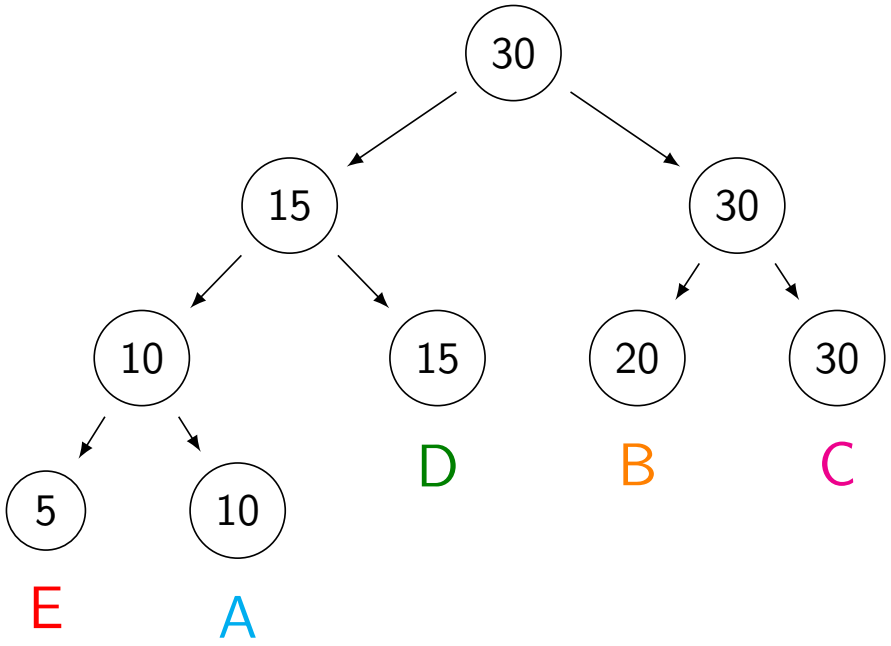

**ABBB dos Dequeues:**

# Exemplo: fila retroativa

**Operações:**

Queue()
Enqueue(10, A)
Enqueue(20, B)
Enqueue(30, C)
Enqueue(15, D)
Dequeue(25)
Dequeue(35)
First(22) : A
First(32) : D
Enqueue(5, E)
First(32) : A
Delete(25)
First(32) : E

**ABBB dos Enqueues:**




**ABBB dos Dequeues:**

# Uma aplicação: em que janela está o mouse?

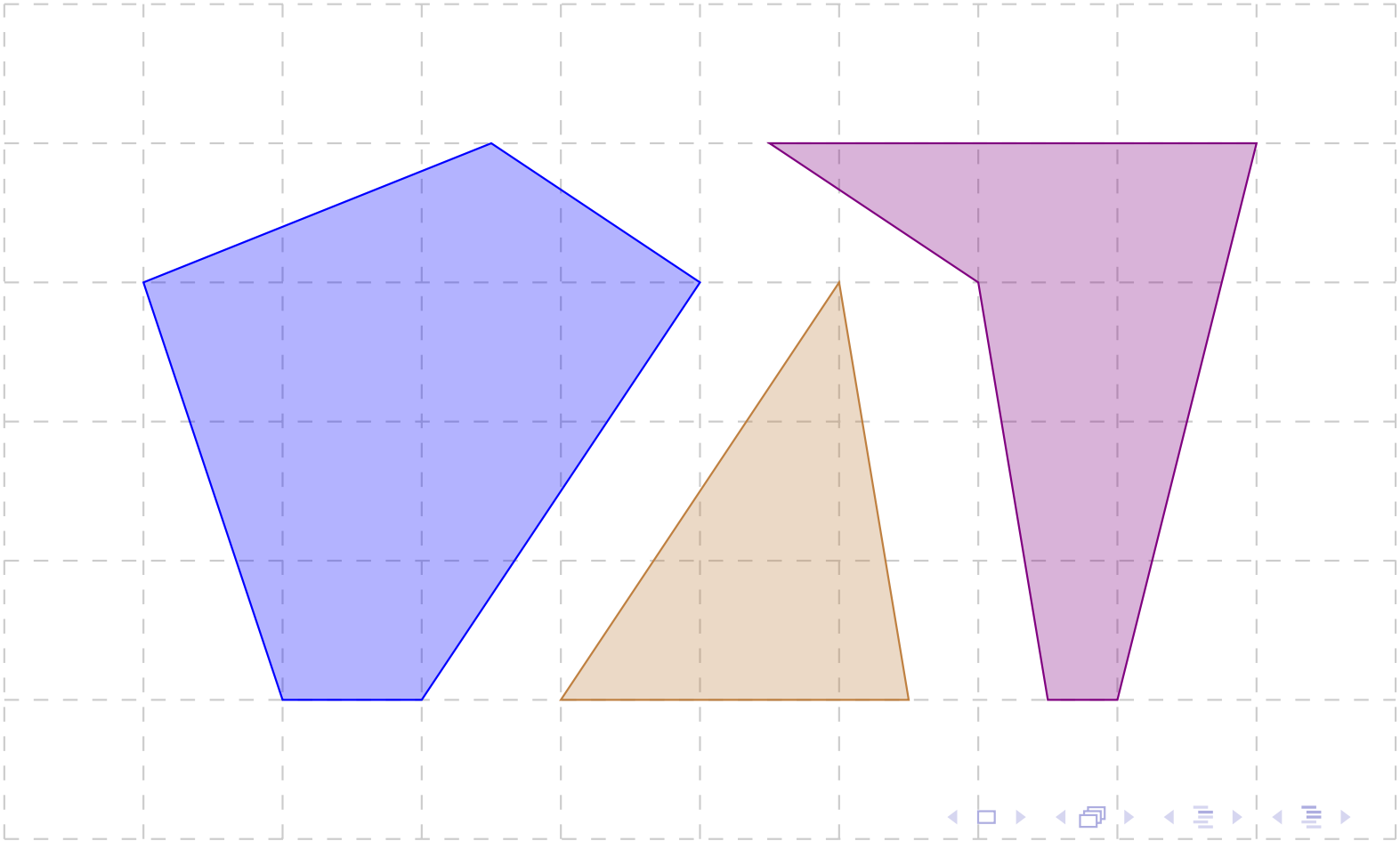
## Localização de ponto

Considere uma coleção de polígonos disjuntos e consultas do tipo: dado um ponto, em qual polígono o ponto se encontra?

# Uma aplicação: em que janela está o mouse?

## Localização de ponto

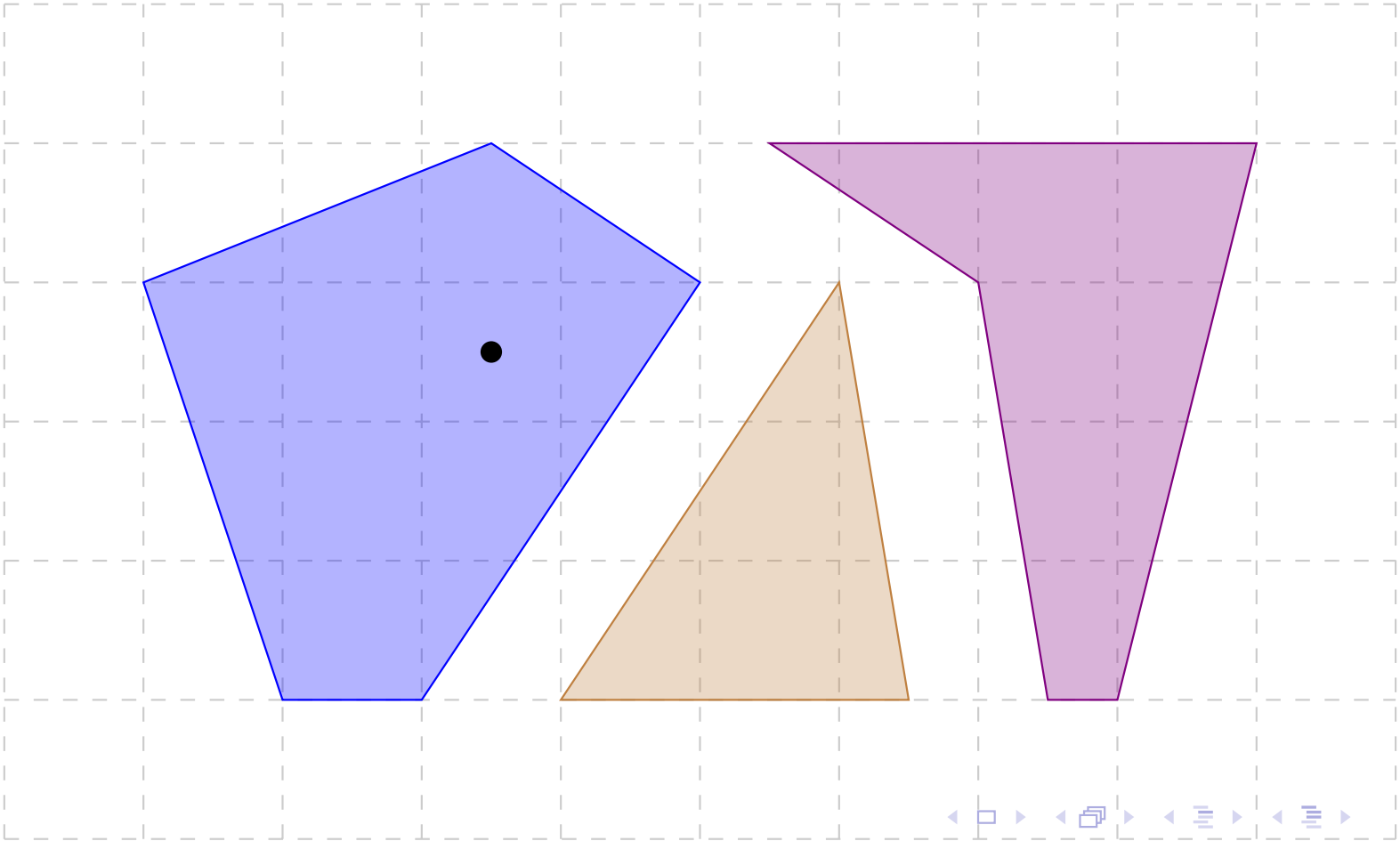
Considere uma coleção de polígonos disjuntos e consultas do tipo: dado um ponto, em qual polígono o ponto se encontra?



# Uma aplicação: em que janela está o mouse?

## Localização de ponto

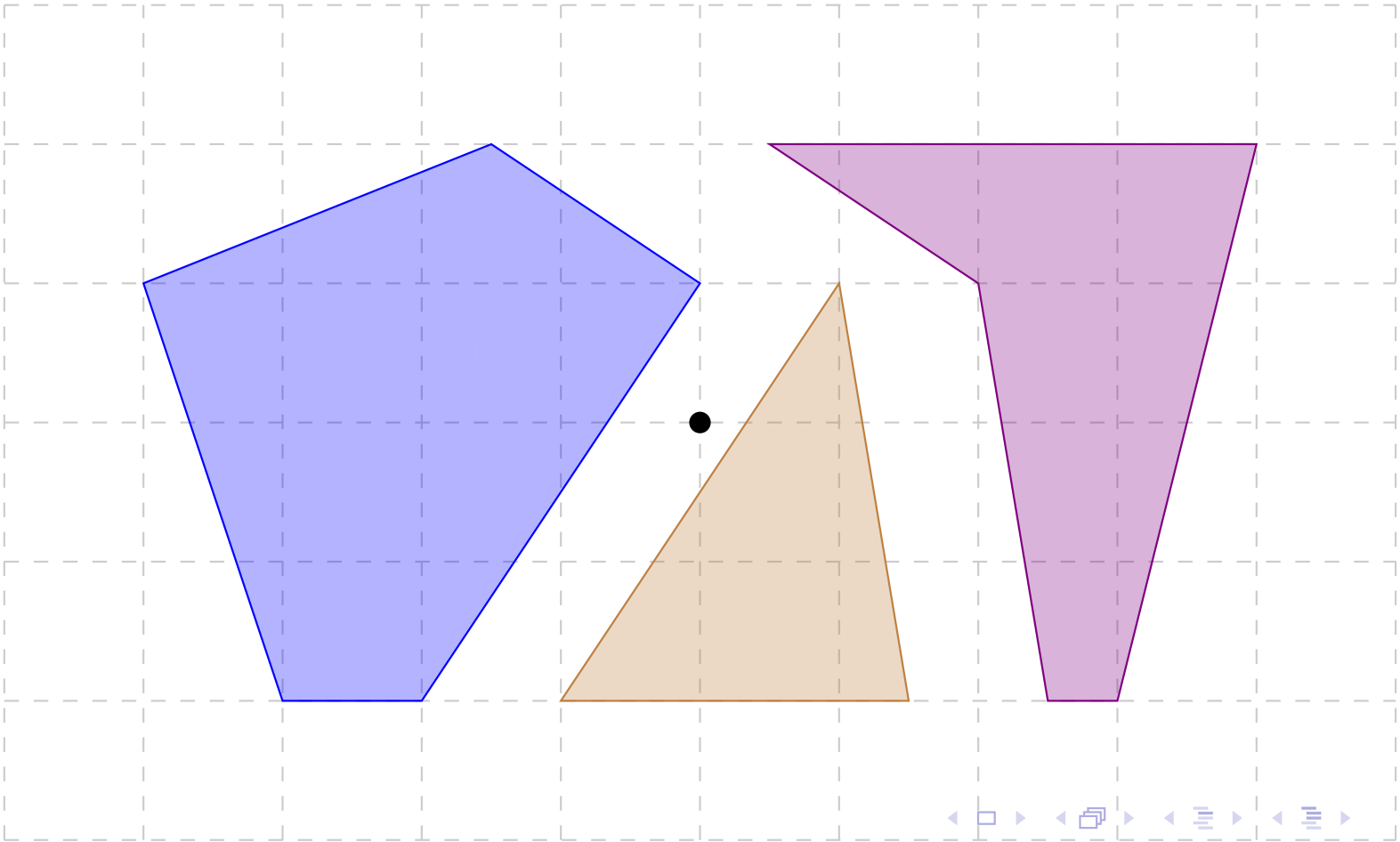
Considere uma coleção de polígonos disjuntos e consultas do tipo: dado um ponto, em qual polígono o ponto se encontra?



# Uma aplicação: em que janela está o mouse?

## Localização de ponto

Considere uma coleção de polígonos disjuntos e consultas do tipo: dado um ponto, em qual polígono o ponto se encontra?



# Uma aplicação: em que janela está o mouse?

## Localização de ponto

Considere uma coleção de polígonos disjuntos e consultas do tipo: dado um ponto, em qual polígono o ponto se encontra?

Ideia ingênua: teste se o ponto pertence a algum dos polígonos.

# Uma aplicação: em que janela está o mouse?

## Localização de ponto

Considere uma coleção de polígonos disjuntos e consultas do tipo: dado um ponto, em qual polígono o ponto se encontra?

Ideia ingênua: teste se o ponto pertence a algum dos polígonos.

Para cada ponto, no pior caso consome tempo $O(n)$
onde $n$ é o número total de vértices dos polígonos.

# Uma aplicação: em que janela está o mouse?

## Localização de ponto

Considere uma coleção de polígonos disjuntos e consultas do tipo:
dado um ponto, em qual polígono o ponto se encontra?

Ideia ingênua: teste se o ponto pertence a algum dos polígonos.

Para cada ponto, no pior caso consome tempo $O(n)$
onde $n$ é o número total de vértices dos polígonos.

Ideia melhor:
pré-processar os polígonos e guardar informação
para depois responder as consultas rapidamente.

# Uma aplicação: em que janela está o mouse?

## Localização de ponto

Considere uma coleção de polígonos disjuntos e consultas do tipo:
dado um ponto, em qual polígono o ponto se encontra?

Ideia ingênua: teste se o ponto pertence a algum dos polígonos.

Para cada ponto, no pior caso consome tempo $O(n)$
onde $n$ é o número total de vértices dos polígonos.

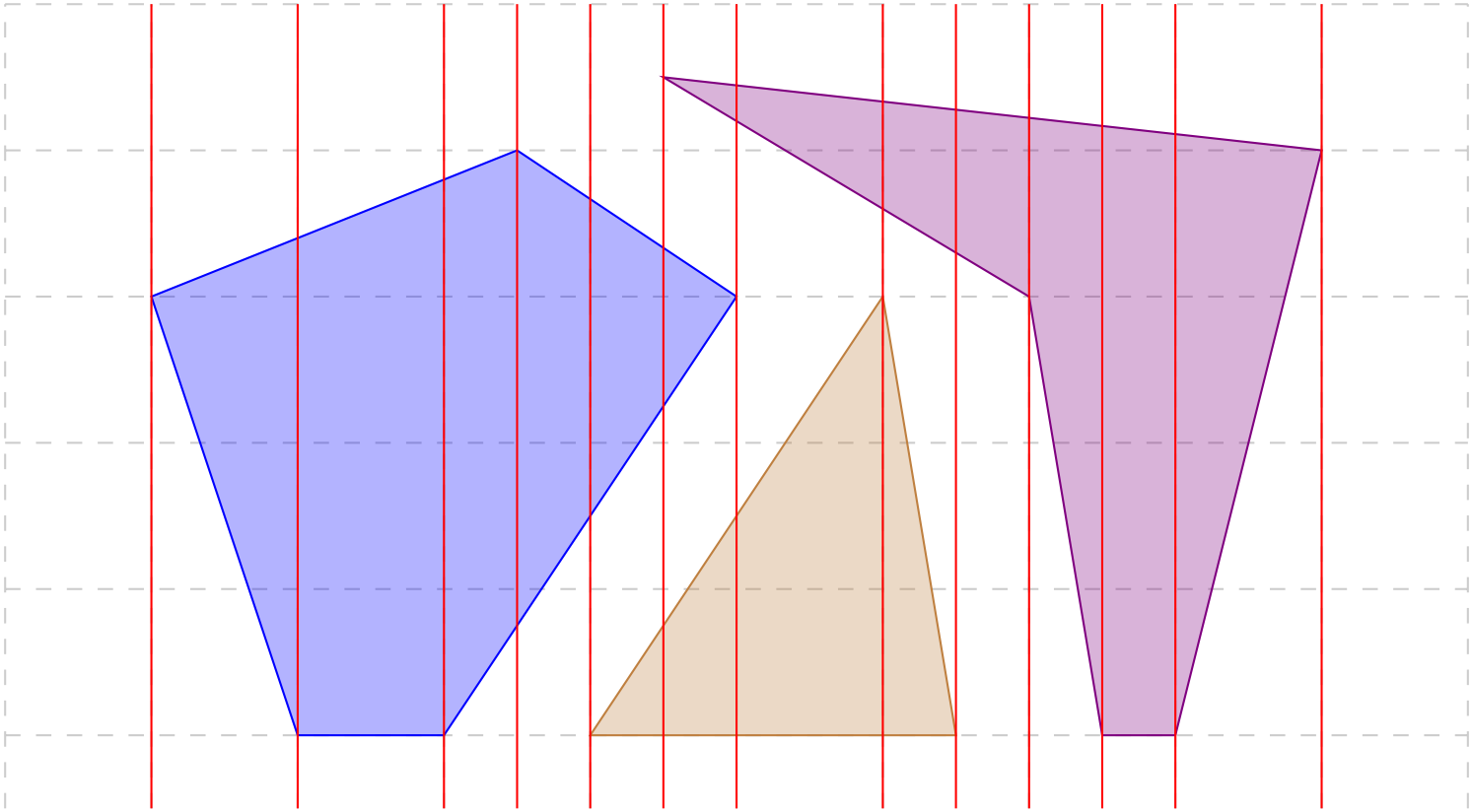
Ideia melhor:
pré-processar os polígonos e guardar informação
para depois responder as consultas rapidamente.

Como pré-processar os polígonos?

Como guardar a informação para as consultas serem rápidas?

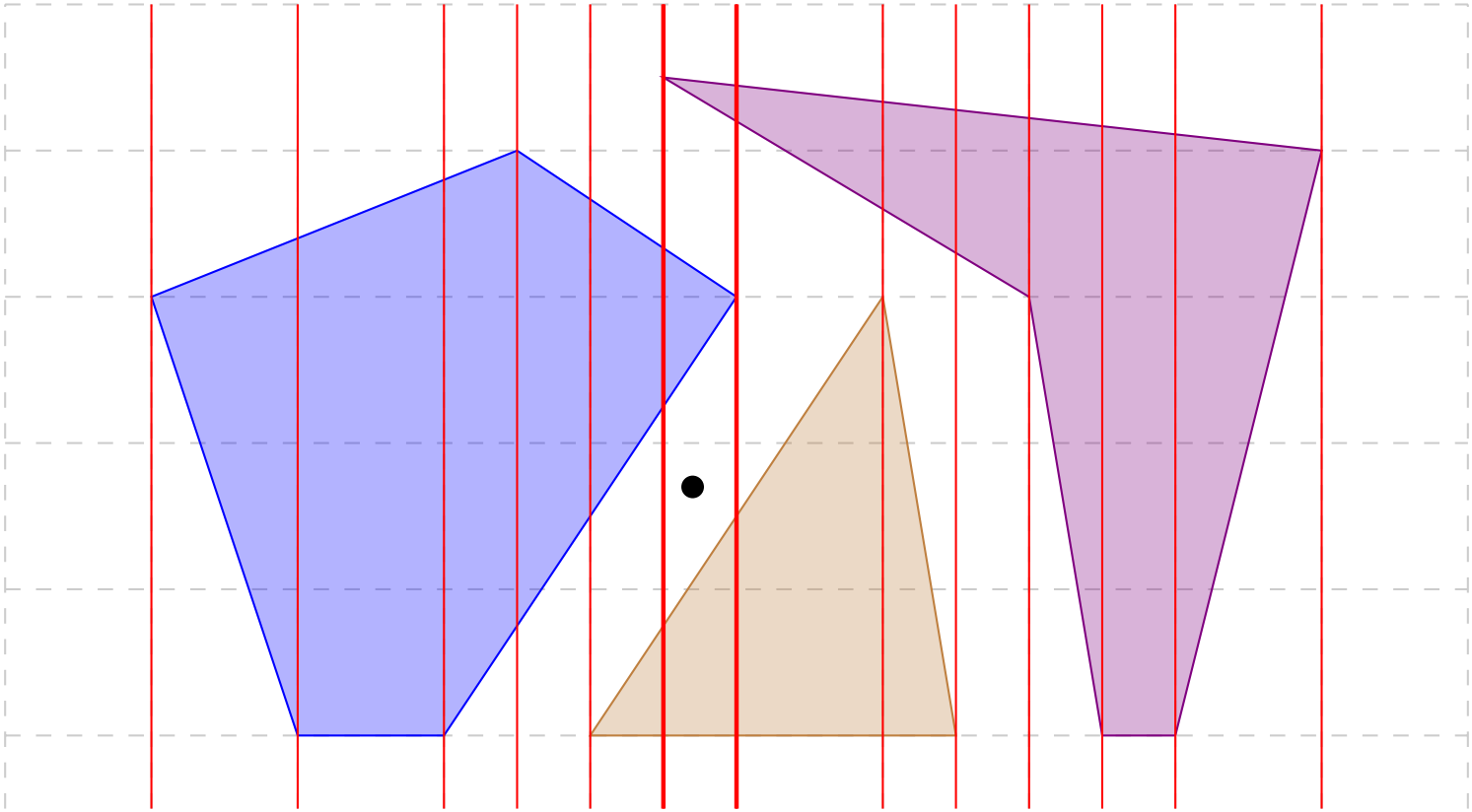
# Decomposição em faixas

Particiona a região em faixas que não contem nenhum vértice no interior.



# Decomposição em faixas

Particiona a região em faixas que não contem nenhum vértice no interior.



Consulta: determine a faixa em que o ponto está pela sua coordenada $x$; depois busque entre quais segmentos da faixa o ponto está.

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

Cada posição de $F$ está associada a uma faixa.

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

Cada posição de $F$ está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

Cada posição de $F$ está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

Cada posição de $F$ está associada a uma faixa.

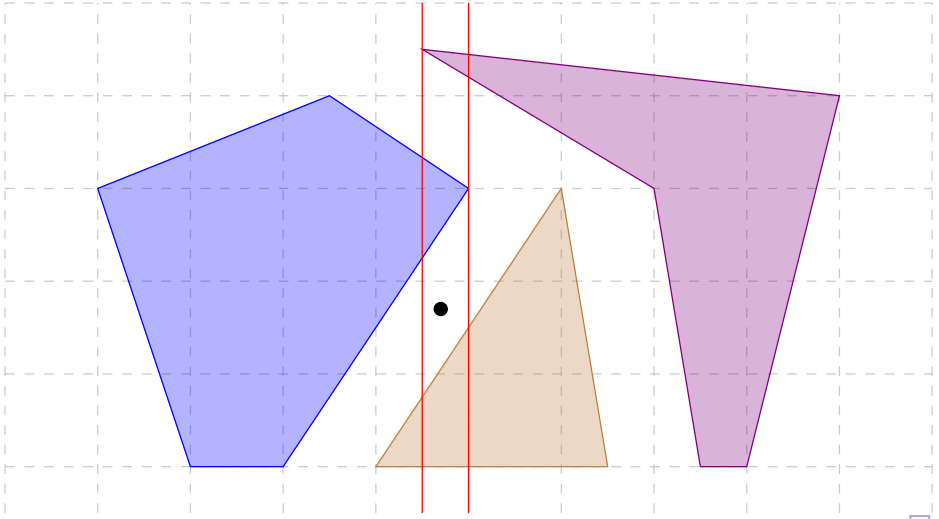
**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.



# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

Cada posição de $F$ está associada a uma faixa.

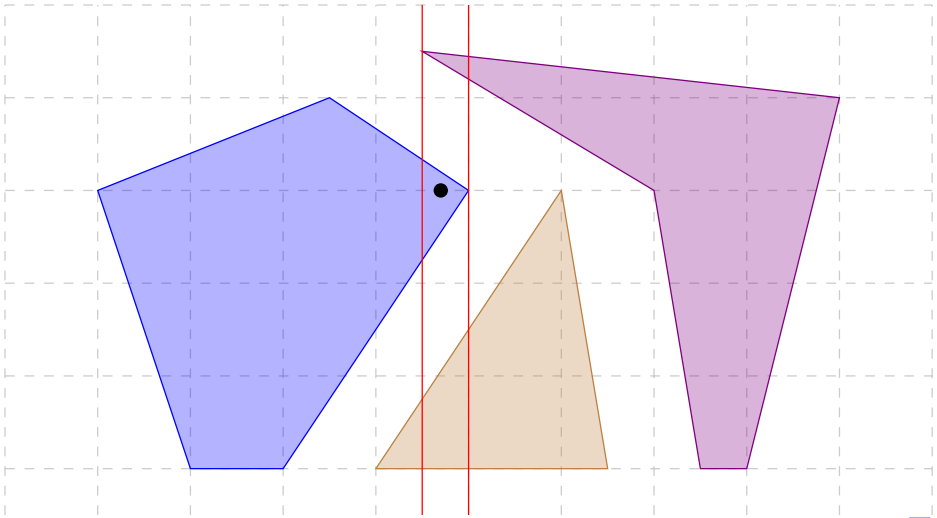
**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.



# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

Cada posição de $F$, exceto a última, está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.

Como decidir se um ponto está de um lado ou de outro de um segmento?

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

Cada posição de $F$, exceto a última, está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

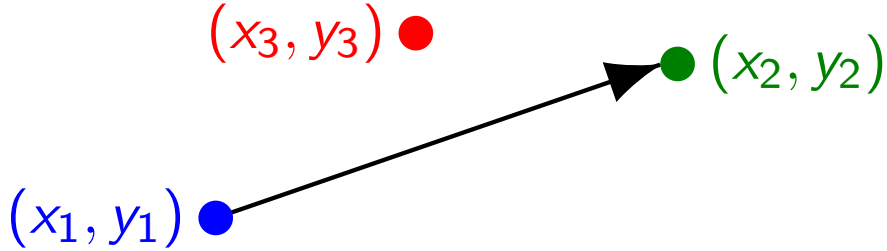
Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.

Como decidir se um ponto está de um lado ou de outro de um segmento?

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3)) =$ verdade



# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos vértices dos polígonos.

Cada posição de $F$, exceto a última, está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

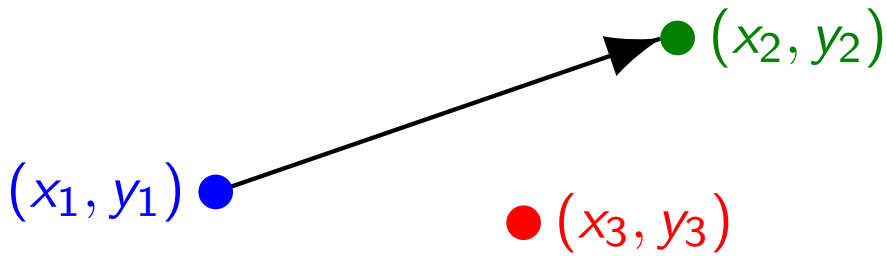
Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.

Como decidir se um ponto está de um lado ou de outro de um segmento?

Esquerda$((x_1, y_1), (x_2, y_2), (x_3, y_3)) =$ falso



# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos pontos dos polígonos.

Cada posição de $F$, exceto a última, está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.

Quanto tempo leva uma consulta?

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos pontos dos polígonos.

Cada posição de $F$, exceto a última, está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.

Quanto tempo leva uma consulta?

$O(\lg n)$ no pior caso, onde $n$ é o número total de vértices.

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos pontos dos polígonos.

Cada posição de $F$, exceto a última, está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.

Quanto tempo leva uma consulta?

$O(\lg n)$ no pior caso, onde $n$ é o número total de vértices.

Quanto espaço esse método usa?

# Decomposição em faixas

$F$: vetor ordenado com a coordenada $x$ dos pontos dos polígonos.

Cada posição de $F$, exceto a última, está associada a uma faixa.

**Como armazenar as arestas de uma faixa?**

Ordenadas de cima para baixo.

Ao processar um ponto,

(a) faça uma busca binária no vetor $F$ pela coordenada $x$ do ponto.

(b) faça uma busca binária nas arestas da faixa pelo ponto.

Quanto tempo leva uma consulta?

$O(\lg n)$ no pior caso, onde $n$ é o número total de vértices.

Quanto espaço esse método usa?

$O(n)$ pelo vetor $F$ e no pior caso $O(n)$ por faixa, num total de $O(n^2)$.

# Como obter as faixas?

Ideia ingênua: teste uma a uma cada aresta, depois ordene a faixa.

# Como obter as faixas?

Ideia ingênua: teste uma a uma cada aresta, depois ordene a faixa.

Consume tempo $\mathrm{O}(n^2 \lg n)$ pois são $n$ faixas, cada uma com $\mathrm{O}(n)$ arestas.

# Como obter as faixas?

Ideia ingênua: teste uma a uma cada aresta, depois ordene a faixa.

Consume tempo $\mathrm{O}(n^2 \lg n)$ pois são $n$ faixas, cada uma com $\mathrm{O}(n)$ arestas.

## Linha de varredura

reduzir um problema estático bidimensional
a um problema dinâmico unidimensional.

# Como obter as faixas?

Ideia ingênua: teste uma a uma cada aresta, depois ordene a faixa.

Consume tempo $\mathrm{O}(n^2 \lg n)$ pois são $n$ faixas, cada uma com $\mathrm{O}(n)$ arestas.

## Linha de varredura

reduzir um problema estático bidimensional
a um problema dinâmico unidimensional.

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

# Como obter as faixas?

Ideia ingênua: teste uma a uma cada aresta, depois ordene a faixa.

Consume tempo $\mathrm{O}(n^2 \lg n)$ pois são $n$ faixas, cada uma com $\mathrm{O}(n)$ arestas.

**Linha de varredura**

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 que está à esquerda dela é processado.

# Como obter as faixas?

Ideia ingênua: teste uma a uma cada aresta, depois ordene a faixa.

Consume tempo $\mathrm{O}(n^2 \lg n)$ pois são $n$ faixas, cada uma com $\mathrm{O}(n)$ arestas.

## Linha de varredura

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 que está à esquerda dela é processado.

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

# Como obter as faixas?

Ideia ingênua: teste uma a uma cada aresta, depois ordene a faixa.

Consume tempo $\mathrm{O}(n^2 \lg n)$ pois são $n$ faixas, cada uma com $\mathrm{O}(n)$ arestas.

**Linha de varredura**

reduzir um problema estático bidimensional
a um problema dinâmico unidimensional.

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

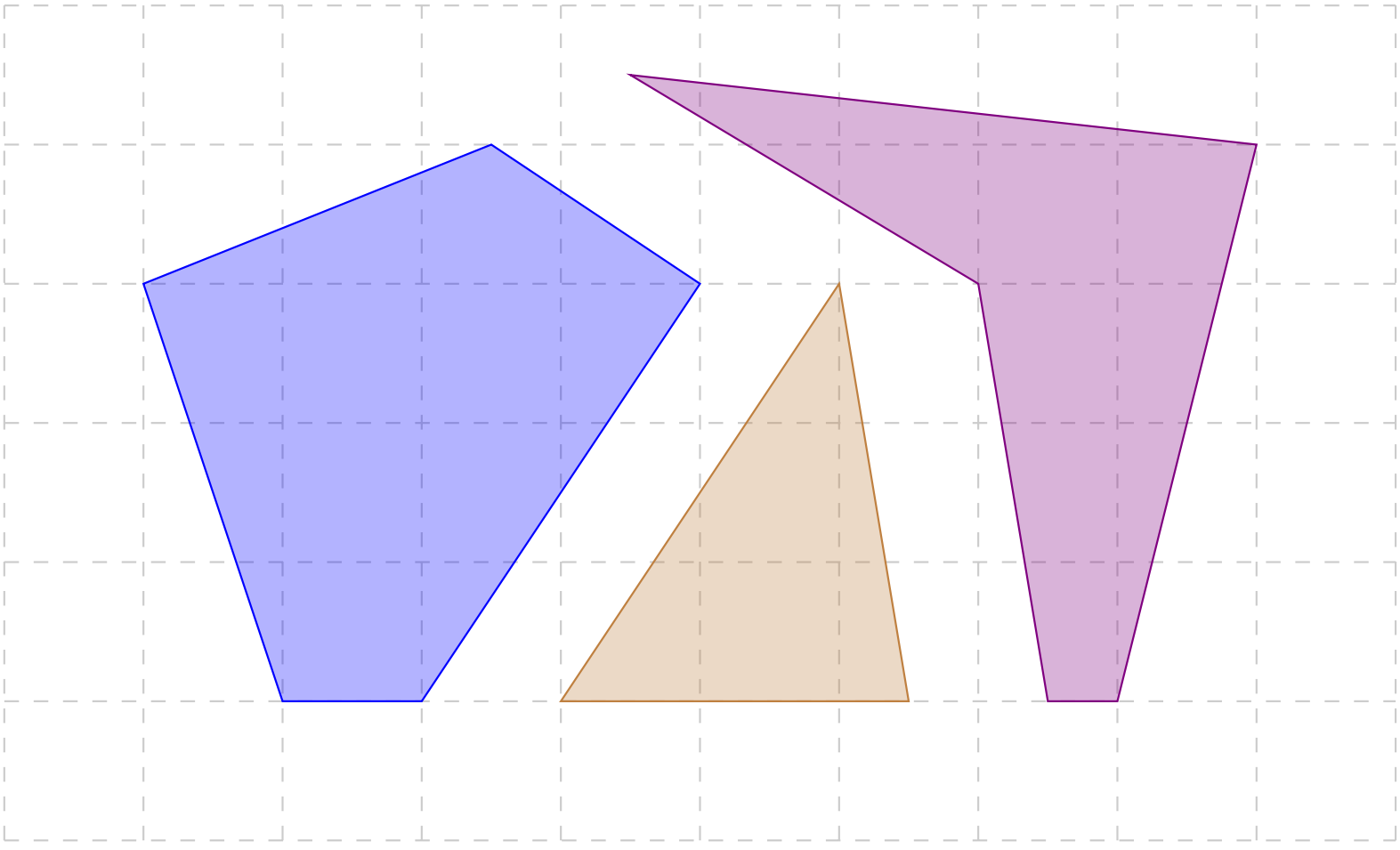
À medida que ela move, o que está à esquerda dela é processado.

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

Muda apenas em posições chaves: os pontos eventos.

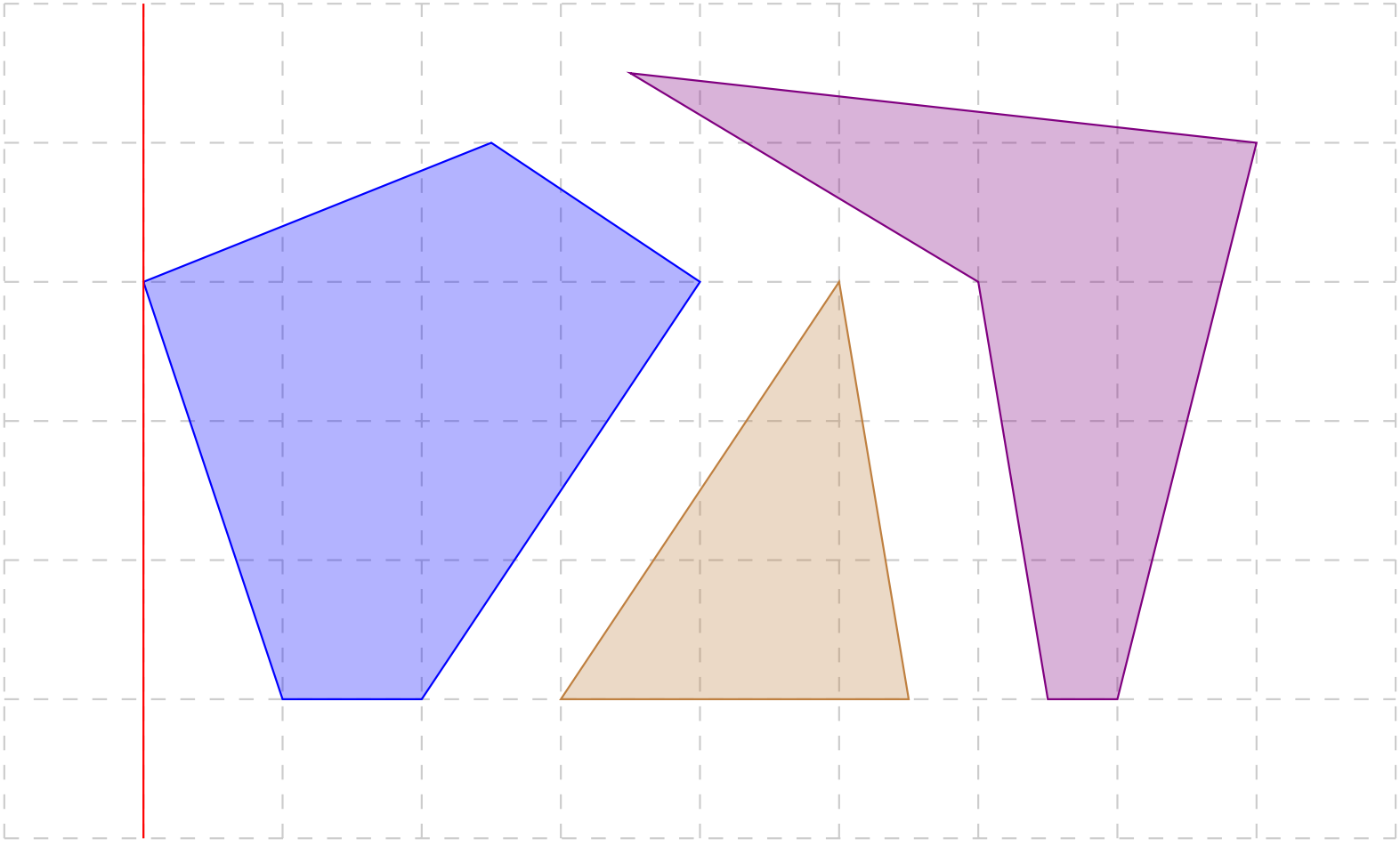
# Linha de varredura

Pontos eventos: vértices dos polígonos



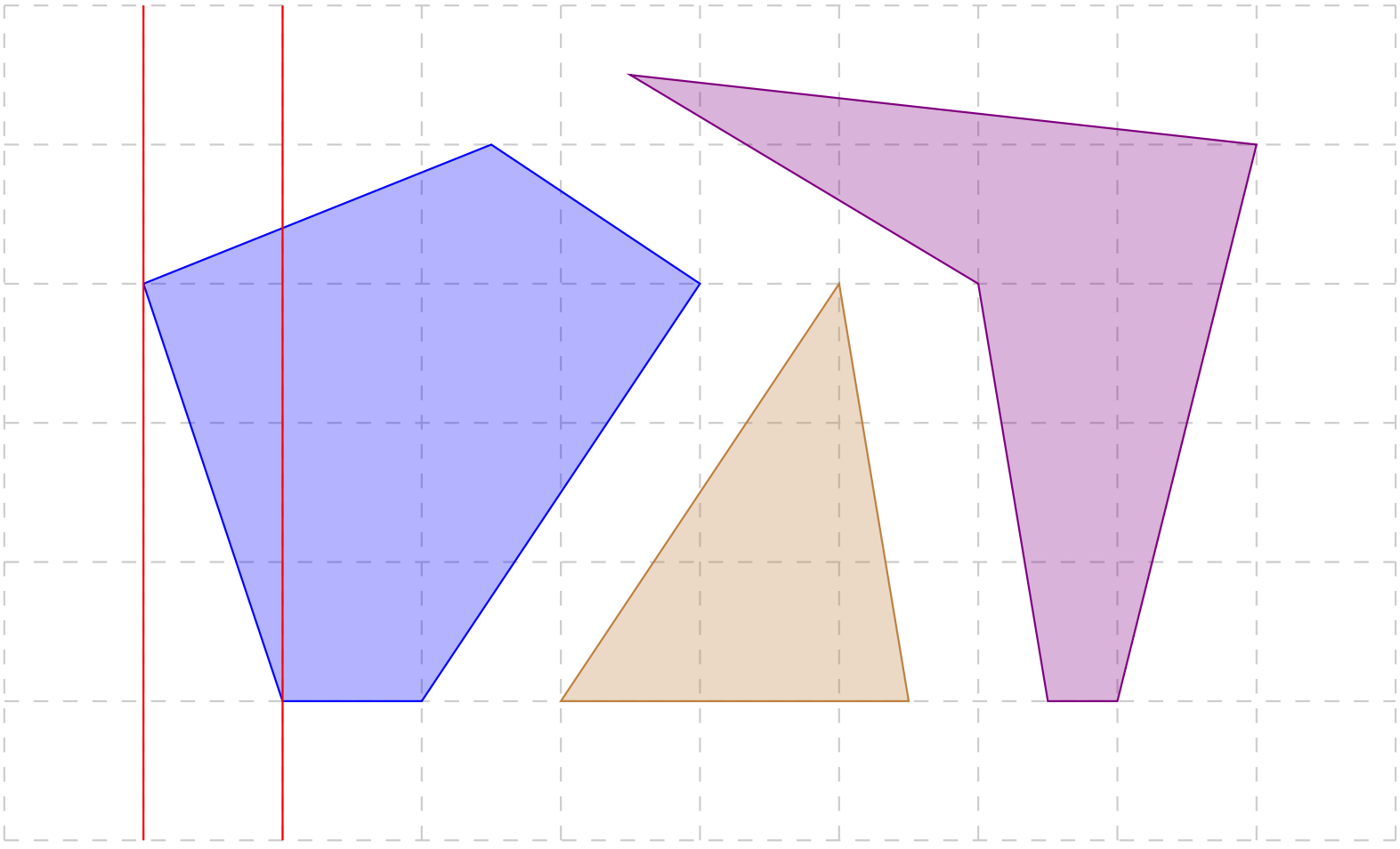
# Linha de varredura

Pontos eventos: vértices dos polígonos



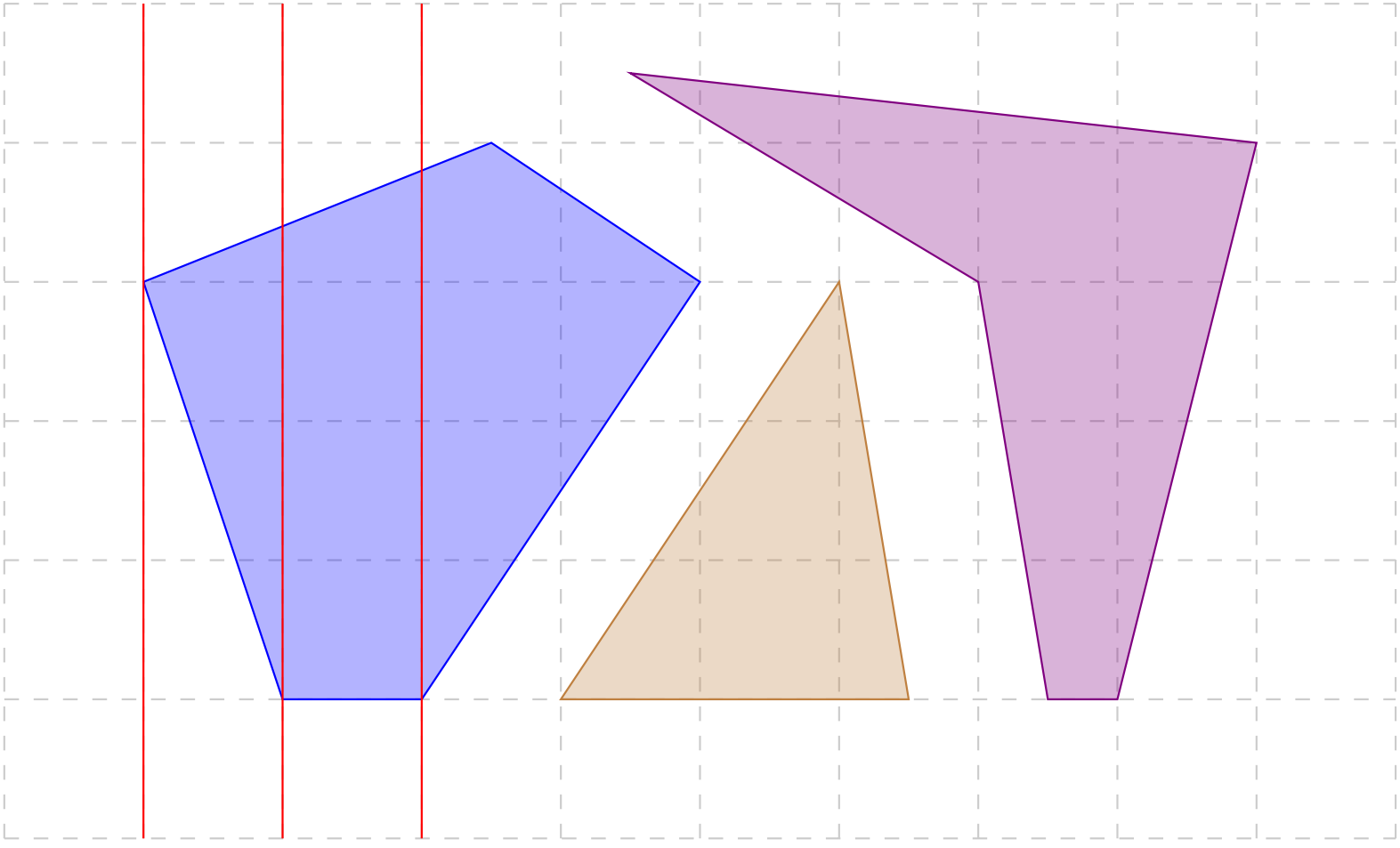
# Linha de varredura

Pontos eventos: vértices dos polígonos



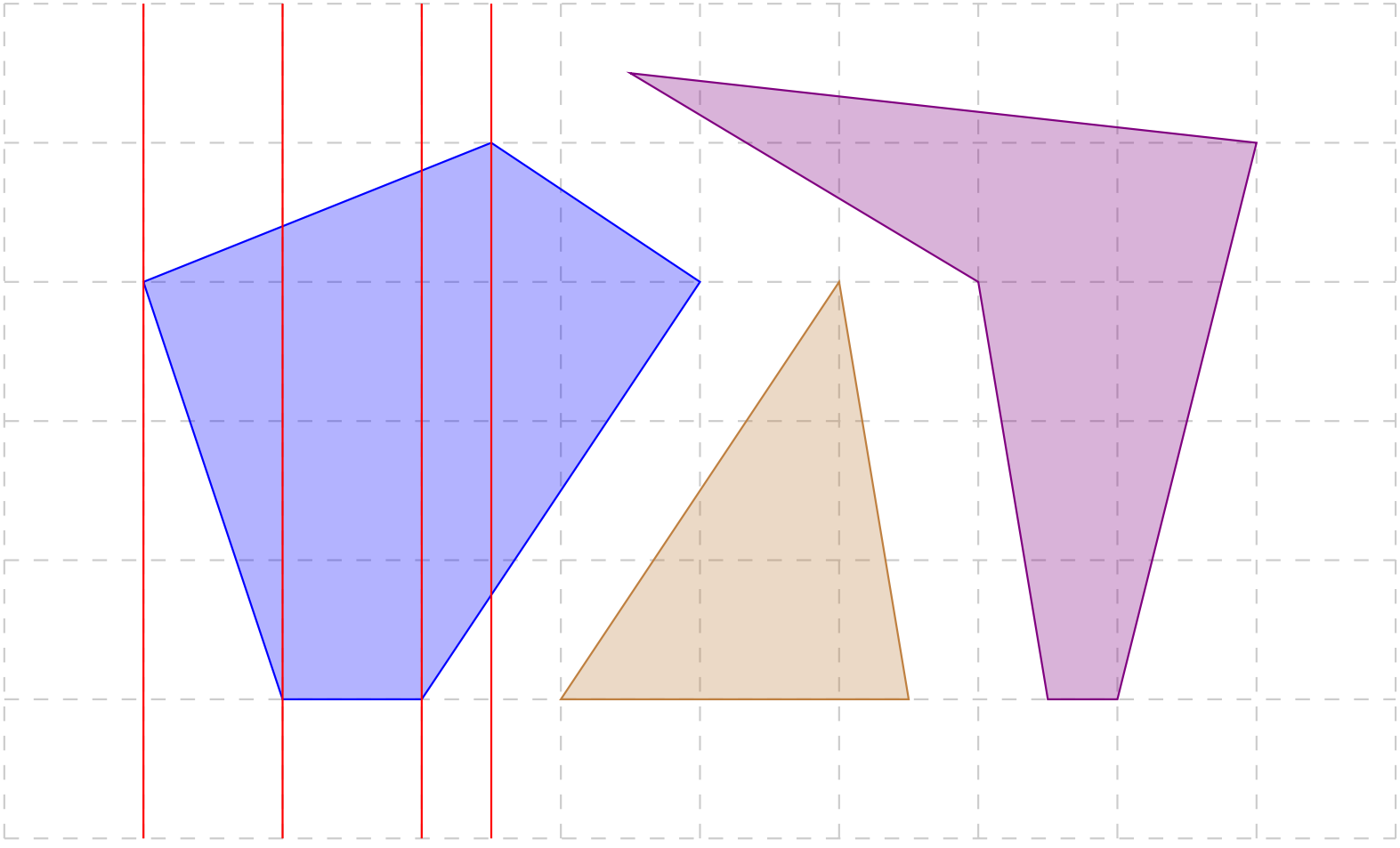
# Linha de varredura

Pontos eventos: vértices dos polígonos



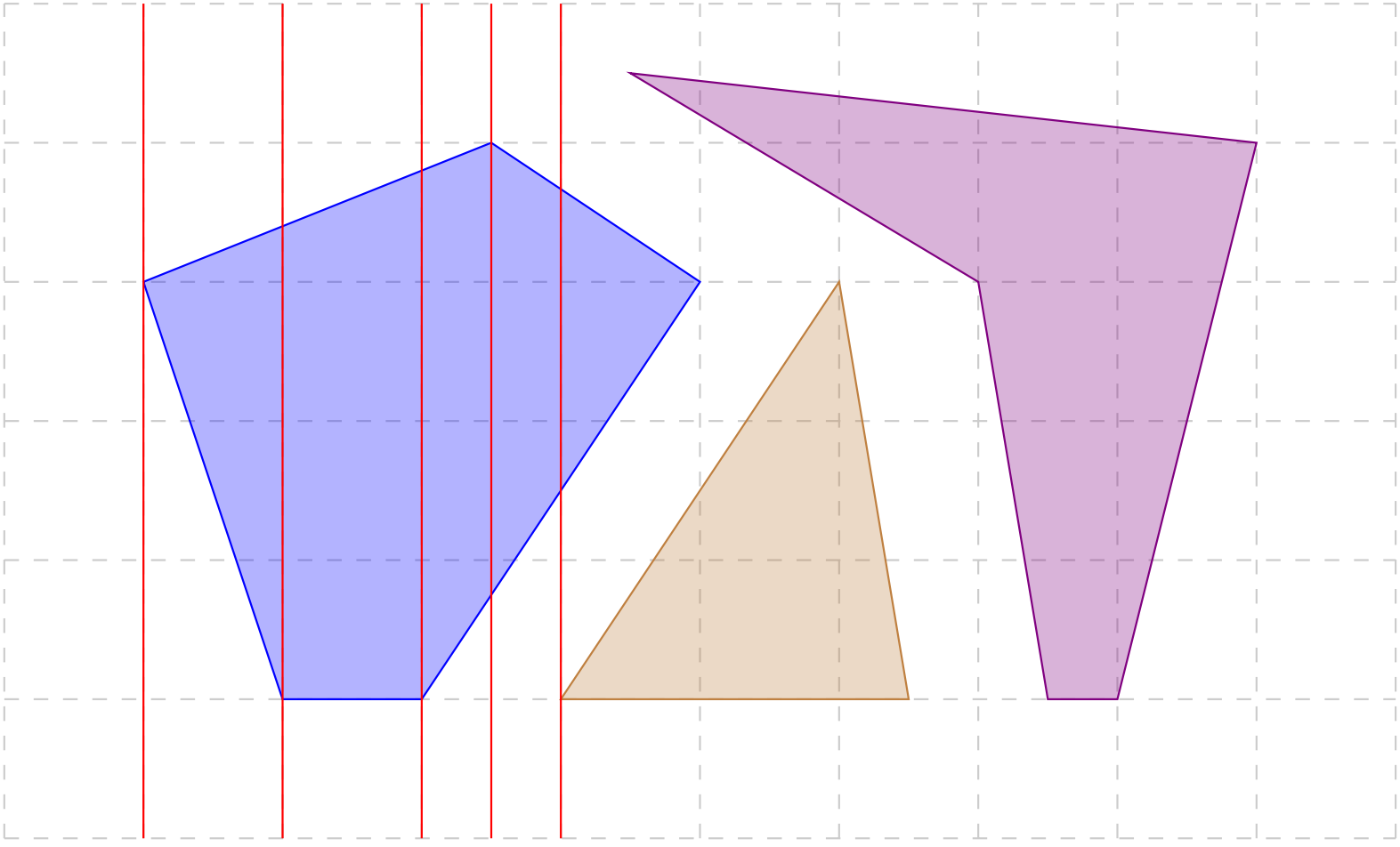
# Linha de varredura

Pontos eventos: vértices dos polígonos



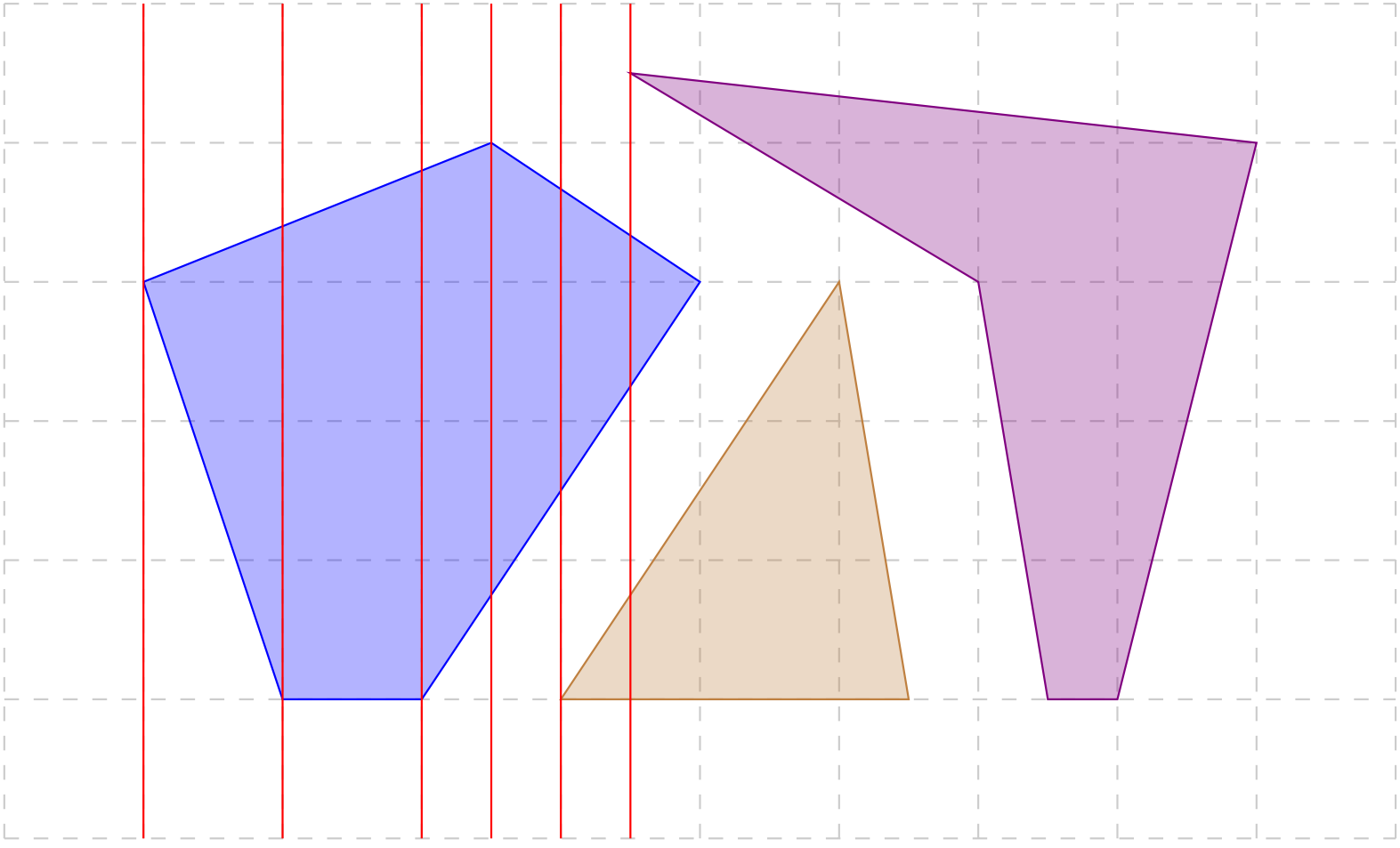
# Linha de varredura

Pontos eventos: vértices dos polígonos



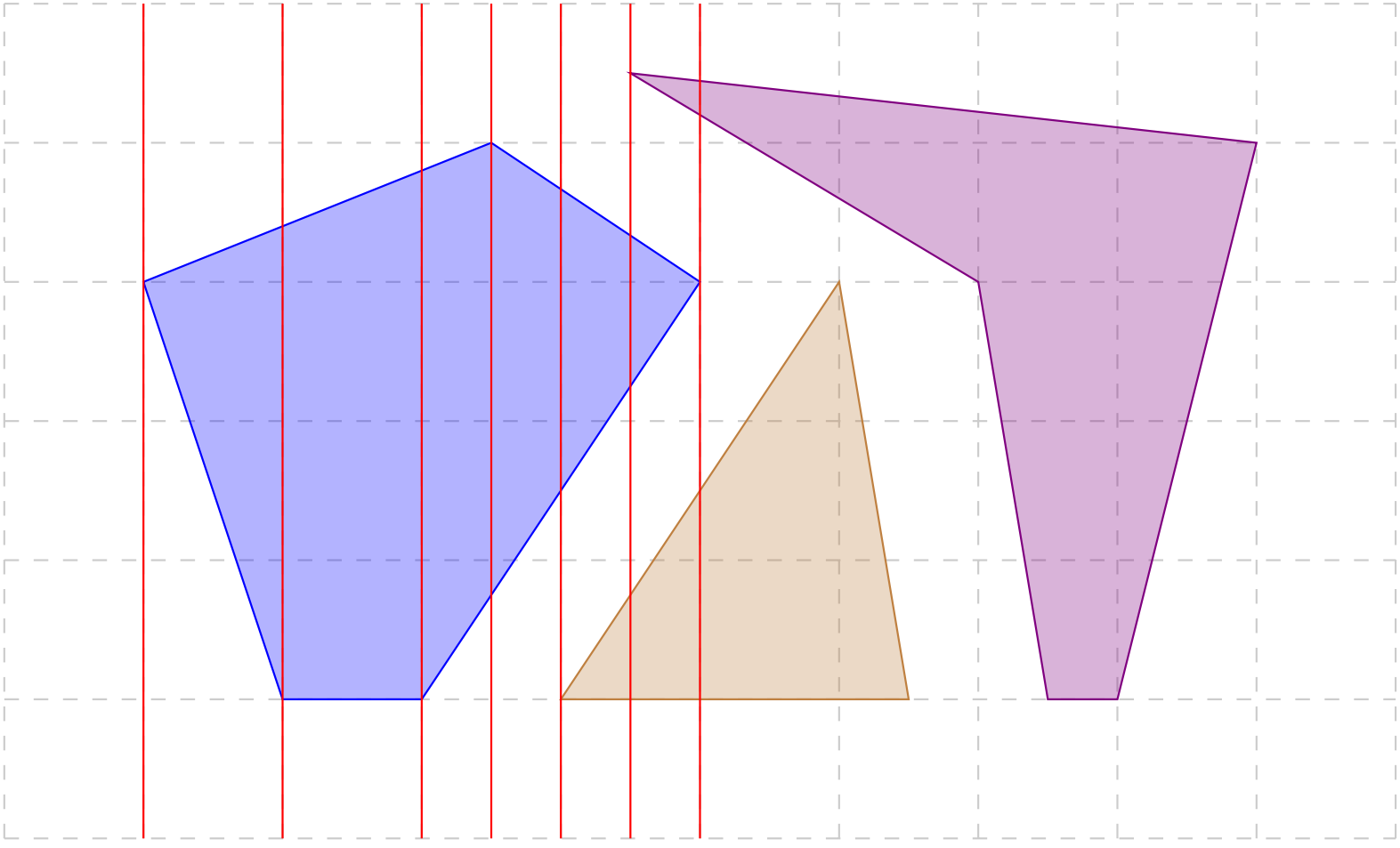
# Linha de varredura

Pontos eventos: vértices dos polígonos



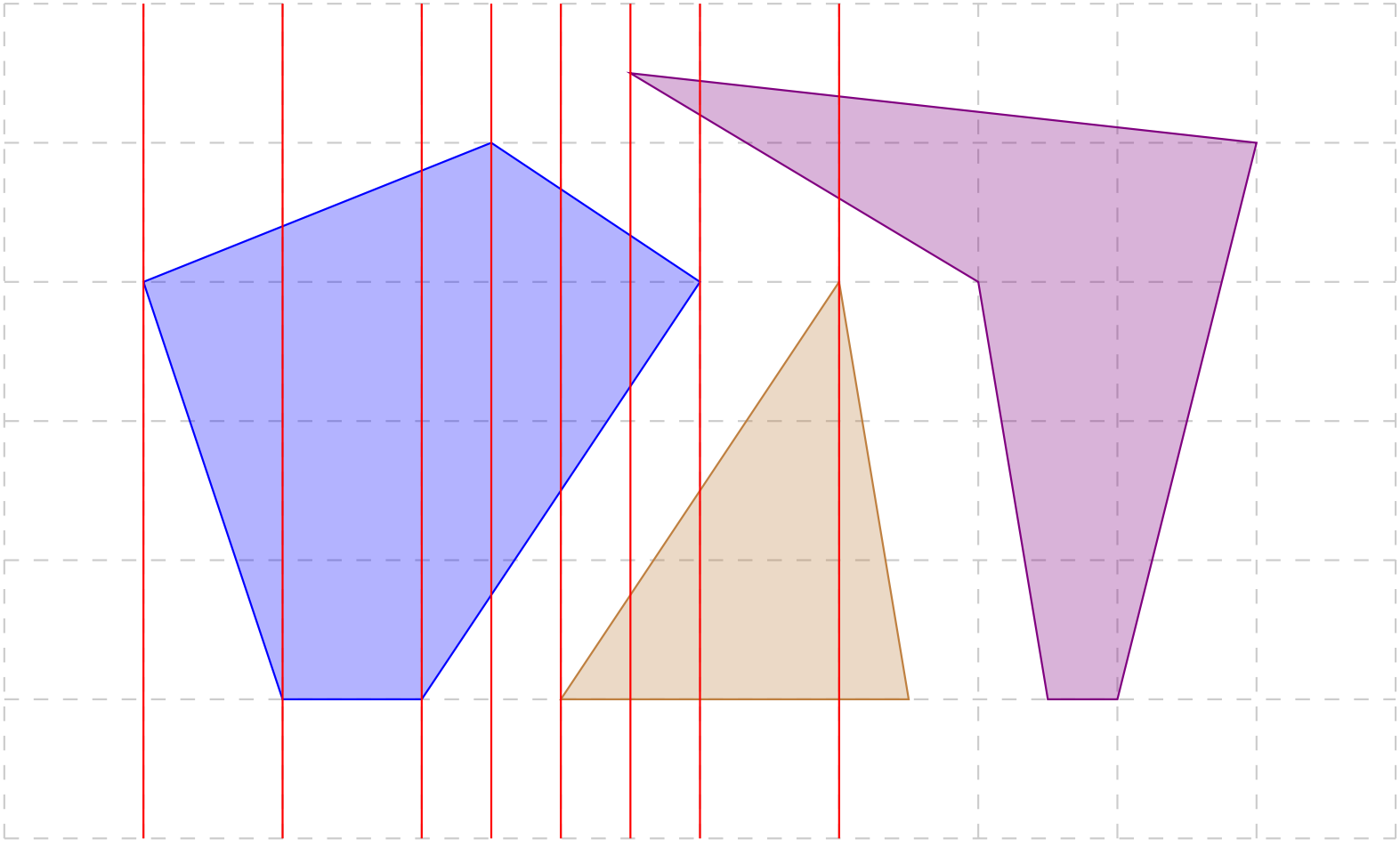
# Linha de varredura

Pontos eventos: vértices dos polígonos



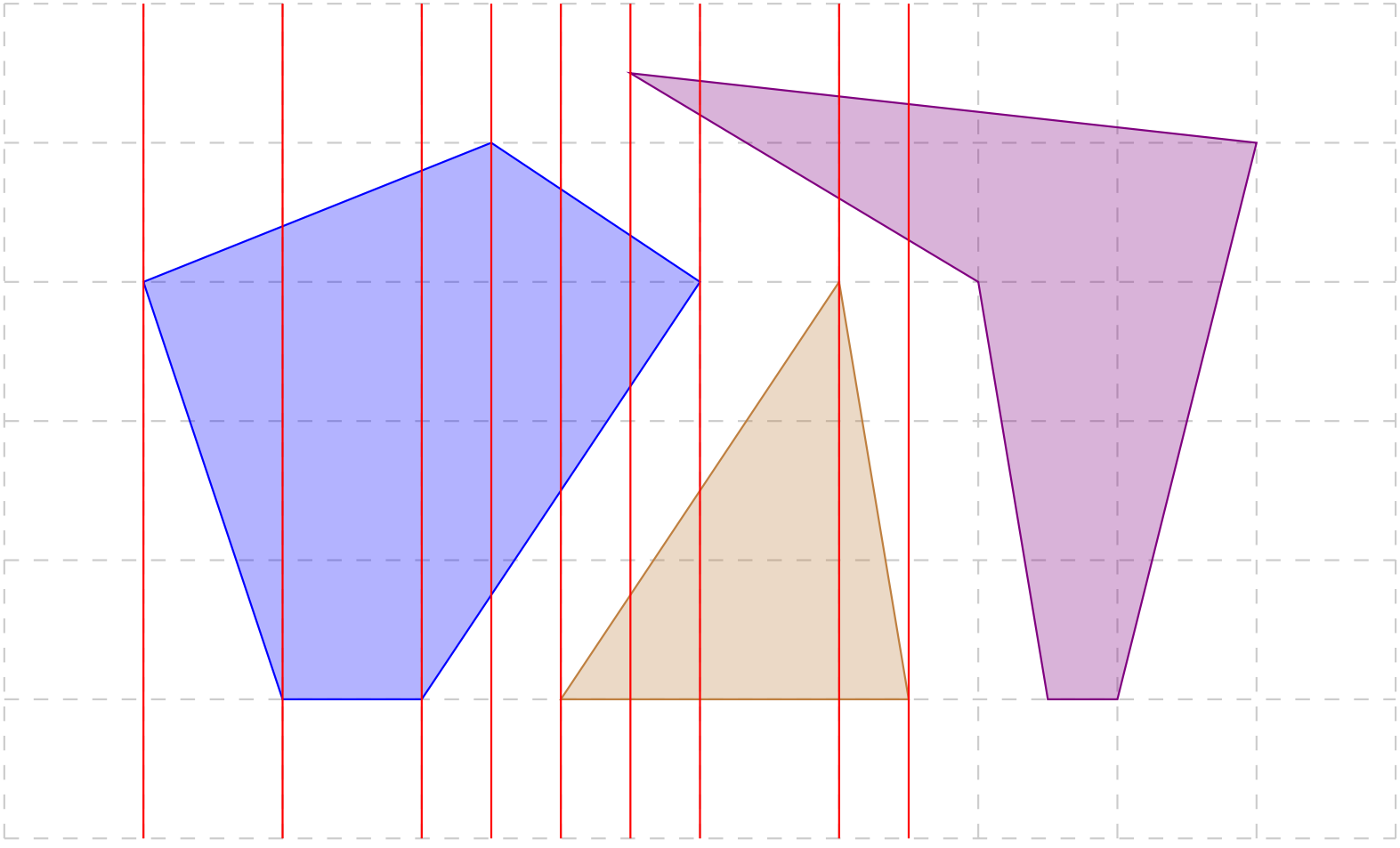
# Linha de varredura

Pontos eventos: vértices dos polígonos



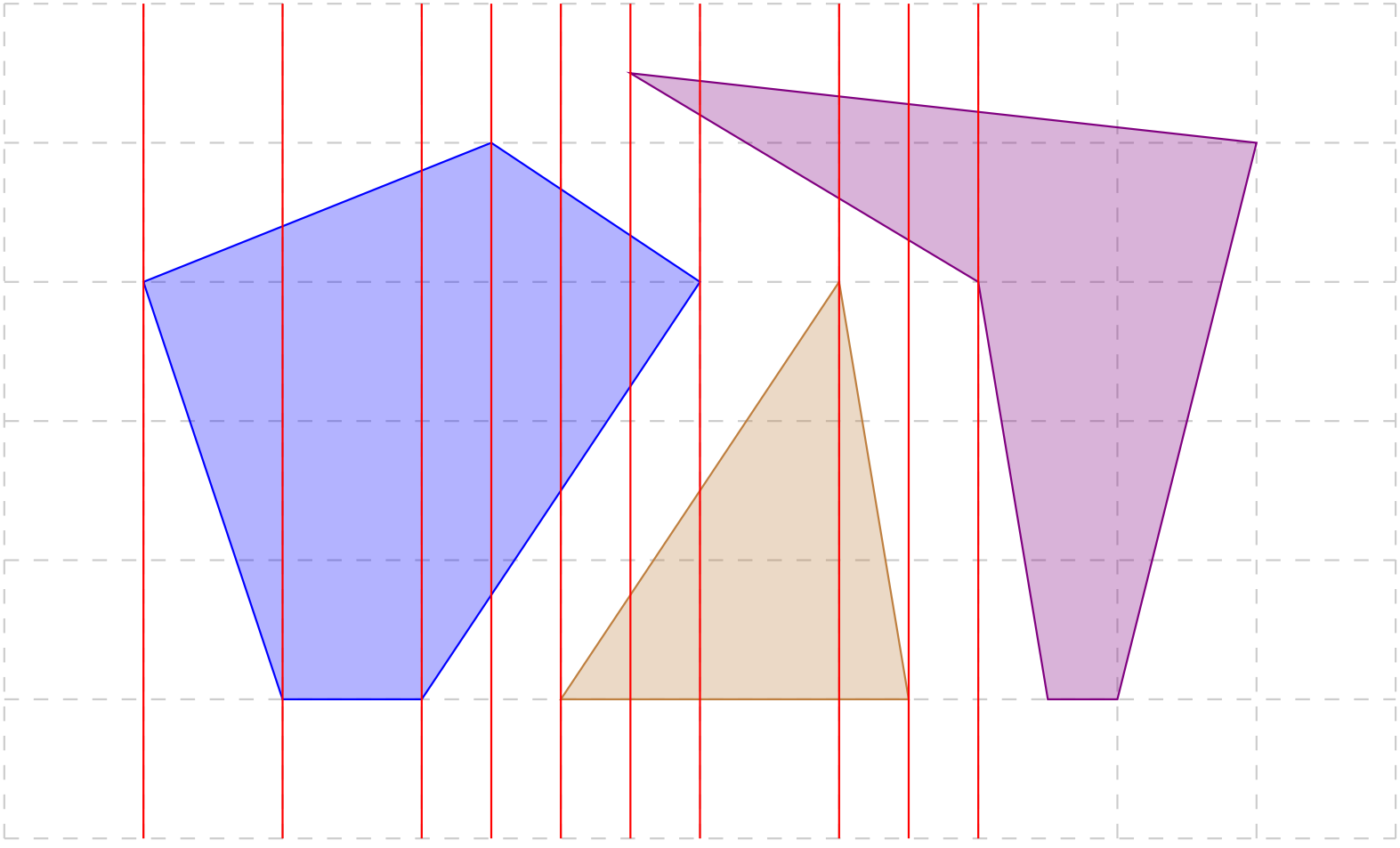
# Linha de varredura

Pontos eventos: vértices dos polígonos



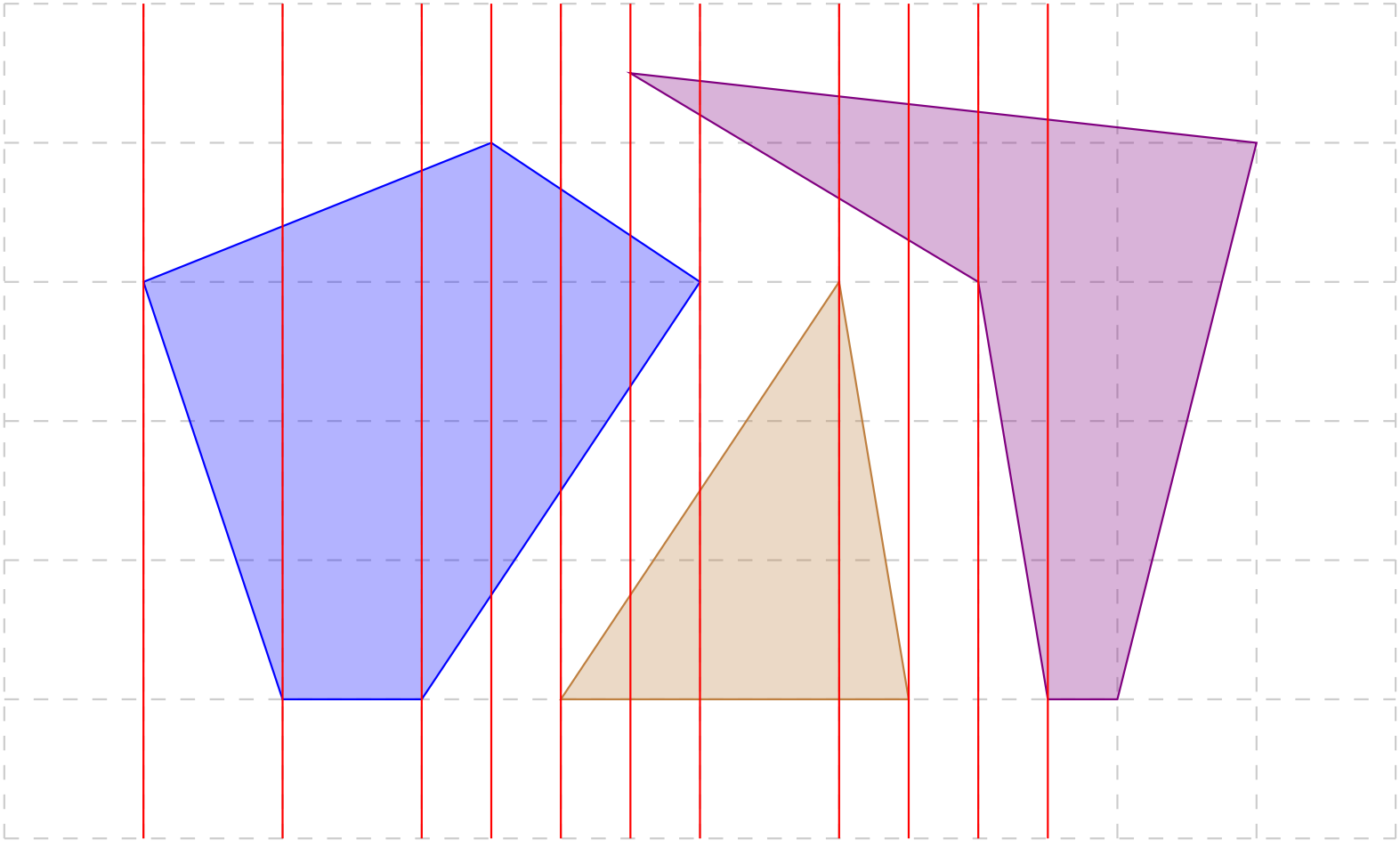
# Linha de varredura

Pontos eventos: vértices dos polígonos



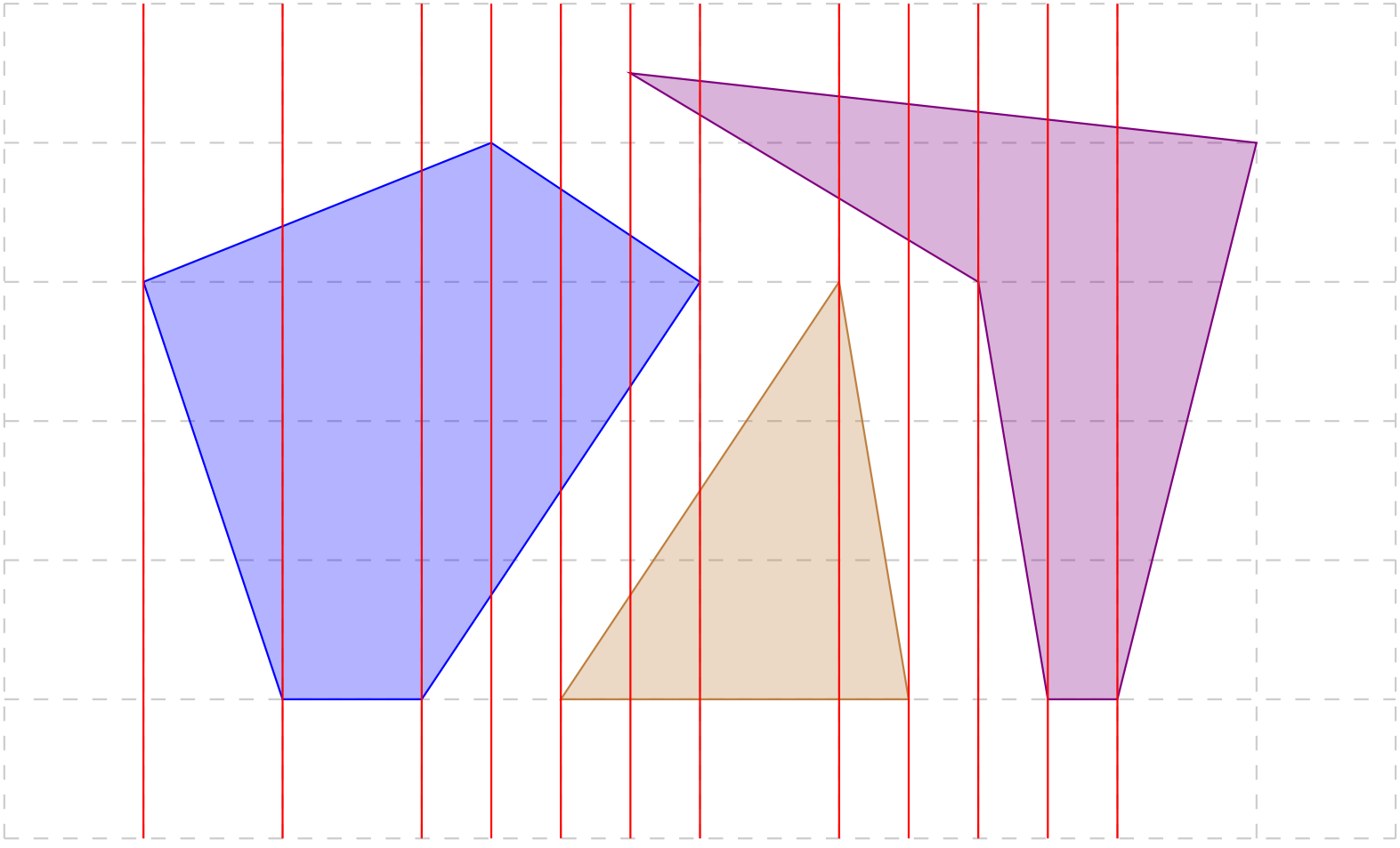
# Linha de varredura

Pontos eventos: vértices dos polígonos



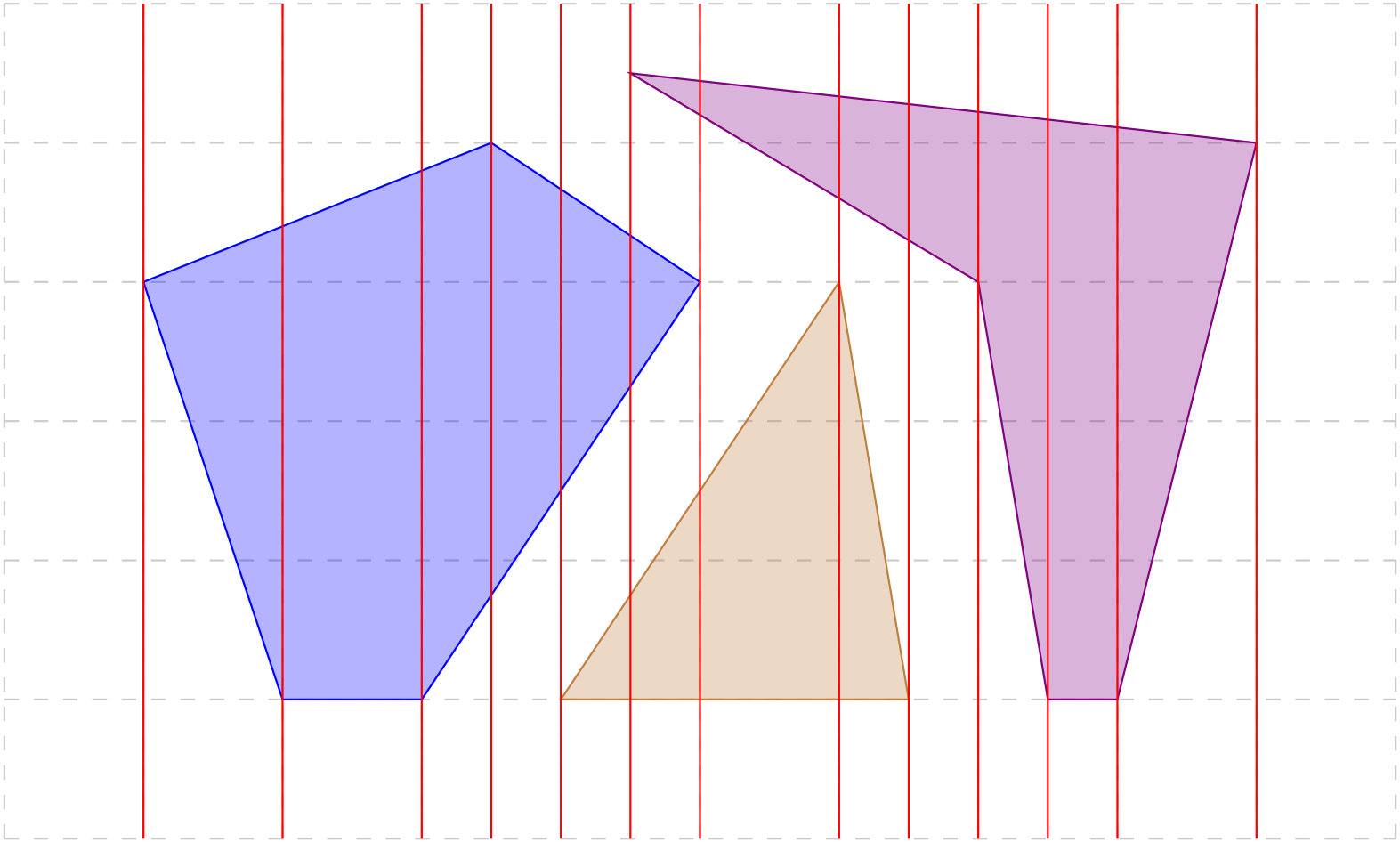
# Linha de varredura

Pontos eventos: vértices dos polígonos



# Linha de varredura

Pontos eventos: vértices dos polígonos



# Como determinar que arestas estão em cada faixa?

Linha de varredura: processaremos um a um cada evento,
ou seja, cada vértice dos polígonos, da esquerda para a direita.

# Como determinar que arestas estão em cada faixa?

Linha de varredura: processaremos um a um cada evento,
ou seja, cada vértice dos polígonos, da esquerda para a direita.

Ao processar um vértice,
arestas podem "entrar na faixa" e arestas podem "sair da faixa".

# Como determinar que arestas estão em cada faixa?

Linha de varredura: processaremos um a um cada evento,
ou seja, cada vértice dos polígonos, da esquerda para a direita.

Ao processar um vértice,
arestas podem "entrar na faixa" e arestas podem "sair da faixa".

Ou seja, o conjunto de arestas que atravessa uma faixa
sofre inserções e remoções e deve ser mantido ordenado.

# Como determinar que arestas estão em cada faixa?

Linha de varredura: processaremos um a um cada evento,
ou seja, cada vértice dos polígonos, da esquerda para a direita.

Ao processar um vértice,
arestas podem "entrar na faixa" e arestas podem "sair da faixa".

Ou seja, o conjunto de arestas que atravessa uma faixa
sofre inserções e remoções e deve ser mantido ordenado.

Que estrutura de dados usar para manter as arestas de uma faixa?

# Como determinar que arestas estão em cada faixa?

Linha de varredura: processaremos um a um cada evento,
ou seja, cada vértice dos polígonos, da esquerda para a direita.

Ao processar um vértice,
arestas podem "entrar na faixa" e arestas podem "sair da faixa".

Ou seja, o conjunto de arestas que atravessa uma faixa
sofre inserções e remoções e deve ser mantido ordenado.

**Que estrutura de dados usar para manter as arestas de uma faixa?**

Árvore de Busca Binária Balanceada (ABBB)

# Linha de varredura

**Que estrutura de dados usar para manter as arestas de uma faixa?**

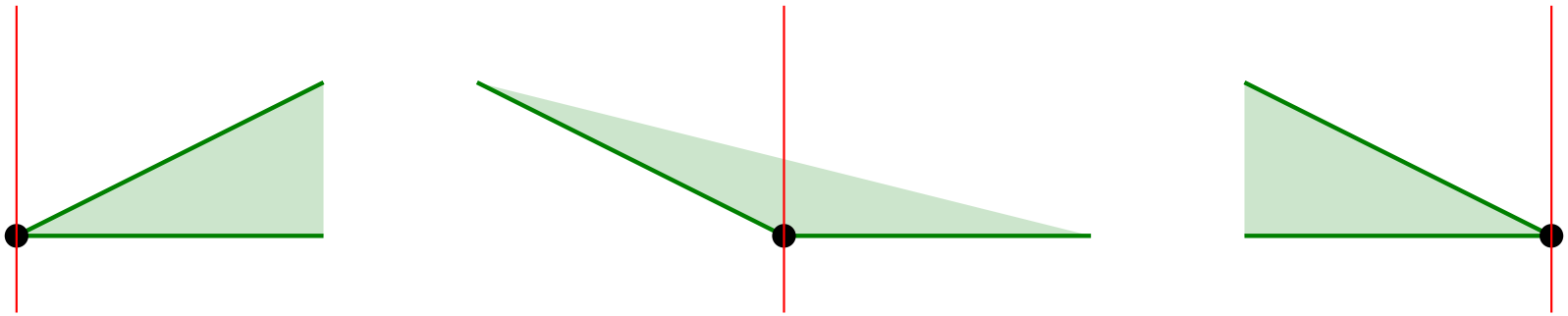
Árvore de Busca Binária Balanceada (ABBB)

# Linha de varredura

**Que estrutura de dados usar para manter as arestas de uma faixa?**

Árvore de Busca Binária Balanceada (ABBB)

Ao processar um vértice, há vários casos:

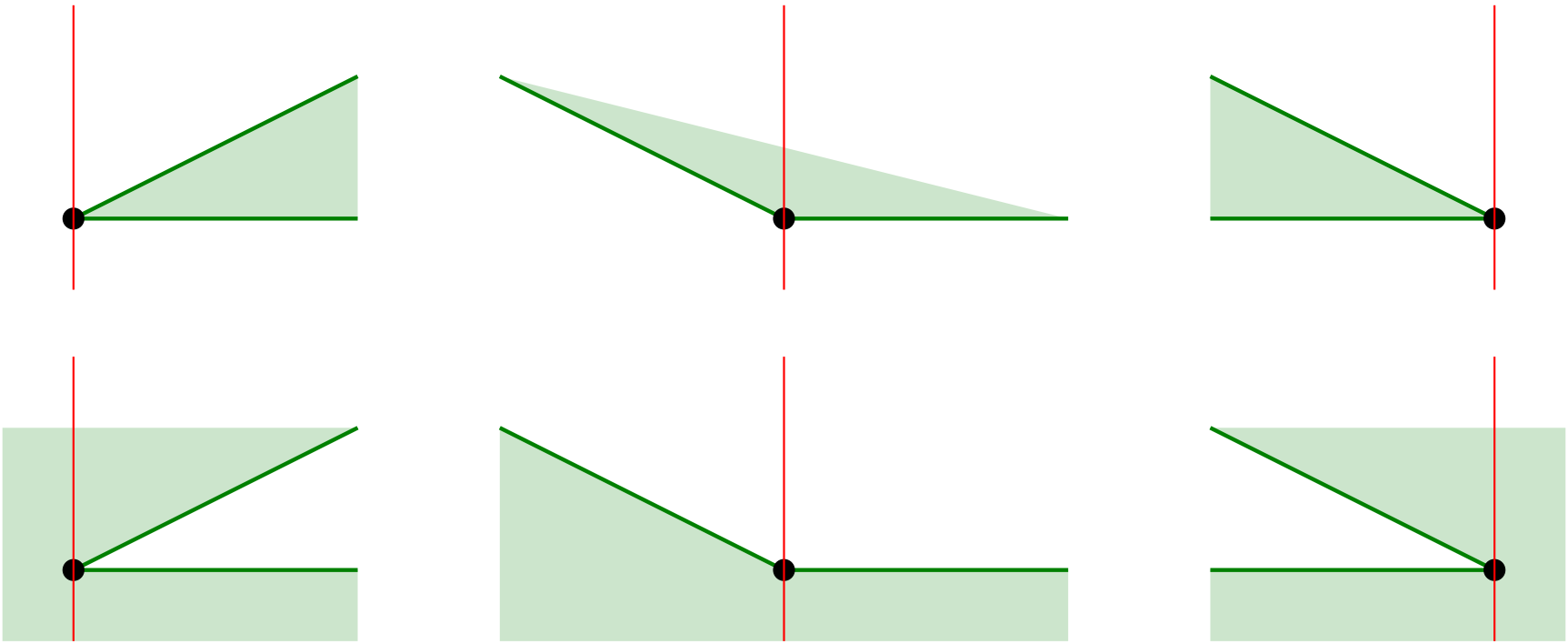


# Linha de varredura

**Que estrutura de dados usar para manter as arestas de uma faixa?**

Árvore de Busca Binária Balanceada (ABBB)

Ao processar um vértice, há vários casos:

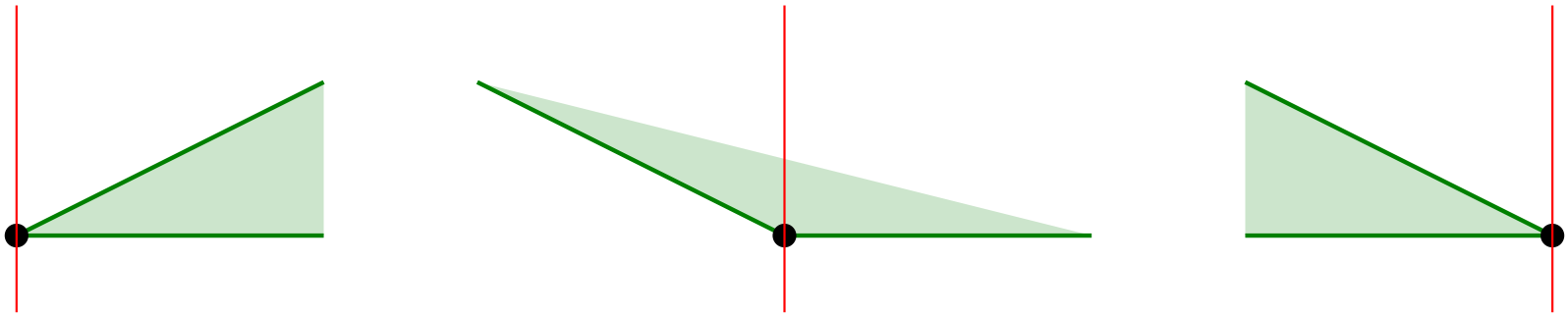


# Linha de varredura

**Que estrutura de dados usar para manter as arestas de uma faixa?**

Árvore de Busca Binária Balanceada (ABBB)

Ao processar um vértice, há vários casos:



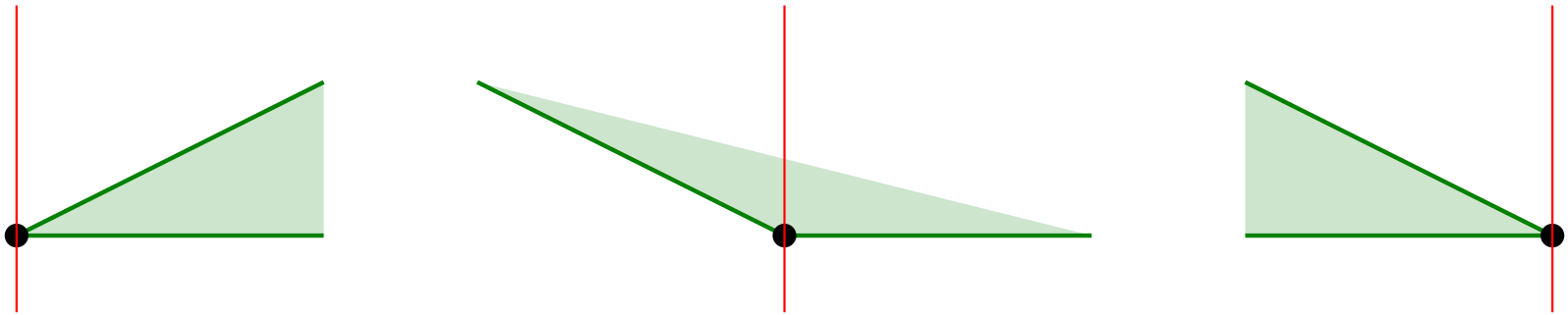
Começamos com a ABBB vazia.

# Linha de varredura

**Que estrutura de dados usar para manter as arestas de uma faixa?**

Árvore de Busca Binária Balanceada (ABBB)

Ao processar um vértice, há vários casos:



Começamos com a ABBB vazia.

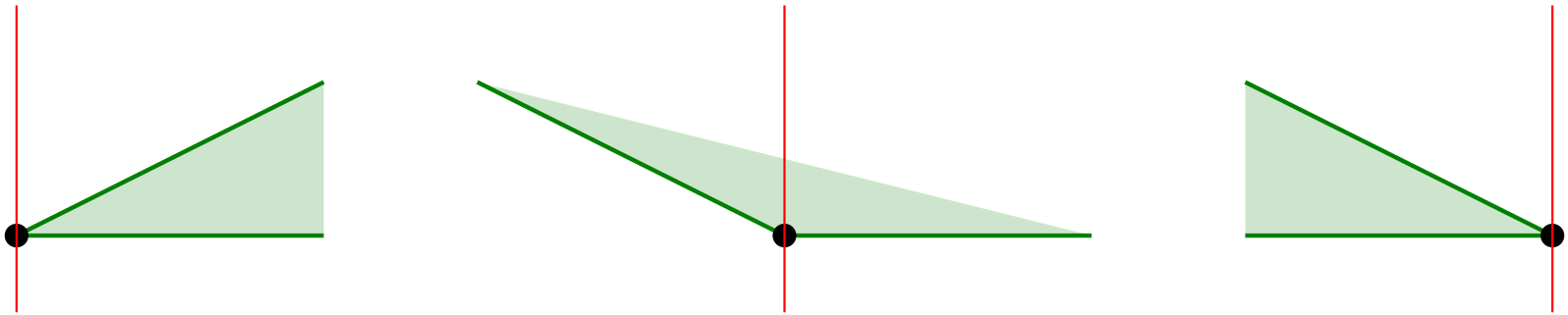
Ao processar um ponto,
o buscamos dentre os segmentos que estão na ABBB

# Linha de varredura

**Que estrutura de dados usar para manter as arestas de uma faixa?**

Árvore de Busca Binária Balanceada (ABBB)

Ao processar um vértice, há vários casos:



Começamos com a ABBB vazia.

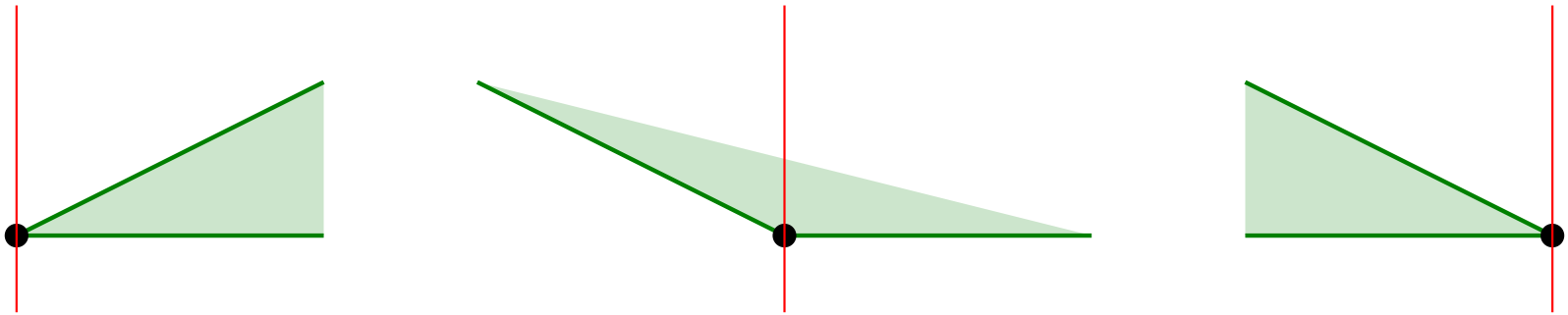
Ao processar um ponto,
o buscamos dentre os segmentos que estão na ABBB
e inserimos e/ou removemos as arestas incidentes ao ponto da ABBB.

# Linha de varredura

**Que estrutura de dados usar para manter as arestas de uma faixa?**

Árvore de Busca Binária Balanceada (ABBB)

Ao processar um vértice, há vários casos:



Começamos com a ABBB vazia.

Ao processar um ponto,
o buscamos dentre os segmentos que estão na ABBB
e inserimos e/ou removemos as arestas incidentes ao ponto da ABBB.

Isso nos dá a próxima faixa.

# Análise de tempo e espaço

Pré-processamento:

- $\mathrm{O}(\lg n)$ por ponto para atualizar a ABBB.

# Análise de tempo e espaço

Pré-processamento:

- $\mathrm{O}(\lg n)$ por ponto para atualizar a ABBB.
- $\mathrm{O}(n)$ por ponto para obter a faixa: teríamos que imprimir todas as arestas da ABBB em ordem depois de processar o ponto.

# Análise de tempo e espaço

Pré-processamento:

- $\mathrm{O}(\lg n)$ por ponto para atualizar a ABBB.
- $\mathrm{O}(n)$ por ponto para obter a faixa: teríamos que imprimir todas as arestas da ABBB em ordem depois de processar o ponto.
- Tempo e espaço total: $\mathrm{O}(n^2)$.

# Análise de tempo e espaço

Pré-processamento:

- $\mathrm{O}(\lg n)$ por ponto para atualizar a ABBB.
- $\mathrm{O}(n)$ por ponto para obter a faixa: teríamos que imprimir todas as arestas da ABBB em ordem depois de processar o ponto.
- Tempo e espaço total: $\mathrm{O}(n^2)$.

Consulta: $\mathrm{O}(\lg n)$ (uma busca binária e uma na ABBB)

# Análise de tempo e espaço

Pré-processamento:

- $O(\lg n)$ por ponto para atualizar a ABBB.
- $O(n)$ por ponto para obter a faixa: teríamos que imprimir todas as arestas da ABBB em ordem depois de processar o ponto.
- Tempo e espaço total: $O(n^2)$.

Consulta: $O(\lg n)$ (uma busca binária e uma na ABBB)

E se usarmos uma ABBB (parcialmente) persistente?

# Análise de tempo e espaço

Pré-processamento:

- $O(\lg n)$ por ponto para atualizar a ABBB.
- $O(n)$ por ponto para obter a faixa: teríamos que imprimir todas as arestas da ABBB em ordem depois de processar o ponto.
- Tempo e espaço total: $O(n^2)$.

Consulta: $O(\lg n)$ (uma busca binária e uma na ABBB)

E se usarmos uma ABBB (parcialmente) persistente?

Consulta:

Não precisa imprimir as faixas! **Basta fazer uma consulta no passado!**

# Análise de tempo e espaço

Pré-processamento:

- $O(\lg n)$ por ponto para atualizar a ABBB.
- $O(n)$ por ponto para obter a faixa: teríamos que imprimir todas as arestas da ABBB em ordem depois de processar o ponto.
- Tempo e espaço total: $O(n^2)$.

Consulta: $O(\lg n)$ (uma busca binária e uma na ABBB)

E se usarmos uma ABBB (parcialmente) persistente?

Consulta: $O(\lg n)$

Não precisa imprimir as faixas! **Basta fazer uma consulta no passado!**

# Análise de tempo e espaço

Pré-processamento:

- $O(\lg n)$ por ponto para atualizar a ABBB.
- $O(n)$ por ponto para obter a faixa: teríamos que imprimir todas as arestas da ABBB em ordem depois de processar o ponto.
- Tempo e espaço total: $O(n^2)$.

Consulta: $O(\lg n)$ (uma busca binária e uma na ABBB)

E se usarmos uma ABBB (parcialmente) persistente?

Consulta: $O(\lg n)$
Não precisa imprimir as faixas! **Basta fazer uma consulta no passado!**

Pré-processamento:

- $O(\lg n)$ por ponto para atualizar a ABBB.

# Análise de tempo e espaço

Pré-processamento:

- $O(\lg n)$ por ponto para atualizar a ABBB.
- $O(n)$ por ponto para obter a faixa: teríamos que imprimir todas as arestas da ABBB em ordem depois de processar o ponto.
- Tempo e espaço total: $O(n^2)$.

Consulta: $O(\lg n)$ (uma busca binária e uma na ABBB)

E se usarmos uma ABBB (parcialmente) persistente?

Consulta: $O(\lg n)$

Não precisa imprimir as faixas! **Basta fazer uma consulta no passado!**

Pré-processamento:

- $O(\lg n)$ por ponto para atualizar a ABBB.
- Tempo e espaço total: $O(n \lg n)$.

# Algumas referências

Sobre persistência:

Dissertação de Mestrado no IME-USP de Yan Soares Couto

`https://yancouto.github.io/mestrado/thesis.pdf`

Sobre retroatividade:

Artigo *Retroactive Data Structures*, por Demaine, Iacono e Langerman

`https://erikdemaine.org/papers/Retroactive_TALG/paper.pdf`

(Em novembro, teremos um texto em português sobre o assunto, por Beatriz Figueiredo Marouelli, seu trabalho de formatura.)

Sobre geometria computacional:

Notas de aula do minicurso *Convite à Geometria Computacional*

`https://www.ime.usp.br/~cris/jai2009/`

# Fim da viagem...

# Fim da viagem...

OBRIGADA!

Perguntas???