Universidade do Estado do Amazonas
Escola Superior de Tecnologia
Núcleo de Computação - NUCOMP

Estruturas de Dados

Árvores AVL

Márcia Sampaio Lima

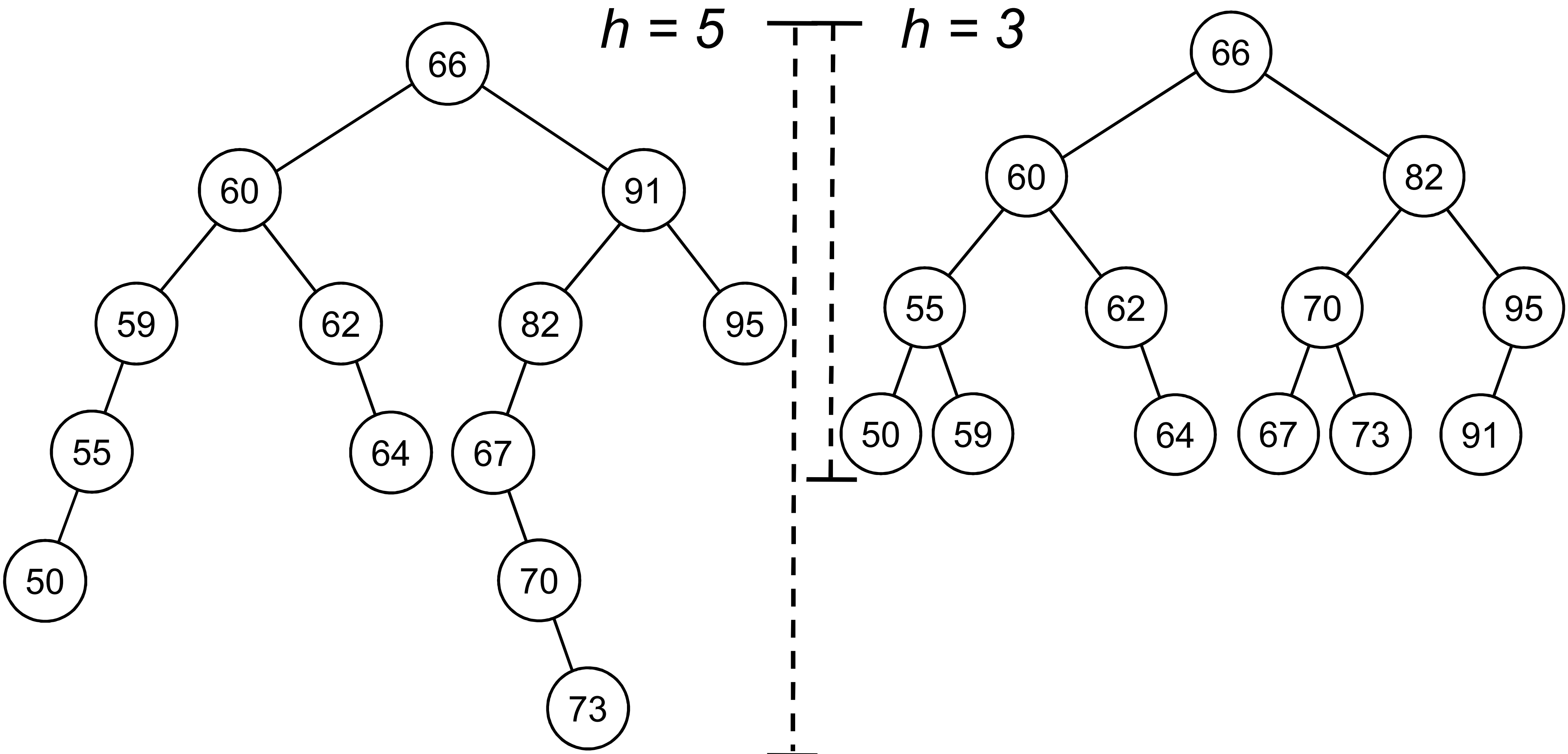
**Adaptação dos slides de
Prof. Flávio José Mendes Coelho**

Problema do balanceamento

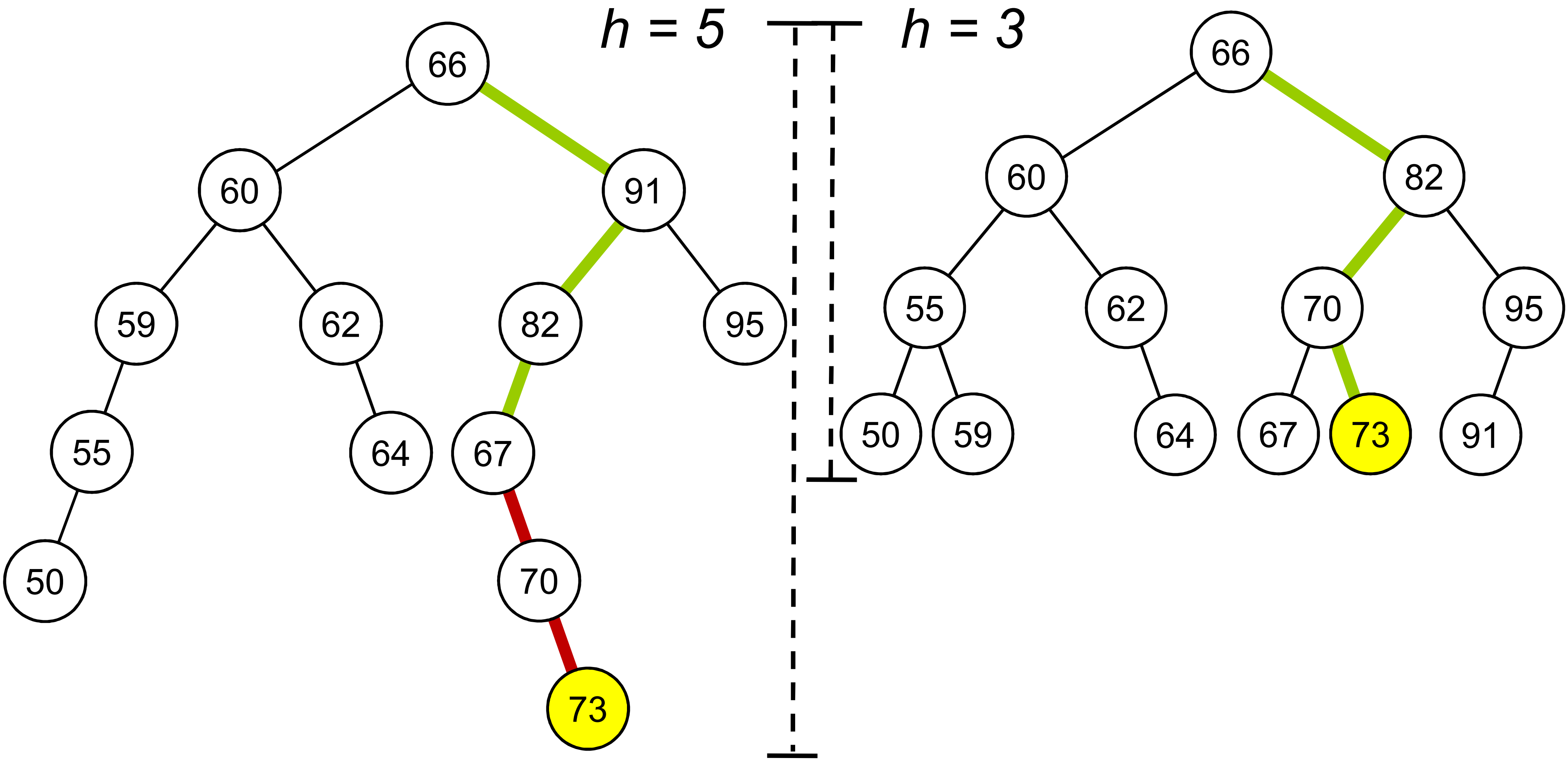
- A eficiência da operação Busca() em uma ABB está relacionada à altura da árvore.

- Quanto menor a altura, melhor o tempo de busca.

Problema do balanceamento



Problema do balanceamento



Problema do balanceamento

- Como rearranjar uma ABB para que sua altura seja a menor possível?

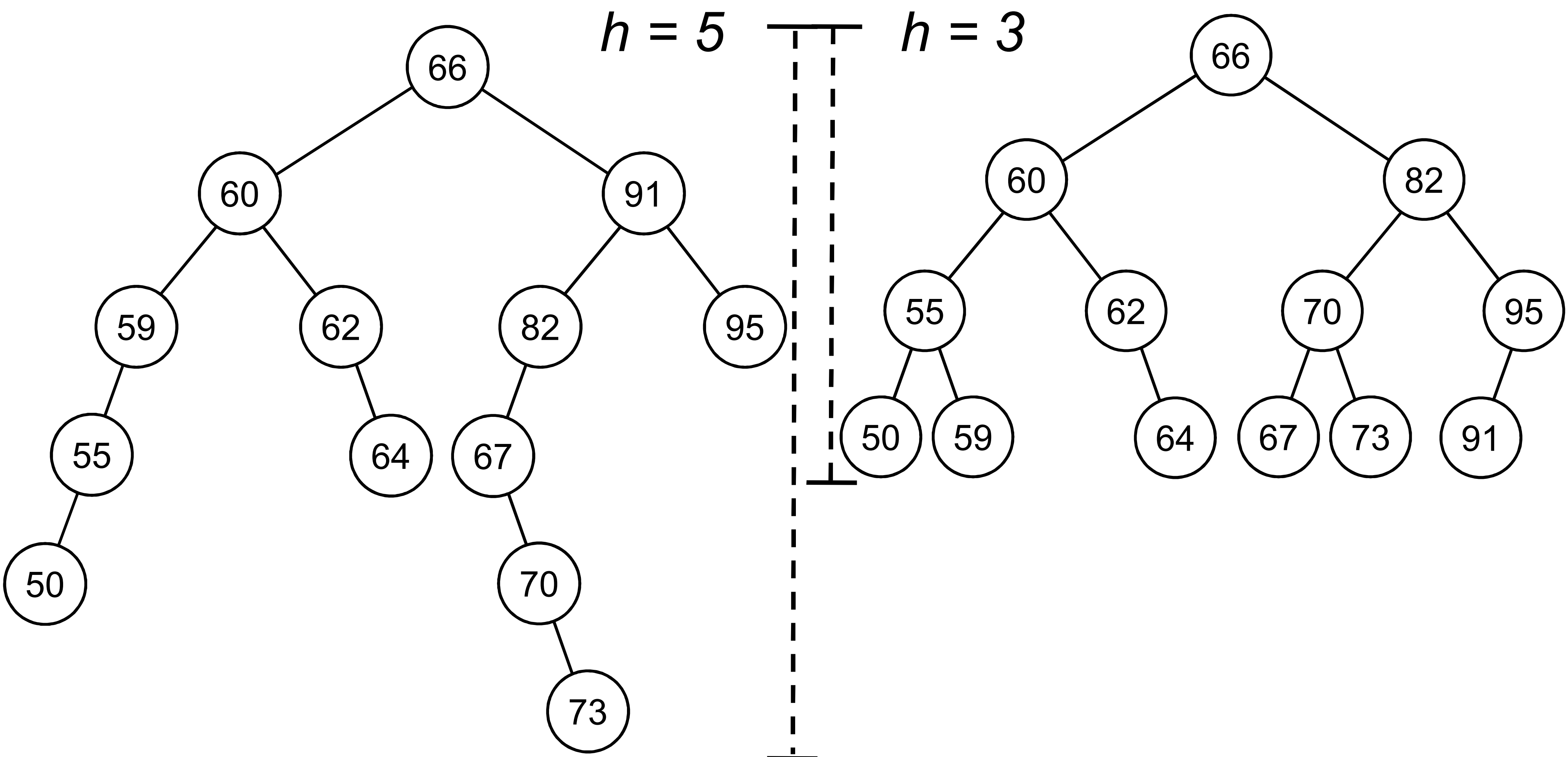
- A solução está em submeter a ABB a um processo de balanceamento.

Problema do balanceamento

- (Def. informal) Uma ABB está balanceada quando seus nós estão distribuídos uniformemente em suas sub-árvores esquerda e direita.

Árvore NÃO balanceada

Árvore balanceada



Problema do balanceamento

- Maior balanceamento:

  Menor altura da árvore.

  Menor tempo de busca por um item.

- Menor balanceamento:

  Maior altura da árvore.

  Maior tempo de busca por um item.

Problema do balanceamento

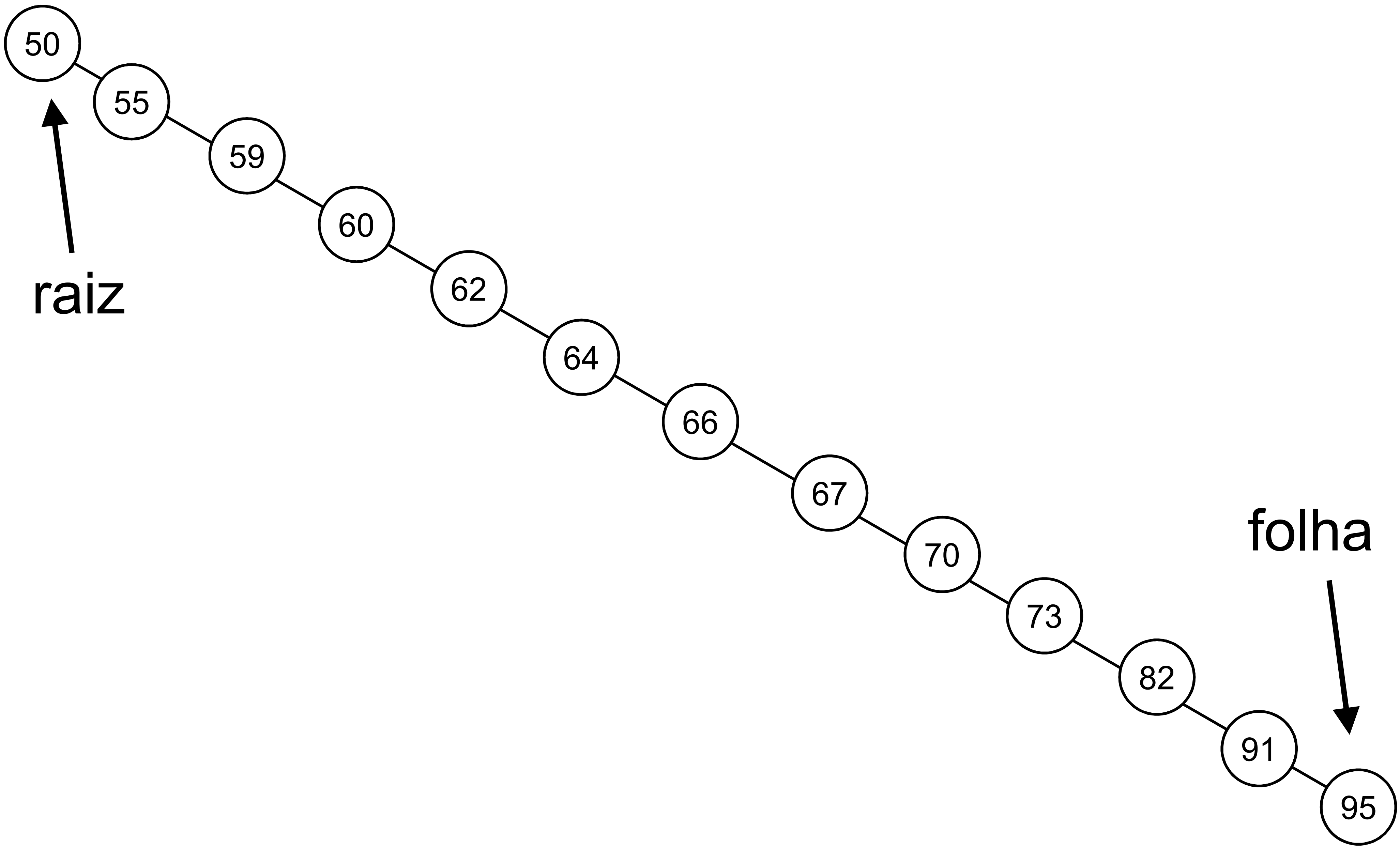
- Em que condições uma ABB torna-se desbalanceada?

- Como balancear e manter uma ABB balanceada?

Problema do balanceamento

- O desbalanceamento de uma ABB é provocado conforme se adiciona ou se remove itens da árvore.

- O pior caso ocorre quando a árvore fica tão desbalanceada que degenera-se em uma lista ligada simples.



Árvore degenerada em uma lista ligada

Problema do balanceamento

- Com a árvore balanceada
  - Tempo médio de busca equivale ao da busca binária ($O(\log_2 n)$).
- No pior caso (“árvore-caminho”)
  - Tempo médio de busca equivale ao da busca sequencial.

Árvores AVL

- Apresentam algoritmos para o problema do balanceamento de ABBs.

- Apresentados por Adel’son-Vel’skii e Landis em:

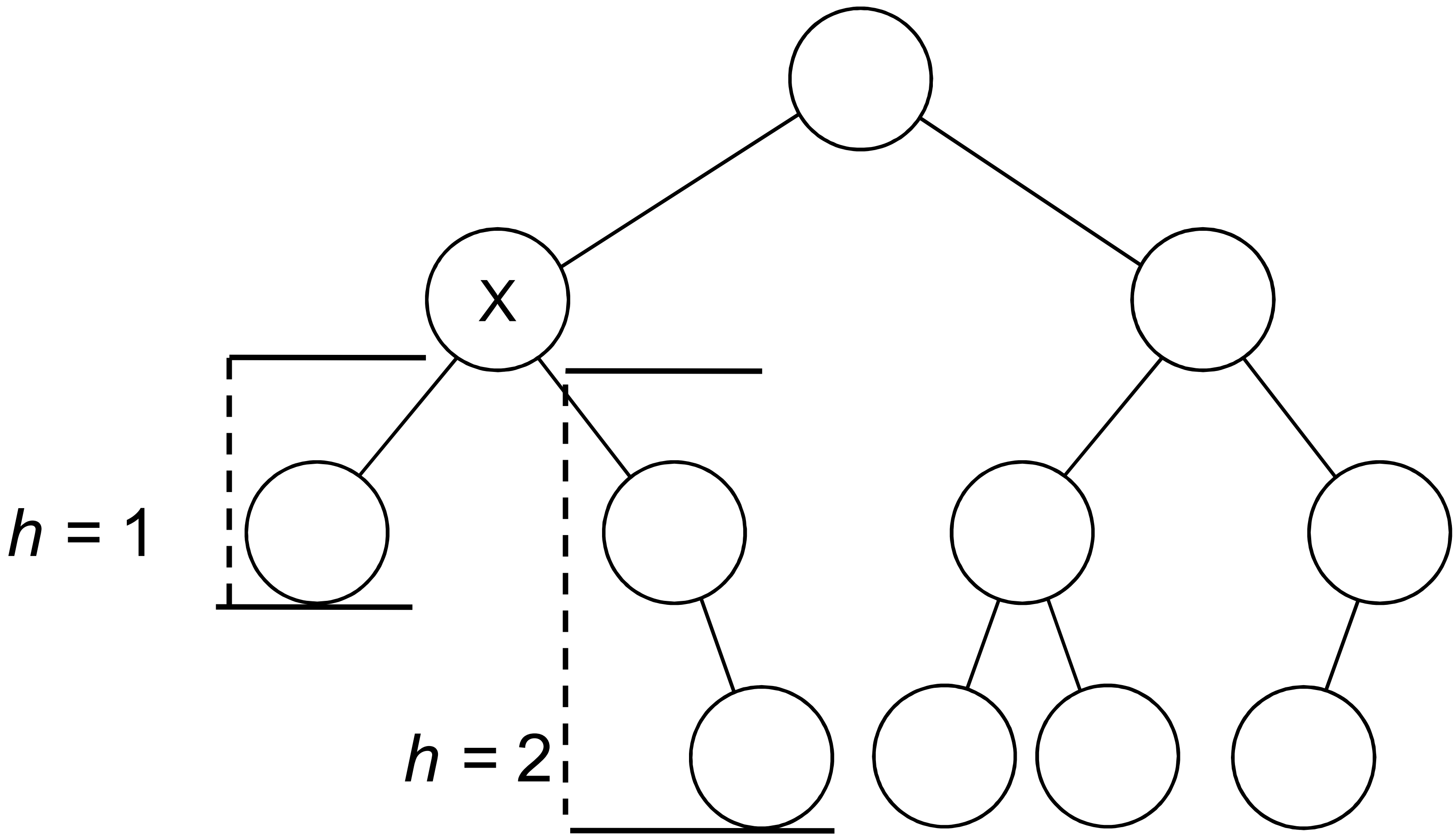
Adel'son-Vel'skii, G. M.; Landis, E. M. (1962), "An algorithm for organization of information", Doklady Akademii Nauk SSSR **146**: 263–266).

Árvores AVL

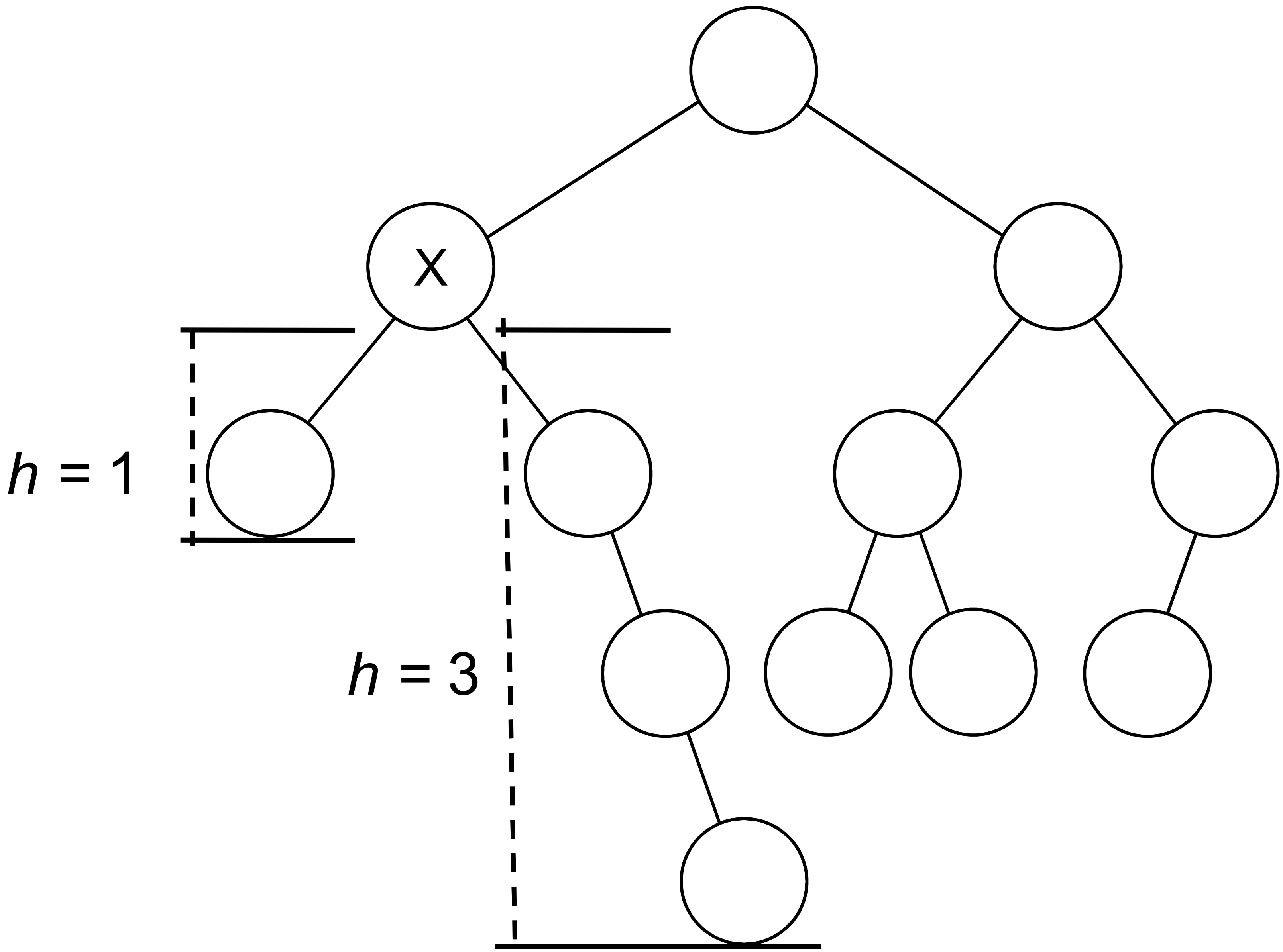
- (Def. informal) Uma árvore balanceada ou AVL é uma ABB *A* com a seguinte propriedade de balanceamento:

> Para cada nó interno *x* de *A* as alturas das sub-árvores esquerda e direita de *x* diferem por no máximo 1.

Árvore AVL



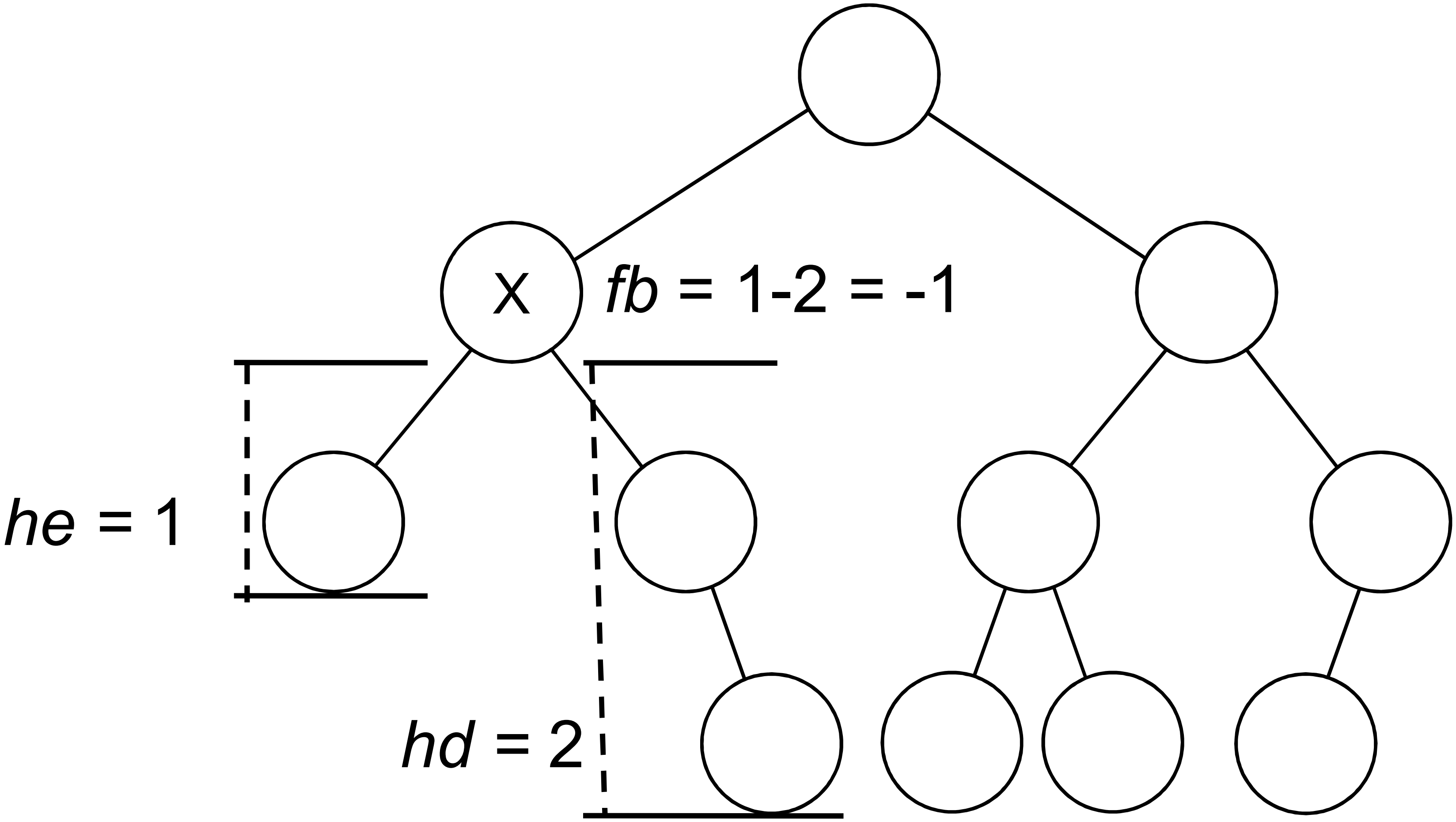
Árvore Não-AVL



Árvores AVL

- Seja *A* uma ABB.
- Seja *x* um nó interno qualquer de *A*.
- Seja *he*(*x*) a **altura da sub-árvore esquerda** de *x.*
- Seja *hd*(*x*) a **altura da sub-árvore direita** de *x.*
- Seja *fb*(*x*) o **fator de balanceamento** de *x*, dado por $fb(x) = he(x) - hd(x)$

Árvore AVL



Árvores AVL

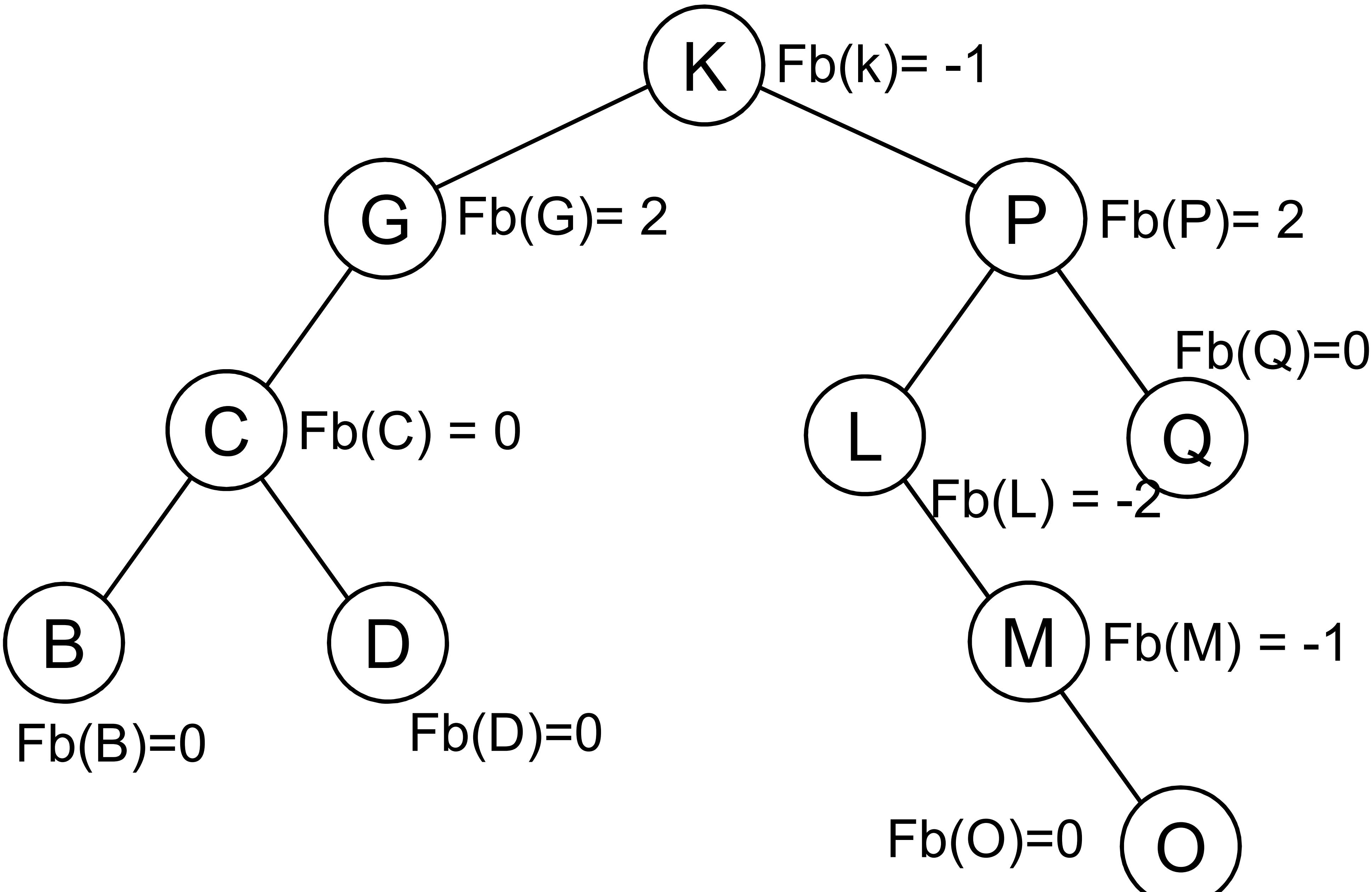
- Uma árvore AVL é uma ABB *A* com a seguinte propriedade de balanceamento:

Para cada nó interno *x* de *A* vale

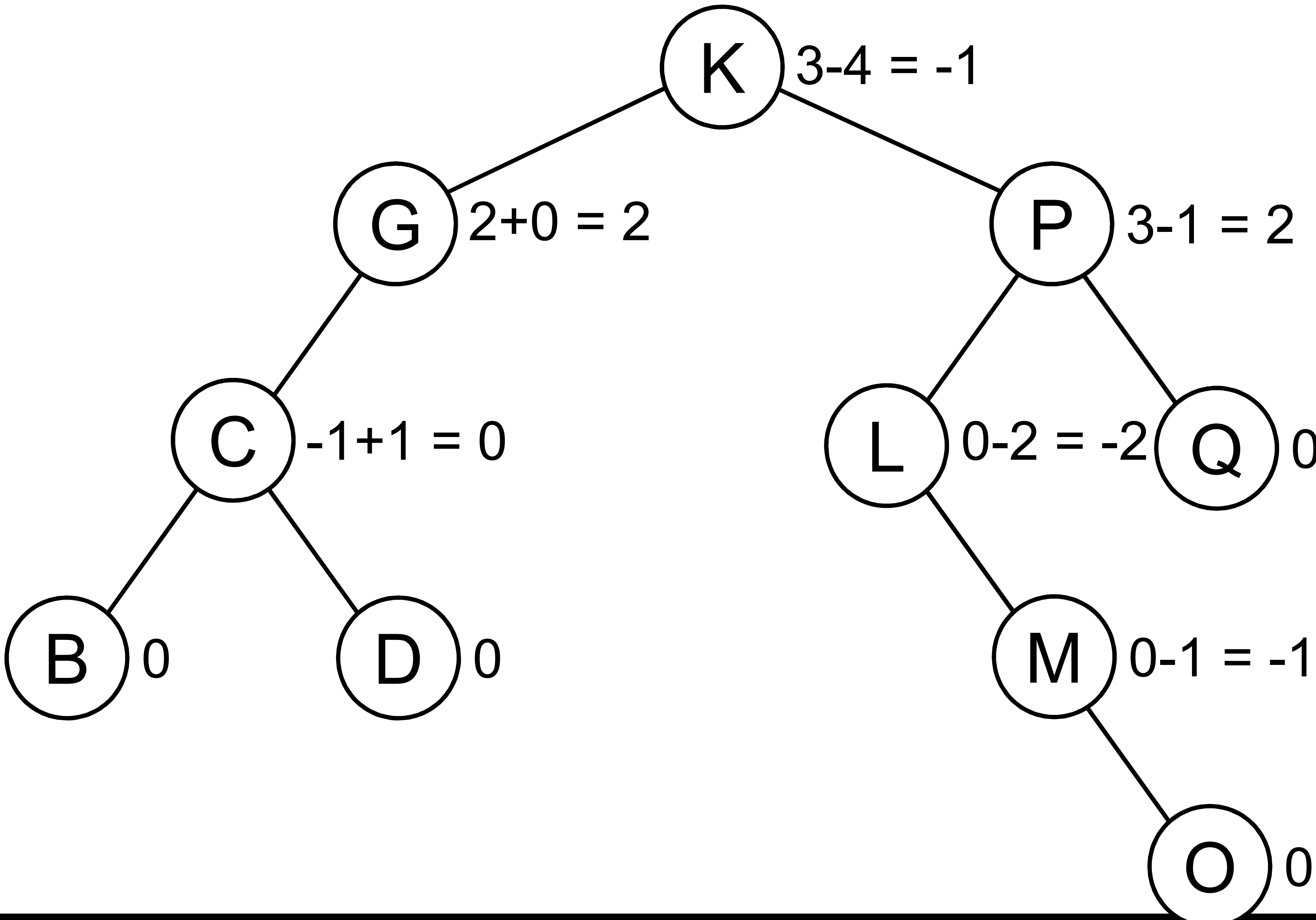
$$-1 \leq fb(x) \leq 1.$$

- Portanto, se $fb(x) < -1$ ou $fb(x) > 1$, então a árvore não é AVL e está desbalanceada.

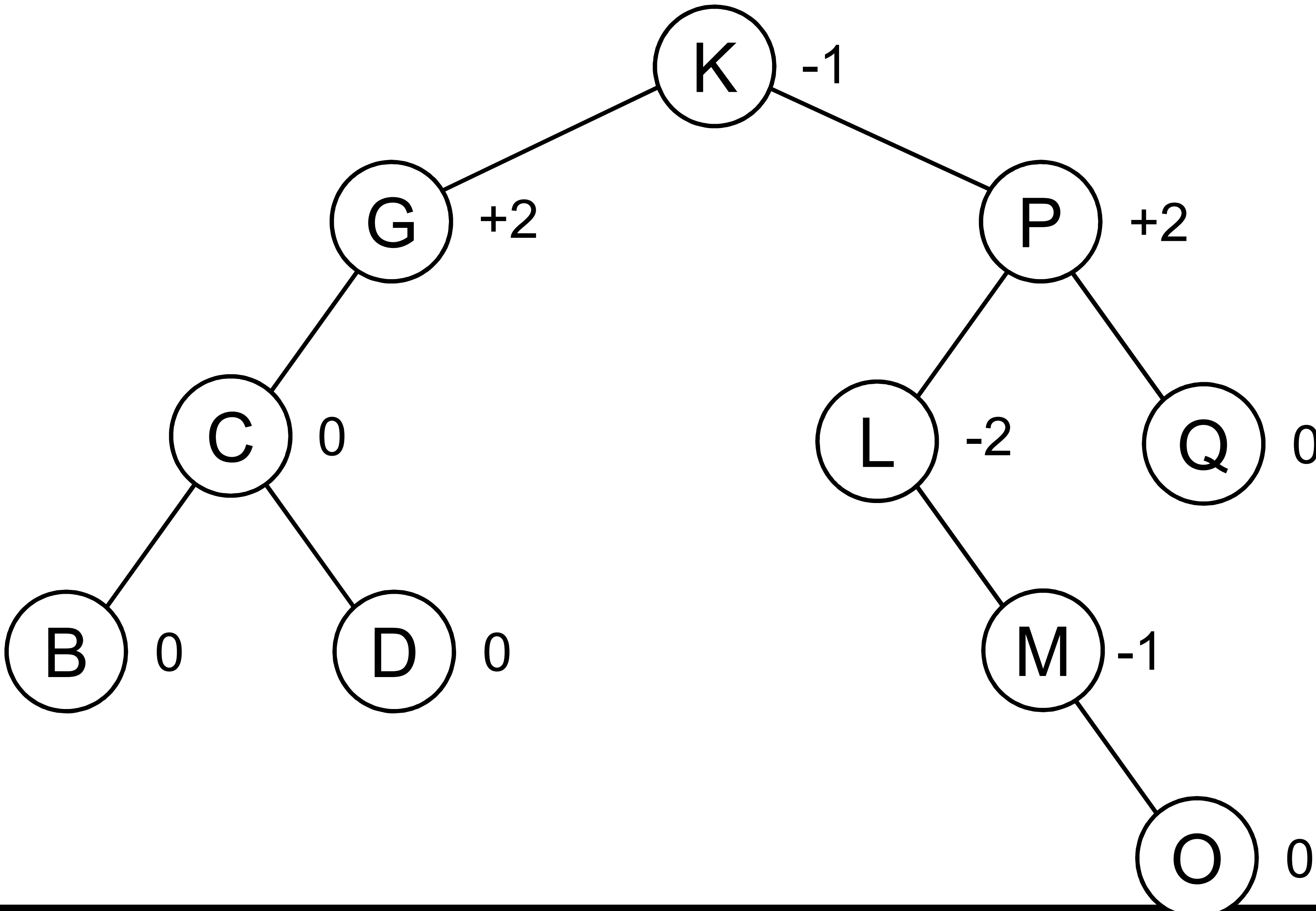
Árvore com fatores de balanceamento



Árvore com fatores de balanceamento



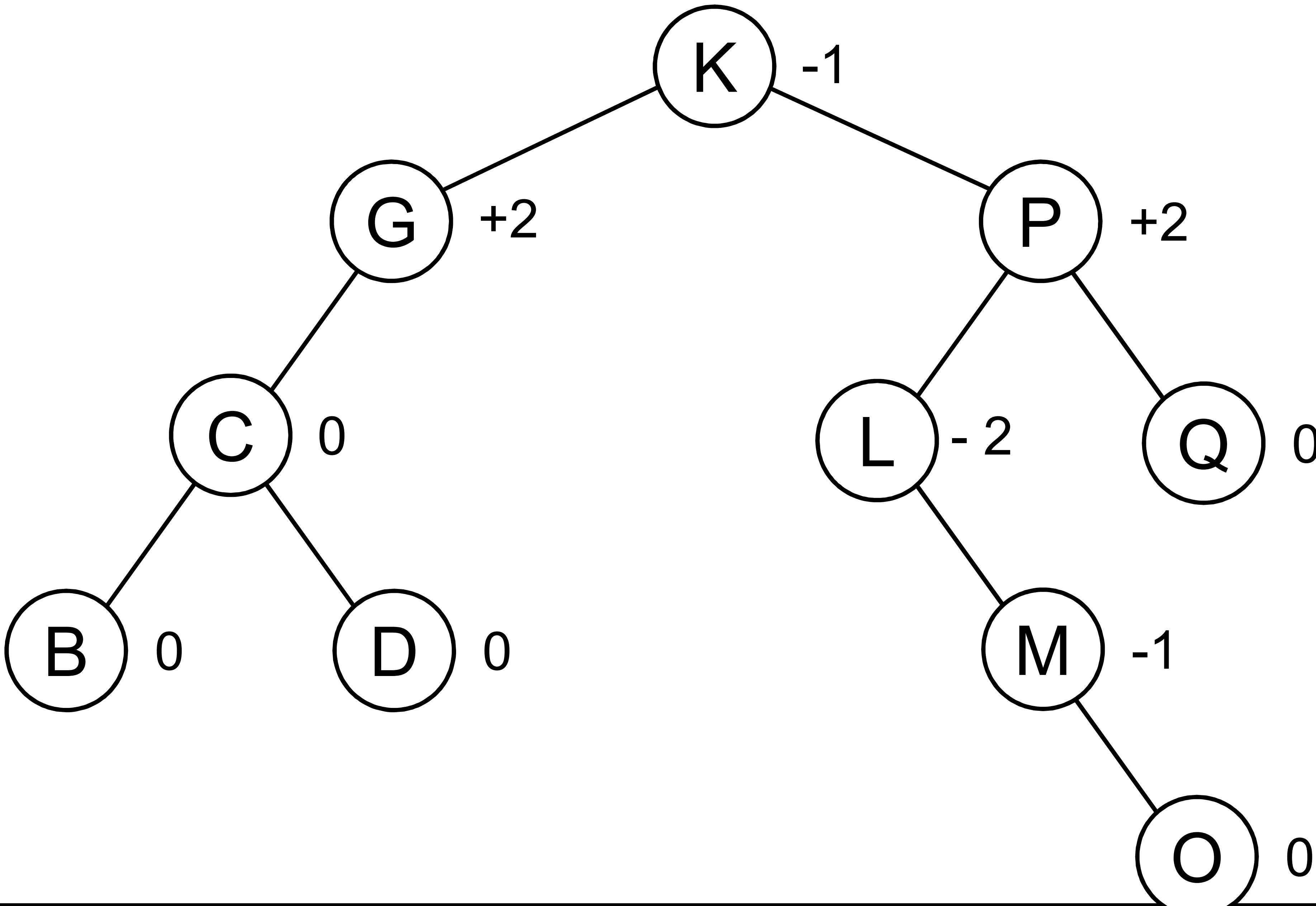
Árvore com fatores de balanceamento



Árvores AVL

- Considera-se que, se x é um nó folha o $fb(x) = 0$.
- $fb(x) > 0$ indica que a árvore está mais pesada à esquerda.
- $fb(x) < 0$ indica que a árvore está mais pesada à direita.

Árvore com fatores de balanceamento



Árvores AVL

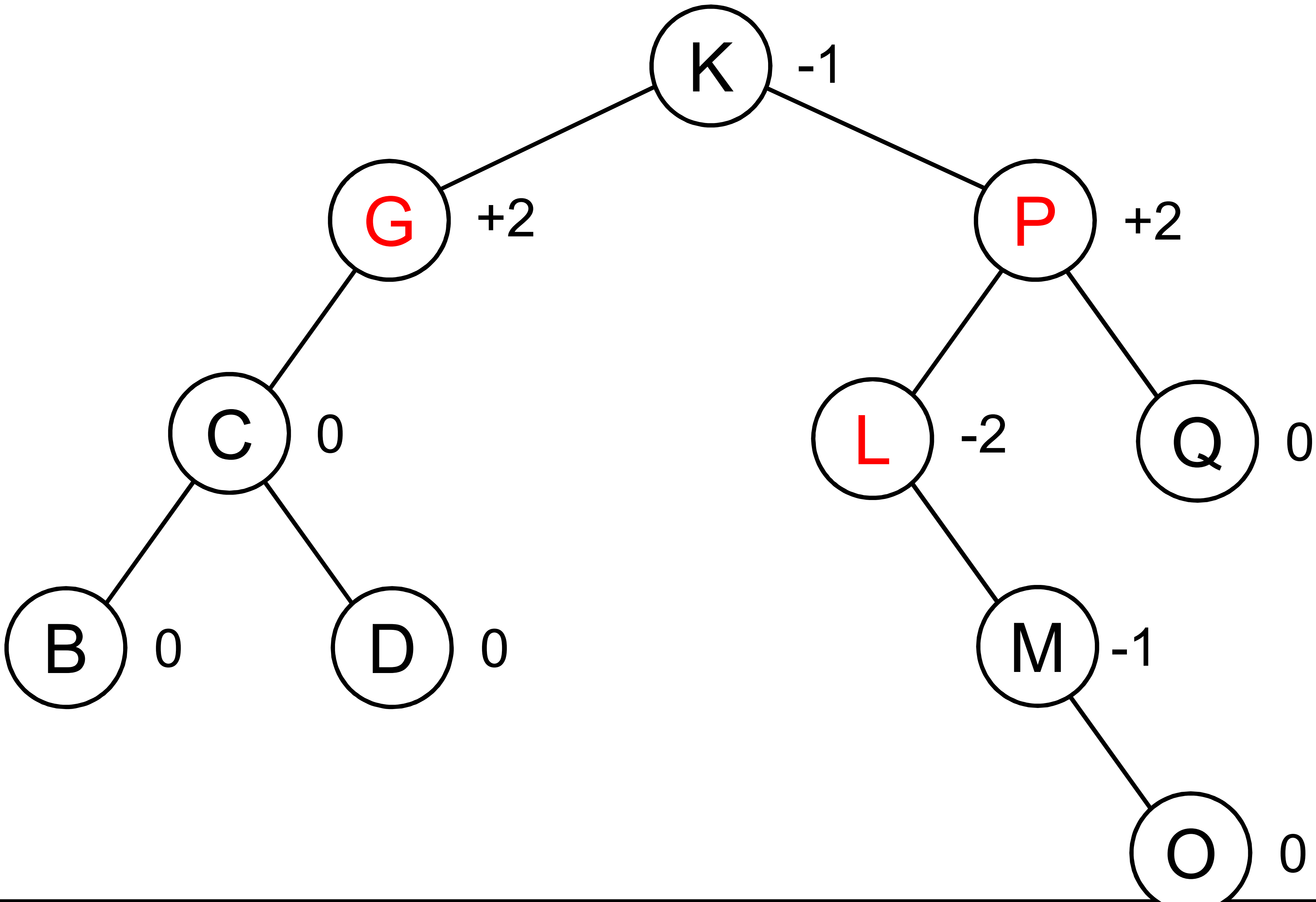
- Para todo nó interno *x* de uma ABB *A*:

  (a) Se $-1 \leq fb(x) \leq 1$, então *A* está **balanceada**.

  (b) Senão, *A* está desbalanceada.

- Quando ocorre a situação (b), reorganiza-se os nós de *A* de forma que fique balanceada.
- Esta reorganização é chamada de **operação de balanceamento.**

Árvore com fatores de balanceamento



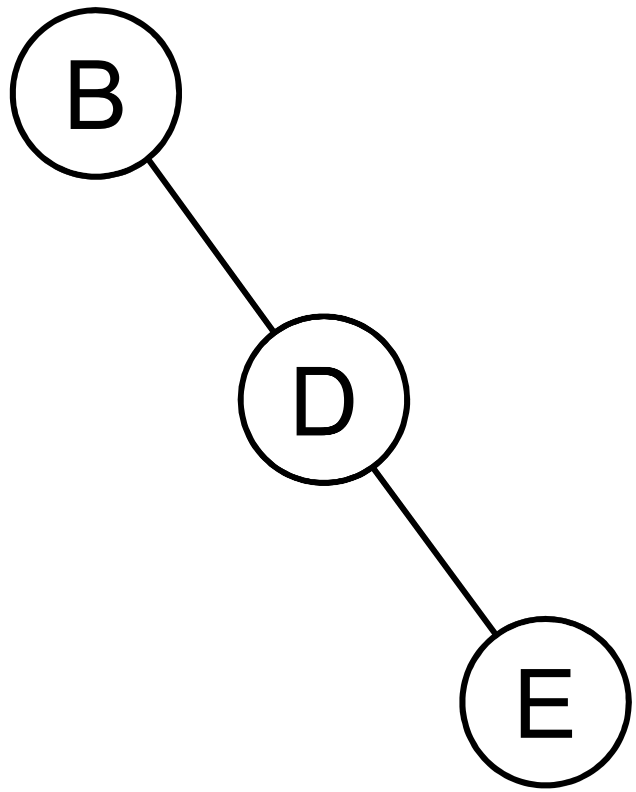
Operações de balanceamento

- Há quatro operações de balanceamento:
  - Rotação simples à esquerda – RSE.
  - Rotação simples à direita – RSD.
  - Rotação dupla à esquerda – RDE.
  - Rotação dupla à direita – RDD.

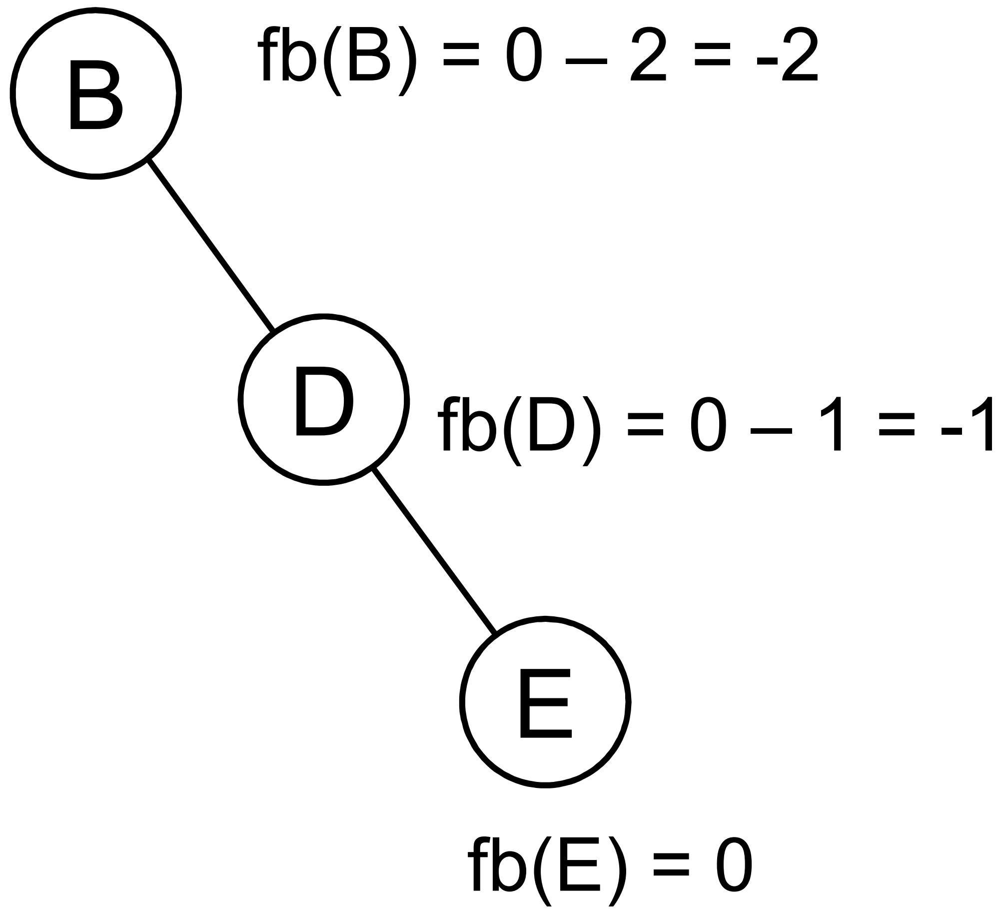
Rotação simples à esquerda - RSE

Rotação simples à esquerda - RSE

1. Calcula-se o FB(x)



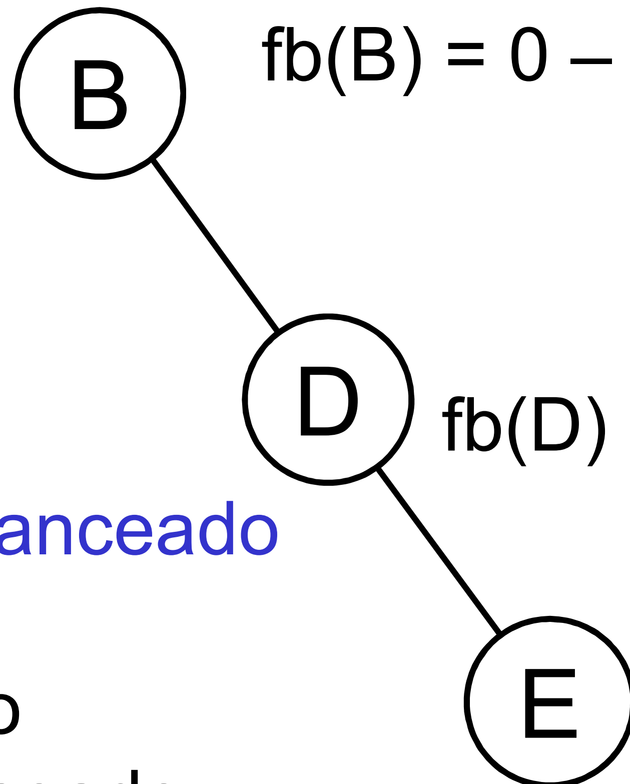
Rotação simples à esquerda - RSE



Calcula-se o FB(x)

FB(x) = hEsq(x) – hDir(x)

Rotação simples à esquerda - RSE



fb(B) = 0 – 2 = -2

fb(D) = 0 – 1 = -1

fb(E) = 0

Conclusão: B está desbalanceado

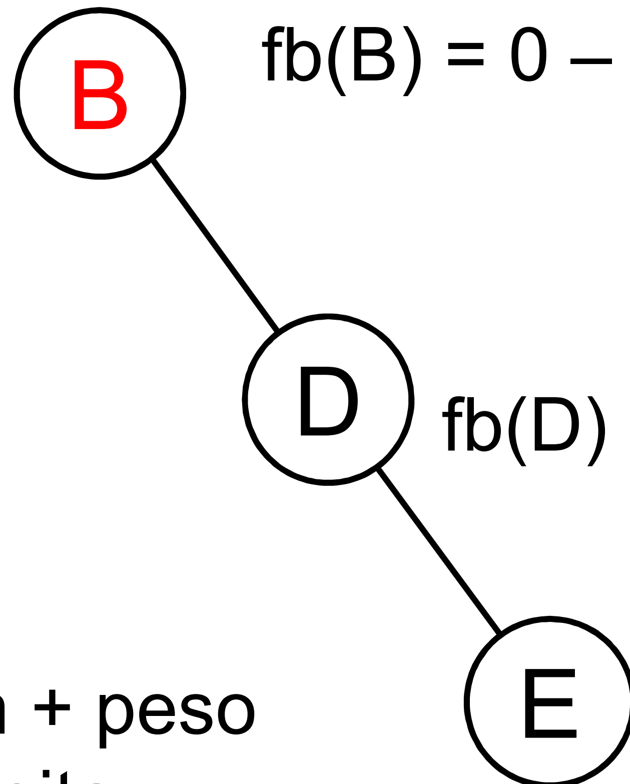
<u>Se</u> $-1 \leq fb(x) \leq 1$

<u>então</u> x está balanceado

<u>senão</u> x não está balanceado

Rotação simples à esquerda - RSE

Nó B está desbalanceado!!!



fb(B) = 0 – 2 = -2

fb(D) = 0 – 1 = -1

fb(E) = 0

fb(B) é < 0 (negativo)

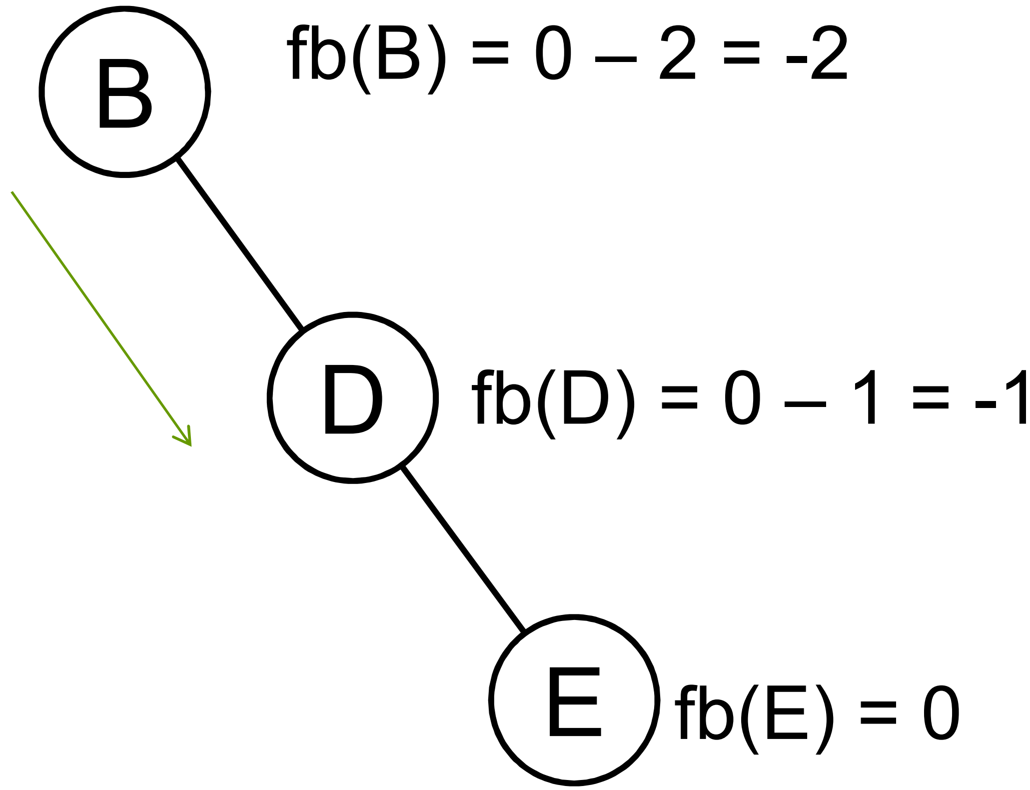
Se *fb*(*x*) < 0

então subArvDireita tem + peso

analisa filho a direita

fb(D) < 0

Rotação simples à esquerda - RSE



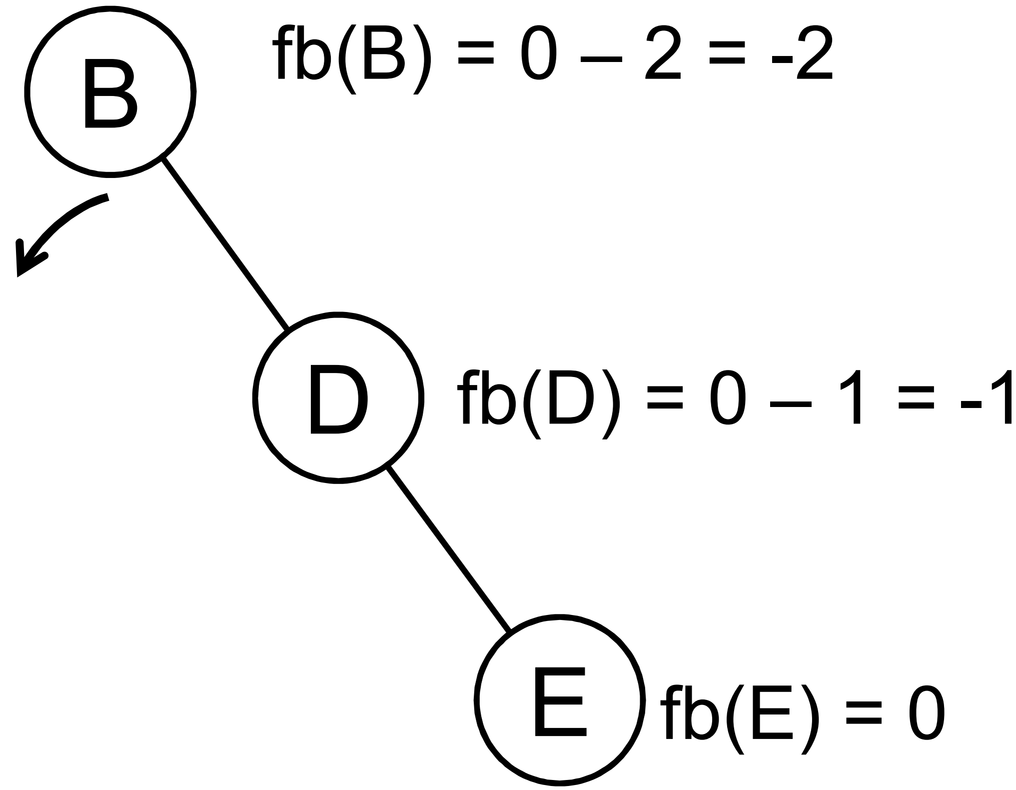
Temos fb(B) < 0 e fb(D) < 0, sinais iguais negativos

Logo, imaginamos uma reta tendendo para direita

se fb(B) > 0 e fb(D) > 0, sinais iguais positivos

Logo, imaginamos uma reta tendendo para esquerda

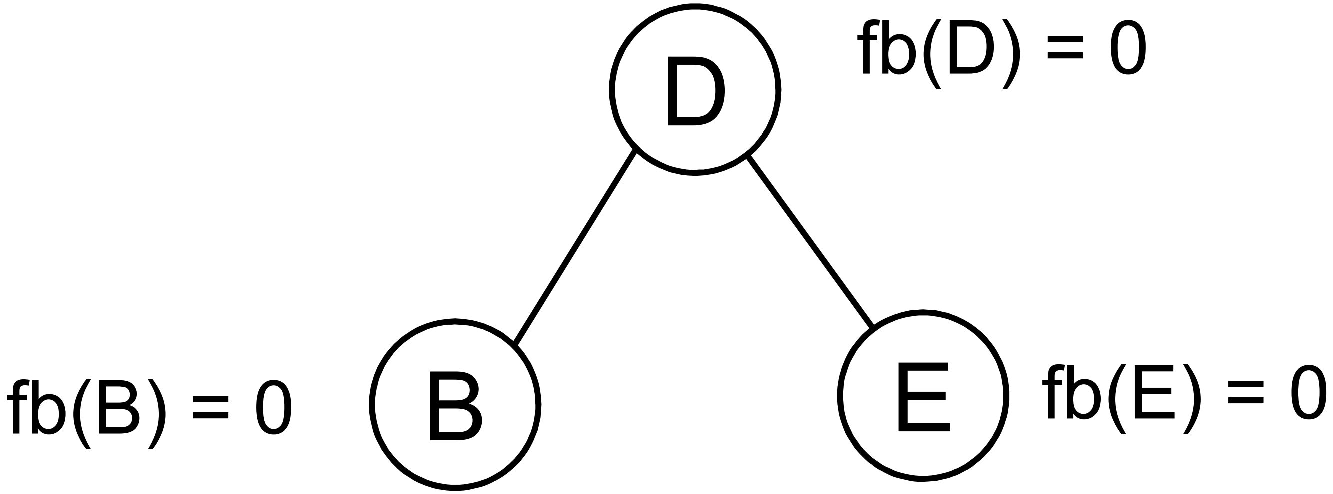
Rotação simples à esquerda - RSE



Resolvendo problema da falta de balanceamento:

1. Aplica rotação simples a esquerda **RSE(D,B)**, sendo B o nó com problema e D nó filho direita de B

Rotação simples à esquerda - RSE

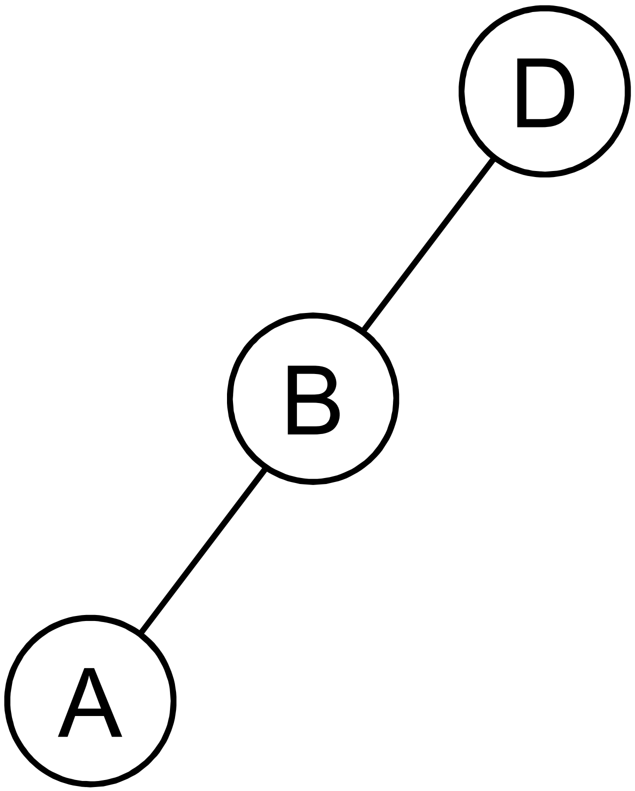


1. Aplica rotação simples a esquerda
   RSE(B,D): Rotaciona no sentido anti-horário
   D → vira pai de B

Rotação simples à direita - RSD

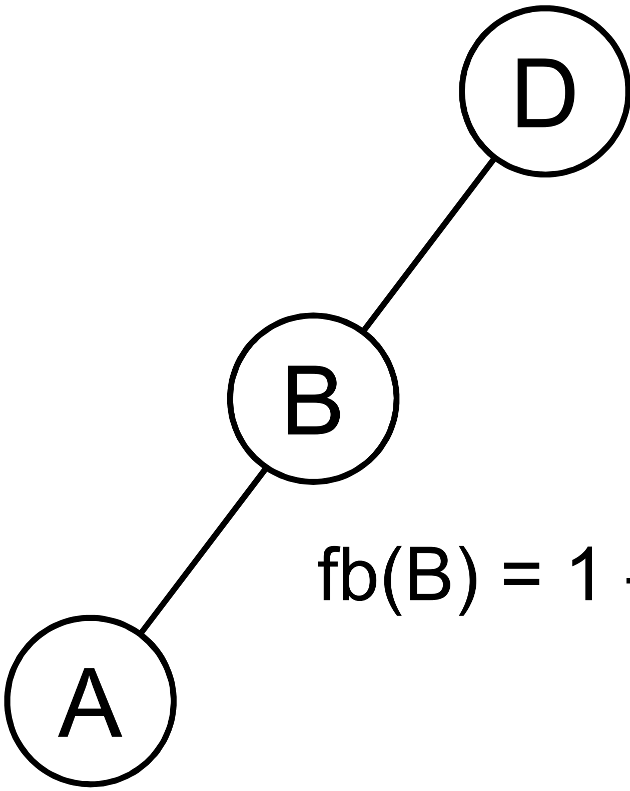
- .

Rotação simples à direita - RSD



1. Calcula-se o FB(x)

Rotação simples à direita - RSD



$fb(D) = 2 - 0 = +2$

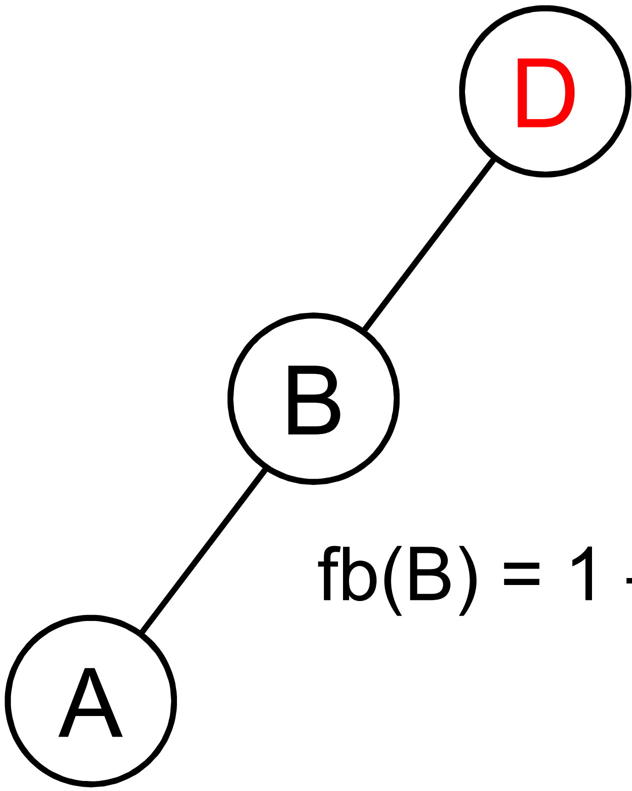
$fb(B) = 1 - 0 = +1$

$fb(E) = 0$

Calcula-se o FB(x)

$FB(x) = hEsq(x) - hDir(x)$

Rotação simples à direita - RSD



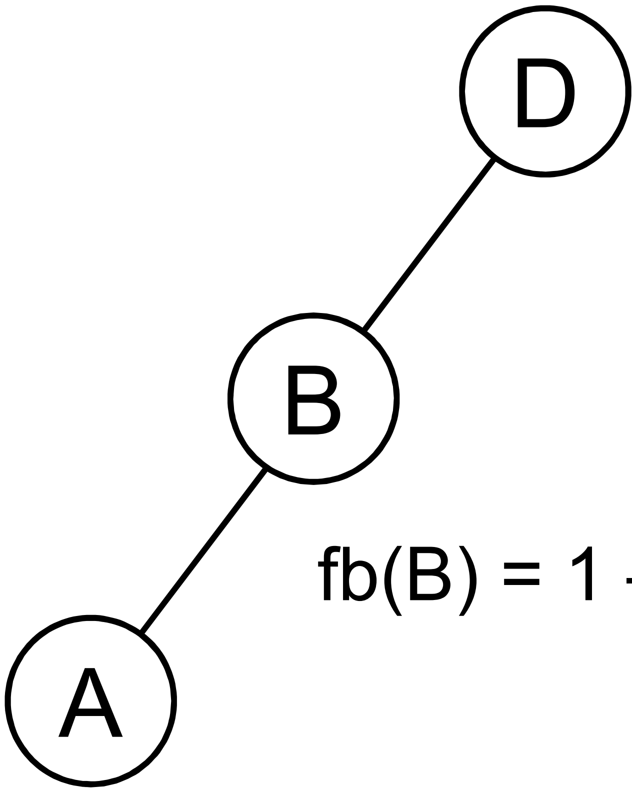
Conclusão: D está desbalanceado

Se $-1 \leq fb(x) \leq 1$

então x está balanceado

senão x não está balanceado

Rotação simples à direita - RSD



fb(D) = 2 – 0 = +2

fb(B) = 1 – 0 = +1

fb(E) = 0

fb(D) é > 0 (positivo)

<u>Se</u> *fb*(*x*) > 0

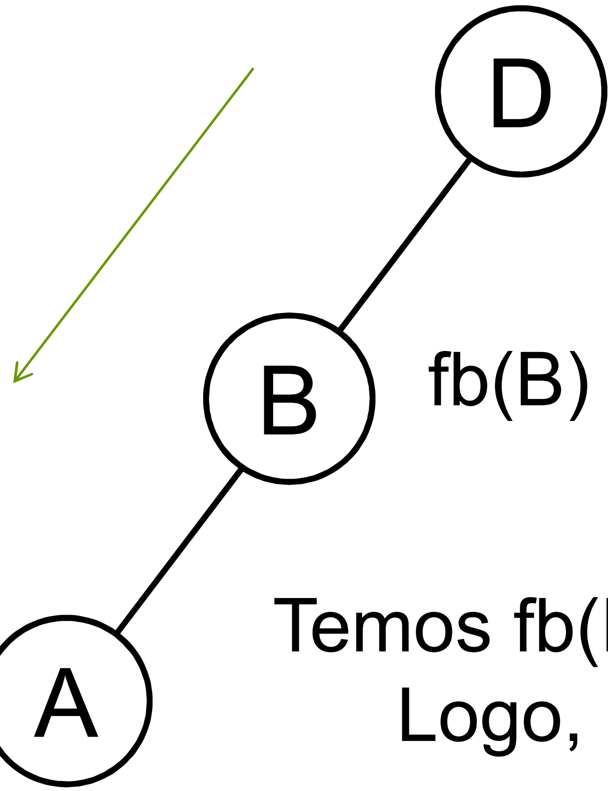
<u>então</u> subArvEsquerda tem + peso

Analisa nó esquerdo

fb(B) > 0

Rotação simples à direita - RSD

fb(D) = 2 – 0 = +2



fb(B) = 1 – 0 = +1

fb(A) = 0

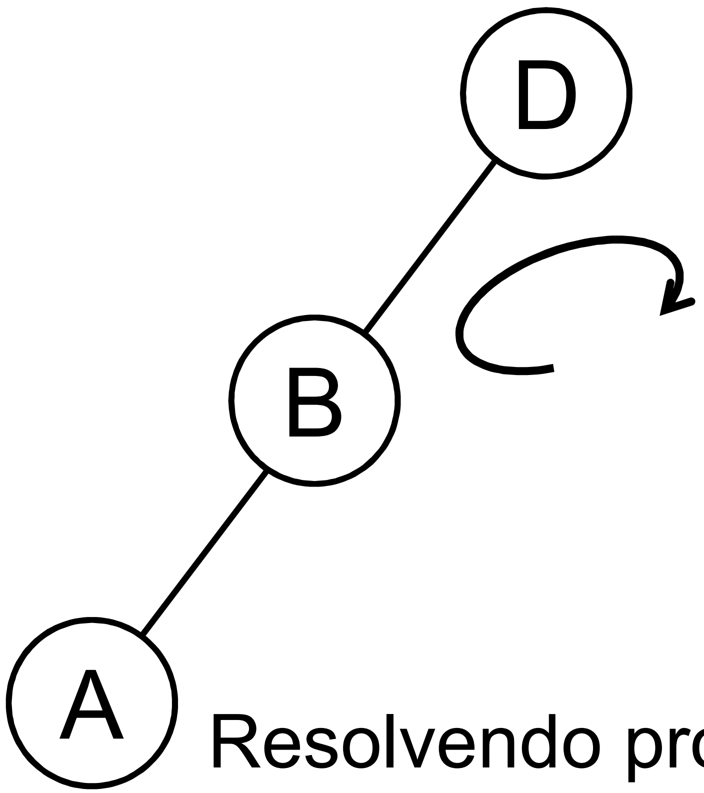
Temos fb(D) > 0 e fb(B) > 0, sinais iguais positivos

Logo, imaginamos uma reta tendendo para esquerda

se fb(D) < 0 e fb(B) < 0, sinais iguais negativos

Logo, imaginamos uma reta tendendo para direita

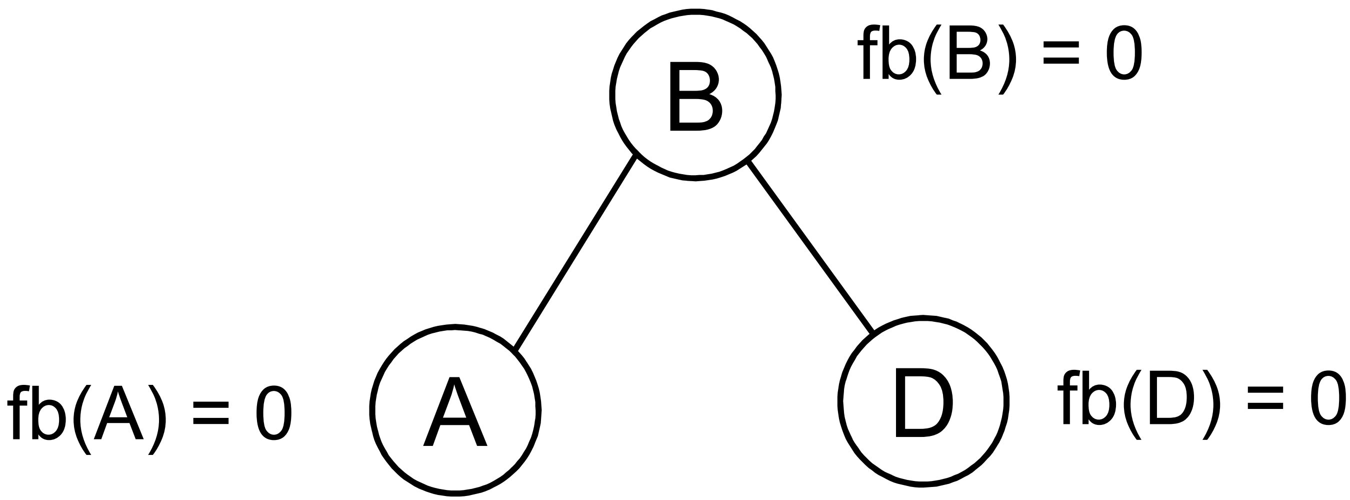
Rotação simples à direita - RSD



Resolvendo problema da falta de balanceamento:

1. Aplica rotação simples a direita
   RSD(B,D), sendo D o nó com problema e B nó filho direita de B
   RSD(B,D) → rotação sentido horário

Rotação simples à direita - RSD



Resolvendo problema da falta de balanceamento:

1. Aplica rotação simples a direita **RSD(D,B)**, sendo D o nó com problema e B nó filho direita de B

Rotações simples

- As operações RSD e RSE são mutuamente inversas:

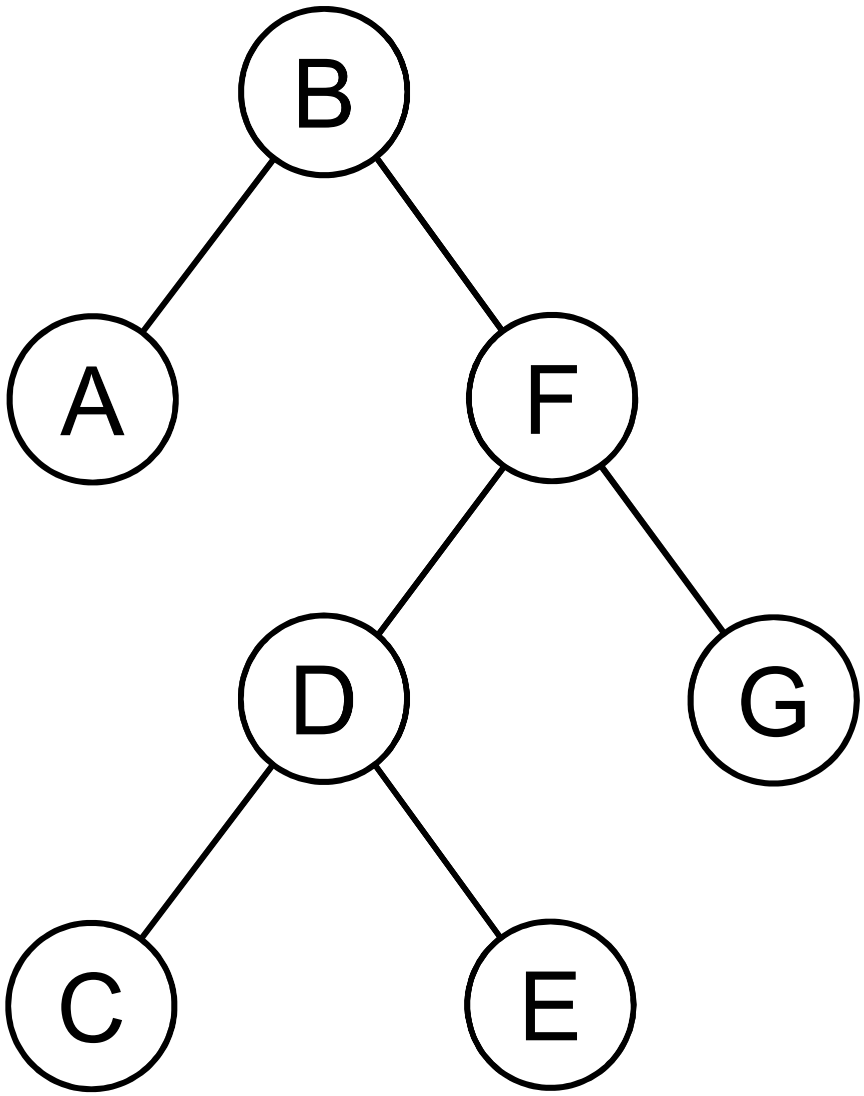
RSD(*A*) → *A'*

RSE(*A'*) → *A*

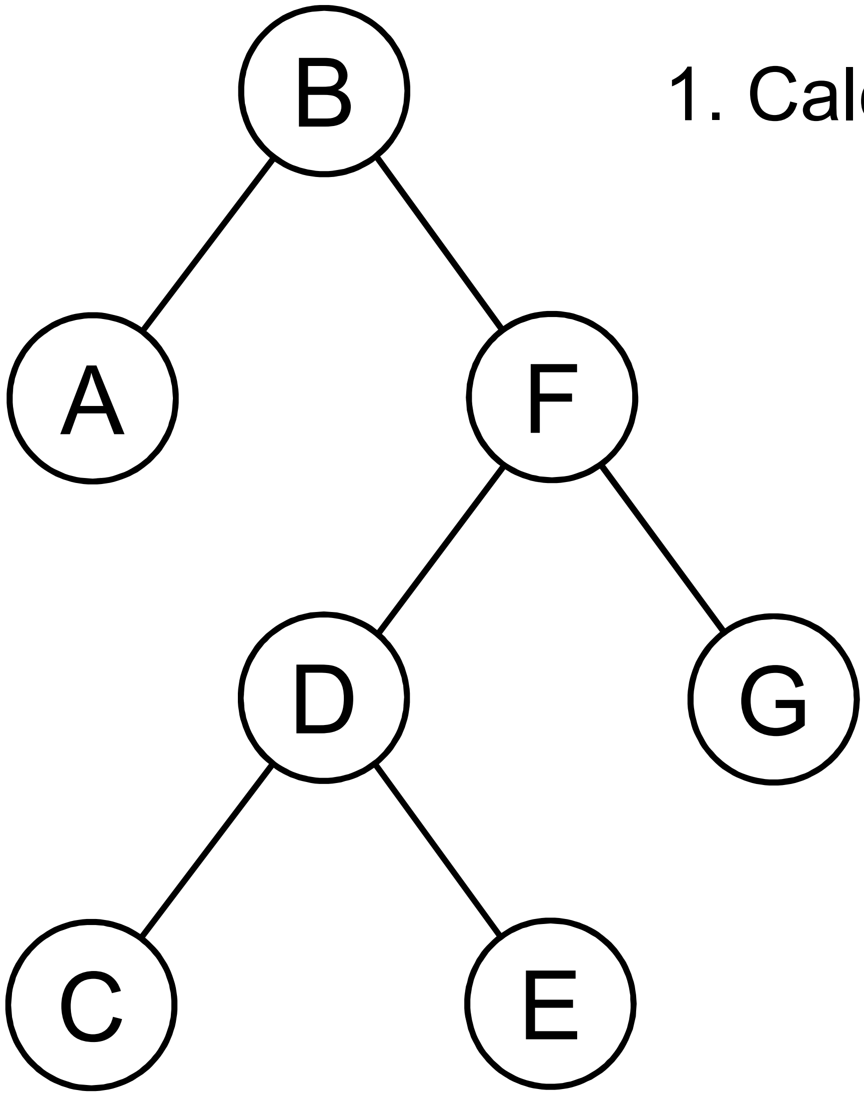
Rotação dupla à esquerda - RDE

- Operação mais complexa que envolve as duas operações simples.
- Regra: aplica-se uma RSD seguida de uma RSE
- RDE = RSD + RSE.

Rotação dupla à esquerda - RDE

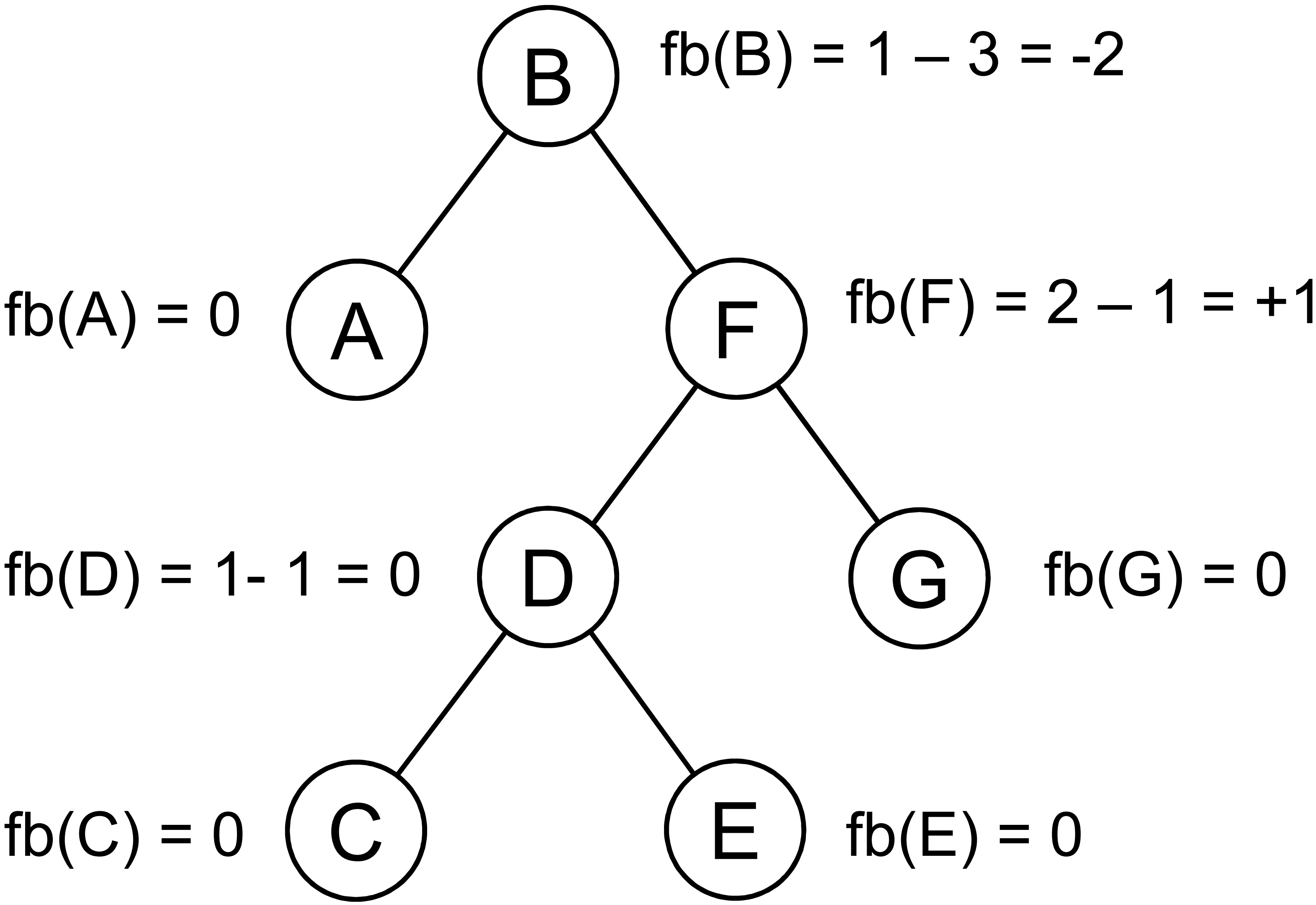


Rotação dupla à esquerda - RDE

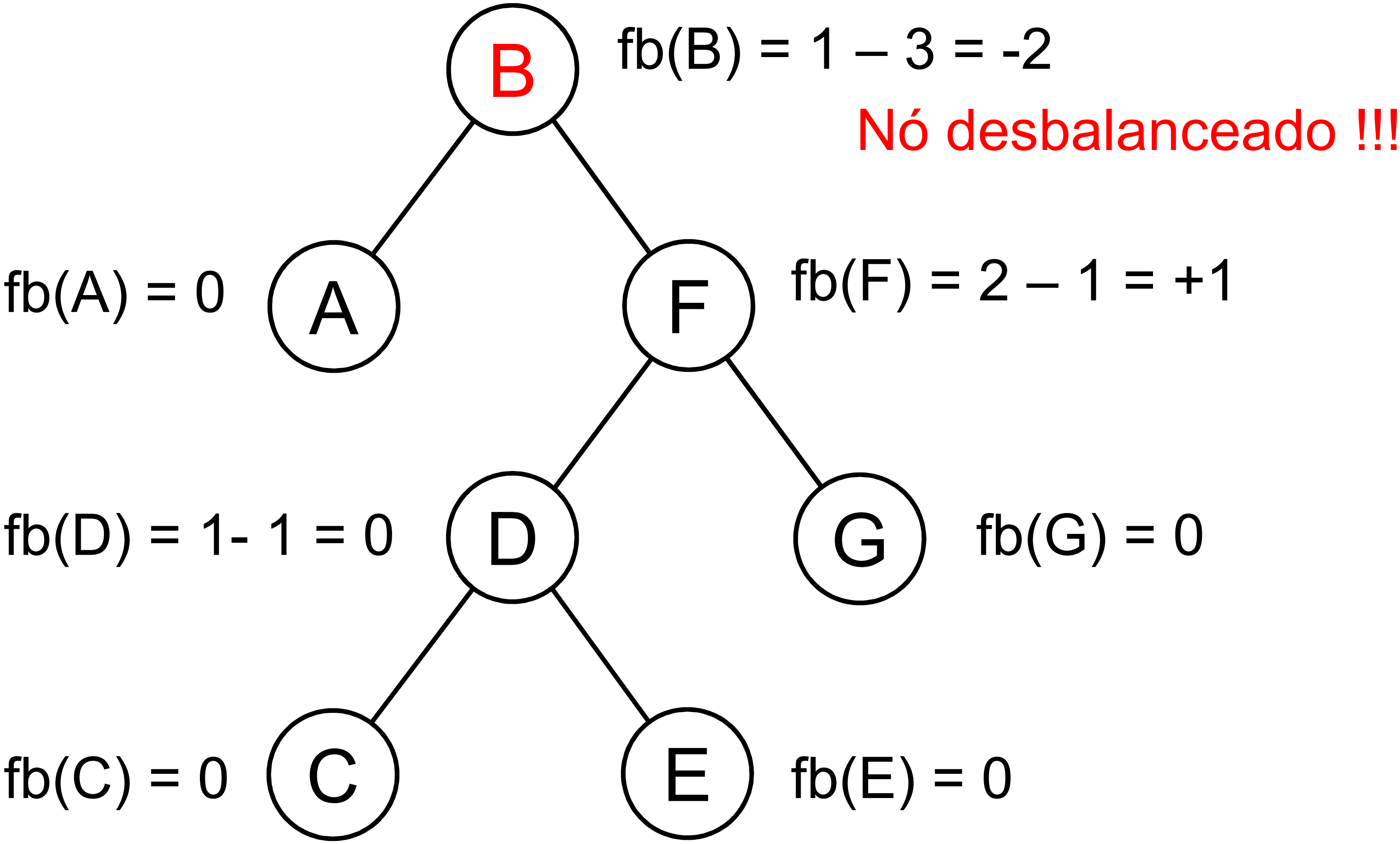


1. Calcula-se o FB(x)

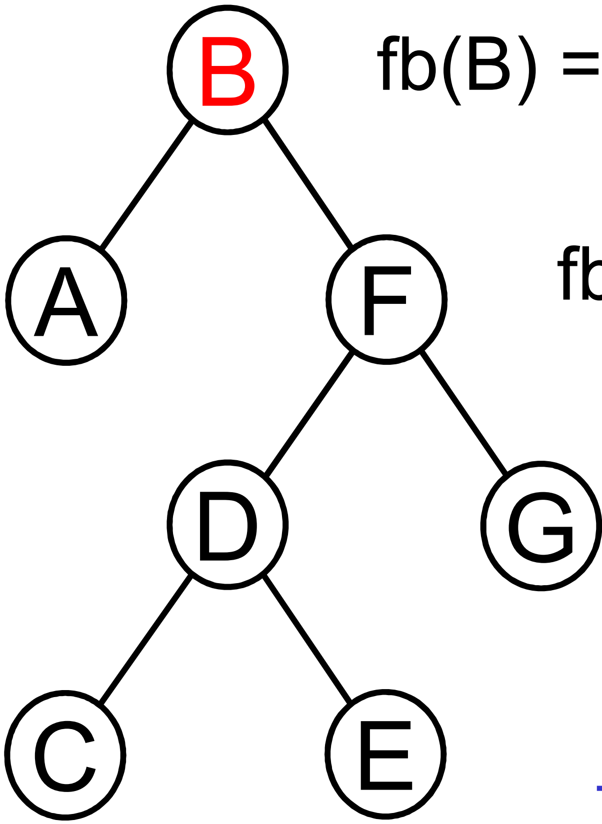
Rotação dupla à esquerda - RDE



Rotação dupla à esquerda - RDE



Rotação dupla à esquerda - RDE



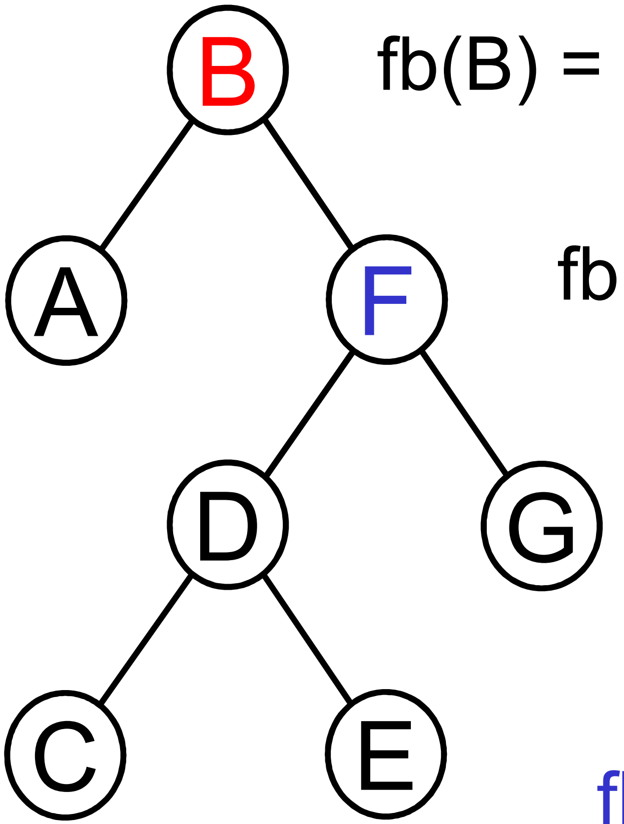
fb(B) = 1 – 3 = -2

fb(F) = 2 – 1 = +1

fb(B) é < 0 (negativo)

Se *fb*(*x*) < 0
  então subArvDireita tem + peso

Rotação dupla à esquerda - RDE



fb(B) = 1 – 3 = -2

fb(F) = 2 – 1 = +1

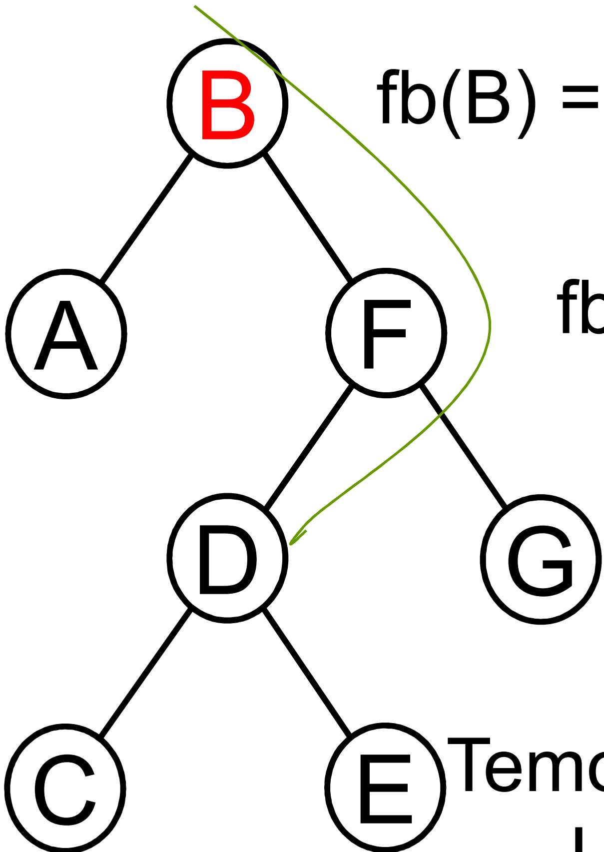
fb(B) é < 0 (negativo)

Se *fb*(*x*) < 0

então subArvDireita tem + peso

analisa o filho a direita de B

Rotação dupla à esquerda - RDE

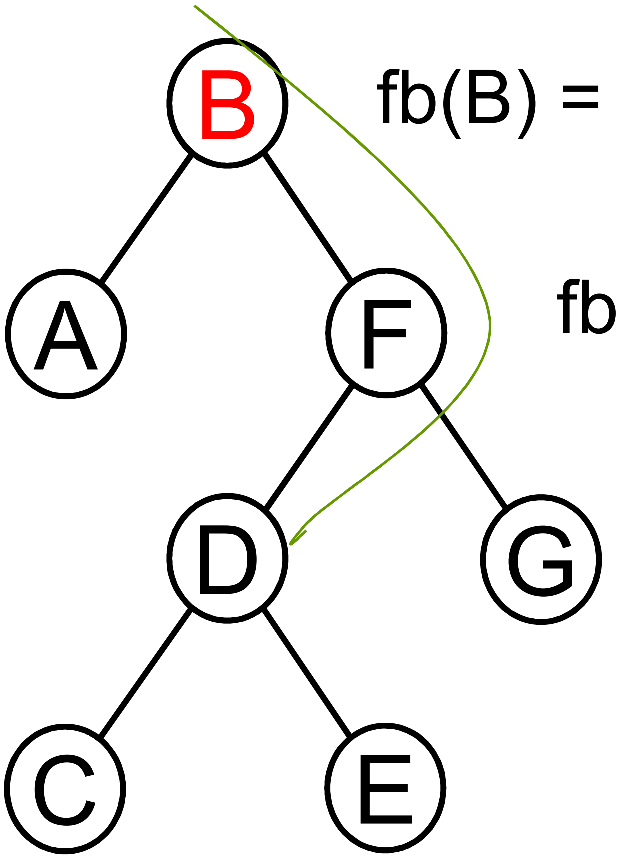


fb(B) = 1 – 3 = -2

fb(F) = 2 – 1 = +1

Temos fb(B) < 0 e fb(F) > 0, sinais diferentes
Logo, imaginamos joelho para direita
se fb(B) > 0 e fb(F) < 0, sinais diferentes
Logo, imaginamos joelho para direita

Rotação dupla à esquerda - RDE

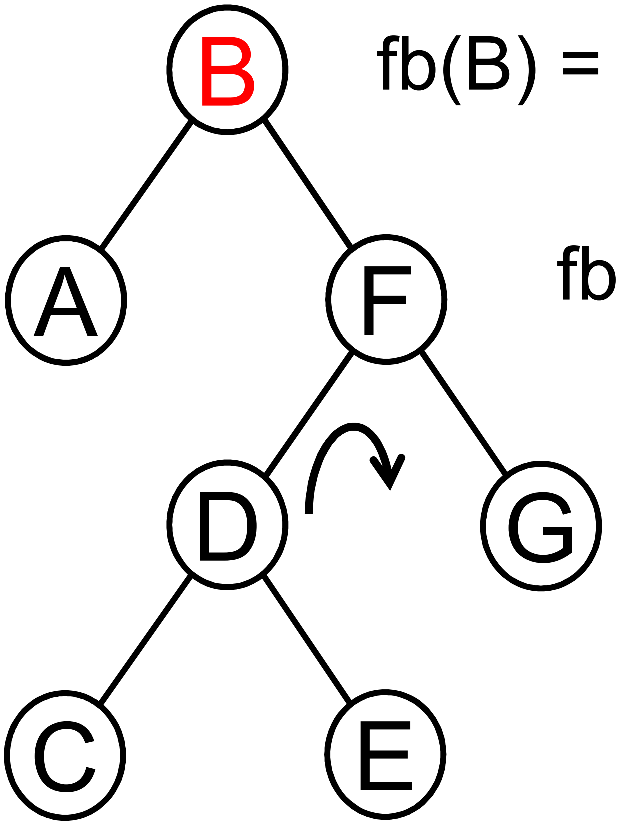


fb(B) = 1 – 3 = -2

fb(F) = 2 – 1 = +1

Resolvendo o problema:
RDE() = RSD() + RSE()

Rotação dupla à esquerda - RDE

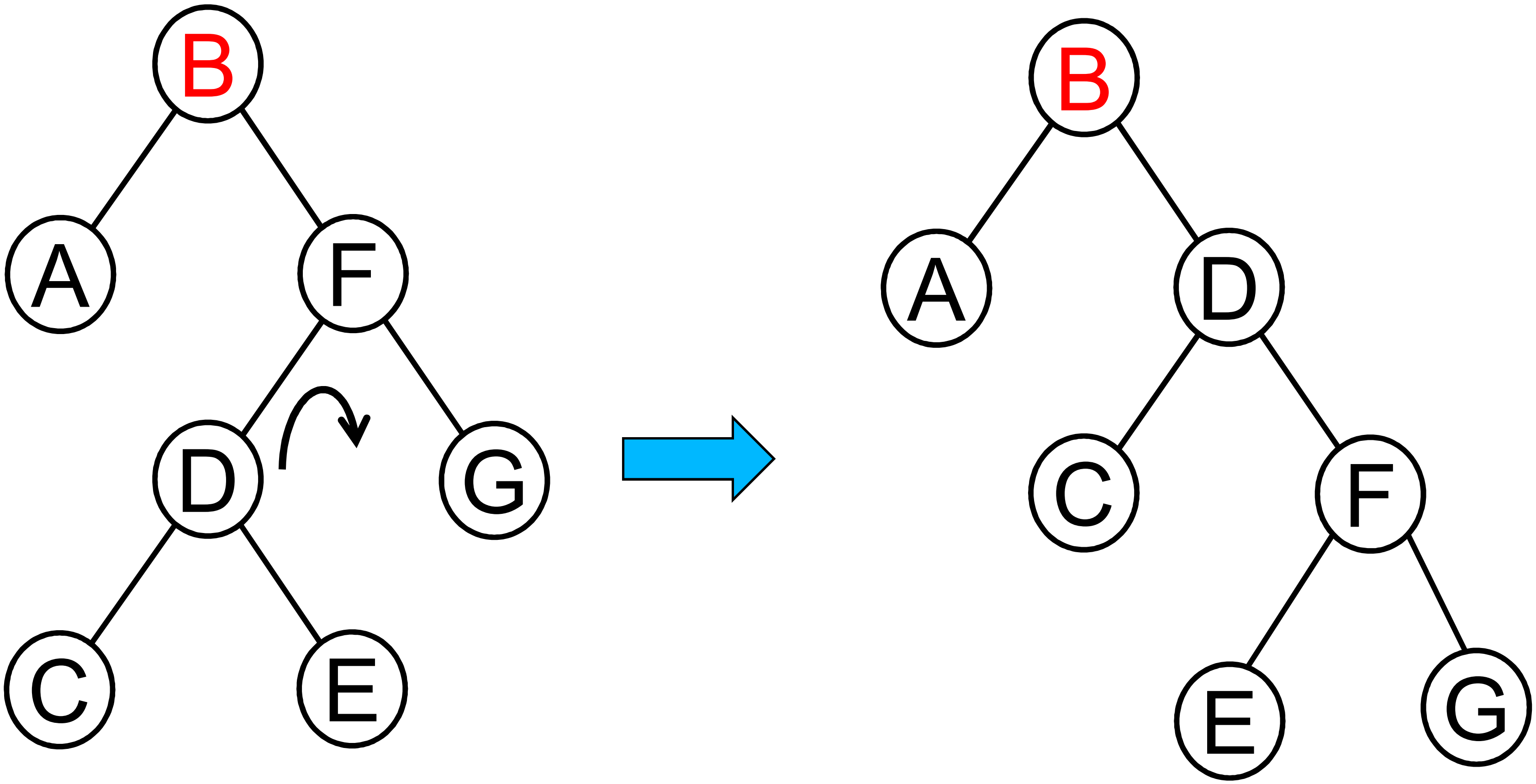


fb(B) = 1 – 3 = -2

fb(F) = 2 – 1 = +1

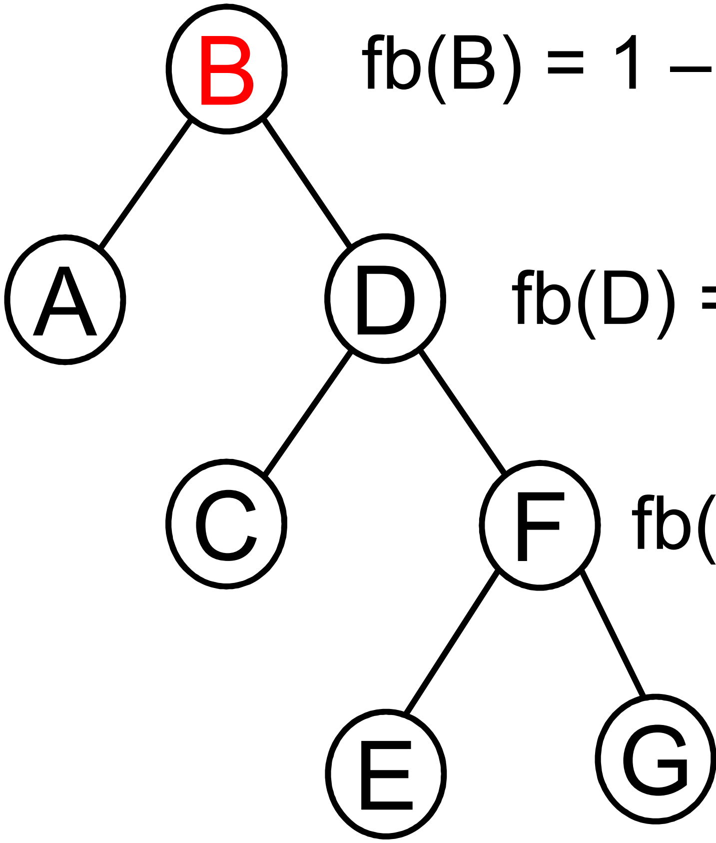
Resolvendo o problema:
RDE() = **RSD(D,F)** + RSE()

Rotação dupla à esquerda - RDE



RSD(F,D) → D vai ser pai de F
F vai ser filho a direita de D
F herda o filho de D

Rotação dupla à esquerda - RDE



$fb(B) = 1 – 3 = -2$

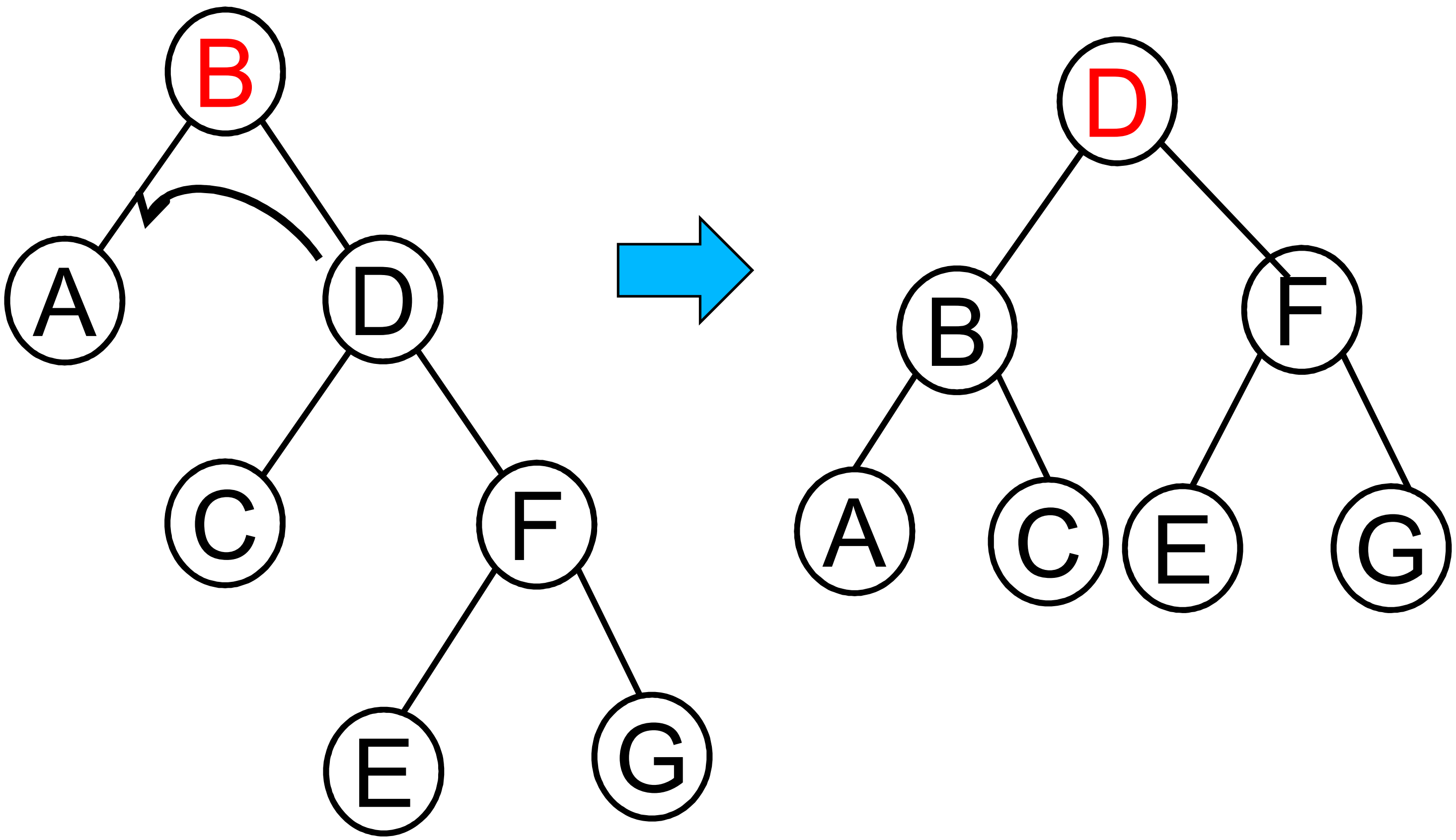
$fb(D) = 1 – 2 = -1$

$fb(F) = 0$

Resolvendo o problema:

RDE() = RSD(D,F) **+ RSE(B,D)**

Rotação dupla à esquerda - RDE



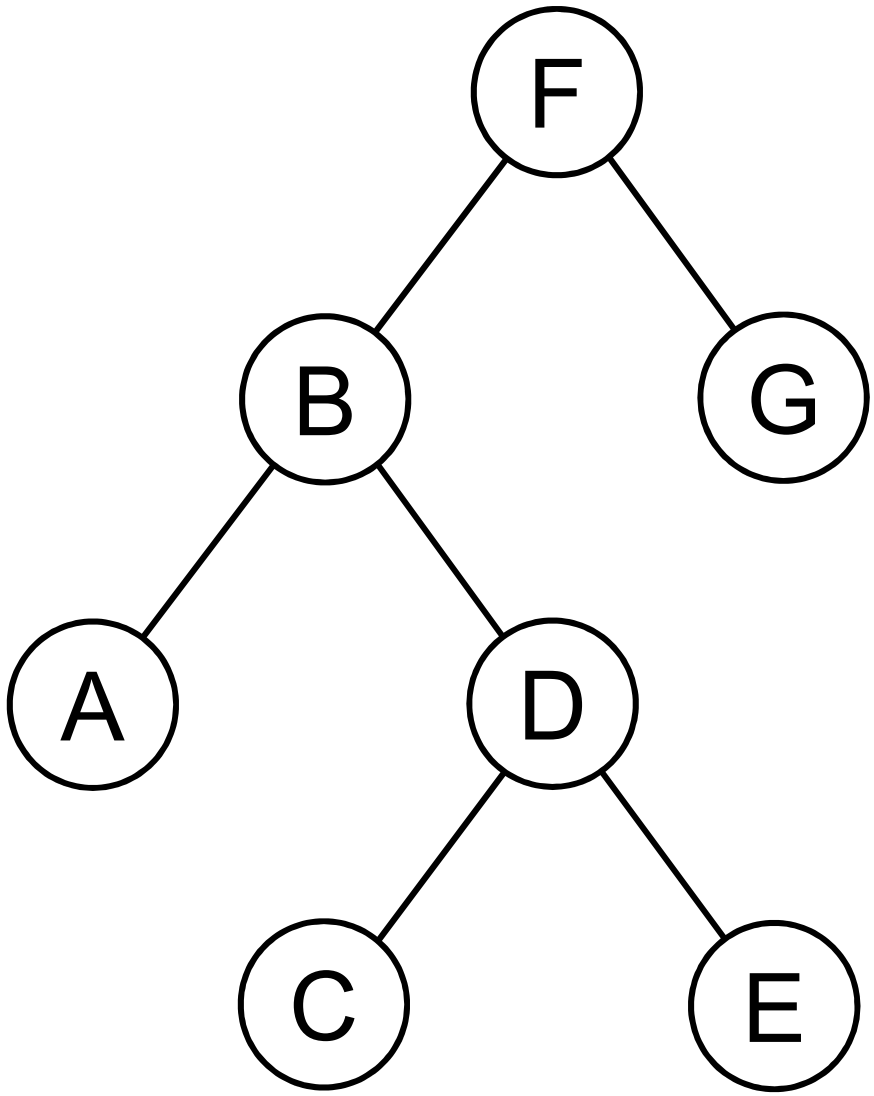
RSD(B,D) → D vai ser pai de B

B herda o filho de D

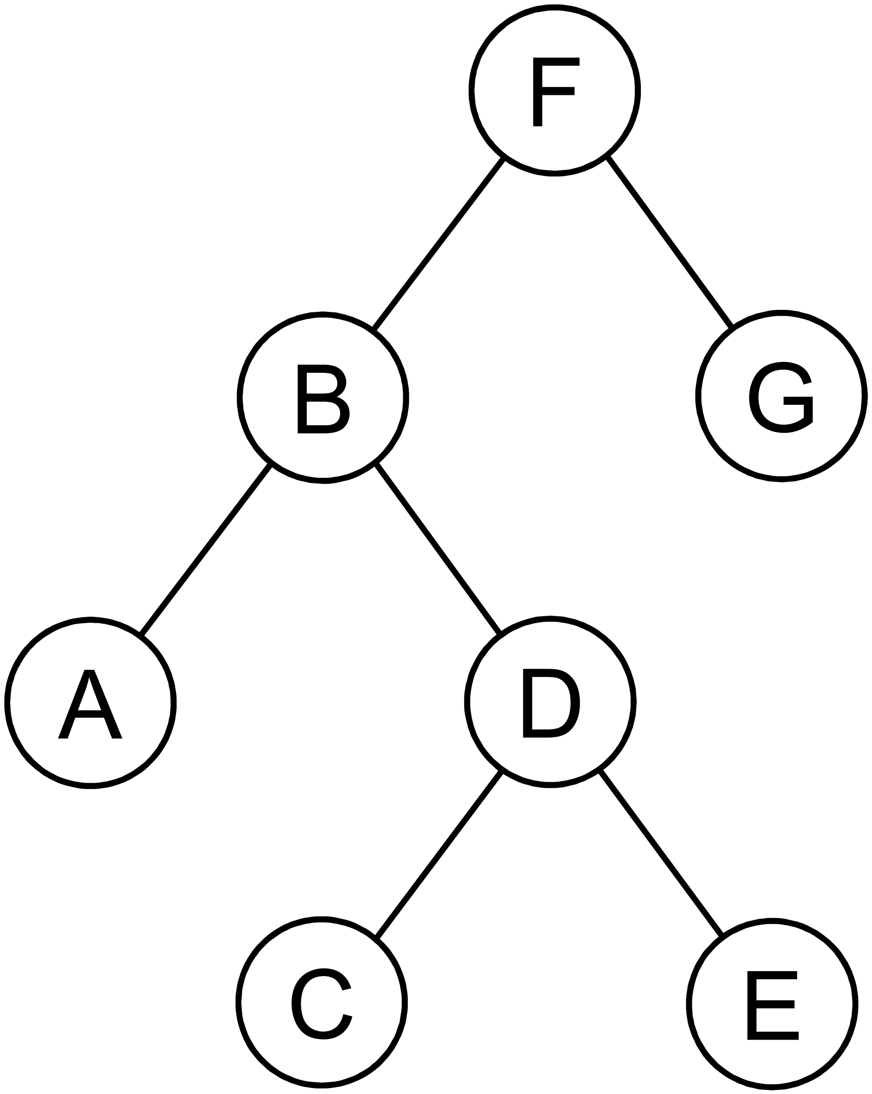
Rotação dupla à direita - RDD

- Operação também complexa que envolve as duas operações simples.
- Regra: aplica-se uma RSE seguida de uma RSD.
- RDD = RSE + RSD.

Rotação dupla à direita - RDD

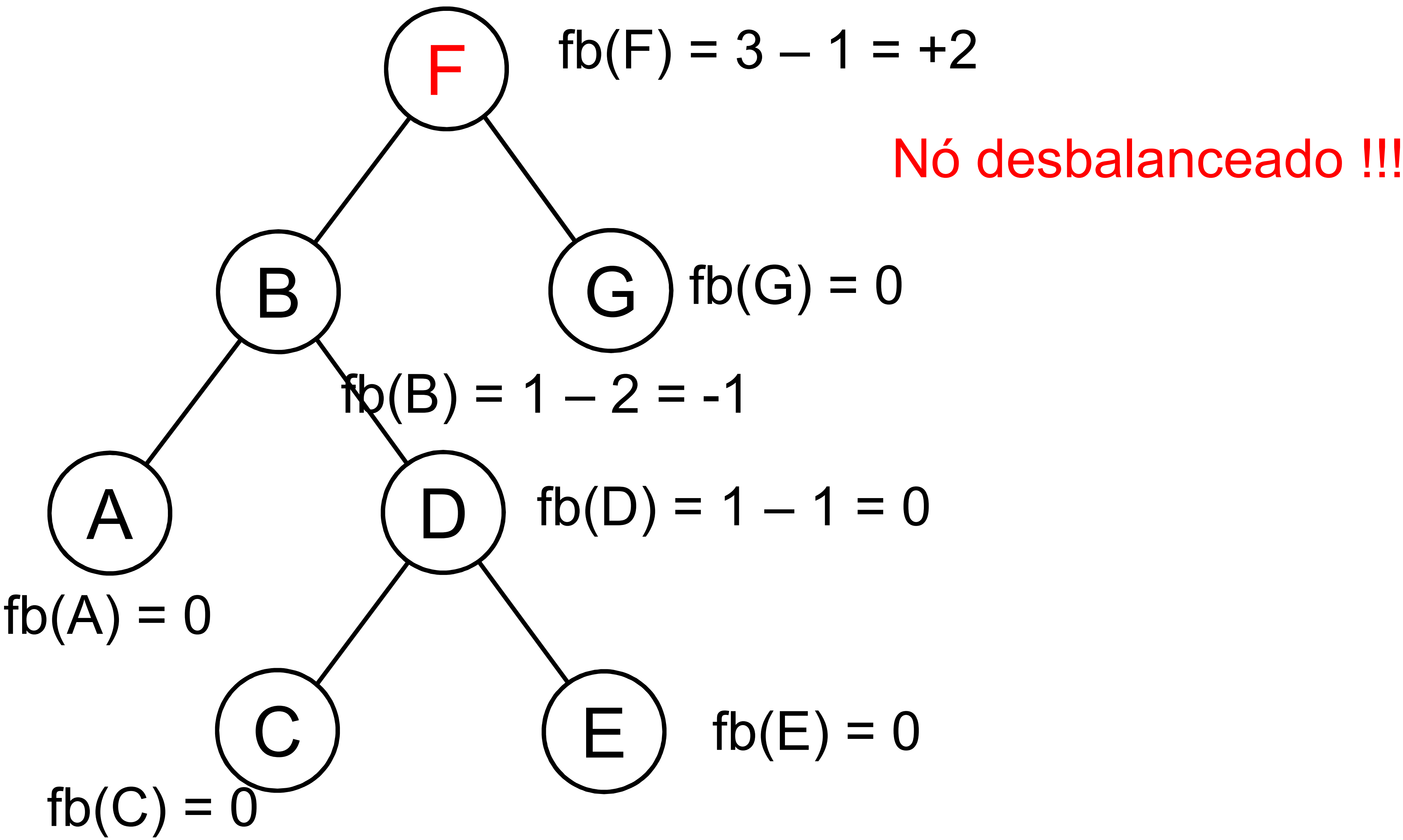


Rotação dupla à direita - RDD

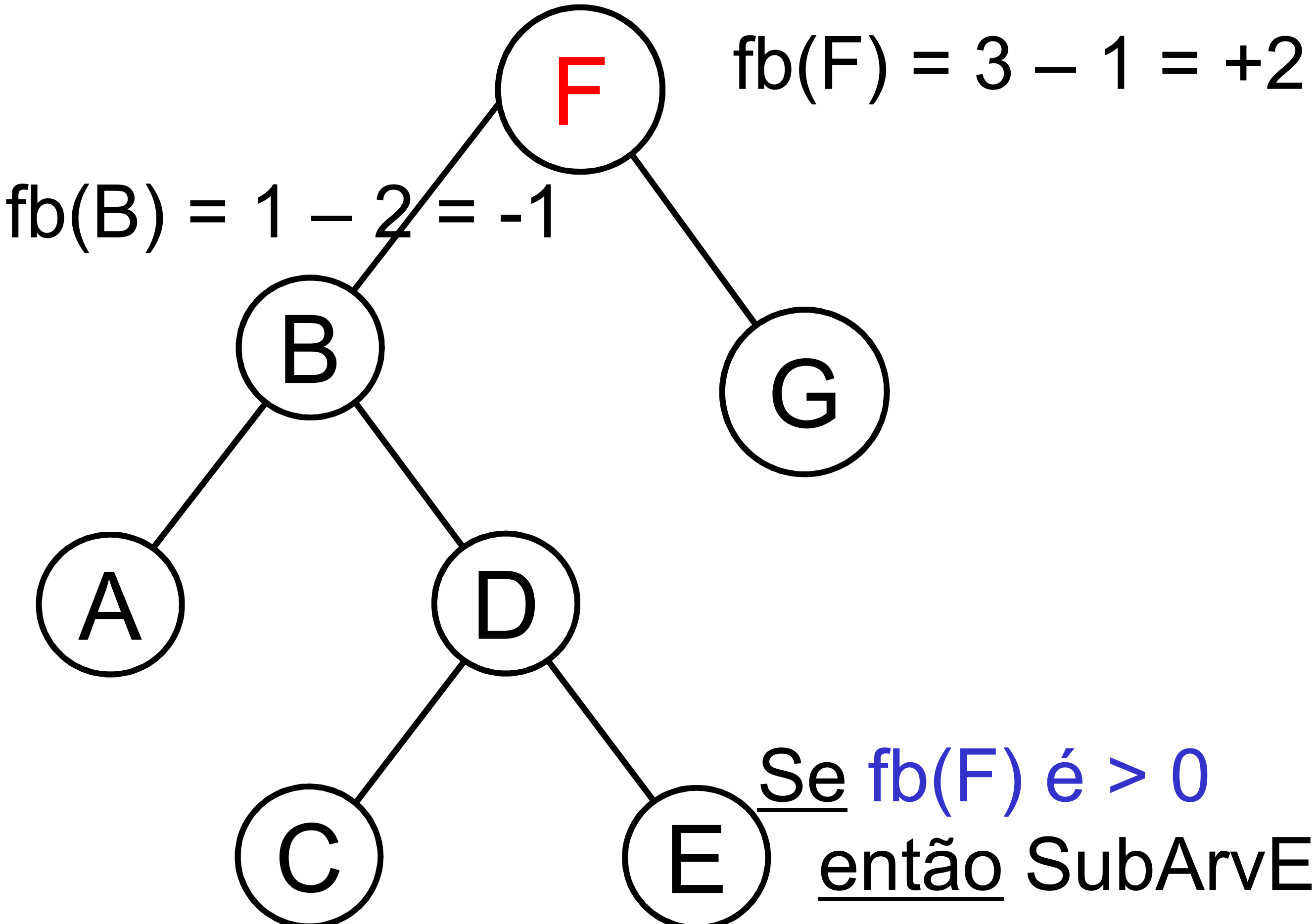


1. Calcula-se o FB(x)

Rotação dupla à direita- RDD

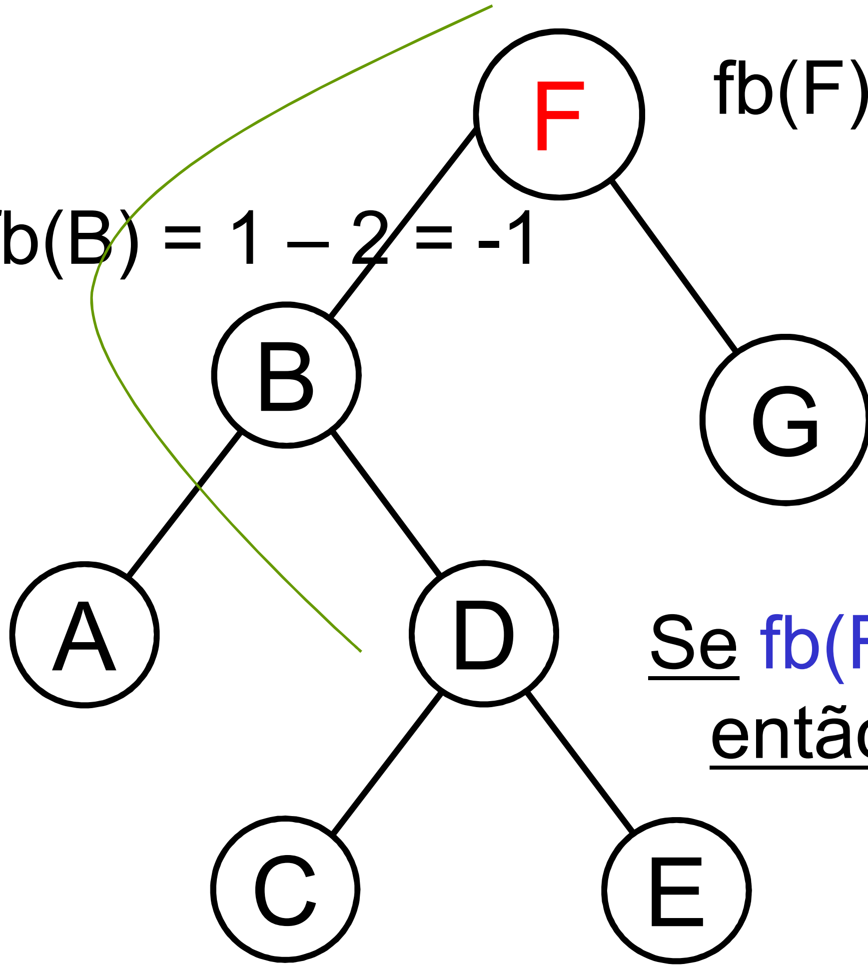


Rotação dupla à direita- RDD



Se fb(F) é > 0
então SubArvEsquerda + pesada
Analisa filho Esquerda: B
fb(B) < 0

Rotação dupla à direita- RDD



fb(F) = 3 – 1 = +2

fb(B) = 1 – 2 = -1

<u>Se</u> fb(F) é > 0 e fb(B) < 0

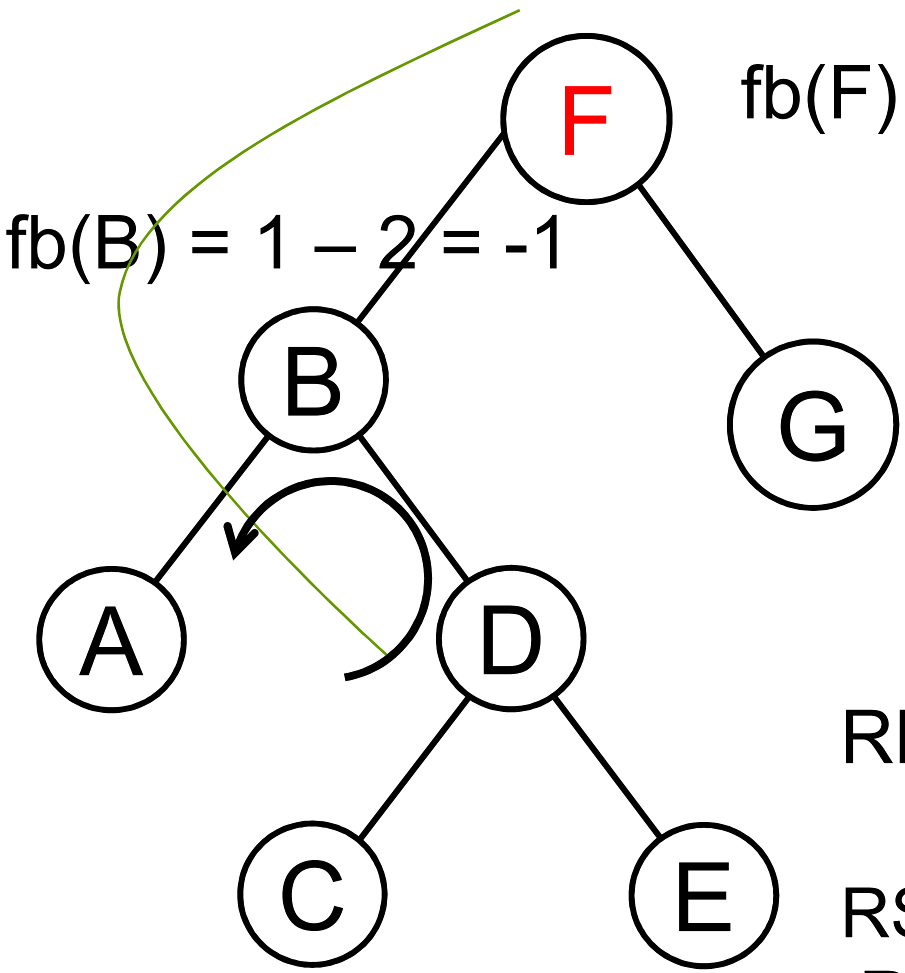
<u>então</u> joelho para esquerda

objetivo: transformar numa reta

como?

RDD() = RSE() + RSD()

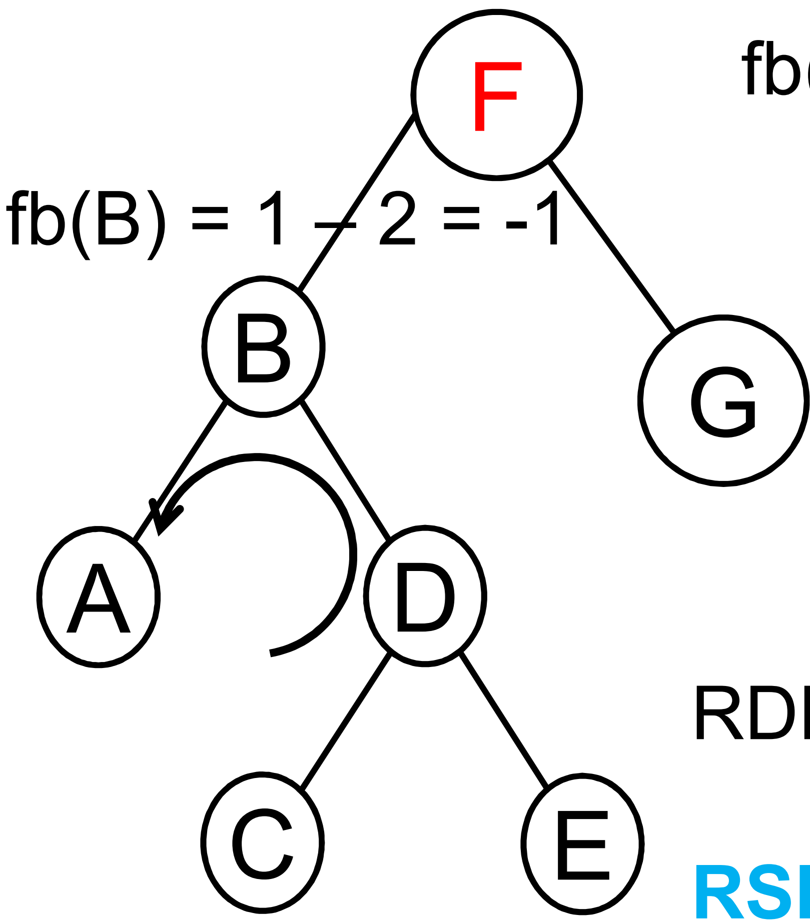
Rotação dupla à direita- RDD



RDD() = **RSE(D,B)** + RSD()

RSE() → sentido anti-horário
RSE() será sempre no filho do nó desbalanceado

Rotação dupla à direita- RDD

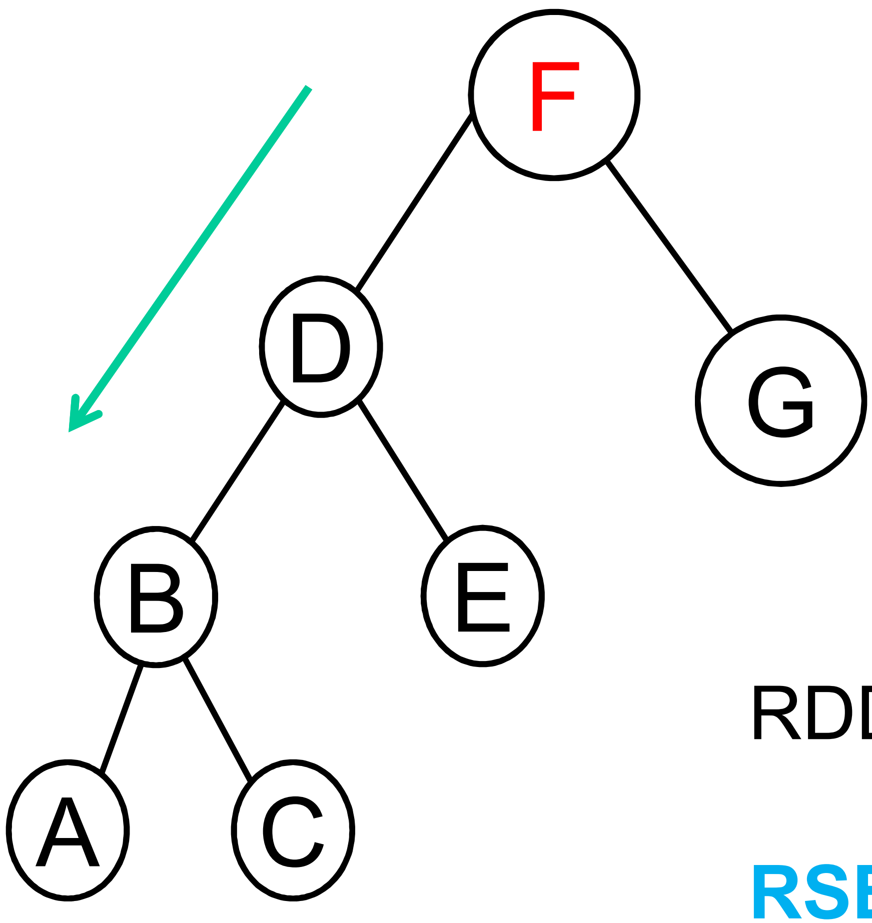


fb(F) = 3 – 1 = +2

RDD() = **RSE(D,B)** + RSD()

RSE(D,B) → D vira pai de B
B herda filho esq de D

Rotação dupla à direita- RDD

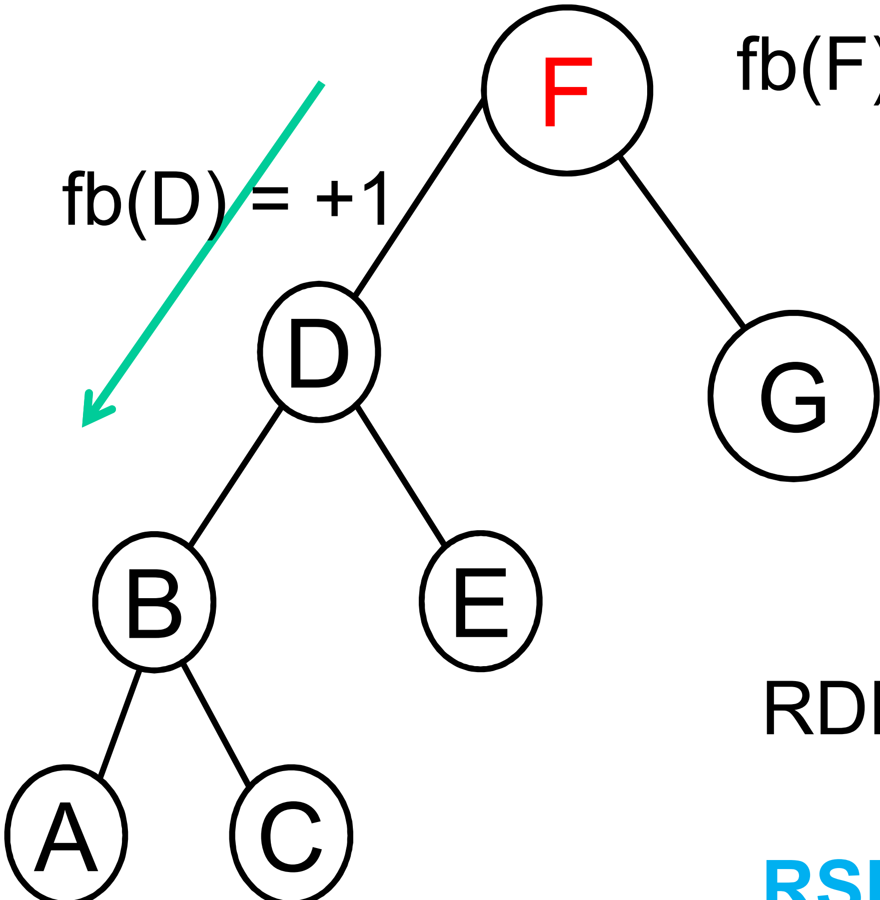


RDD() = **RSE(D,B)** + RSD()

RSE(D,B) → Transf Joelho em Reta

Objetivo: resolver a reta

Rotação dupla à direita- RDD



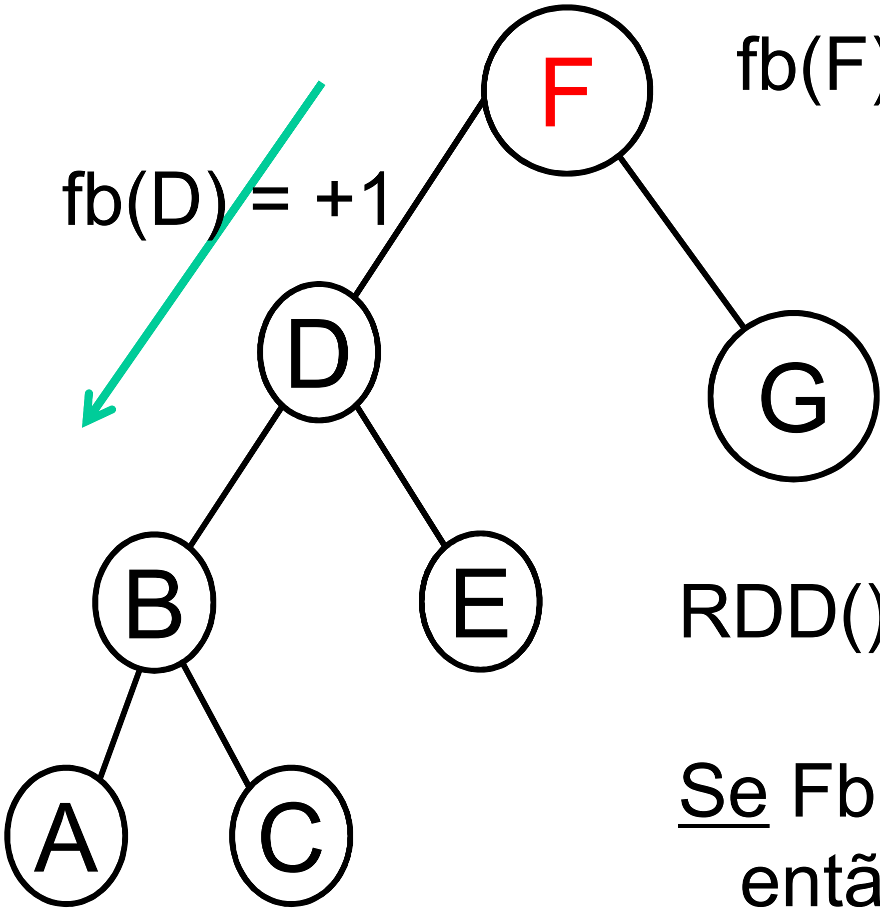
fb(F) = 3 – 1 = +2

RDD() = **RSE(D,B)** + RSD()

RSE(D,B) → Transf Joelho em Reta

Objetivo: resolver a reta

Rotação dupla à direita- RDD

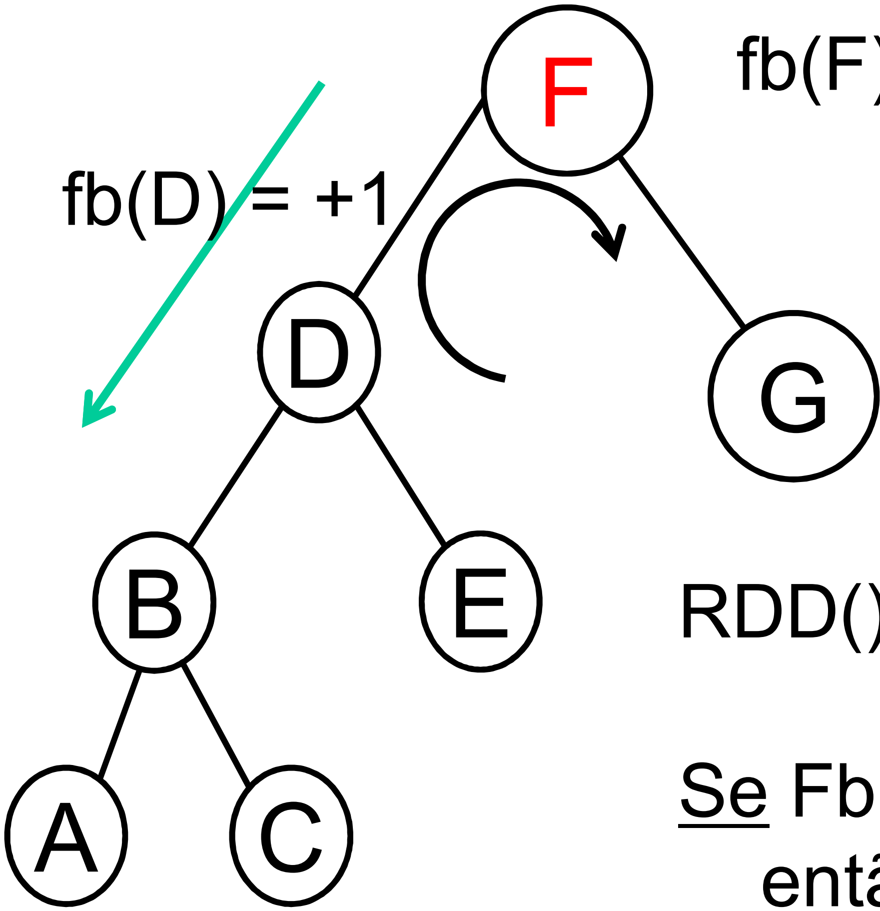


fb(F) = 3 – 1 = +2

RDD() = RSE(D,B) + **RSD()**

<u>Se</u> Fb(F) > 0
<u>então</u> subArvoreEsquerda + pesada
Analisa filho Esquerda
fb(D) = +1

Rotação dupla à direita- RDD



fb(F) = 3 – 1 = +2

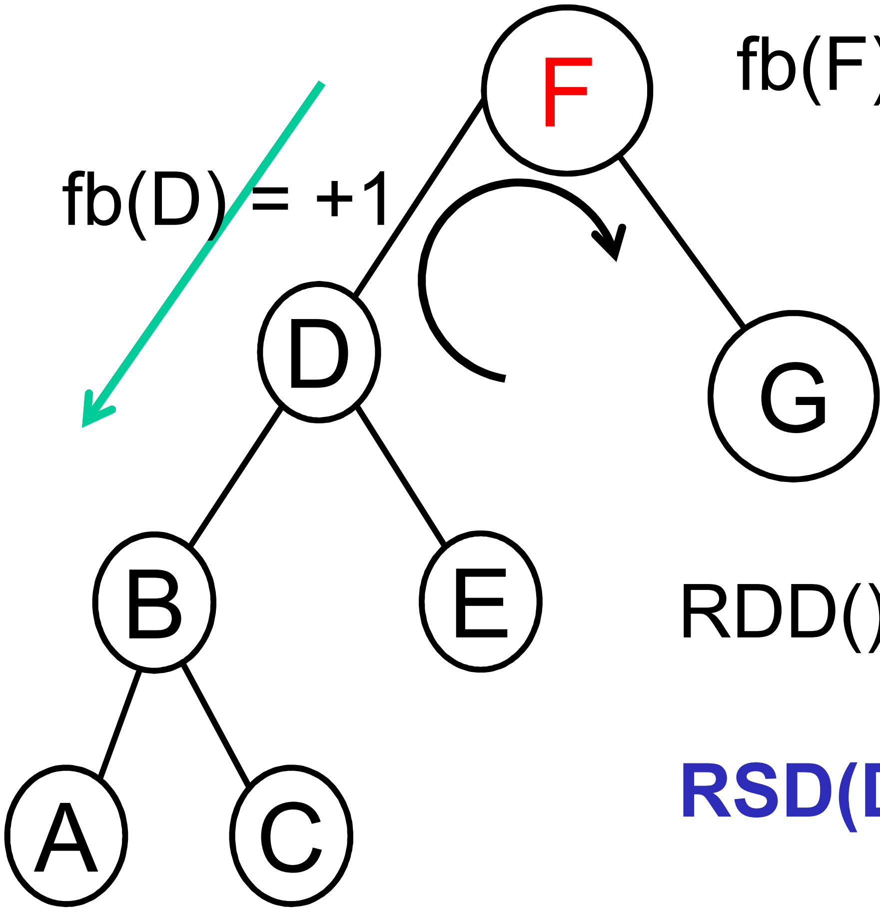
RDD() = RSE(D,B) + **RSD()**

<u>Se</u> Fb(F) > 0 e fb(D) > 0

<u>então</u> reta para direita

solução? Aplica RSD(D,F)

Rotação dupla à direita- RDD

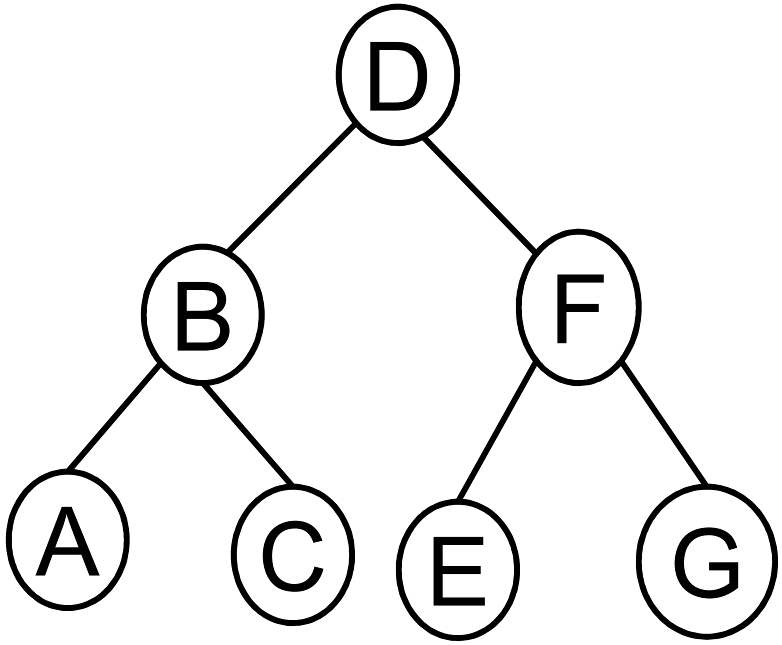


fb(F) = 3 – 1 = +2

RDD() = RSE(D,B) + **RSD(D,F)**

RSD(D,F) → D vira pai de F
F herda filho direito de D

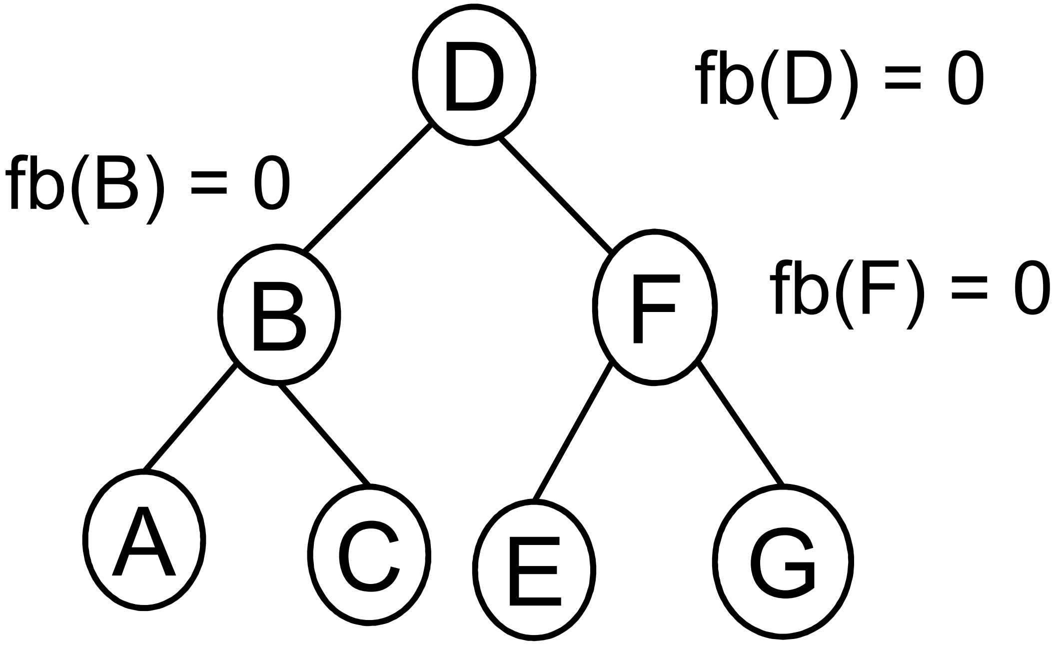
Rotação dupla à direita- RDD



RDD() = RSE(D,B) + **RSD(D,F)**

RSD(D,F) → D vira pai de F
F herda filho direito de D

Rotação dupla à direita- RDD



Árvore está balanceada!!!

Operações de balanceamento

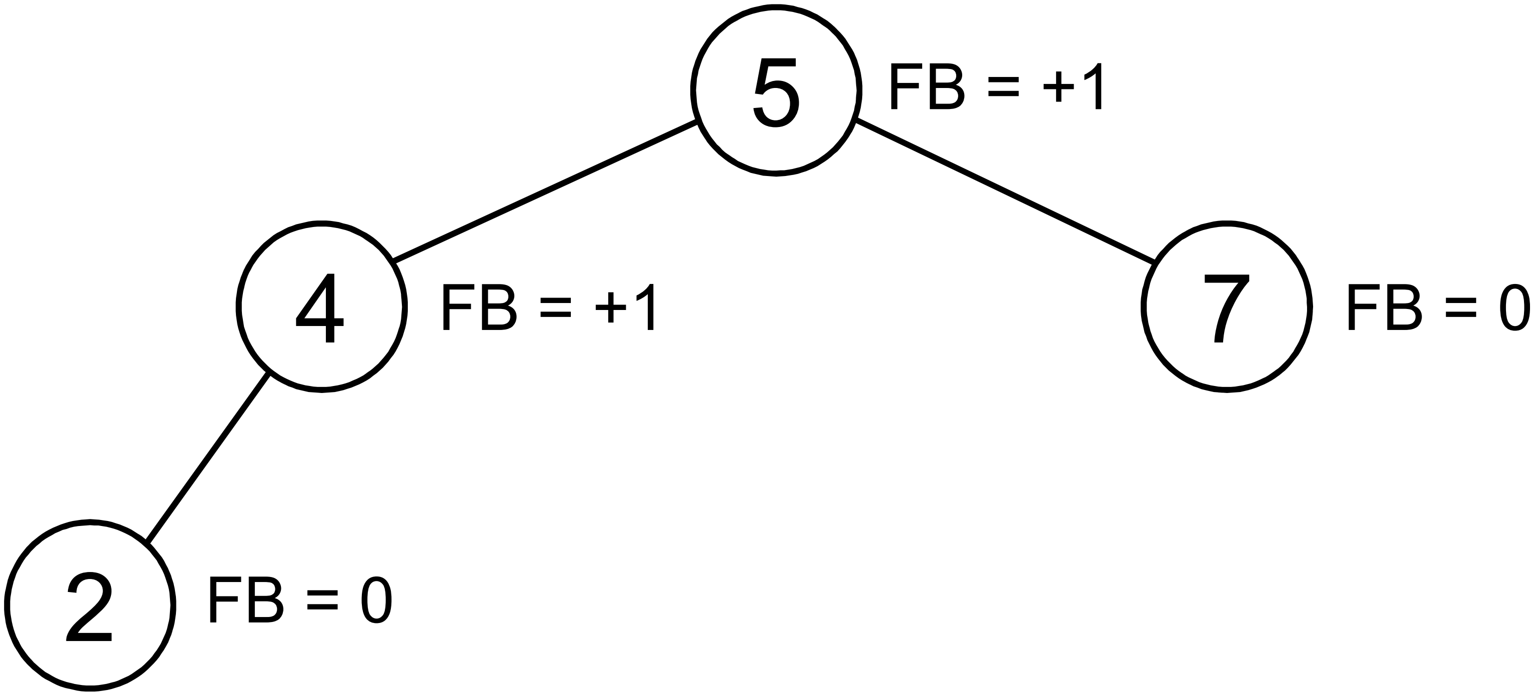
- As quatro operações de balanceamento podem ser aplicadas arbitrariamente.
- Porém, têm por fim balancear uma ABB.
- Portanto, sua aplicação deve ficar condicionada ao fator de balanceamento de cada nó, de acordo com as seguintes regras:

Regras para balanceamento

R1) Seja *x* um nó de uma árvore *A* AVL.

Se $-1 \leq fb(x) \leq 1$, então a árvore com raiz em *x* está **balanceada**, não sendo necessária a aplicação de nenhuma operação de rotação.

Regras para balanceamento: R1



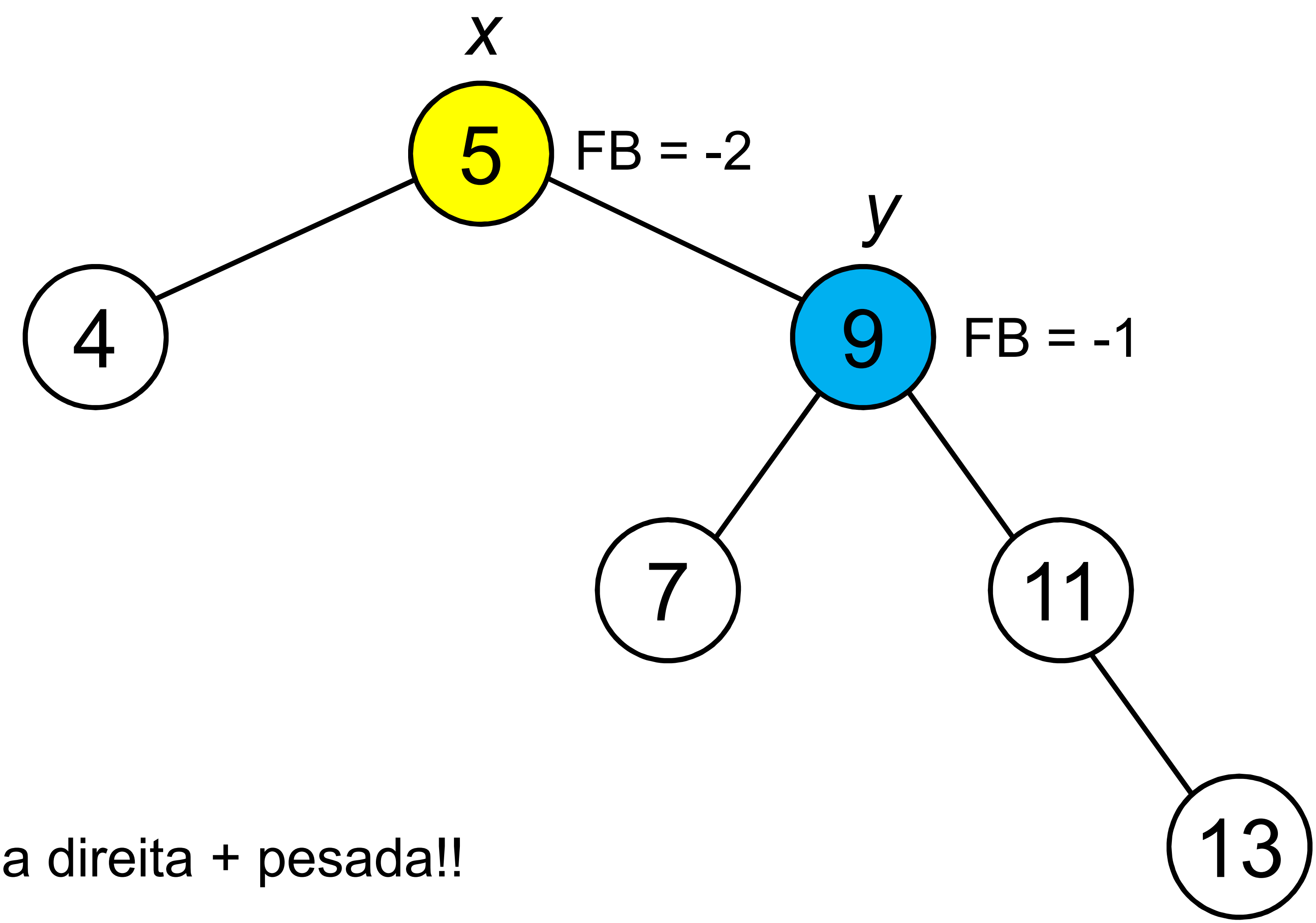
Regras para balanceamento

R2) Seja *z* o nó raiz da subárvore esquerda de *x*. Se *fb*(*x*) < -1, a árvore com raiz em *x* está **mais pesada à direita**, o que requer uma operação de rotação à esquerda para torná-la balanceada. Então,

(a) Se *fb*(*z*) ≤ 0, será aplicada a operação RSE;

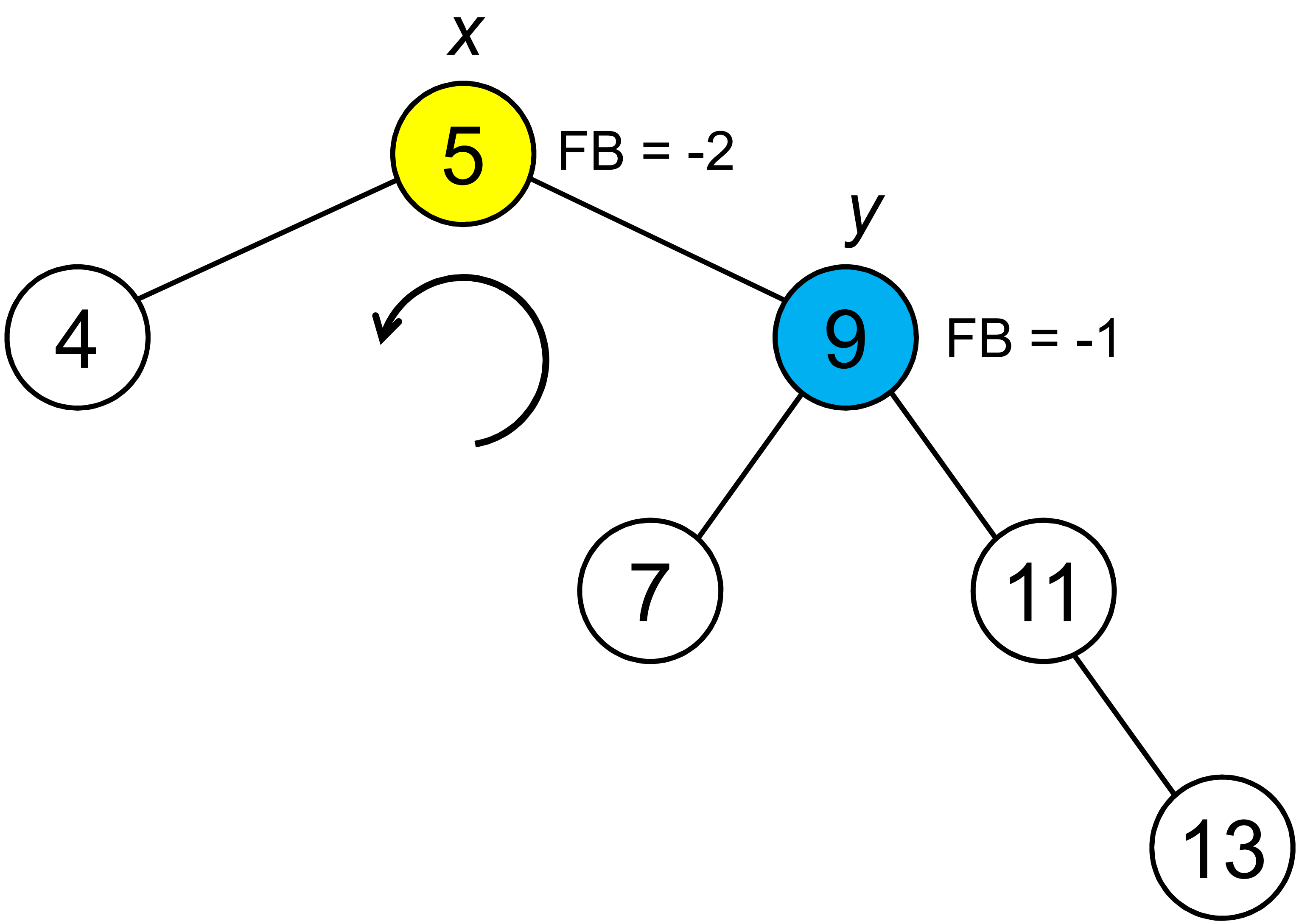
(b) Se *fb*(*z*) > 0, será aplicada a operação RDE.

R2a: ($fb(x)$ < 0 e $fb(y)$ <= 0) ➔ RSE

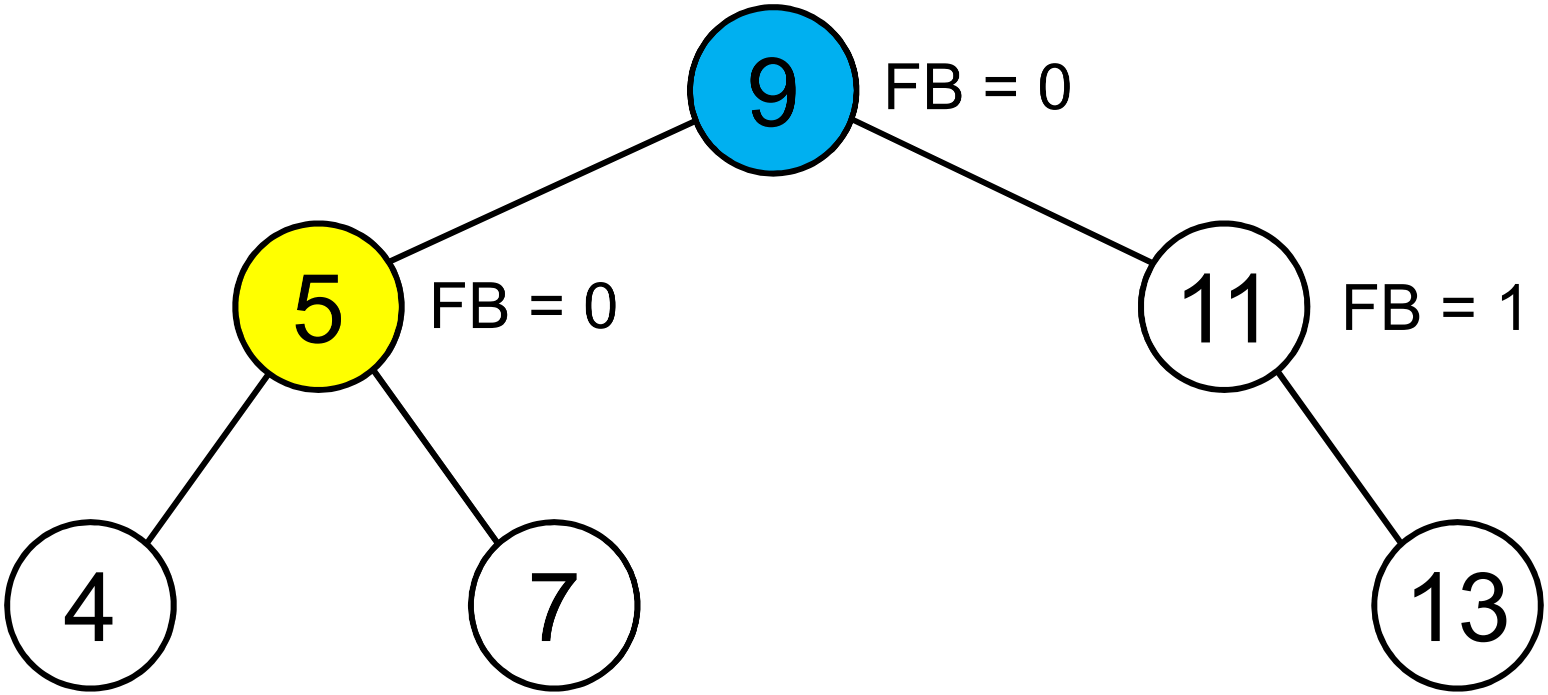


Subarvore a direita + pesada!!

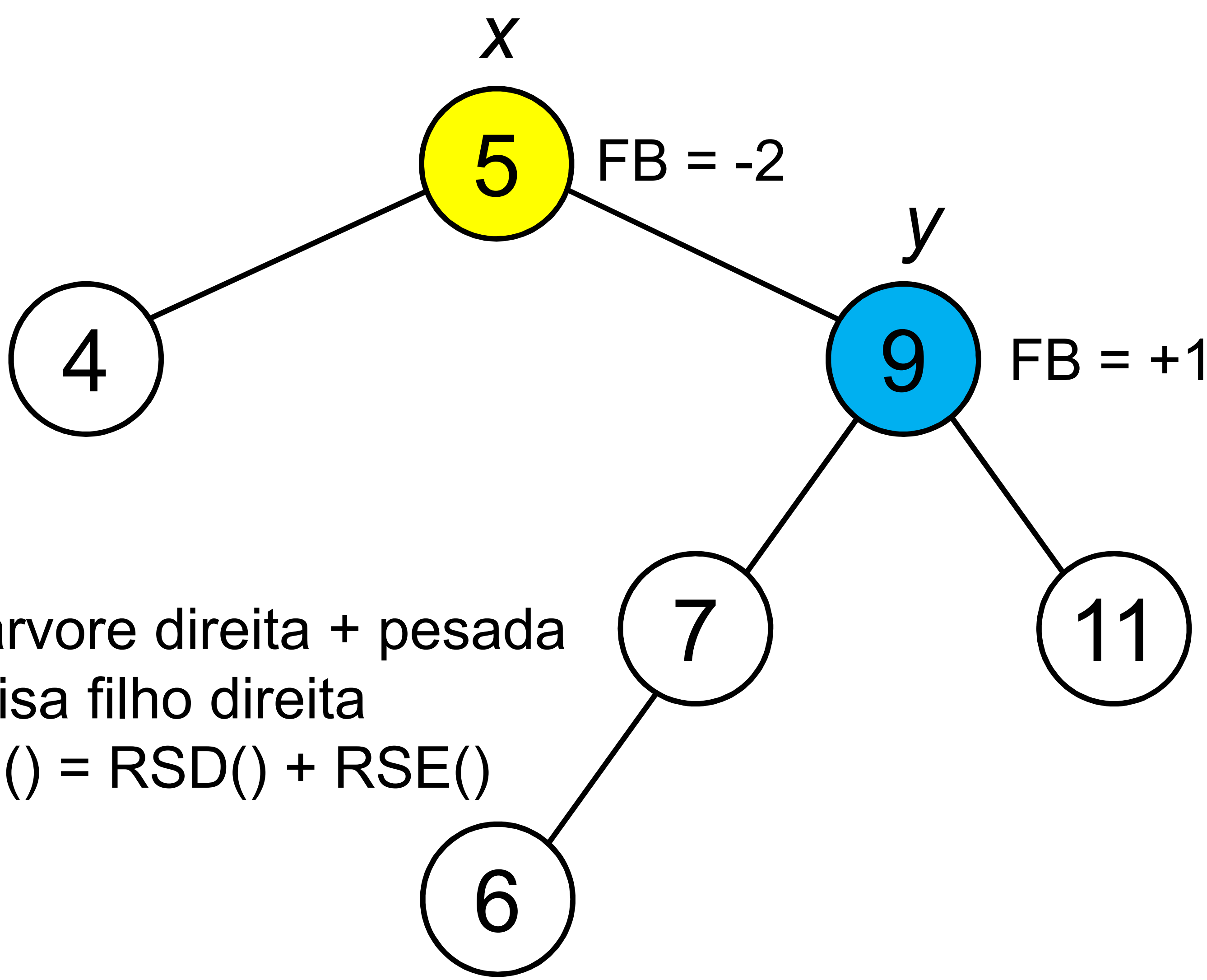
R2a: (*fb*(*x*) < 0 e *fb*(*y*) <= 0) ➔ RSE



R2a: (*fb*(*x*) < 0 e *fb*(*y*) <= 0) ➔ RSE

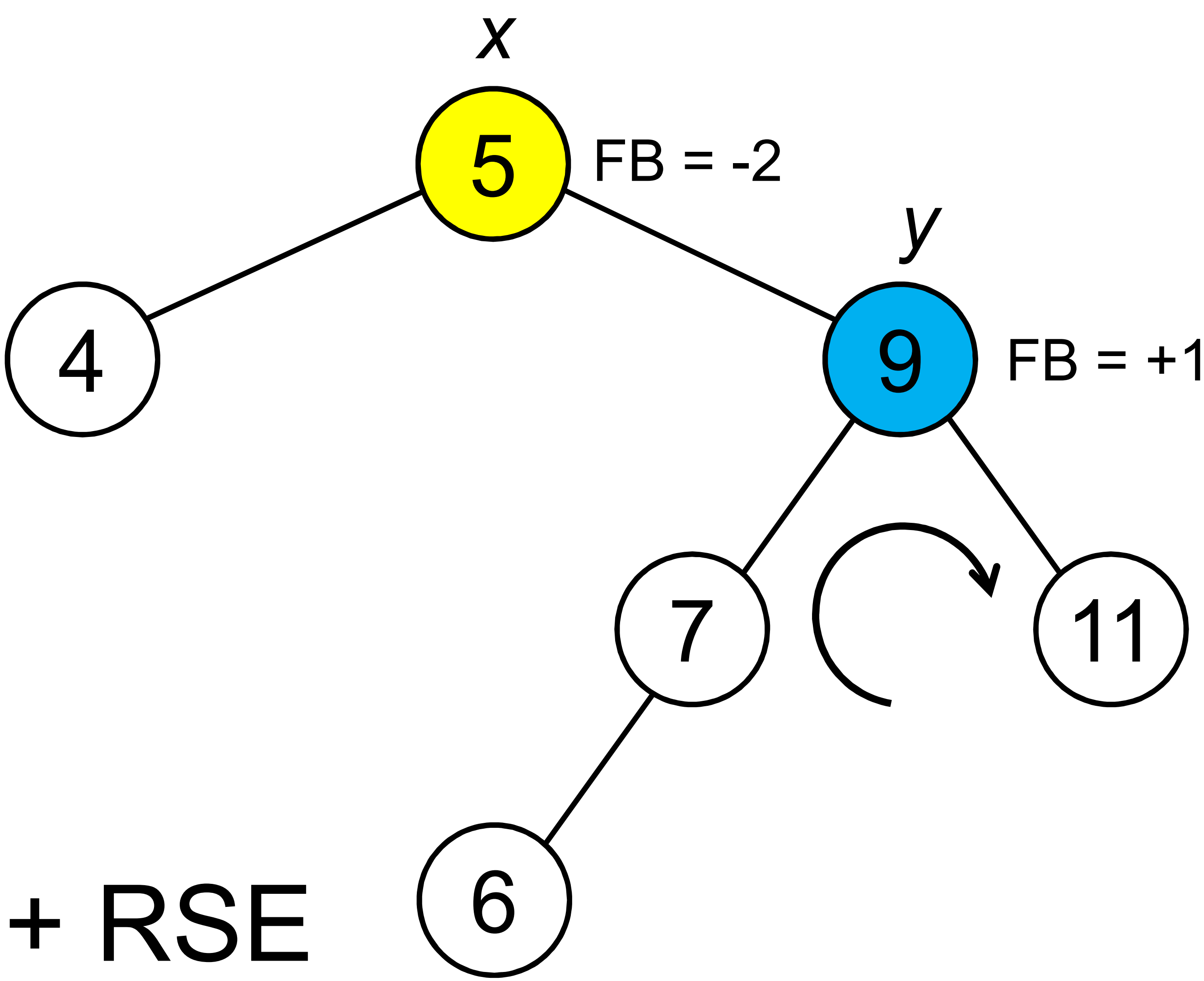


R2b: (*fb*(*x*) < 0 e *fb*(*y*) > 0) → RDE



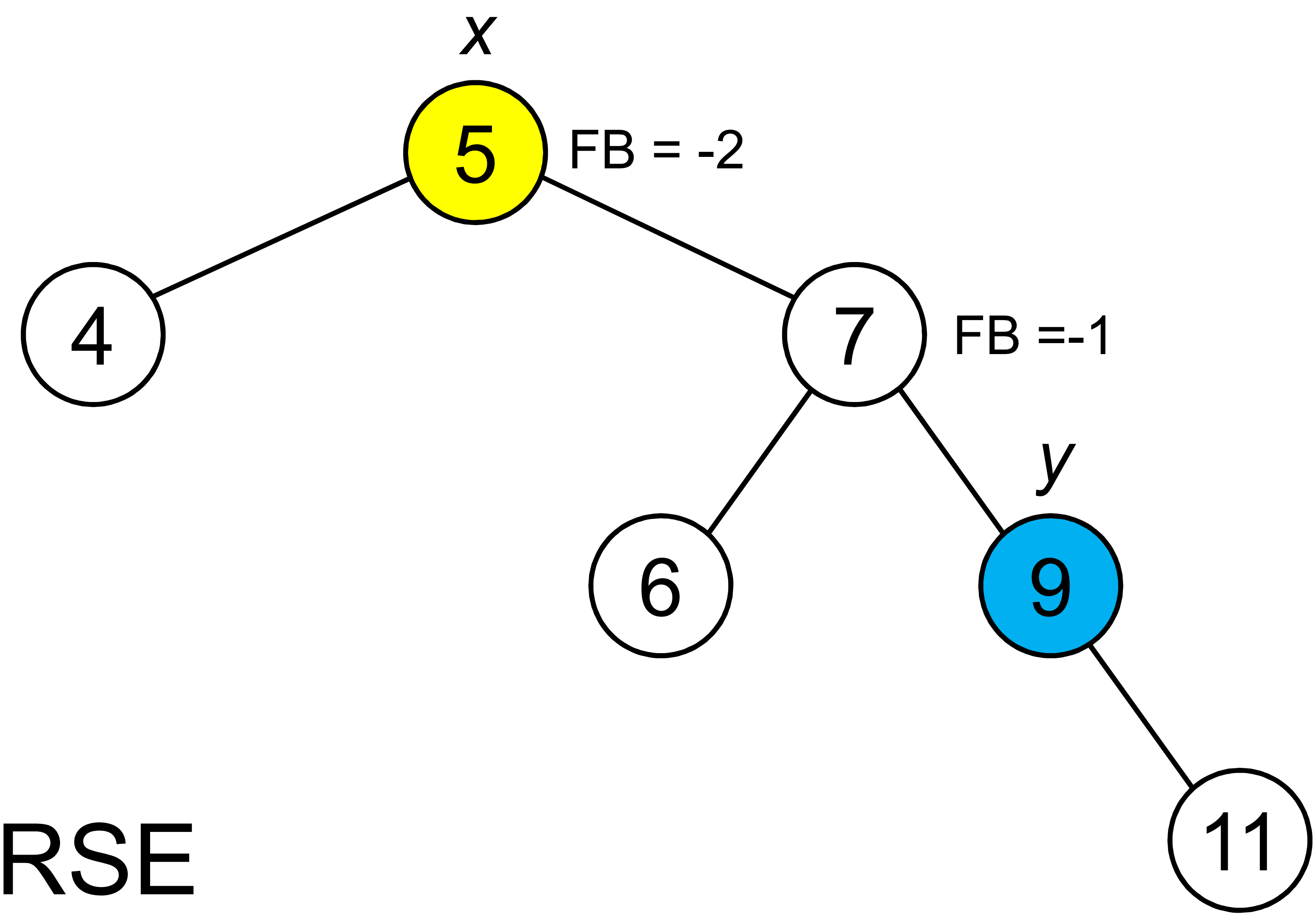
Fb(x) < 0 → subarvore direita + pesada
→ analisa filho direita
Fb(y) > 0 → RDE() = RSD() + RSE()

R2b: (*fb*(*x*) < 0 e *fb*(*y*) > 0) ➔ RDE



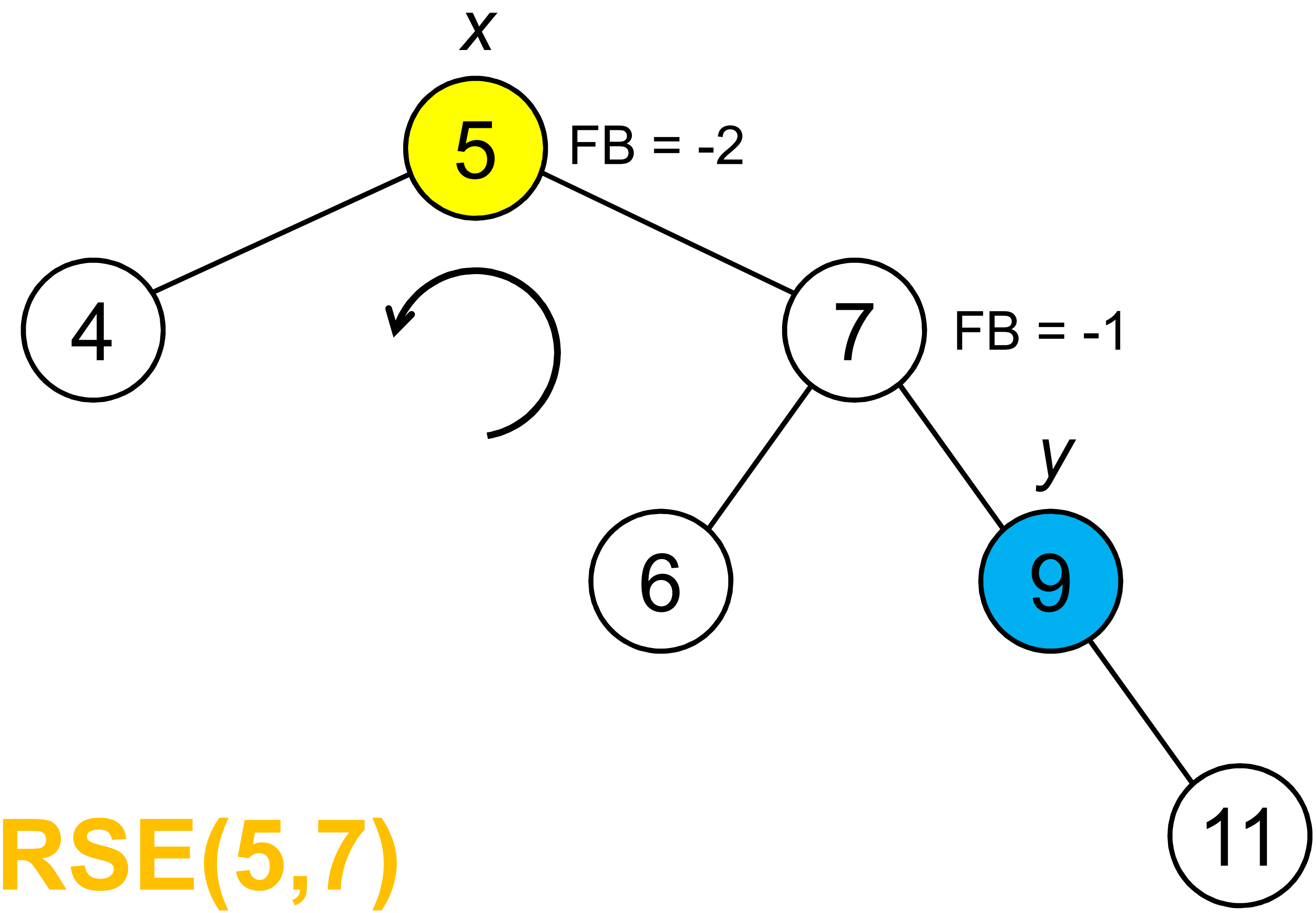
RSD(7,9) + RSE

R2b: (*fb*(*x*) < 0 e *fb*(*y*) > 0) ➔ RDE



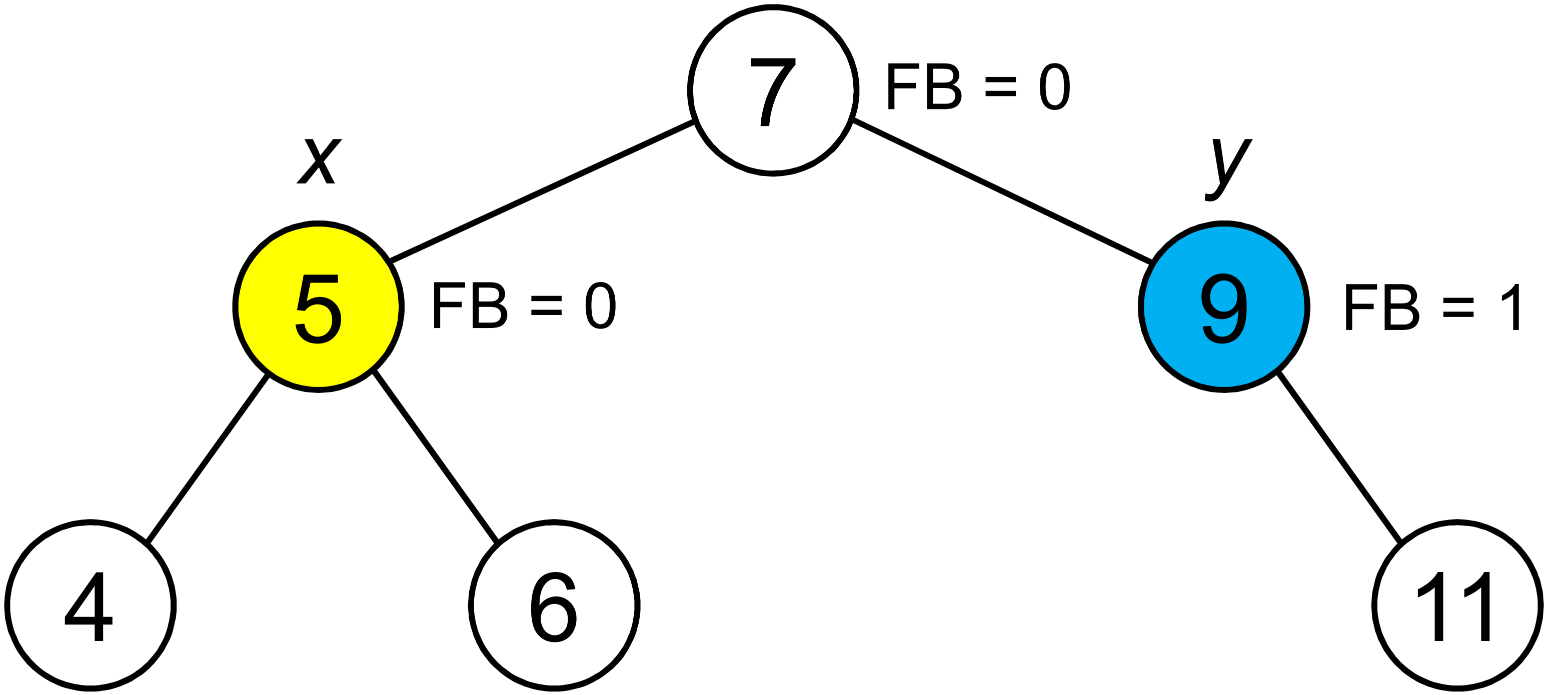
RSD + RSE

R2b: (*fb*(*x*) < 1 e *fb*(*7*) < 0) → RDE



RSD + **RSE(5,7)**

R2b: RDE



RSD + RSE

Regras para balanceamento

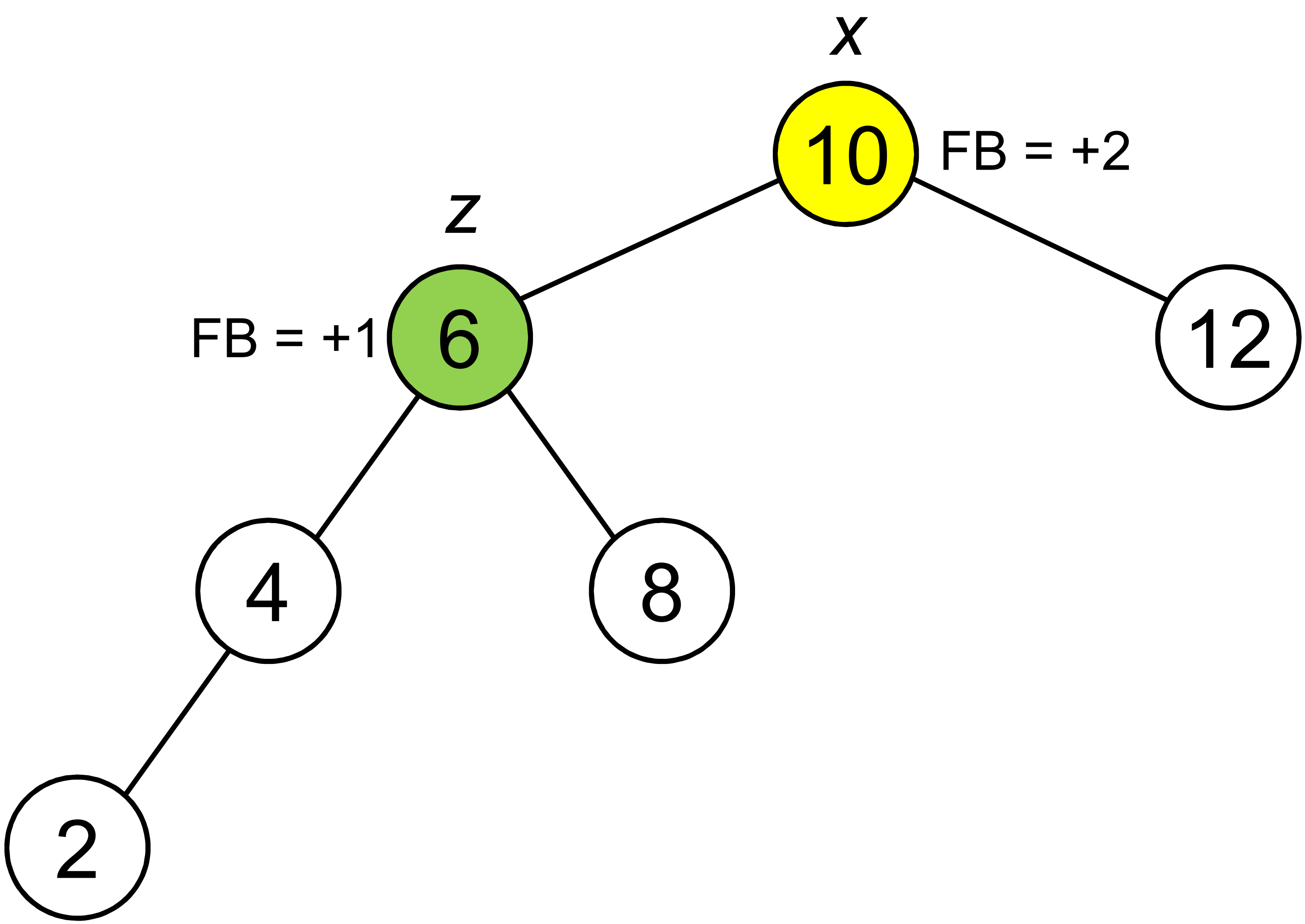
R3) Seja *y* o nó raiz da subárvore direita de *x*.

Se $fb(x) > 1$, a árvore com raiz em *x* está **mais pesada à esquerda**, o que requer uma operação de rotação à direita para torná-la balanceada. Então,

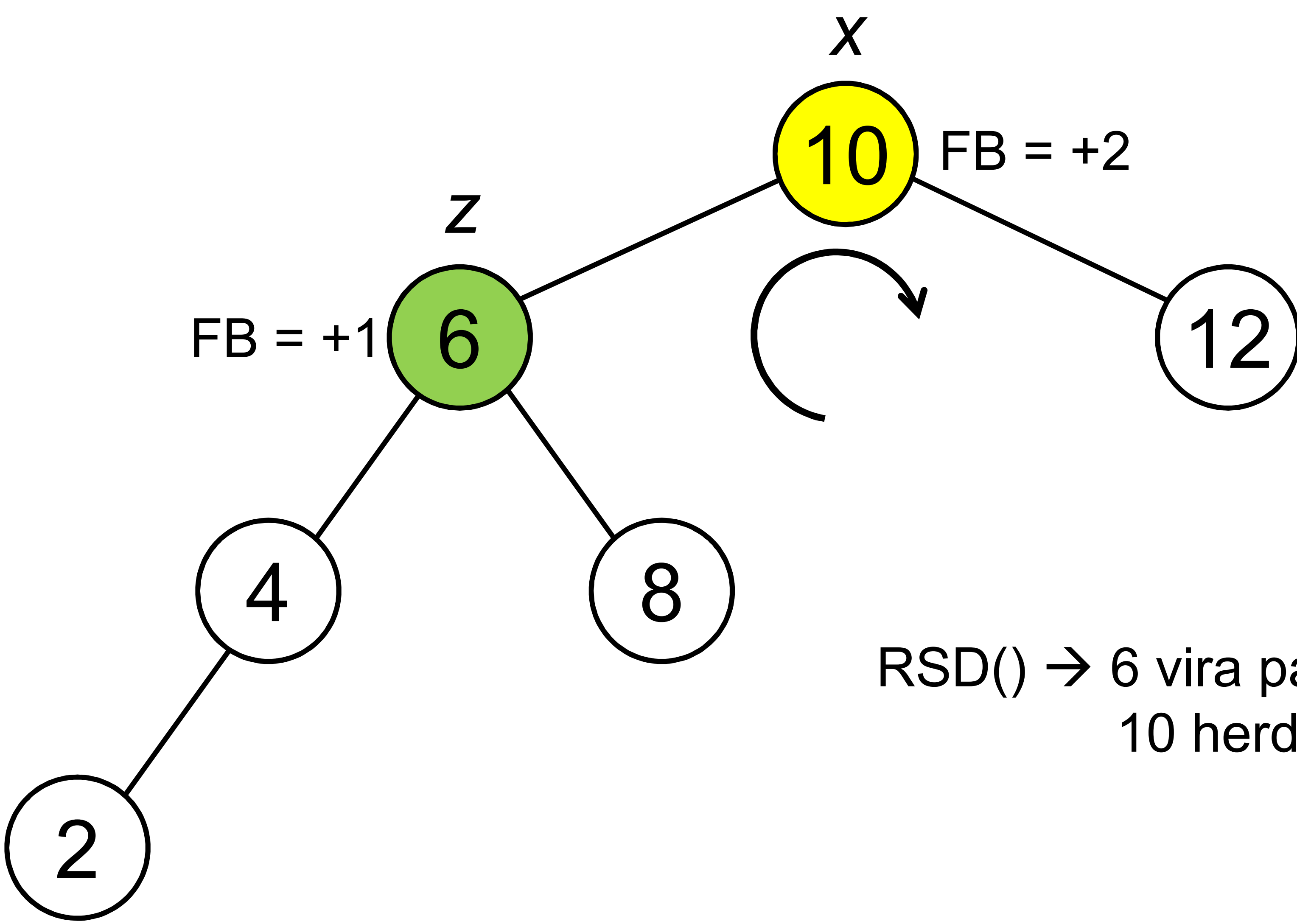
(a) se $fb(y) \geq 0$, será aplicada a operação RSD;

(b) se $fb(y) < 0$, será aplicada a operação RDD.

R3a: ($fb(x) > 0$ e $fb(z) >= 0$) ➔ RSD

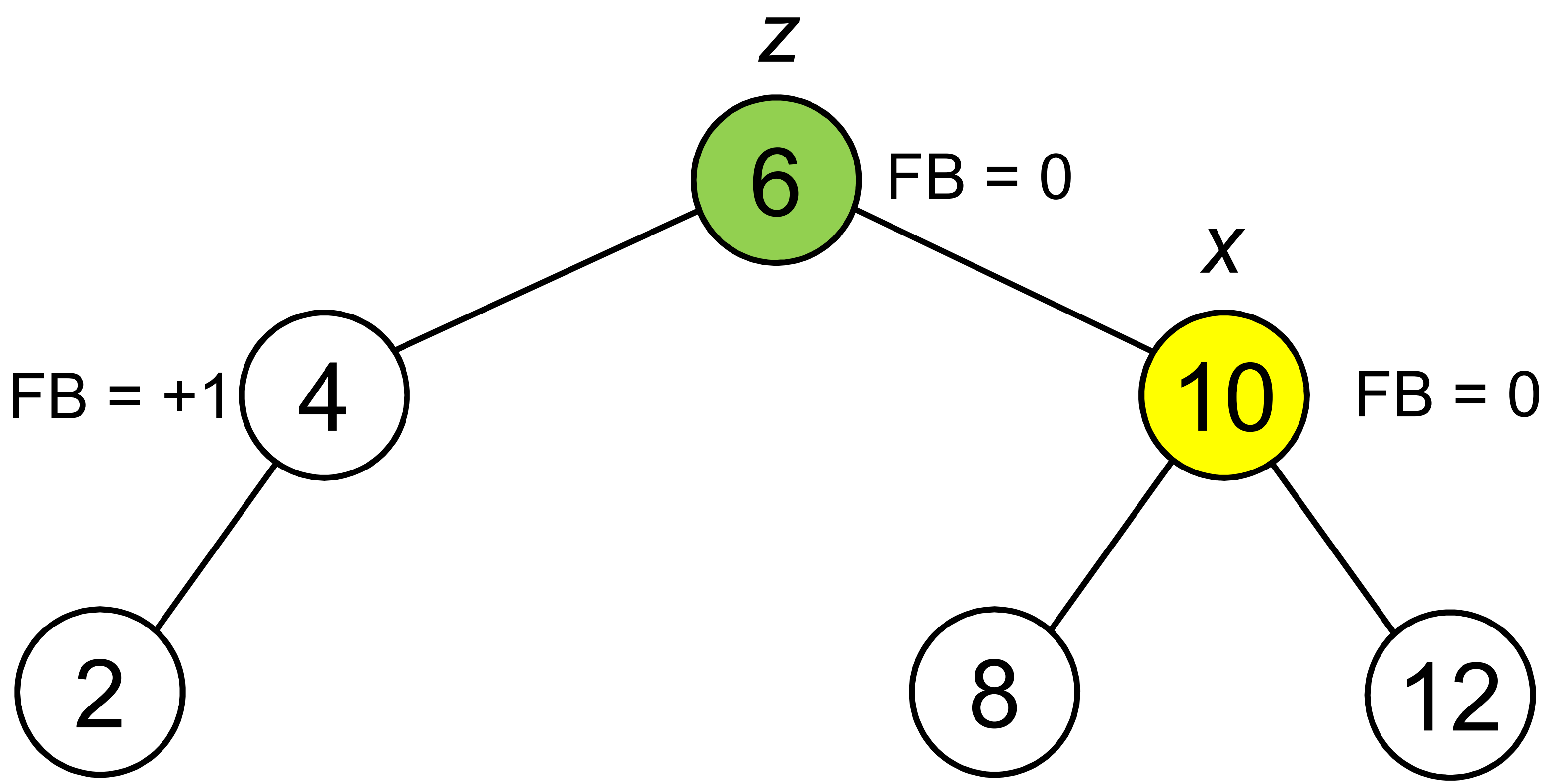


R3a: (*fb*(*x*) > 0 e *fb*(*z*) >= 0) ➔ RSD

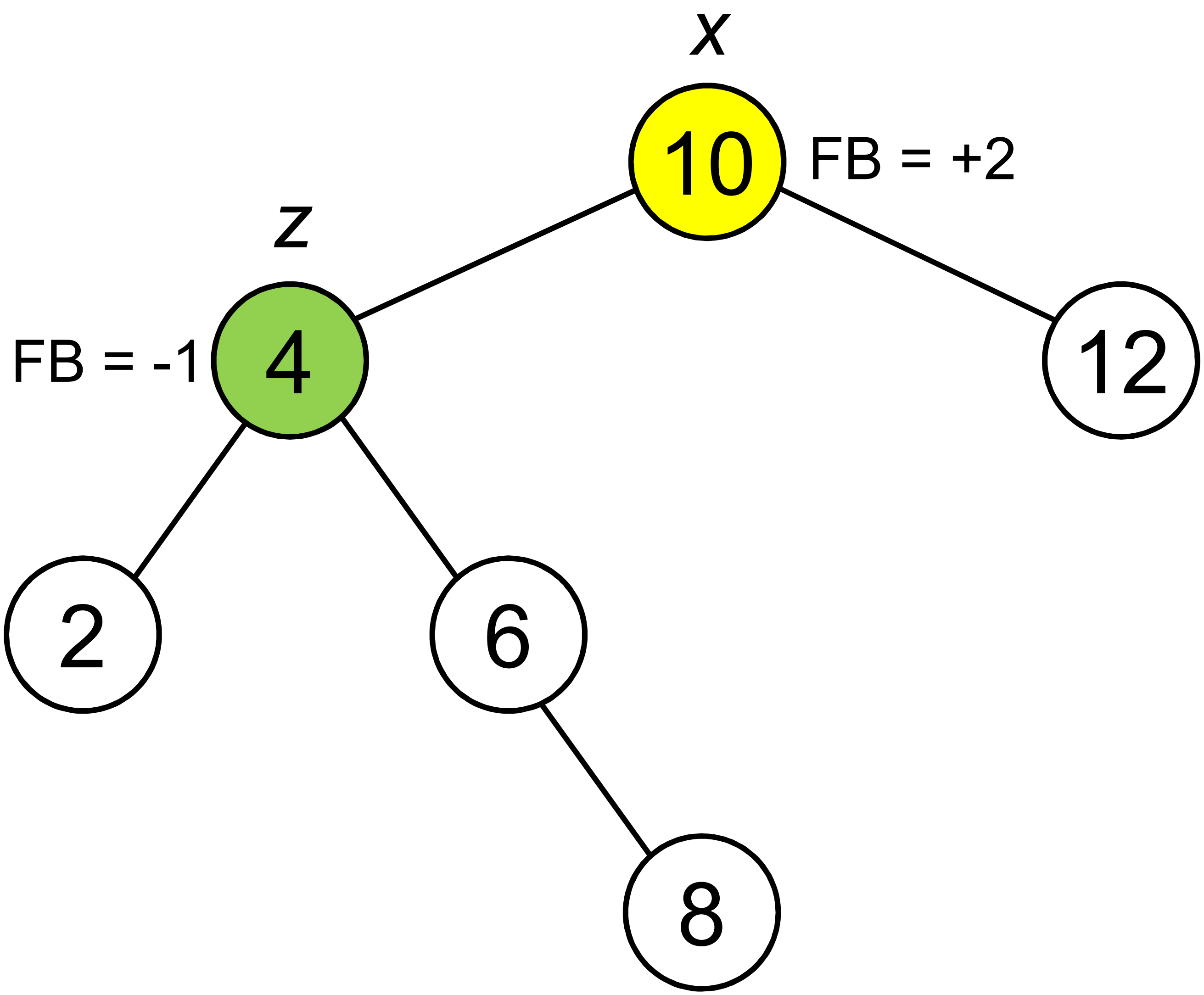


RSD() → 6 vira pai do 10
10 herda filho direita 6

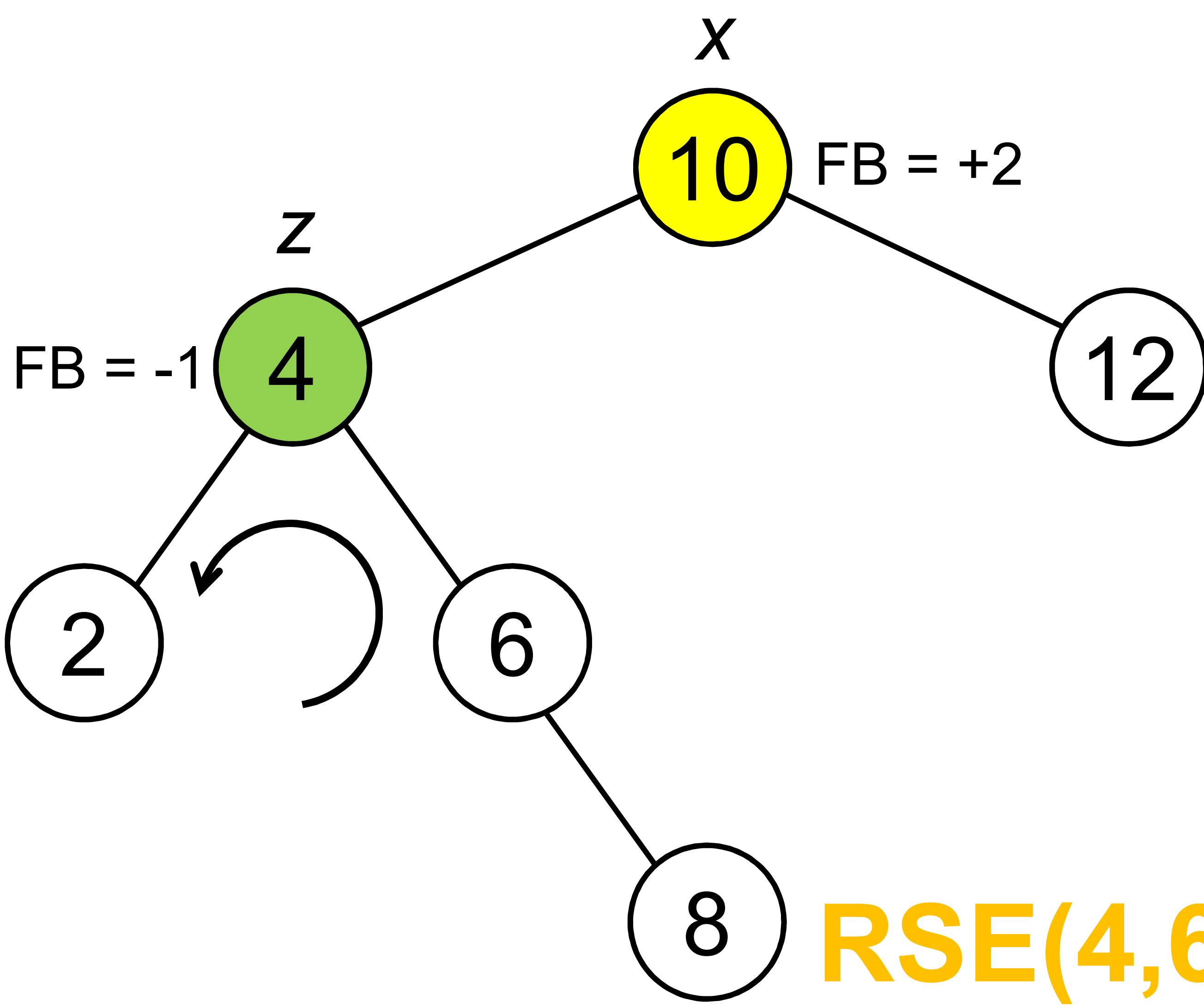
R3a: (*fb*(*x*) > 0 e *fb*(*z*) >= 0) ➔ RSD



R3b: (*fb*(*10*) > 1 e *fb*(*4*) <= -1) ➔ RDD

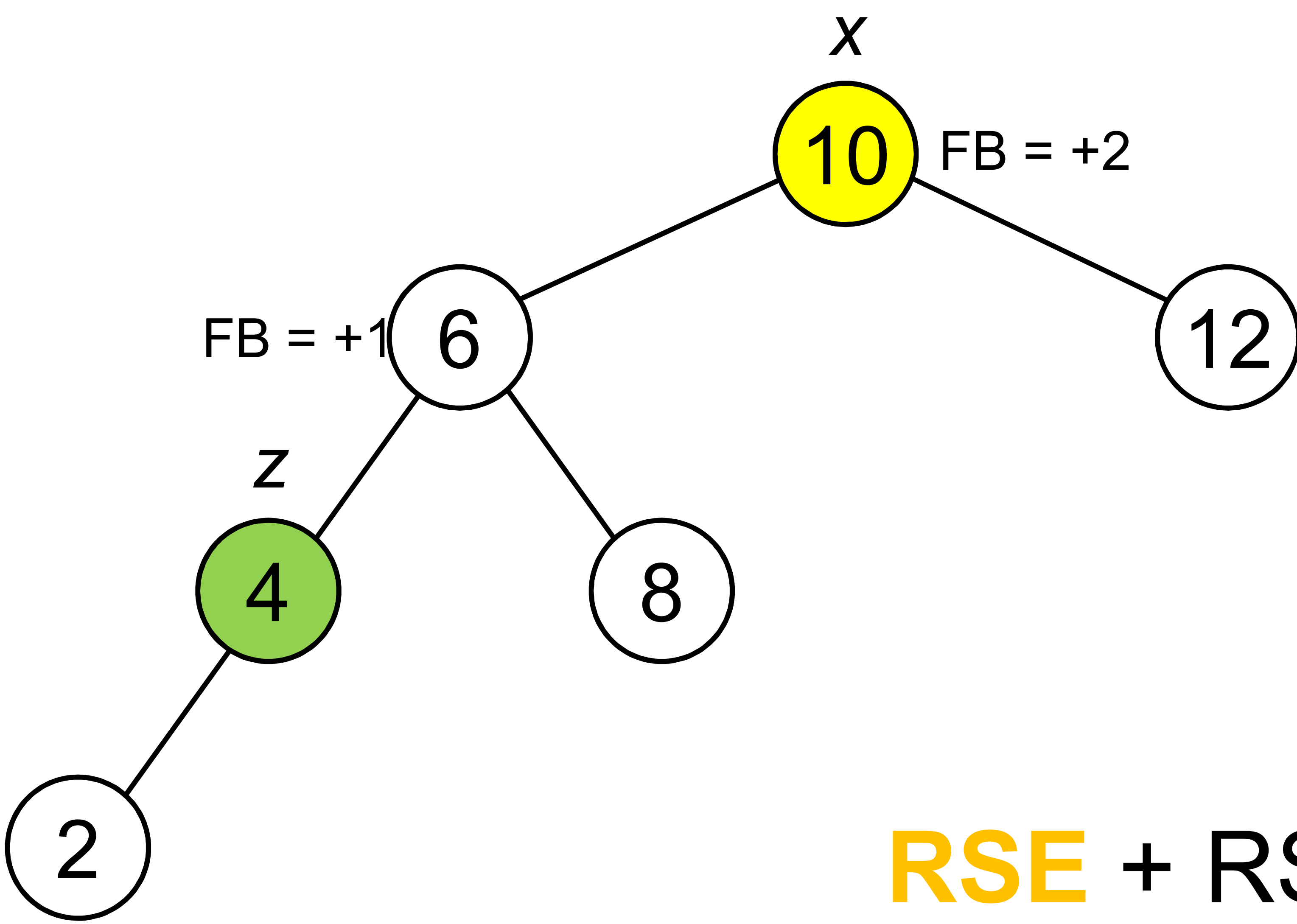


R3b: (*fb*(*10*) > 1 e *fb*(*4*) <= -1) ➔ RDD



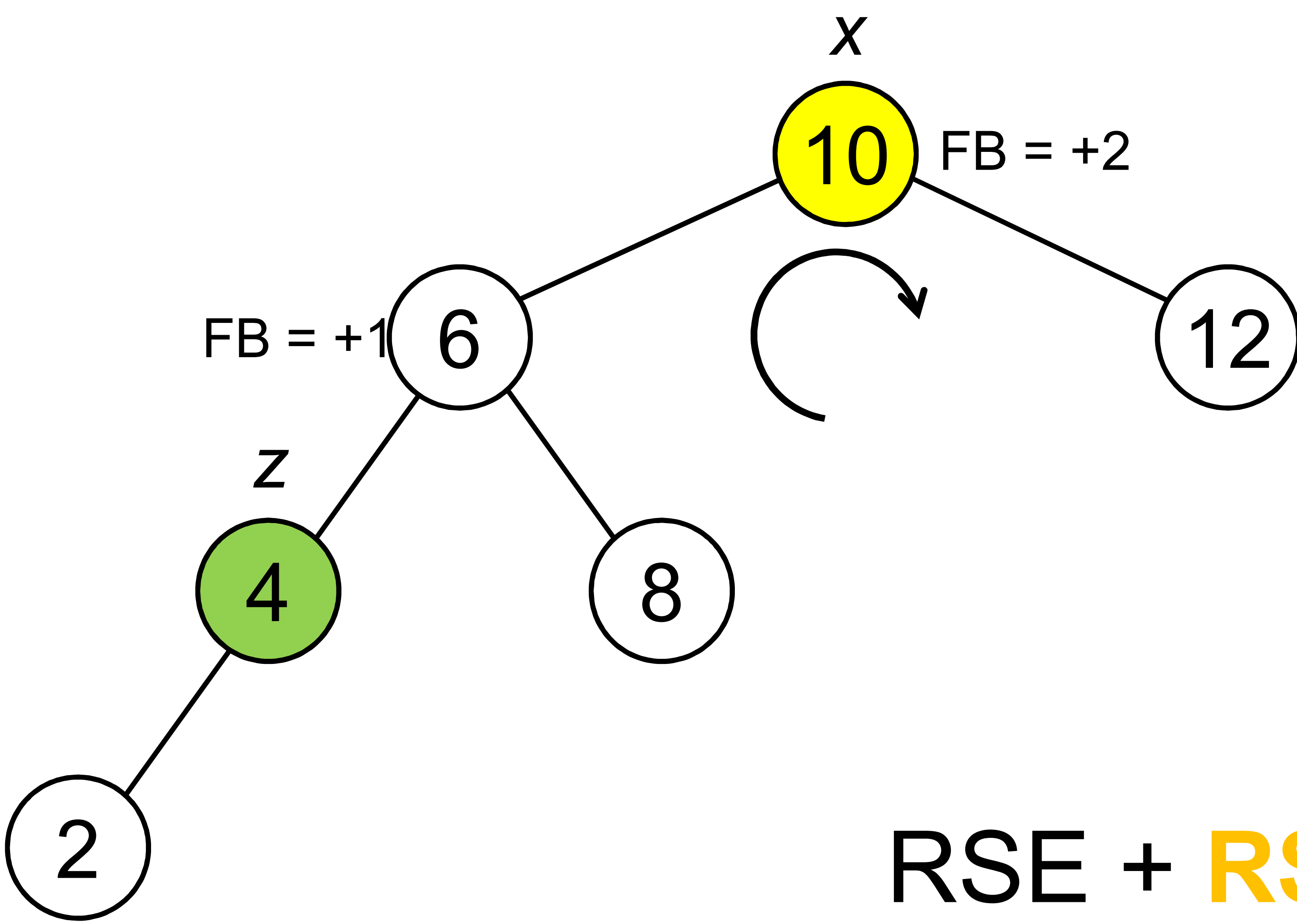
RSE(4,6) + RSD

R3b: (*fb*(*10*) > 1 e *fb*(*6*) > 0) ➔ RDD



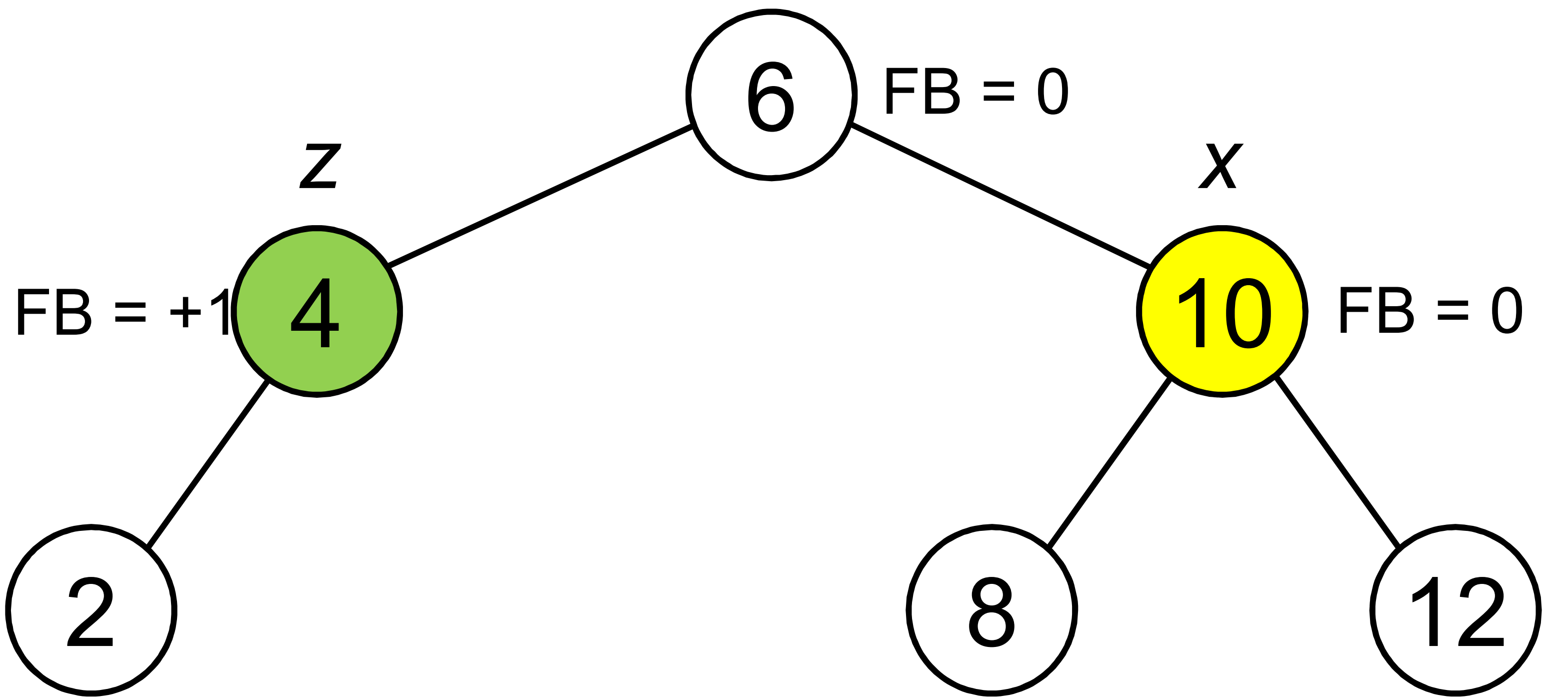
RSE + RSD(6,10)

R3b: (*fb*(*10*) > 1 e *fb*(*6*) > 0) ➔ RDD



RSE + **RSD(6,10)**

R3b: () ➔ RDD

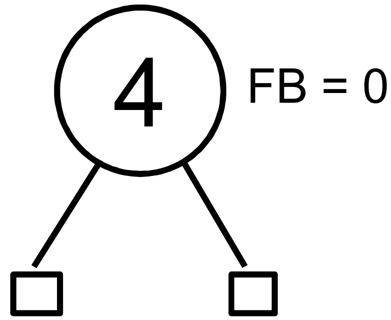


RSE + RSD

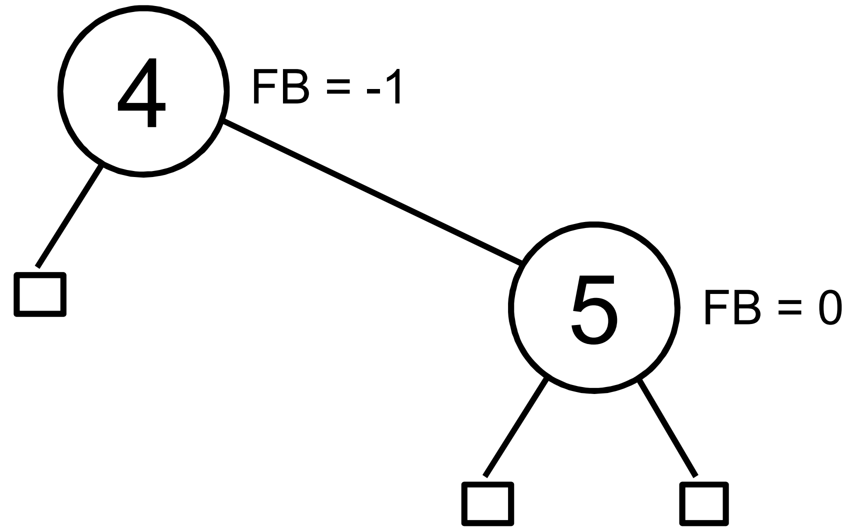
Construção de uma árvore AVL

- Vamos aplicar as regras e operações para construir uma árvore AVL para a sequência de valores {4, 5, 7, 2, 1, 3, 6}.

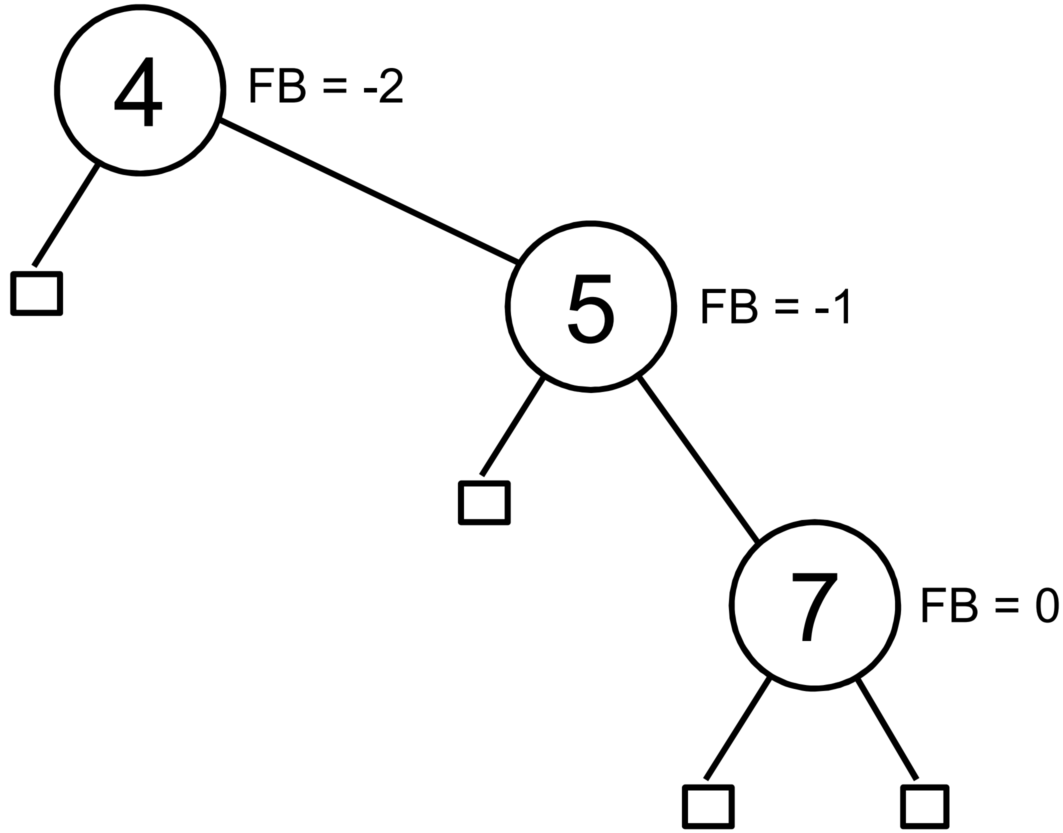
Construção de uma árvore AVL



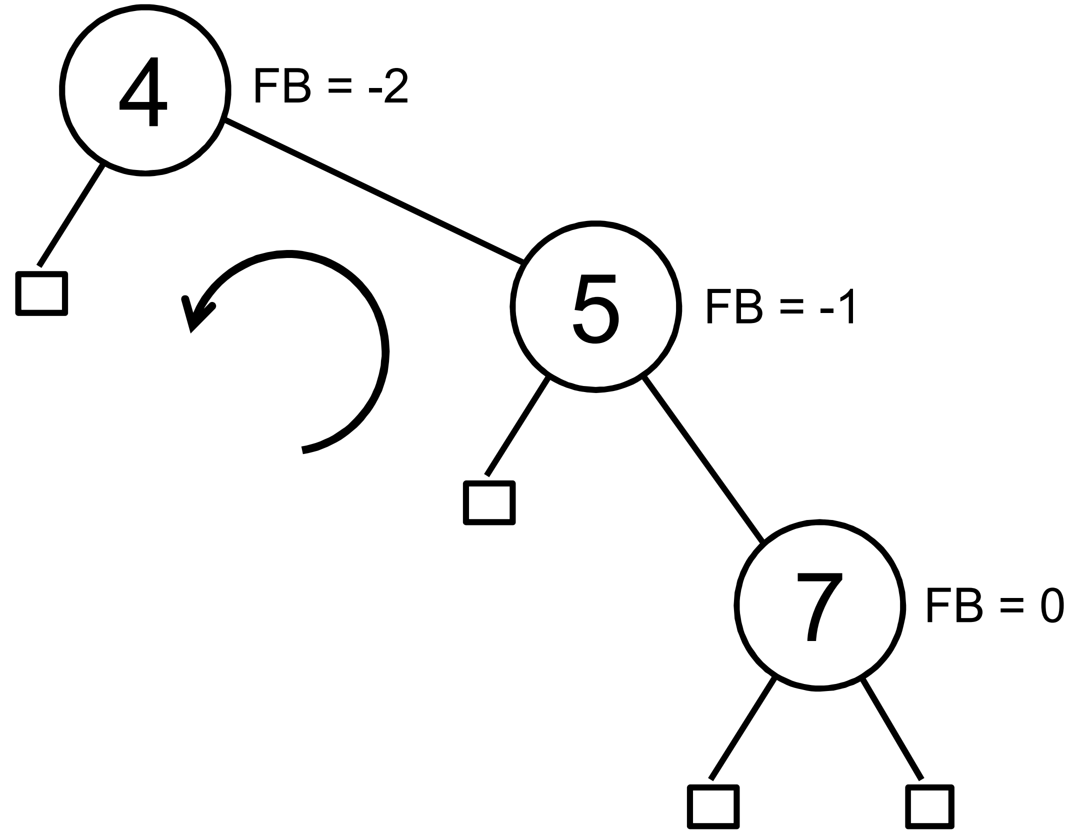
Construção de uma árvore AVL



Construção de uma árvore AVL

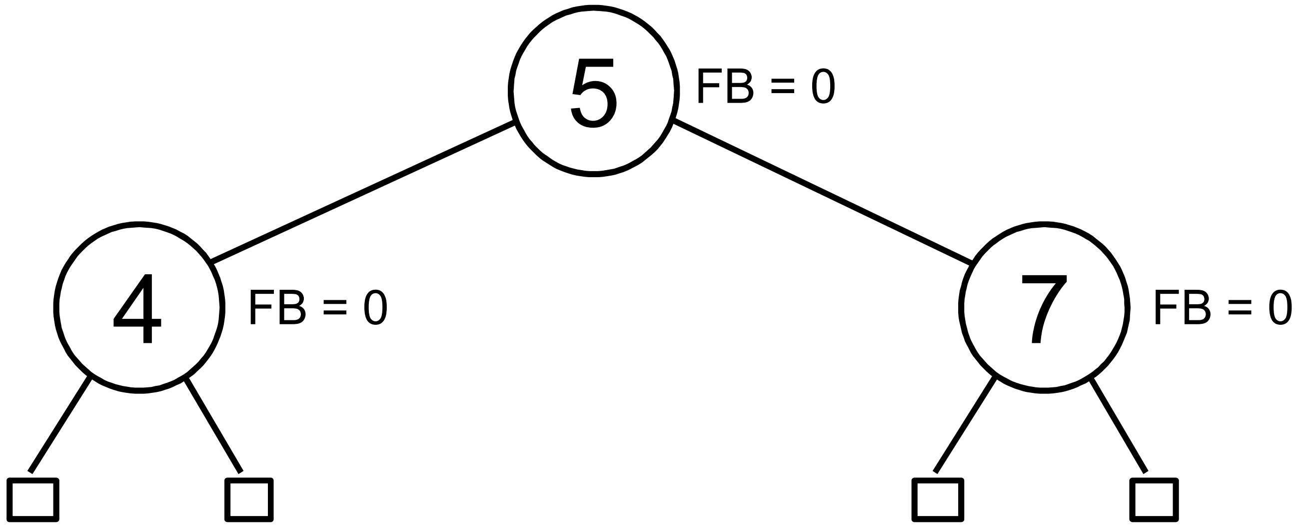


Construção de uma árvore AVL

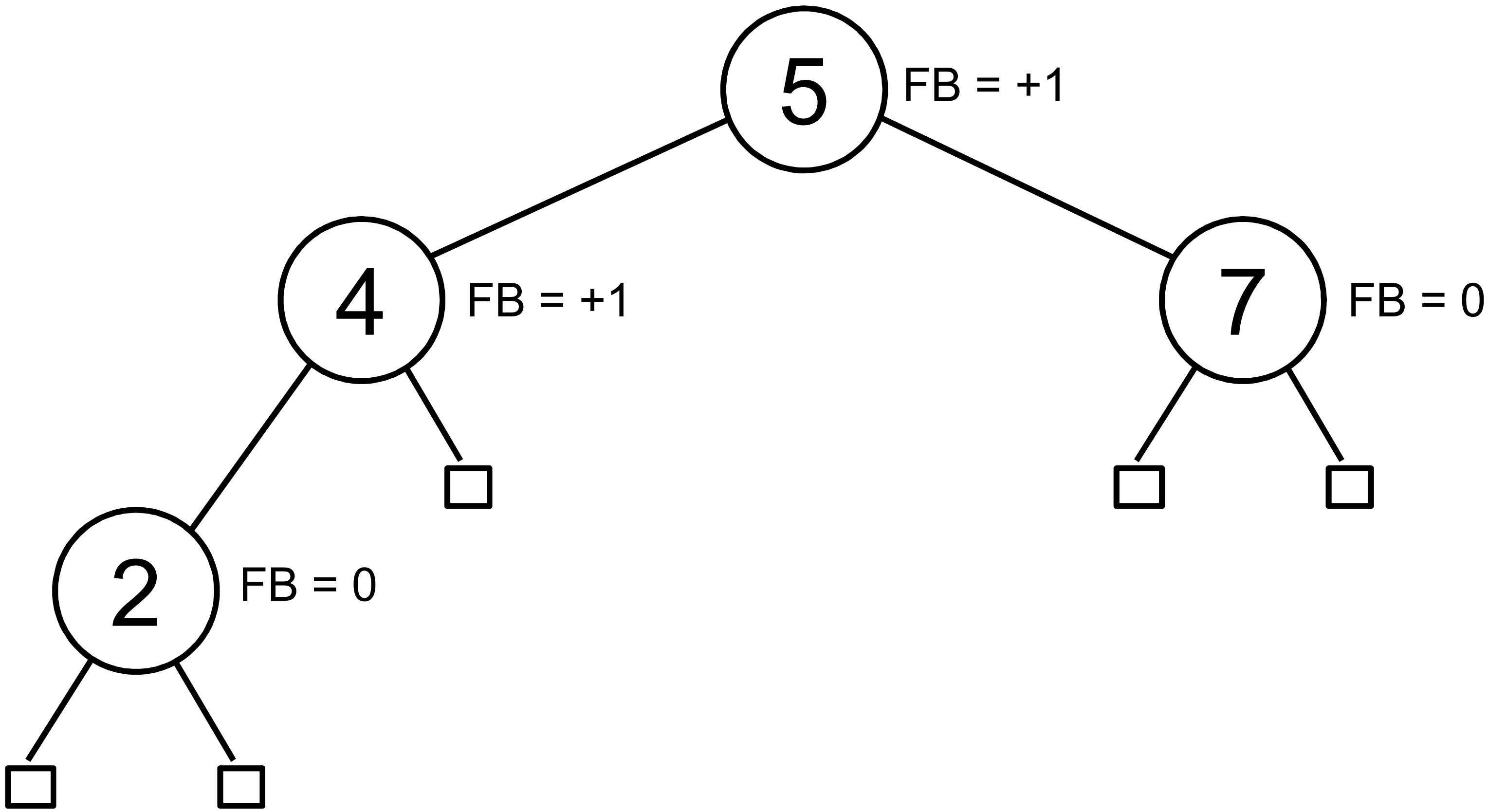


aplicar RSE

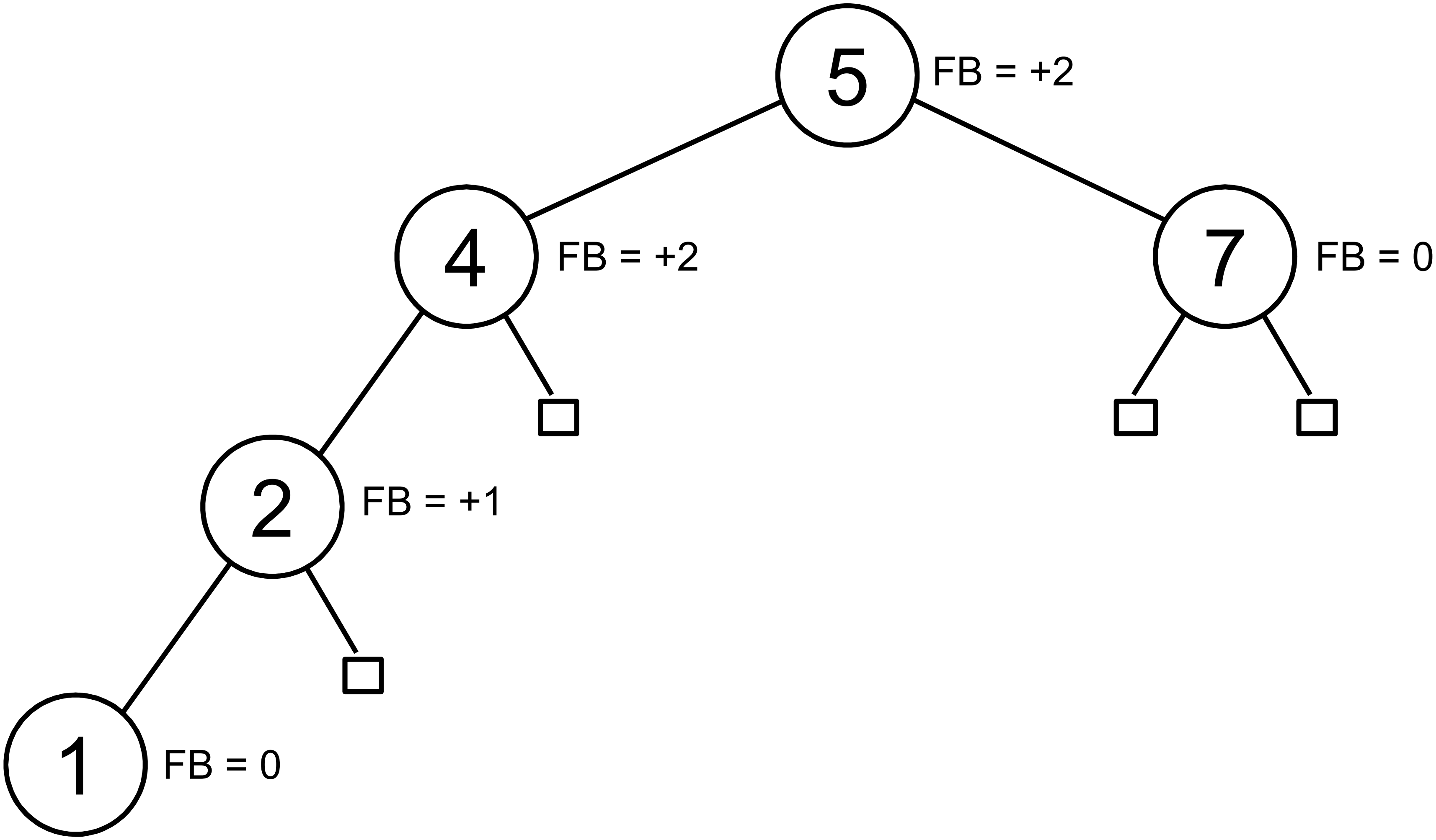
Construção de uma árvore AVL



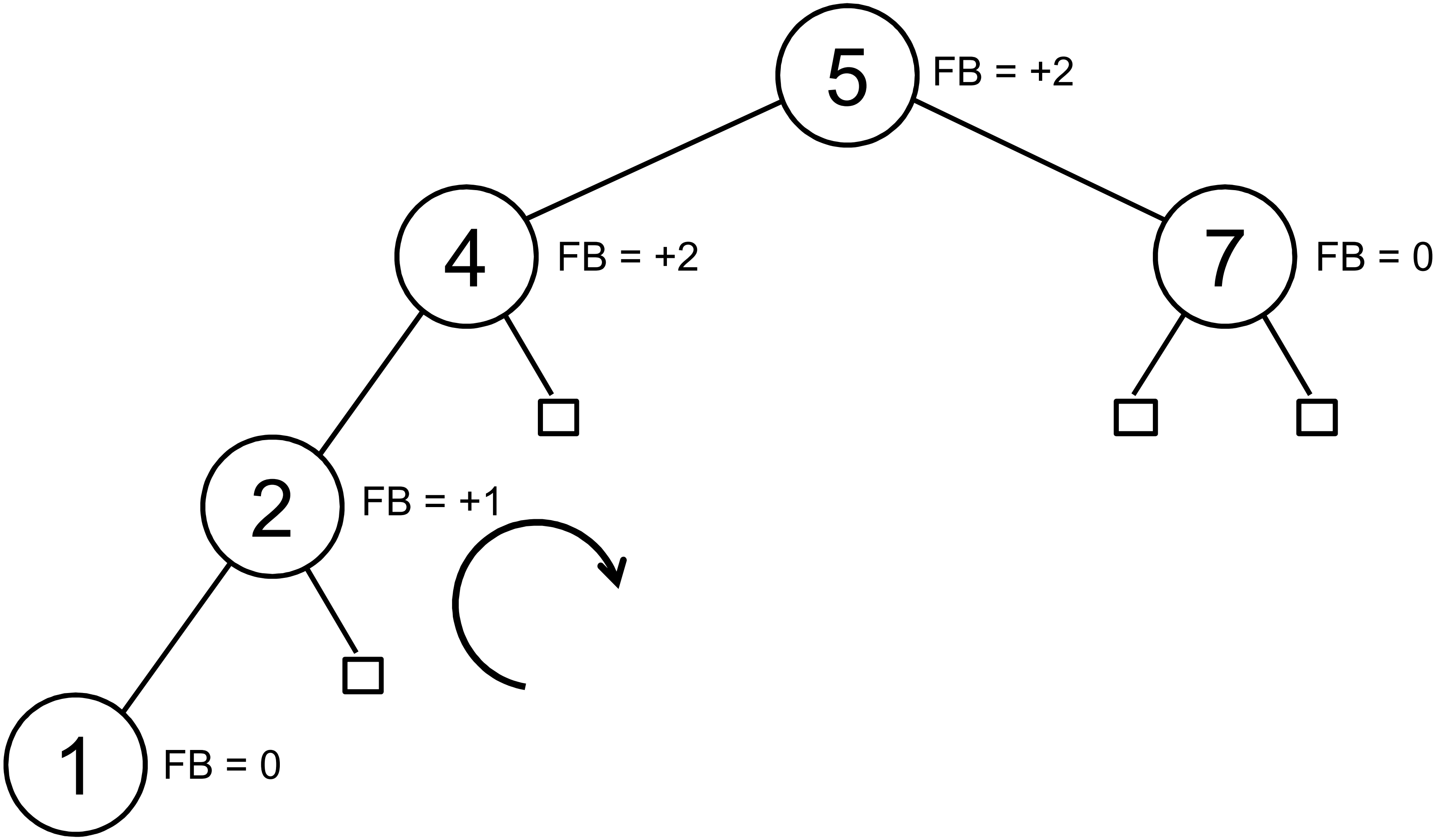
Construção de uma árvore AVL



Construção de uma árvore AVL

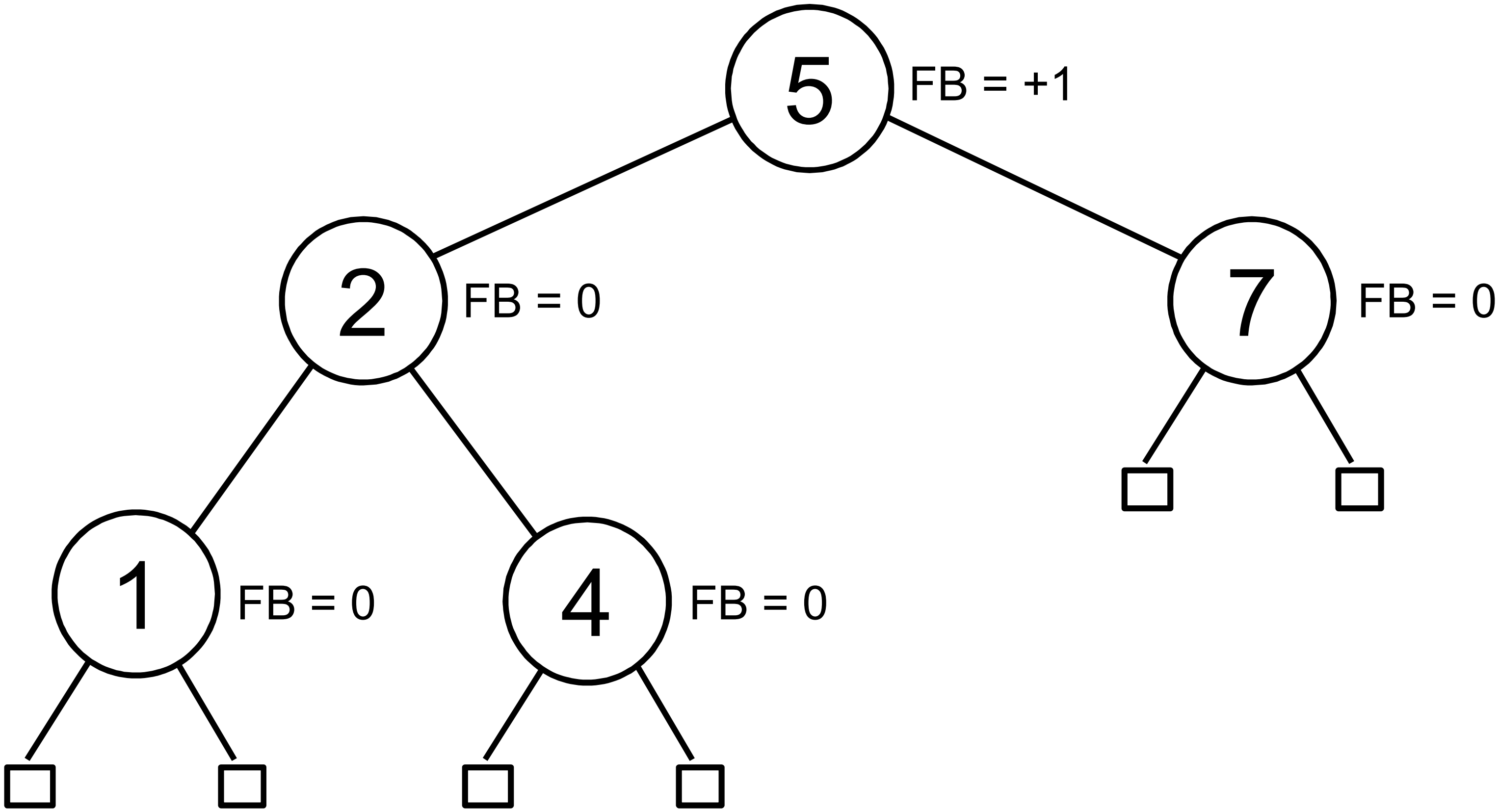


Construção de uma árvore AVL

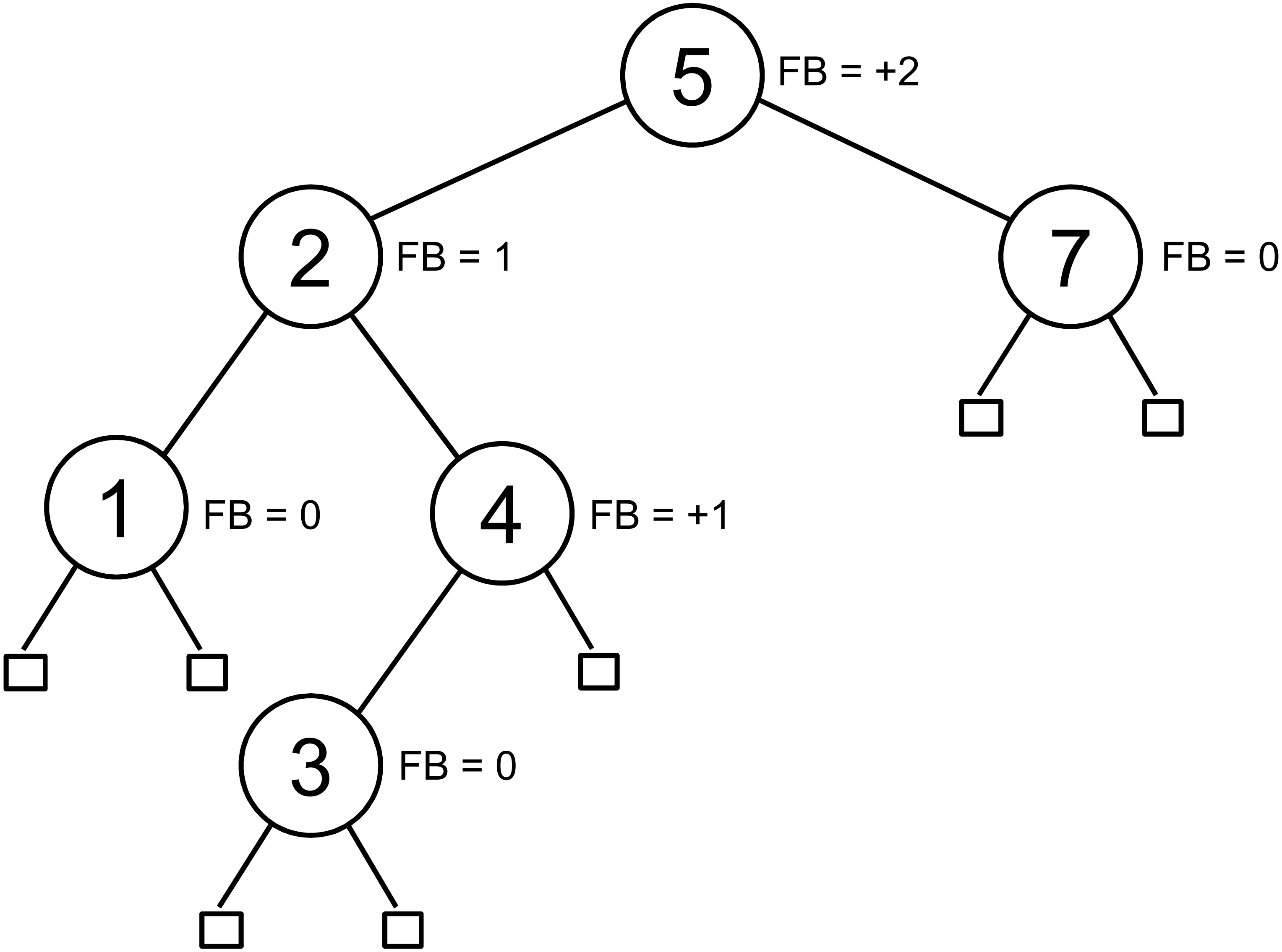


aplicar RSD

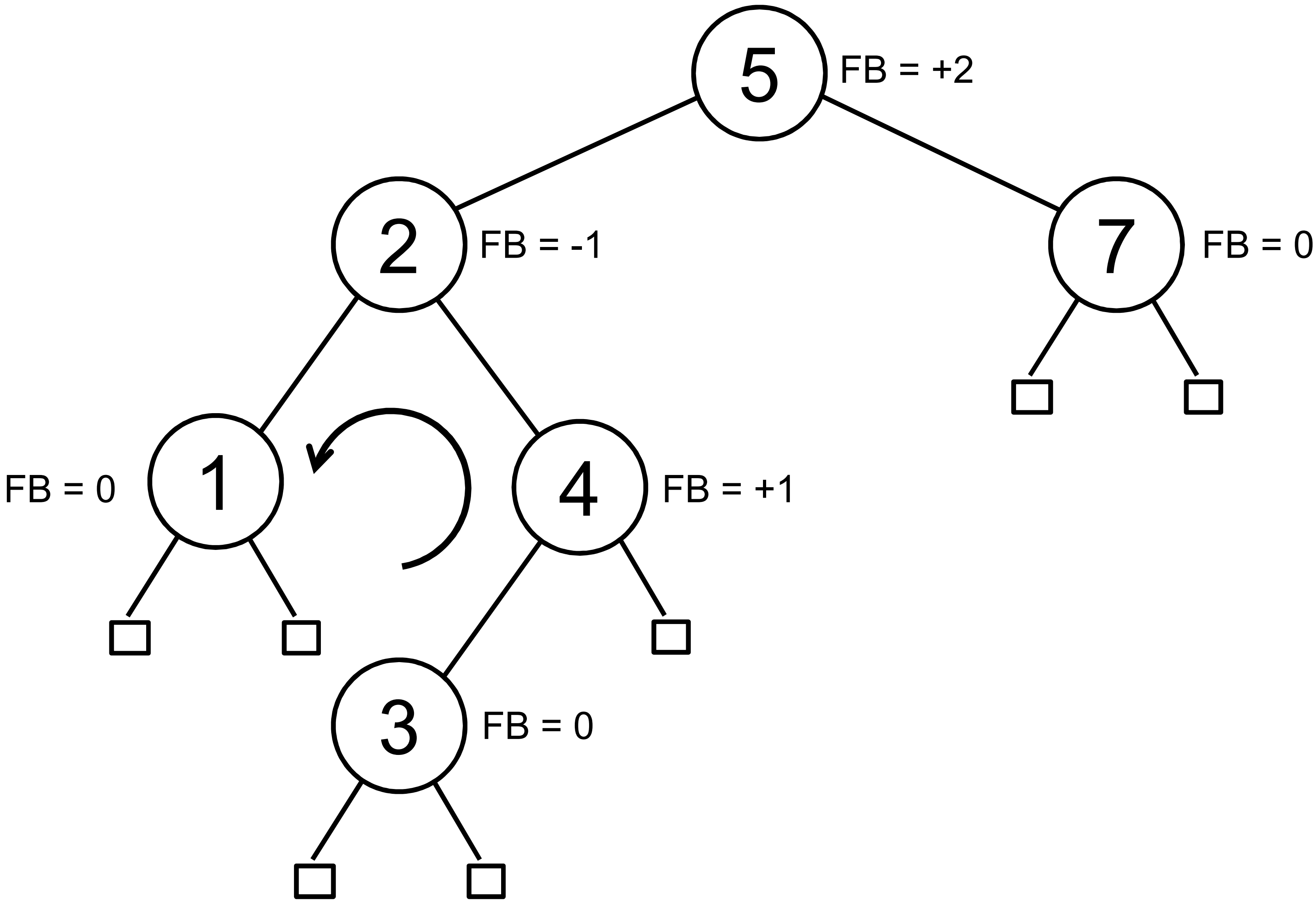
Construção de uma árvore AVL



Construção de uma árvore AVL

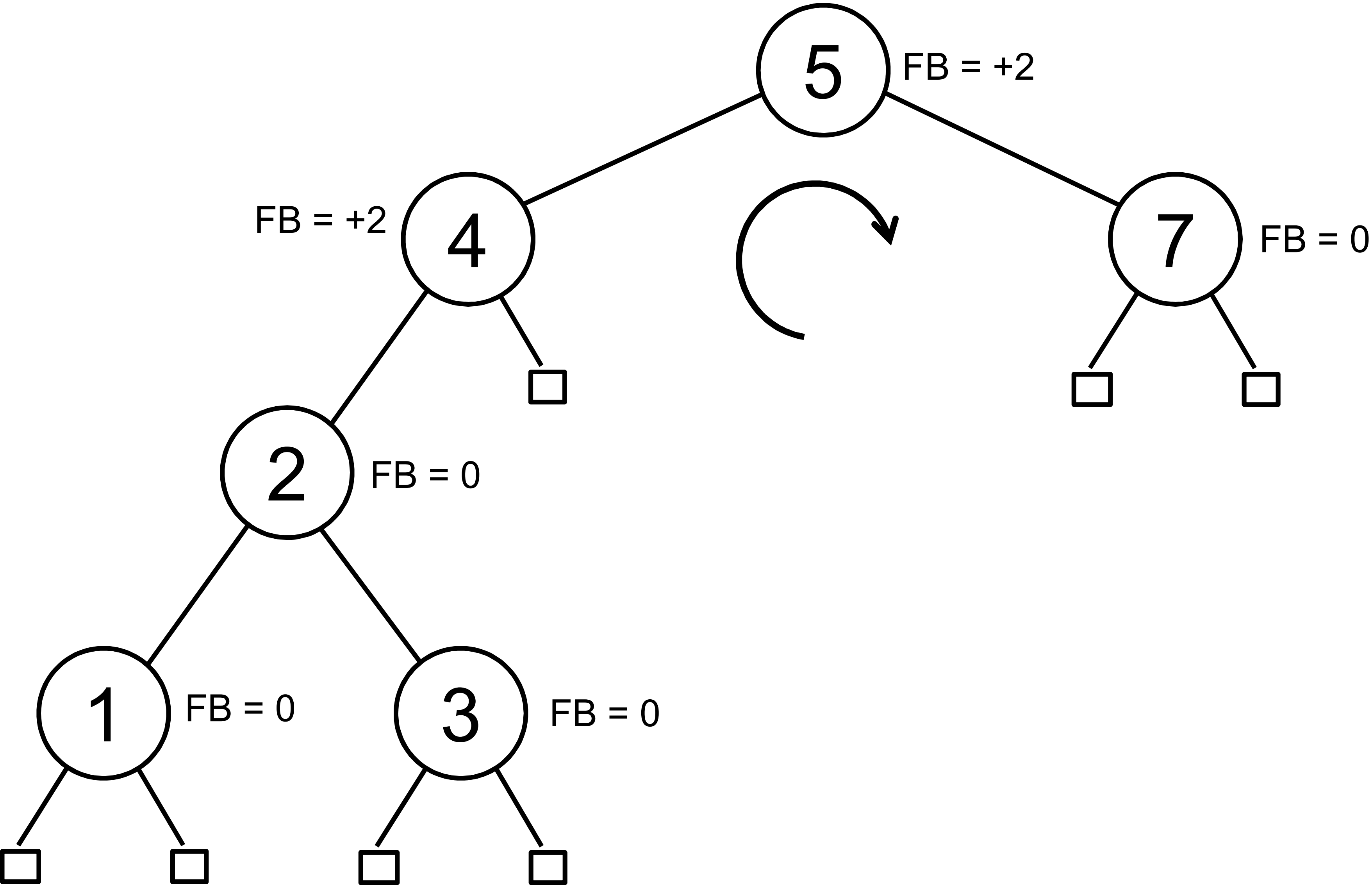


Construção de uma árvore AVL



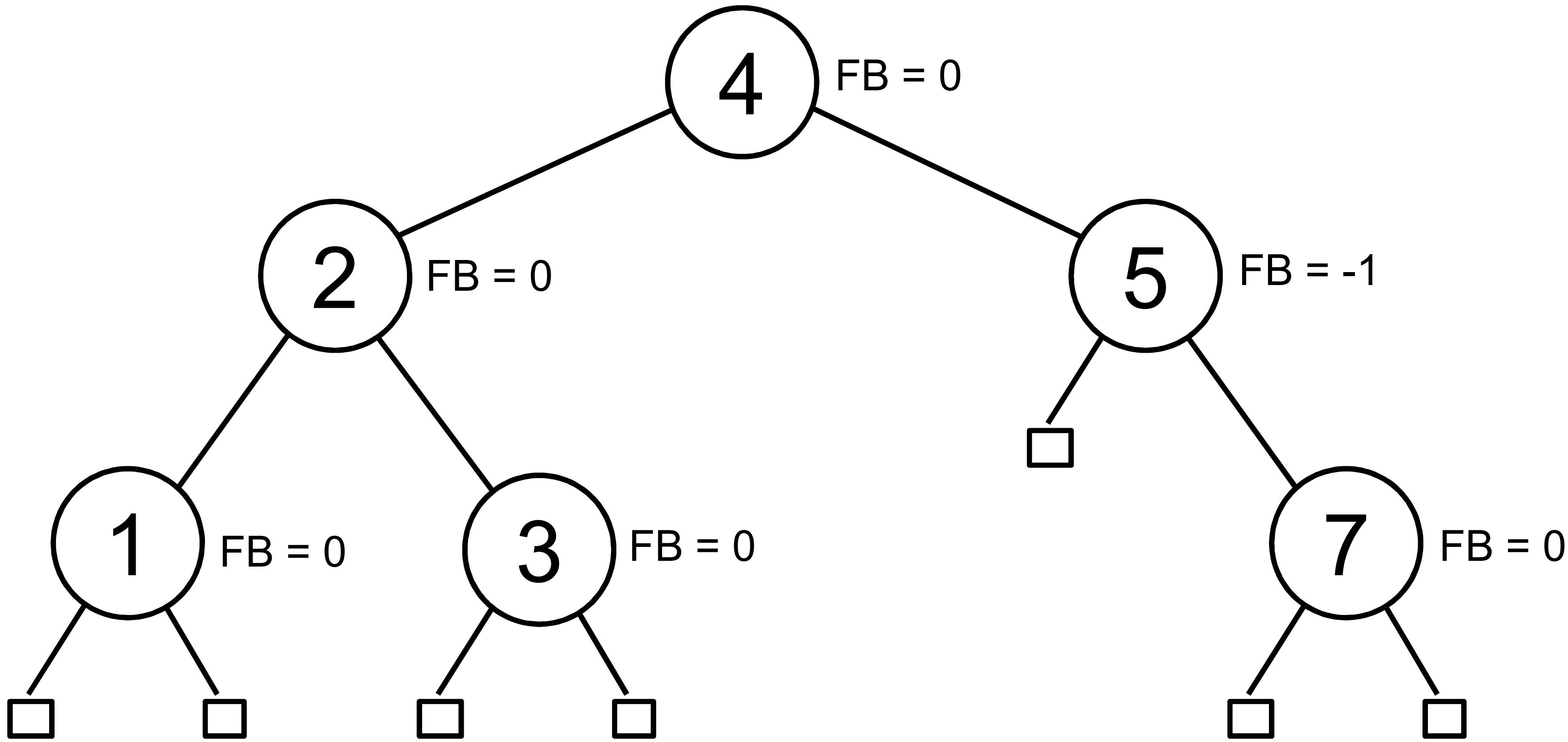
aplicar RDD

Construção de uma árvore AVL

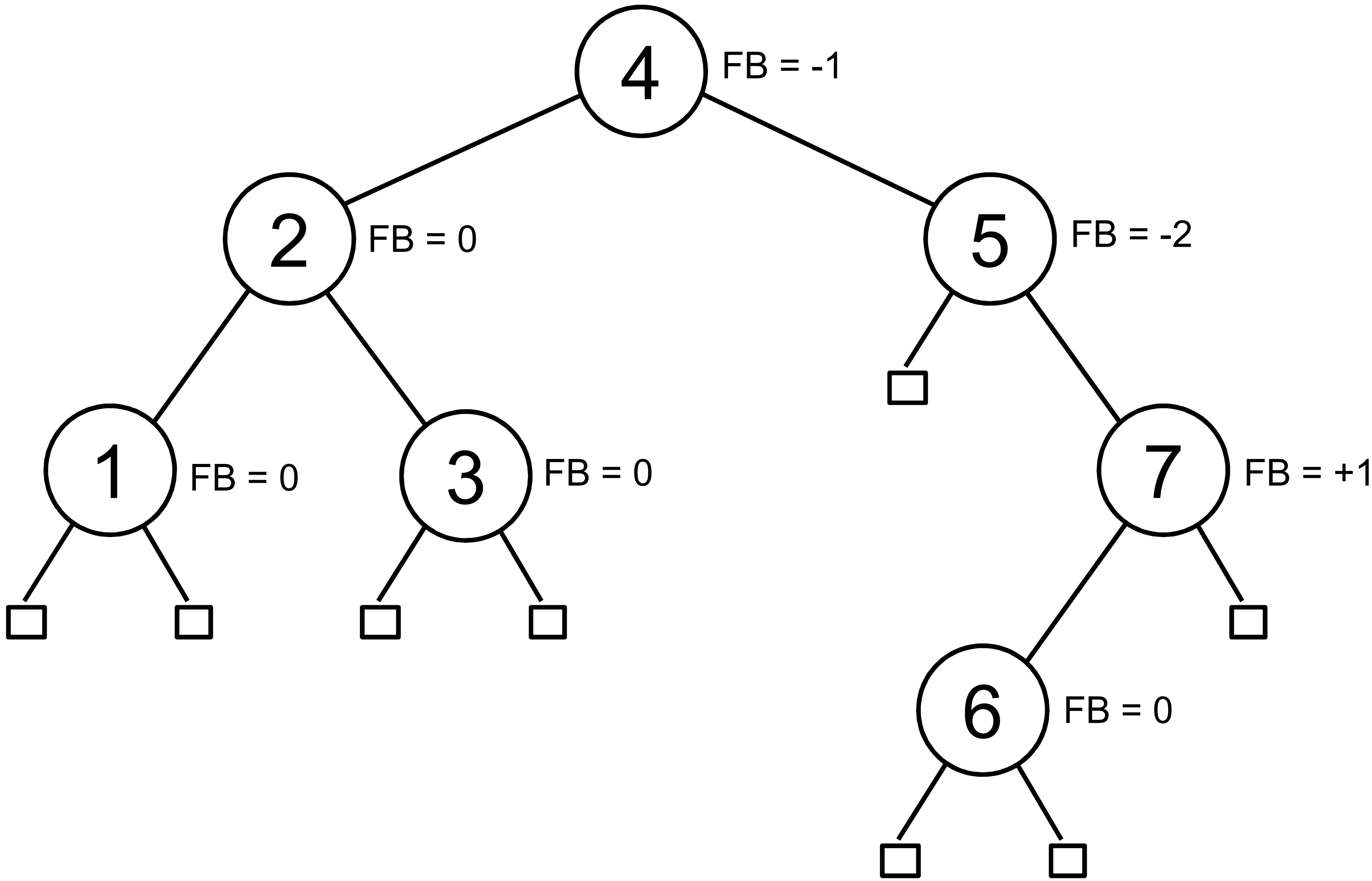


aplicar RDD

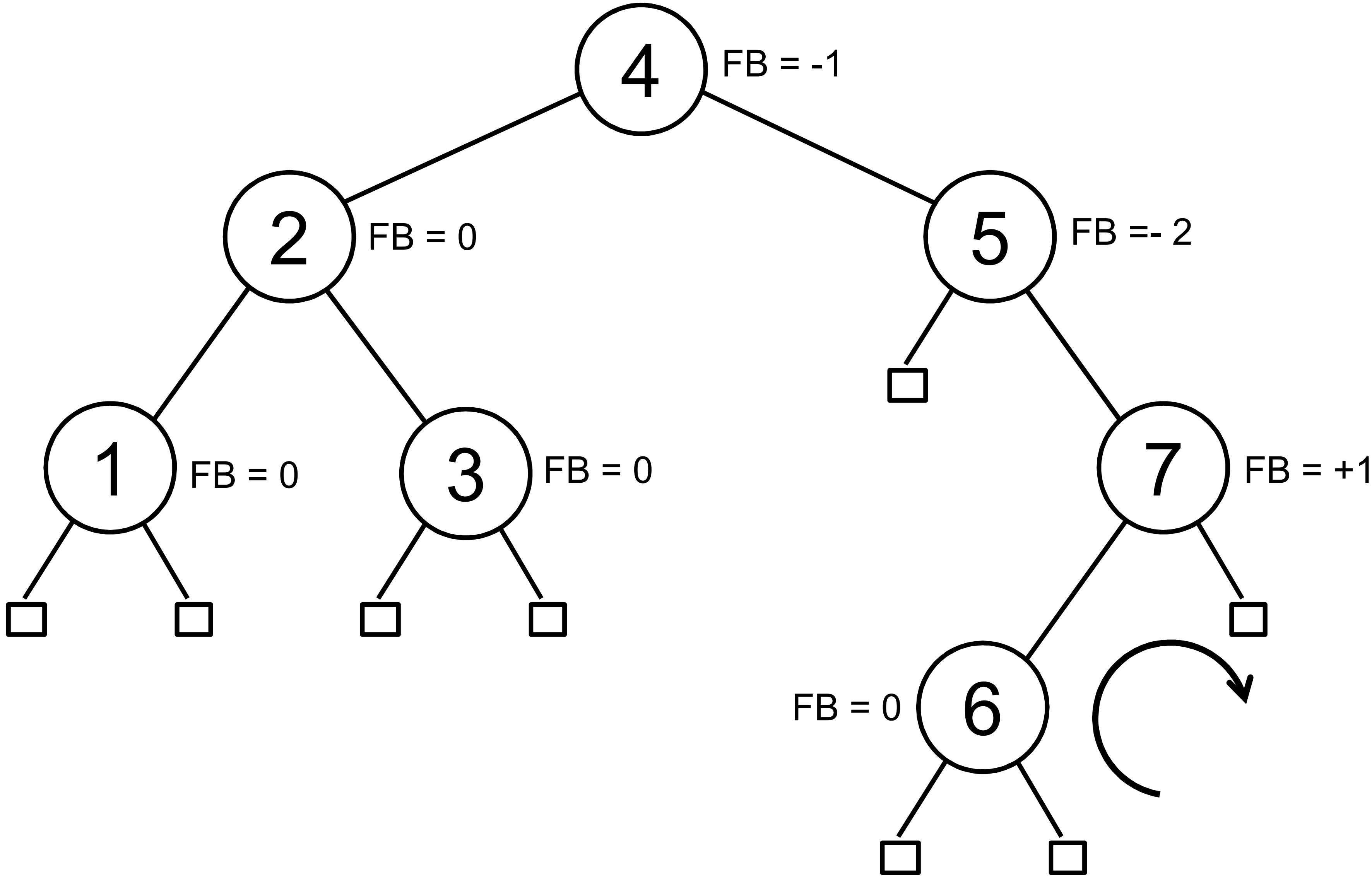
Construção de uma árvore AVL



Construção de uma árvore AVL

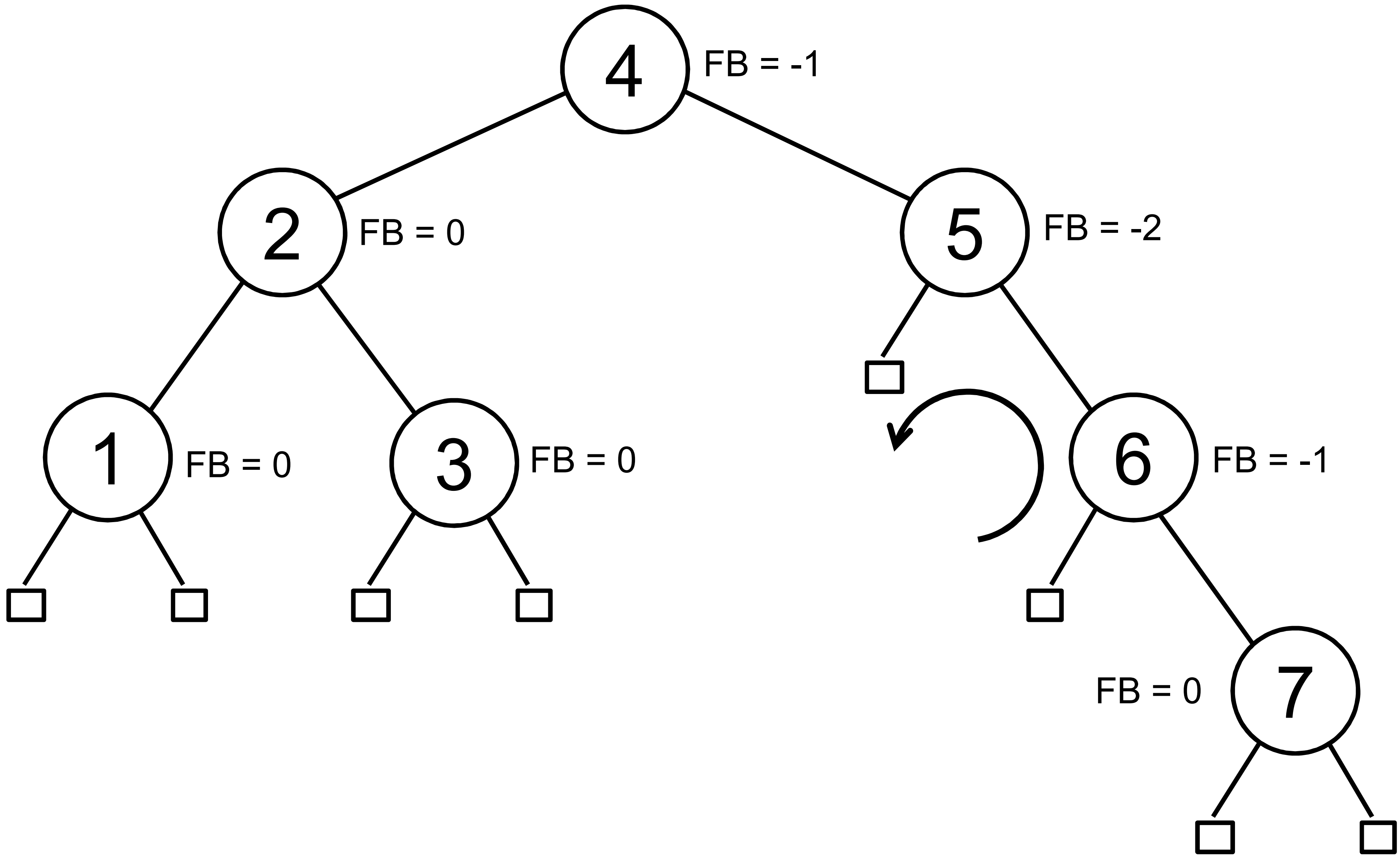


Construção de uma árvore AVL



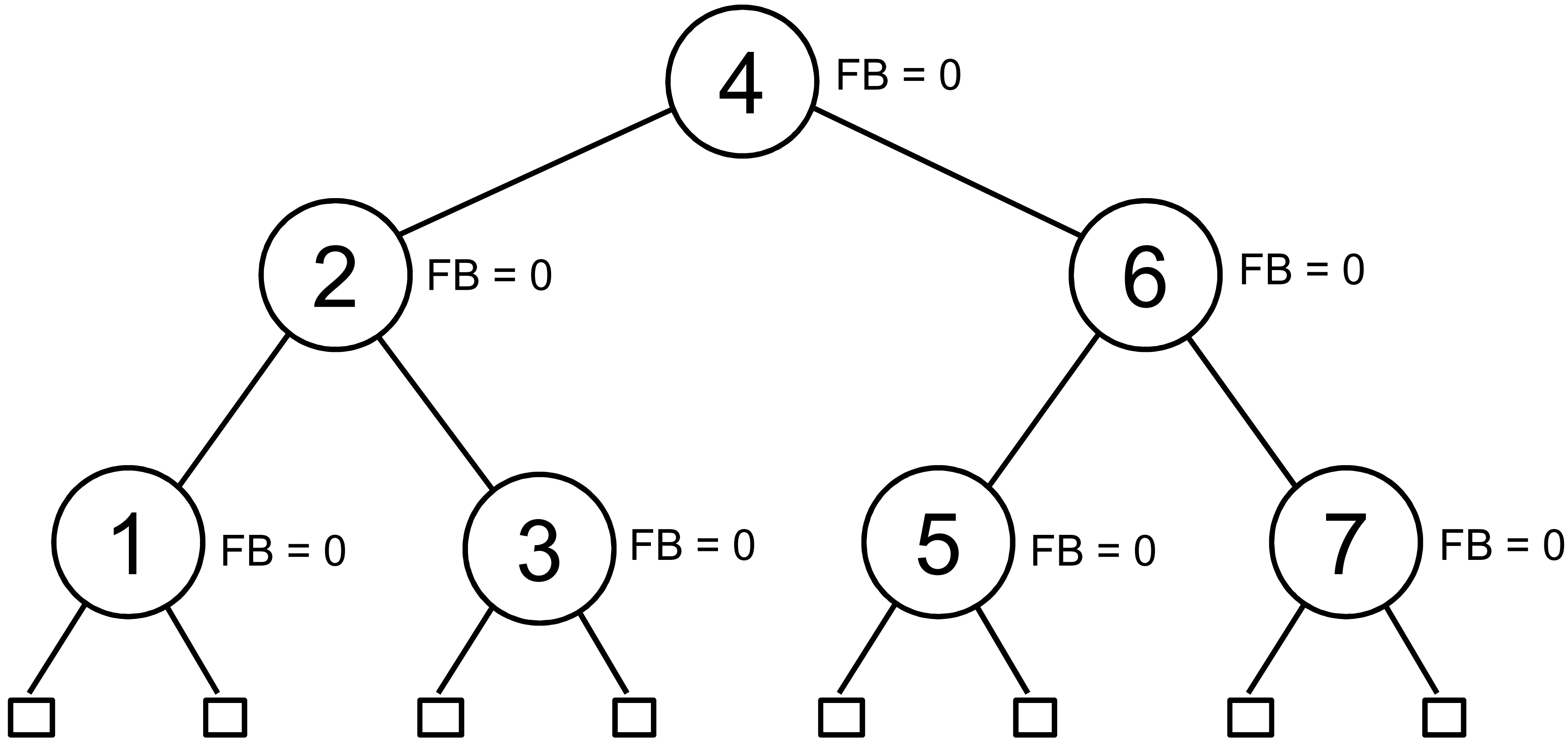
Aplicar RDE

Construção de uma árvore AVL



Aplicar RDE

Construção de uma árvore AVL



Exercícios

1. Desenhe à mão, a árvore AVL correspondente à sequência de valores {51, 57, 98, 19, 11, 45, 79 }, e escreva a sequência correta de operações de rotação (RSE, RSD, RDE e RDD) empregadas durante a construção da árvore.