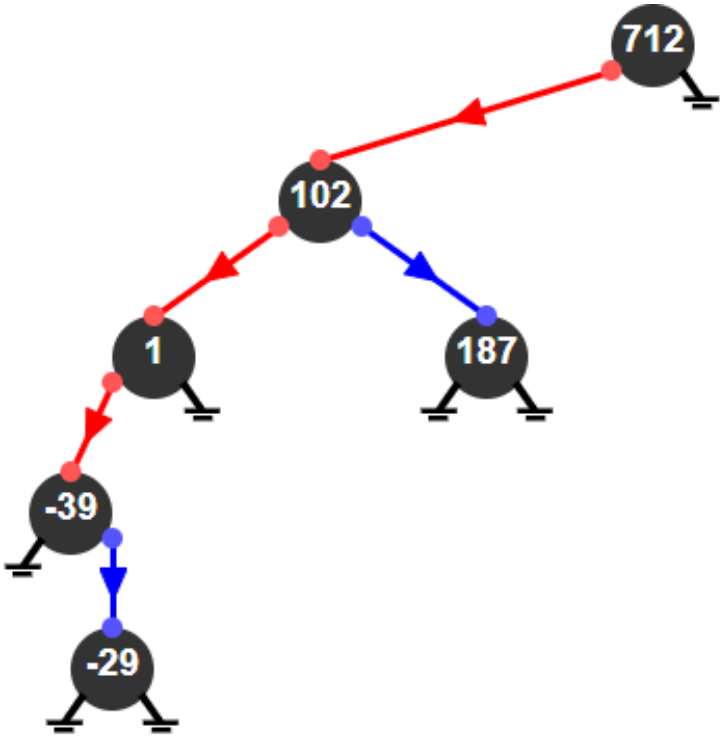
Árvore AVL


Left Nodes
Right nodes
Perfectly balanced
AVL Tree
...as all things should be

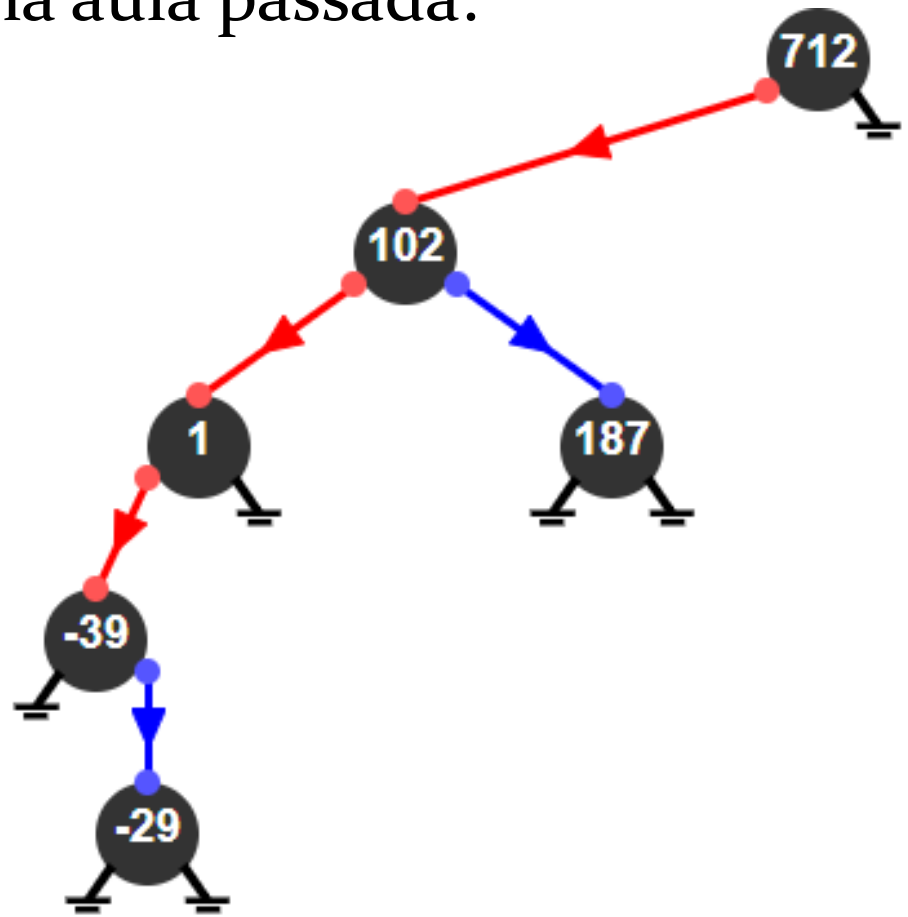
Árvore Binária de Busca (ABB) – recapitulando

- Cada nó da ABB possui um campo chave, que identifica de forma **única** o registro/dado salvo.
- ABBs estabelecem uma **relação de ordem** mediante à chave.
- Para todo nó x da ABB:
  - $x.chave > y.chave$, para todo nó y da subárvore esquerda;
  - $x.chave < y.chave$, para todo nó y da subárvore direita;



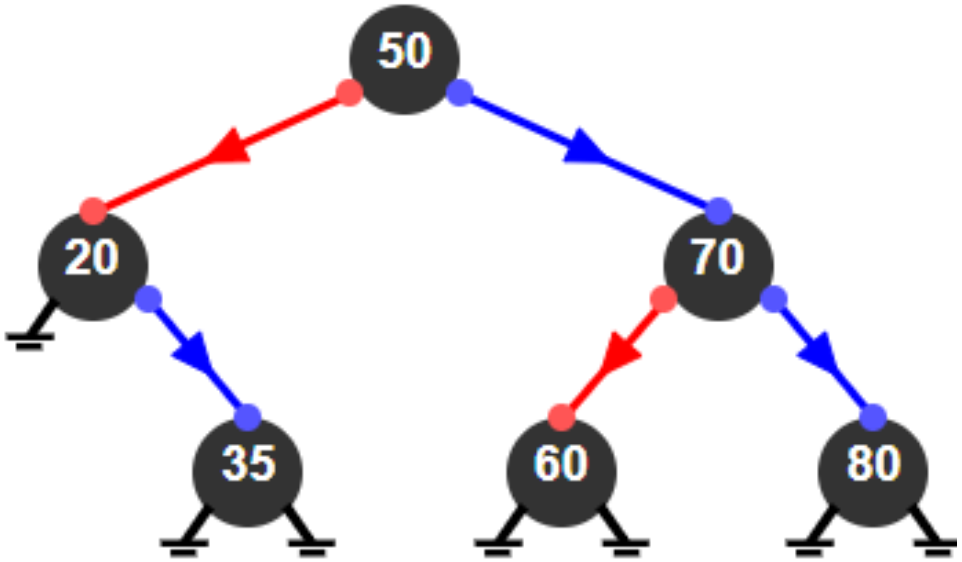
Árvore Binária de Busca (ABB) – recapitulando

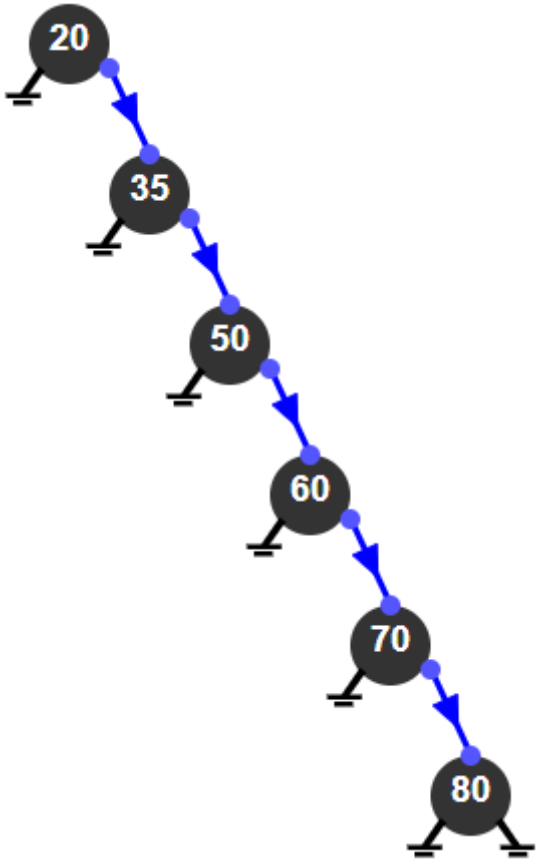
- Operações estudadas na aula passada:
  - Busca
  - Inclusão
  - Exclusão



Balanceamento de ABB - Motivação

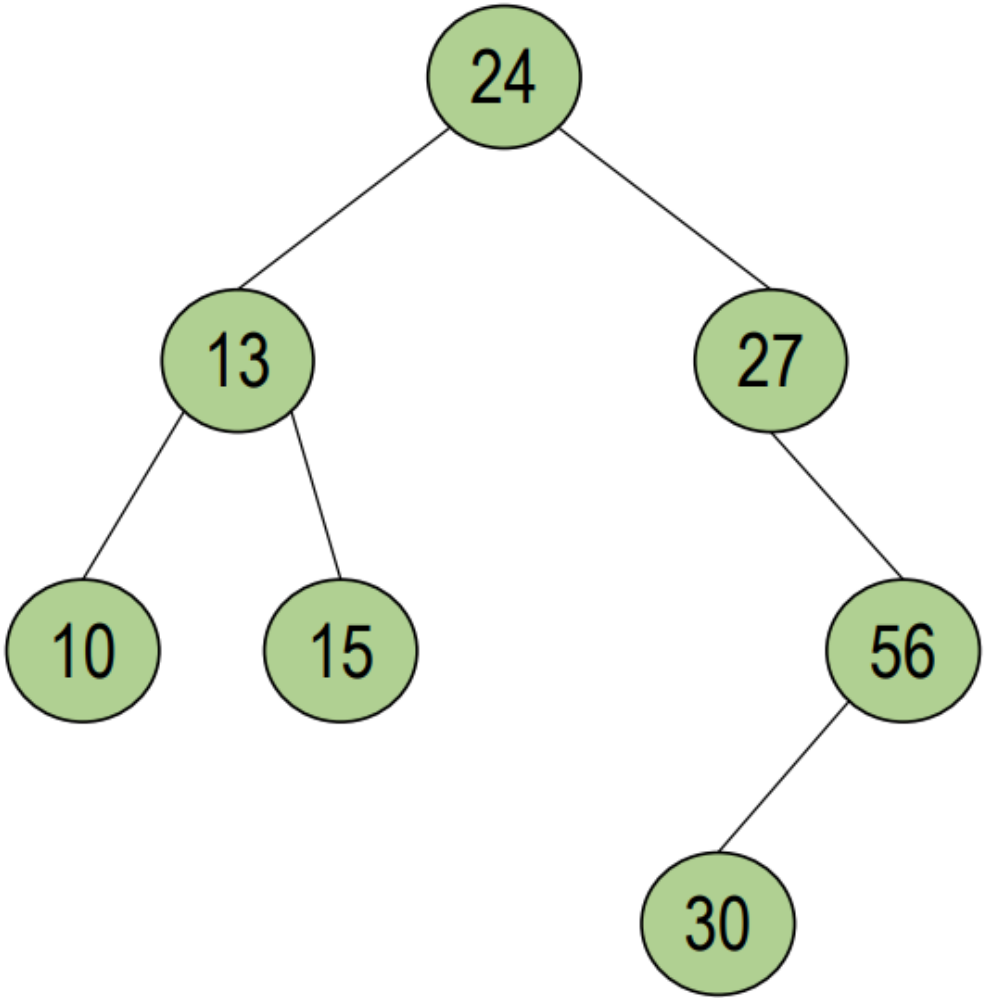
- A **ordem de inserção** (**ou remoção**) determina o **formato** de uma Árvore de Busca Binária.
- Essa ordem é a diferença entre a árvore se comportar como uma **busca binária** ou uma **busca sequencial**.



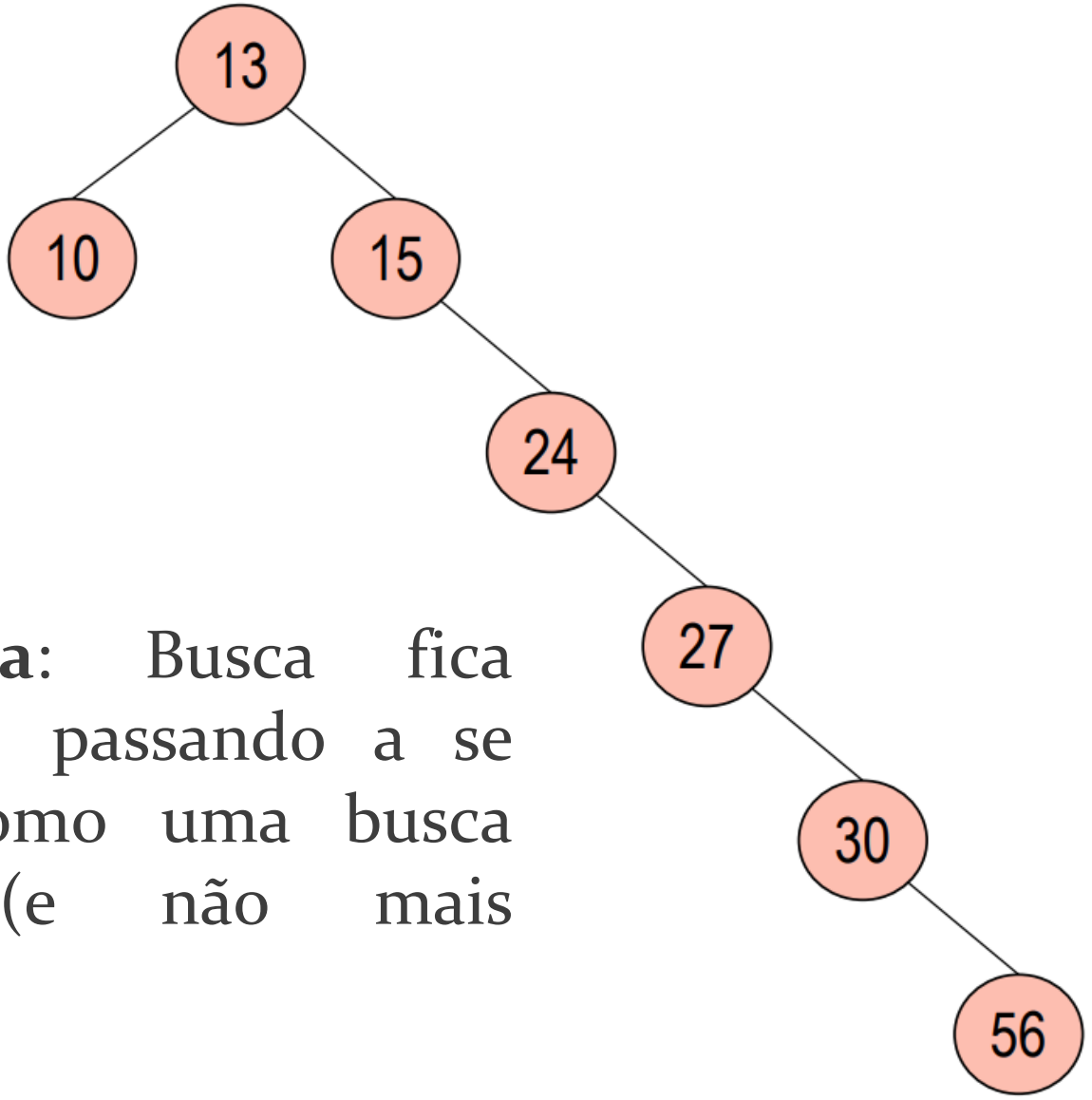


Problema de desbalanceamento progressivo

- Exemplo: inserção dos elementos S = {24, 27, 13, 10, 56, 15, 30}



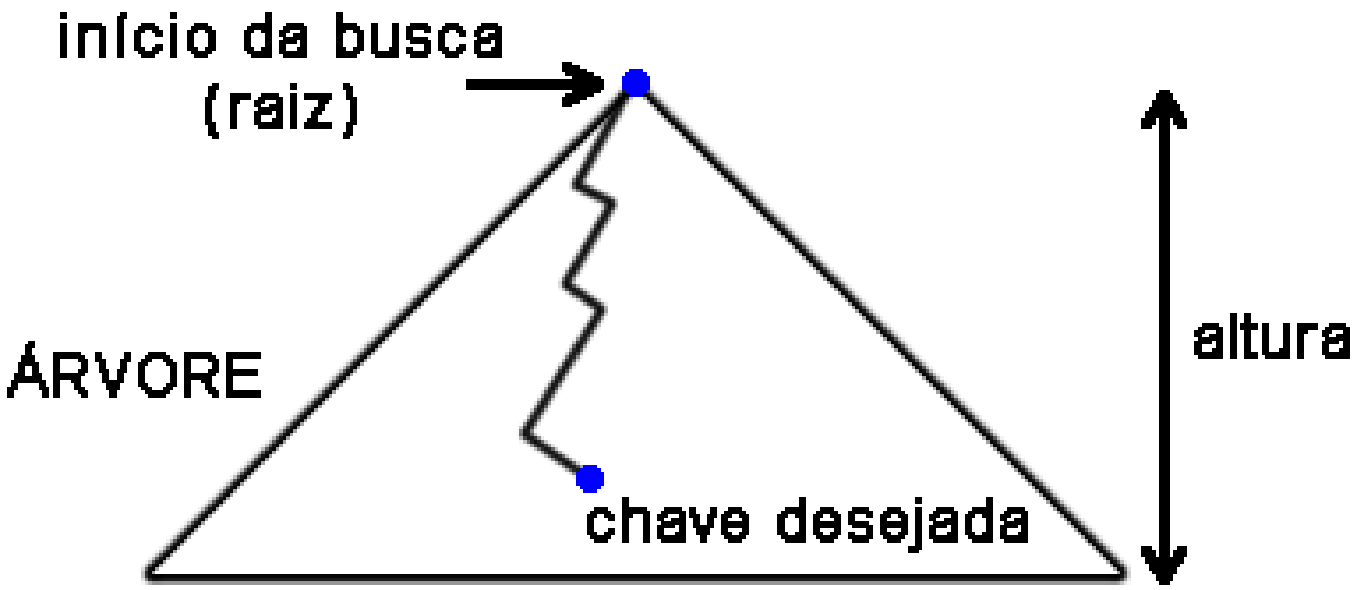
- Exemplo: inserção dos mesmos elementos S = {13, 10, 15, 24, 27, 30}



- **Consequência**: Busca fica mais custosa, passando a se comportar como uma busca sequencial (e não mais binária)!!!

Balanceamento - Objetivos

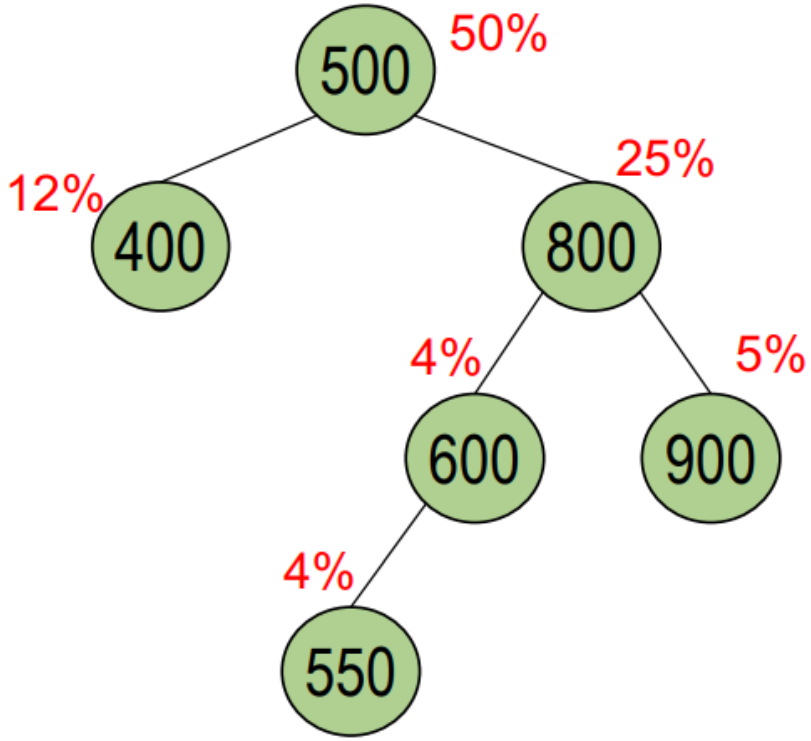
- Otimizar as operações básicas ➔ custo das operações é proporcional ao número de níveis da árvore.
- Diminuir o número médio de comparações.
- Ideia central do balanceamento: manter o **custo de acesso aos nós na mesma ordem de grandeza** de uma árvore ótima, $O(\log n)$.



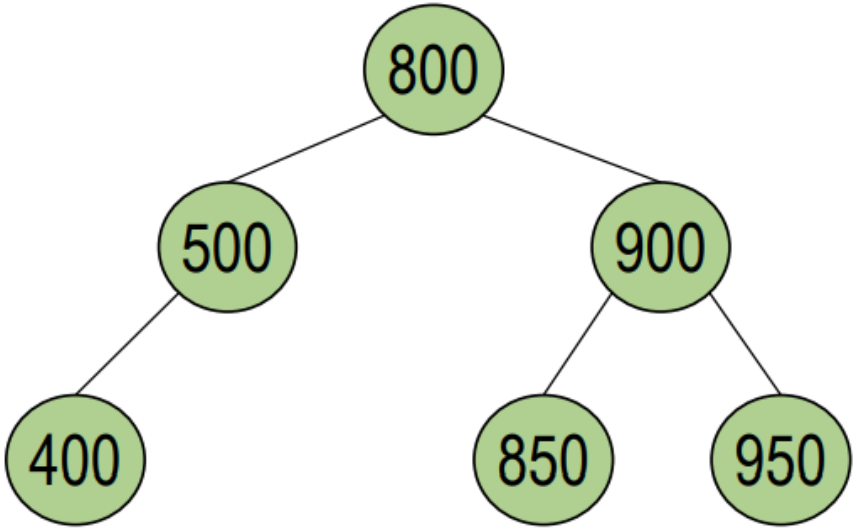
Balanceamento

Tipos de distribuição de balanceamento

- **Não Uniforme - Splay:** Nós mais acessados ficam perto da raiz.



- **Uniforme - Árvores AVL, Rubro-Negras:** Diferença das alturas das sub-árvores não excedem um valor pré-definido.



Balanceamento via Árvores AVL

- O balanceamento perfeito é computacionalmente caro!
- O que fazer então?
  - Nada? Depender da sorte, chorar?

Balanceamento via Árvores AVL

- O balanceamento perfeito é computacionalmente caro!
- O que fazer então?
  - Nada? Depender da sorte, chorar?

Balanceamento via Árvores AVL

- O balanceamento perfeito é computacionalmente caro!
- O que fazer então?
  - Nada? Depender da sorte, chorar?
  - **Ou realizar um "bom" balanceamento**?

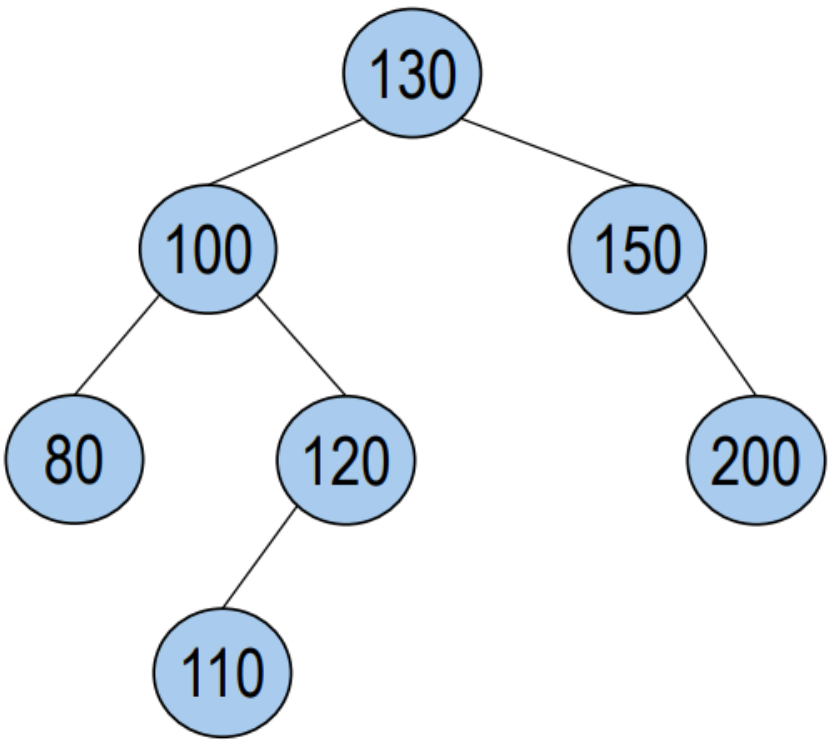
Balanceamento via Árvores AVL

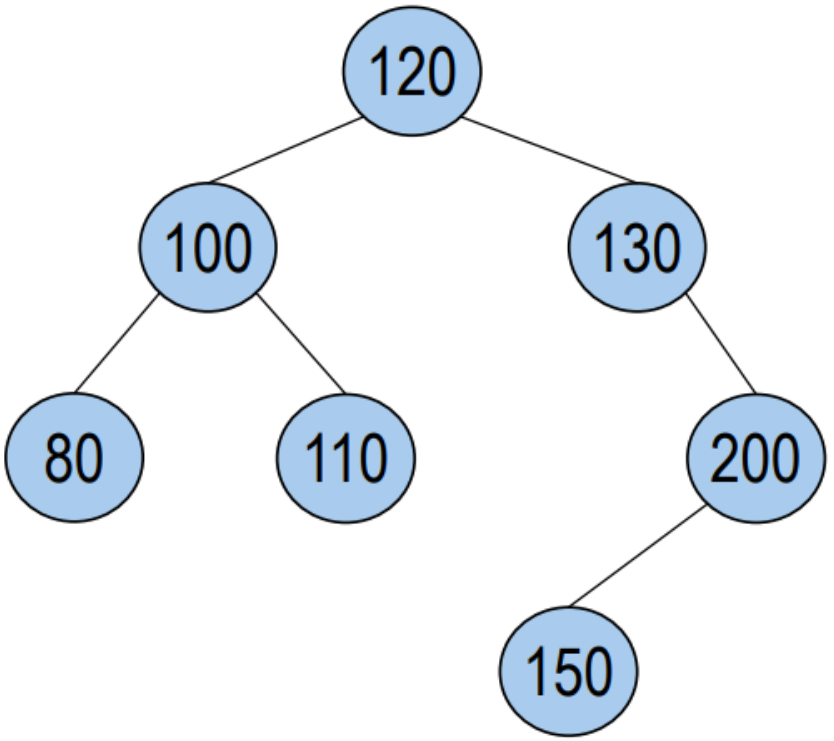
- Árvores **AVL**
  - **A**DELSON-**V**ELSKII e **L**ANDIS (1962)

- Uma **árvore binária de busca** é uma **AVL** quando, para qualquer um de seus nós, a diferença entre as alturas de suas sub-árvore direita e esquerda é, no máximo, 1.
- Conclusão: uma AVL é uma ABB balanceada!

Para praticar ...

- Verifique quais ABBs são AVL

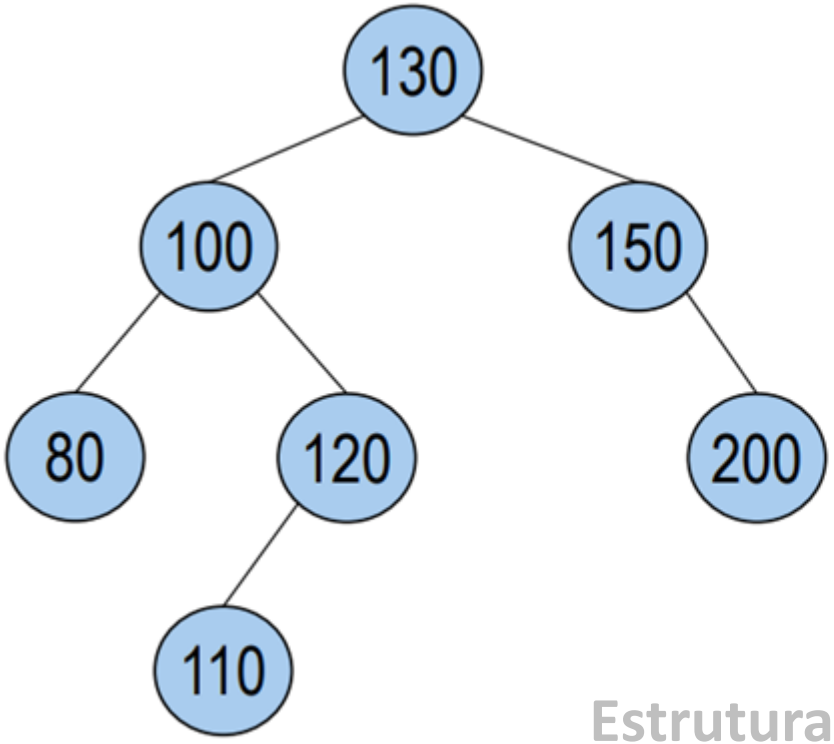




Para praticar ...

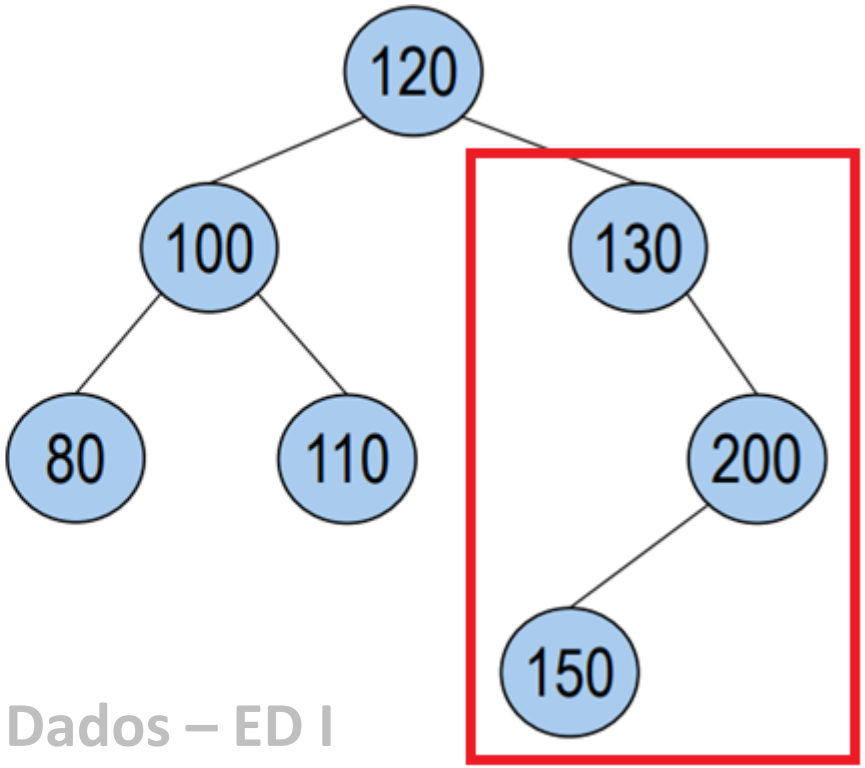
- Verifique quais das ABB são AVL

AVL

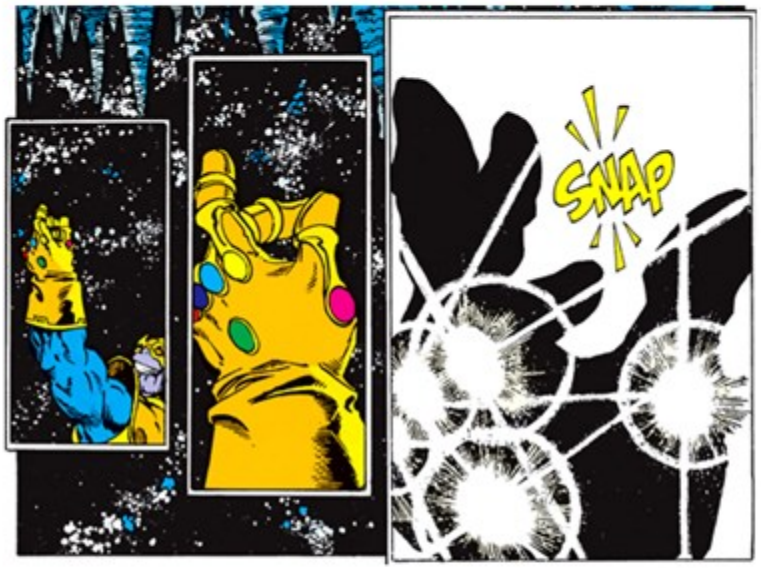


ABB

Diferença das alturas das sub-árvores do nó 130 é 2



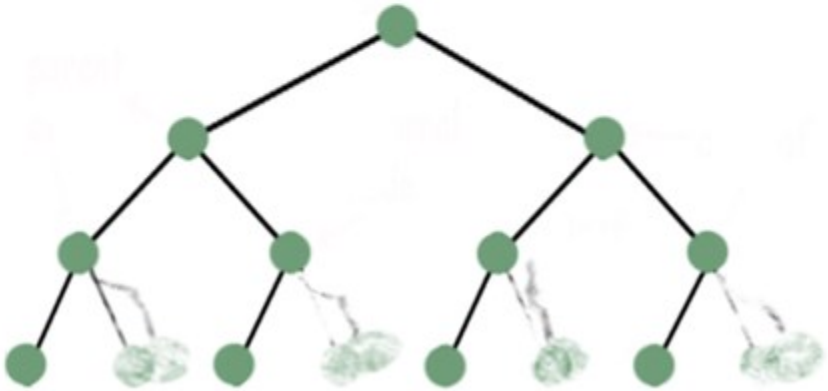
If Thanpos snapped his fingers at a binary tree, would it end up

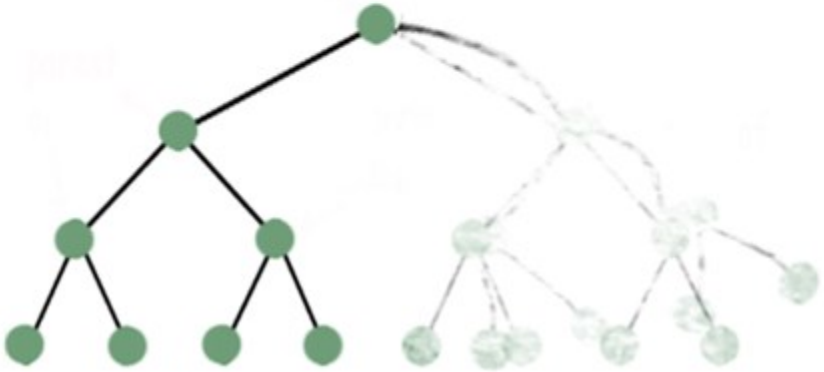


like this

or

like this?



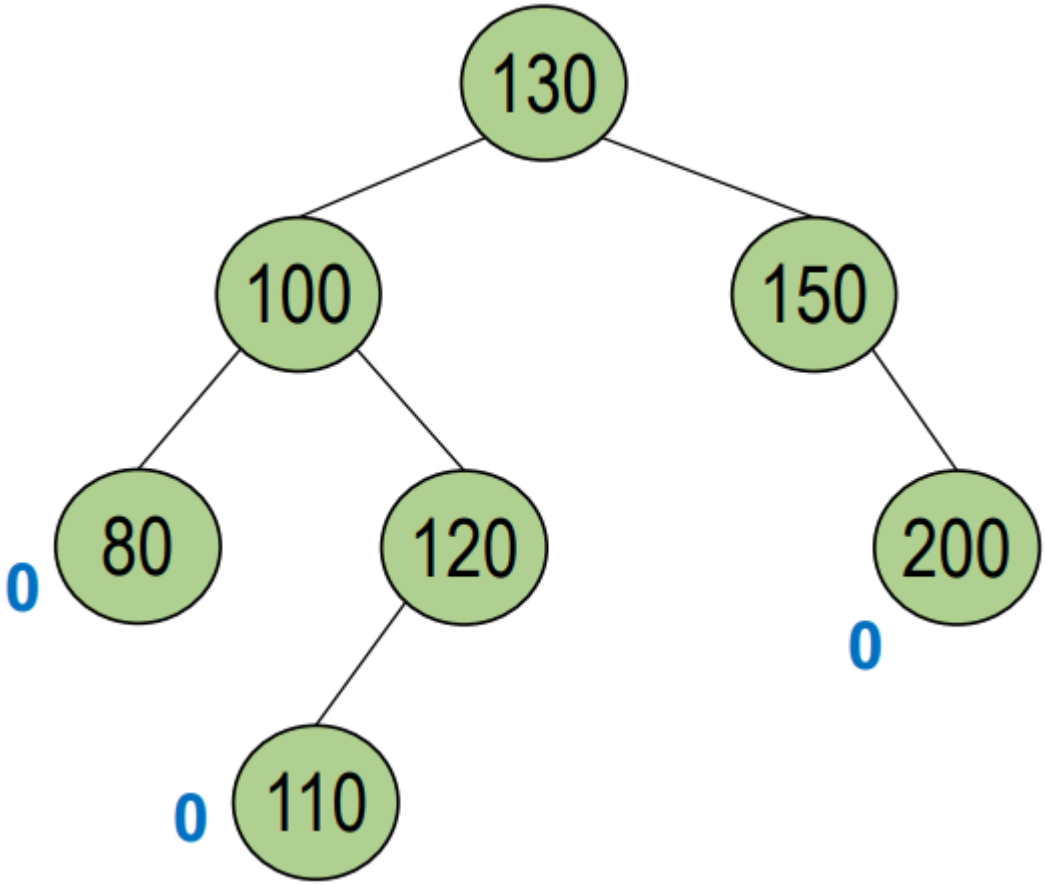


Árvores AVL

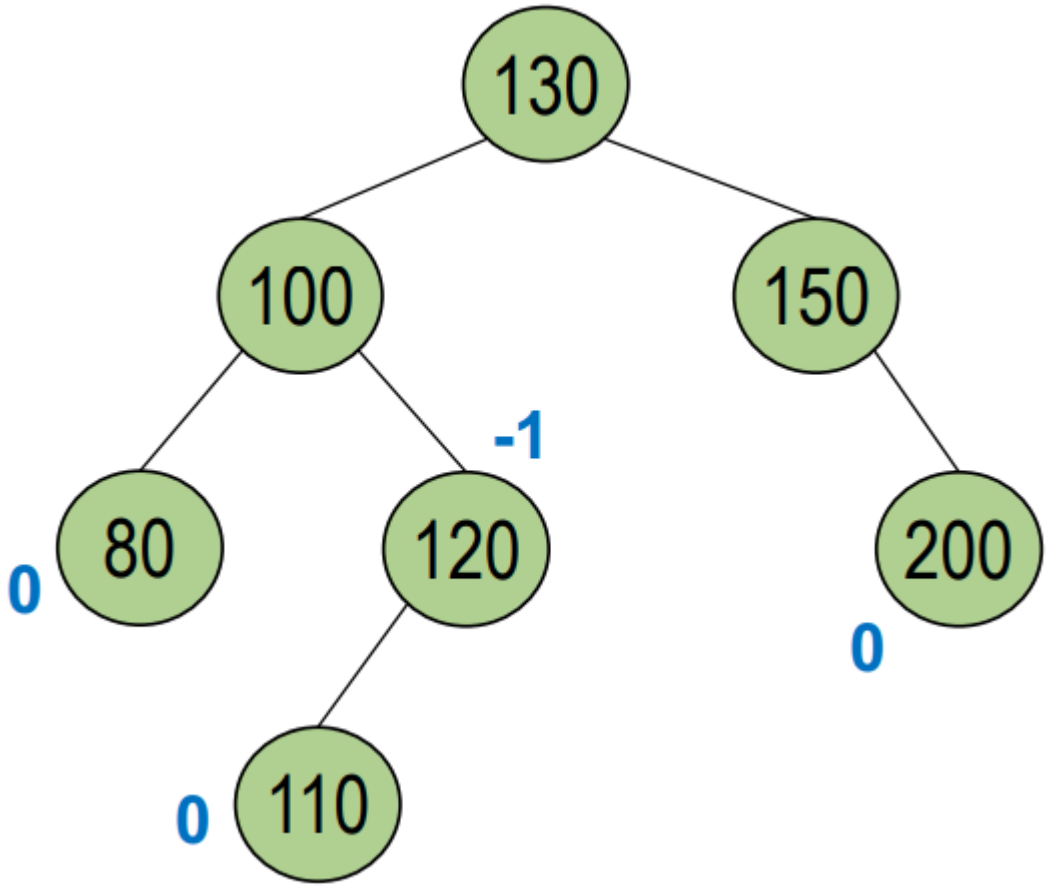
- Definição (**Fator de Balanceamento**): diferença entre a altura da sub-árvore direita e esquerda de um nó n
  - FB(n) = altura(n->dir) – altura(n->esq)

- Nota: o fator de balanceamento é calculado **para cada nó da árvore.**

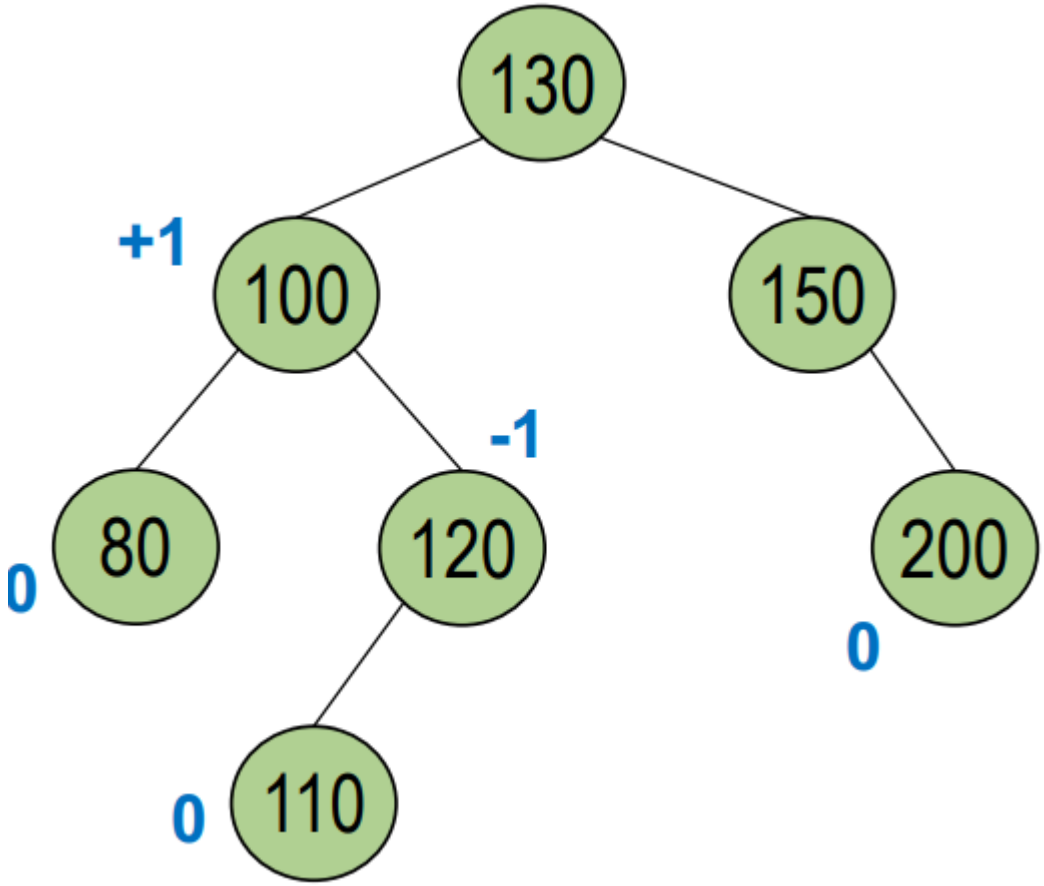
- Definição (**Fator de Balanceamento**): diferença entre altura da sub-árvore direita e esquerda de um nó n
  - FB(n) = altura(n->dir) – altura(n->esq)



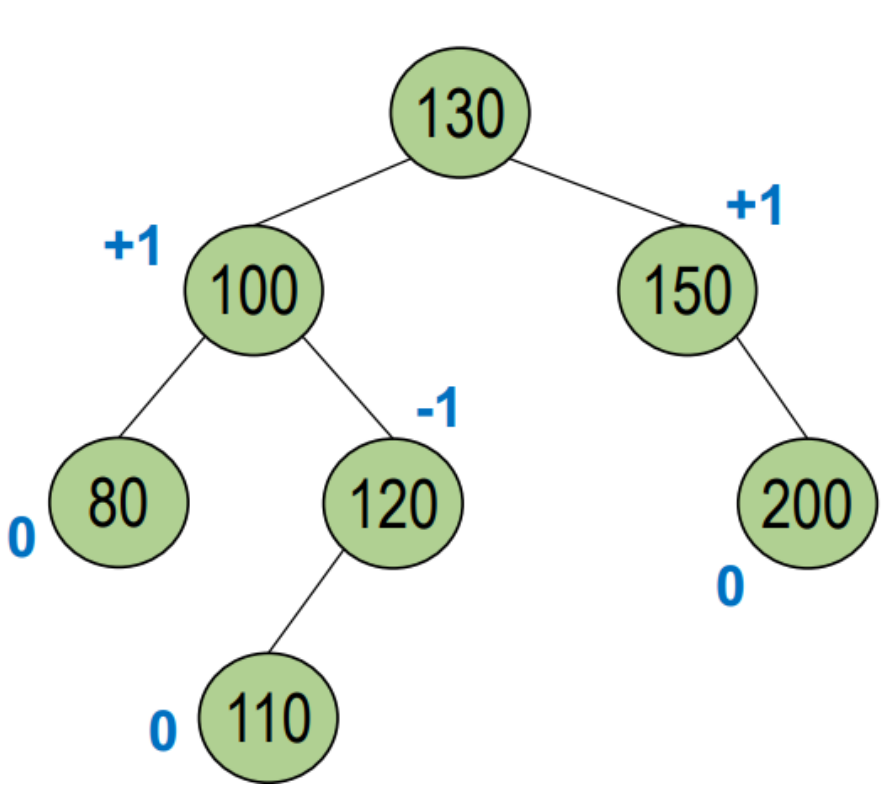
- Definição (**Fator de Balanceamento**): diferença entre altura da sub-árvore direita e esquerda de um nó n
  - FB(n) = altura(n->dir) – altura(n->esq)

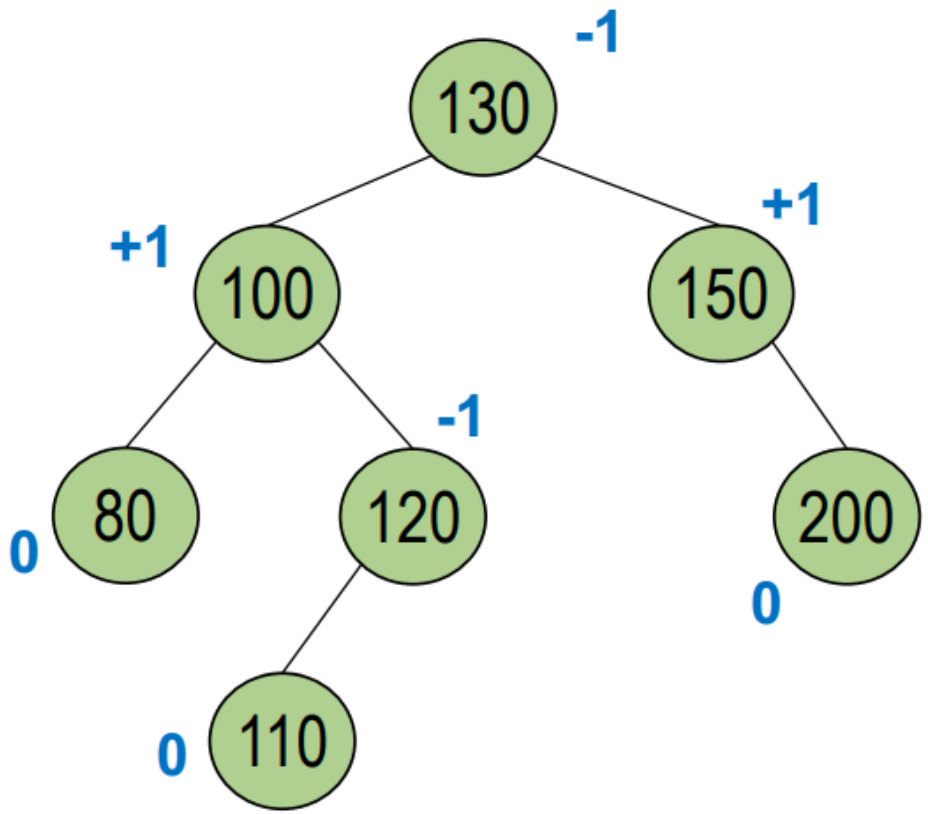


- Definição (**Fator de Balanceamento**): diferença entre altura da sub-árvore direita e esquerda de um nó n
  - FB(n) = altura(n->dir) – altura(n->esq)

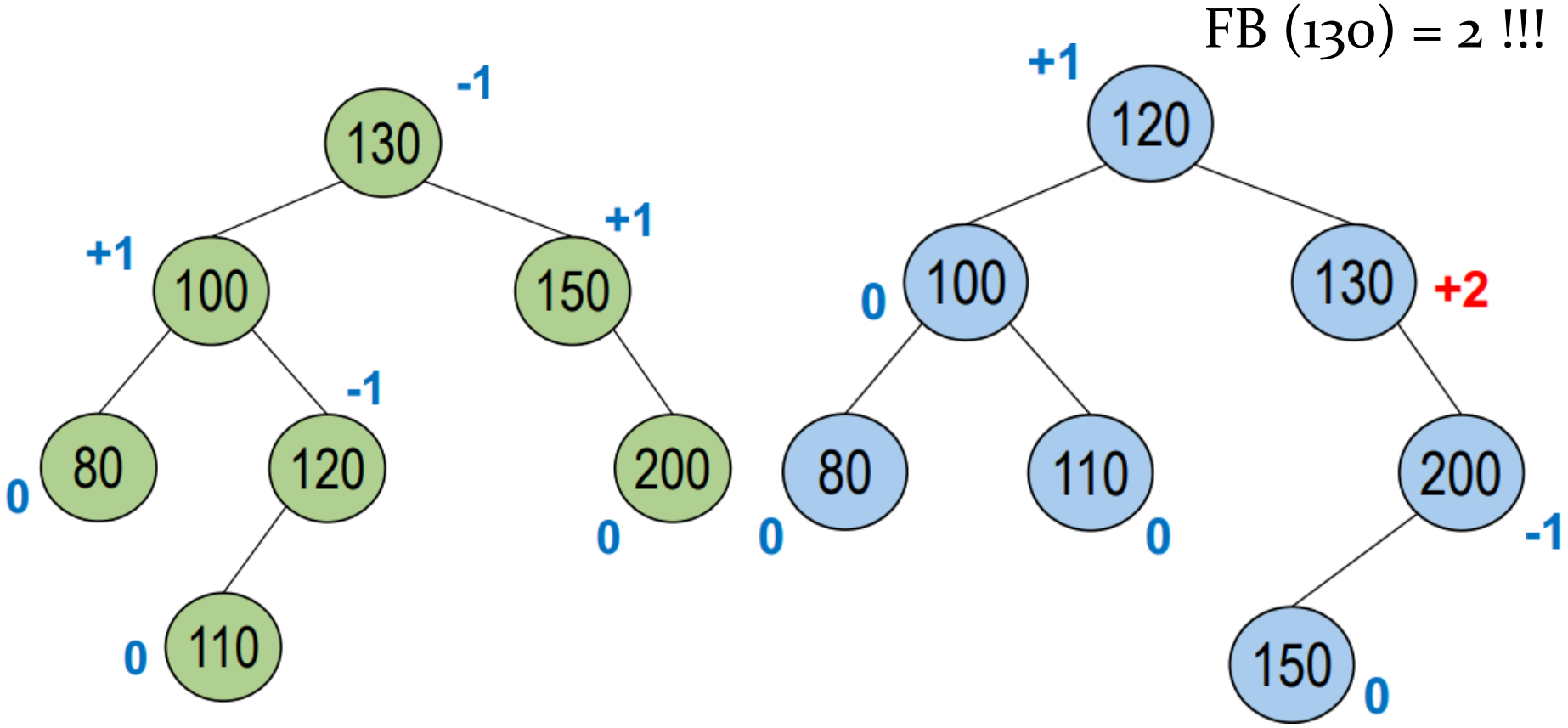


- Definição (**Fator de Balanceamento**): diferença entre altura da sub-árvore direita e esquerda de um nó n
  - FB(n) = altura(n->dir) – altura(n->esq)



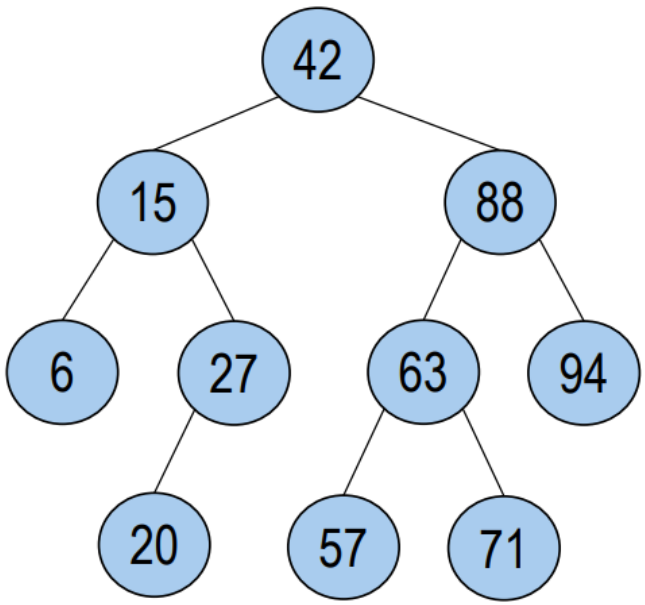


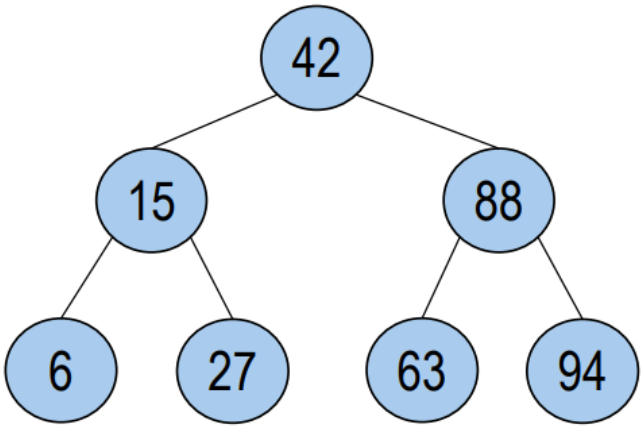
- **Propriedade:** Para uma ABB ser AVL (se manter balanceada), o FB precisa ser **-1, ou zero, ou +1** para **todos os nós** da árvore !!!

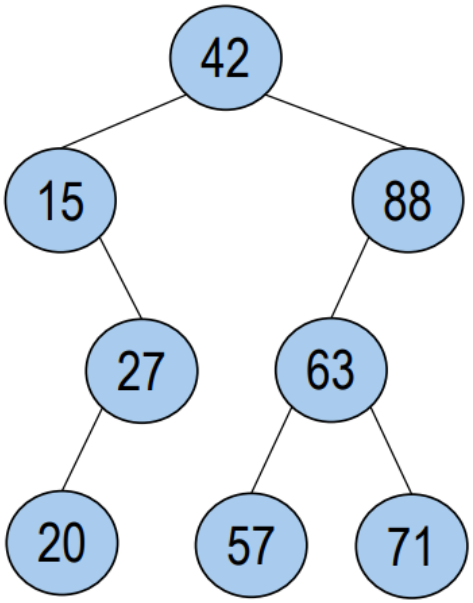


Para praticar ...

- Verifique quais das ABB são AVL, e calcule o FB para cada nó das árvores.

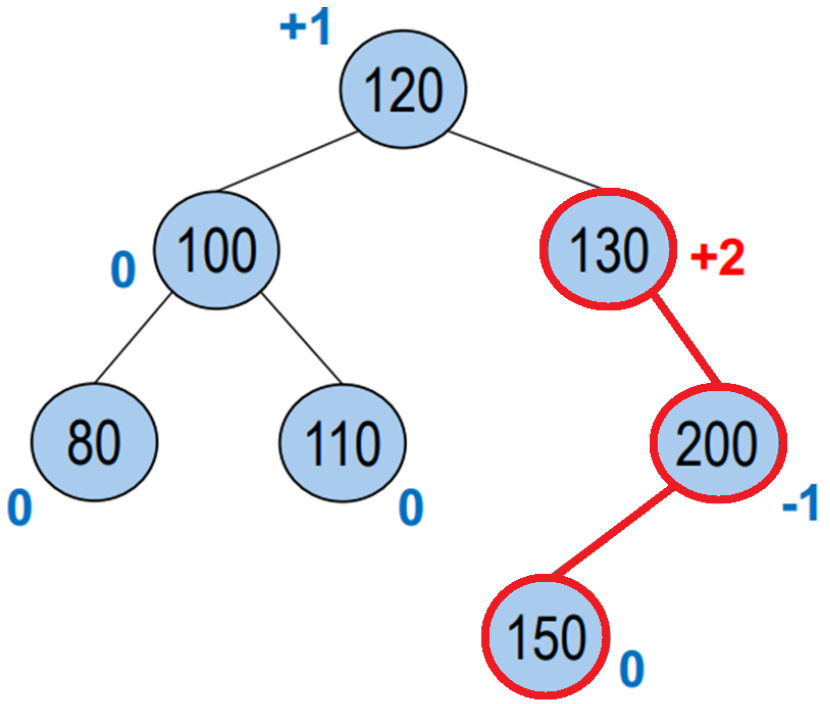






Árvores AVL – Observações importantes

- Fator de Balanceamento (FB): calculado **para cada nó da árvore!**
  - O que assegura o **balanceamento de uma AVL** é a verificação do FB **para cada nó da árvore**, já que cada nó **é a raiz de uma sub-árvore**.
  - **Conclusão**: desequilíbrio é uma **propriedade do nó**! ➔ sub-árvore **vermelha** (com nó raiz 130) está desequilibrada, pois FB(130) = 2

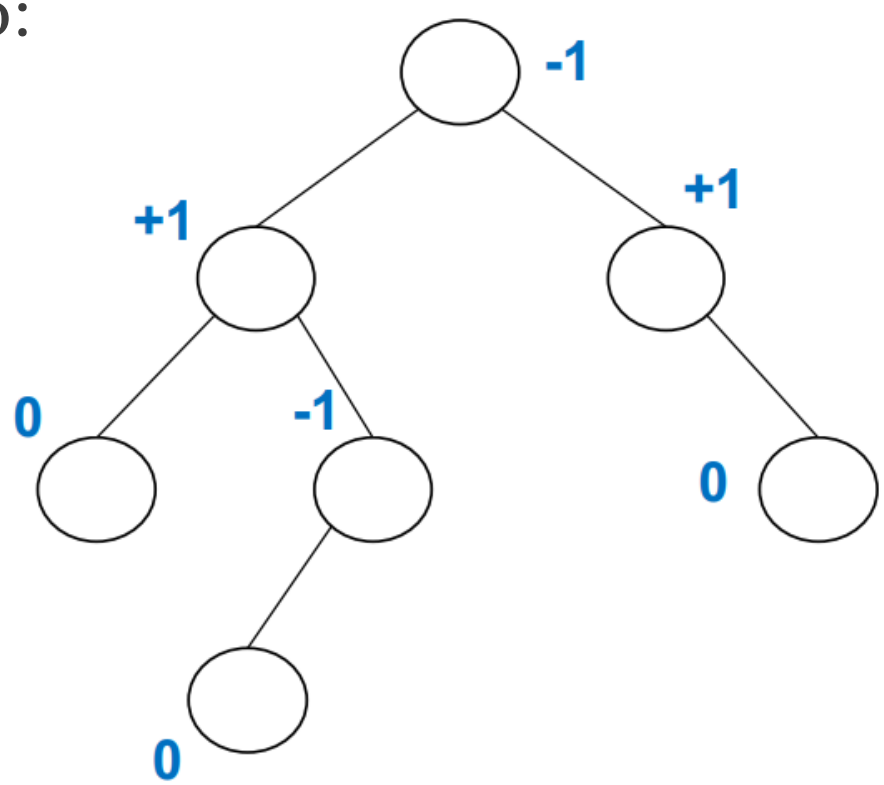


Árvores AVL

Operações

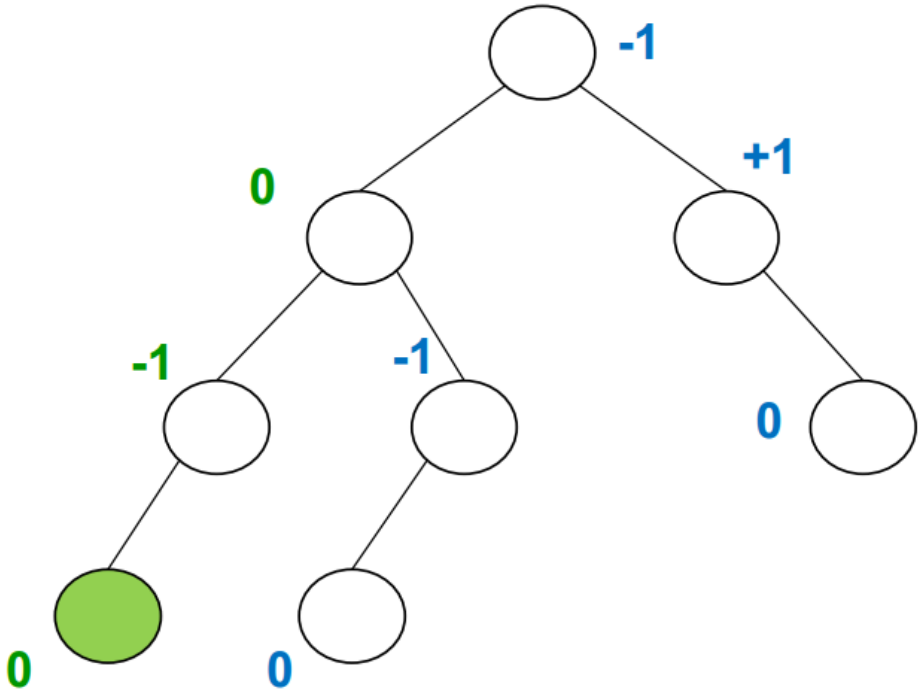
Árvores AVL

- Operações de **Inserção** e **Exclusão:** devem sempre preservar as propriedades da ABB = AVL!!!
- Vejamos alguns exemplos de **inserção** a partir da AVL inicial abaixo:

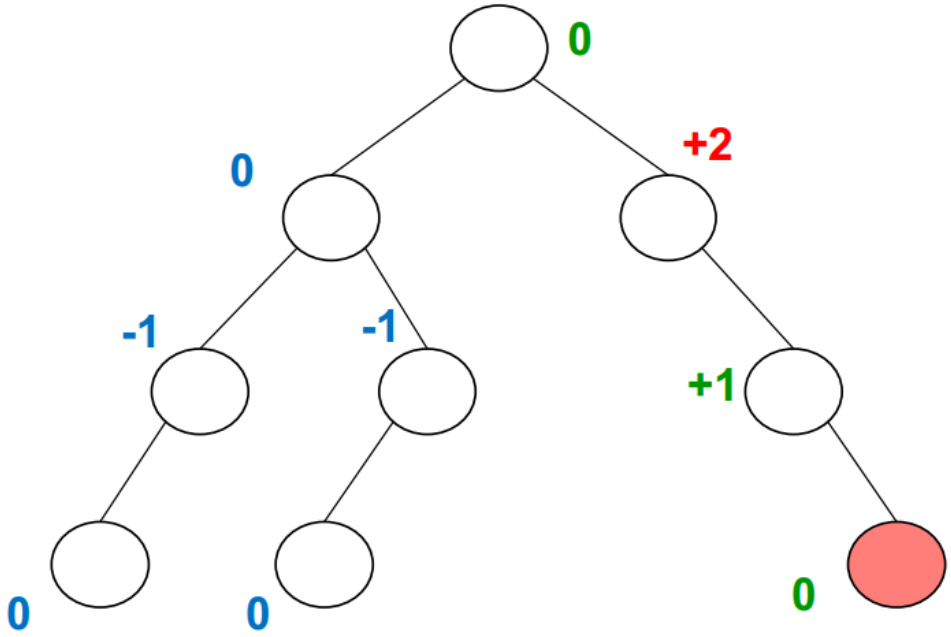


Árvores AVL – Inserção

- **Exemplos (inserções):**



Inserção ok!



Inserção desbalanceou a árvore

Como corrigir o desequilíbrio gerado?

- **ROTAÇÕES:**
  - Caso a inserção/remoção ocasione no desbalanceamento da árvore, é necessário executar operações que são conhecidas como **Rotação.**
- **OPERAÇÕES DE ROTAÇÃO:**
  - Tem como finalidade preservar: **(i) a ordenação das chaves** (garante que a árvore será sempre uma ABB); **(ii) o balanceamento da árvore** (assegura que ela seja AVL).
  - Rotações são aplicadas no ancestral mais próximo do nó inserido, cujo FB passou a ser +2 ou -2.

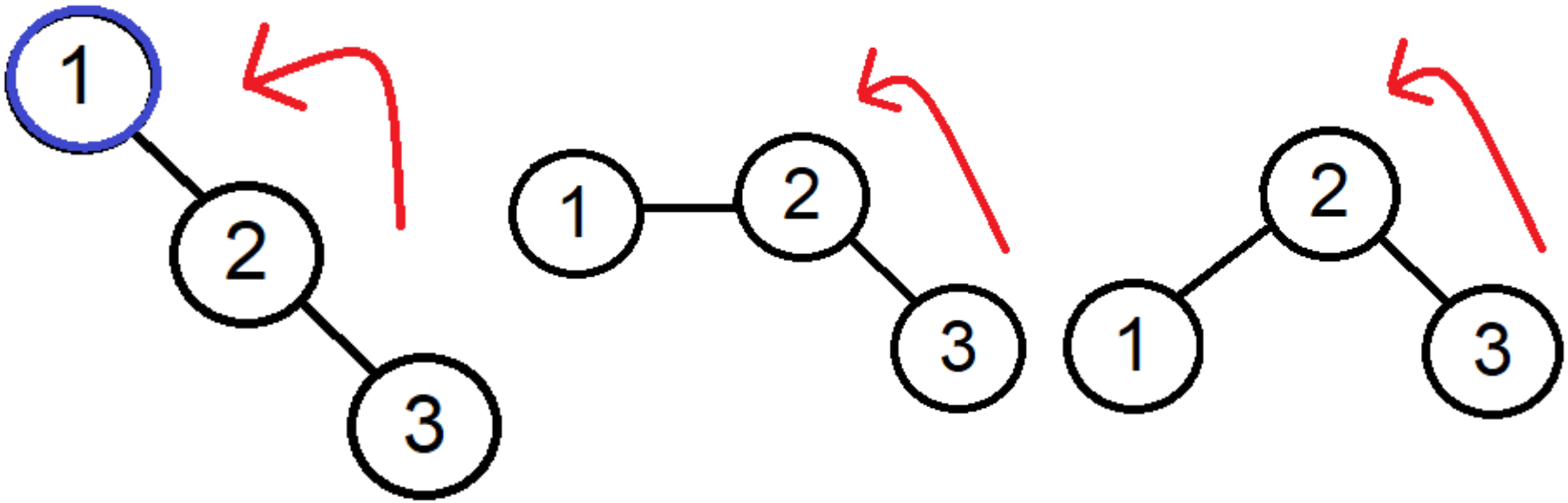
Balanceamento via Rotação - AVL

- **Quatro Tipos de Rotação Existentes:**
  1. Rotação simples:
     A. Esquerda
     B. Direita
  2. Rotação dupla:
     A. Direita (esquerda-direita)
     B. Esquerda (direita-esquerda)

- Nota: as rotações diferem entre si pelo **sentido da inclinação** entre o **nó pai e filho**.

1.A) Rotação simples: esquerda

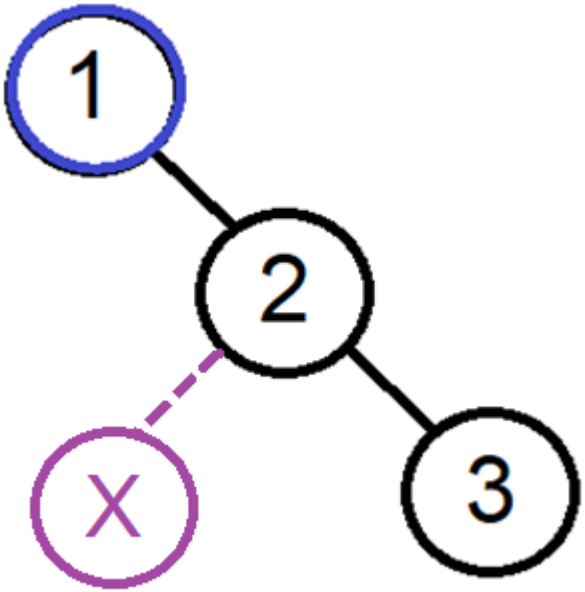
- Rotação à esquerda: consiste em **mover os nós que estão na sub-árvore da direita para a esquerda**.
  - Faz com que o filho da direita se torne a nova raiz.
  - Raiz original se torna o filho da esquerda da nova raiz.



- Quando usar? O nó desbalanceado (pai), seu filho e neto estão todos no mesmo sentido de inclinação, na diagonal principal.

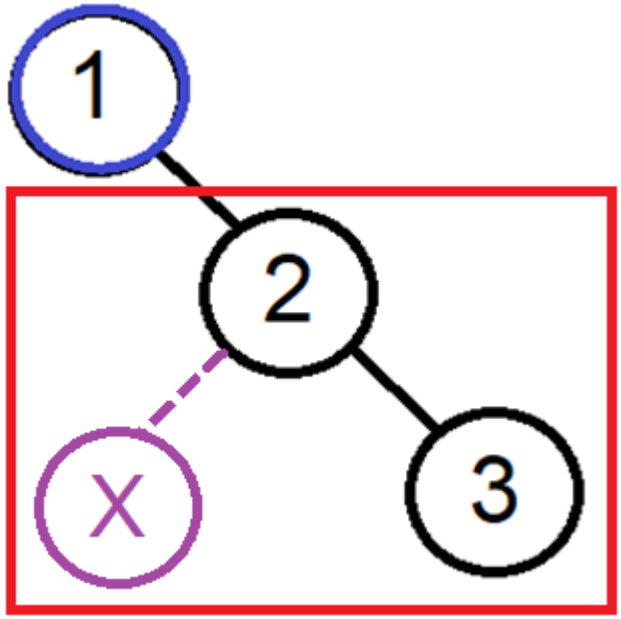
1.A) Rotação simples: esquerda

- **Caso particular**: E se o filho da direita (nó=2) já possuir um filho à esquerda (nó=X)?



1.A) Rotação simples: esquerda

- **Caso particular**: E se o filho da direita (nó=2) já possuir um filho à esquerda (nó=X)?
  - **Observação:** Note que todos os elementos da sub-árvore da direita (delimitado abaixo em vermelho) são **maiores** que o nó raiz (por definição de ABB)!
  - Portanto: X > nó=1.



- Rotação à esquerda: consiste em **mover os nós que estão na sub-árvore da direita para a esquerda**.
  - Filho da direita se torna a nova raiz.
  - Filho da esquerda (nó=X; gráfico 1) do filho da direita (nó=2; gráfico 1) se torna o filho da direita (nó=X; gráfico 2) do novo filho da esquerda (nó=1; gráfico 2).

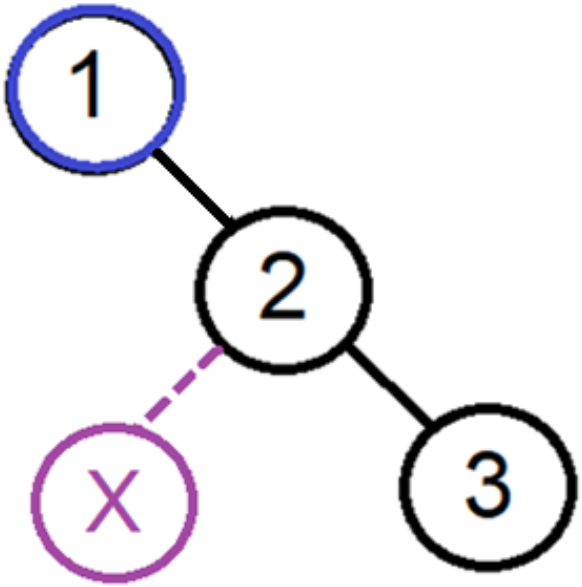


Gráfico 1

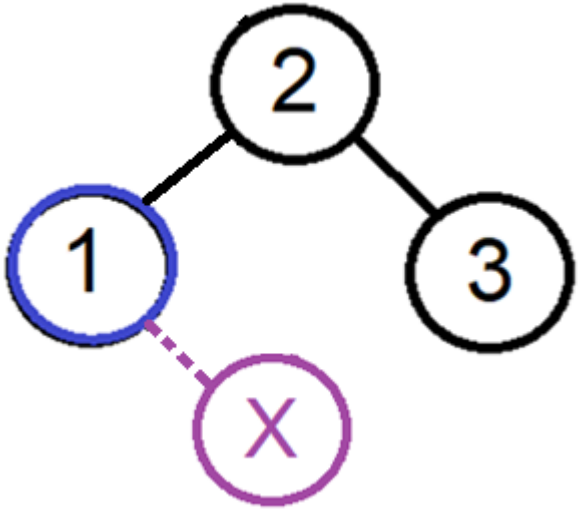
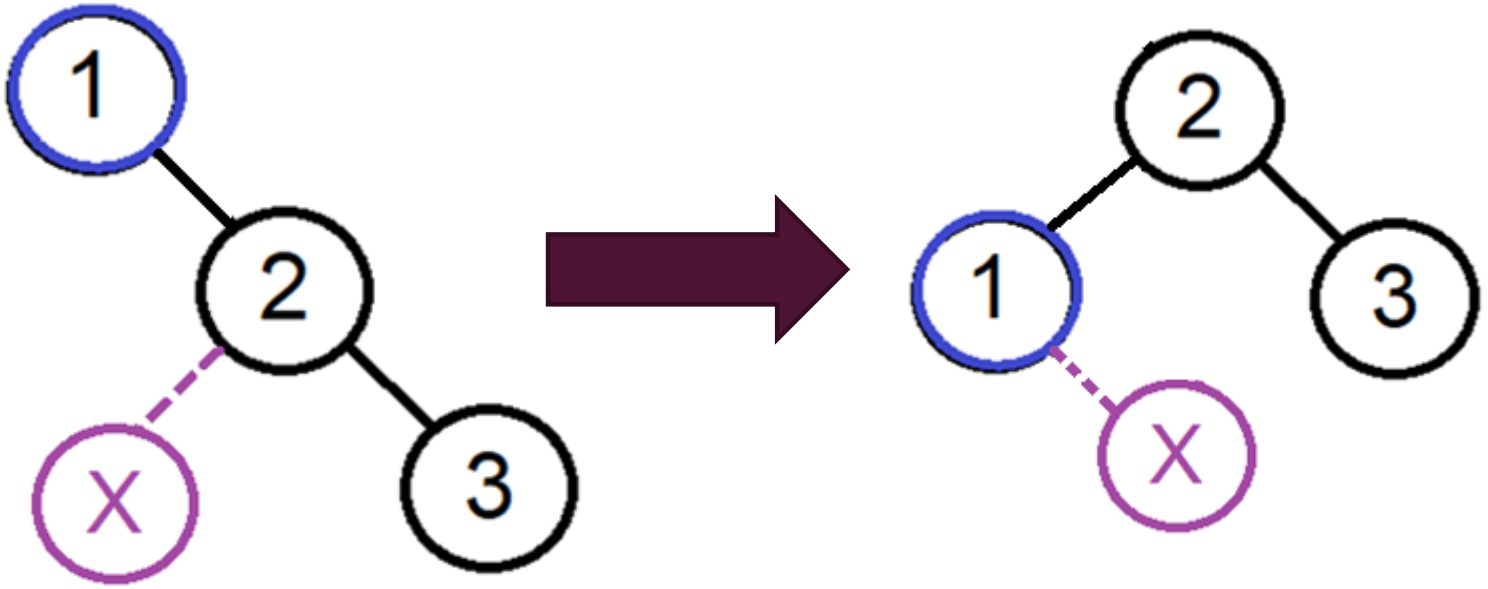


Gráfico 2

Rotação à esquerda (Caso Particular)

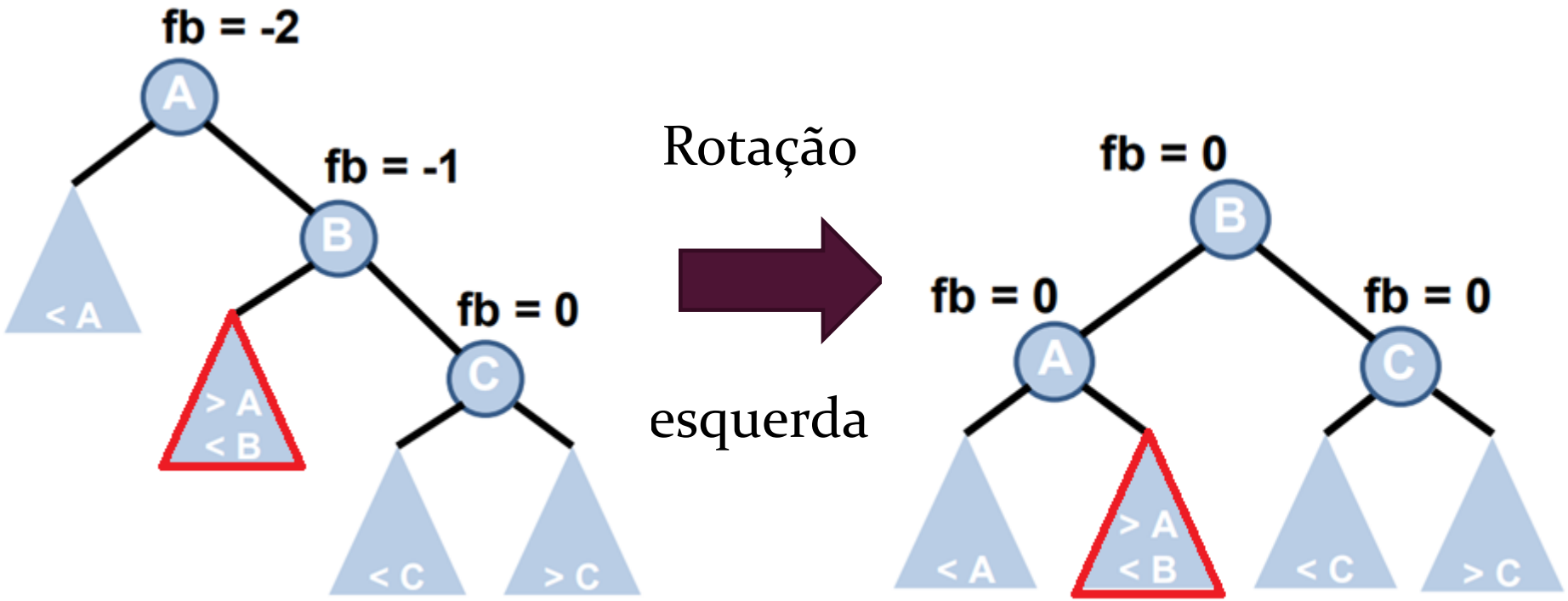
Em resumo:

1. O filho nó = 2 será a nova raiz.
2. Filho da esquerda do nó = 2 será o filho da direita do nó = 1.
3. O nó = 1 se tornará o filho da esquerda do nó = 2

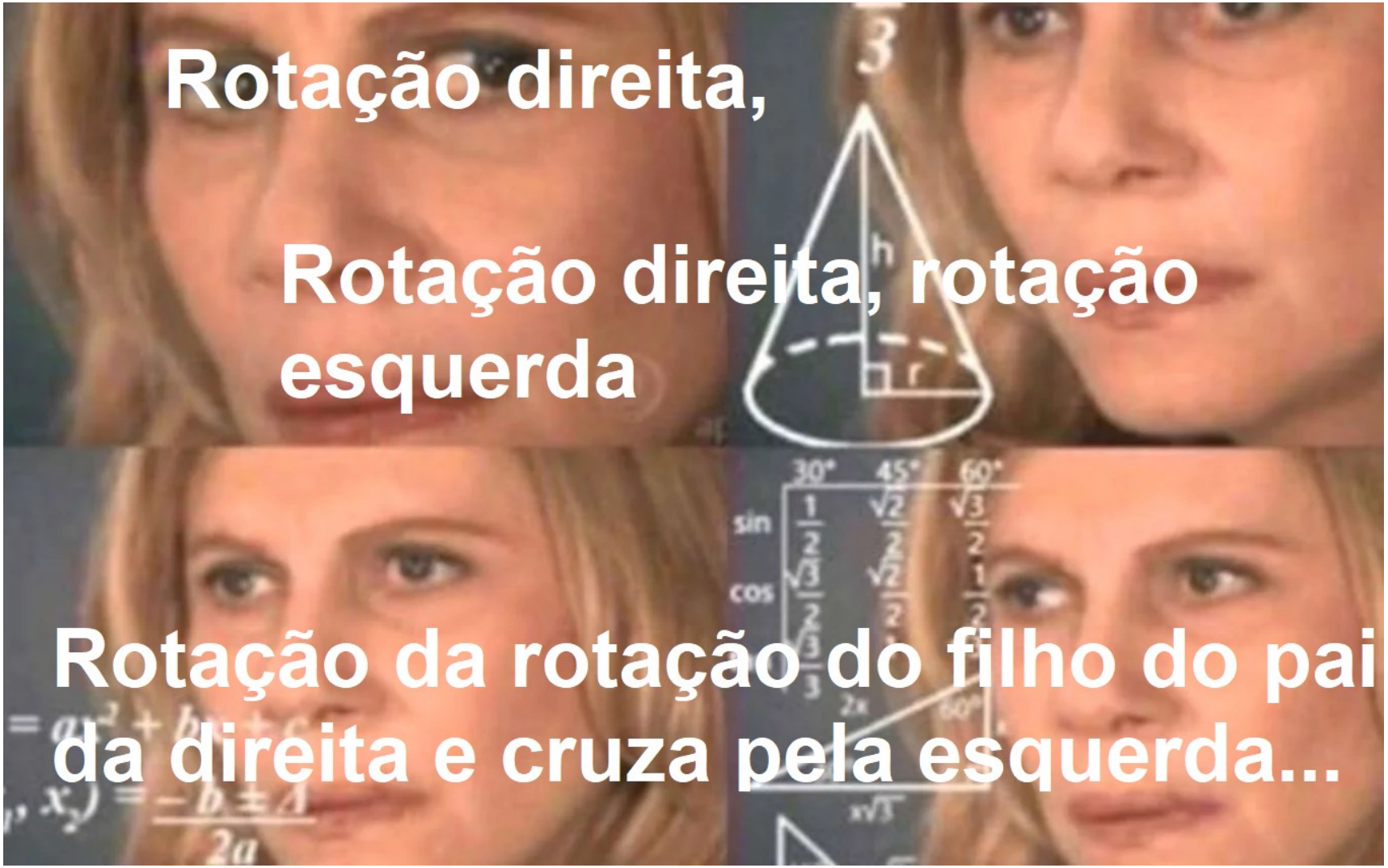


Rotação à esquerda (Caso Particular)

- **Nota:** As conexões com as sub-ároves de cada nó são preservadas, exceto com o nó filho (nó=B), cuja sub-árvore esquerda migra como sub-árvore direita para o nó A.

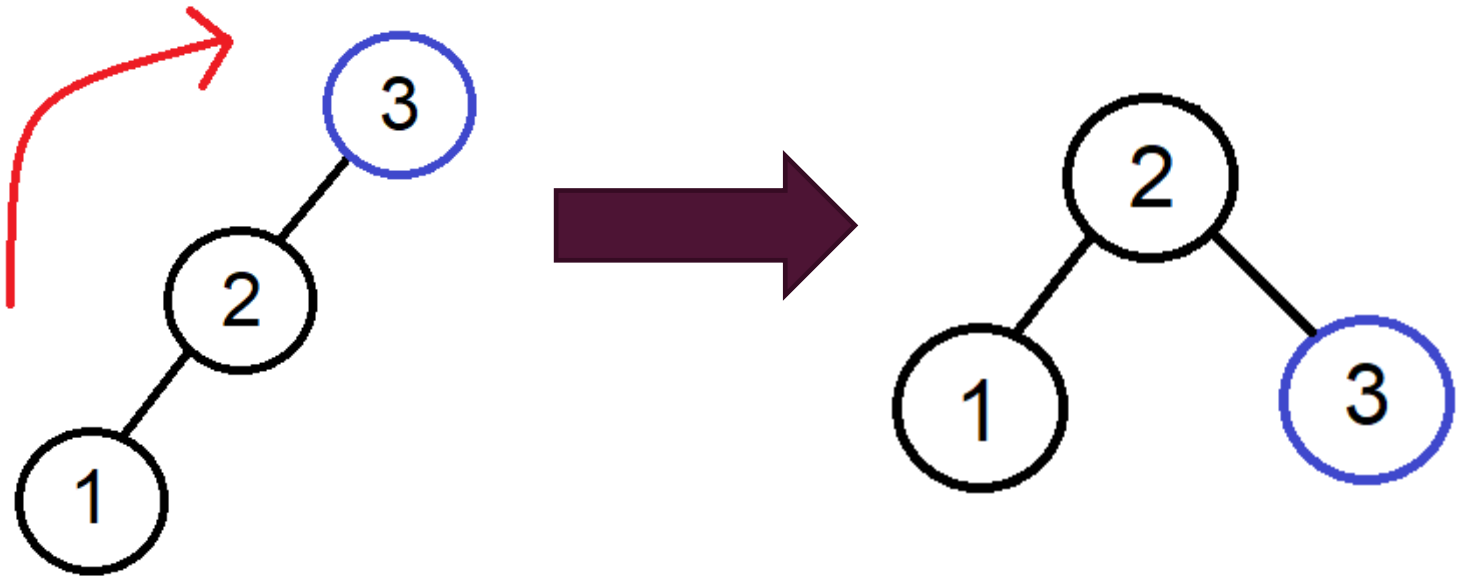


Rotações ...


Rotação direita,
Rotação direita, rotação esquerda
Rotação da rotação do filho do pai da direita e cruza pela esquerda...

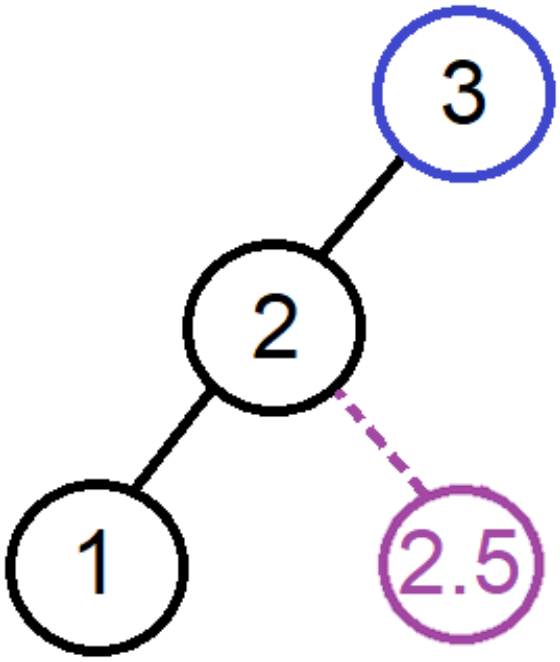
1.B) Rotação simples: direita

- Similar ao caso anterior (agora, a diagonal está à esquerda ao invés da direita)
- Rotação à direita: move os **nós que estão na sub-árvore da esquerda para direita**.
  - Filho da esquerda se torna a nova raiz.
  - Raiz original se torna o filho da direita da nova raiz.



1.B) Rotação simples: direita

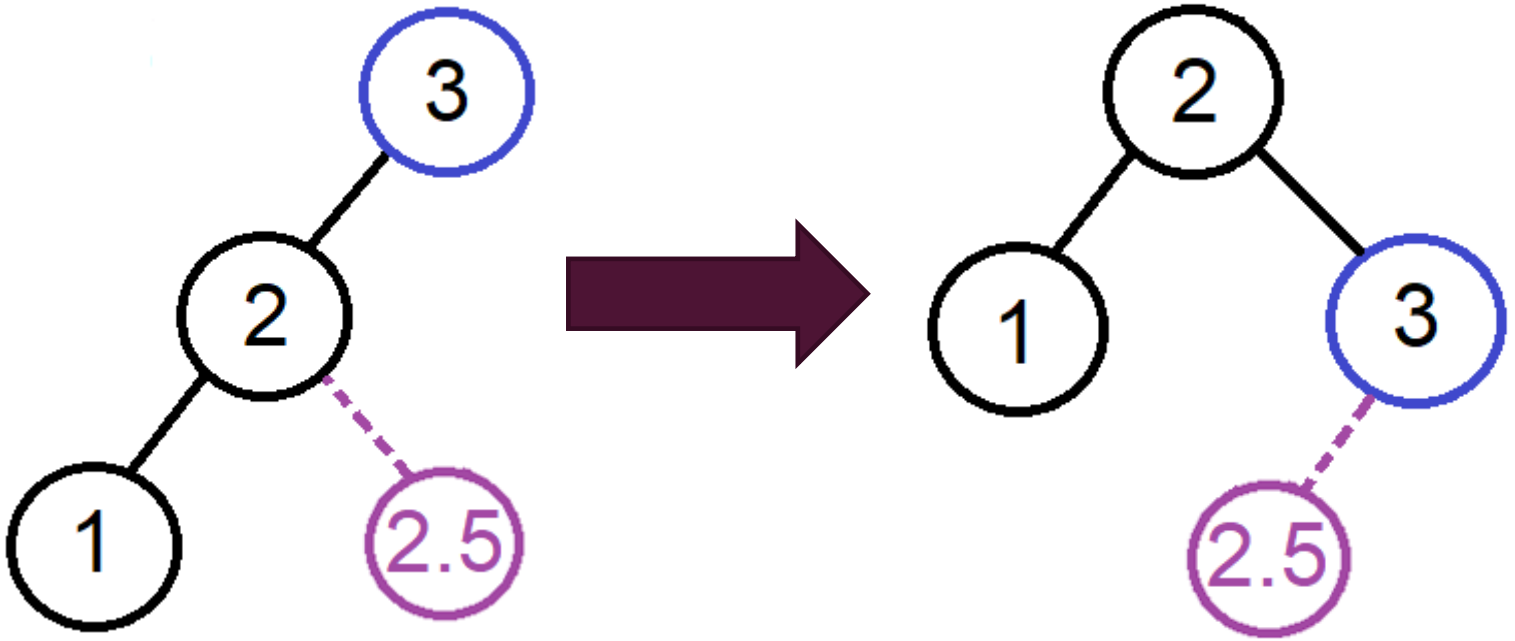
- **Caso particular**: Filho à esquerda (nó=2) já possui um filho à direita (nó=2.5)?
  - **Observação:** Elementos da sub-árvore da esquerda são **menores** que o nó raiz original (por definição de ABB)!



Rotação à direita (Caso Particular)

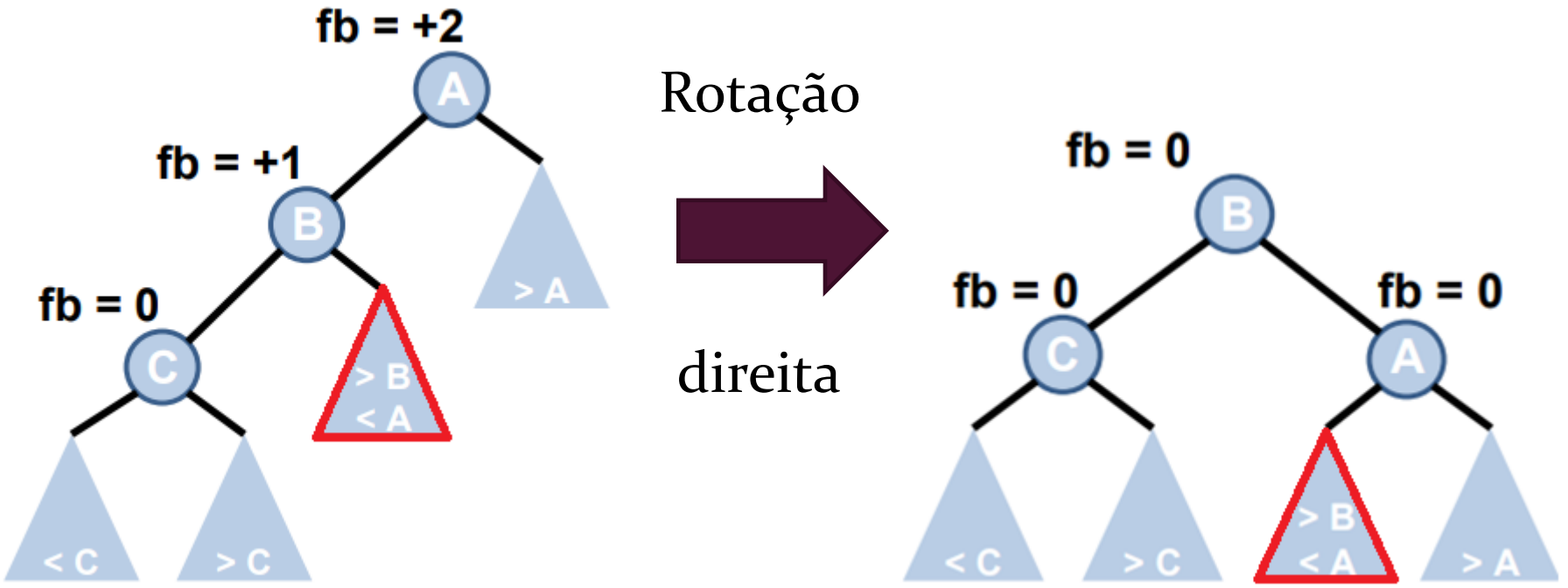
Em resumo:

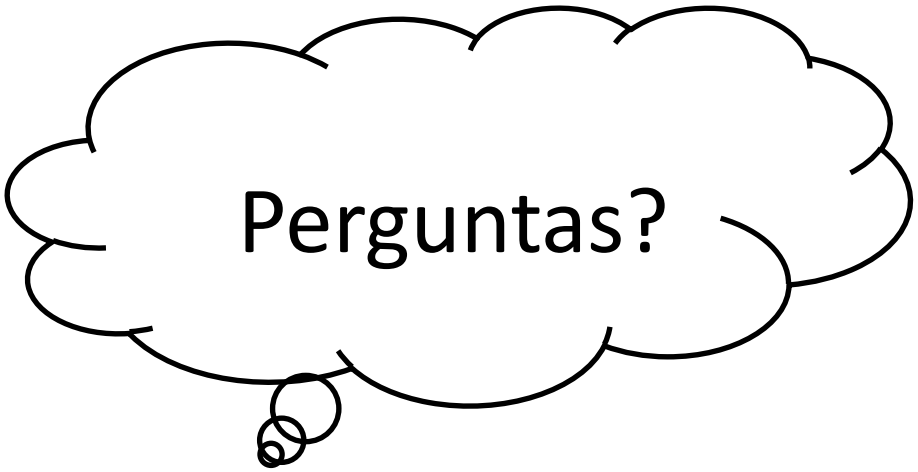
1. O filho nó = 2 será a nova raiz.
2. Filho da direita do nó = 2 será o filho da esquerda do nó = 3.
3. O nó = 3 se torna o filho da direita do nó = 2



Rotação à esquerda (Caso Particular)

- **Nota:** As conexões com as sub-ároves de cada nó são preservadas, exceto com o nó filho (nó=B), cuja sub-árvore direita migra como sub-árvore esquerda para o nó A.




Perguntas?

Balanceamento via Rotação - AVL

- **Quatro Tipos de Rotação:**

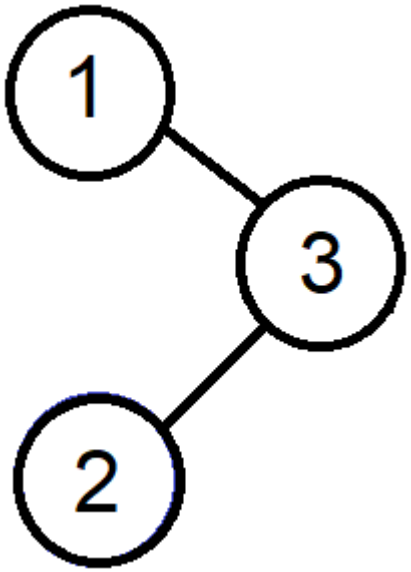
1. Rotação simples:
   A. Esquerda 
   B. Direita 

2. **Rotação dupla:**
   A. **Esquerda (direita-esquerda)**
   B. **Direita (esquerda-direita)**

2.A) Rotação composta à esquerda

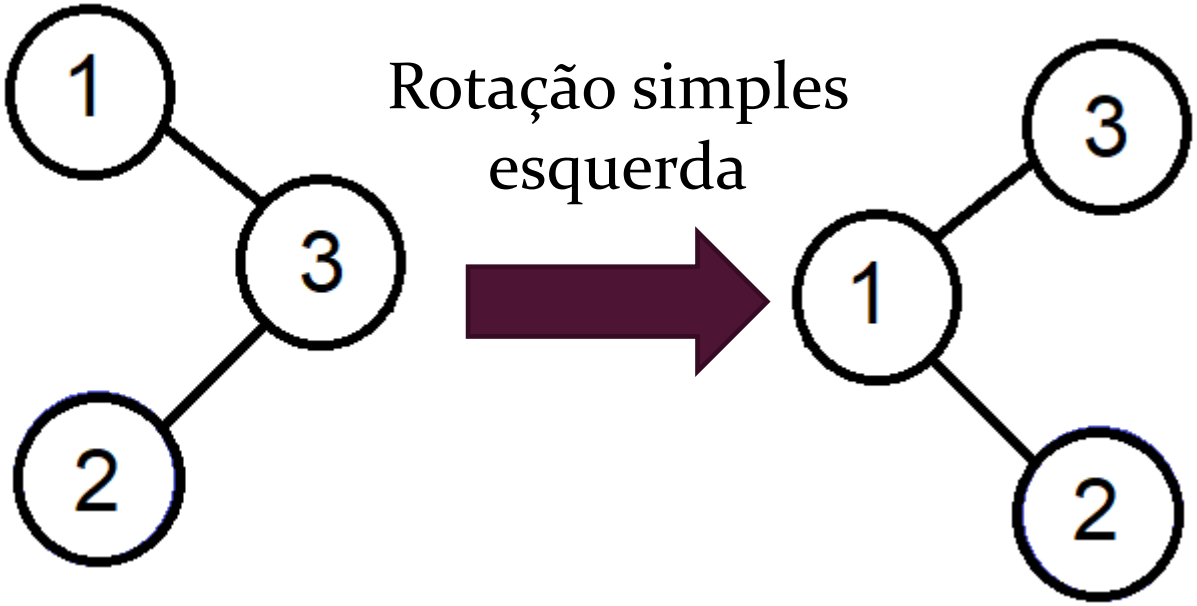
- Usada quando o nó desbalanceado (pai) e seu filho estão inclinados (na diagonal principal), mas no sentido inverso ao neto.
- Utilizada em situações em que apenas uma rotação simples (esq. ou dir.) não resolvem o problema.

Exemplo:



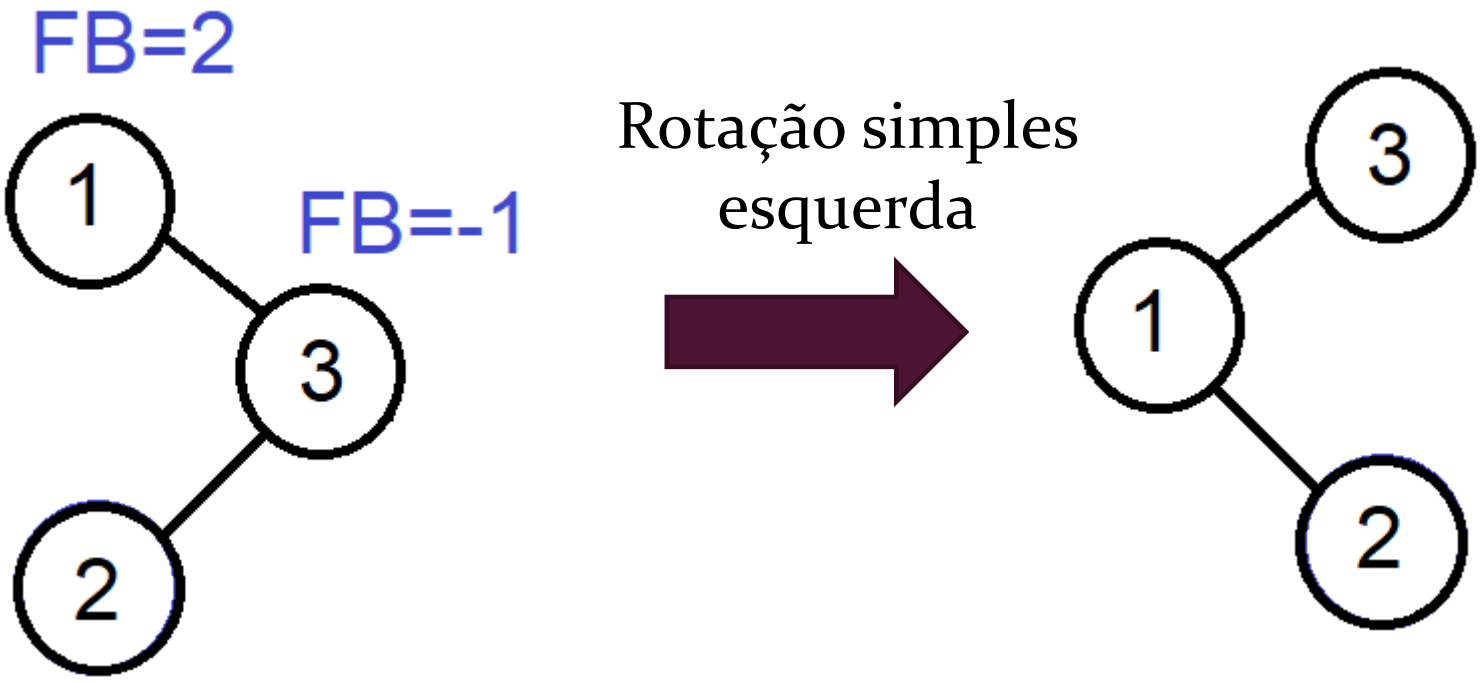
2.A) Rotação composta à esquerda

- Se aplicarmos rotação à esquerda no caso acima?
  - **Não resolvemos a questão do desequilíbrio!**



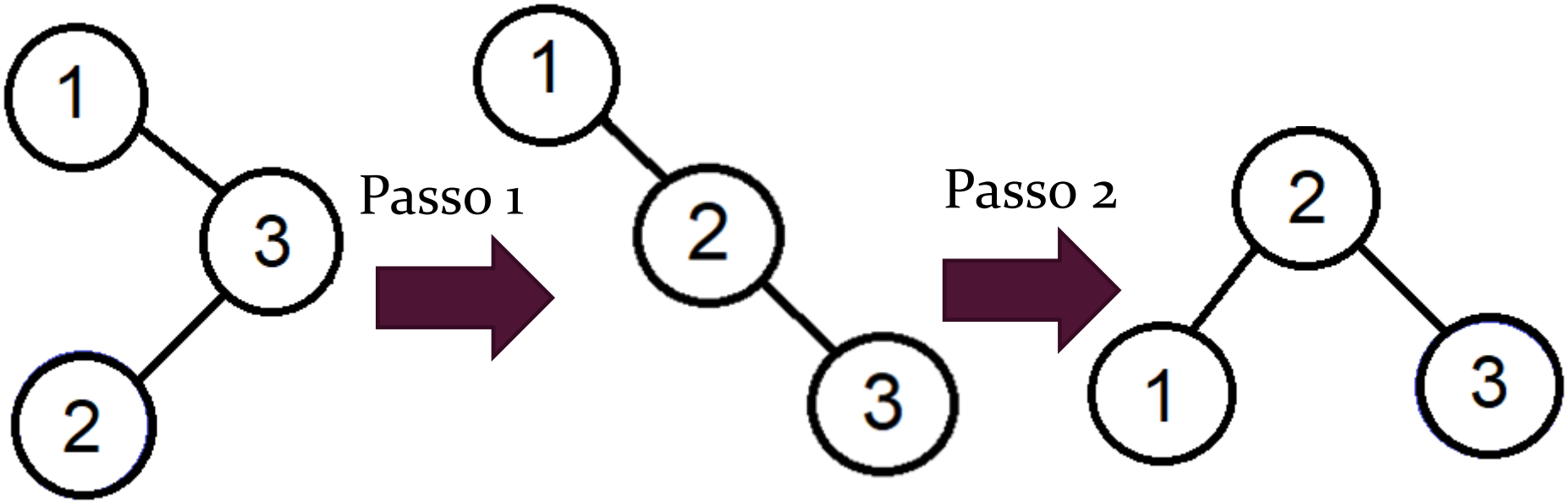
2.A) Rotação composta à esquerda

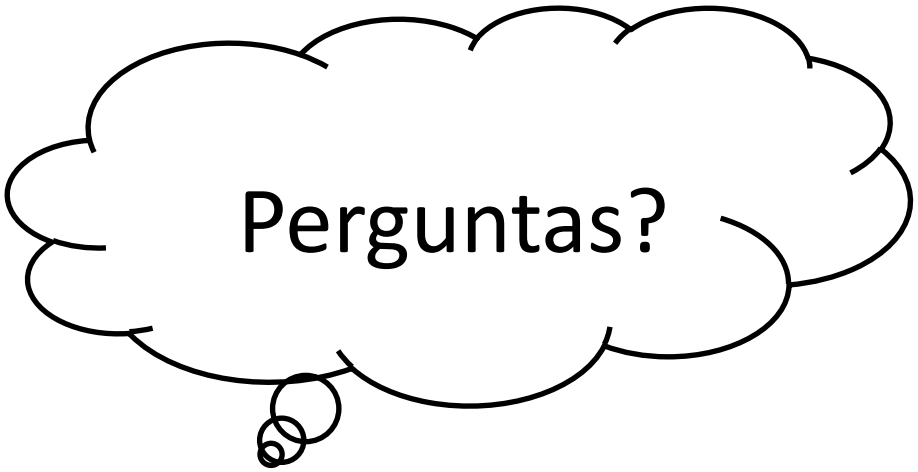
- Como detectar se uma rotação simples resolve o problema?
  - Avaliar os FBs da **raiz inicial** e **nova raiz**.
    - Se FB(raiz inicial) positiva e FB (nova raiz) negativa ➔ rotação simples não é efetiva!



2.A) Rotação composta à esquerda

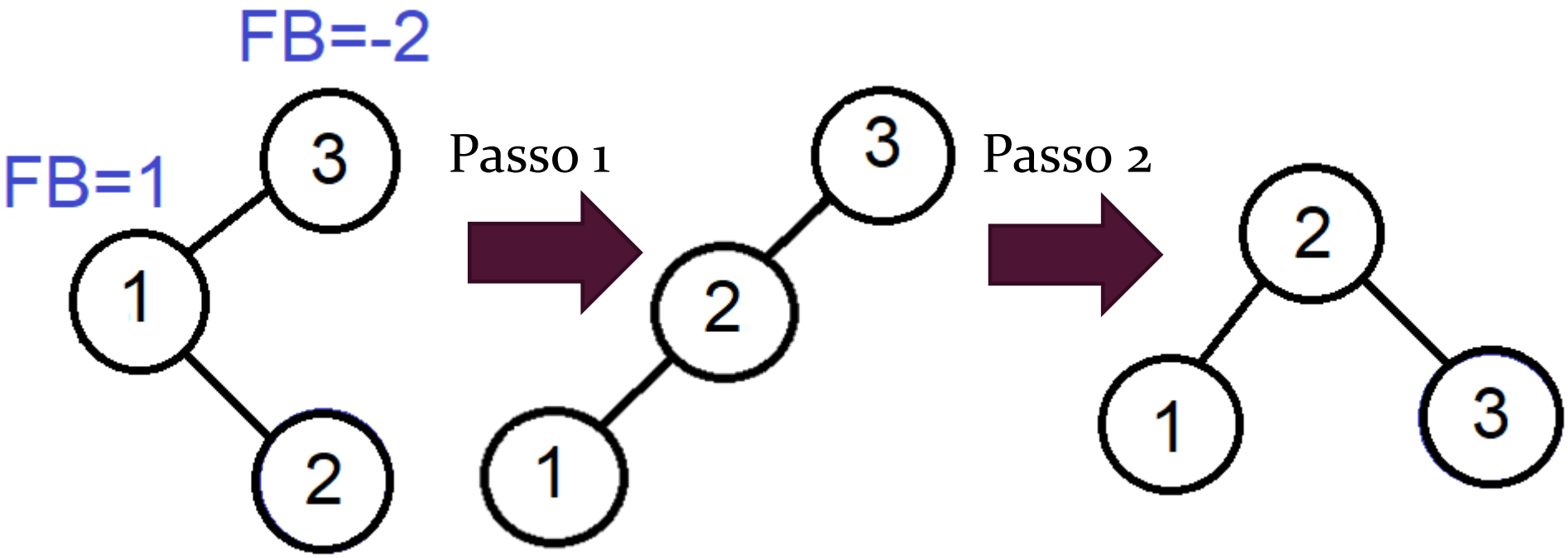
- **SOLUÇÃO:** Rotação dupla à esquerda
  1. Rotação à direita na sub-árvore da direita, mantendo a raiz.
  2. Rotação à esquerda na árvore resultante do passo anterior.




Perguntas?

2.B) Rotação composta à direita

- Similar ao caso anterior.
- Rotação dupla à direita:
  1. Rotação à esquerda na sub-árvore da esquerda, mantendo a raiz
  2. Rotação à direita na árvore resultante do passo anterior.



Algoritmo para classificar as rotações

1. Calcular FB(n), para cada nó n da árvore.
2. Se FB(n) é -1, 0, ou 1 para todo nó n então a árvore está equilibrada.
3. Se FB(n) > 1:
   Se a sub-árvore da direita possui FB(n->dir) < 0
      Rotação dupla à esquerda
   Senão
      Rotação simples à esquerda
   Senão
   Se sub-árvore da esquerda tem FB(n->esq) > 0
      Rotação dupla à direita
   Senão
      Rotação simples à direita