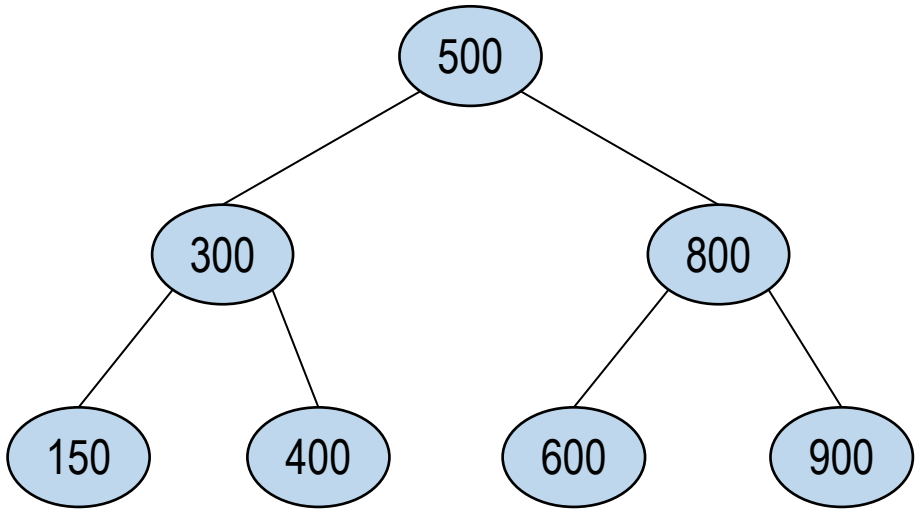
ÁRVORES AVL

Estruturas de Dados e Seus Algoritmos
Turma de verão 2023

Prof. Taiane Ramos

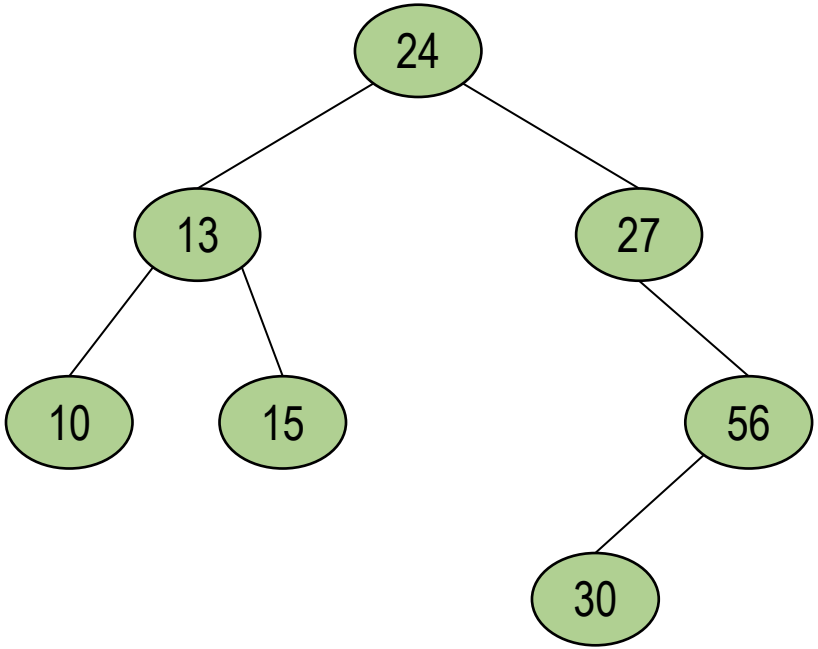
RECAPITULANDO:
ÁRVORES BINÁRIAS DE BUSCA

- Apresentam uma relação de ordem
- A ordem é definida pela chave
- Operações:
  - inserir
  - consultar
  - excluir



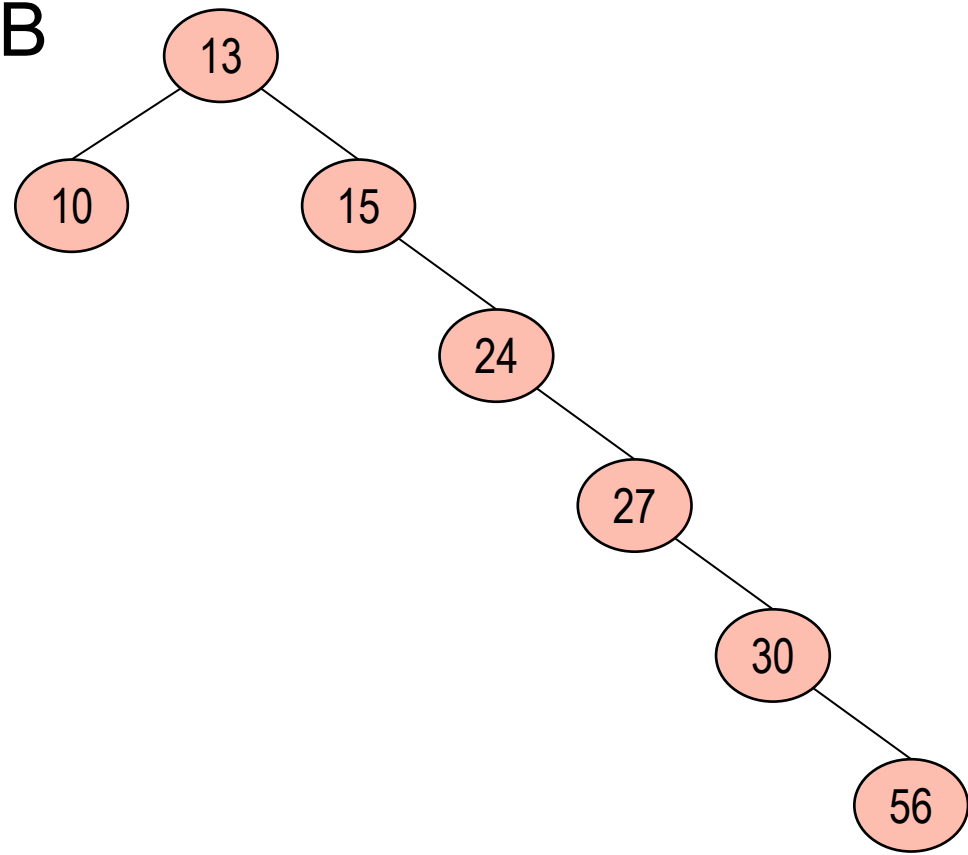
PROBLEMAS COM ÁRVORE BINÁRIA DE BUSCA (ABB)

- Desbalanceamento progressivo
- Exemplo:
  - Inserção: 24, 27, 13, 10, 56, 15, 30



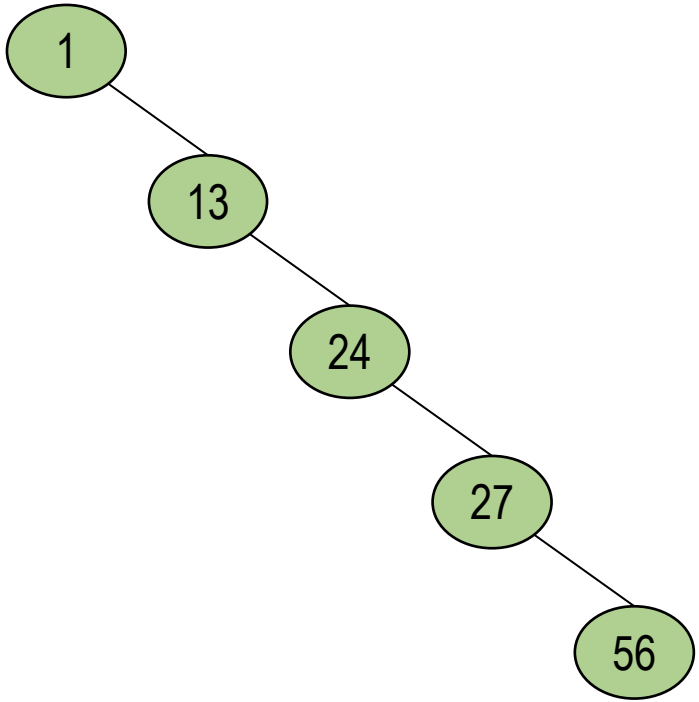
PROBLEMAS COM ABB

- Desbalanceamento progressivo
- Exemplo:
  - Inserção: 13, 10, 15, 24, 27, 30, 56



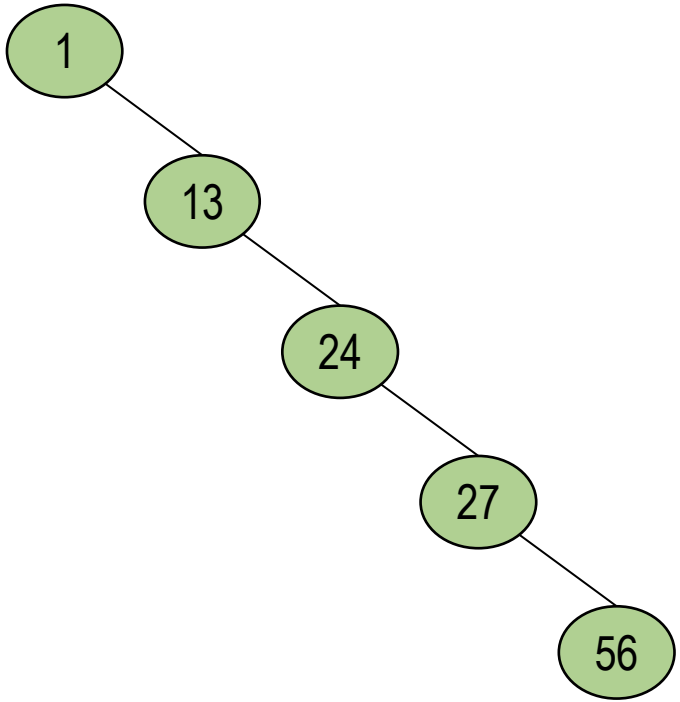
PROBLEMAS COM ABB

- Desbalanceamento progressivo
- Exemplo:
  - inserção: 1, 13, 24, 27, 56



CONSEQUÊNCIA

- Buscas ficam mais custosas



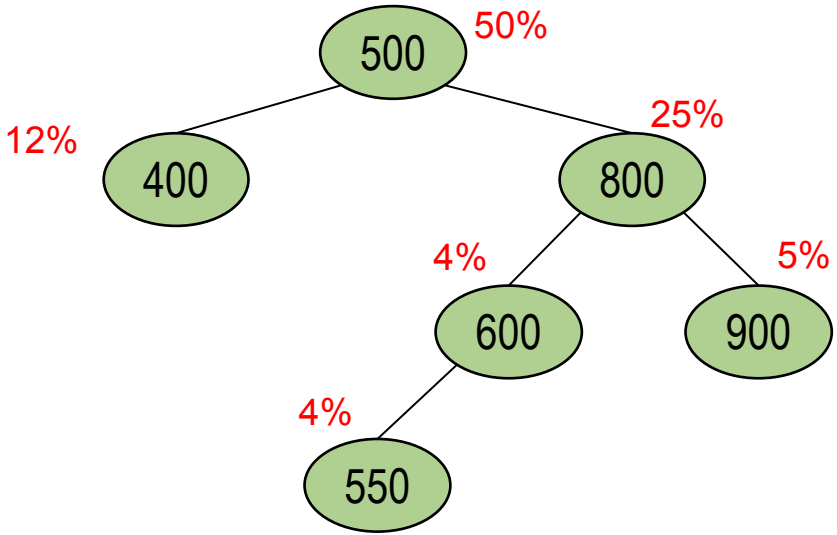
BALANCEAMENTO DE ÁRVORES

- Distribuição eficiente dos nós
- Objetivo:
  - Otimizar as operações de consulta
- Distribuição
  - Uniforme
    - árvore balanceada por altura
  - Não uniforme
    - chaves mais solicitadas mais perto da raiz

POR FREQUÊNCIA X POR ALTURA

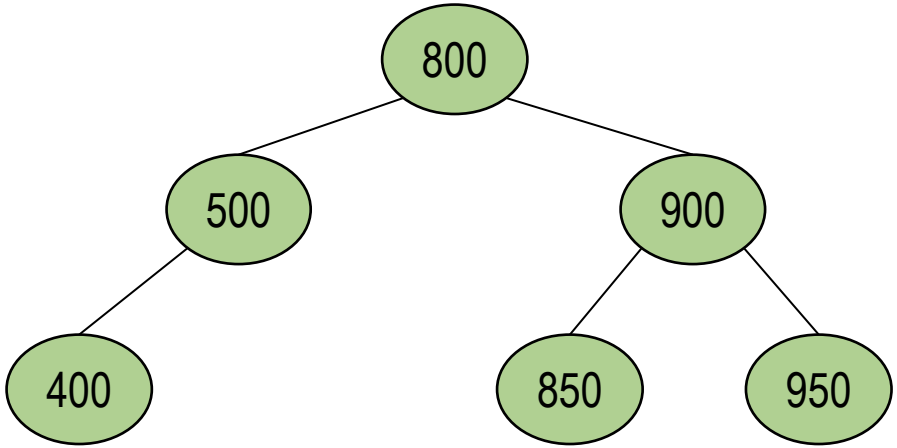
Splay

- Nós mais acessados ficam perto da raiz



AVL, Rubro-Negras

- Diferença das alturas das subárvores não excedem um determinado valor

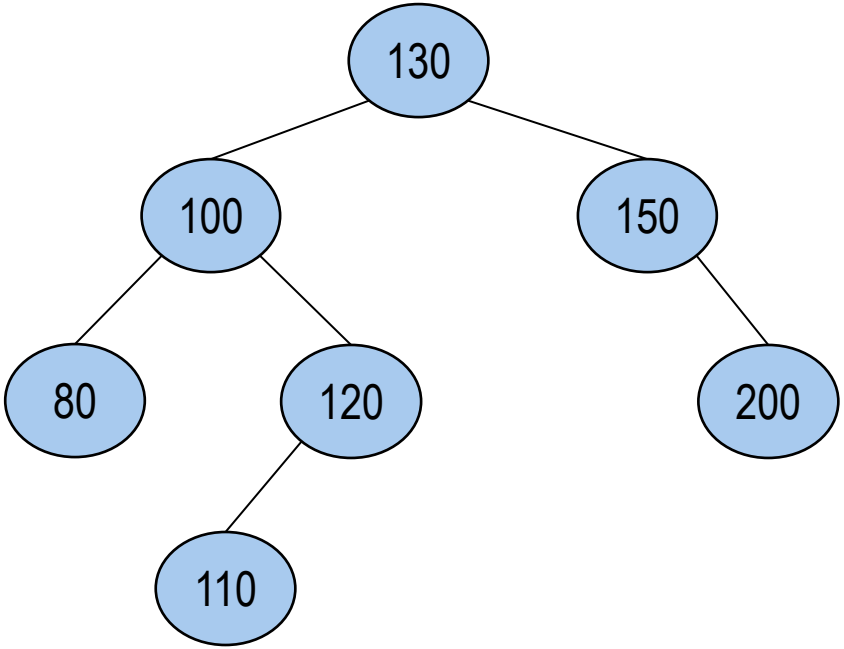


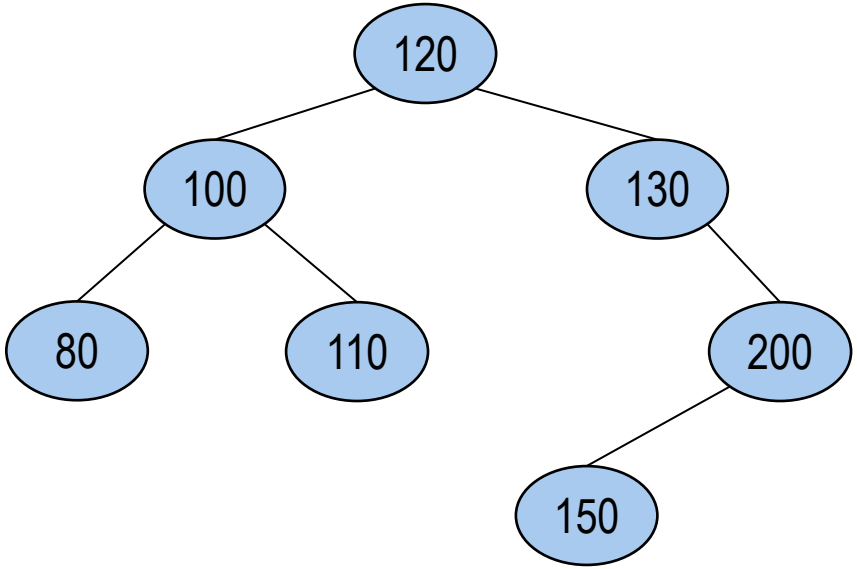
ÁRVORES AVL
ADELSON-VELSKII E LANDIS (1962)

Uma **árvore binária de busca** (ABB) é uma **AVL** quando, para qualquer um de seus nós, **a diferença** entre as **alturas de suas subárvore direita e esquerda** é no **máximo 1.**

EXERCÍCIO

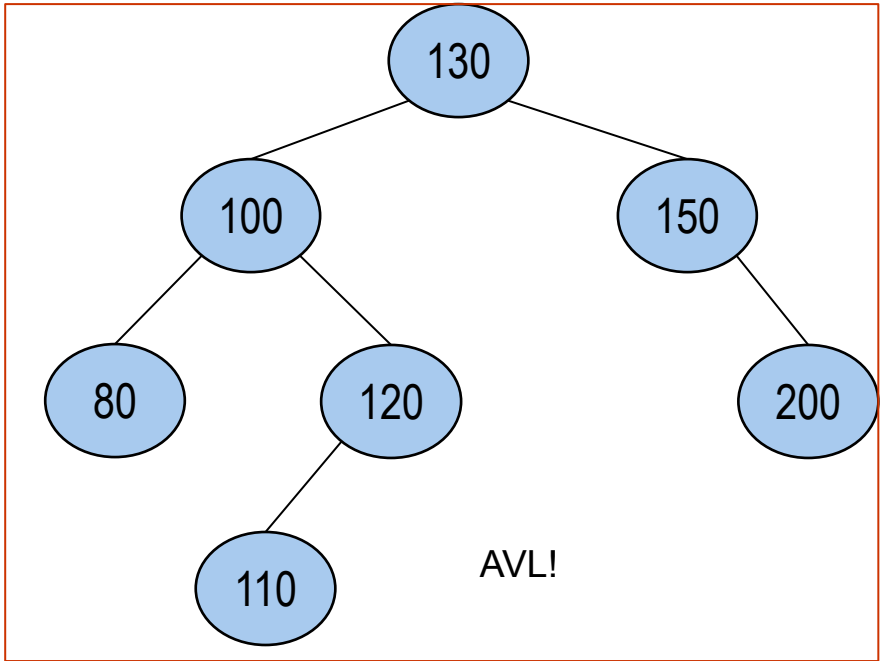
- Verifique quais das ABB são AVL

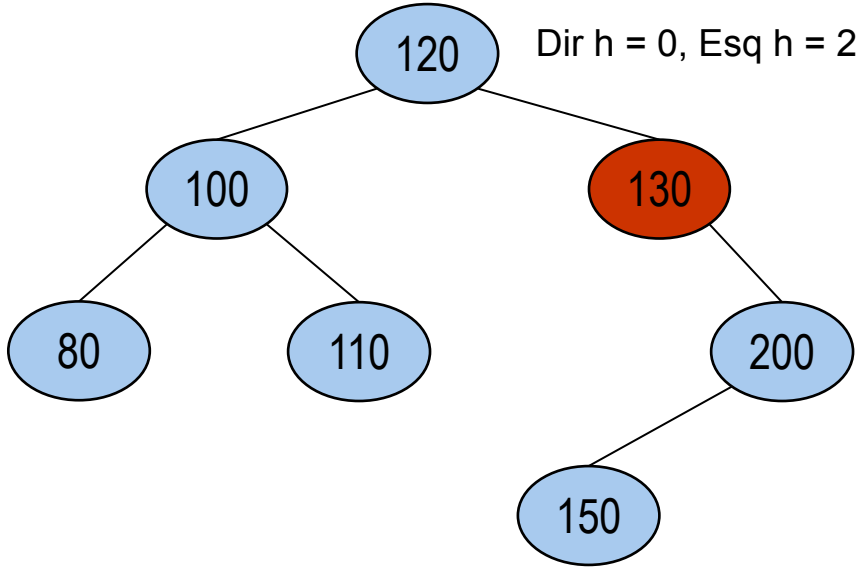




EXERCÍCIO

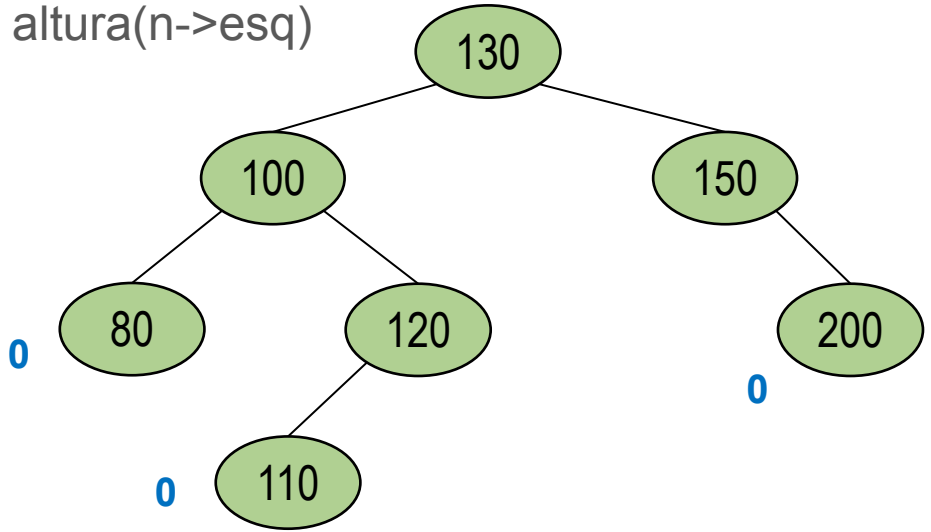
- Verifique quais das ABB são AVL





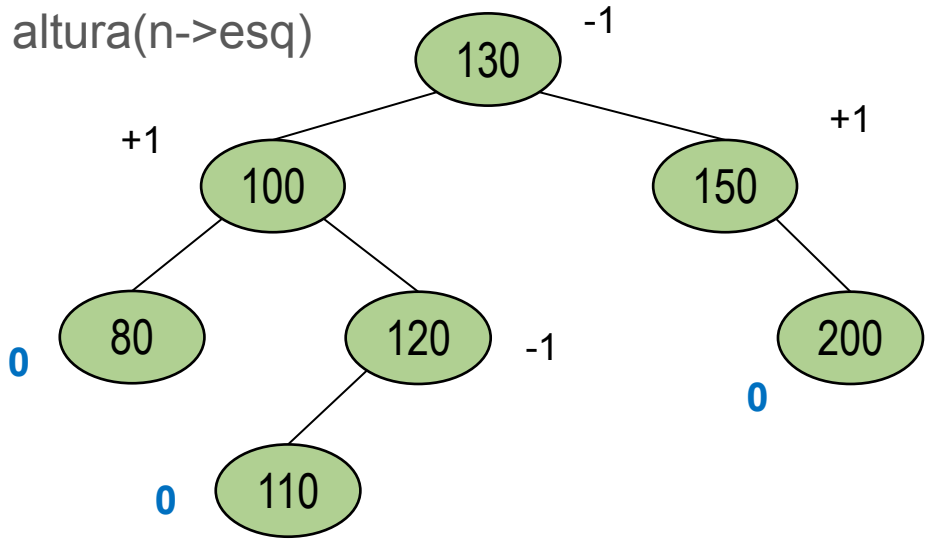
FATOR DE BALANCEAMENTO (FB)

- **Fator de Balanceamento:** diferença entre altura da subárvore direita e esquerda
- FB(n) = altura(n->dir) – altura(n->esq)



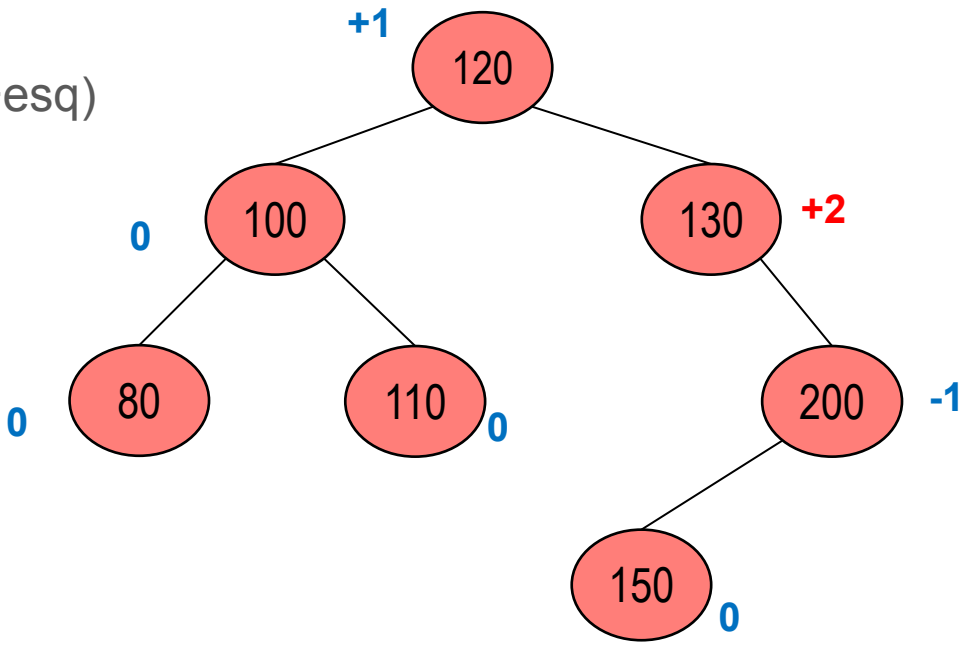
FATOR DE BALANCEAMENTO (FB)

- **Fator de Balanceamento:** diferença entre altura da subárvore direita e esquerda
- FB(n) = altura(n->dir) – altura(n->esq)



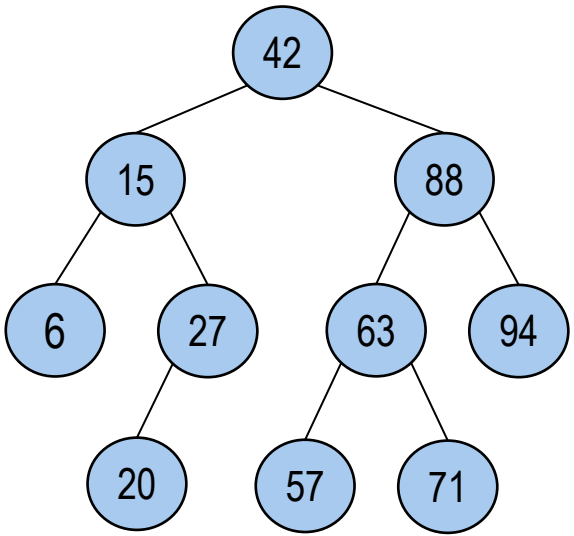
FATOR DE BALANCEAMENTO (FB)

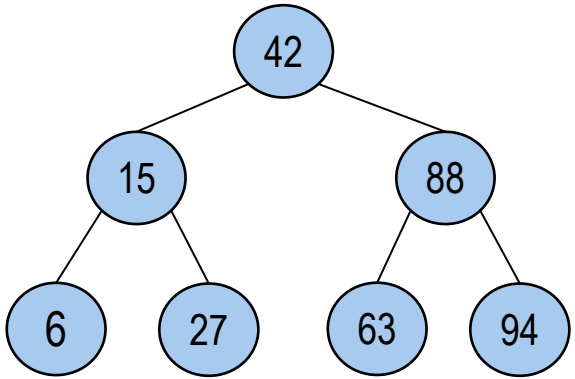
- **Fator de Balanceamento:** diferença entre altura da subárvore direita e esquerda
- FB(n) = altura(n->dir) – altura(n->esq)

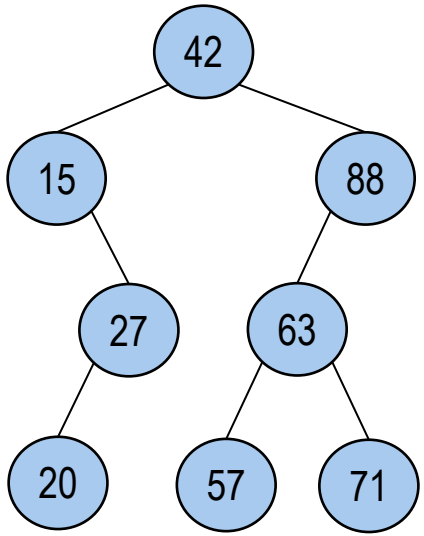


EXERCÍCIO

- Verifique quais das ABB são AVL, calculando o FB de cada nó:

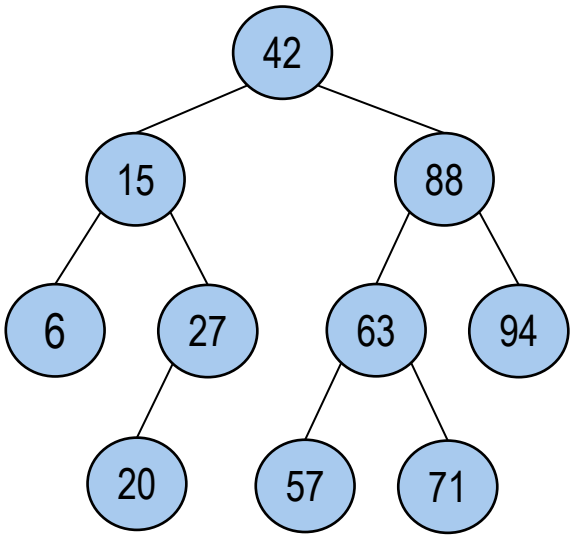


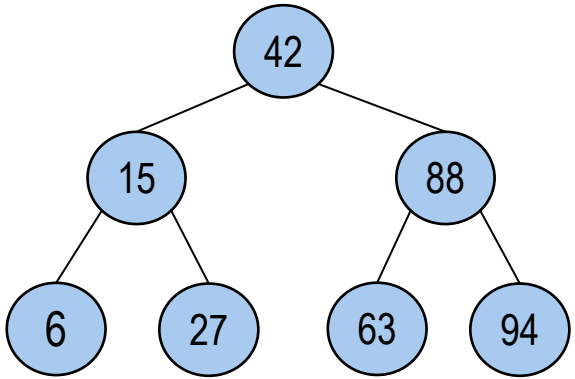


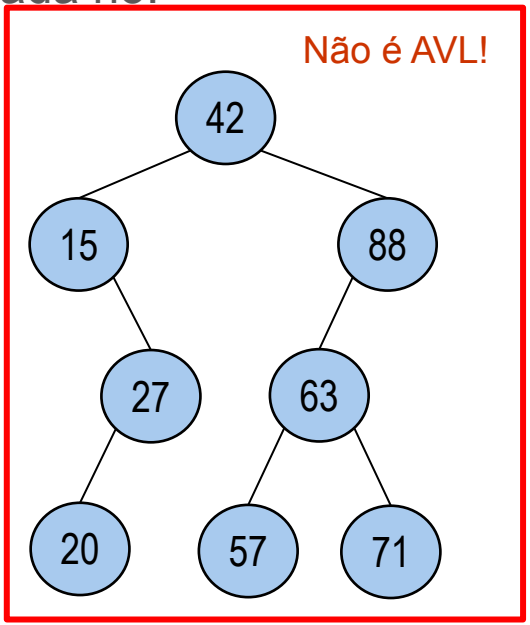


EXERCÍCIO

- Verifique quais das ABB são AVL, calculando o FB de cada nó:

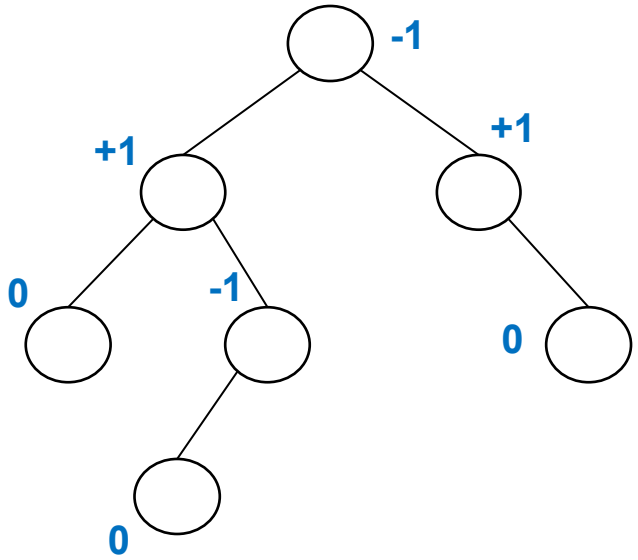




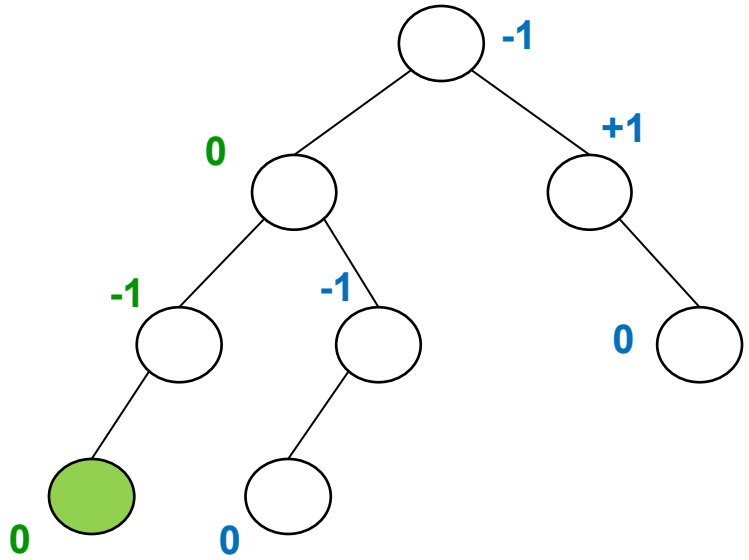


OPERAÇÕES

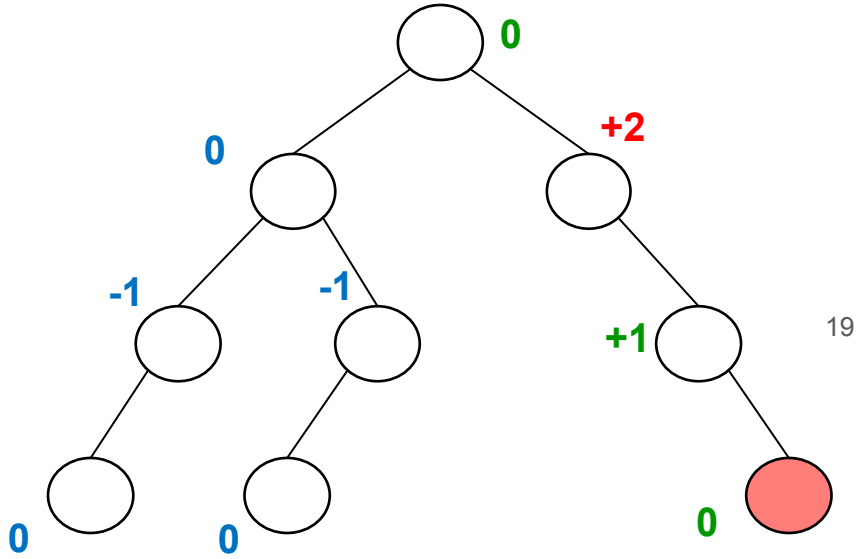
- Inserção e Exclusão devem preservar as propriedades da AVL
- A Busca não muda



INSERÇÃO

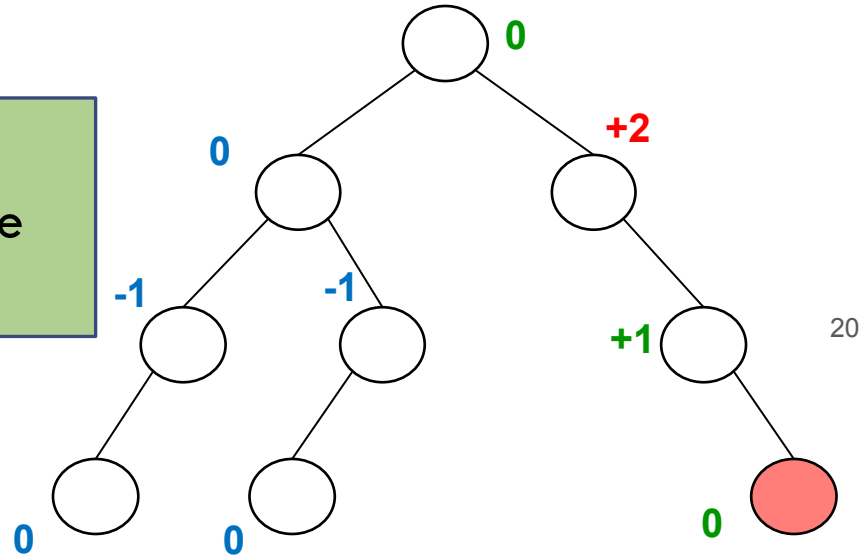


INSERÇÃO



INSERÇÃO

Reestruturar Árvore



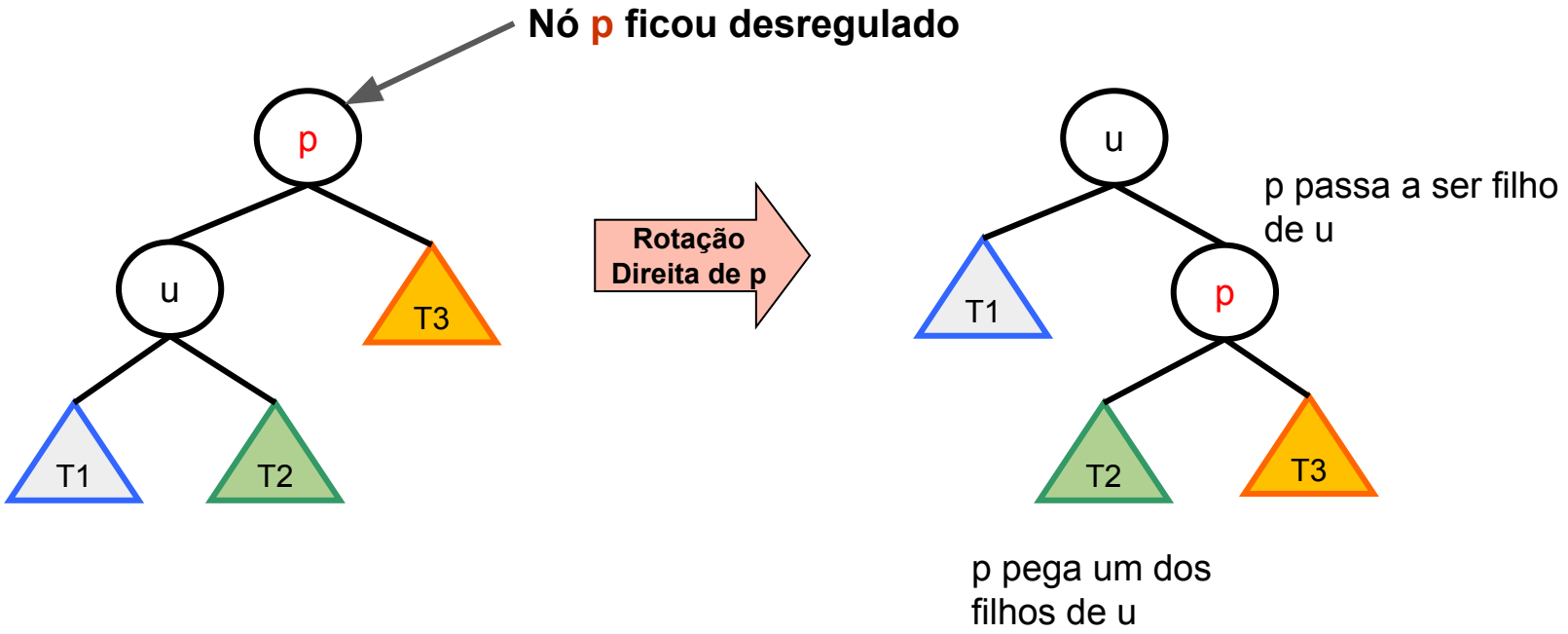
Rotação

- **Rotação** garante que a árvore resultante é uma árvore binária de busca válida e é uma árvore AVL válida

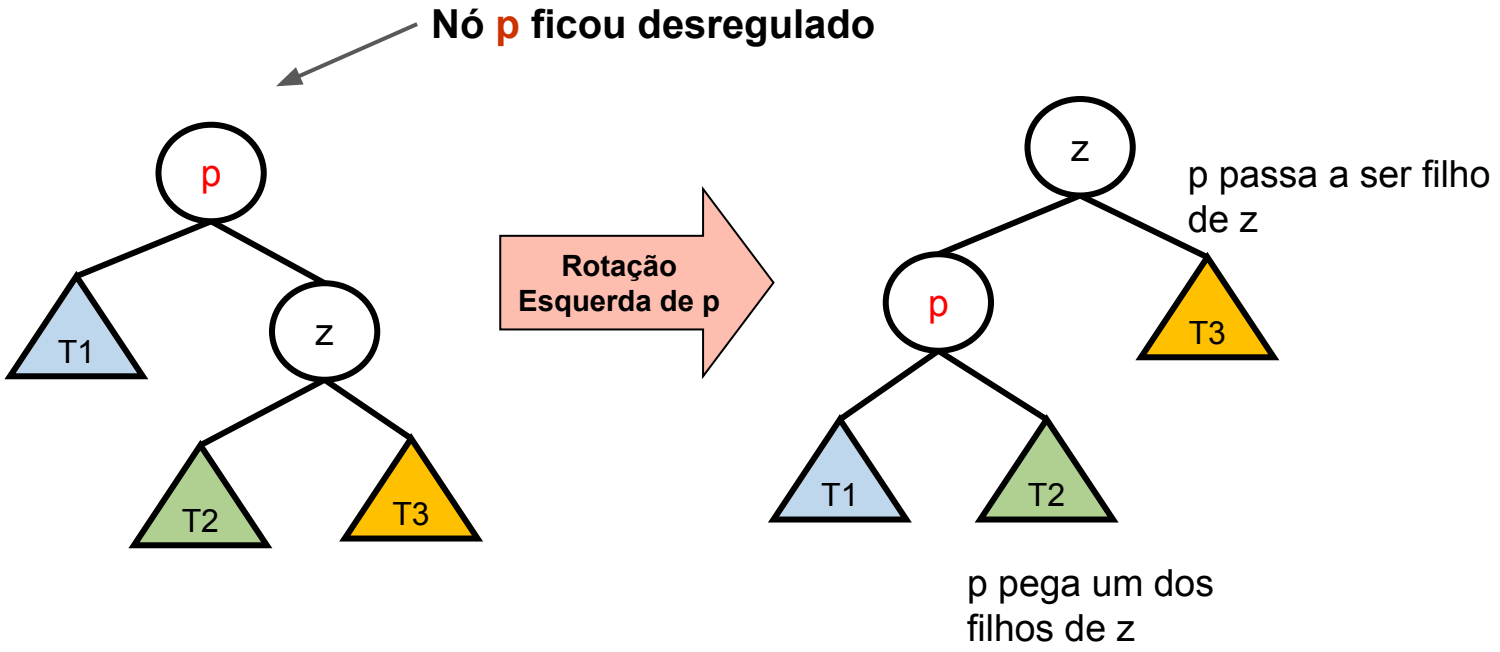
BALANCEAMENTO DE ÁRVORES AVL POR ROTAÇÃO

- Rotação Simples
  - Direita
  - Esquerda
- Rotação Dupla
  - Direita (esquerda-direita)
  - Esquerda (direita-esquerda)

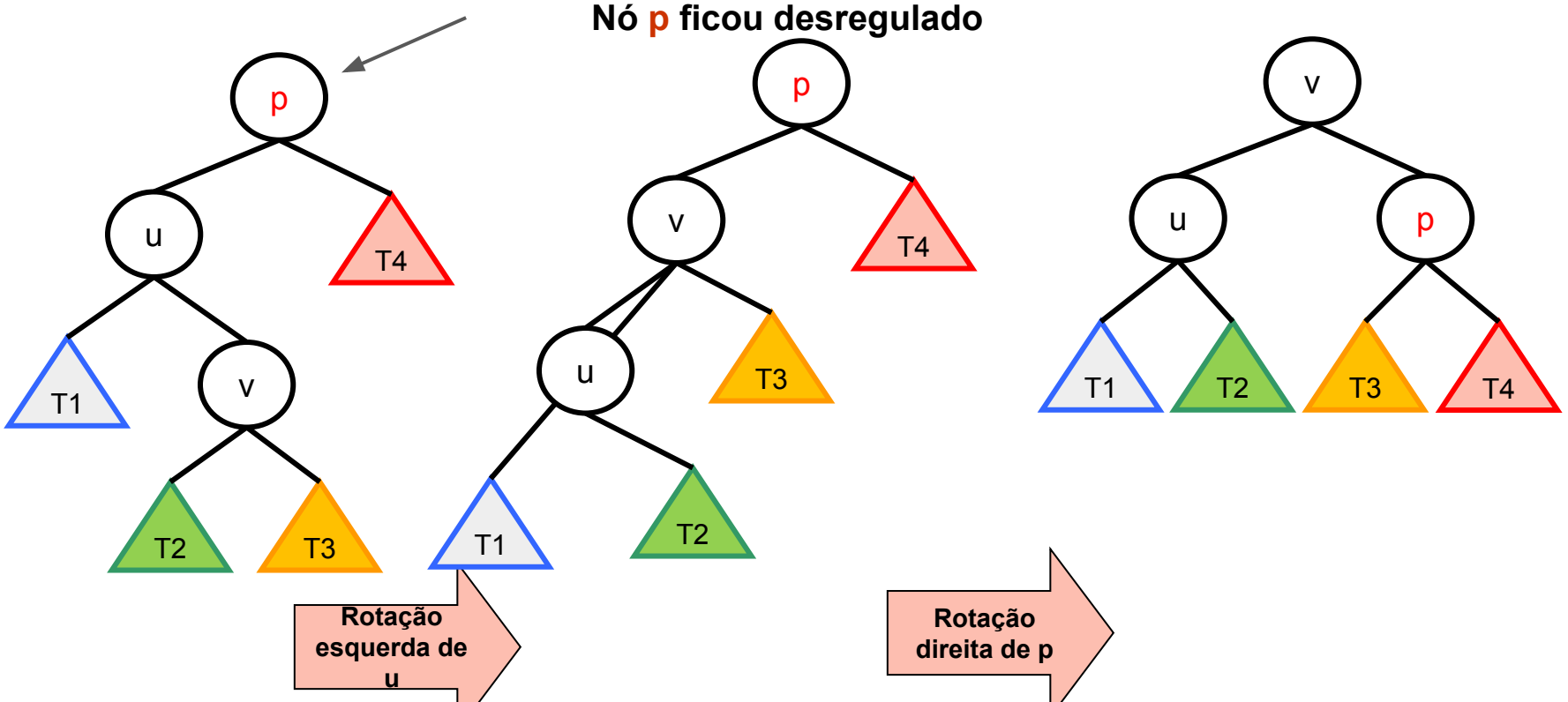
ROTAÇÃO DIREITA



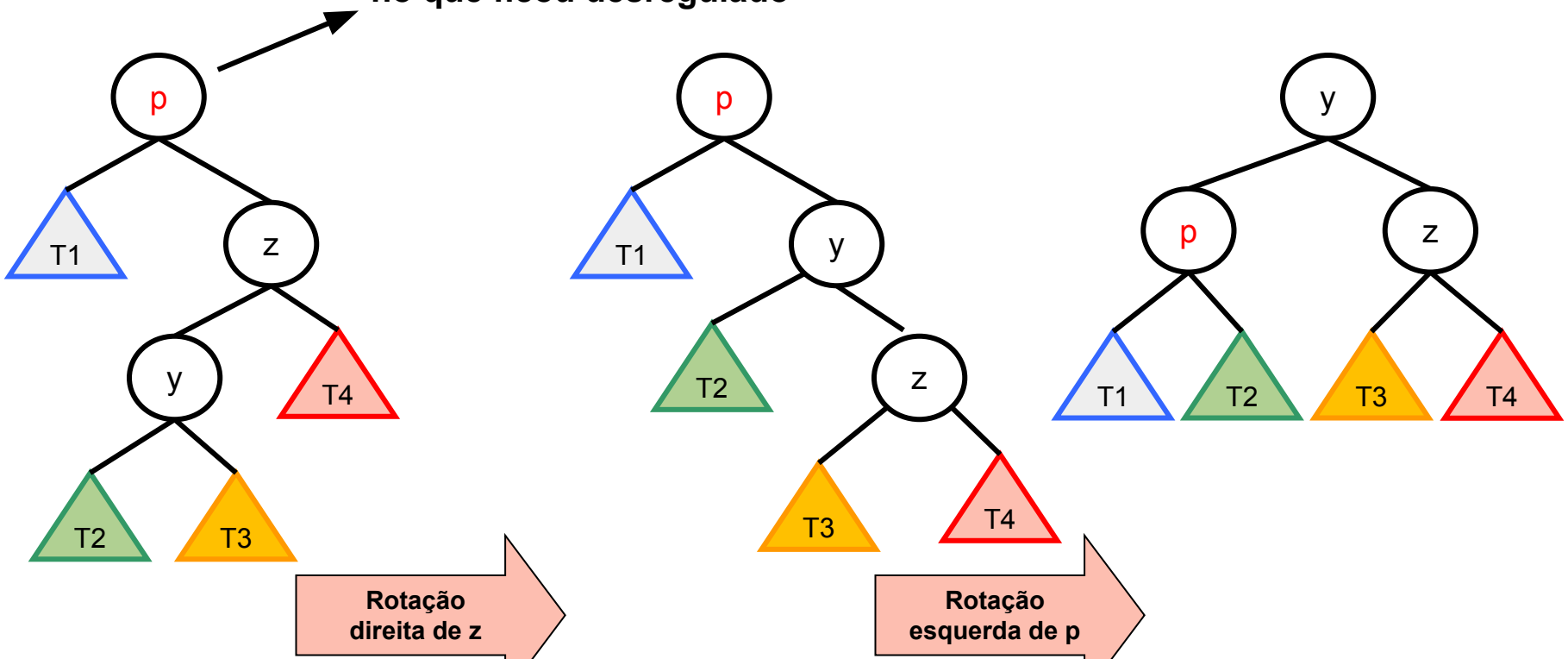
ROTAÇÃO ESQUERDA



ROTAÇÃO DUPLA DIREITA (ESQUERDA-DIREITA)



ROTAÇÃO DUPLA ESQUERDA (DIREITA-ESQUERDA)



INSERÇÃO DE NÓS EM ÁRVORES AVL

- Percorrer a árvore verificando se a chave já existe ou não
  - Se existe, retorna
  - Caso contrário, buscar o local correto de inserção do novo nó
- Se a árvore ficar desbalanceada
  - Descobrir qual a operação de rotação a ser executada
  - Executar a rotação

COMO ESCOLHER A ROTAÇÃO?

- Fator de Balanceamento FB = h(subarv-direita) – h(subarv-esquerda)
- Se FB **positivo** (subárvore da direita é maior):
  - rotações à **esquerda**
- Se FB **negativo** (subárvore da esquerda é maior)
  - rotações à **direita**

QUANDO APLICAR?

- **Nó com FB = -2 e filho com FB = -1 ou 0**:
  - rotação do nó com FB = -2 p/ **direita**
- **Nó com FB = +2 e filho com FB = +1 ou 0**:
  - rotação do nó com FB = +2 p/ **esquerda**
- **Nó com FB = -2 e filho com FB = +1**:
  - rotação do nó com FB = +1 p/ **esquerda**, e
  - rotação do nó com FB = -2 p/ **direita**
- **Nó com FB = +2 e filho com FB = -1**:
  - rotação do nó com FB = -1 p/ **direita**, e
  - rotação do nó com FB = +2 p/ **esquerda**

QUANDO APLICAR?

- **Nó com FB = -2 e filho com FB = -1 ou 0**:
  - rotação do nó com FB = -2 p/ direita
- **Nó com FB = +2 e filho com FB = +1 ou 0**:
  - rotação do nó com FB = +2 p/ esquerda

Mesmo sinal: rotação simples

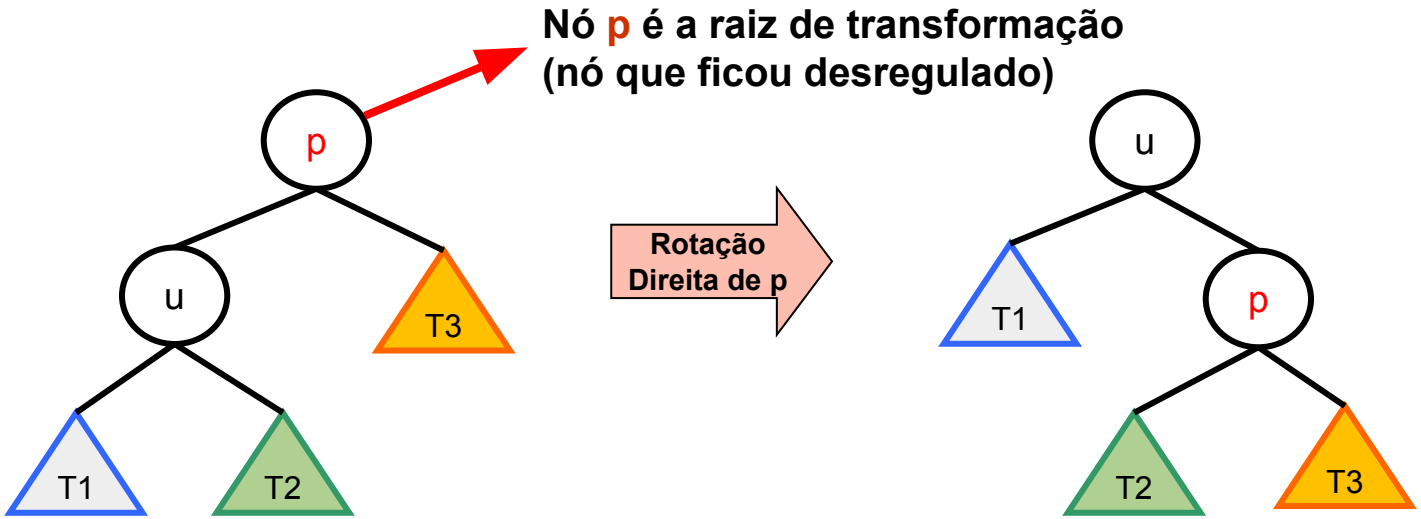
- **Nó com FB = -2 e filho com FB = +1**:
  - rotação do nó com FB = +1 p/ esquerda, e
  - rotação do nó com FB = -2 p/ direita
- **Nó com FB = +2 e filho com FB = -1**:
  - rotação do nó com FB = -1 p/ direita, e
  - rotação do nó com FB = +2 p/ esquerda

Sinais opostos: rotação dupla

ROTAÇÃO SIMPLES DIREITA

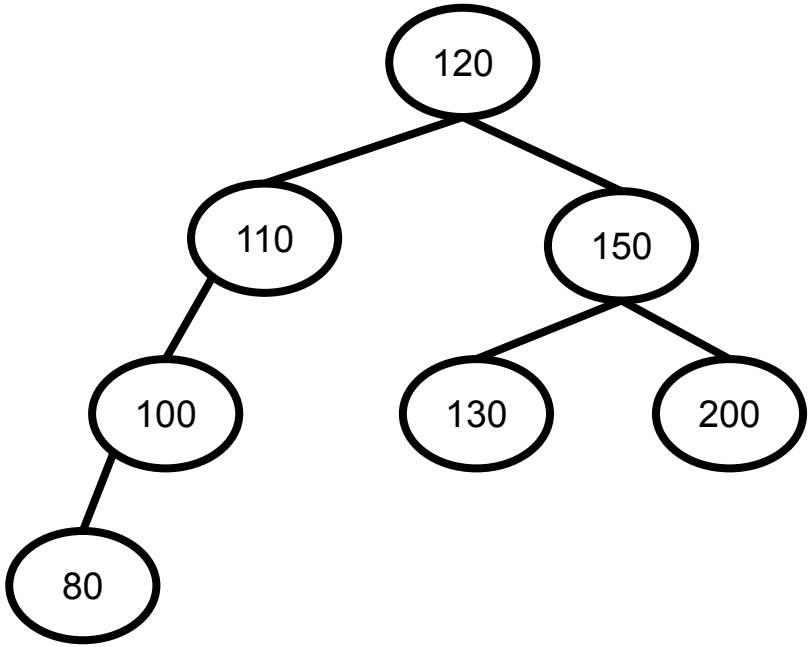
- **Nó com FB = -2 e filho com FB = -1 ou 0**:
  - rotação do nó com FB = -2 p/ direita

ROTAÇÃO DIREITA

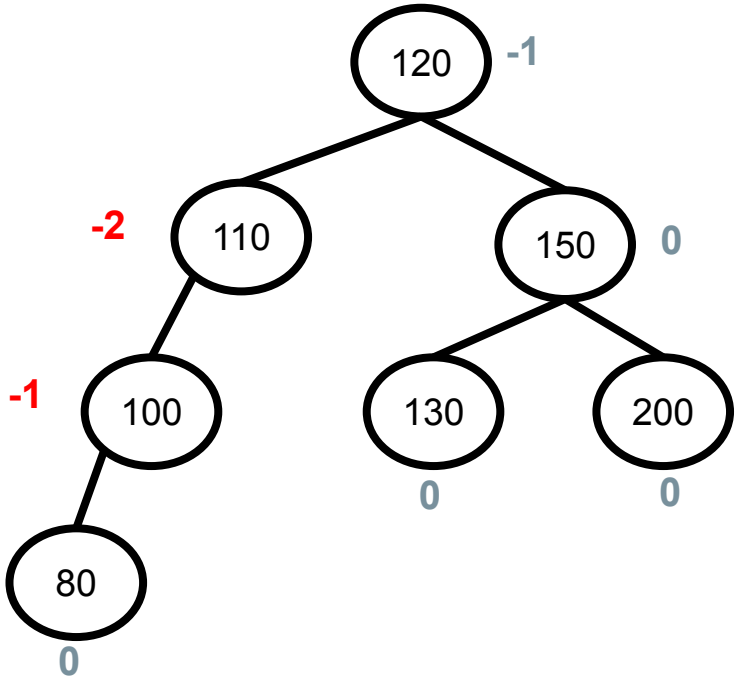


T1, T2, T3 e T4 são subárvores (vazias ou não)

EXEMPLO 1: ROTAÇÃO DIREITA



EXEMPLO 1: ROTAÇÃO DIREITA



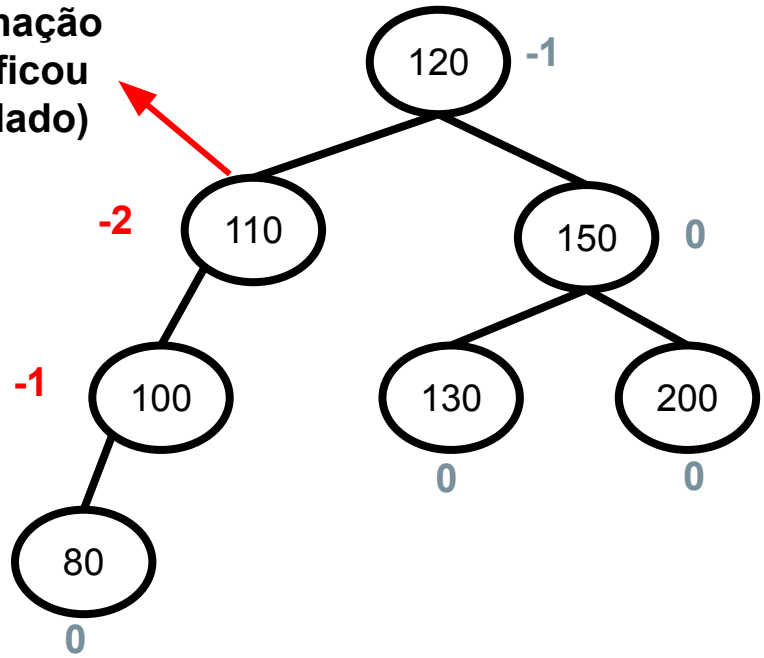
Nó com FB = -2 e filho com FB = -1 ou 0

=

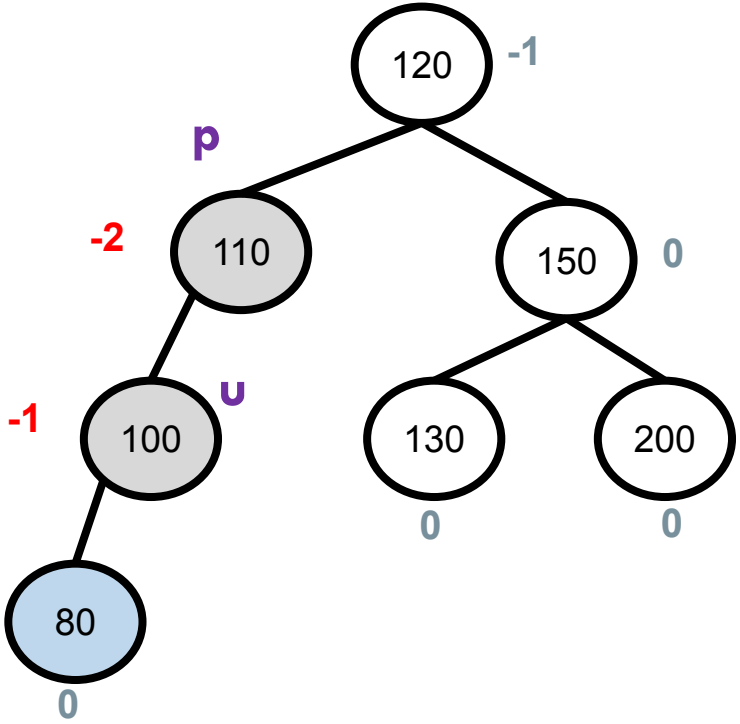
ROTAÇÃO DIREITA

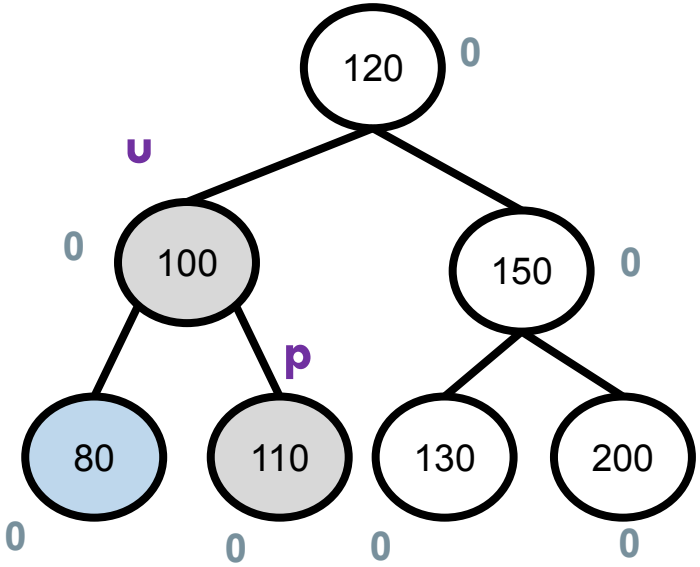
EXEMPLO 1: ROTAÇÃO DIREITA

Nó p é a raiz de transformação (nó que ficou desregulado)

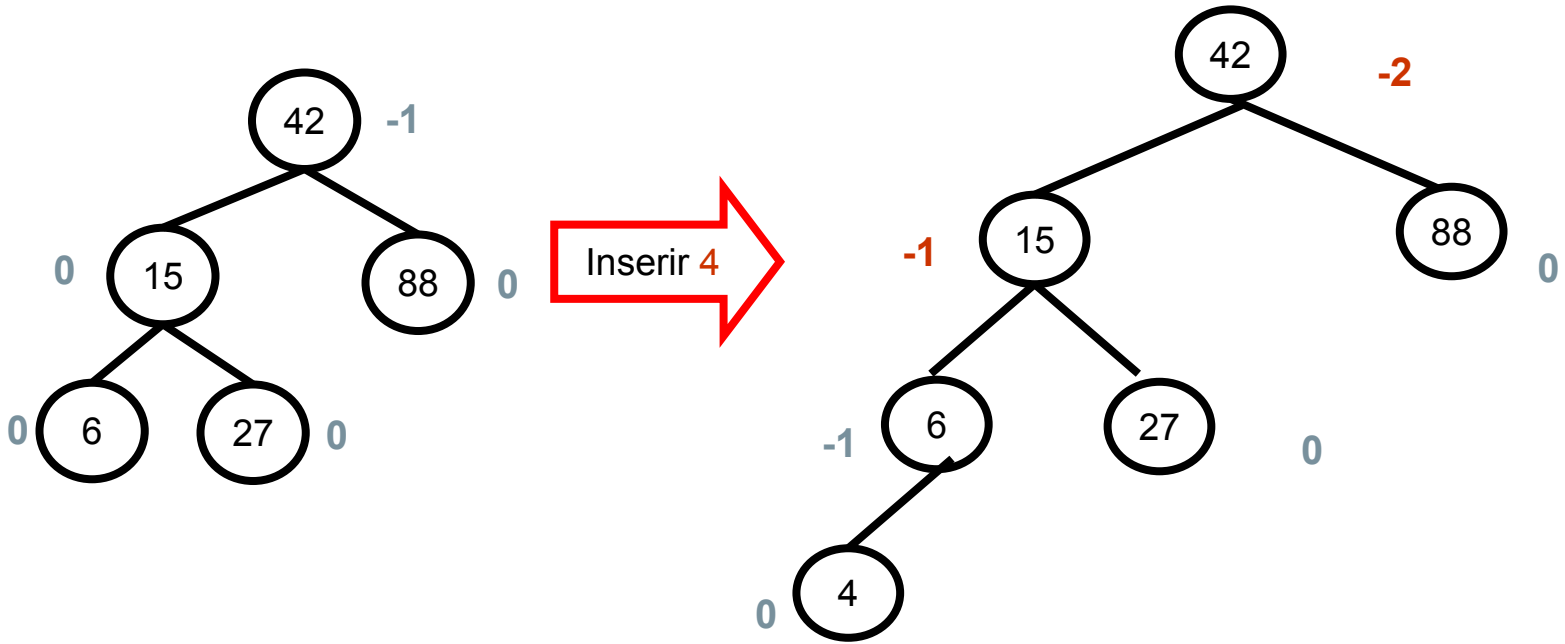


EXEMPLO 1: ROTAÇÃO DIREITA

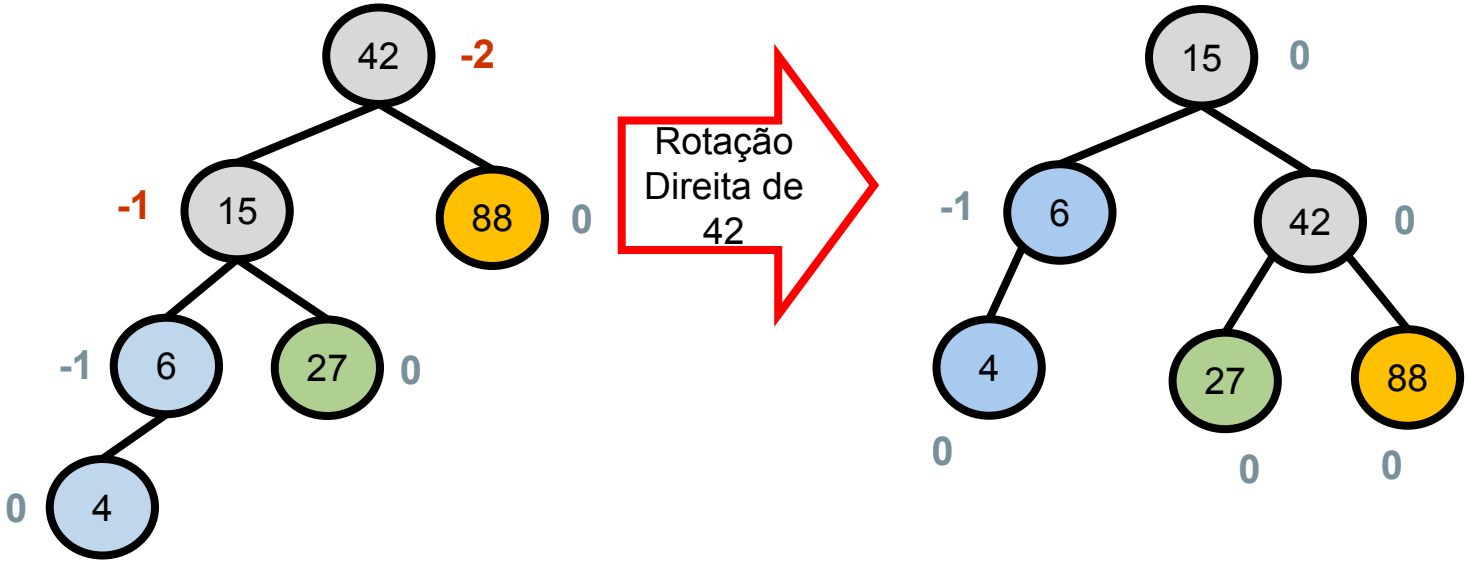




EXEMPLO 2: INSERIR 4

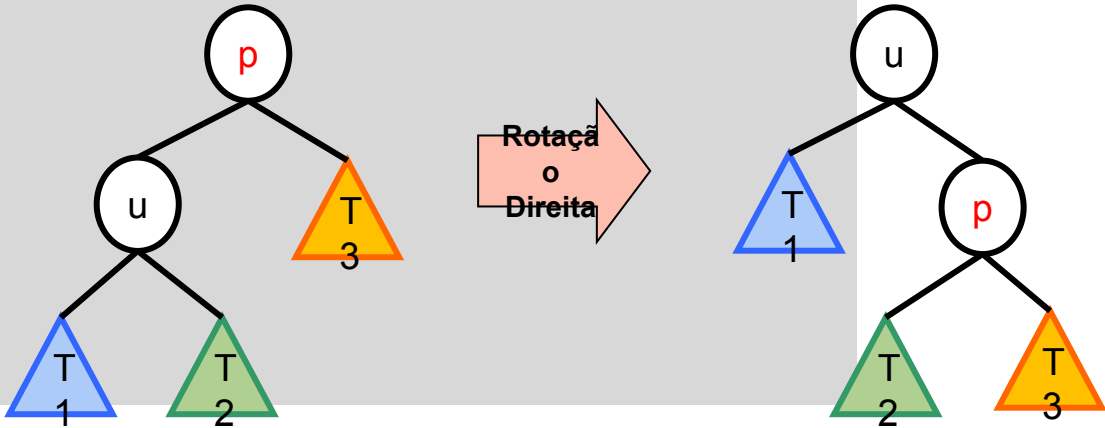


EXEMPLO 2: ROTAÇÃO DIREITA



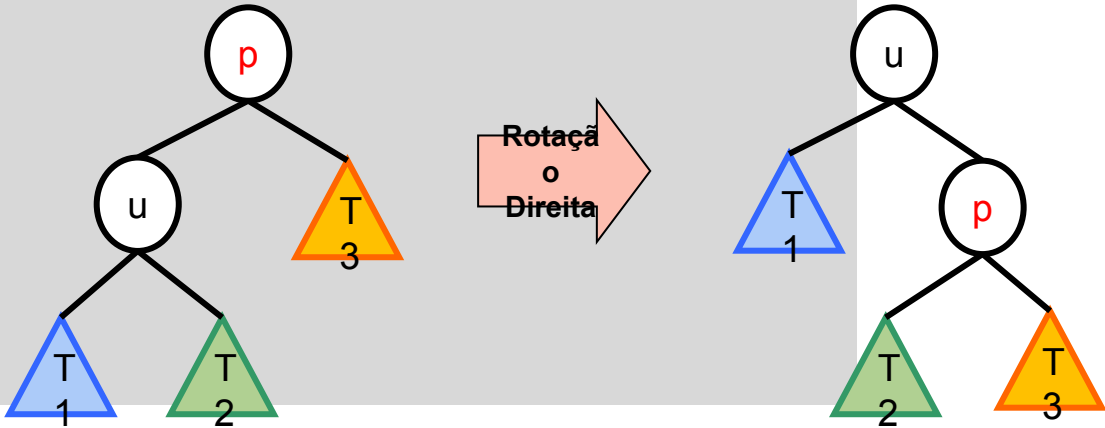
IMPLEMENTAÇÃO STRUCT DO NO

```
/* representação dos nós de Árvore ALV */
typedef struct sNoAVL {
    int chave;
    int fb;
    struct sNoAVL *esq;
    struct sNoAVL *dir;
} TNo
```



IMPLEMENTAÇÃO ROTAÇÃO DIREITA

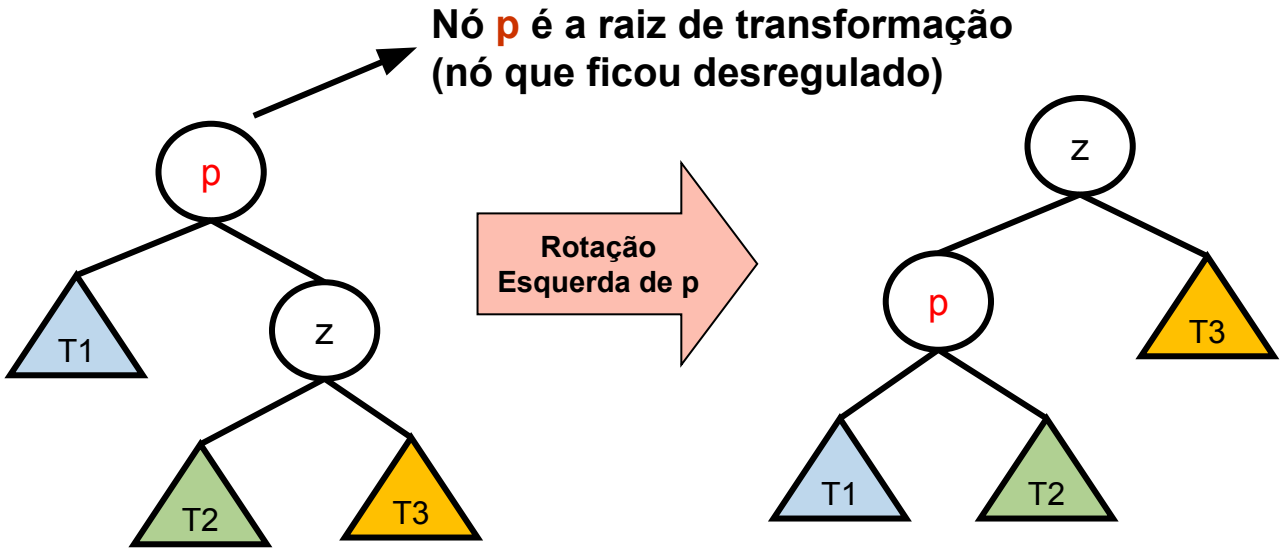
```
TNoAVL *rotacao_direita(TNoAVL *p) {
    TNoAVL *u;
    u = p->esq;
    p->esq = u->dir;
    u->dir = p;
    p->fb = 0;
    p = u;
    return p;
}
```



ROTAÇÃO SIMPLES ESQUERDA

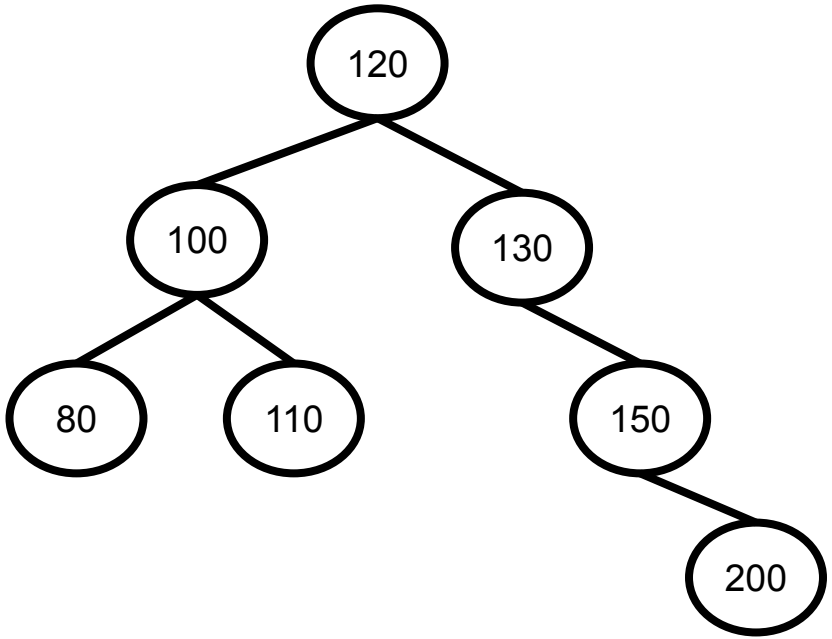
- **Nó com FB = +2 e filho com FB = +1 ou 0**:
  - rotação do nó com FB = +2 p/ esquerda

ROTAÇÃO ESQUERDA

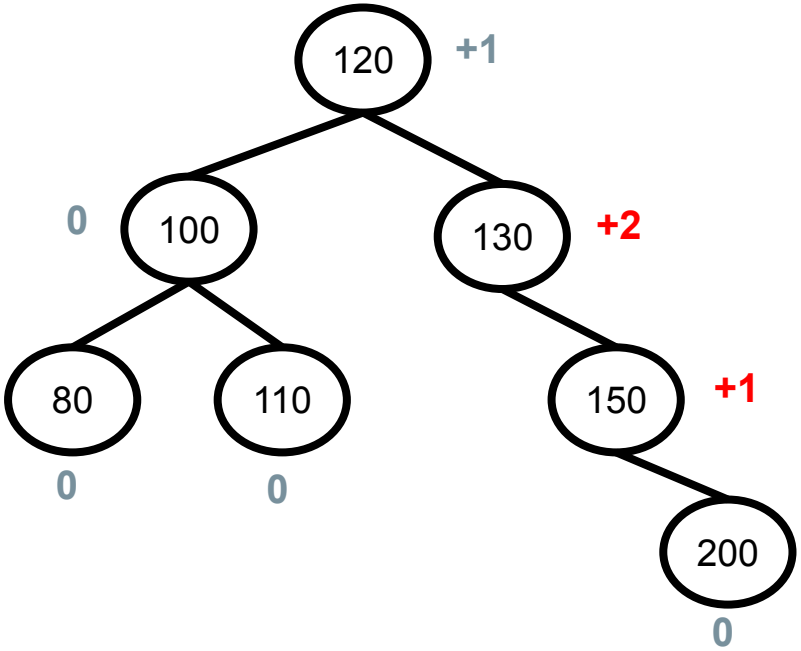


T1, T2, T3 e T4 são subárvores (vazias ou não)

EXEMPLO 1: ROTAÇÃO ESQUERDA



EXEMPLO 1: ROTAÇÃO ESQUERDA

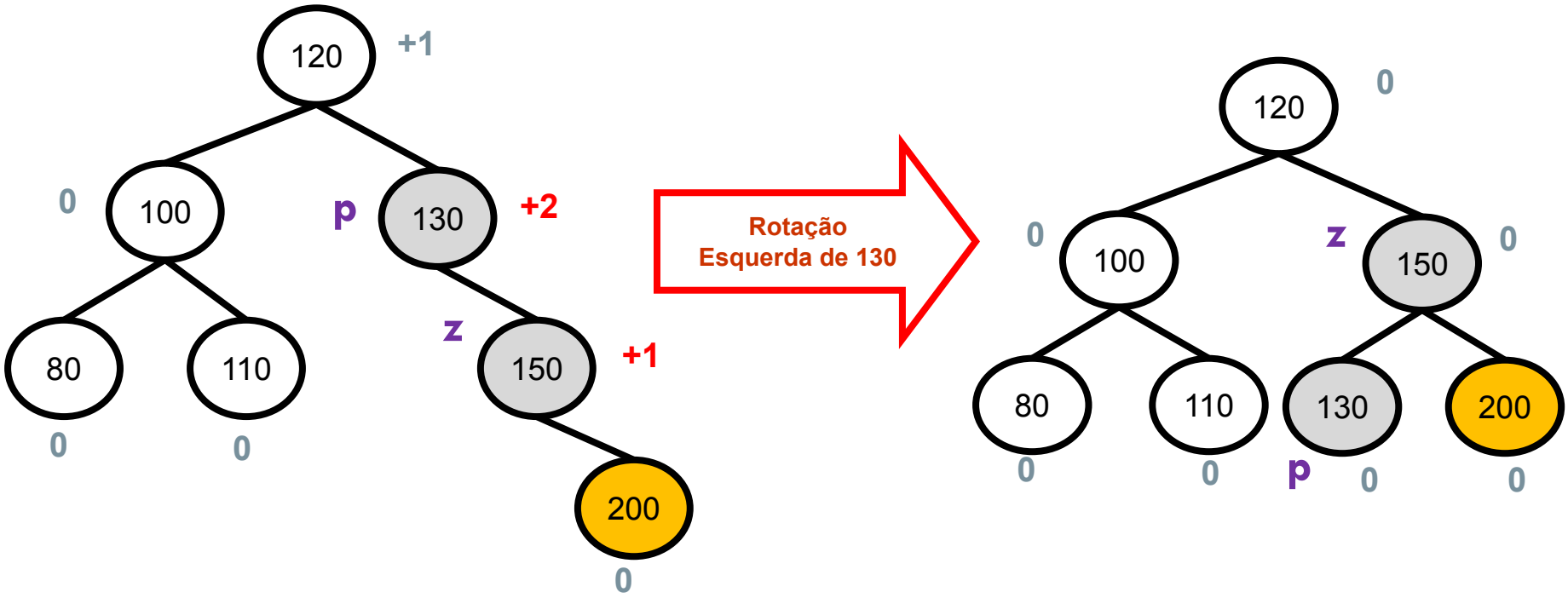


Nó com FB = +2 e filho com FB = +1 ou 0

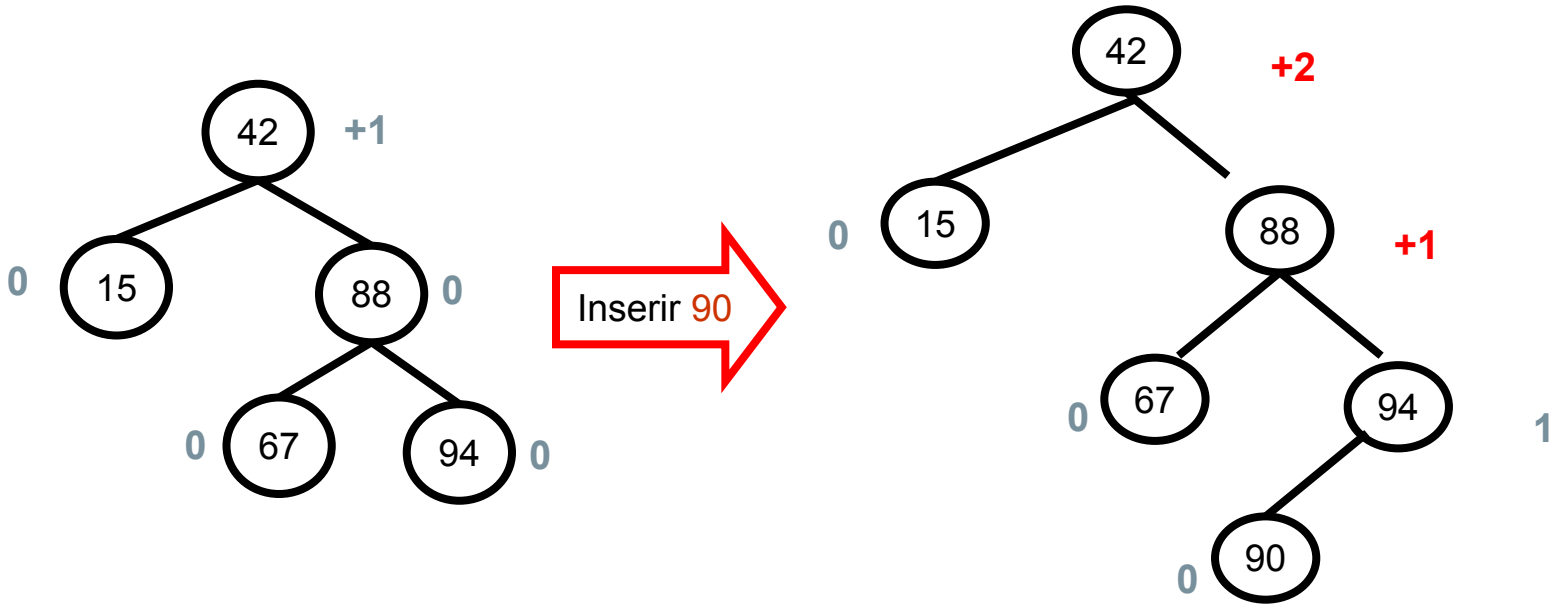
=

ROTAÇÃO ESQUERDA

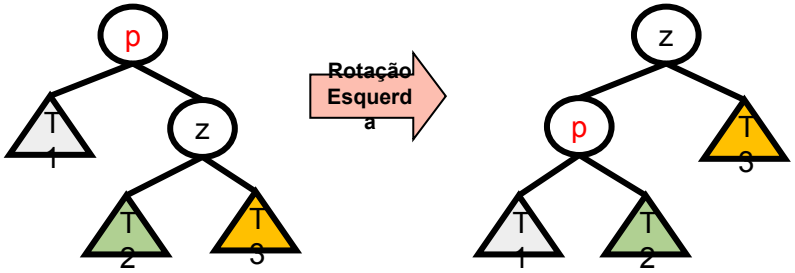
EXEMPLO 1:
ROTAÇÃO ESQUERDA

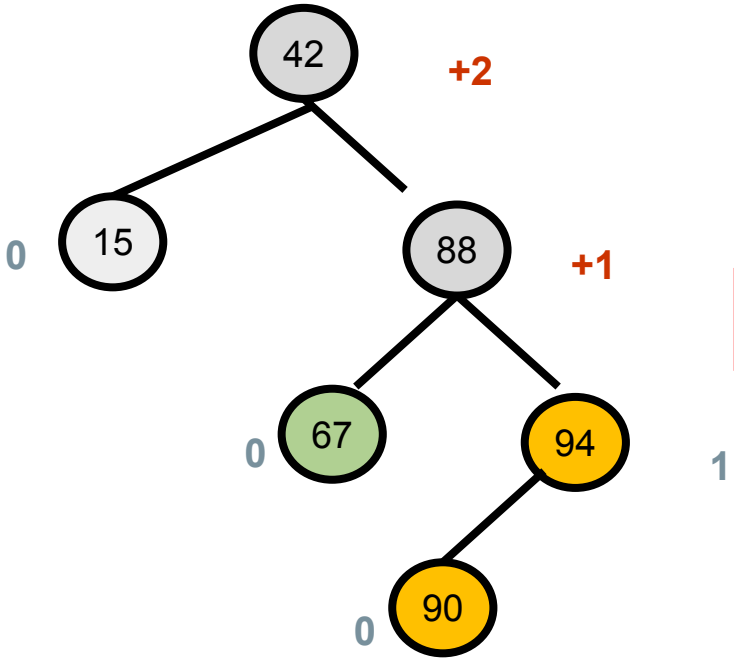


EXEMPLO 2: INSERIR 90

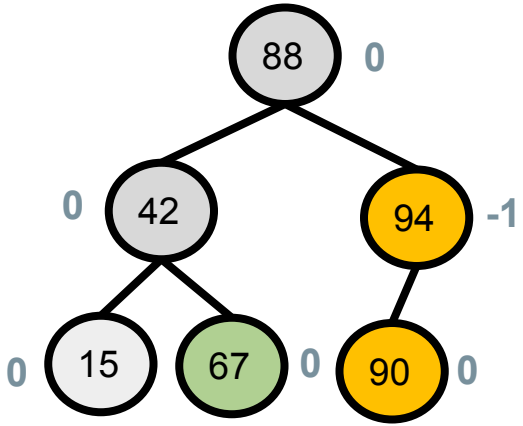


EXEMPLO 2: ROTAÇÃO ESQUERDA



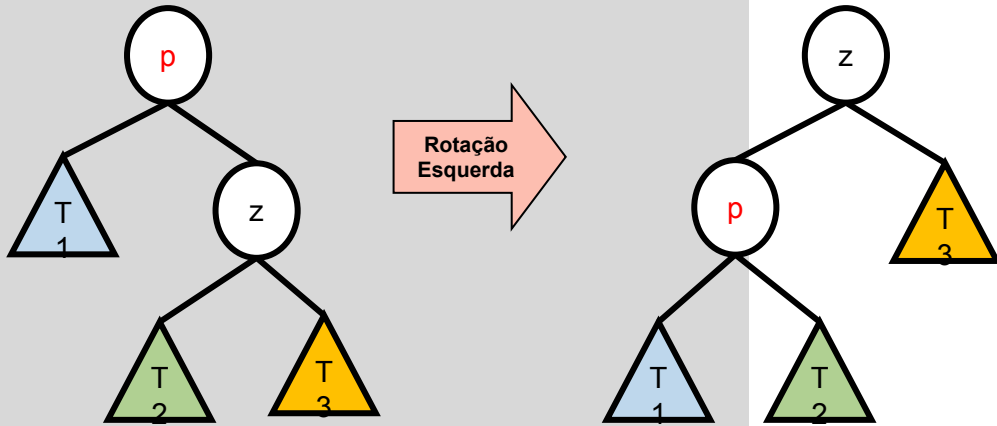


Rotação Esquerda de 42



IMPLEMENTAÇÃO ROTAÇÃO ESQUERDA

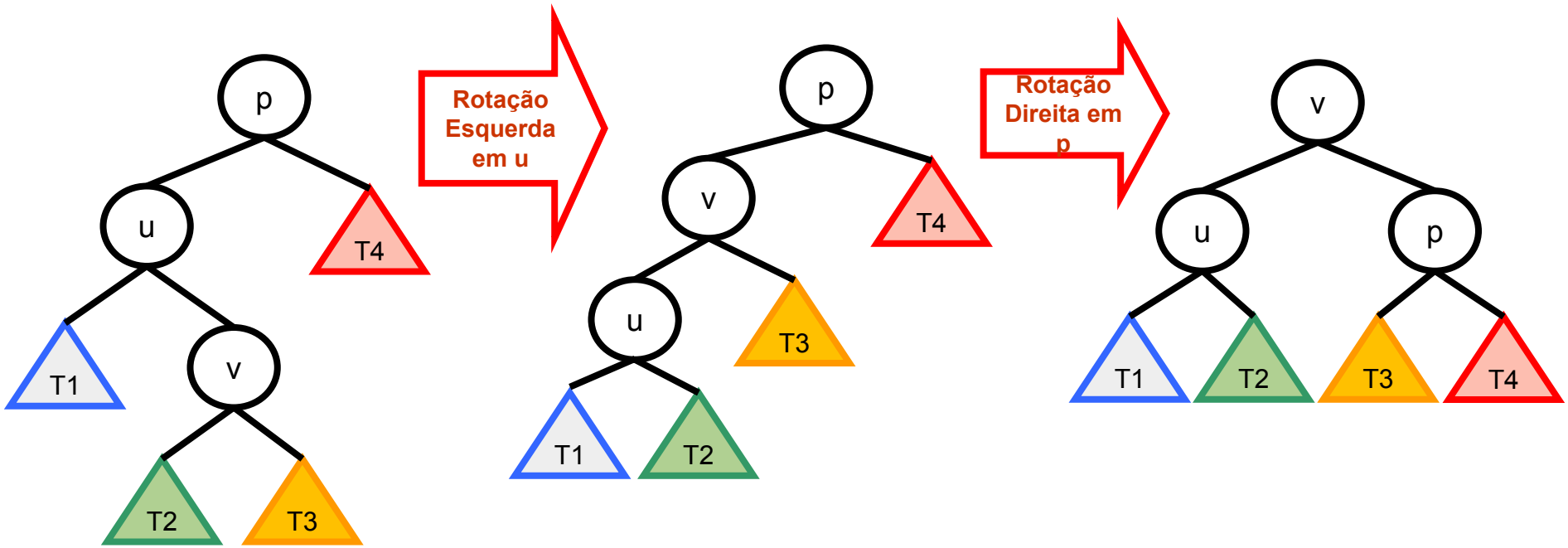
```
TNoAVL *rotacao_esquerda(TNoAVL *p) {
    TNoAVL *u;
    u = p->dir;
    p->dir = u->esq;
    u->esq = p;
    p->fb = 0;
    p = u;
    return p;
}
```



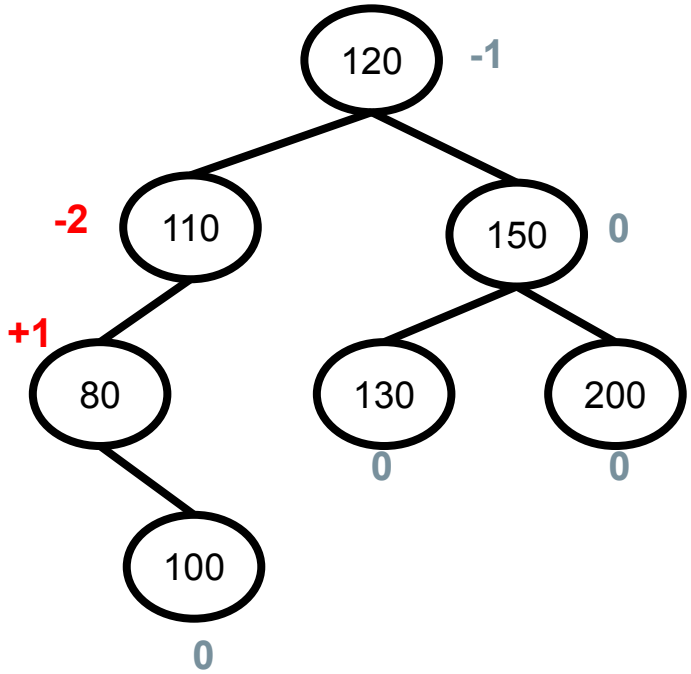
ROTAÇÃO DUPLA DIREITA (ESQUERDA-DIREITA)

- **Nó com FB = -2 e filho com FB = +1**:
  - rotação do nó com FB = +1 p/ esquerda, e
  - rotação do nó com FB = -2 p/ direita

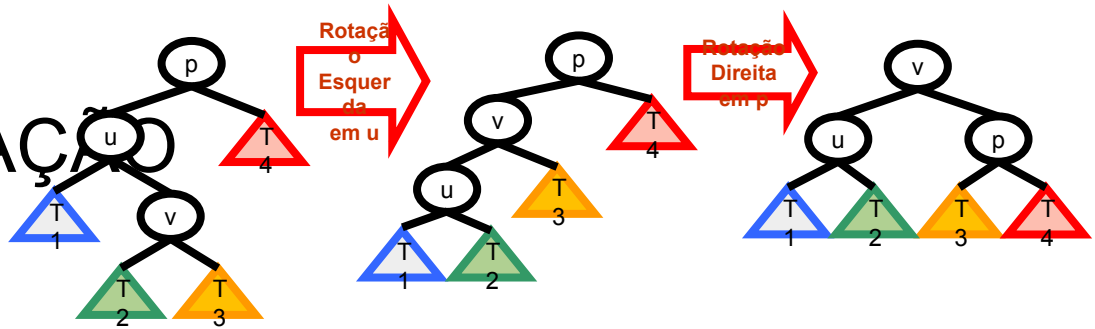
ROTAÇÃO DUPLA DIREITA (ESQUERDA-DIREITA)

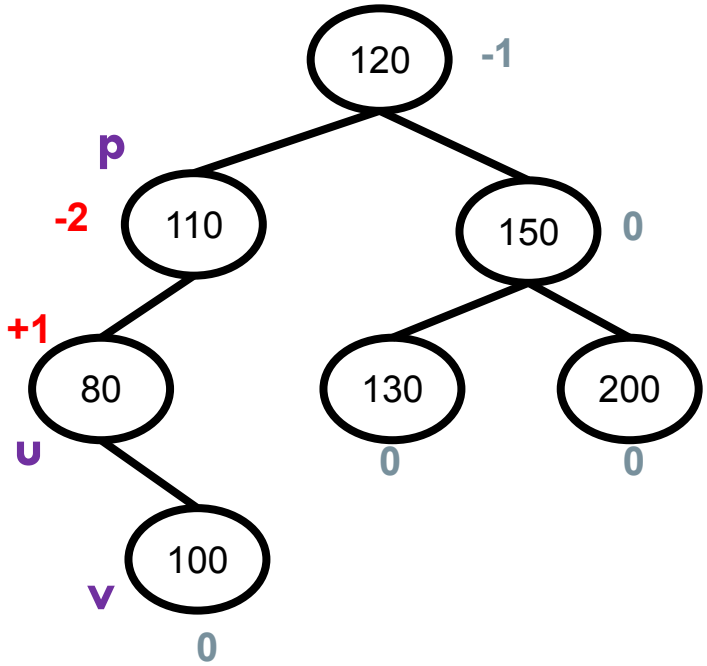


EXEMPLO 1: ROTAÇÃO DUPLA DIREITA

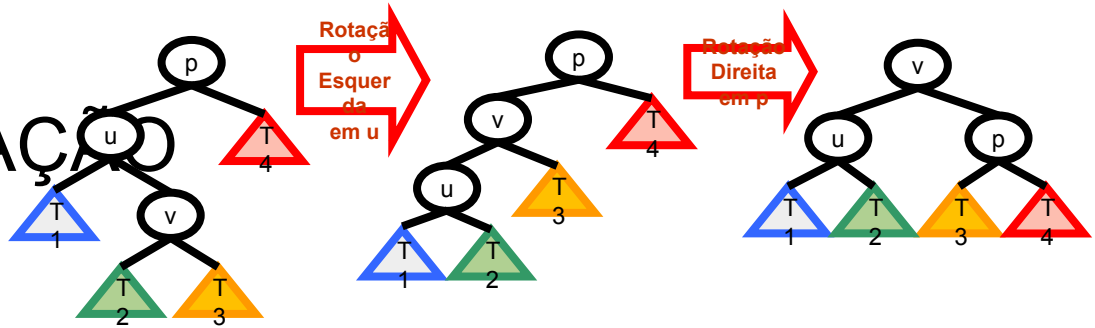


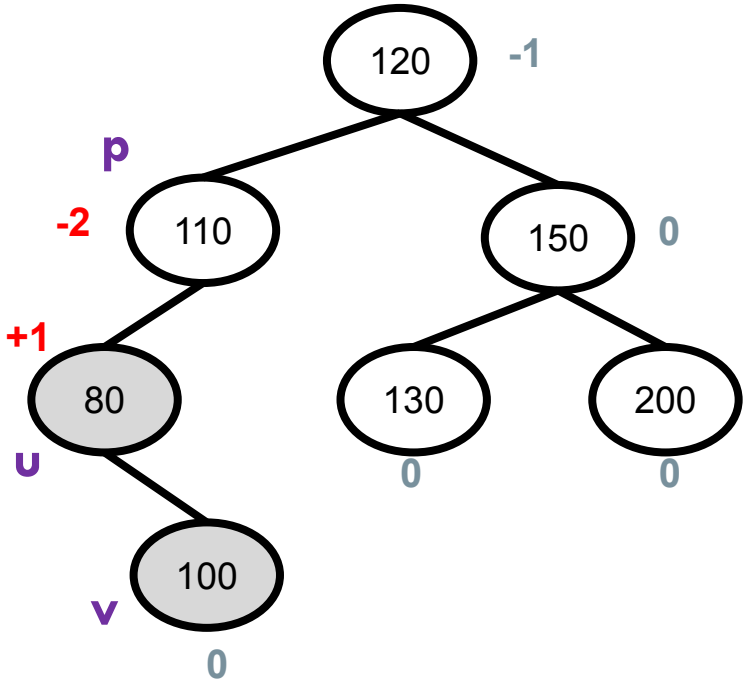
EXEMPLO 1: ROTAÇÃO DUPLA DIREITA



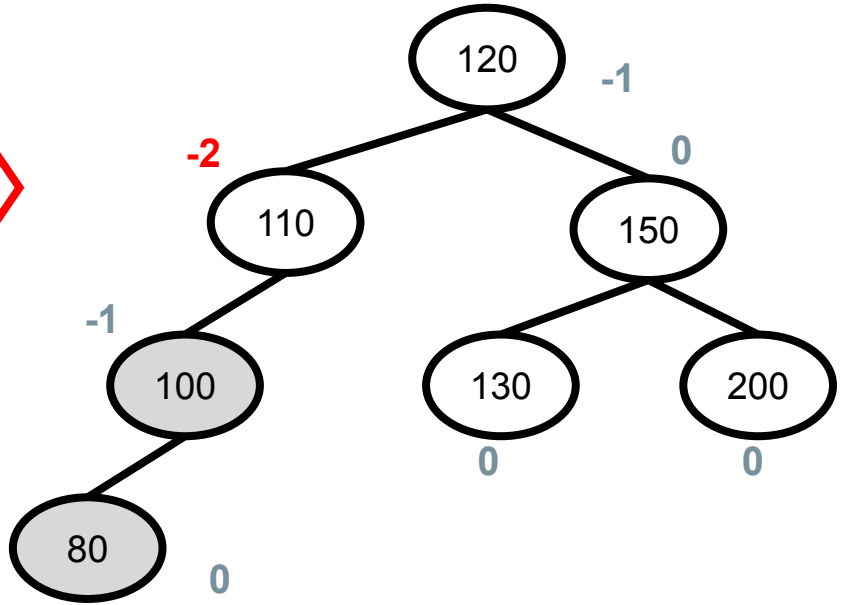


EXEMPLO 1: ROTAÇÃO DUPLA DIREITA

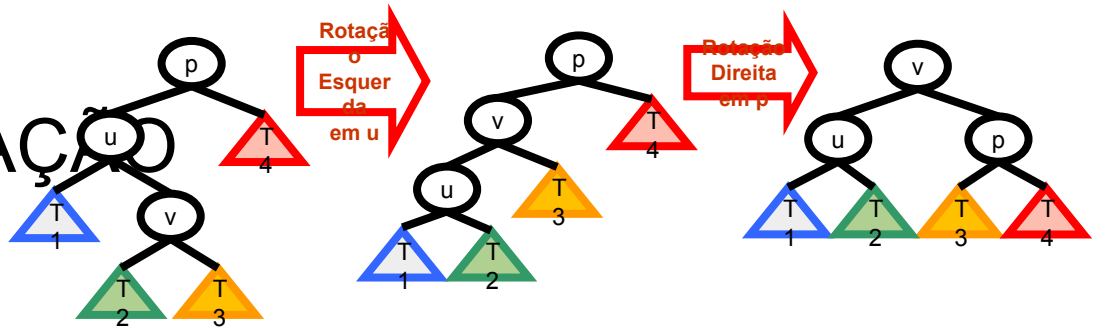


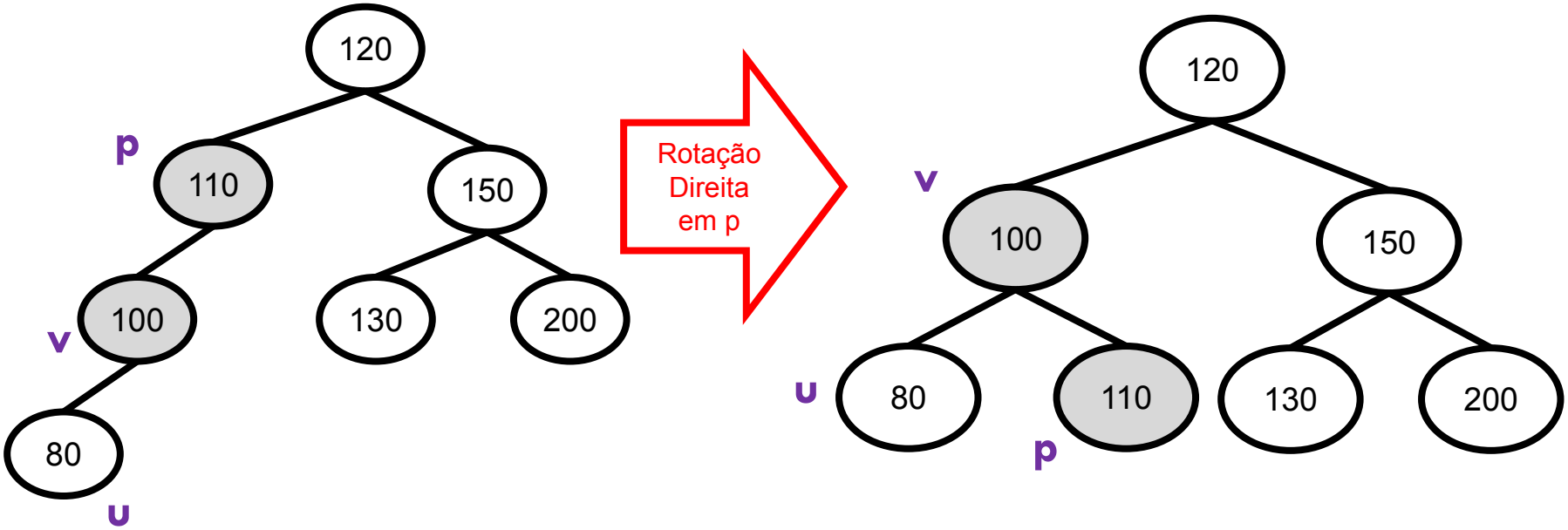


Rotação Esquerda em u

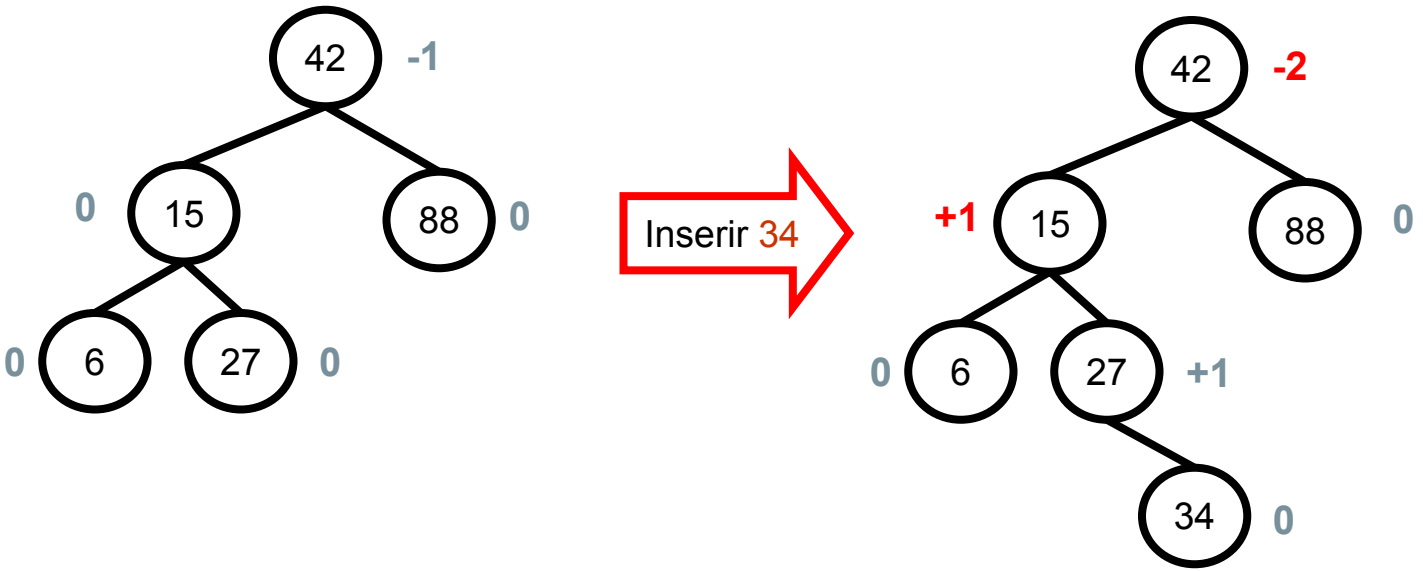


EXEMPLO 1: ROTAÇÃO DUPLA DIREITA

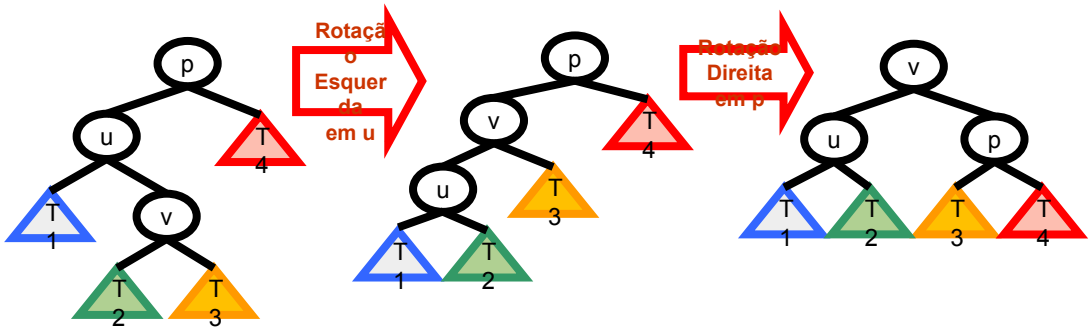


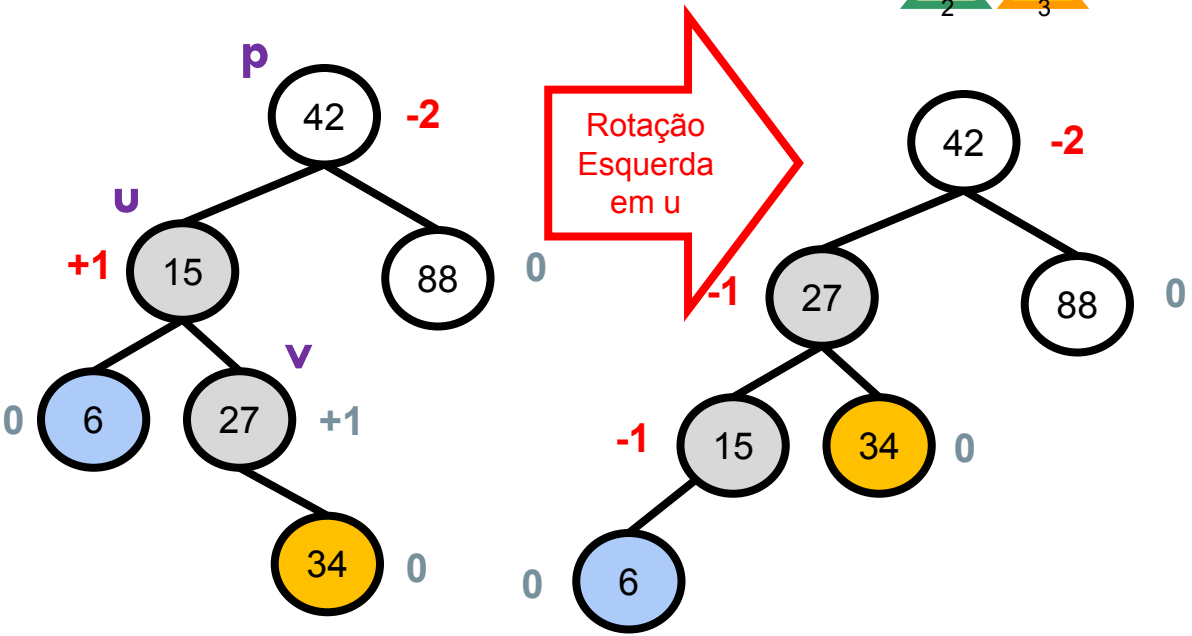


EXEMPLO 2: INSERIR 34

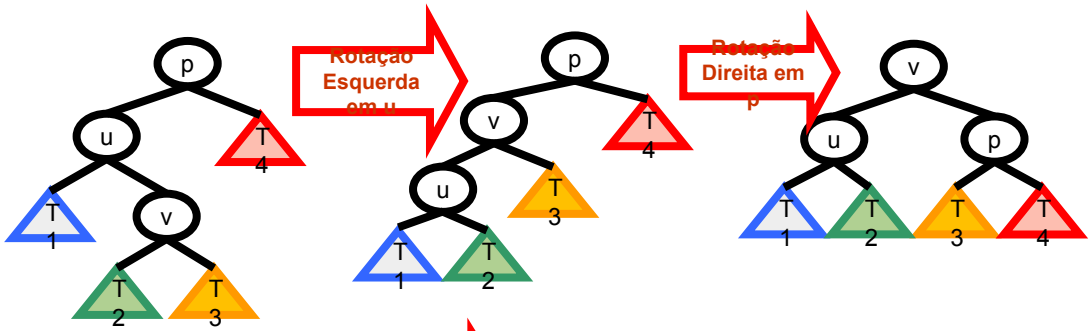


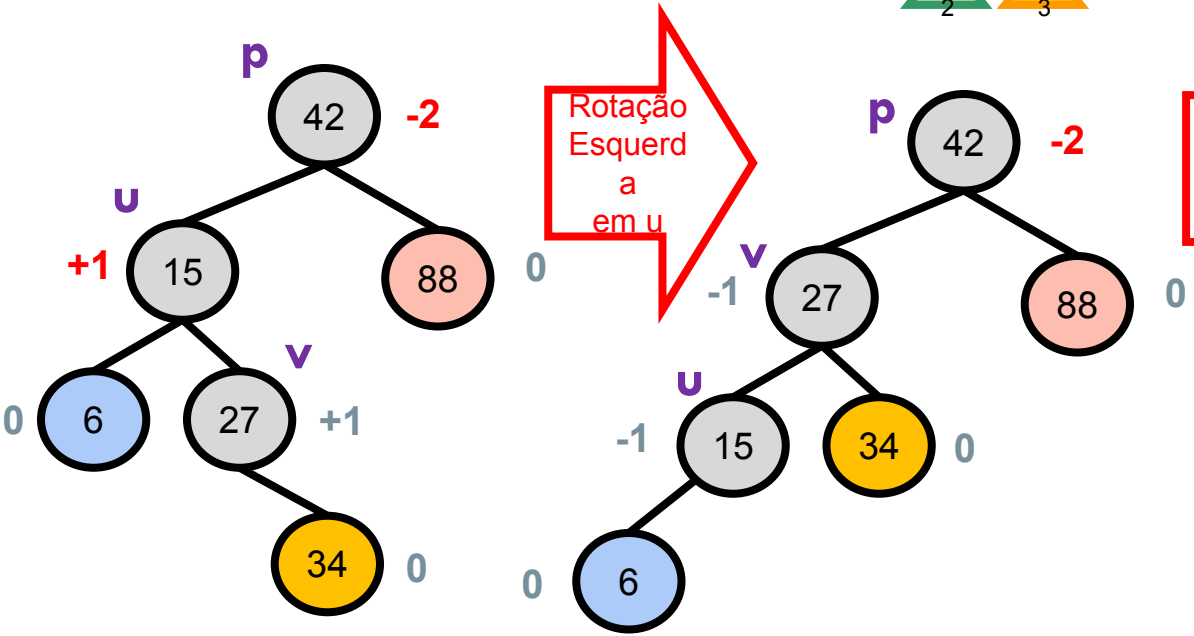
EXEMPLO 2: ROTAÇÃO DUPLA DIREITA



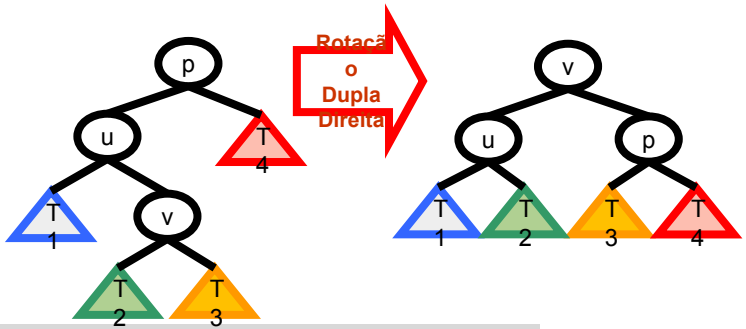


EXEMPLO 2: ROTAÇÃO DUPLA DIREITA





IMPLEMENTAÇÃO ROTAÇÃO DUPLA DIREITA

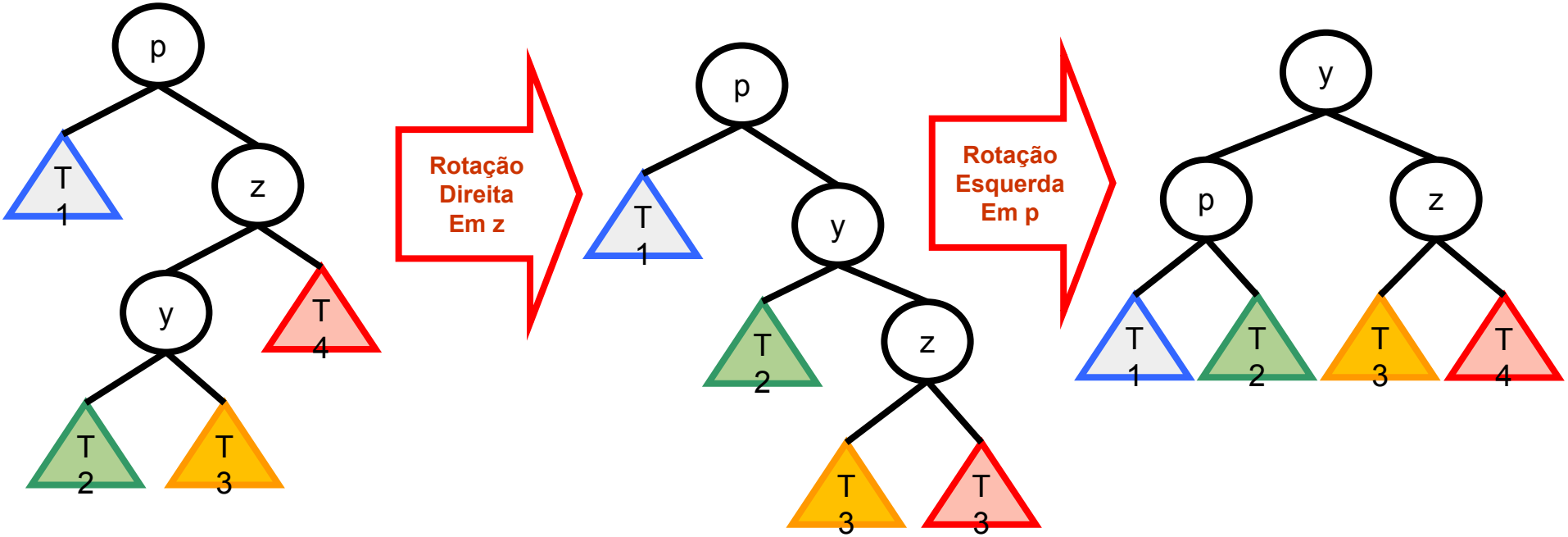



```
pNoA* rotacao_dupla_direita (pNoA* p){
    p->esq = rotacao_esquerda(p->esq);
    p = rotacao_direita(p);
    return p;
}
```

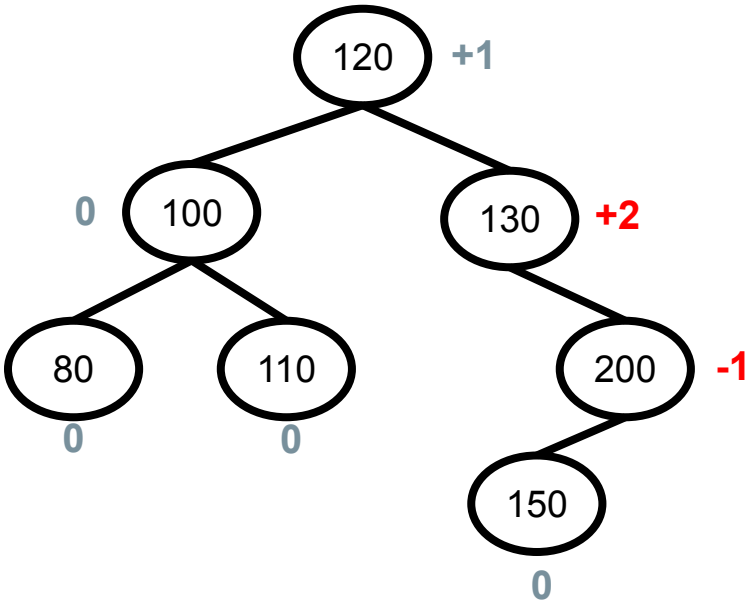
ROTAÇÃO DUPLA ESQUERDA (DIREITA-ESQUERDA)

- **Nó com FB = +2 e filho com FB = -1**:
  - rotação do nó com FB = -1 p/ direita, e
  - rotação do nó com FB = +2 p/ esquerda

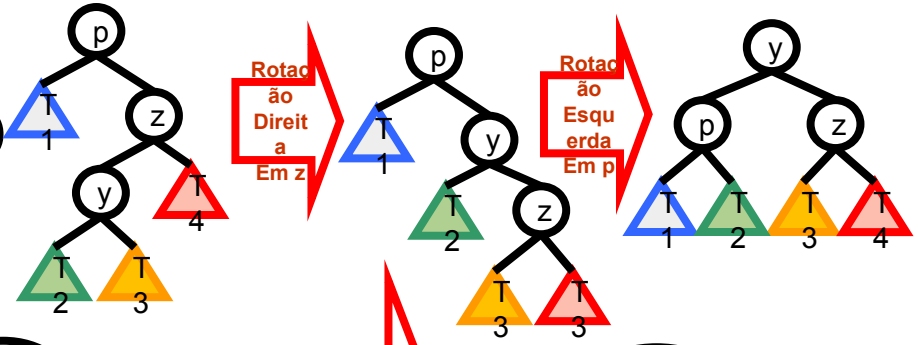
ROTAÇÃO DUPLA ESQUERDA (DIREITA-ESQUERDA)

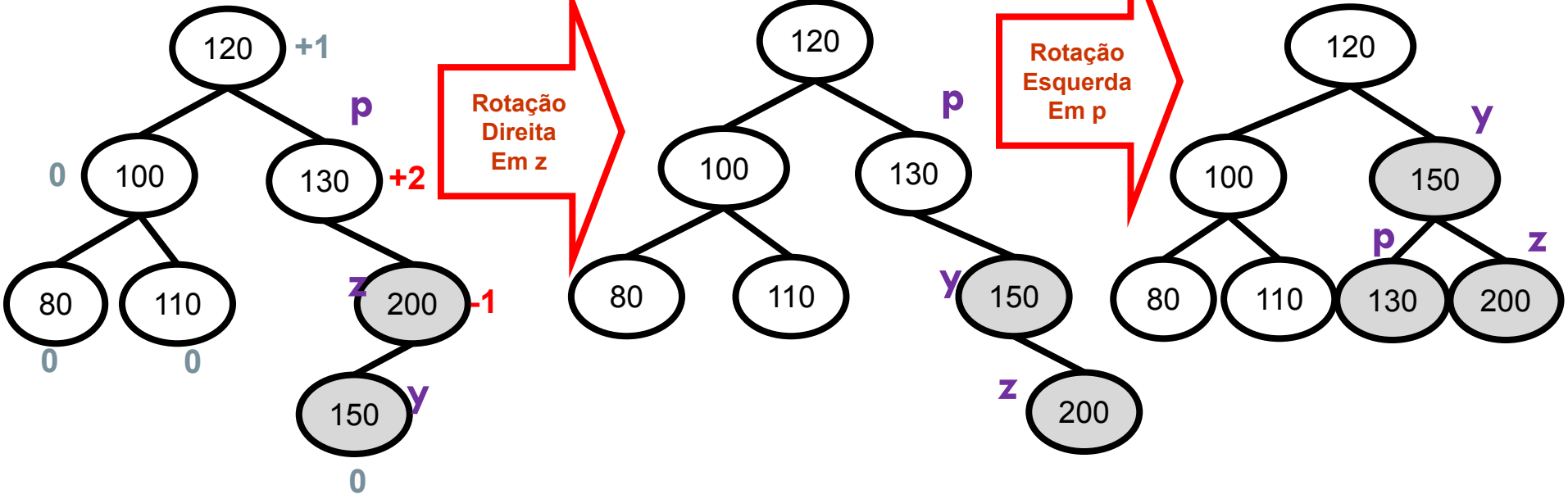


EXEMPLO 1: ROTAÇÃO DUPLA ESQUERDA

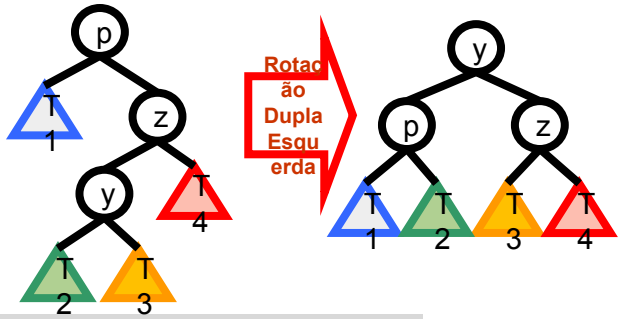


EXEMPLO 1: ROTAÇÃO DUPLA ESQUERDA





IMPLEMENTAÇÃO ROTAÇÃO DUPLA ESQUERDA




```
pNoA rotacao_dupla_esquerda (pNoA *p){
    p->dir = rotacao_direita(p->dir);
    p = rotacao_esquerda(p);
    return p;
}
```

EXERCÍCIO

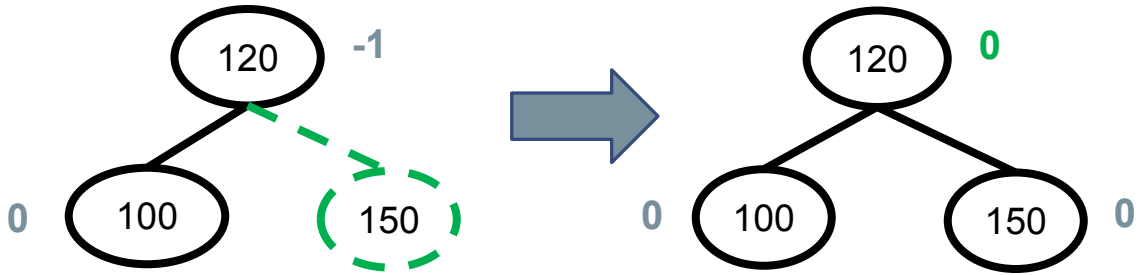
- Inserir nós com as seguintes chaves em uma árvore AVL, refazendo a árvore quando houver rotação e anotando as rotações realizadas:
  - 50, 40, 30, 45, 47, 55, 56, 1, 2, 3, 49

INSERÇÃO DE NÓS EM ÁRVORES AVL: ALGUNS PROBLEMAS

- Como saber se a árvore está balanceada?
  - Verificando se existe um nó “desregulado”
- Como saber se um nó está desregulado?
  - Determina-se as alturas de suas sub-árvores e subtrai-se uma da outra
- Procedimento muito lento!
- Como ser mais eficiente?
  - Para cada nó v de uma árvore, armazena-se uma variável **fb** que registra essa diferença

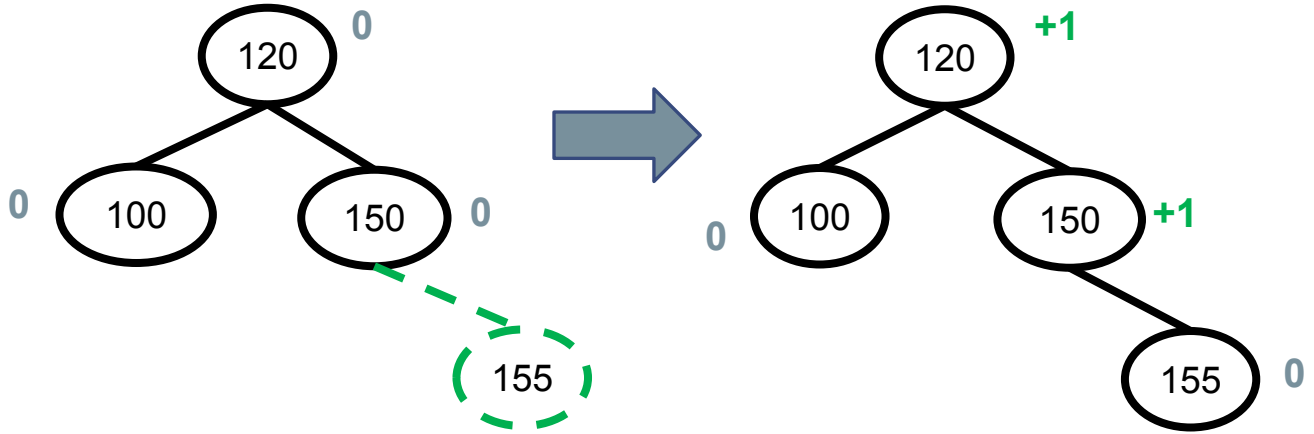
MANUTENÇÃO DE FB: INSERÇÃO À DIREITA DE UM NÓ V

- Se, antes da inclusão, **fb(v) = -1**, então fb(v) se tornará **0**
  - Altura da árvore não foi alterada
  - Por consequência, altura dos outros nós no caminho até a raiz não se altera também



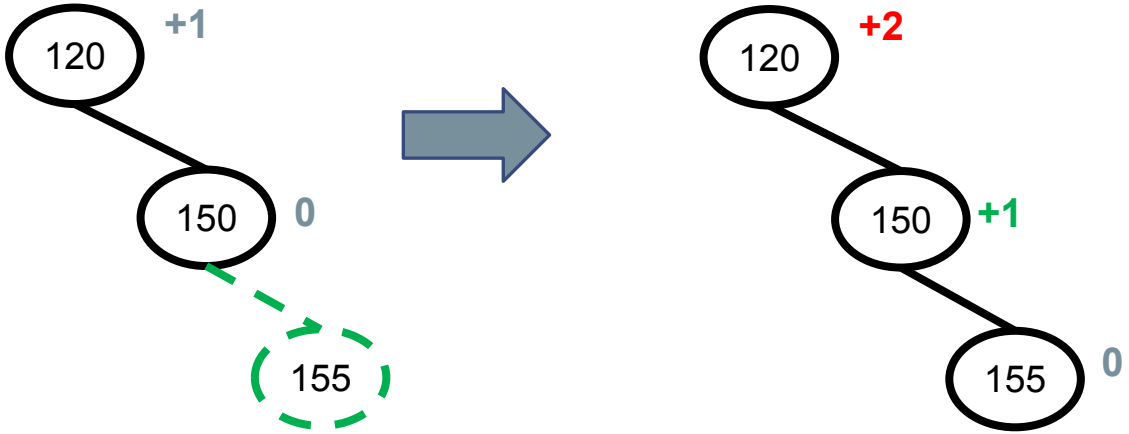
MANUTENÇÃO DE FB: INSERÇÃO À DIREITA DE UM NÓ V

- Se, antes da inclusão, **fb(v) = 0**, então fb(v) se tornará **+1**
  - Altura da árvore foi modificada
  - Por consequência, altura dos outros nós no caminho até a raiz pode ter sido alterada também.
  - Repetir o processo (recursivamente), com v substituído por seu pai.



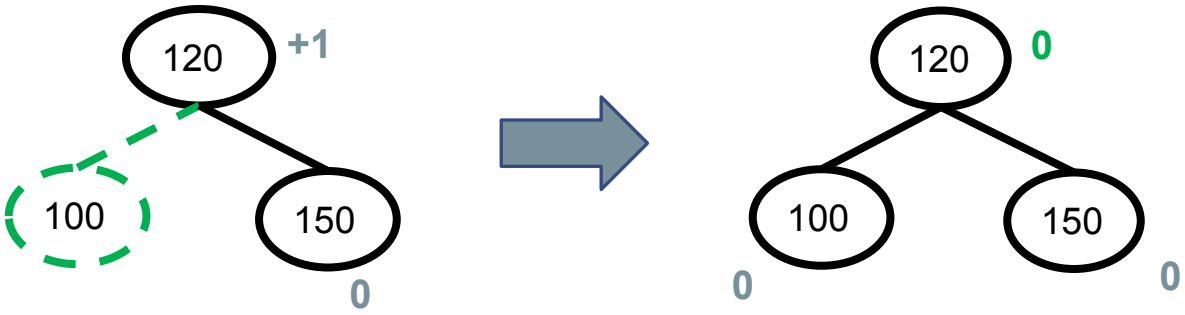
MANUTENÇÃO DE FB: INSERÇÃO À DIREITA DE UM NÓ V

- Se, antes da inserção, **fb(v) = +1**, então fb(v) se tornará **+2**
  - Esse caso só ocorre por propagação de inserção em nó com fb = 0
  - Altura da árvore foi modificada e o nó está desregulado
  - Rotação correta deve ser empregada.
  - Como a árvore será redesenhada, não é necessário verificar os outros nós.



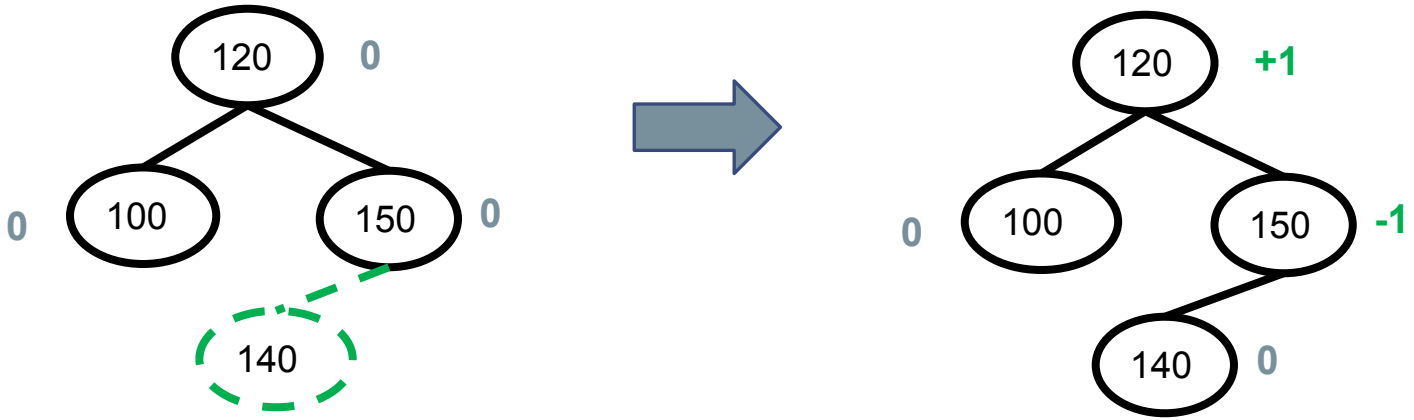
MANUTENÇÃO DE FB: INSERÇÃO À ESQUERDA DE UM NÓ V

- Se, antes da inserção, **fb(v) = +1**, então fb(v) se tornará **0**
  - Altura da árvore não foi alterada
  - Por consequência, altura dos outros nós no caminho até a raiz, não se altera também



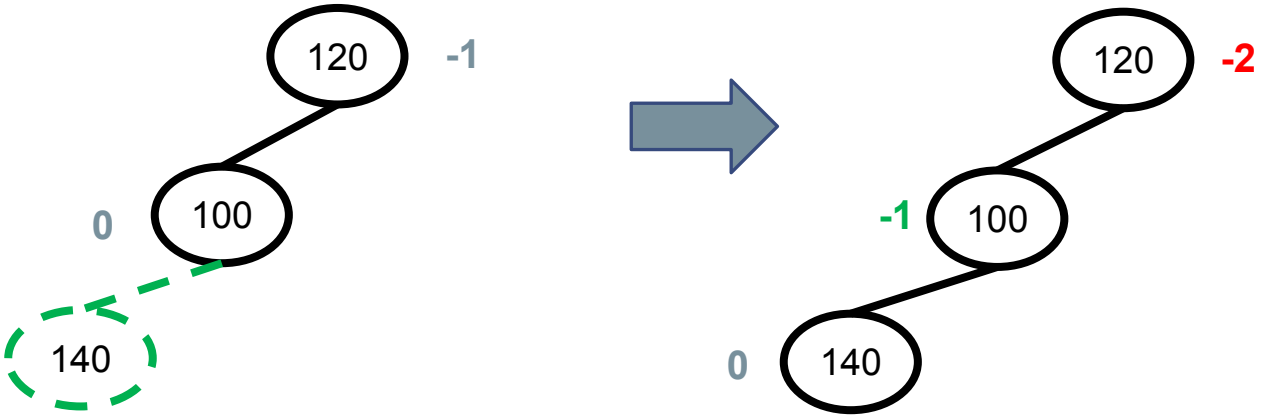
MANUTENÇÃO DE FB: INSERÇÃO À ESQUERDA DE UM NÓ V

- Se, antes da inserção, **fb(v) = 0**, então fb(v) se tornará **-1**
  - Altura da árvore foi modificada
  - Por consequência, altura dos outros nós no caminho até a raiz, pode ter sido alterada também
  - Repetir o processo (recursivamente), com v substituído por seu pai



MANUTENÇÃO DE FB: INSERÇÃO À ESQUERDA DE UM NÓ V

- Se, antes da inserção, **fb(v) = -1**, então fb(v) se tornará **-2**
  - Esse caso só ocorre por propagação de inserção em nó com fb = 0
  - Altura da árvore foi modificada e o nó está desregulado
  - Rotação correta deve ser empregada
  - Como a árvore será redesenhada, não é necessário verificar os outros nós

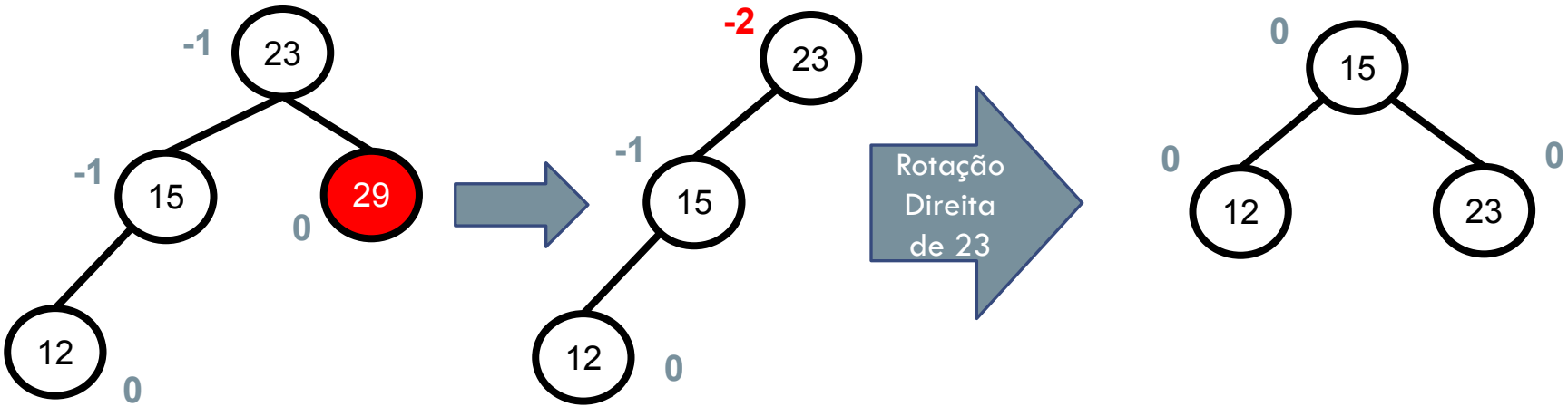


REMOÇÃO DE NÓS EM ÁRVORES AVL

- Caso parecido com as inclusões
- Realizar a remoção, recalcular FB, fazer rotações que forem necessárias

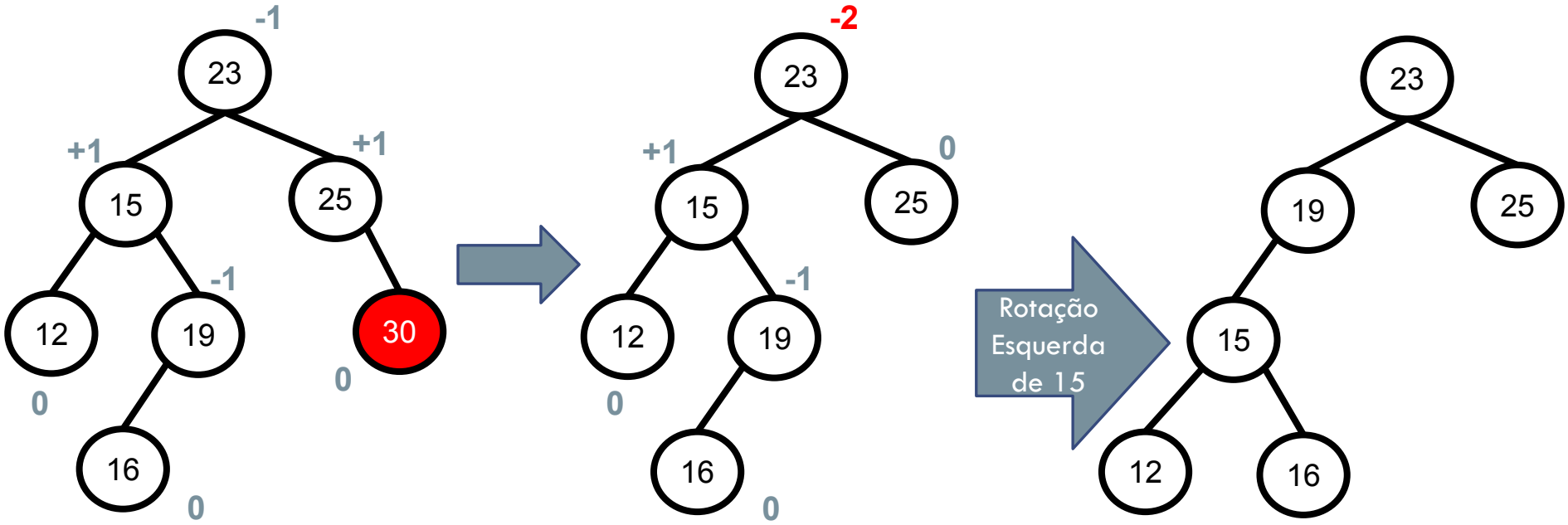
EXEMPLO 1: REMOÇÃO DE 29

Nó com FB = -2 e filho com FB = -1 ou 0:
rotação do nó com FB = -2 p/ direita

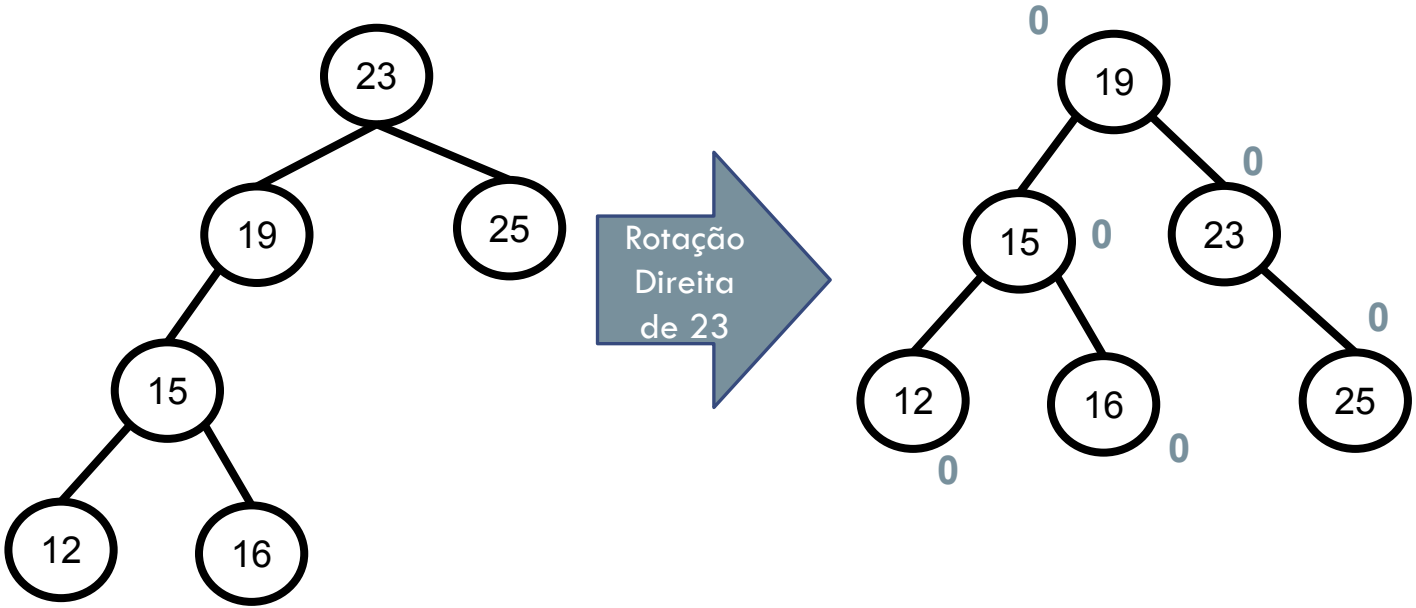


EXEMPLO 2: REMOÇÃO DE 30

Nó com FB = -2 e filho com FB = +1:
rotação do nó com FB = +1 p/ esquerda, e
rotação do nó com FB = -2 p/ direita



EXEMPLO 2:
REMOÇÃO DE 30 (CONT.)



REMOÇÃO DE NÓ INTERMEDIÁRIO

- Mesmo raciocínio
- Lembrete: se nó excluído tem 2 filhos, substituir pelo nó de maior chave da subárvore esquerda, seguindo o algoritmo de remoção em ABB

MANUTENÇÃO DE FB

- Para realizar a manutenção do fator de balanceamento dos nós durante a exclusão, usar o raciocínio da inserção no lado contrário:
  - Exclusão à direita: usar o raciocínio de inclusão à esquerda
  - Exclusão à esquerda: usar o raciocínio de inclusão à direita

EXERCÍCIO

Remover os nós de chave 400,
140, 120, 130, 150, 200, 250, 350