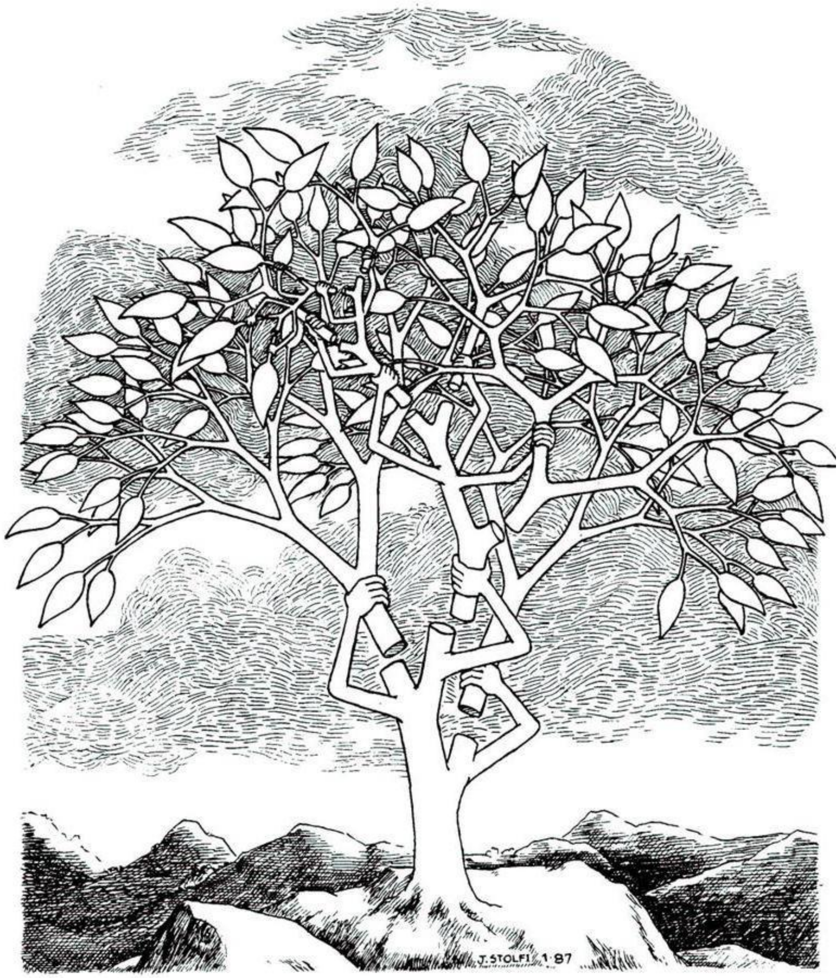


Árvores AVL

Árvores Balanceadas

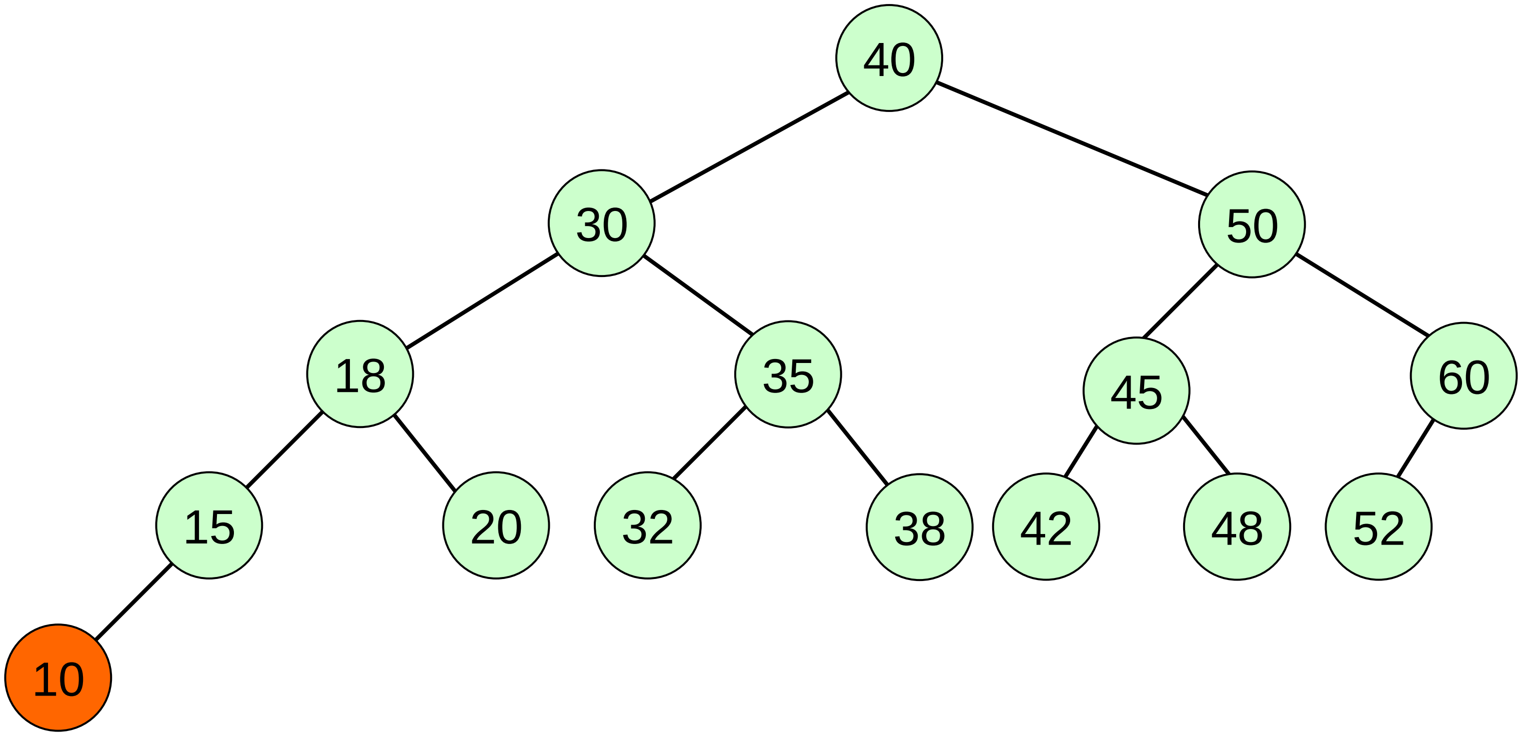
- ✓ **Aspecto importante:** **Custo de Acesso**

- ✓ **Aplicações Estáticas:** **Árvore Binária de Busca Ótima**
  Árvore de Partilha Ótima

- ✓ **Idéia:** **Manter o custo de acesso na mesma ordem de grandeza de uma árvore binária de busca ótima – $O(\log n)$**

- ✓ **Característica:** **A estrutura deve ser alterada periodicamente, balanceada.**

Árvores Balanceadas



✓ **Para a árvore voltar a ser completa são necessárias $\Omega(n)$ operações**

Árvores Balanceadas

Uma árvore binária de busca é dita balanceada se ela possui altura
$h = O(\log n)$

***n* – número de nós na árvore**

Árvores Balanceadas

✓ Árvores AVL

✓ Árvores Graduadas e Rubro-negras

✓ Árvores B

Árvores AVL

Adel'son-**V**el'skii e **L**andis

G.M. Adel'son-Vel'skii, E. M. Landis (1962). An algorithm for the organization of information, *Soviet Mathematics Doklady* 3, 1259-1263.

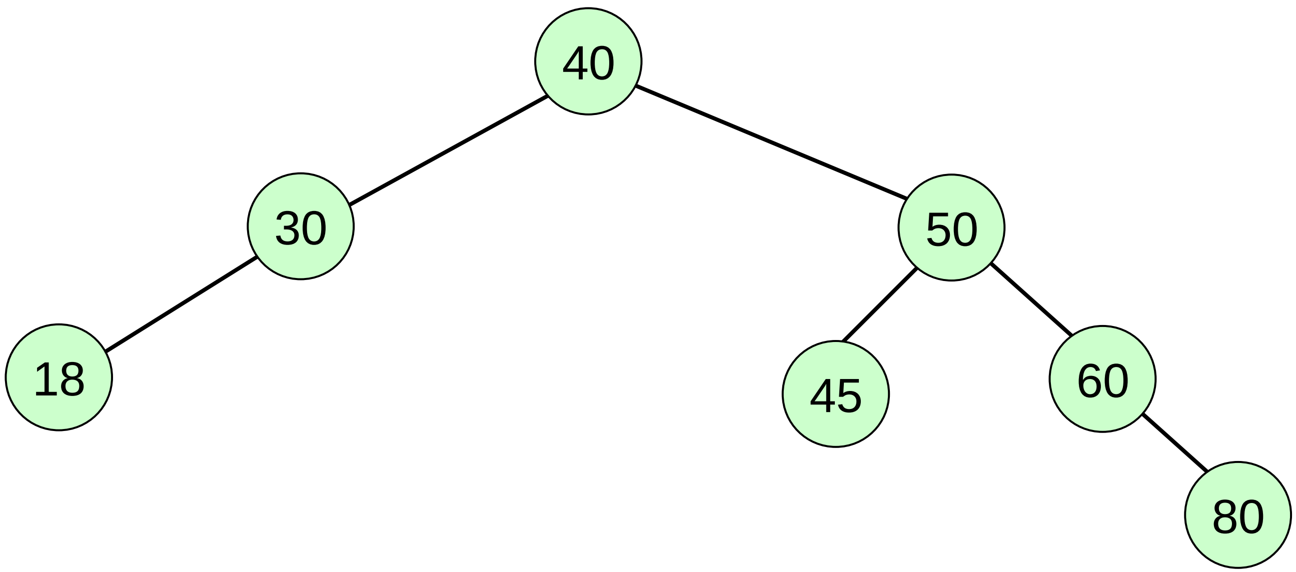
Uma árvore binária de busca onde para cada nó v, tem-se

$$|h_E(v) - h_D(v)| \leq 1$$

$h_E(v)$ – altura da subárvore esquerda de v
$h_D(v)$ – altura da subárvore direita de v

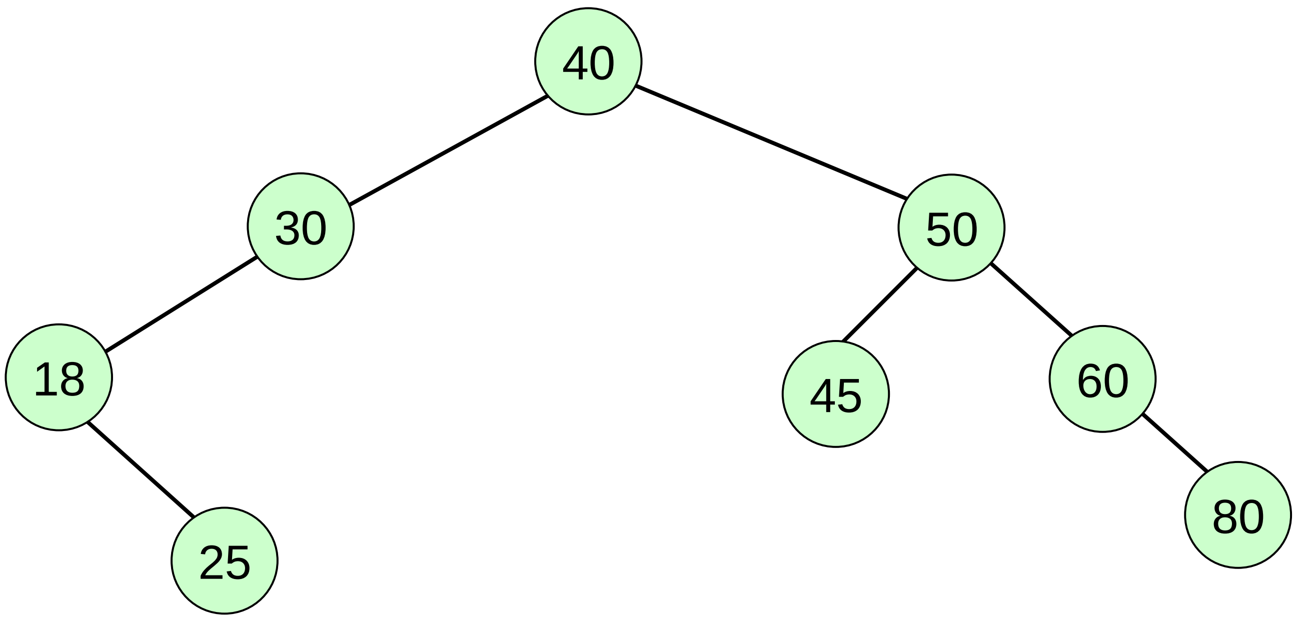
Árvores AVL

Exemplo:



Árvores AVL

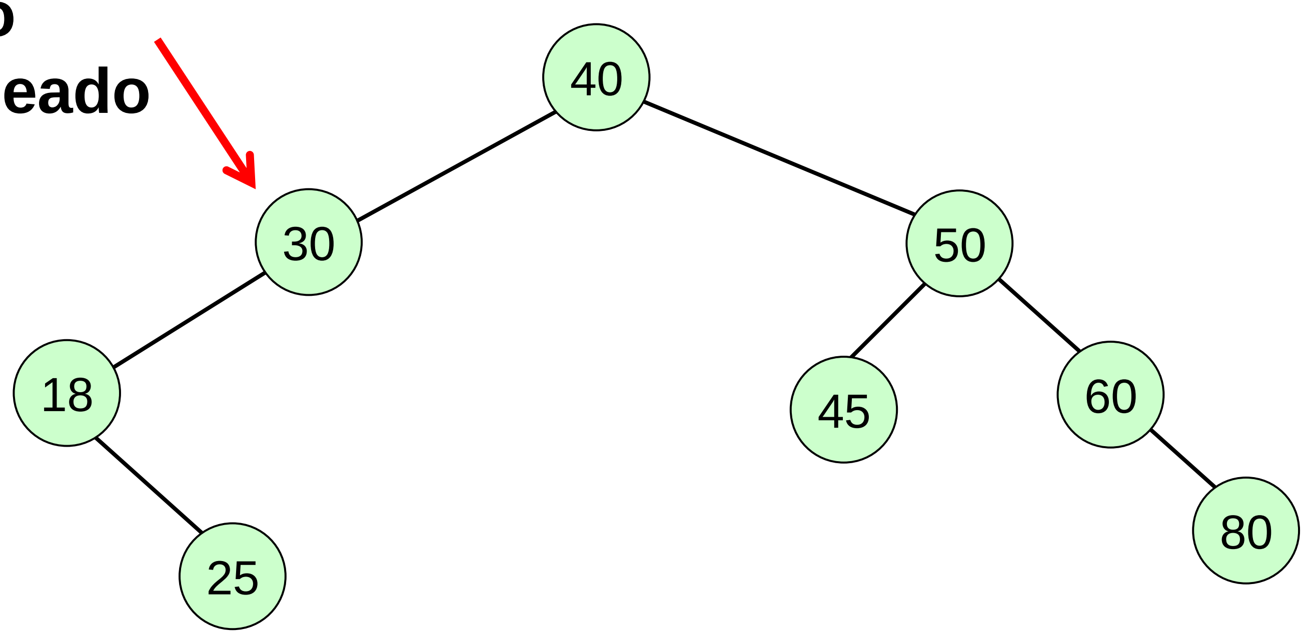
Esta árvore é AVL?



Árvores AVL

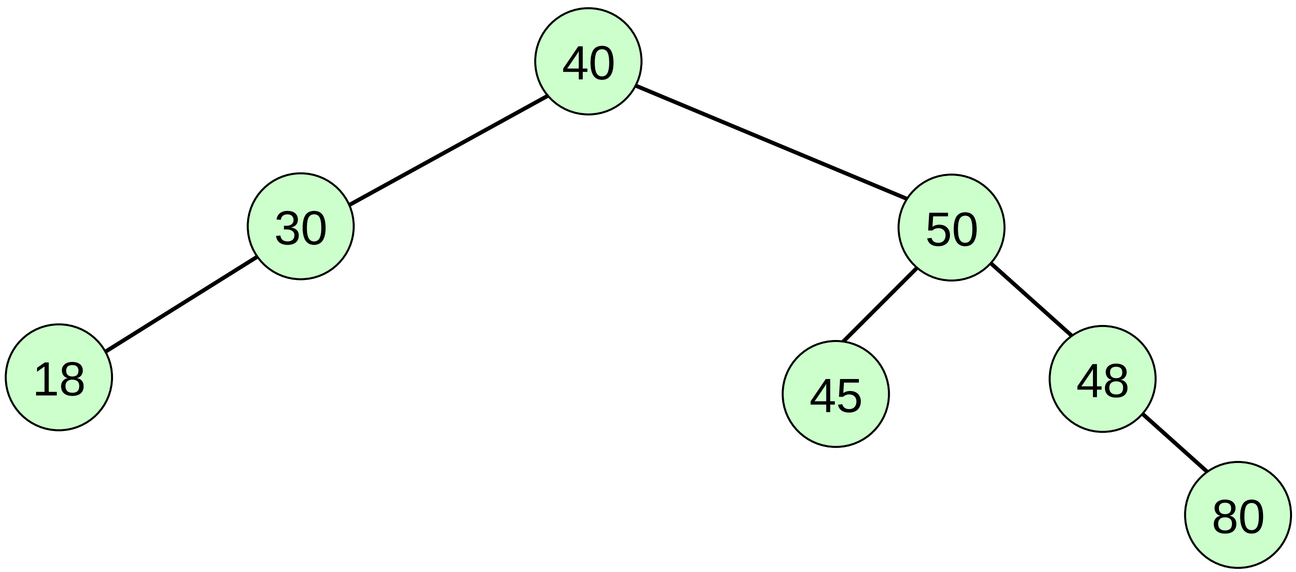
Não

Nó não balanceado



Árvores AVL

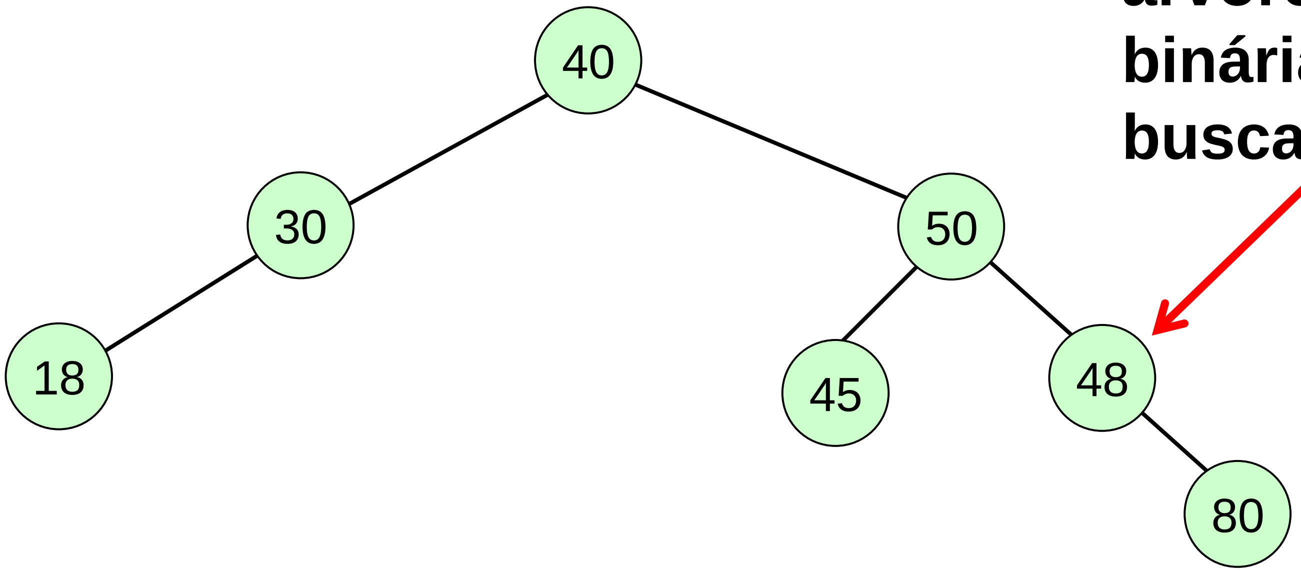
Esta árvore é AVL?



Árvores AVL

Não

Não é árvore binária de busca



Árvores AVL

Teorema:

Toda árvore AVL é balanceada

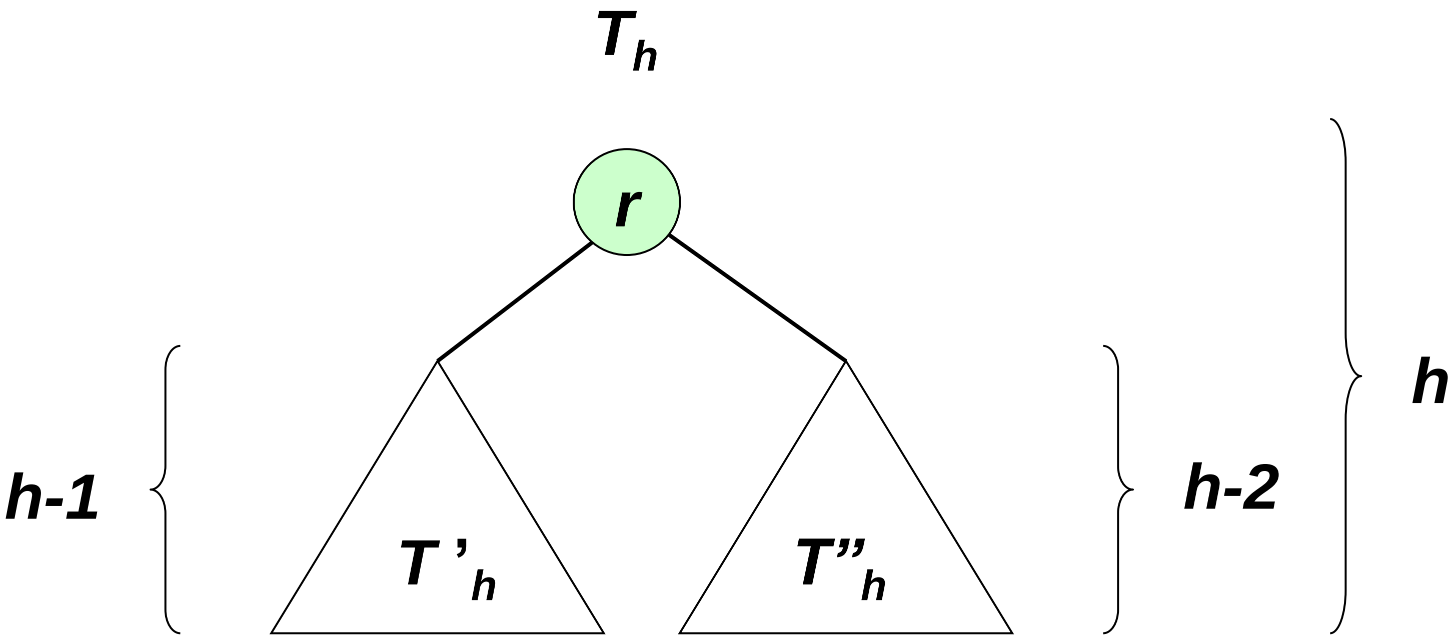
Prova: Deve-se mostrar que para um n fixo, a árvore AVL de altura máxima possui $h = O(\log n)$.

Supor T_h – Árvore de altura h com n mínimo

Árvores AVL

Toda árvore AVL é balanceada

Prova:

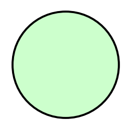


Árvores AVL

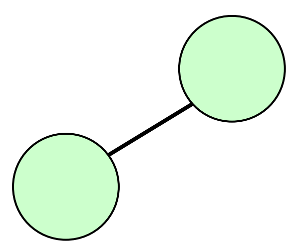
Toda árvore AVL é balanceada

Prova: Recursivamente, temos as seguintes T_h

$h = 1$ $T_1 \rightarrow$



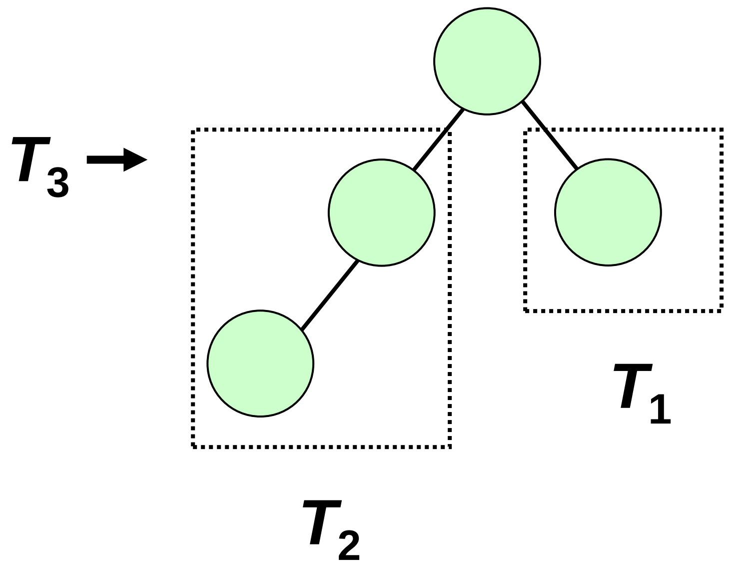
$h = 2$ $T_2 \rightarrow$



Árvores AVL

Toda árvore AVL é balanceada

$h = 3$

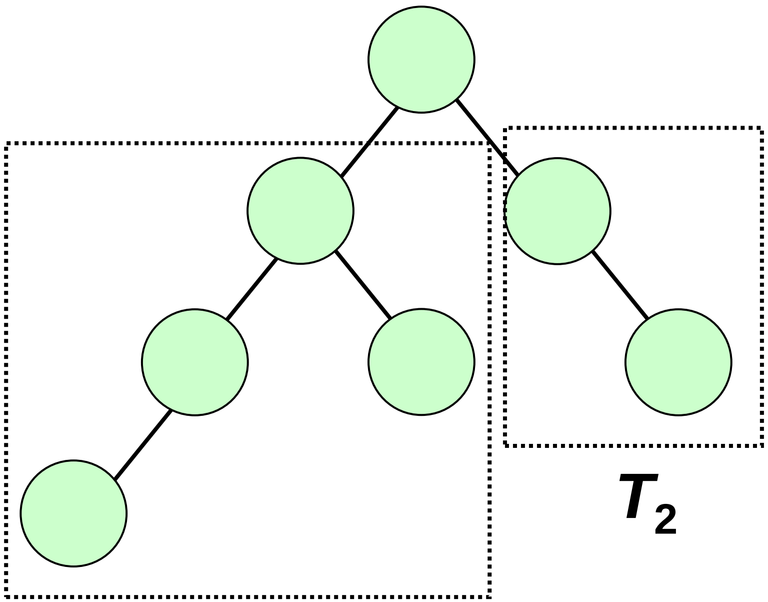


Árvores AVL

Toda árvore AVL é balanceada

$h = 4$

$T_4 \rightarrow$



T_2

T_3

Árvores AVL

Toda árvore AVL é balanceada

Se $h = 0$, então $|T_0| = 0$

Se $h = 1$, então $|T_1| = 1$

Se $h > 1$, então $|T_h| = 1 + \underbrace{|T_{h-1}| + |T_{h-2}|}_{\text{Como na sequência de Fibonacci}}$

Não tem na sequência de Fibonacci

h é $O(\log n)$

Árvores AVL

Para *h* > 1, o *h*-ésimo termo da sequência de Fibonacci é:

$$F_h = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^h - \left(\frac{1-\sqrt{5}}{2}\right)^h\right]$$

Como *h* > 0, o termo:

$$\frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^h < 1$$

Árvores AVL

Portanto, $$|T_h| > \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^h\right] - 1$$

Fazendo $$a = \left(\frac{1+\sqrt{5}}{2}\right)$$

Tem-se $$\log_a(|T_h|+1) > h - \log_a \sqrt{5}$$

Árvores AVL

Transformando o logaritmo para a base 2,

$$h < \frac{1}{\log_2 a} \log_2\left(\left|T_h\right| + 1\right) + \log_a \sqrt{5}$$

Como $|T_h| = n$, h é $O(\log n)$.

***A árvore AVL é balanceada*.**

Árvores AVL

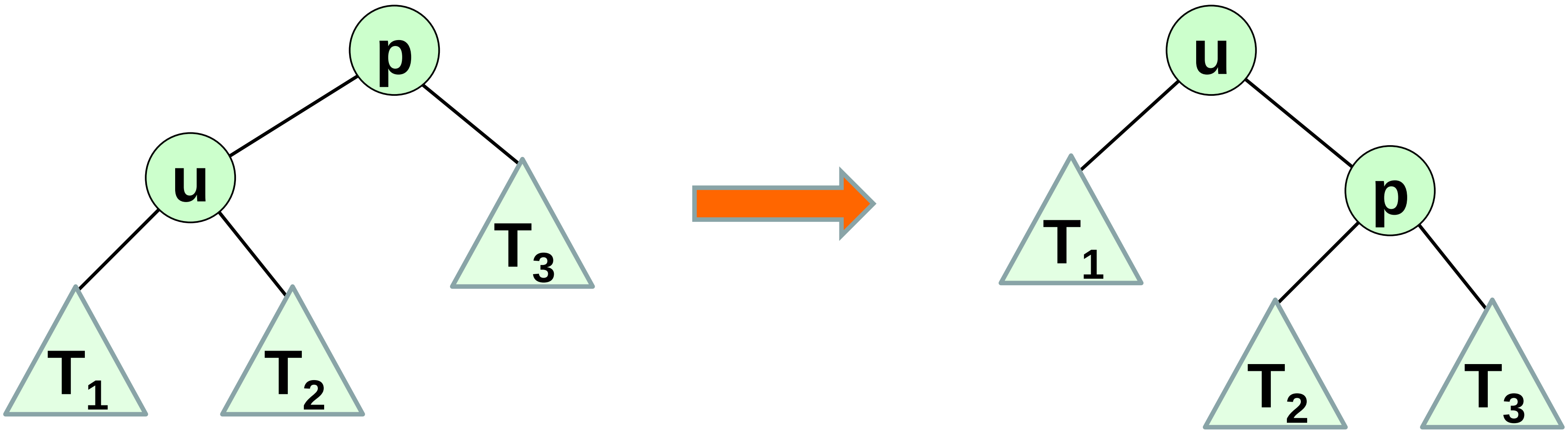
Operações:

Busca – idem árvore binária de busca

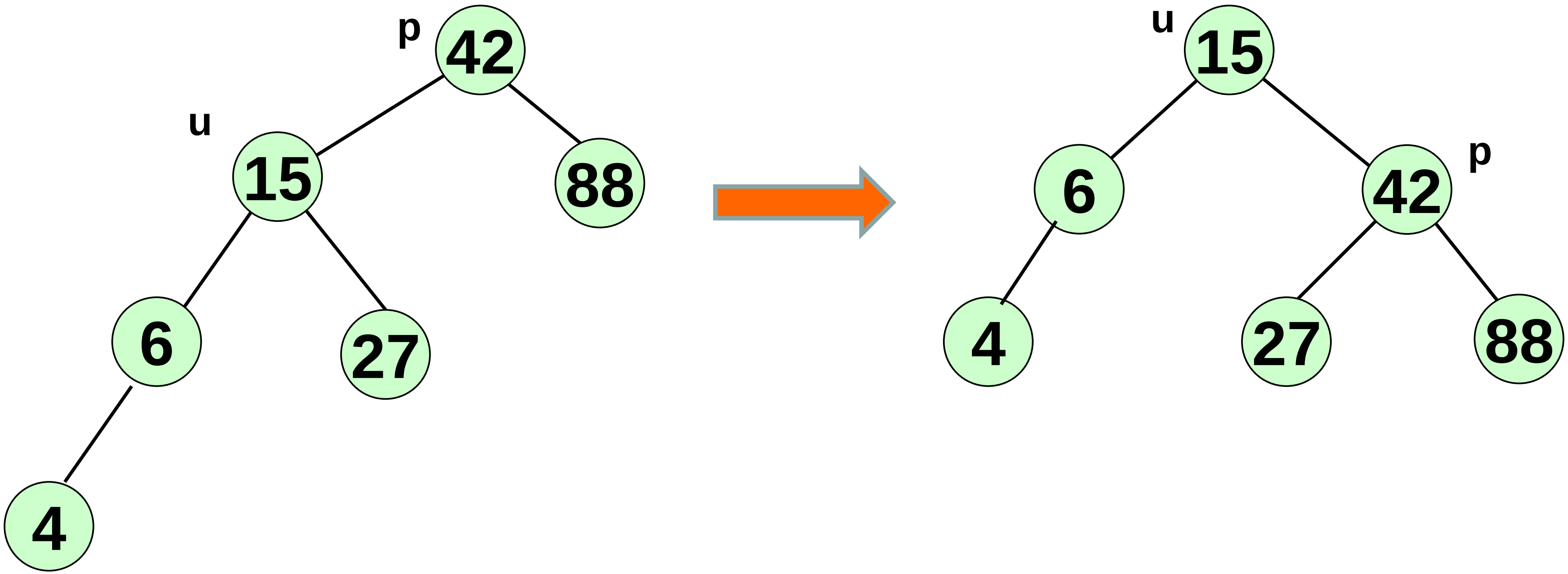
Inclusão

Remoção

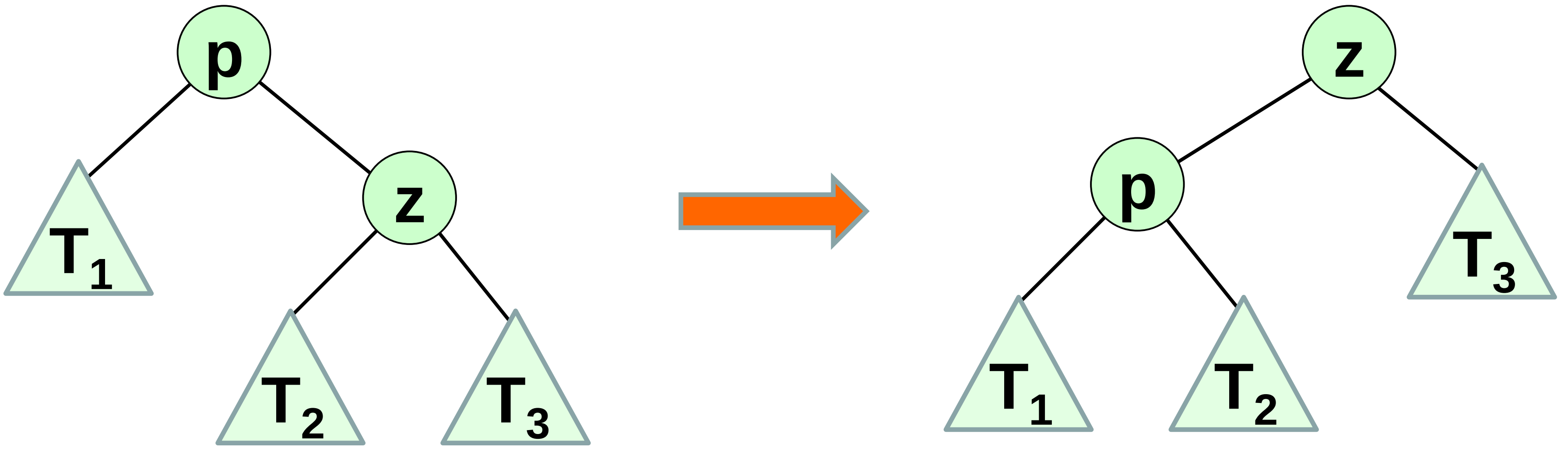
Rotação Direita



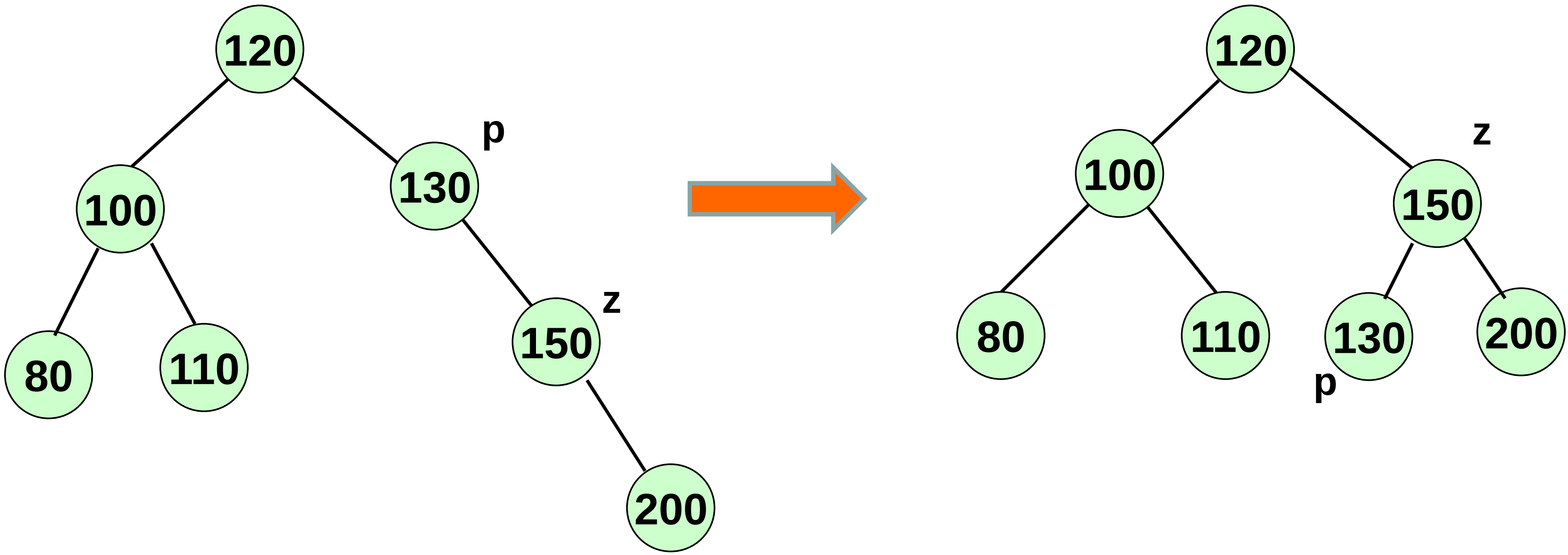
Exemplo: Rotação Direita



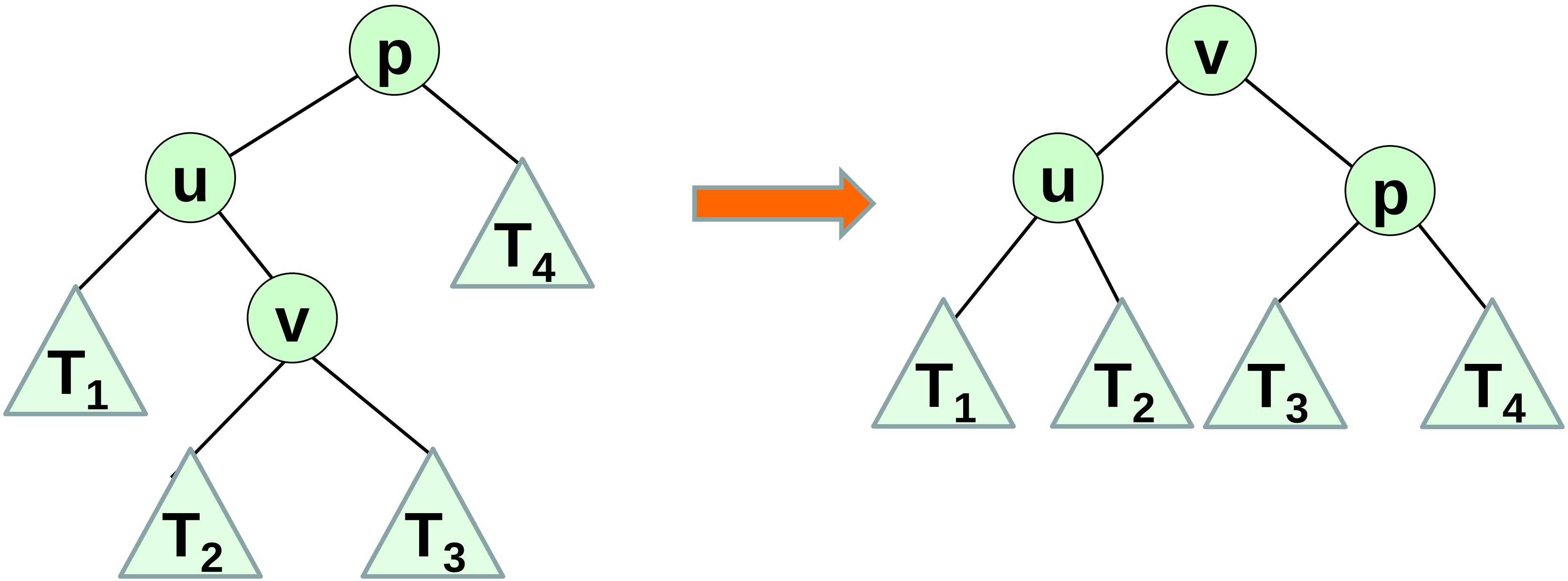
Rotação Esquerda



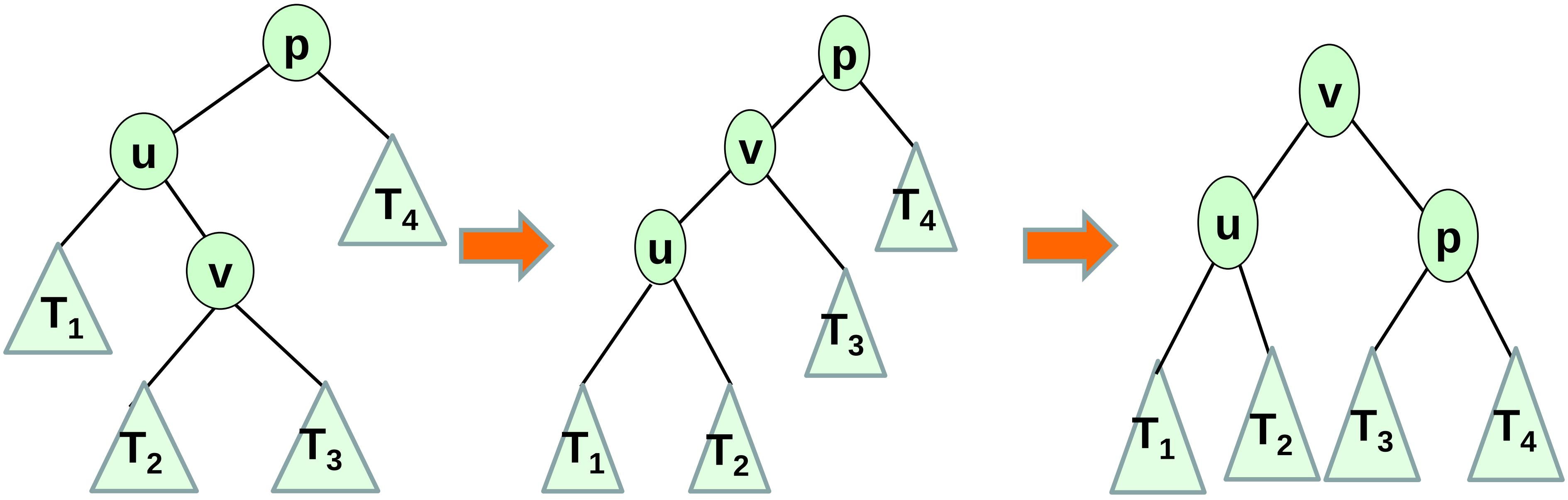
Exemplo: Rotação Esquerda



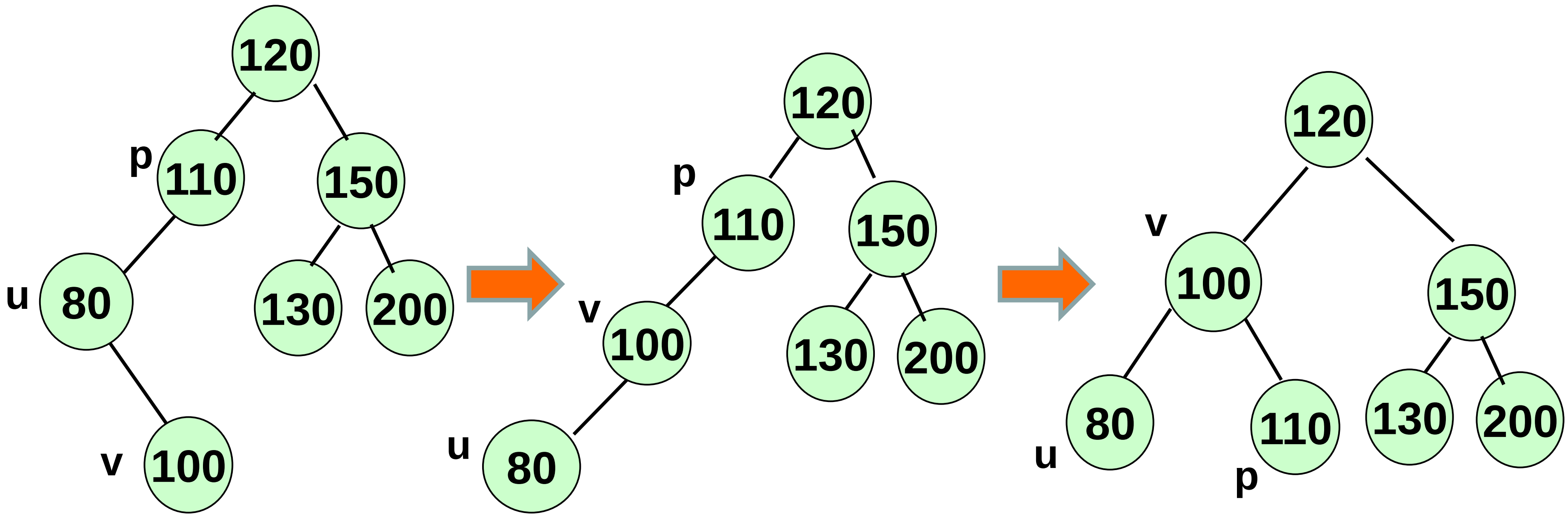
Rotação Dupla Direita



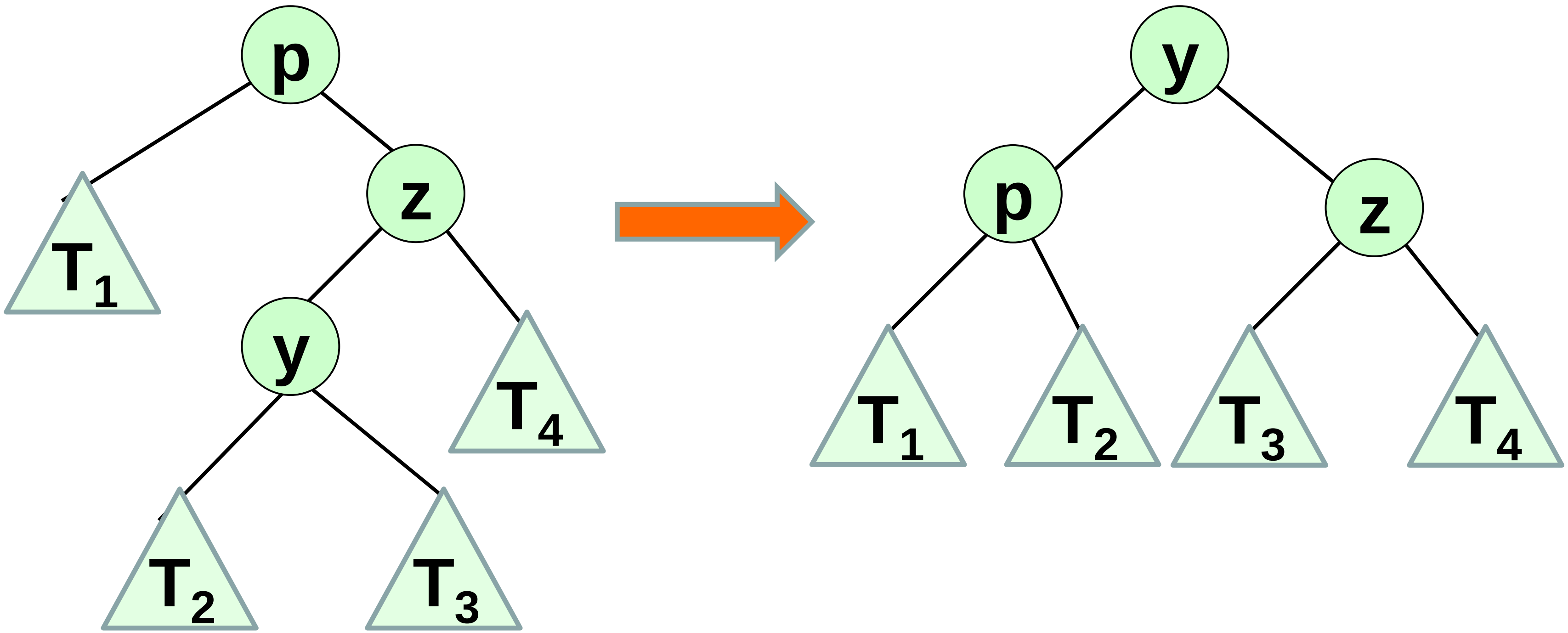
Rotação Dupla Direita



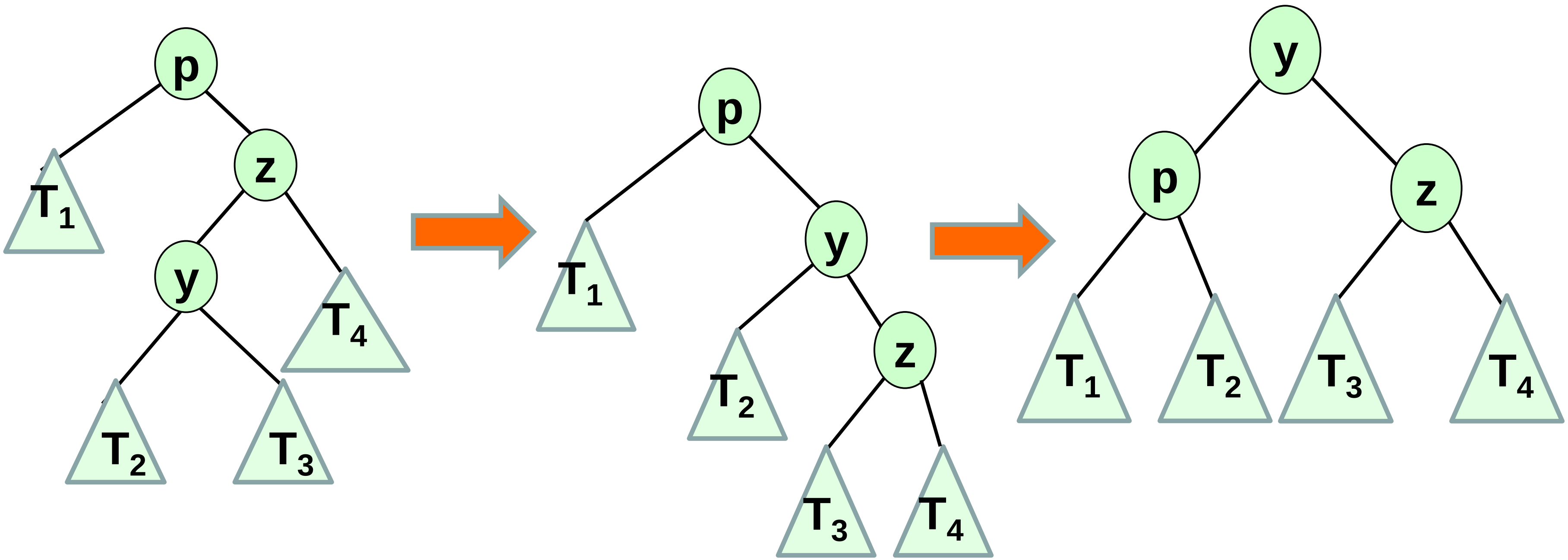
Rotação Dupla Direita



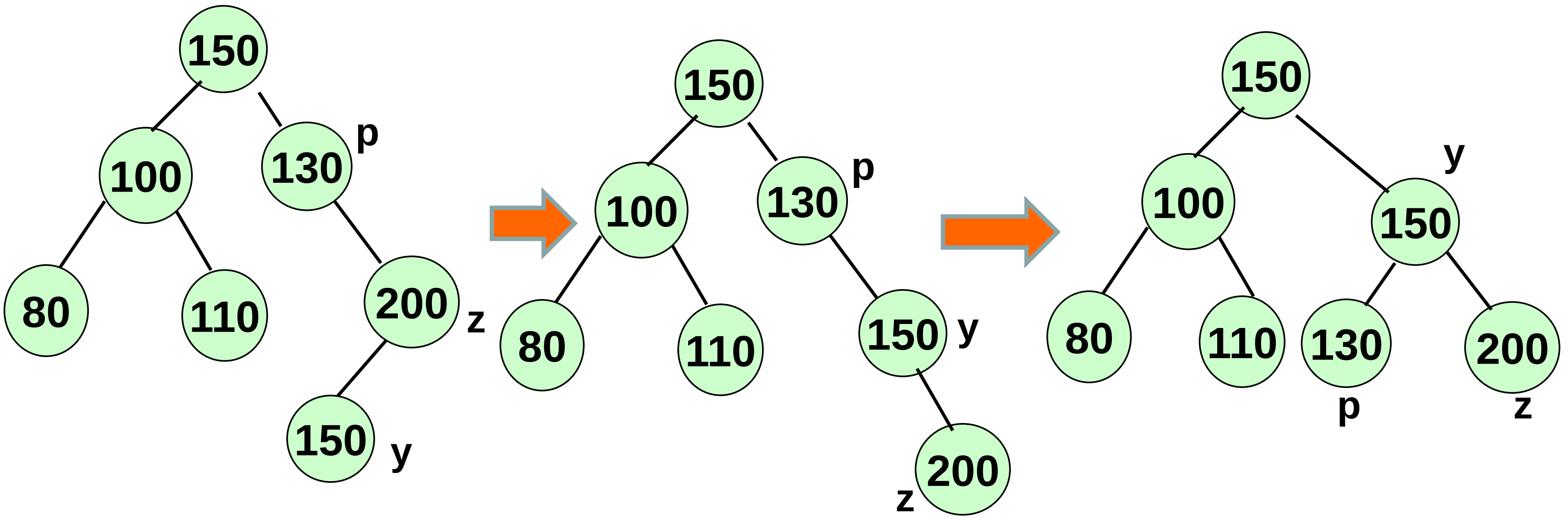
Rotação Dupla Esquerda



Rotação Dupla Esquerda



Rotação Dupla Esquerda



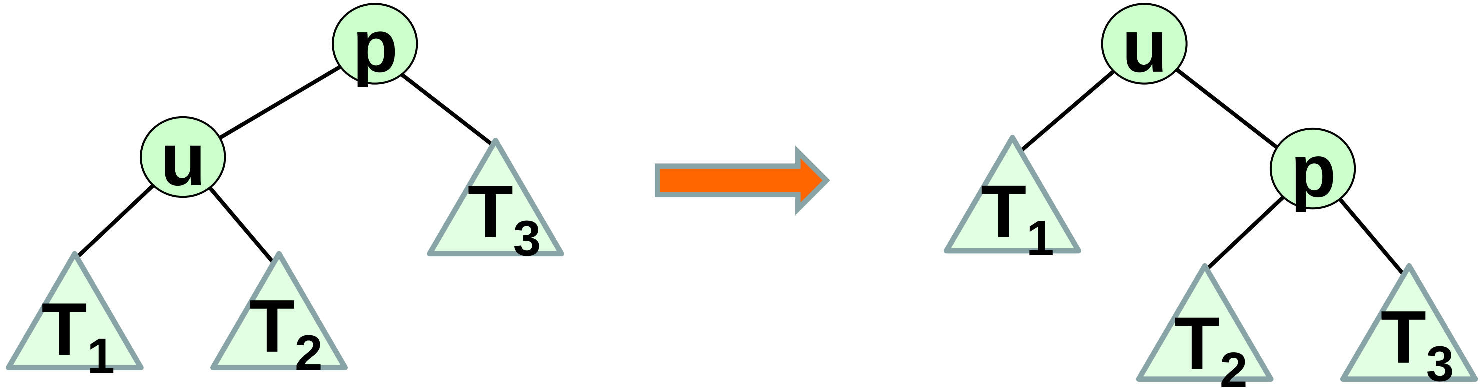
Rotação

O que vocês observaram em relação à altura da árvore antes e depois da rotação?

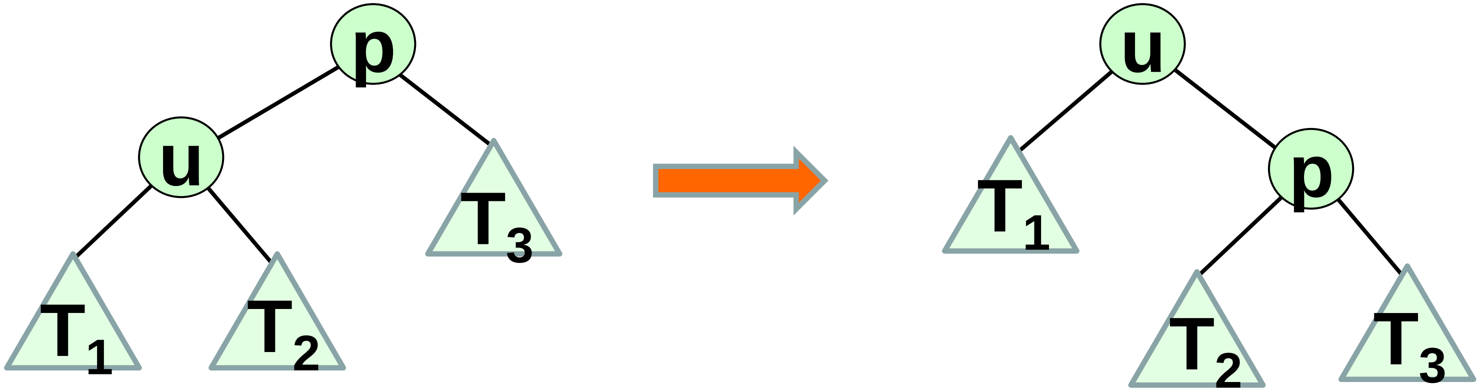
Exercício

Faça o código das rotações.

Faça o código da Rotação Direita



Rotação Direita



Rotacao_direita(*ptp*↑)

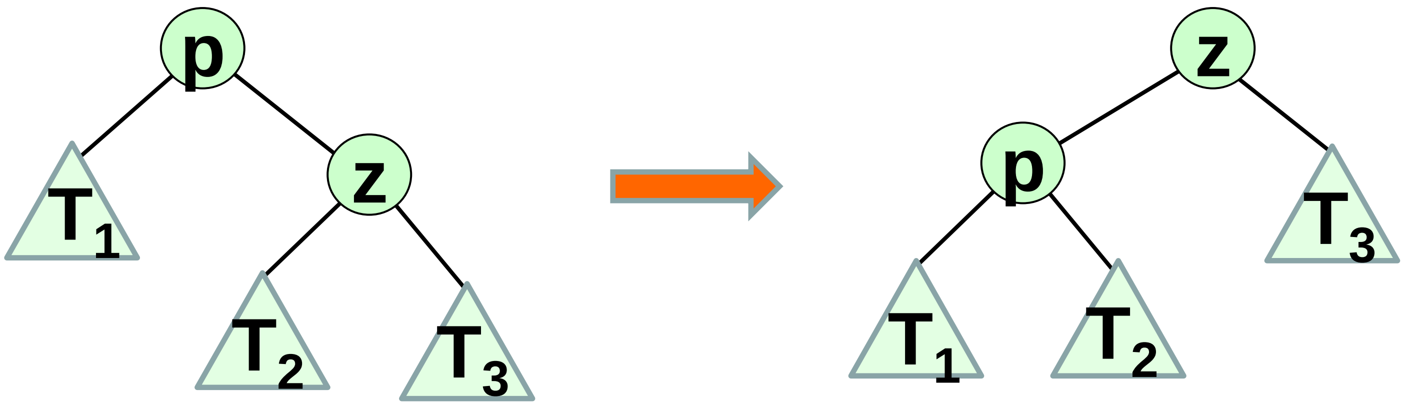
ptu* ← *pt*↑.*esq

ptp*↑.*esq* ← *ptu*↑.*dir;

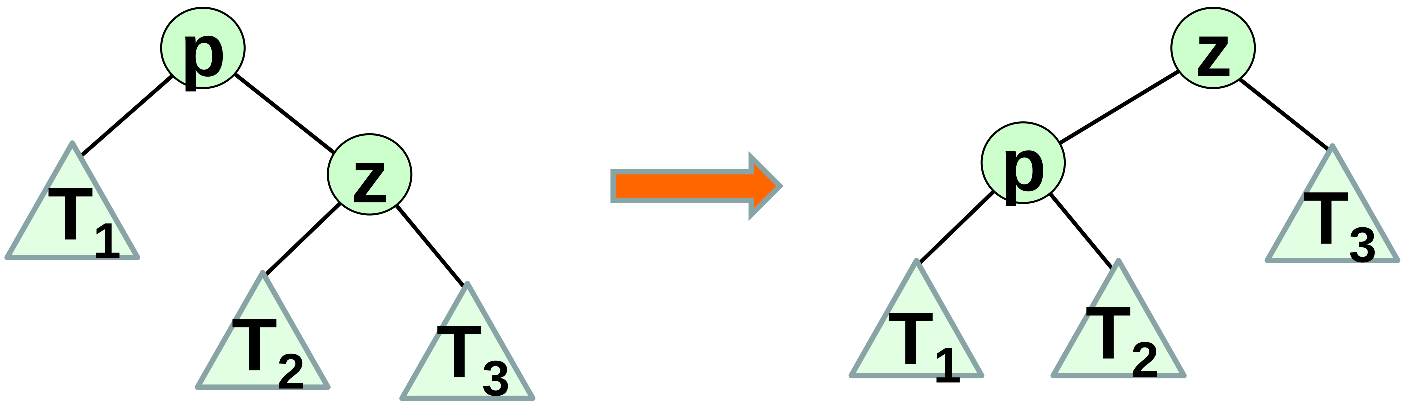
ptu*↑.*dir* ← *ptp

ptp* ← *ptu

Faça o código da Rotação Esquerda



Faça o código da Rotação Esquerda



Rotacao_esquerda(*ptp*↑)

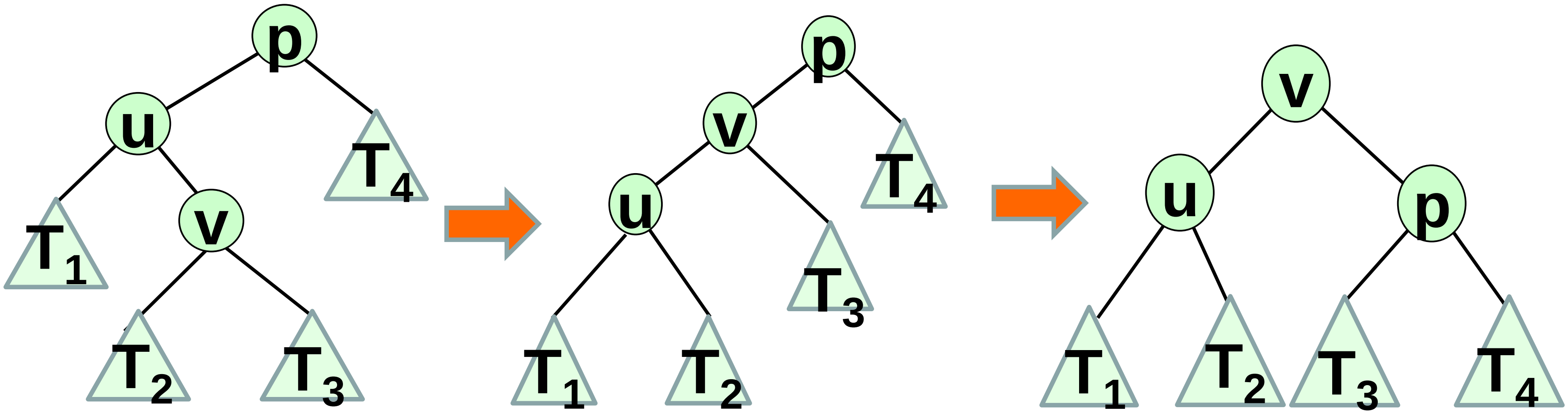
ptz* ← *ptp*↑.*dir

***ptp*↑.*dir* ← *ptz*↑.*esq*;**

ptz*↑.*esq* ← *ptp

ptp* ← *ptz

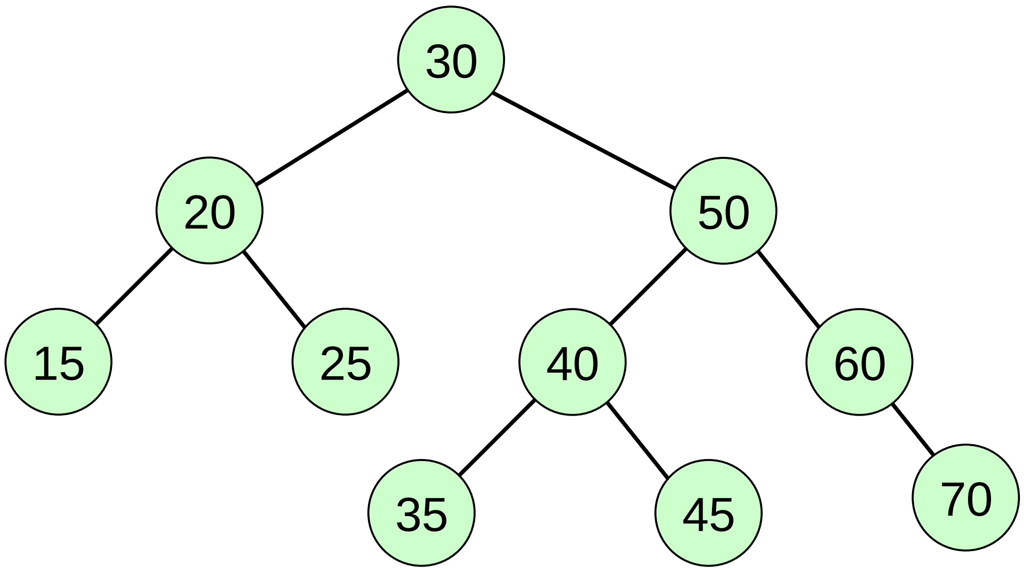
Faça o código da Rotação Dupla Direita



Rotacao_esquerda(*ptu*↑)
Rotacao_direita(*ptp*↑)

Árvore AVL

Essa árvore é AVL?

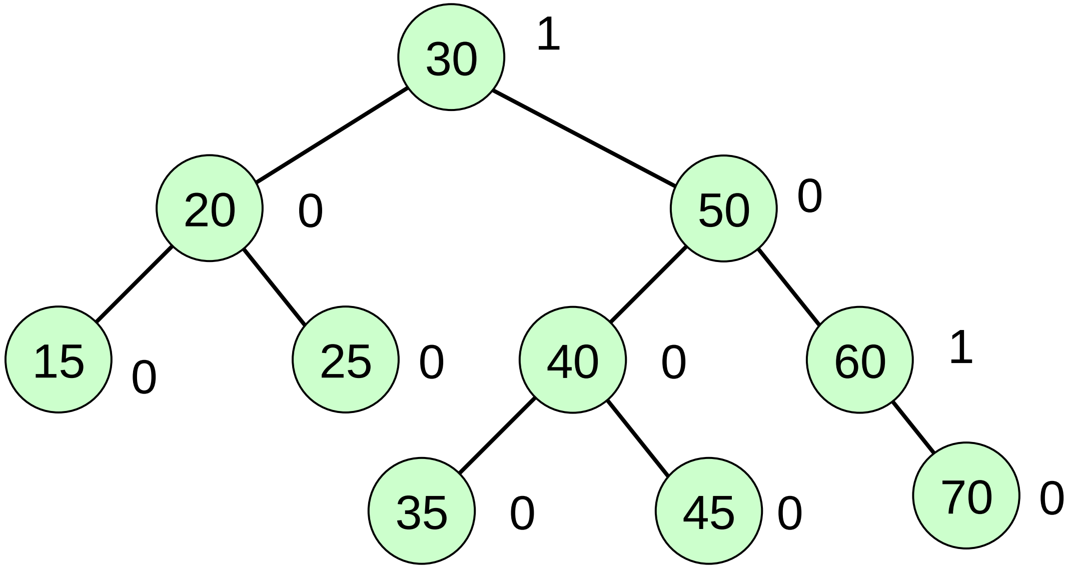


Para responder, vamos usar o conceito de balanço de um nó v

Balanço(v) = $h_D(v) - h_E(v)$

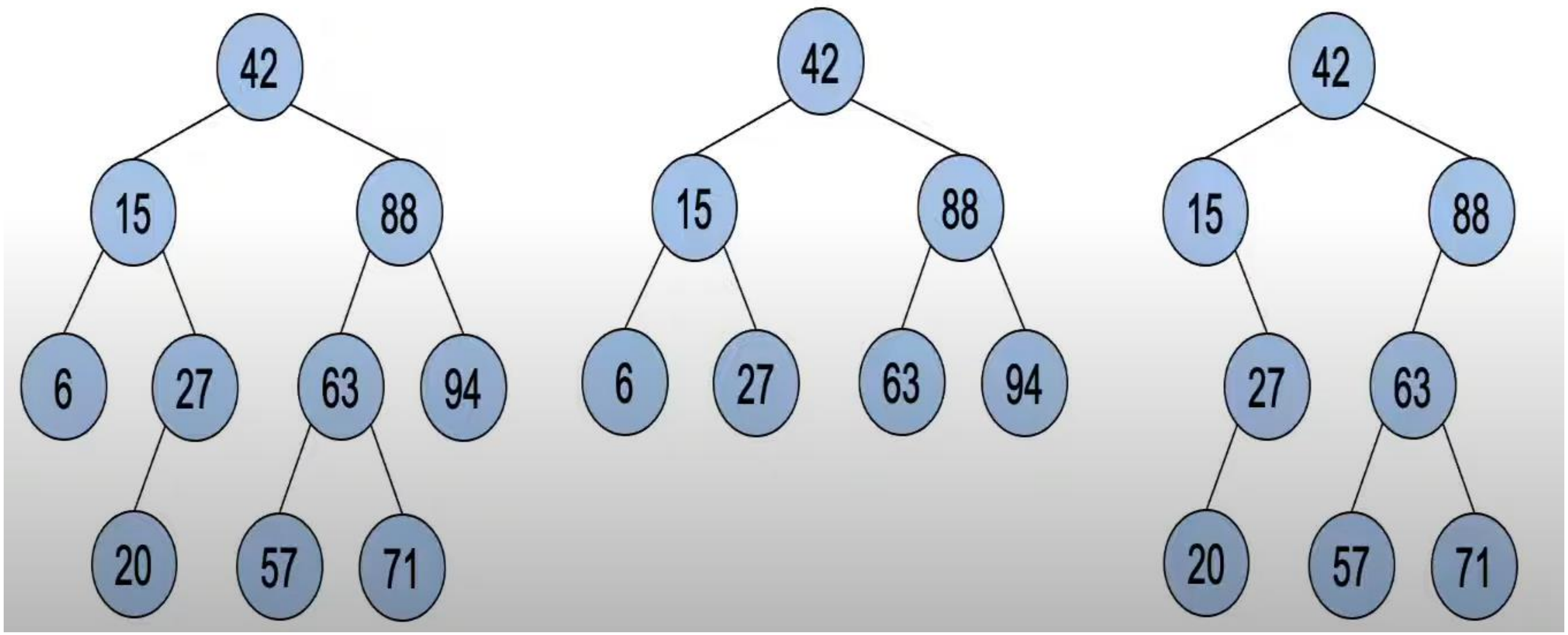
Árvore AVL

Árvore AVL



Árvore AVL

Exercício: quais dessas árvores são AVL?



Árvore AVL

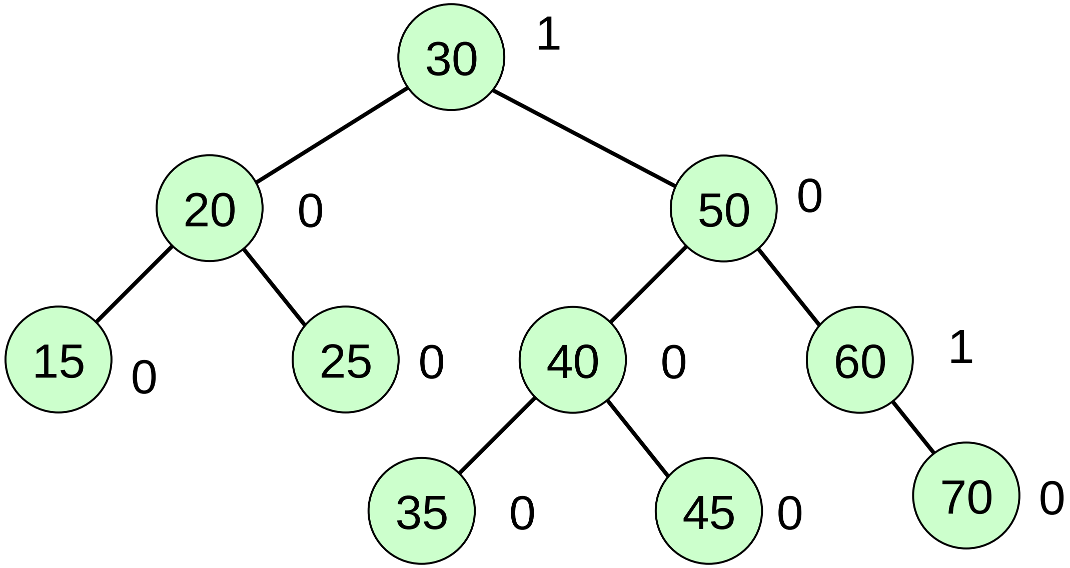
Exercício

Insira os elementos: 400, 140, 120, 130, 150, 200, 250, 350

Inclusão em Árvore AVL

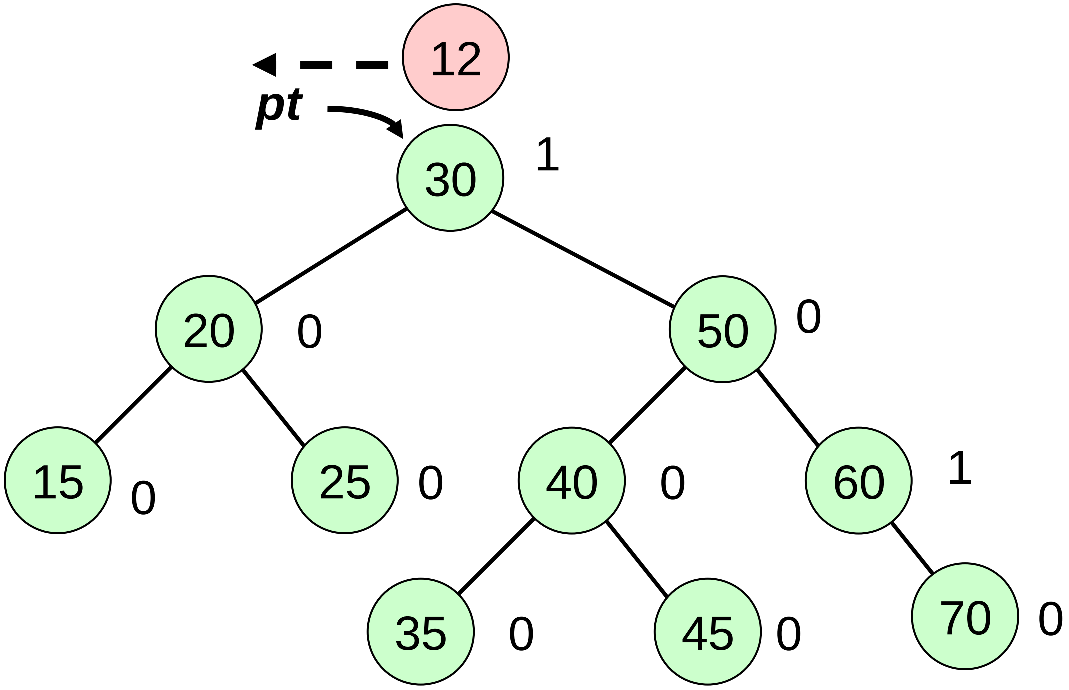
Inclusão em Árvore AVL

Árvore AVL



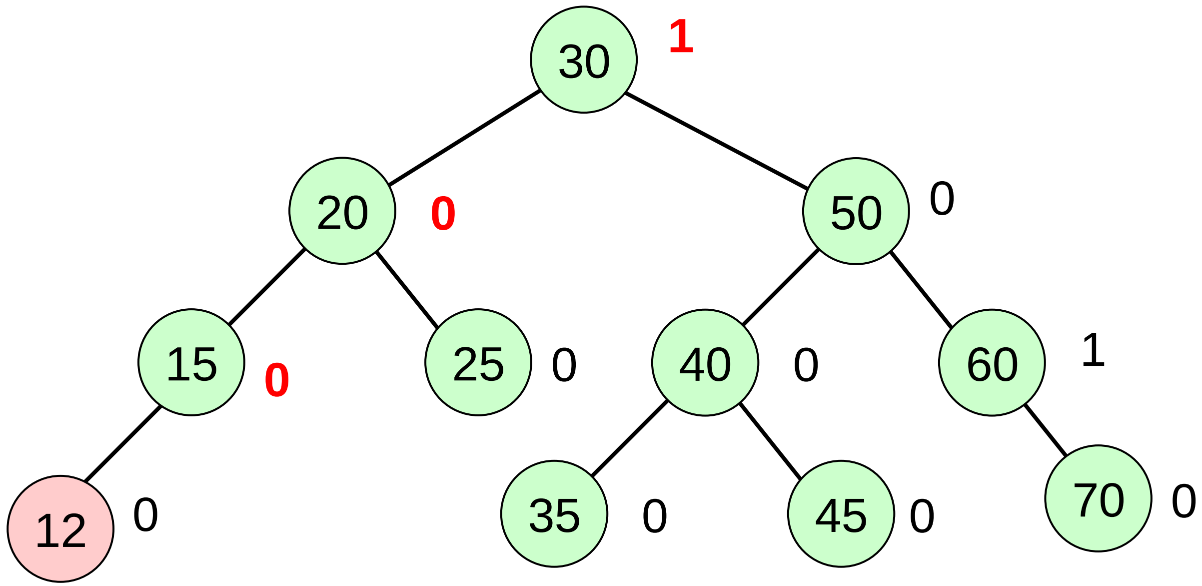
Inclusão em Árvore AVL

Inclusão da chave 12



Inclusão em Árvore AVL

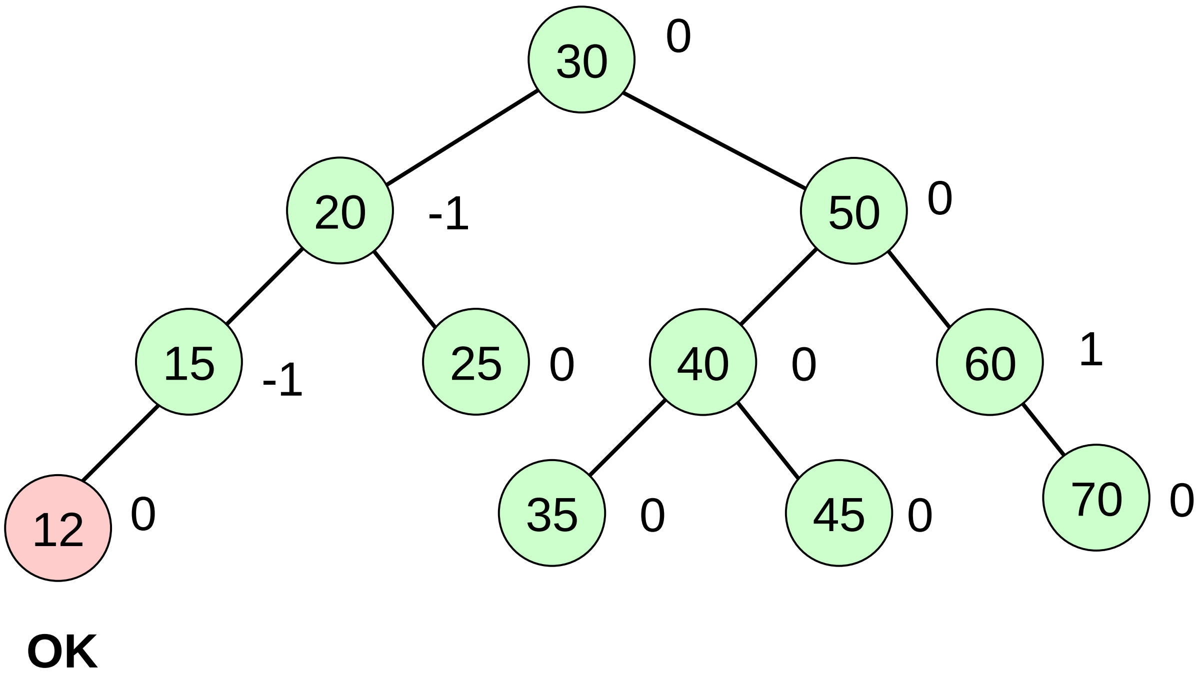
Inclusão da chave 12



O balanceamento pode mudar com a inserção

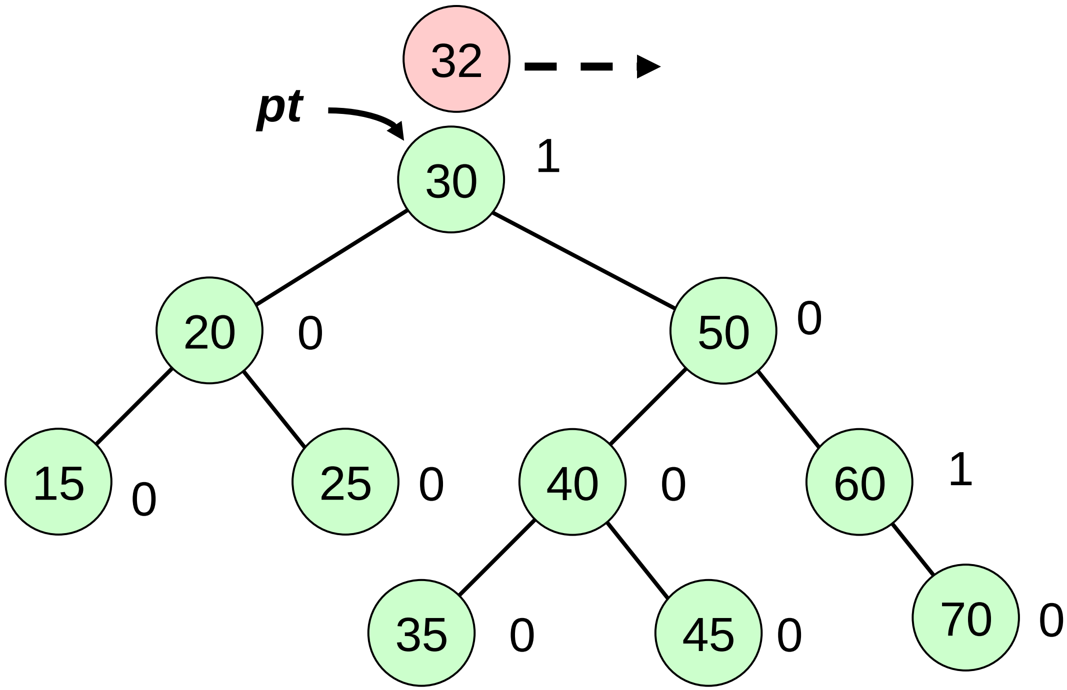
Inclusão em Árvore AVL

Inclusão da chave 12



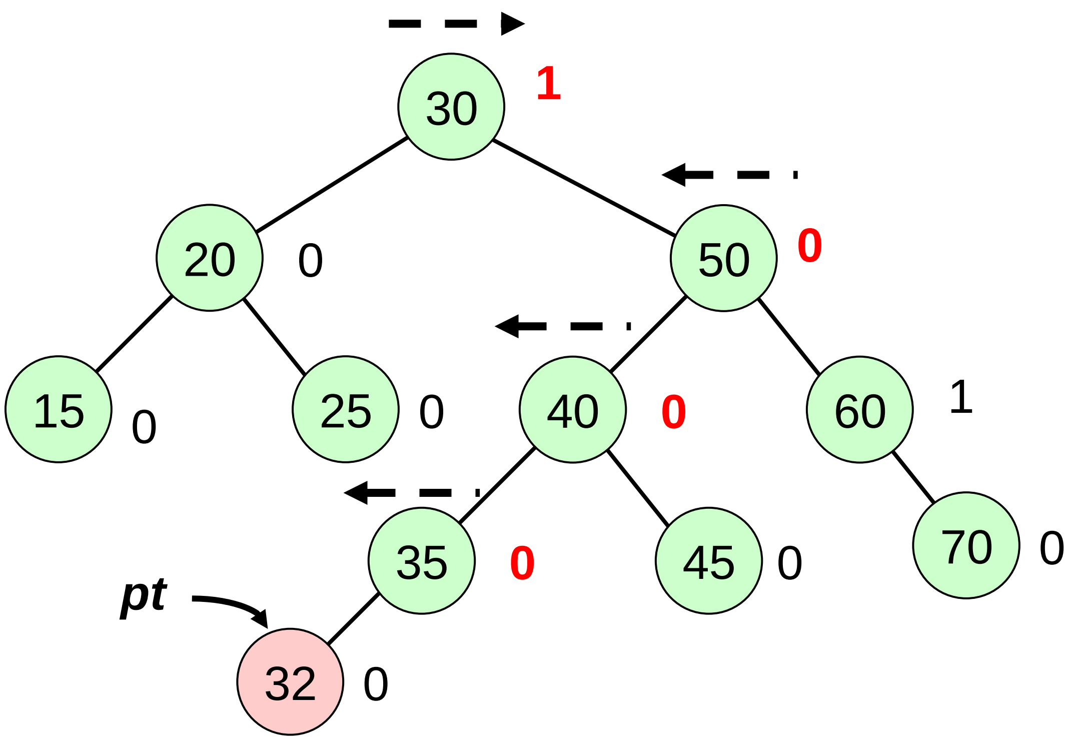
Inclusão em Árvore AVL

Inclusão da chave 32



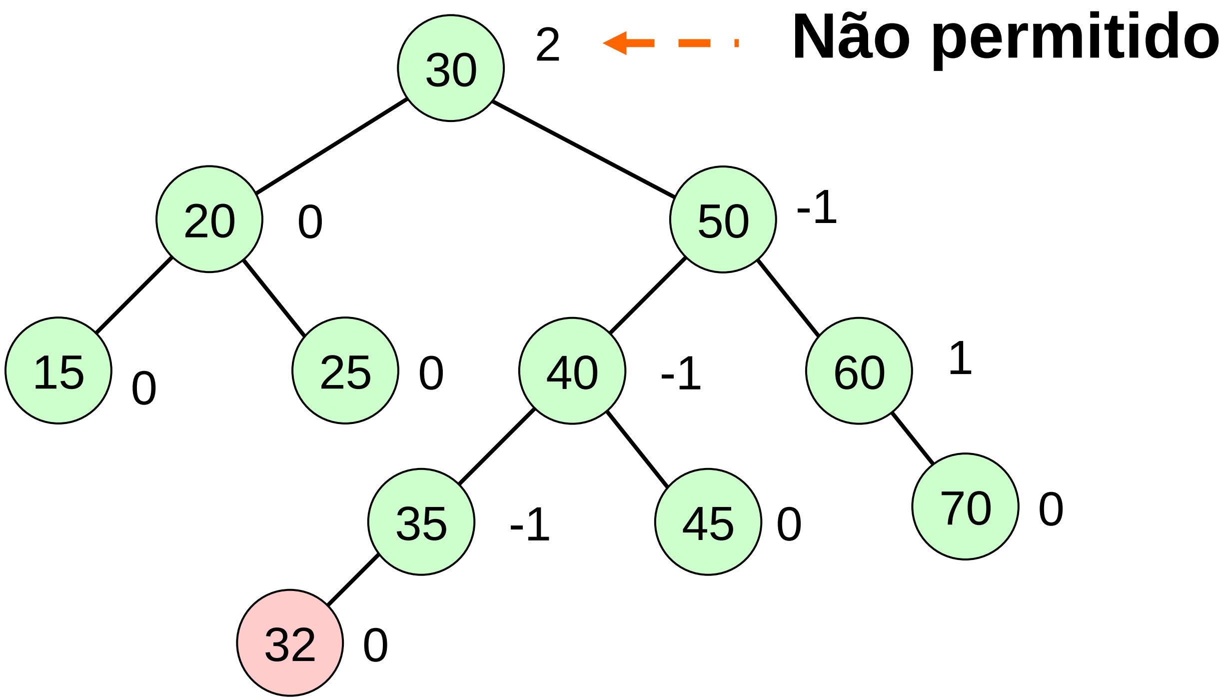
Inclusão em Árvore AVL

Inclusão da chave 32



Inclusão em Árvore AVL

Inclusão da chave 32



Inclusão em Árvore AVL

Como resolver isto?

Inclusão em Árvore AVL

1. Busca chave

2. Se a chave não é encontrada, inclui nó *q*.

3. Se algum nó *p* ficou desregulado, então regular *p*.

OBS:

Quando para algum nó p, $|h_E(p) - h_D(p)| = 2$,

É necessário balanceamento.

Nó desregulado: p

|he(p)-hd(p)| = 2

he(p) > hd(p)	u = esq (p) he(u) > hd(u)	RD (p)
he(p) > hd(p)	u = esq (p) he(u) < hd(u)	RDD (p)
he(p) < hd(p)	u = dir (p) he(u) < hd(u)	RE (p)
he(p) < hd(p)	u = dir (p) he(u) > hd(u)	RDE (p)

Inclusão em Árvores AVL

Quando para algum nó p, $|h_E(p) - h_D(p)| = 2$, pode-se identificar as seguintes situações:

Caso 1. $h_E(p) > h_D(p)$

Neste caso q pertence à subárvore esquerda de p. Além disso, p possui filho esquerdo $u \neq q$. Sabe-se que $h_E(u) \neq h_D(u)$.

Caso 1.1 $h_E(u) > h_D(u)$.

$h(T_1) – h(T_2) = 1$ e $h(T_2) = h(T_3)$

Aplicar Rotação Direita

Inclusão em Árvores AVL

Caso 1.2 $h_E(u) < h_D(u)$.

Então u possui filho direito v.

$|h(T_2) - h(T_3)| \leq 1$

e

$\max\{h(T_2), h(T_3)\} = h(T_1) = h(T_4)$

Aplicar Rotação Dupla Direita

Caso 2. Análogo

Inclusão em Árvores AVL

Balanço do nó v

$$\text{Balanço}(v) = h_D(v) - h_E(v)$$

Nó regulado: $-1 \leq \text{balanço}(v) \leq 1$

Suponha que o nó *q* foi inserido na subárvore esquerda de *v*.

Caso 1. balanço(v) = 1 (antes da inclusão)

Nesse caso balanço(v) se torna 0 e a altura de *v* não foi modificada. Consequentemente, a altura dos nós entre a raiz e v também não se modificam.

Inclusão em Árvores AVL

$$\text{Balanço}(v) = h_D(v) - h_E(v)$$

Caso 2. balanço(v) = 0 (antes da inclusão)
Nesse caso balanço(v) torna-se -1 e a altura de *v* foi modificada. Consequentemente, a altura dos nós entre a raiz e v podem ter sido modificados e devem ser analisados.

Se v é a raiz, então a análise se encerra. Caso contrário, substituir *v* por seu pai e continuar a análise.

Inclusão em Árvores AVL

$$\text{Balanço}(v) = h_D(v) - h_E(v)$$

Caso 3. balanço(v) = -1 (antes da inclusão)
Nesse caso balanço(v) torna-se -2 e o nó está desregulado.
A rotação correta deve ser empregada.
Qualquer rotação implica que a subárvore resultante tenha a mesma altura da subárvore antes da inclusão. As alturas dos ancestrais de v não mais necessitam de avaliação.

Alterações de Balanço

Bal antes	**Ins esq**	**Ins dir**
1	0 Parar a verificação	RE (baldir = 1) RDE (baldir = -1) Parar a verificação
-1	RD (balesq = -1) RDD (balesq = 1) Parar a verificação	0 Parar a verificação
0	-1 Continuar verificação	1 Continuar verificação

$$\text{Balanço}(v) = h_D(v) - h_E(v)$$

Inclusão em Árvore AVL

Procedimento cria_no(*pt*)
    ocupar(*pt*)
    $pt\uparrow.esq \leftarrow \lambda$
    $pt\uparrow.dir \leftarrow \lambda$
    $pt\uparrow.chave \leftarrow x$
    $pt\uparrow.bal \leftarrow 0$

Inclusão em Árvore AVL

```
Procedimento insere_AVL(x, pt, h)
   se pt = λ então cria_no(pt); h ← "V"
   senão
       se (x = pt↑.chave) então PARE
       se (x < pt↑.chave) então
          insere_AVL(x, pt↑.esq,h)
          se h então
              caso pt↑.bal seja
              1 : pt↑.bal ← 0; h ← "F"
              0 : pt↑.bal ← -1
              -1: caso1(pt, h)    //Rebalanceamento
       senão
          insere_AVL(x, pt↑.dir,h)
          se h então
              caso pt↑.bal seja
              -1 : pt↑.bal ← 0; h ← "F"
               0 : pt↑.bal ← 1
               1 : caso2(pt, h)     //Rebalanceamento
```

Inclusão em Árvore AVL

Procedimento caso1(*pt*,*h*)

ptu ← _pt_↑._esq_

se _ptu_↑._bal_ = -1 então

_pt_↑._esq_ ← _ptu_↑._dir_; _ptu_↑._dir_ ← _pt_

_pt_↑._bal_ ← 0; _pt_ ← _ptu_

Rotação Direita

senão

Rot Dup Dir

ptv ← _ptu_↑._dir_

ptu↑._dir_ ← _ptv_↑._esq_ ; _ptv_↑._esq_ ← _ptu_

_pt_↑._esq_ ← _ptv_↑._dir_ ; _ptv_↑._dir_ ← _pt_

se (_ptv_↑._bal_ = -1) então _pt_↑._bal_ ← 1 senão _pt_↑._bal_ ← 0

se (_ptv_↑._bal_ = 1) então _ptu_↑._bal_ ← -1 senão _ptu_↑._bal_ ← 0

_pt_ ← _ptv_

_pt_↑._bal_ ← 0 ; _h_ ← “F”

Inclusão em Árvore AVL

Procedimento caso2(*pt*,*h*)

$pt u \leftarrow pt\uparrow.dir$

se $ptu\uparrow.bal = 1$ então

$pt\uparrow.dir \leftarrow ptu\uparrow.esq$; $ptu\uparrow.esq \leftarrow pt$

$pt\uparrow.bal \leftarrow 0$; $pt \leftarrow ptu$

Rotação Esquerda

senão

Rot Dup Esq

$ptv \leftarrow ptu\uparrow.esq$

$ptu\uparrow.esq \leftarrow ptv\uparrow.dir$; $ptv\uparrow.dir \leftarrow ptu$

$pt\uparrow.dir \leftarrow ptv\uparrow.esq$; $ptv\uparrow.esq \leftarrow pt$

se ($ptv\uparrow.bal = 1$) então $pt\uparrow.bal \leftarrow -1$ senão $pt\uparrow.bal \leftarrow 0$

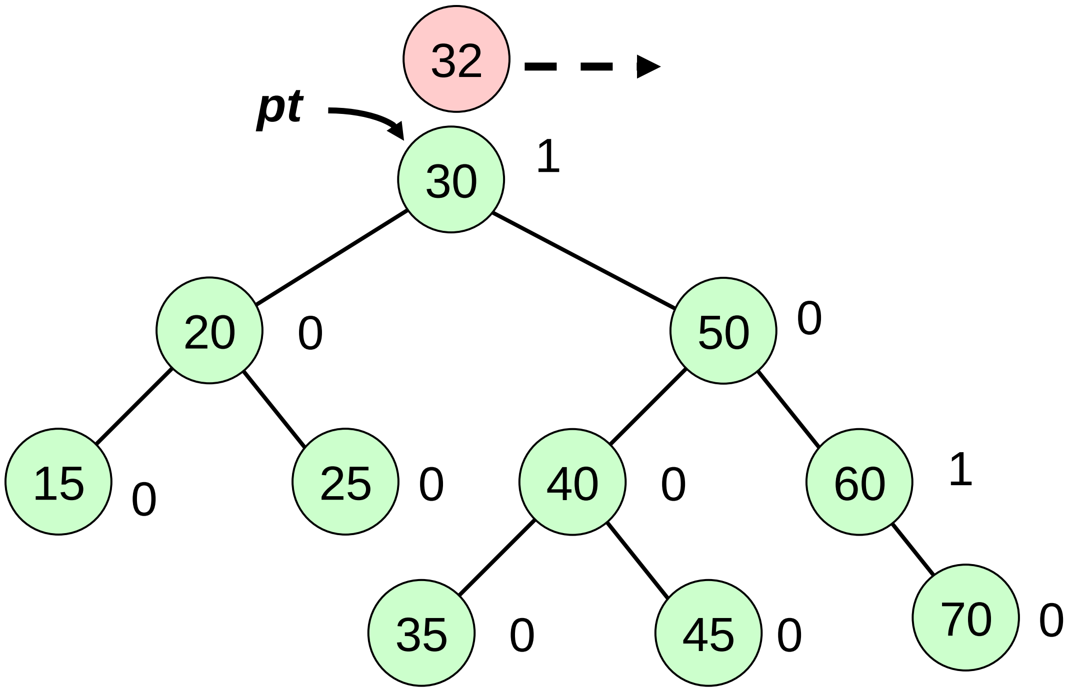
se ($ptv\uparrow.bal = -1$) então $ptu\uparrow.bal \leftarrow 1$ senão $ptu\uparrow.bal \leftarrow 0$

$pt \leftarrow ptv$

$pt\uparrow.bal \leftarrow 0$; $h \leftarrow$ “F”

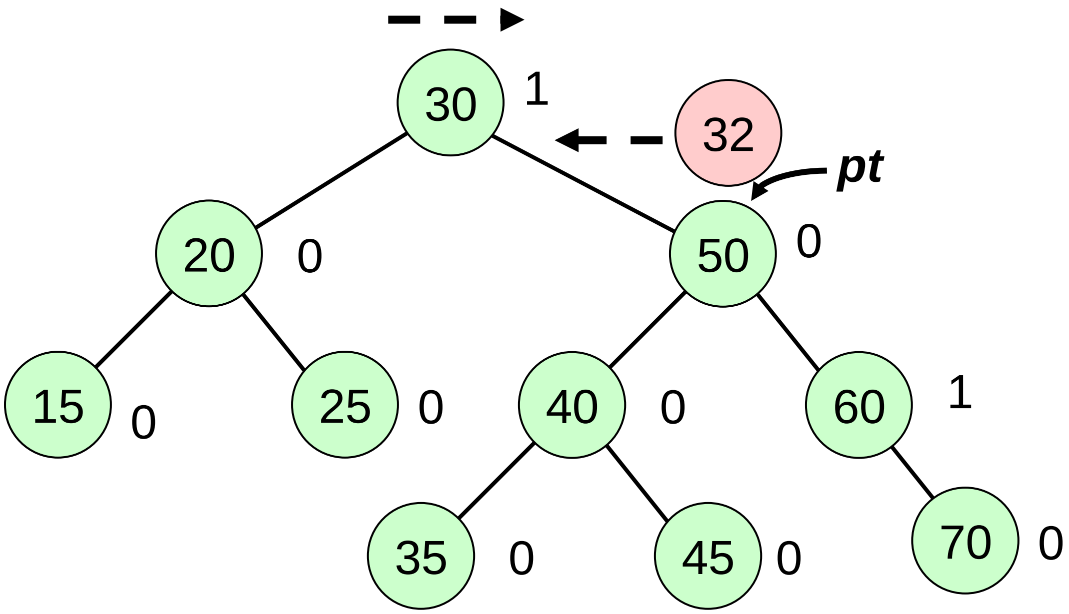
Inclusão em Árvore AVL

Inclusão da chave 32



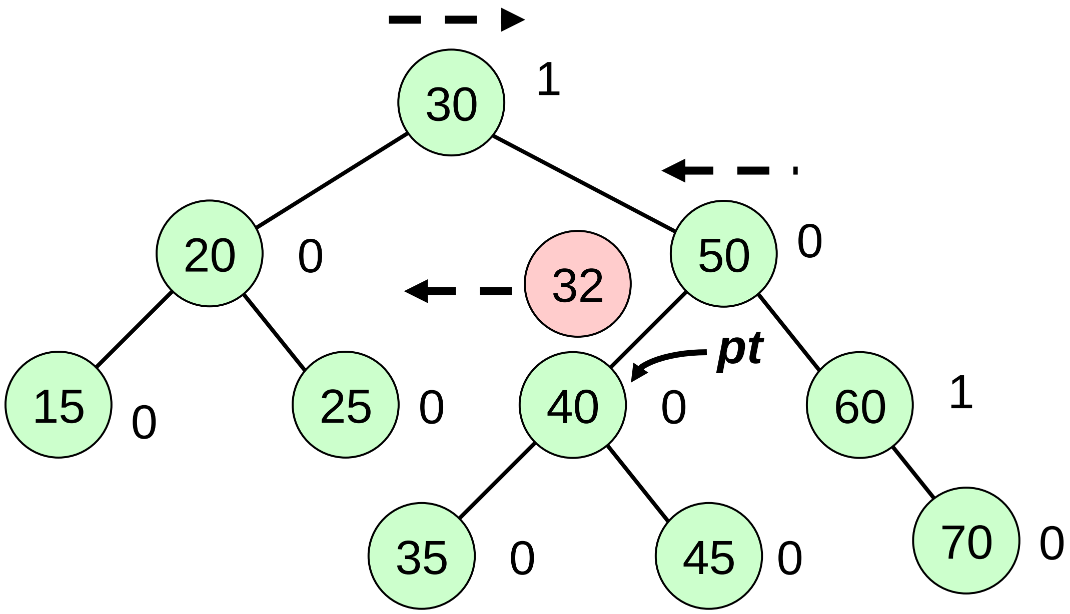
Inclusão em Árvore AVL

Inclusão da chave 32



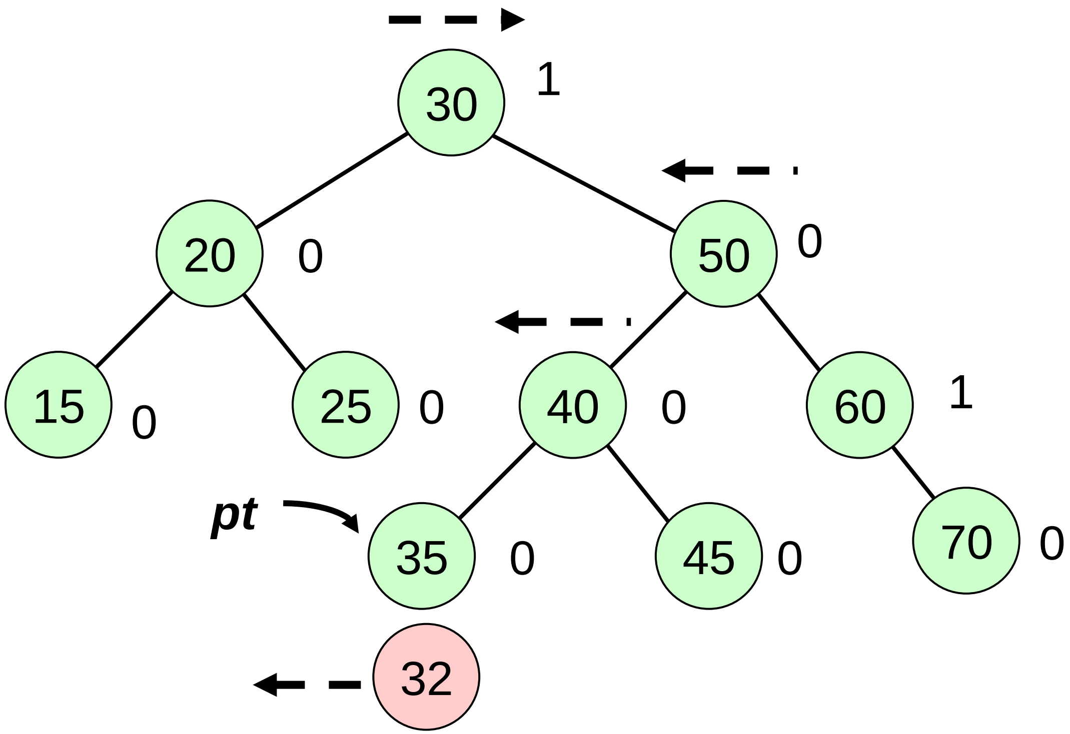
Inclusão em Árvore AVL

Inclusão da chave 32



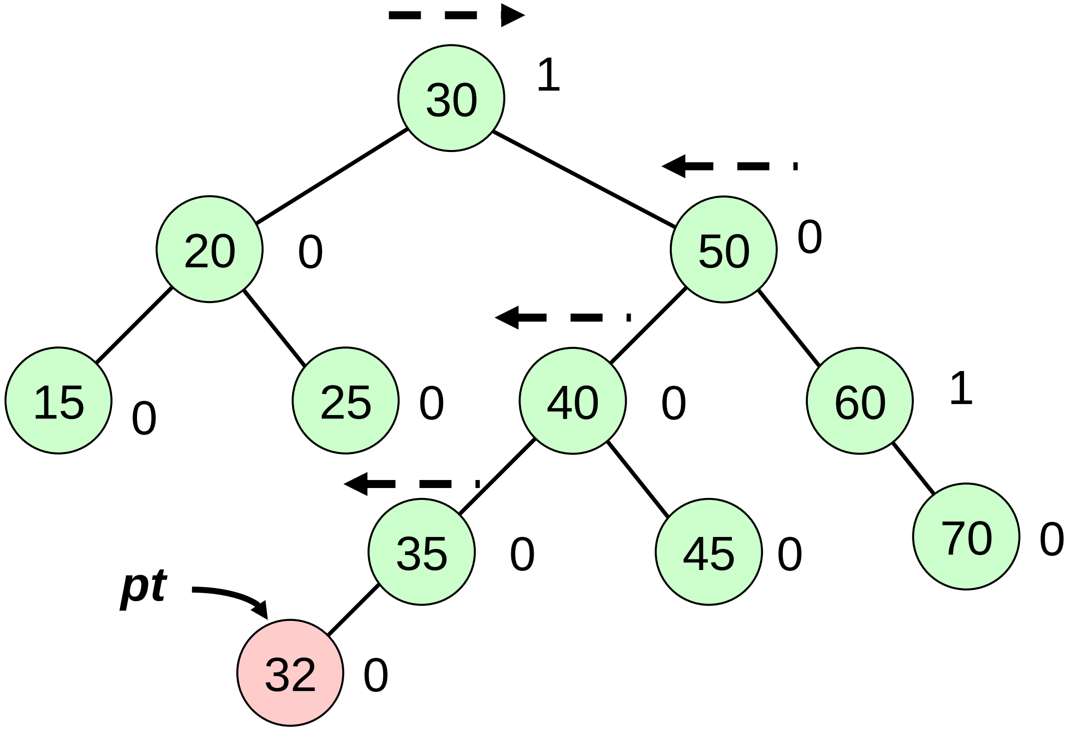
Inclusão em Árvore AVL

Inclusão da chave 32



Inclusão em Árvore AVL

Inclusão da chave 32

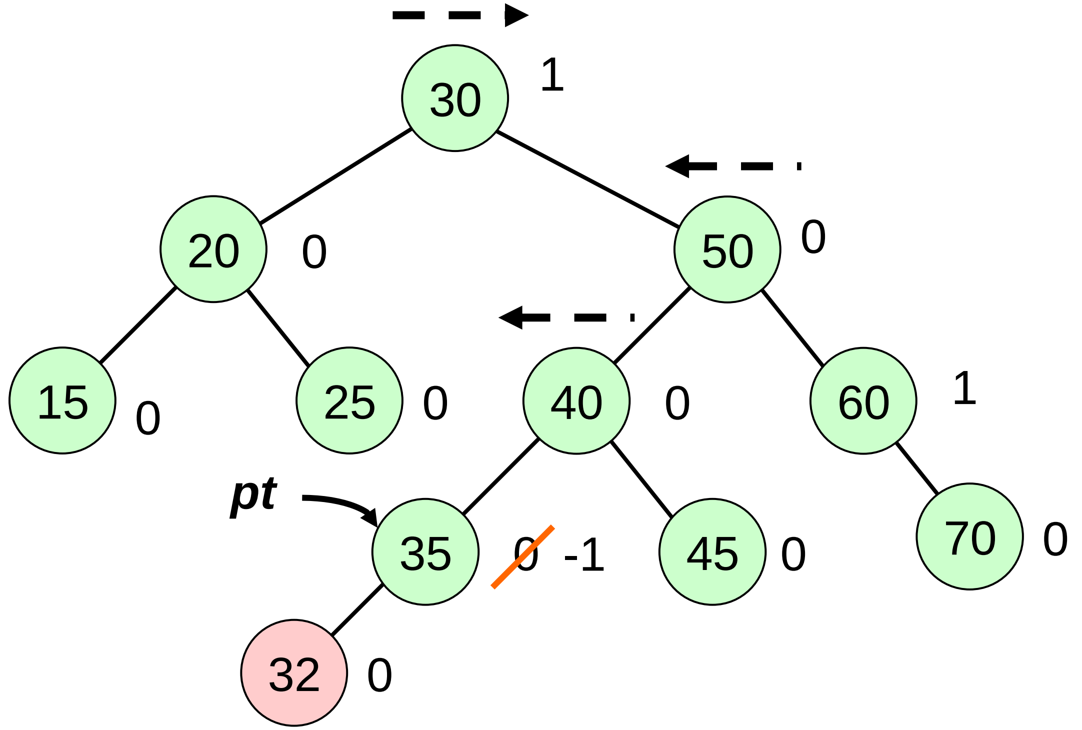


Inclusão em Árvore AVL

```
Procedimento insere_AVL(x, pt, h)
   se pt = λ então inicio_no(pt); h ← "V"
   senão
       se (x = pt↑.chave) então PARE
       se (x < pt↑.chave) então
          insere_AVL(x, pt↑.esq,h)
          se h então
              caso pt↑.bal seja
              1 : pt↑.bal ← 0; h ← "F"
              0 : pt↑.bal ← -1
              -1: caso1(pt, h)    //Rebalanceamento
       senão
          insere_AVL(x, pt↑.dir,h)
          se h então
              caso pt↑.bal seja
              -1 : pt↑.bal ← 0; h ← "F"
               0 : pt↑.bal ← 1
               1 : caso2(pt, h)     //Rebalanceamento
```

Inclusão em Árvore AVL

Volta do Algoritmo Recalculando o Balanço

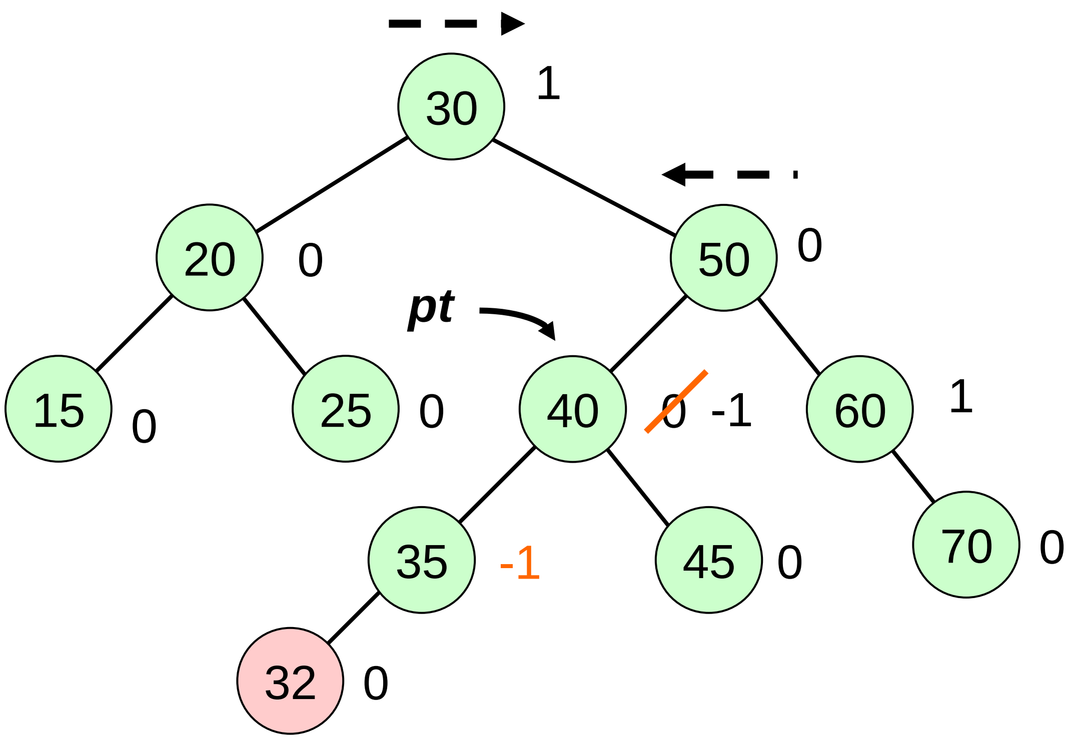


Inclusão em Árvore AVL

```
Procedimento insere_AVL(x, pt, h)
   se pt = λ então inicio_no(pt); h ← "V"
   senão
       se (x = pt↑.chave) então PARE
       se (x < pt↑.chave) então
          insere_AVL(x, pt↑.esq,h)
          se h então
              caso pt↑.bal seja
              1 : pt↑.bal ← 0; h ← "F"
              0 : pt↑.bal ← -1
              -1: caso1(pt, h)    //Rebalanceamento
       senão
          insere_AVL(x, pt↑.dir,h)
          se h então
              caso pt↑.bal seja
              -1 : pt↑.bal ← 0; h ← "F"
               0 : pt↑.bal ← 1
               1 : caso2(pt, h)     //Rebalanceamento
```

Inclusão em Árvore AVL

Volta do Algoritmo Recalculando o Balanço

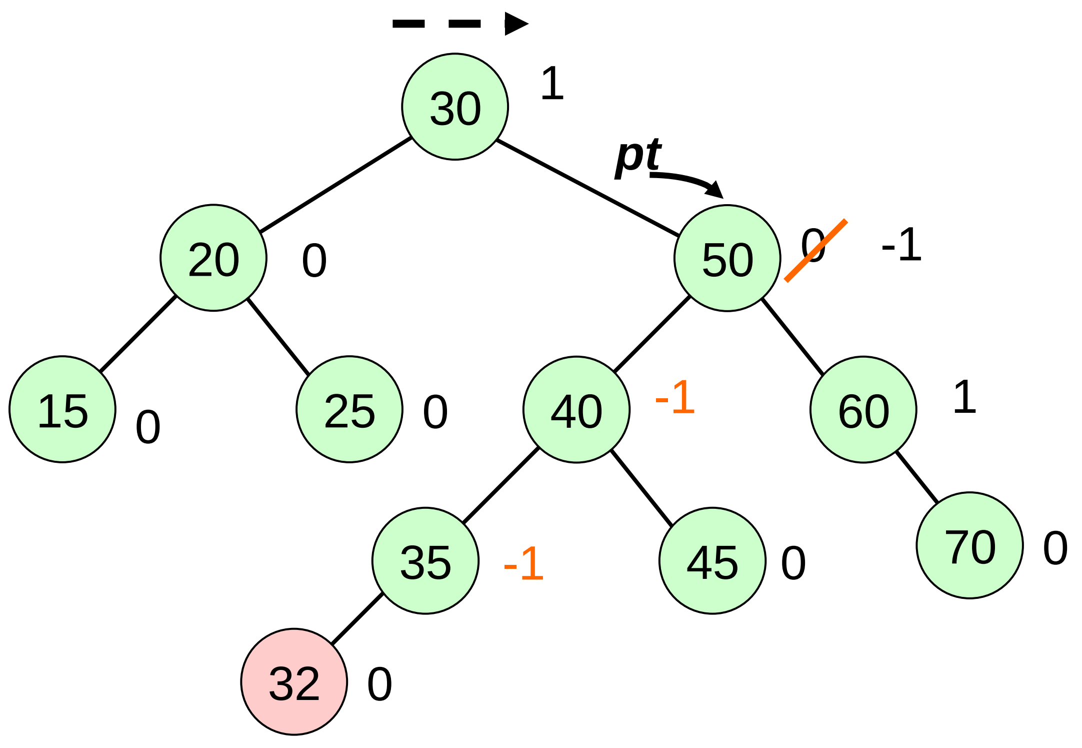


Inclusão em Árvore AVL

```
Procedimento insere_AVL(x, pt, h)
   se pt = λ então inicio_no(pt); h ← "V"
   senão
       se (x = pt↑.chave) então PARE
       se (x < pt↑.chave) então
          insere_AVL(x, pt↑.esq,h)
          se h então
              caso pt↑.bal seja
              1 : pt↑.bal ← 0; h ← "F"
              0 : pt↑.bal ← -1
              -1: caso1(pt, h)    //Rebalanceamento
       senão
          insere_AVL(x, pt↑.dir,h)
          se h então
              caso pt↑.bal seja
              -1 : pt↑.bal ← 0; h ← "F"
               0 : pt↑.bal ← 1
               1 : caso2(pt, h)     //Rebalanceamento
```

Inclusão em Árvore AVL

Volta do Algoritmo Recalculando o Balanço

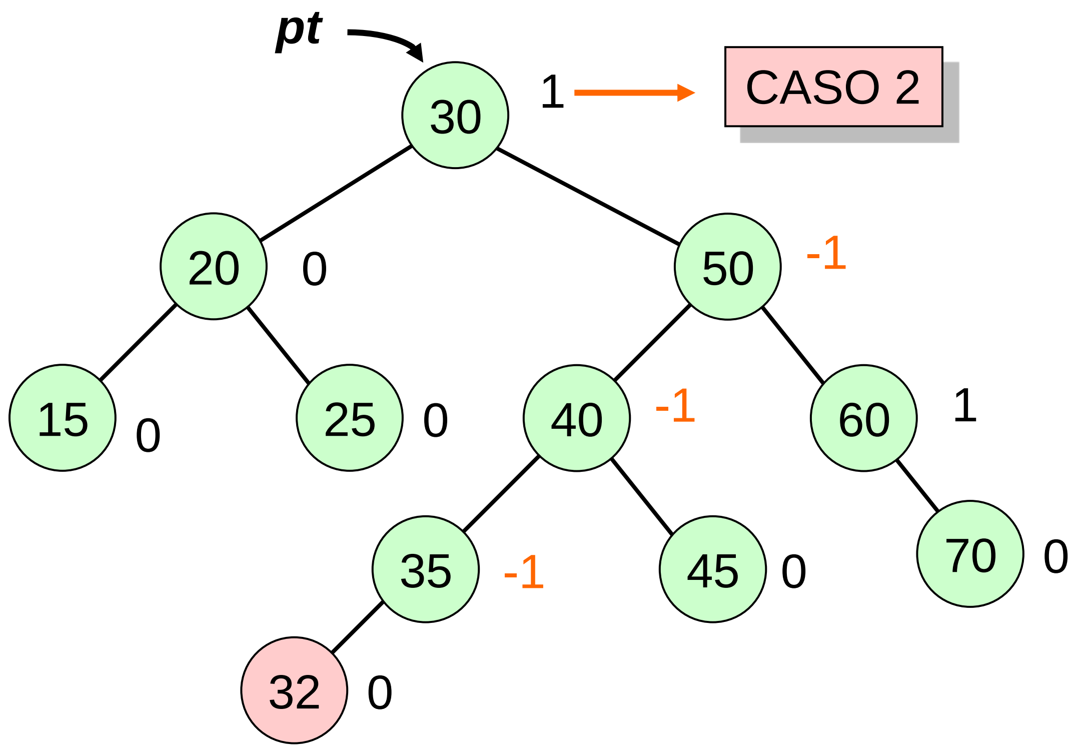


Inclusão em Árvore AVL

```
Procedimento insere_AVL(x, pt, h)
   se pt = λ então inicio_no(pt); h ← “V”
   senão
       se (x = pt↑.chave) então PARE
       se (x < pt↑.chave) então
          insere_AVL(x, pt↑.esq,h)
          se h então
              caso pt↑.bal seja
              1 : pt↑.bal ← 0; h ← “F”
              0 : pt↑.bal ← -1
              -1: caso1(pt, h)    //Rebalanceamento
       senão
          insere_AVL(x, pt↑.dir,h)
          se h então
              caso pt↑.bal seja
              -1 : pt↑.bal ← 0; h ← “F”
               0 : pt↑.bal ← 1
               1 : caso2(pt, h)     //Rebalanceamento
```

Inclusão em Árvore AVL

Volta do Algoritmo Recalculando o Balanço



Inclusão em Árvore AVL

Procedimento caso2(*pt*,*h*)

  ptu* ← *pt*↑*.dir

  se *ptu*↑*.bal* = 1 então

    pt*↑*.dir* ← *ptu*↑*.esq*; *ptu*↑*.esq* ← *pt

    pt*↑*.bal* ← 0; *pt* ← *ptu

Rotação Esquerda

  senão

Rot Dup Esq

    ptv* ← *ptu*↑*.esq

    ptu*↑*.esq* ← *ptv*↑*.dir* ; *ptv*↑*.dir* ← *ptu

    pt*↑*.dir*← *ptv*↑*.esq* ; *ptv*↑*.esq* ← *pt

    se (*ptv*↑*.bal* = 1) então *pt*↑*.bal* ← -1 senão *pt*↑*.bal* ← 0

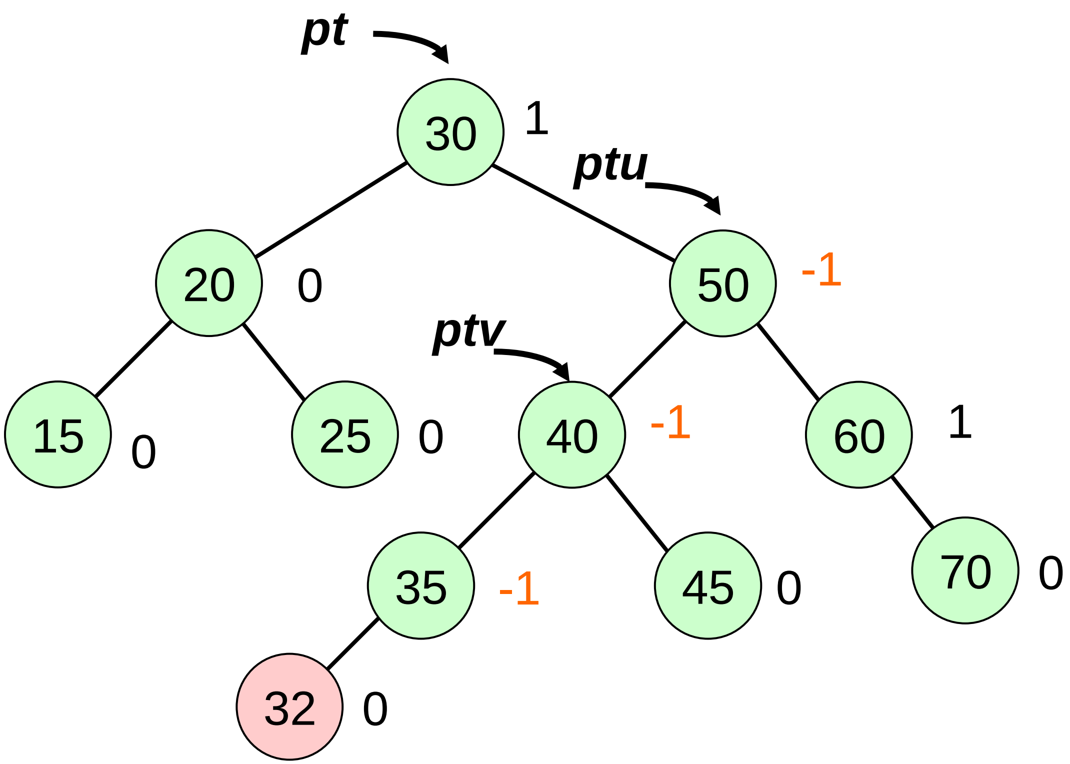
    se (*ptv*↑*.bal* = -1) então *ptu*↑*.bal* ← 1 senão *ptu*↑*.bal* ← 0

    pt* ← *ptv

  ***pt*↑*.bal* ← 0 ; *h* ← "F"**

Inclusão em Árvore AVL

Balanceamento: Rotação Dupla Esquerda

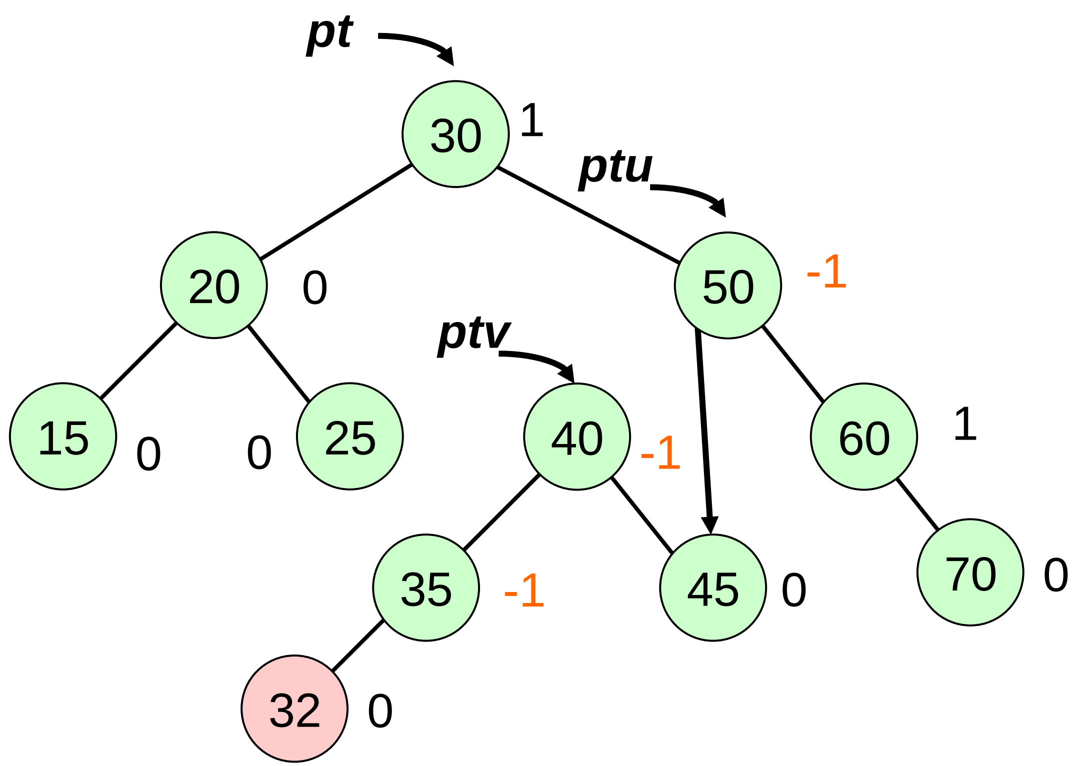


Procedimento caso2(*pt*,*h*)

** *ptu* ← *pt*↑.*dir***
** se *ptu*↑.*bal* = 1 então**
**   *pt*↑.*dir* ← *ptu*↑.*esq; ptu*↑.*esq* ← *pt***
**   *pt*↑.*bal* ← 0; *pt* ← *ptu***
** senão**
**   *ptv* ← *ptu*↑.*esq***
**   *ptu*↑.*esq* ← *ptv*↑.*dir* ; *ptv*↑.*dir* ← *ptu***
**   *pt*↑.*dir*← *ptv*↑.*esq* ; *ptv*↑.*esq* ← *pt***
**   se (*ptv*↑.*bal* = 1) então *pt*↑.*bal* ← -1 senão *pt*↑.*bal* ← 0**
**   se (*ptv*↑.*bal* = -1) então *ptu*↑.*bal* ← 1 senão *ptu*↑.*bal* ← 0**
**   *pt* ← *ptv***
** *pt*↑.*bal* ← 0 ; *h* ← "F"**

Inclusão em Árvore AVL

Balanceamento: Rotação Dupla Esquerda



Procedimento caso2(*pt*,*h*)

ptu ← _pt_↑._dir_

se _ptu_↑._bal_ = 1 então

_pt_↑._dir_ ← _ptu_↑._esq_; _ptu_↑._esq_ ← _pt_

_pt_↑._bal_ ← 0; _pt_ ← _ptu_

senão

ptv ← _ptu_↑._esq_

ptu↑._esq_ ← _ptv_↑._dir_ ; _ptv_↑._dir_ ← _ptu_

_pt_↑._dir_ ← _ptv_↑._esq_ ; _ptv_↑._esq_ ← _pt_

se (_ptv_↑._bal_ = 1) então _pt_↑._bal_ ← -1 senão _pt_↑._bal_ ← 0

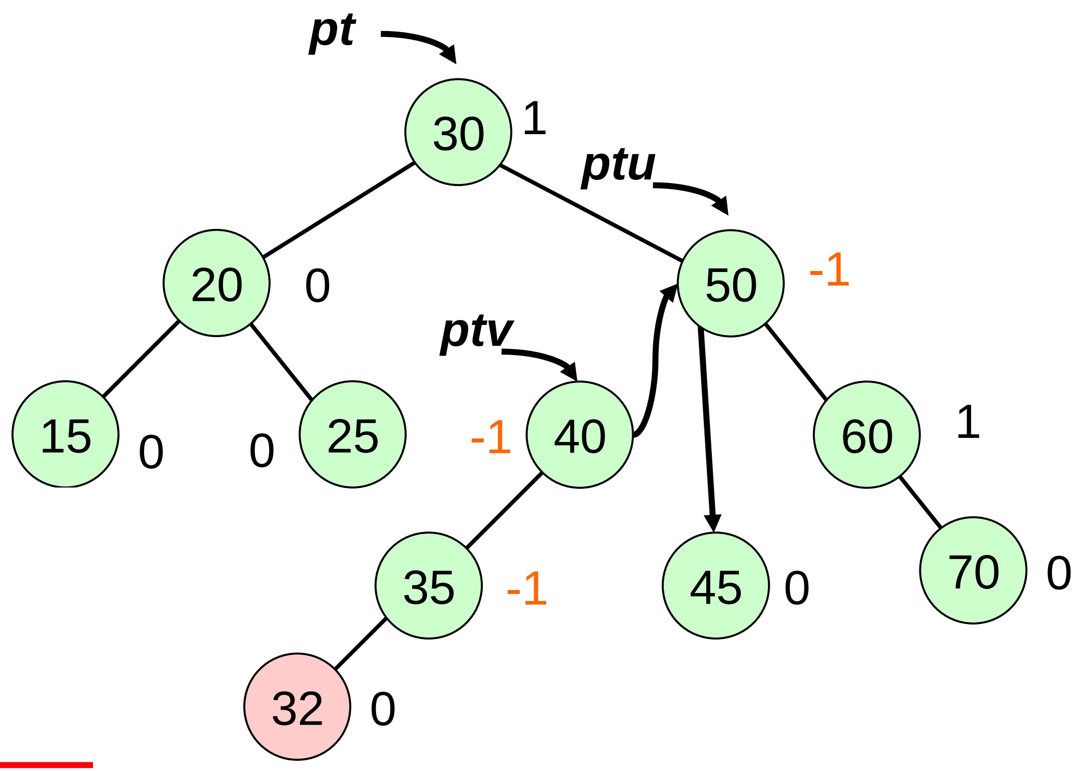
se (_ptv_↑._bal_ = -1) então _ptu_↑._bal_ ← 1 senão _ptu_↑._bal_ ← 0

_pt_ ← _ptv_

_pt_↑._bal_ ← 0 ; _h_ ← “F”

Inclusão em Árvore AVL

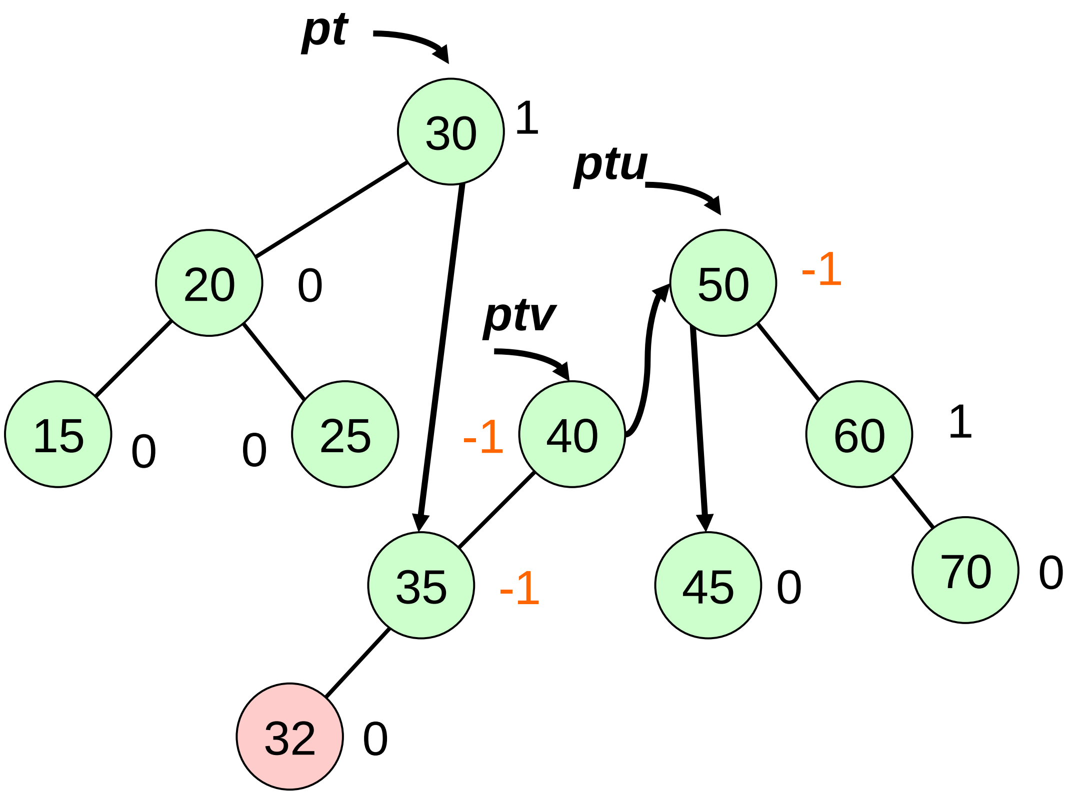
Balanceamento: Rotação Dupla Esquerda



Procedimento caso2(*pt*,*h*)
  ptu* ← *pt*↑.*dir
  se *ptu*↑.*bal* = 1 então
    pt*↑.*dir* ← *ptu*↑.*esq*; *ptu*↑.*esq* ← *pt
    pt*↑.*bal* ← 0; *pt* ← *ptu
  senão
    ptv* ← *ptu*↑.*esq
    ptu*↑.*esq* ← *ptv*↑.*dir* ; *ptv*↑.*dir* ← *ptu
    pt*↑.*dir*← *ptv*↑.*esq* ; *ptv*↑.*esq* ← *pt
    se (*ptv*↑.*bal* = 1) então *pt*↑.*bal* ← -1 senão *pt*↑.*bal* ← 0
    se (*ptv*↑.*bal* = -1) então *ptu*↑.*bal* ← 1 senão *ptu*↑.*bal* ← 0
    pt* ← *ptv
  ***pt*↑.*bal* ← 0 ; *h* ← "F"**

Inclusão em Árvore AVL

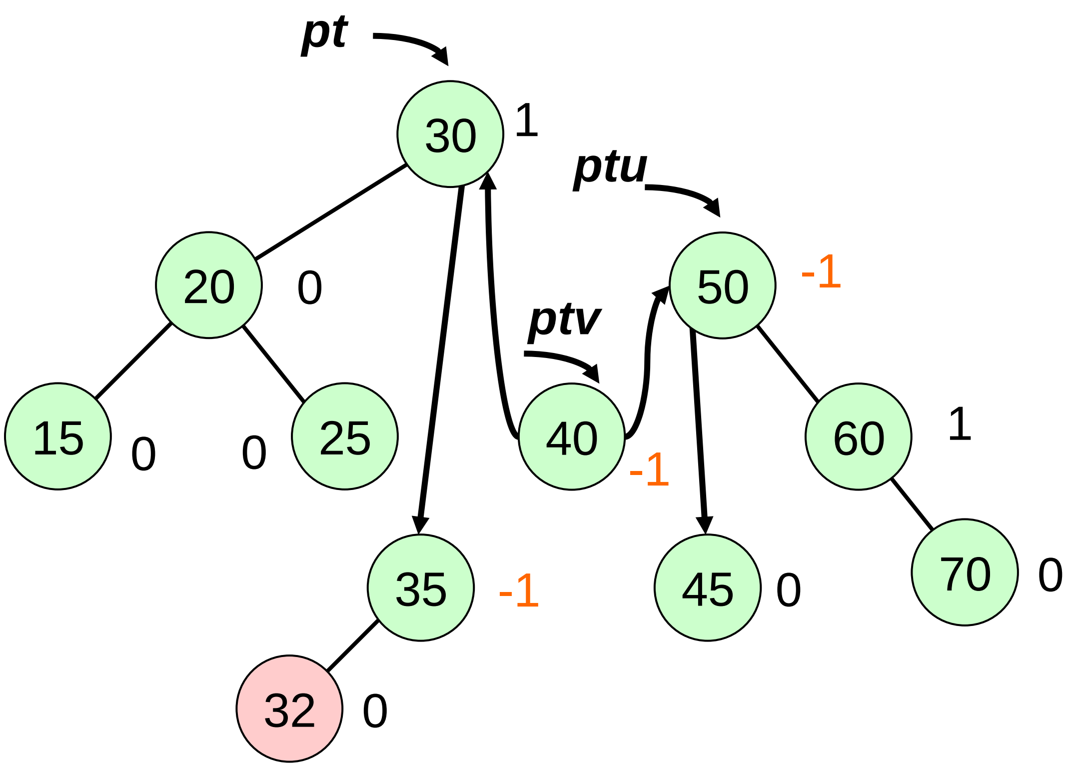
Balanceamento: Rotação Dupla Esquerda



Procedimento caso2(*pt*,*h*)
  ptu **←** ***pt*↑.*dir***
  se *ptu*↑.*bal* = 1 então
    pt*↑.*dir* ← *ptu*↑.*esq*; *ptu*↑.*esq* ← *pt
    pt*↑.*bal* ← 0; *pt* ← *ptu
  senão
    ptv* ← *ptu*↑.*esq
    ptu*↑.*esq* ← *ptv*↑.*dir* ; *ptv*↑.*dir* ← *ptu
    pt*↑.*dir*← *ptv*↑.*esq* ; *ptv*↑.*esq* ← *pt
    se (*ptv*↑.*bal* = 1) então *pt*↑.*bal* ← -1 senão *pt*↑.*bal* ← 0
    se (*ptv*↑.*bal* = -1) então *ptu*↑.*bal* ← 1 senão *ptu*↑.*bal* ← 0
    pt* ← *ptv
  ***pt*↑.*bal* ← 0 ; *h* ← "F"**

Inclusão em Árvore AVL

Balanceamento: Rotação Dupla Esquerda



Procedimento caso2(*pt*,*h*)

ptu ← _pt_↑._dir_

se _ptu_↑._bal_ = 1 então

_pt_↑._dir_ ← _ptu_↑._esq_; _ptu_↑._esq_ ← _pt_

_pt_↑._bal_ ← 0; _pt_ ← _ptu_

senão

ptv ← _ptu_↑._esq_

ptu↑._esq_ ← _ptv_↑._dir_ ; _ptv_↑._dir_ ← _ptu_

_pt_↑._dir_ ← _ptv_↑._esq_ ; _ptv_↑._esq_ ← _pt_

se (_ptv_↑._bal_ = 1) então _pt_↑._bal_ ← -1 senão _pt_↑._bal_ ← 0

se (_ptv_↑._bal_ = -1) então _ptu_↑._bal_ ← 1 senão _ptu_↑._bal_ ← 0

_pt_ ← _ptv_

_pt_↑._bal_ ← 0 ; _h_ ← "F"

Redesenhando!!!

Inclusão em Árvore AVL

Procedimento caso2(*pt*,*h*)

$ptu \leftarrow pt\uparrow.dir$

se $ptu\uparrow.bal$ = 1 então

$pt\uparrow.dir \leftarrow ptu\uparrow.esq$; $ptu\uparrow.esq \leftarrow pt$

$pt\uparrow.bal \leftarrow 0$; $pt \leftarrow ptu$

Rotação Esquerda

senão

$ptv \leftarrow ptu\uparrow.esq$

$ptu\uparrow.esq \leftarrow ptv\uparrow.dir$; $ptv\uparrow.dir \leftarrow ptu$

$pt\uparrow.dir \leftarrow ptv\uparrow.esq$; $ptv\uparrow.esq \leftarrow pt$

se ($ptv\uparrow.bal$ = 1) então $pt\uparrow.bal \leftarrow -1$ senão $pt\uparrow.bal \leftarrow 0$

se ($ptv\uparrow.bal$ = -1) então $ptu\uparrow.bal \leftarrow 1$ senão $ptu\uparrow.bal \leftarrow 0$

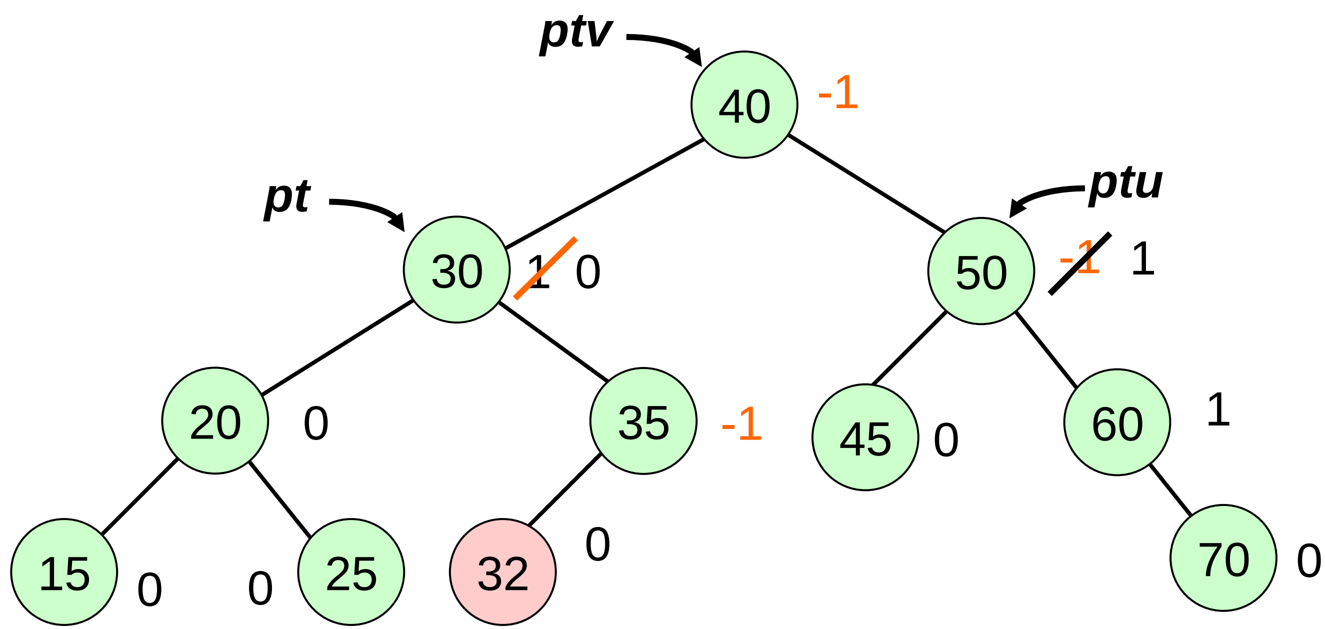
$pt \leftarrow ptv$

Rot Dup Esq

$pt\uparrow.bal \leftarrow 0$; $h \leftarrow$ “F”

Inclusão em Árvore AVL

Balanceamento: Rotação Dupla Esquerda

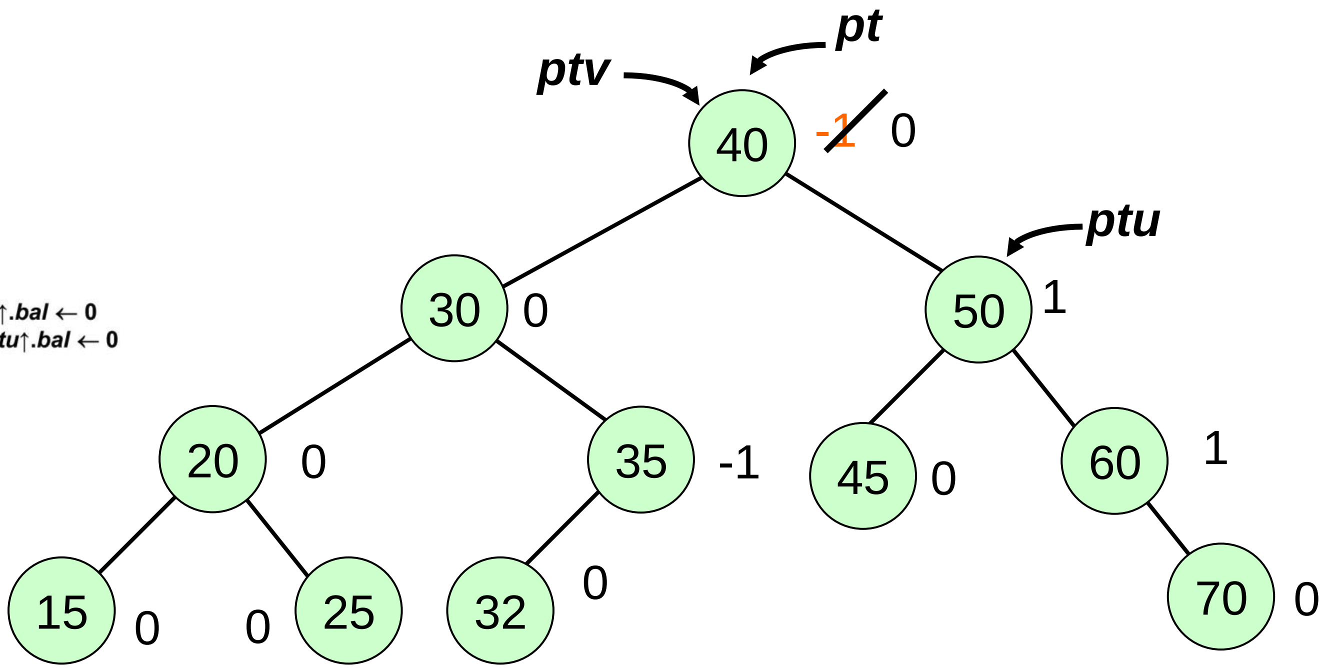



```
Procedimento caso2(pt,h)
   ptu ← pt↑.dir
   se ptu↑.bal = 1 então
      pt↑.dir ← ptu↑.esq; ptu↑.esq ← pt
      pt↑.bal ← 0; pt ← ptu
   senão
      ptv ← ptu↑.esq
      ptu↑.esq ← ptv↑.dir ; ptv↑.dir ← ptu
      pt↑.dir← ptv↑.esq ; ptv↑.esq ← pt
      se (ptv↑.bal =  1) então pt↑.bal ← -1 senão pt↑.bal ← 0
      se (ptv↑.bal = -1) então ptu↑.bal ← 1 senão ptu↑.bal ← 0
      pt ← ptv
   pt↑.bal ← 0 ; h ← "F"
```

Inclusão em Árvore AVL

Balanceamento: Rotação Dupla Esquerda

```
Procedimento caso2(pt,h)
   ptu ← pt↑.dir
   se ptu↑.bal = 1 então
        pt↑.dir ← ptu↑.esq; ptu↑.esq ← pt
        pt↑.bal ← 0; pt ← ptu
   senão
       ptv ←  ptu↑.esq
       ptu↑.esq ← ptv↑.dir ; ptv↑.dir ← ptu
       pt↑.dir← ptv↑.esq ; ptv↑.esq ← pt
       se (ptv↑.bal =  1) então pt↑.bal ← -1 senão pt↑.bal ← 0
       se (ptv↑.bal = -1) então ptu↑.bal ← 1 senão ptu↑.bal ← 0
       pt ←  ptv
   pt↑.bal ← 0 ; h ← "F"
```



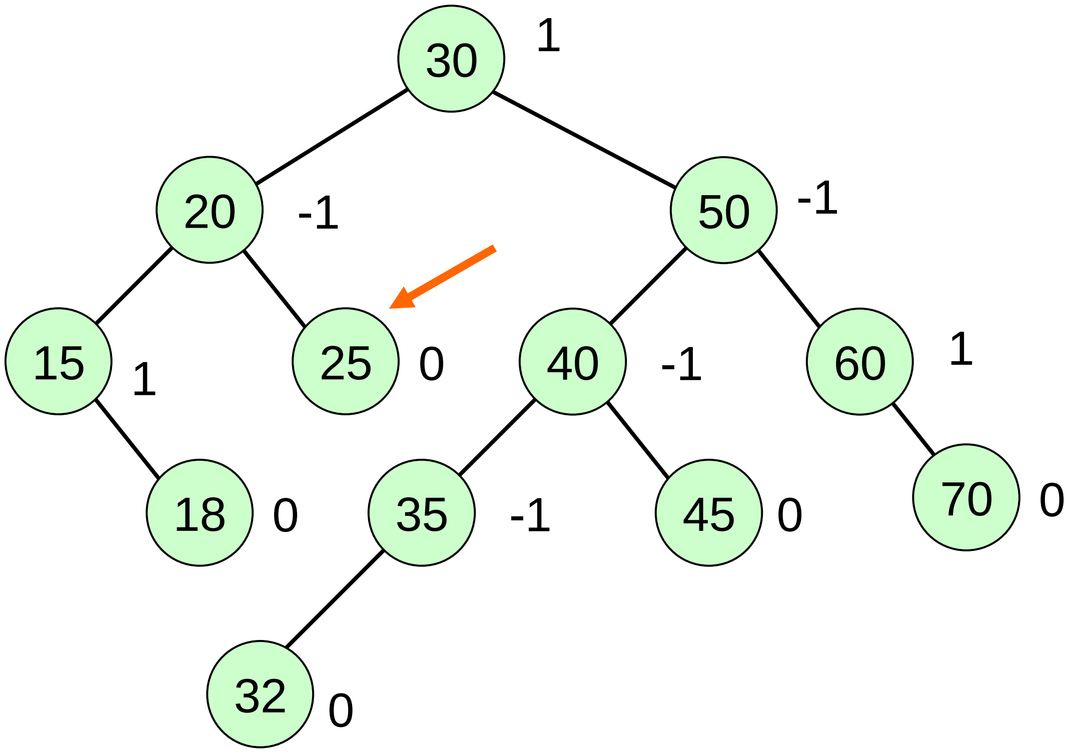
h ←"F"

Exercício

1. Mostrar a árvore AVL obtida pela sequência de inserções das chaves: 20, 18, 16, 15, 2, 17, 19 nesta ordem.

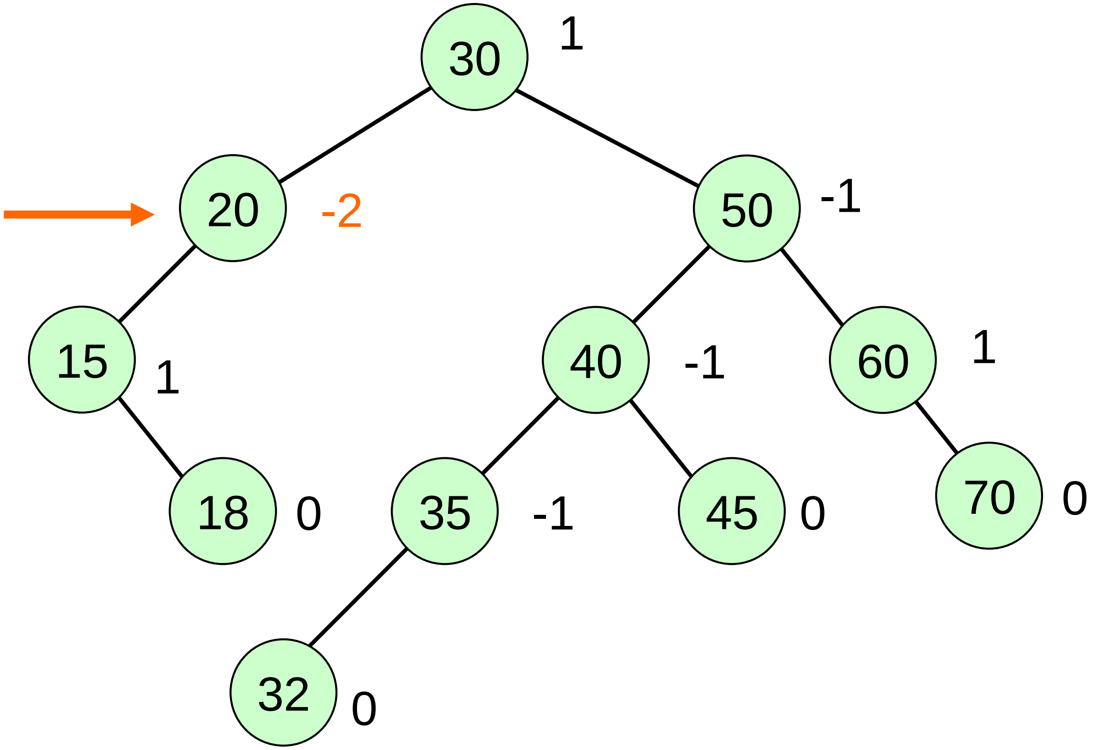
2. Escreva um algoritmo que receba como entrada uma árvore binária de busca e retorne “verdadeiro” se a mesma é uma árvore AVL e “falso”, caso contrário.
Qual o tempo de processamento do seu algoritmo?

Remoção em Árvore AVL



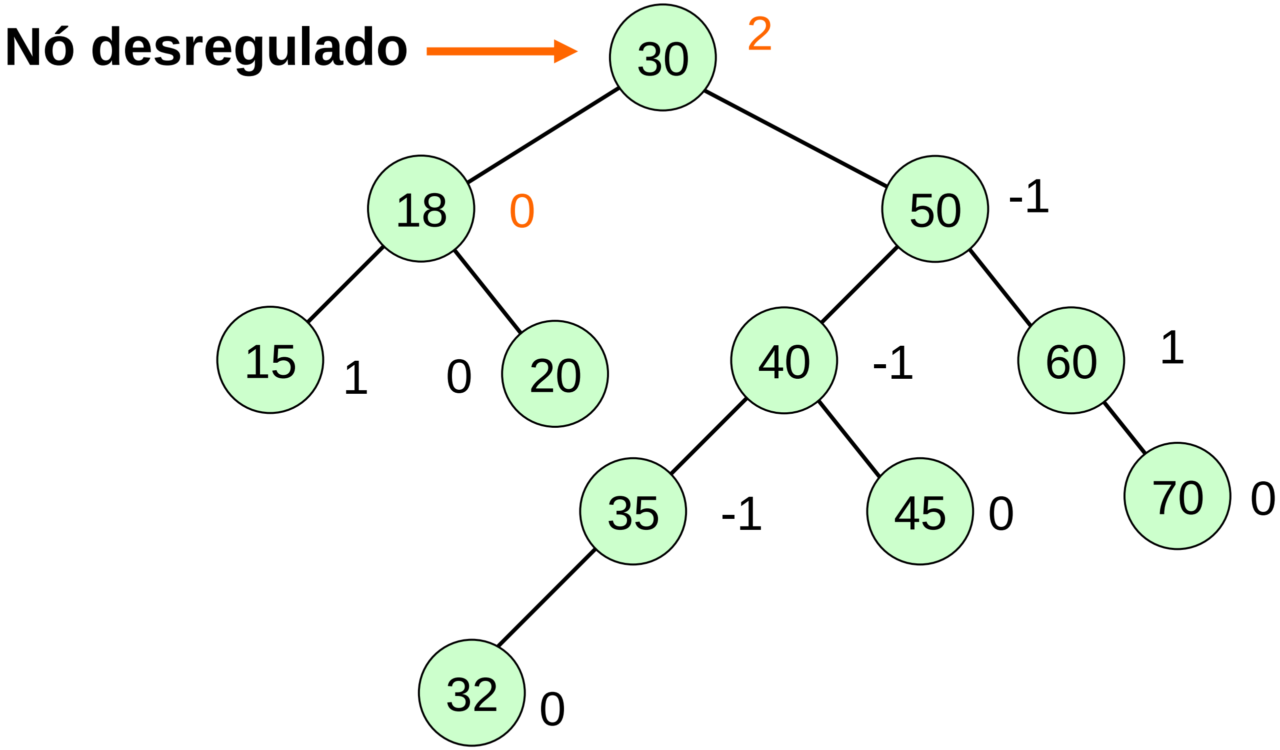
Remoção em Árvore AVL

Nó desregulado



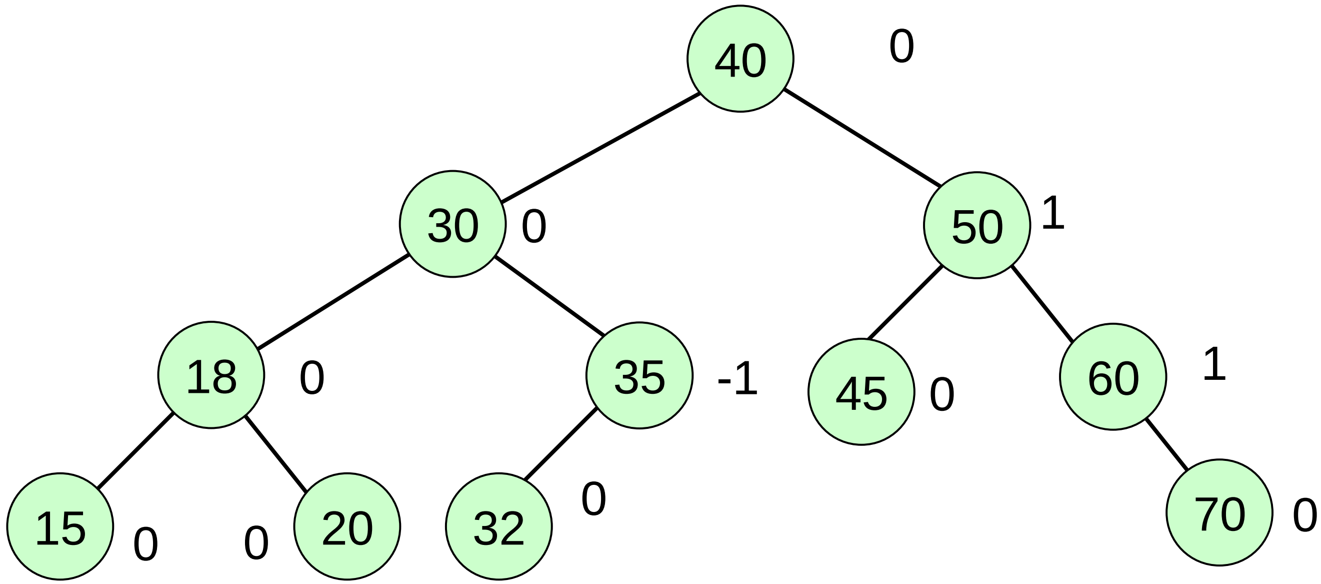
Aplicar: Rotação Dupla Direita

Remoção em Árvore AVL



Aplicar: Rotação Dupla Esquerda

Remoção em Árvore AVL



Exercícios de Árvore AVL

1. **Provar ou dar um contra-exemplo:**
   Toda árvore estritamente binária é AVL.

2. **Desenhar a árvore de Fibonacci T_5**

3. **Mostrar que a rotação dupla direita pode ser obtida por uma rotação esquerda seguida por uma rotação direita.**

4. **Detalhar o algoritmo de exclusão em árvore AVL.**