UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Árvores Balanceadas

Prof. Vinicius Ramos

Árvores Balanceadas

- É fundamental pensarmos no "custo" de acesso a um elemento desejado
- As árvores binárias de busca ou de partilha são exemplos de bons custos de acesso quando temos um número fixo de elementos
- Para tanto, foram desenvolvidas árvores que mantêm os custos na mesma ordem de grandeza de uma árvore ótima, ou seja, $O(\log n)$

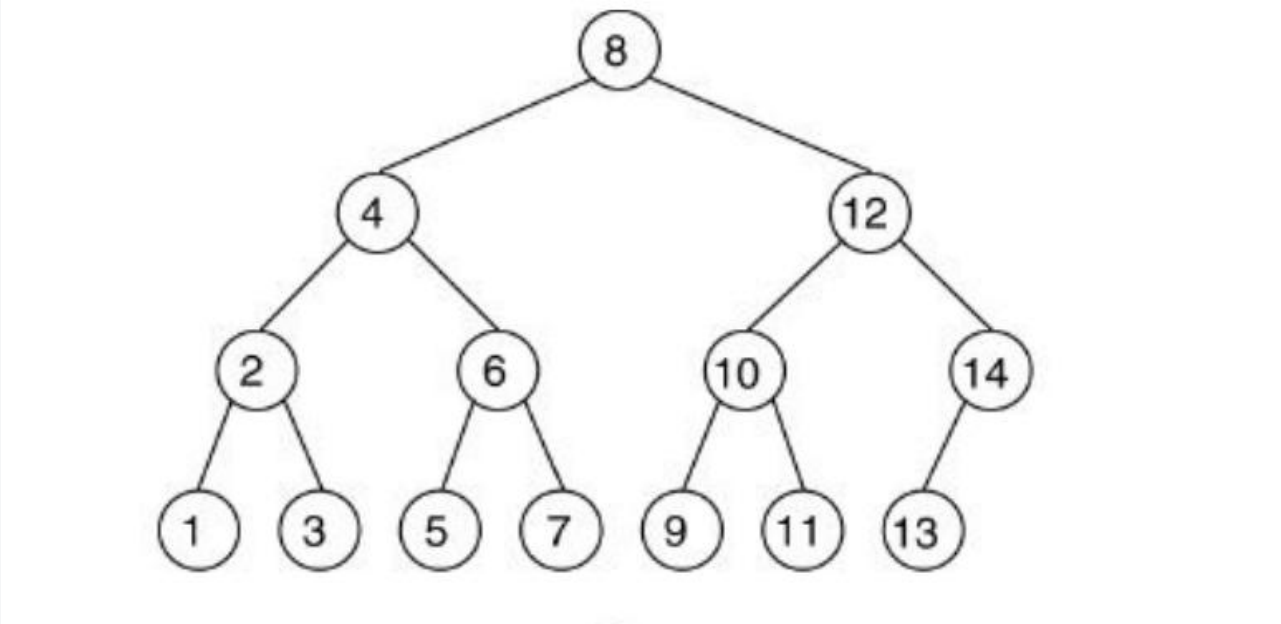
Árvores Balanceadas

- A ideia é manter os custos de acesso na mesma ordem de grandeza
- Inclusive, após inclusões e remoções
- Assim, periodicamente, essa estrutura precisará ser modificada para moldar-se aos novos dados
- O custo dessa manutenção, entretanto, é da ordem de O(log n)
- Essa estrutura é chamada de árvore balanceada

Balanceamento

- As árvores completas minimizam o número de comparações efetuadas no pior caso (para uma busca com chaves de probabilidades de ocorrência idênticas)
- Entretanto, no pior dos casos, o custo de manter uma árvore completa pode ser na ordem de O(n) passos

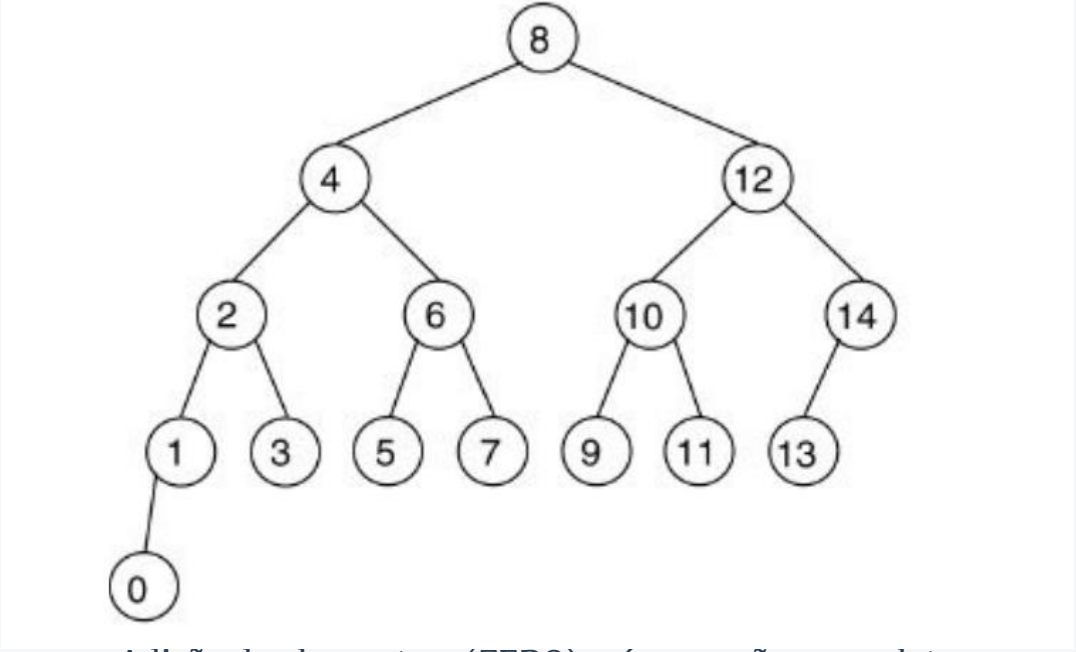
Árvore Completa - Exemplo



Árvore completa com apenas um espaço possível para inserção

Extraído de Szwarcfiter & Markezon, 2010

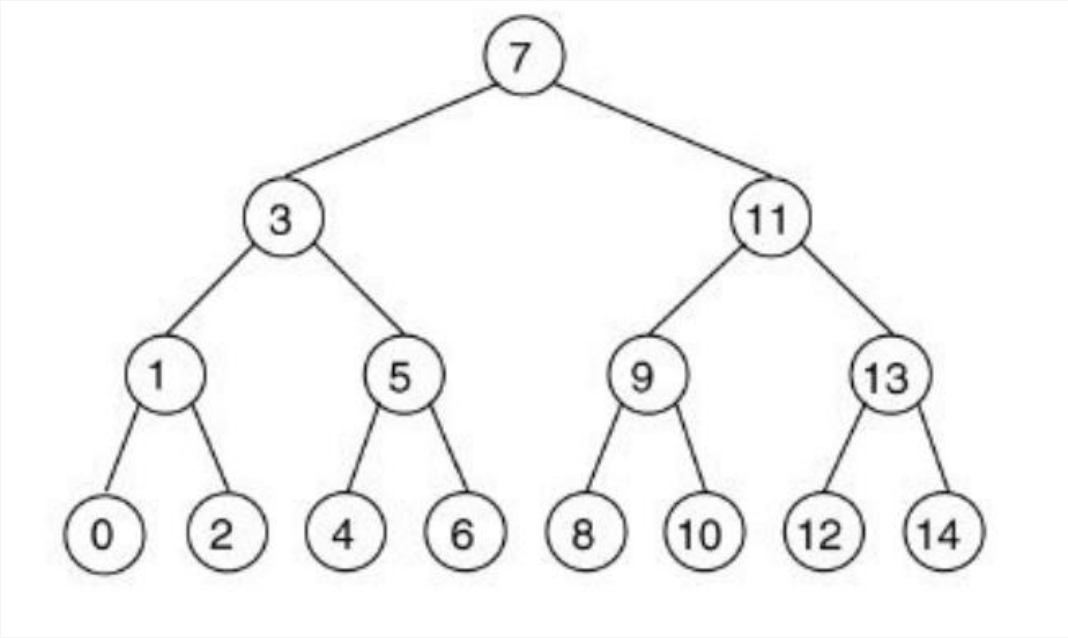
Árvore Completa - Exemplo



Adição do elemento 0 (ZERO) – árvore não-completa

Extraído de Szwarcfiter & Markezon, 2010

Árvore Completa - Exemplo



Árvore completa reestabelecida – percorrimento de toda árvore Ω(n) passos
Ω(n) – limite inferior

Extraído de Szwarcfiter & Markezon, 2010

Árvores Balanceadas

- A ideia é que a altura seja na mesma ordem de grandeza que a de uma árvore completa com o mesmo número de nós
  - Altura igual a O(log n)
- Esta propriedade deve se estender a todas as subárvores: cada subárvore com *m* nós deve ter altura O(log m)
- Assim, teremos uma árvore balanceada

Referências

- GOODRICH, M. T., TAMASSIA, R. & GOLDWASSER, M. H.. Data Structures and Algorithms in Java. 6ª Edição. 2014
- SZWARCFITER, J. L., MARKEZON, L.. Estruturas de Dados e Seus Algoritmos. 3 Edição. 2015.

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Obrigado!

Prof. Vinicius Ramos
https://viniciusramos.pro.br
v.ramos@ufsc.br

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Árvores AVL

Prof. Vinicius Ramos

Árvores AVL

Prof. Vinicius Ramos

Árvores AVL

- Para qualquer nó v de T, as alturas (h) de suas duas subárvores, esquerda e direita, diferem em módulo de até uma unidade
- Chamaremos de balanceamento

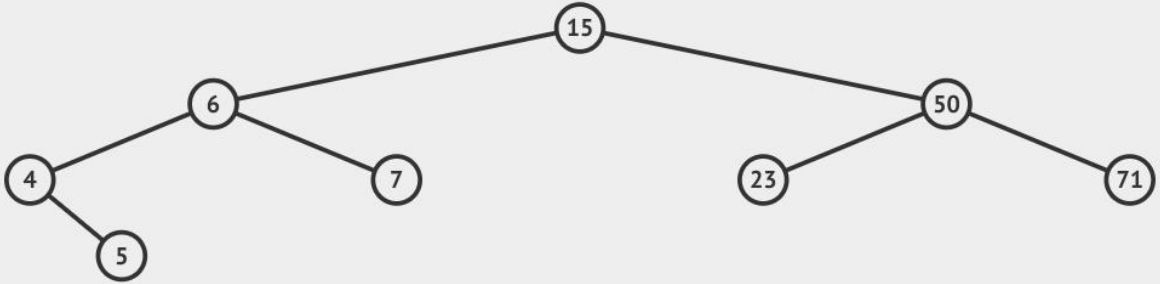
$$-1 >= bal <- h_{esq} - h_{dir} <= 1$$

$$bal -> \{-1, 0, 1\}$$

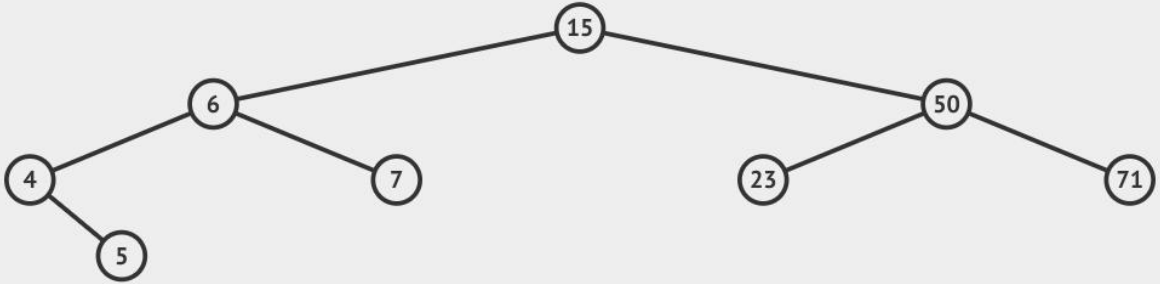
Árvores AVL

- Quando isso acontece, dizemos que v é um nó está regulado
- Em contrapartida, um nó que **não** satisfaça essa condição de altura é dito desregulado
- Naturalmente, toda árvore completa é AVL, mas a recíproca não é verdadeira, necessariamente.

Árvores AVL

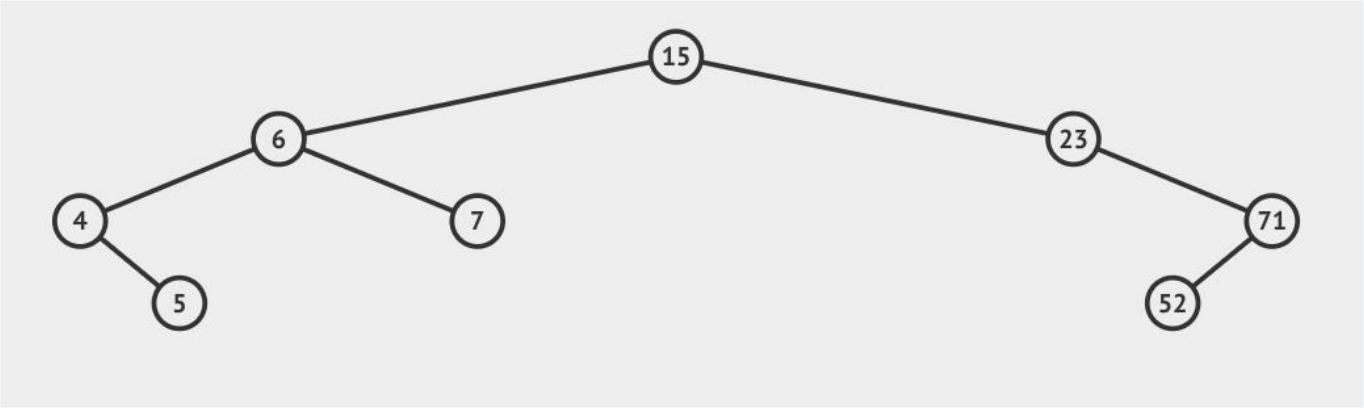


Árvores AVL



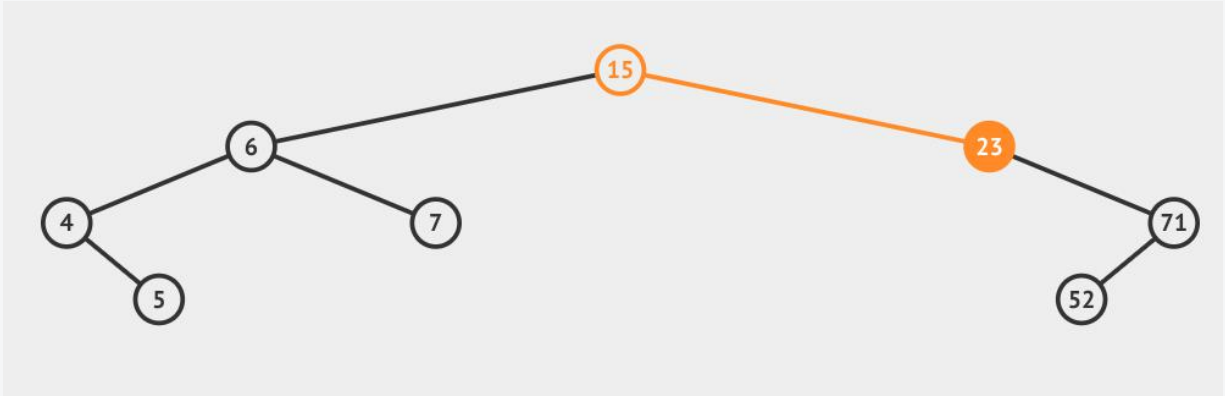
Árvores NÃO-AVL

- **NÃO-AVL**



Árvores NÃO-AVL

- **NÃO-AVL**



Árvores AVL

- Vejamos alguns exemplos em:
  - https://visualgo.net/pt/avl

Referências

- SZWARCFITER, J. L., MARKEZON, L.. Estruturas de Dados e Seus Algoritmos. 3 Edição. 2015.

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Obrigado!

Prof. Vinicius Ramos
https://viniciusramos.pro.br
v.ramos@ufsc.br

Árvores AVL

* Rotações Simples

Prof. Vinicius Ramos

Árvores AVL

Inclusão

- Quando incluímos elementos, é necessário verificar o balanceamento da subárvore envolvida nessa inclusão e os seus ancestrais também
- A ideia é poder verificar, após a inclusão, se um desses nós está desregulado
- Em caso positivo, precisaremos aplicar transformações para torná-lo regulado novamente

Árvores AVL

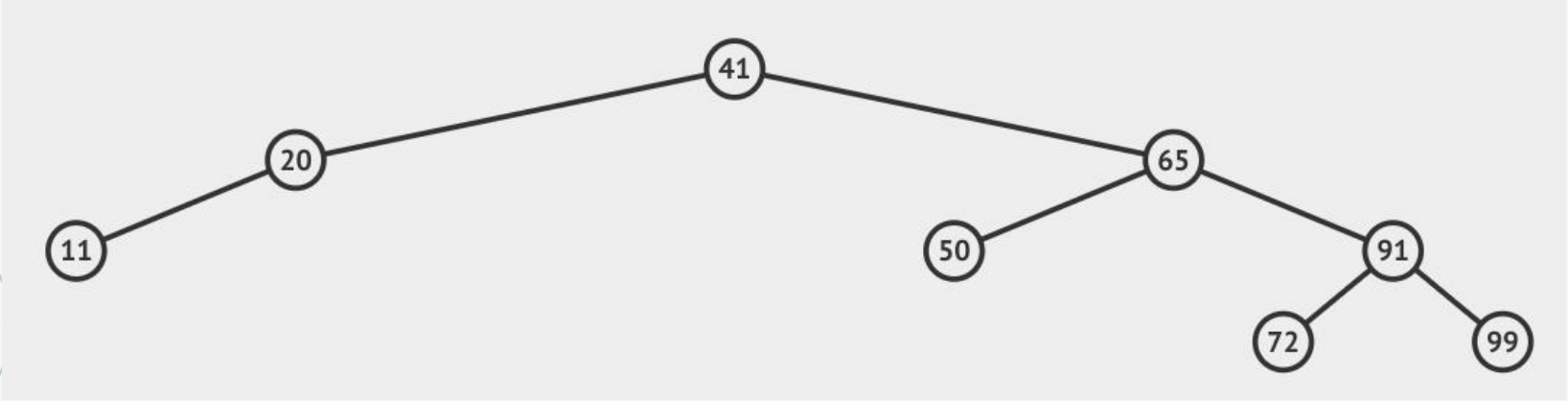
Rotações/Transformações

- São quatro rotações/transformações:
  - Simples Direita
  - Simples Esquerda
  - Dupla Direita
  - Dupla Esquerda

Árvores AVL

Rotações/Transformações

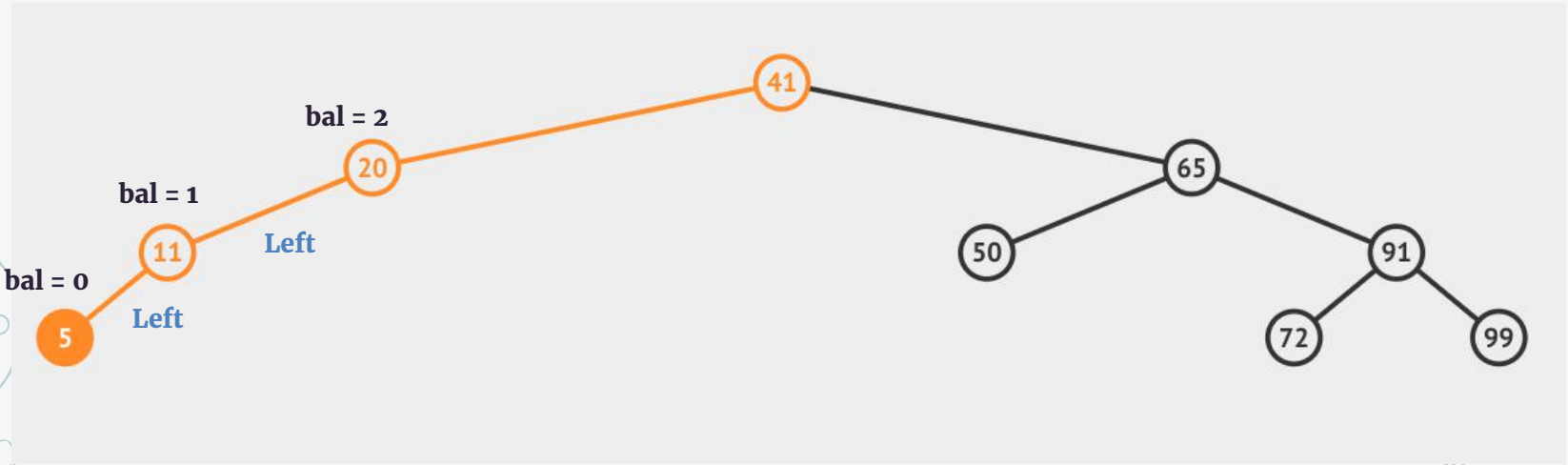
- Simples Direita (LL):



Árvores AVL

Rotações/Transformações

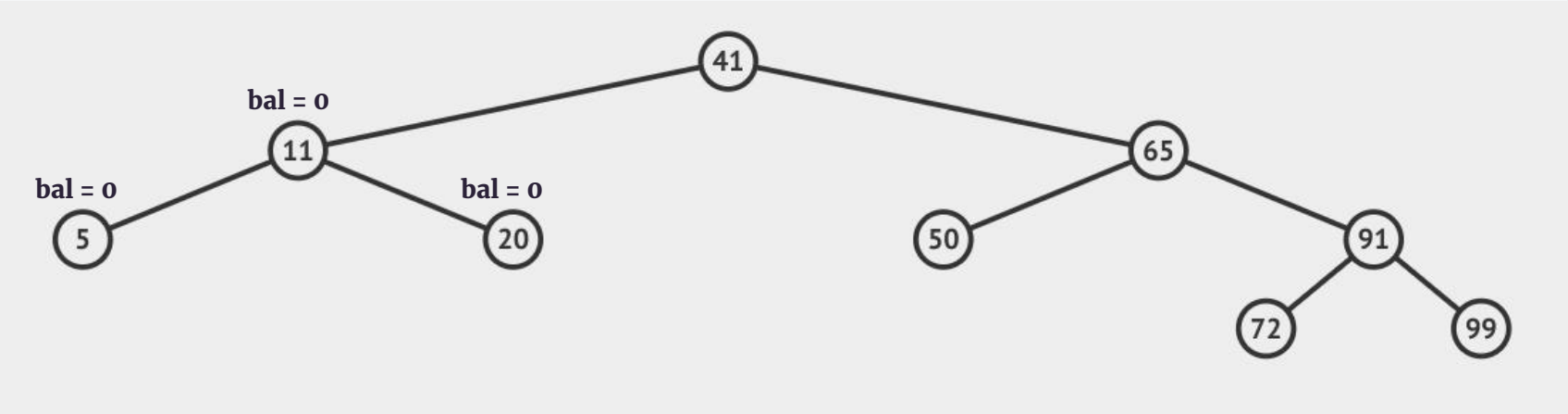
- Simples Direita (**LL**):



Árvores AVL

Rotações/Transformações

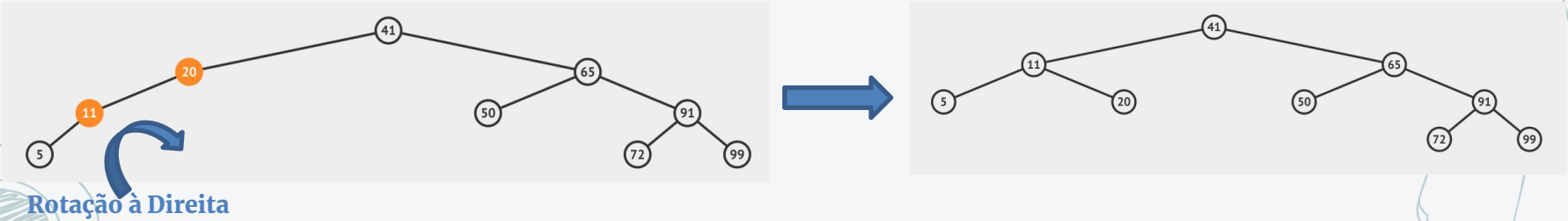
- Simples Direita (LL):



Árvores AVL

Rotações/Transformações

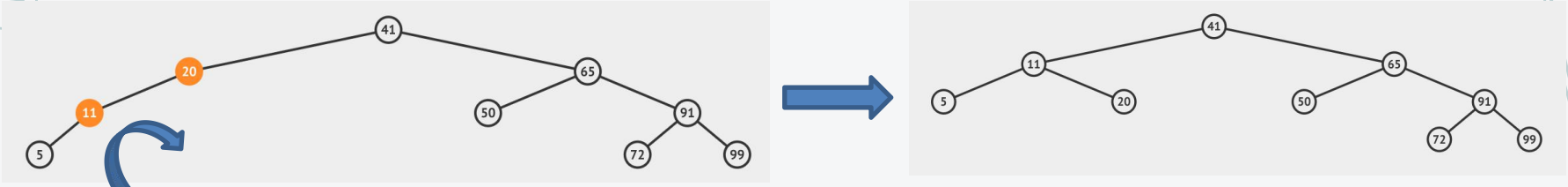
- Simples Direita (LL):



Árvores AVL

Rotações/Transformações

- Simples Direita (LL):

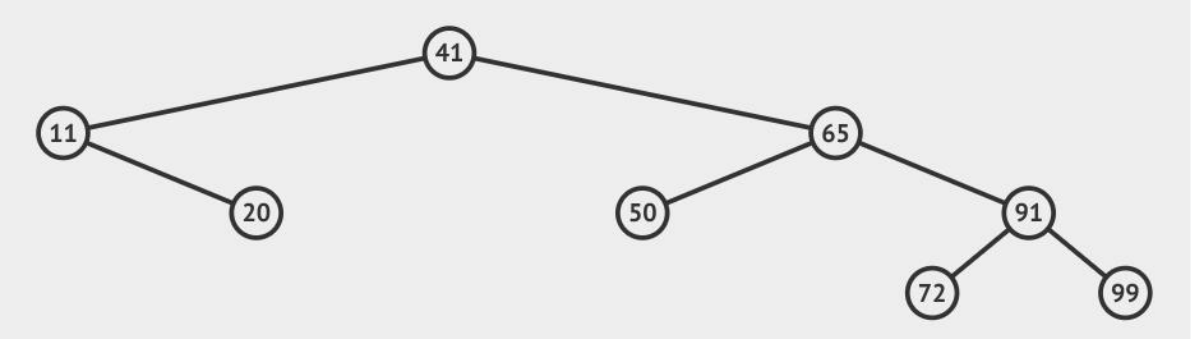



```
202     NodoArvoreAvl rotacaoDireita(NodoArvoreAvl a){
203         NodoArvoreAvl aux = a.nodoEsquerda;
204         a.nodoEsquerda = aux.nodoDireita;
205         aux.nodoDireita = a;
206         return aux;
207     }
```

Árvores AVL

Rotações/Transformações

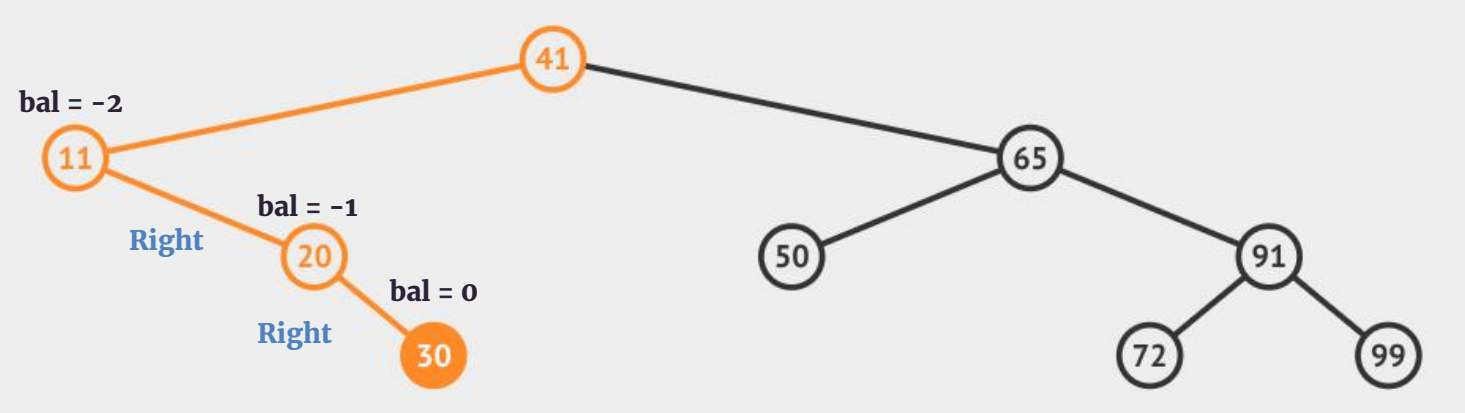
- Simples Esquerda (RR):



Árvores AVL

Rotações/Transformações

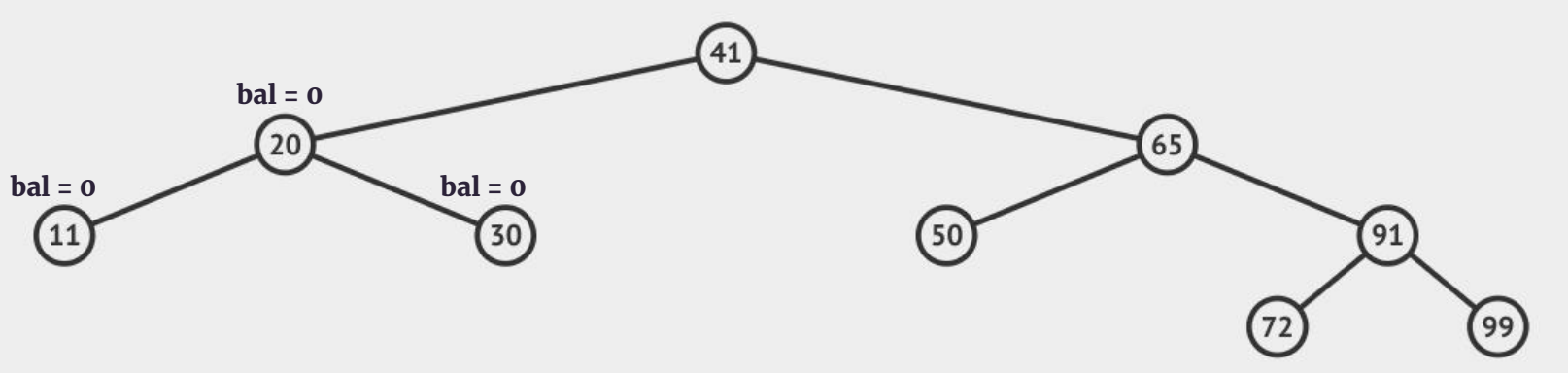
- Simples Esquerda (**RR**):



Árvores AVL

Rotações/Transformações

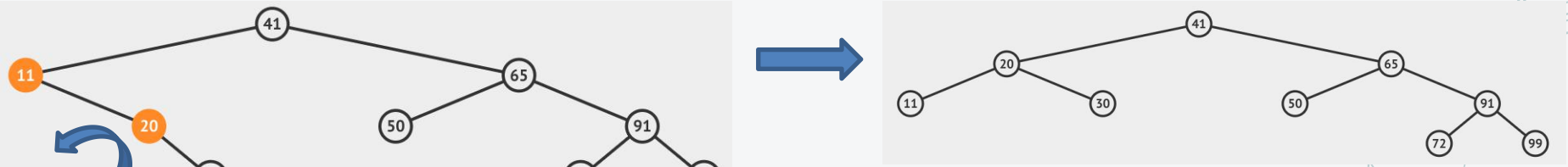
- Simples Esquerda (RR):



Árvores AVL

Rotações/Transformações

- Simples Esquerda (RR):

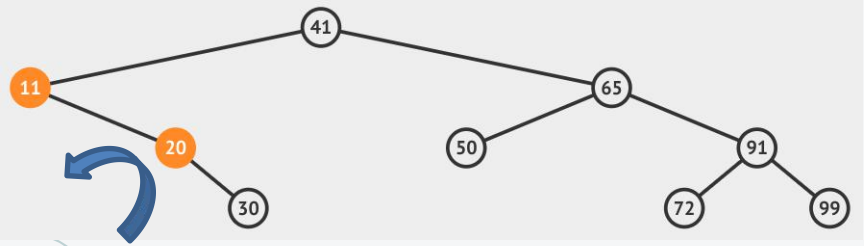


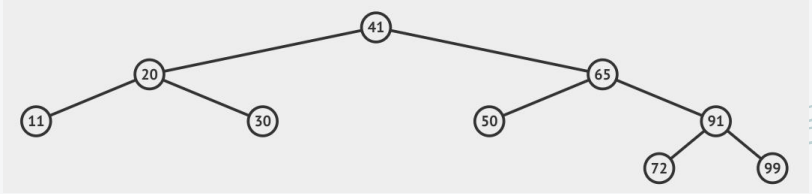
Rotação à Esquerda

Árvores AVL

Rotações/Transformações

- Simples Esquerda (RR):





Rotação à Esquerda

```
100    NodoArvoreAVL rotacaoEsquerda(NodoArvoreAVL a){
101        NodoArvoreAVL aux = a.direita;
102        a.direita = aux.esquerda;
103        aux.esquerda = a;
104        return aux;
105    }
```

Árvores AVL

- Façam os testes em:
  - https://visualgo.net/pt/avl

Referências

- SZWARCFITER, J. L., MARKEZON, L.. Estruturas de Dados e Seus Algoritmos. 3 Edição. 2015.

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Obrigado!

Prof. Vinicius Ramos
https://viniciusramos.pro.br
v.ramos@ufsc.br

Árvores AVL

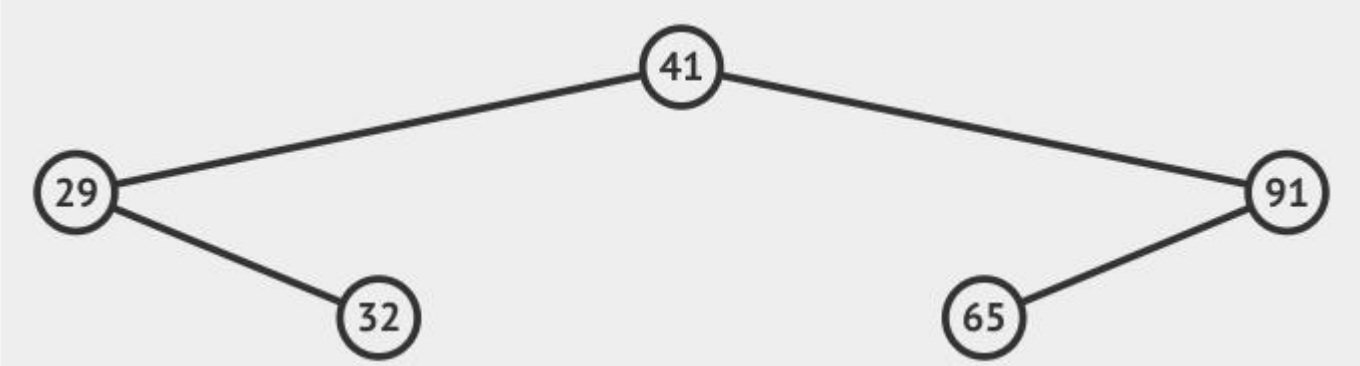
* Rotações Duplas

Prof. Vinicius Ramos

Árvores AVL

Rotações/Transformações

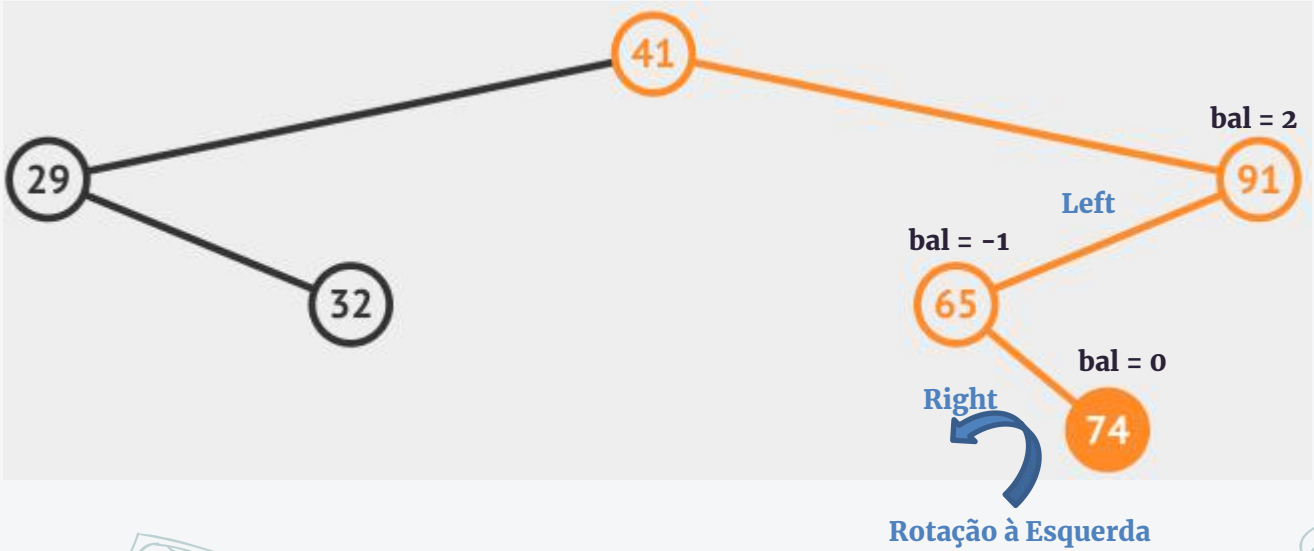
- Dupla Direita (LR):



Árvores AVL

Rotações/Transformações

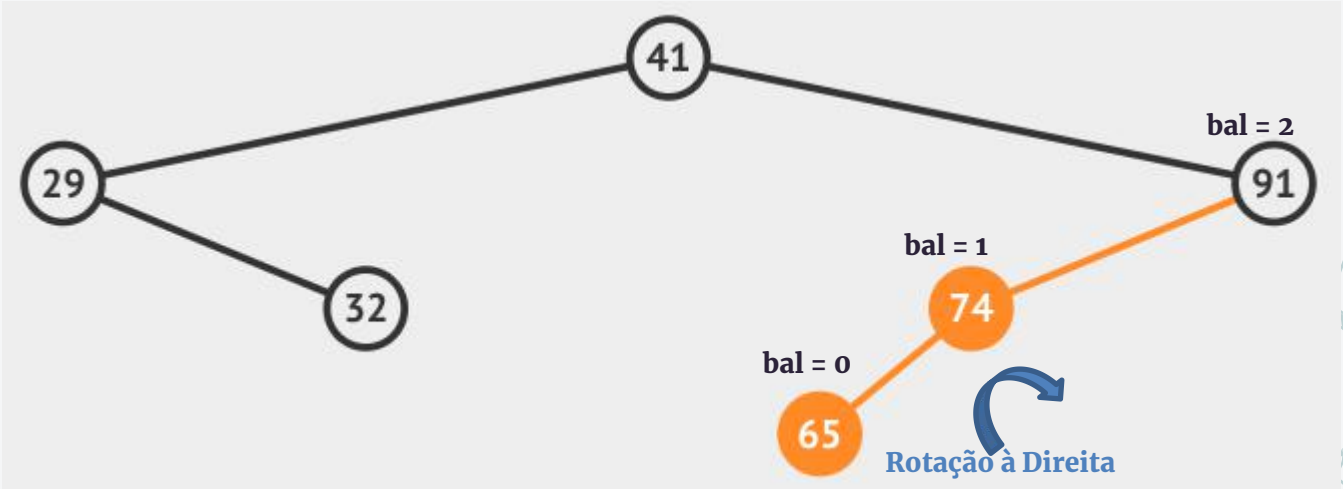
- Dupla Direita (**LR**):



Árvores AVL

Rotações/Transformações

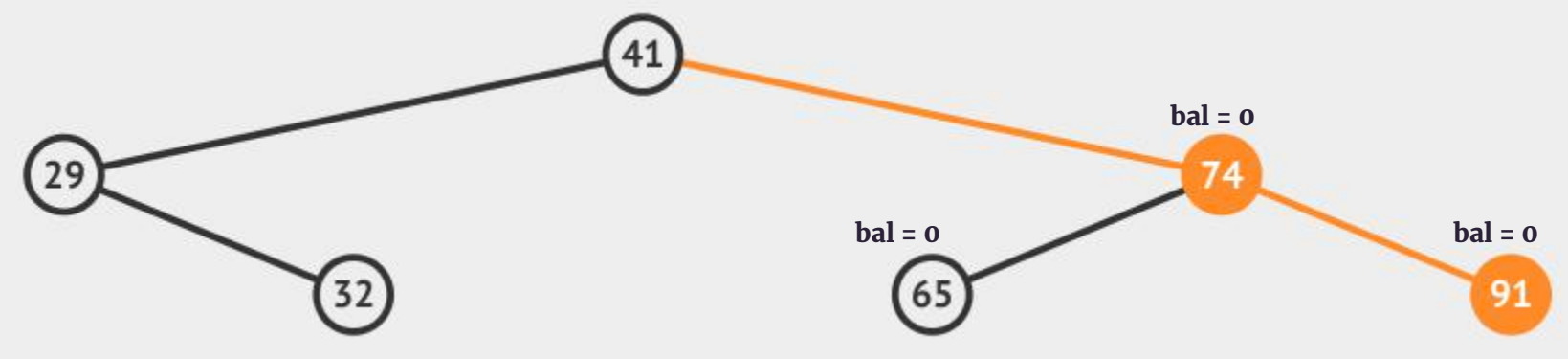
- Dupla Direita (LR):



Árvores AVL

Rotações/Transformações

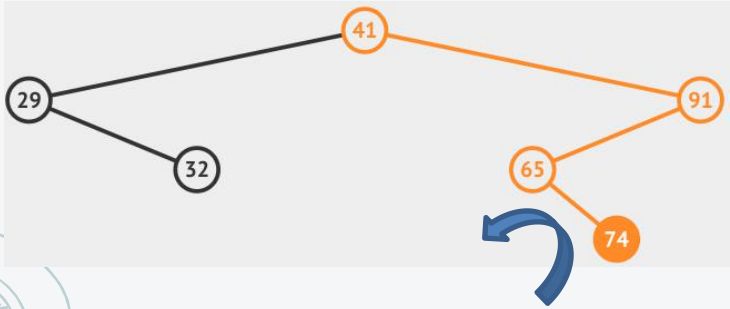
- Dupla Direita (LR):



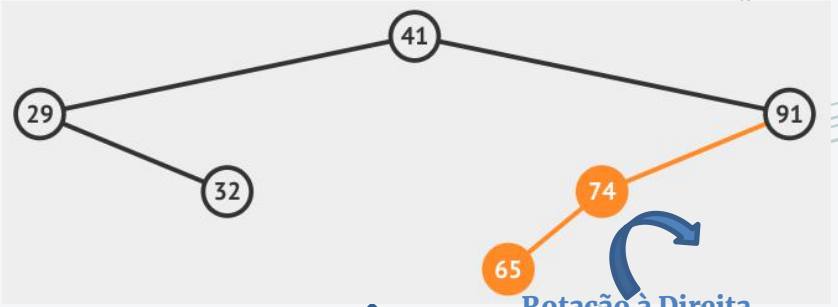
Árvores AVL

Rotações/Transformações

- Dupla Direita (LR):



Rotação à Esquerda

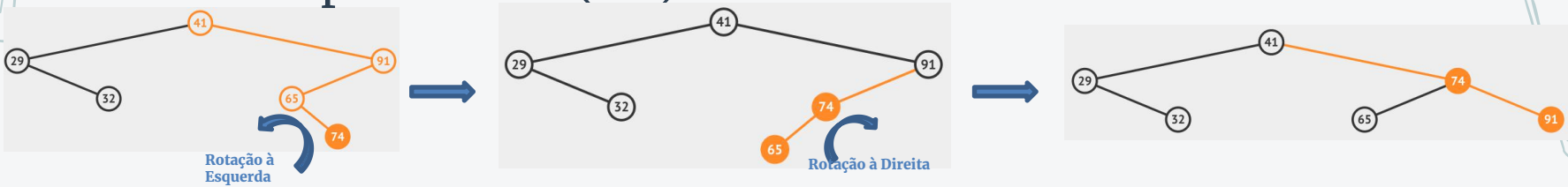


Rotação à Direita

Árvores AVL

Rotações/Transformações

- Dupla Direita (LR):

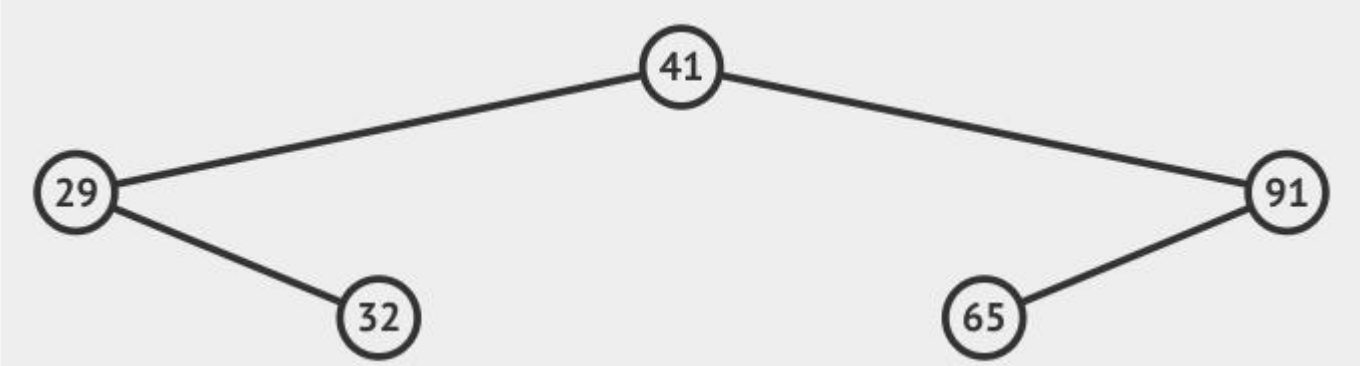



```
188     NodoArvoreAvl rotacaoEsquerdaDireita(NodoArvoreAvl a){
189         a.nodoEsquerda = a.rotacaoEsquerda(a.nodoEsquerda);
190         a = a.rotacaoDireita(a);
191         return a;
192     }
```

Árvores AVL

Rotações/Transformações

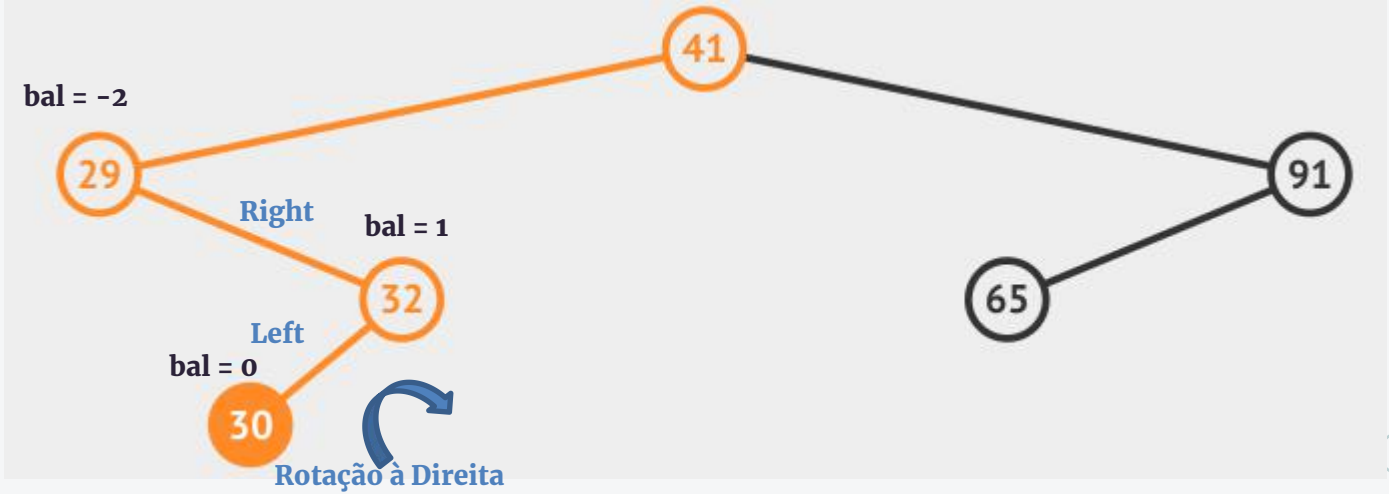
- Dupla Esquerda (RL):



Árvores AVL

Rotações/Transformações

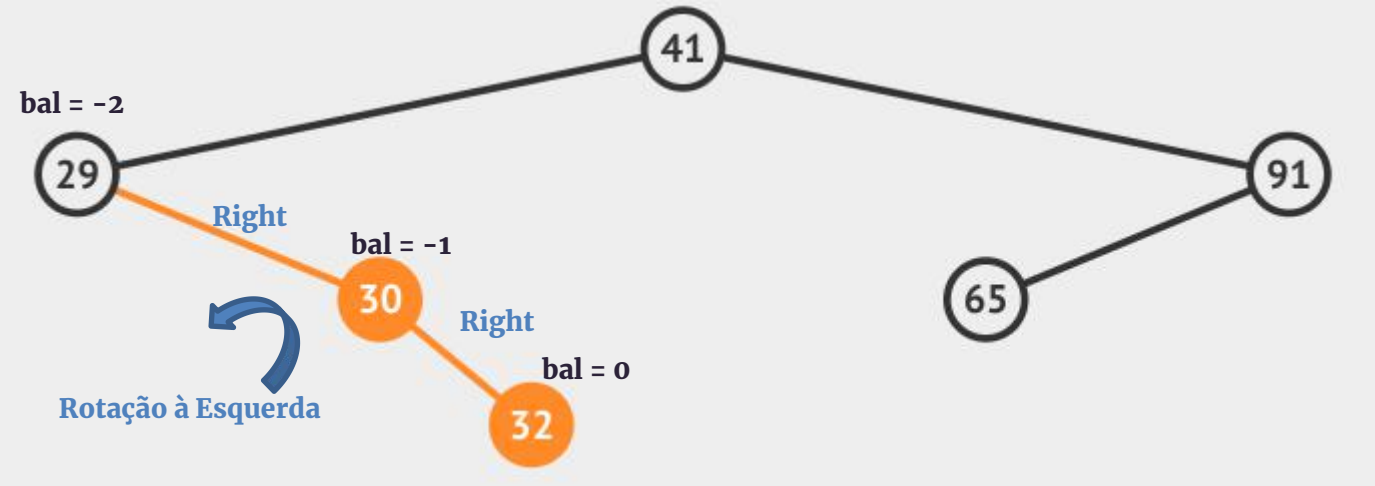
- Dupla Esquerda (**RL**):



Árvores AVL

Rotações/Transformações

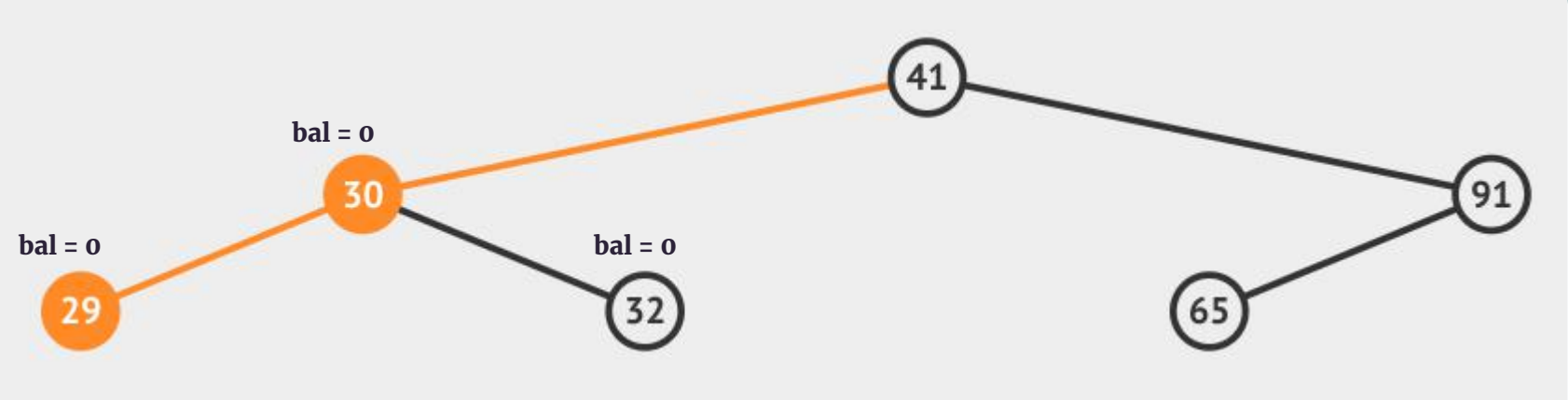
- Dupla Esquerda (**RL**):



Árvores AVL

Rotações/Transformações

- Dupla Esquerda(RL):

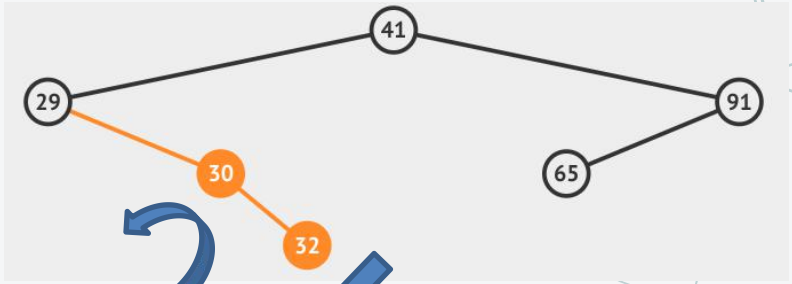


Árvores AVL

Rotações/Transformações

- Dupla Esquerda(RL):

Rotação à Direita

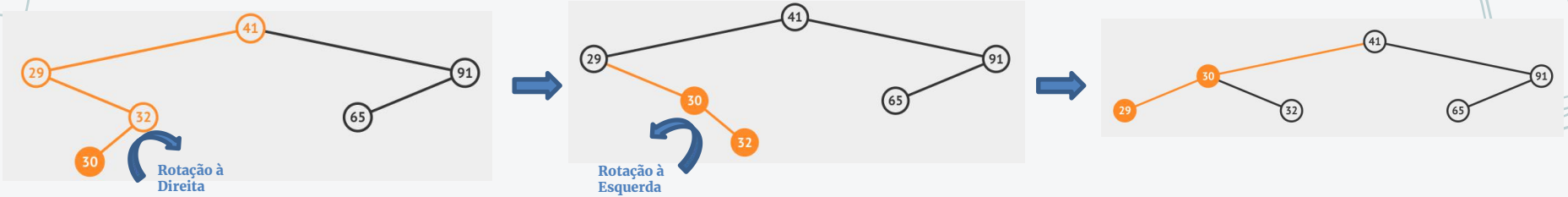


Rotação à Esquerda

Árvores AVL

Rotações/Transformações

- Dupla Esquerda(RL):




```
87    NodoArvoreAVL rotacaoDireitaEsquerda(NodoArvoreAVL a){
88        a.direita = a.rotacaoDireita(a.direita);
89        a = a.rotacaoEsquerda(a);
90        return a;
91    }
```

Referências

- SZWARCFITER, J. L., MARKEZON, L.. Estruturas de Dados e Seus Algoritmos. 3 Edição. 2015.

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Obrigado!

Prof. Vinicius Ramos
https://viniciusramos.pro.br
v.ramos@ufsc.br

Árvores AVL

* Implementações

Prof. Vinicius Ramos

Árvores AVL - Estrutura

- Criar a classe ArvoreAvl
- Criar a classe NodoArvoreAvl
- Criar um construtor
- Incluir as rotações:
  - Primeiro no Nodo
  - Depois na Árvore

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Obrigado!

Prof. Vinicius Ramos
https://viniciusramos.pro.br
v.ramos@ufsc.br

Árvores AVL

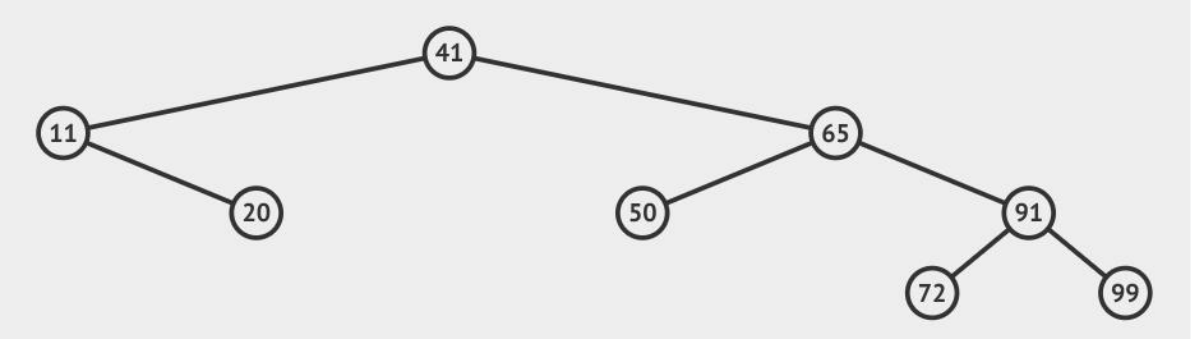
*Inclusão

Prof. Vinicius Ramos

Árvores AVL

Inclusão

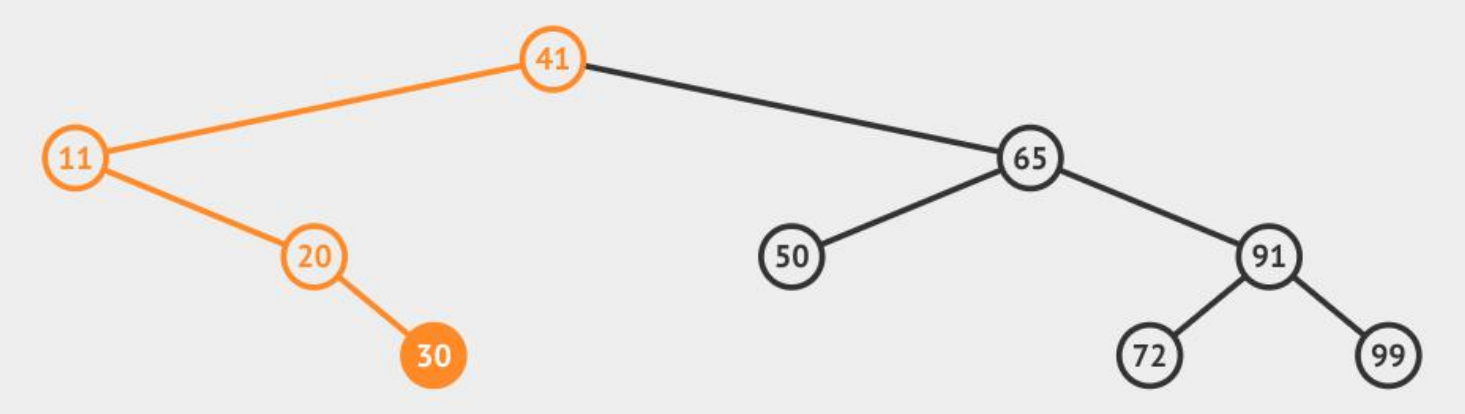
- Incluir o nodo 30:



Árvores AVL

Inclusão

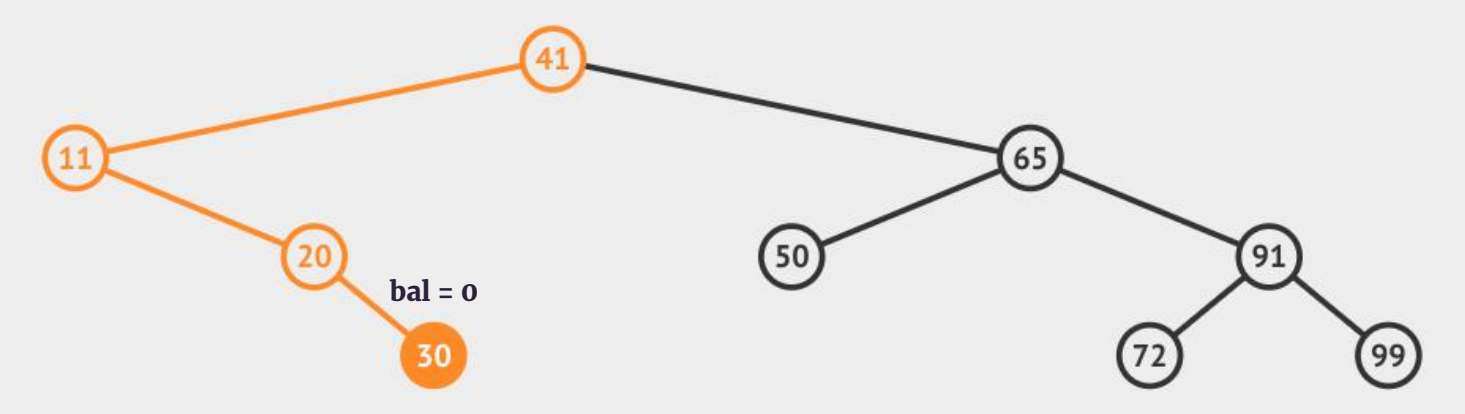
- Incluir o nodo 30:



Árvores AVL

Inclusão

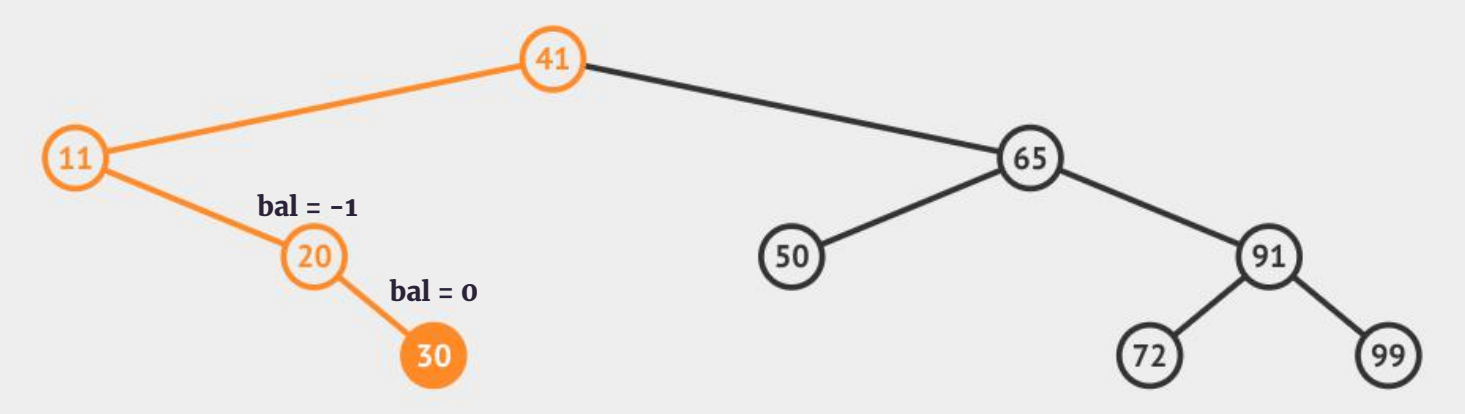
- Incluir o nodo 30:



Árvores AVL

Inclusão

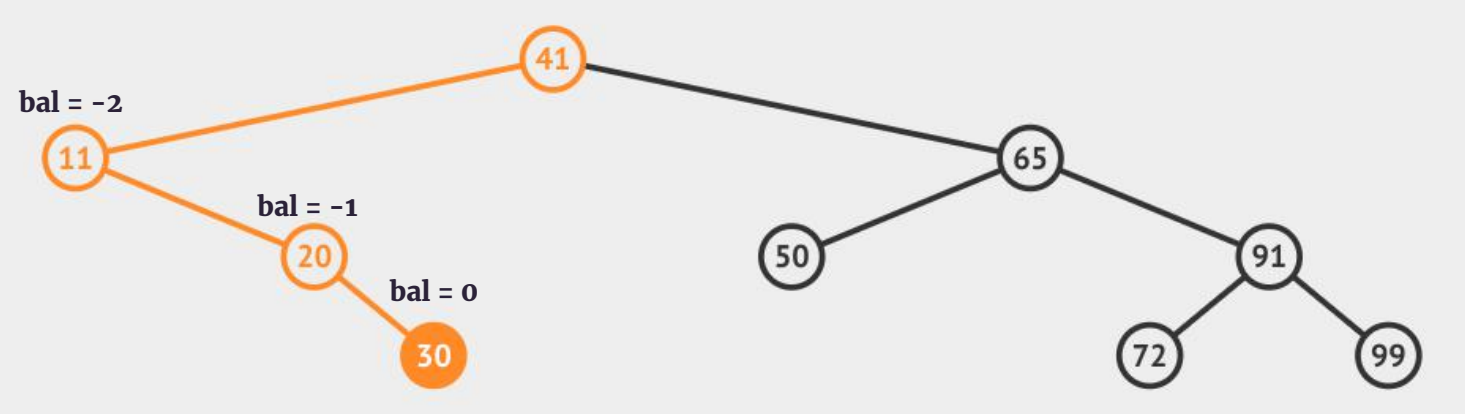
- Incluir o nodo 30:



Árvores AVL

Inclusão

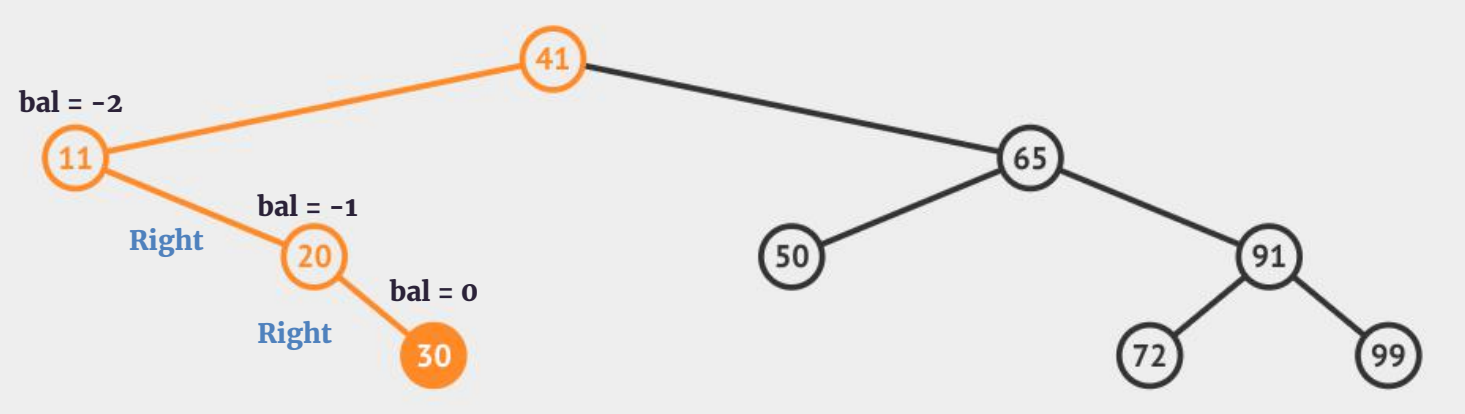
- Incluir o nodo 30:



Árvores AVL

Inclusão

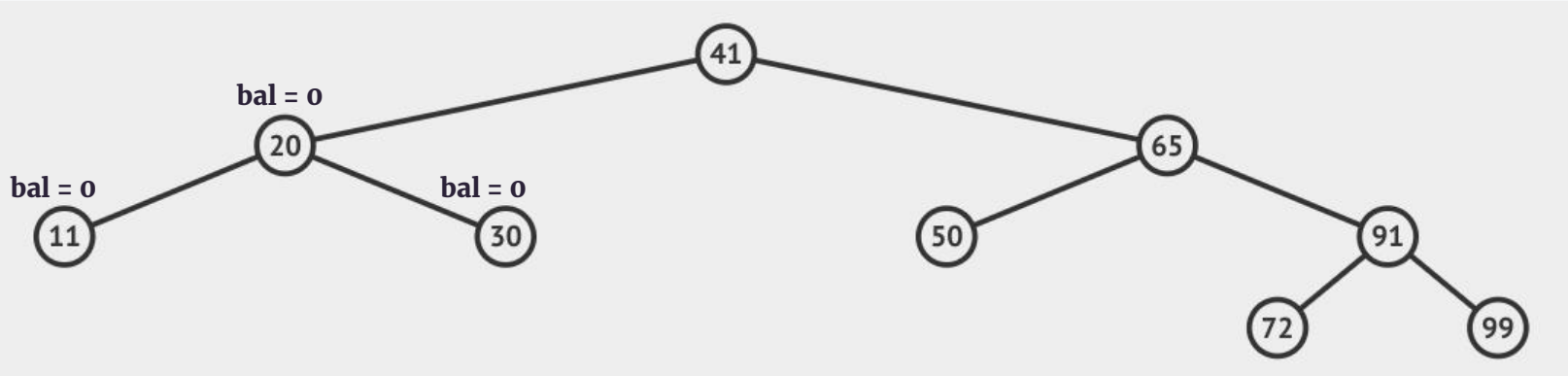
- Incluir o nodo 30:



Árvores AVL

Inclusão

- Incluir o nodo 30
- Rotação Simples Esquerda (RR):



```
61      NodoArvoreAvl insereBalanceado(NodoArvoreAvl a, int valor) {
62          if (a == null)
63              a = new NodoArvoreAvl(valor, null, null);
64          else if (valor < a.valor) { //insere na esquerda
65              a.esquerda = insereBalanceado(a.esquerda, valor);
66              if (this.altura(a.esquerda) - this.altura(a.direita) == 2){
67                  if (valor < a.esquerda.valor)
68                      a = a.rotacaoDireita(a);
69                  else
70                      a = a.rotacaoEsquerdaDireita(a);
71              }
72          } else { // insere na direita
73                  a.direita = insereBalanceado(a.direita, valor);
74                  if (this.altura(a.esquerda)- this.altura(a.direita) == -2)
75                      if (valor > a.direita.valor)
76                          a = a.rotacaoEsquerda(a);
77                      else
78                          a = a.rotacaoDireitaEsquerda(a);
79          }
80          return a;
81      }
```

```
51      int max(int a, int b) {
52          return (a > b? a:b);
53      }
54
55      int altura(NodoArvoreAvl nodo) {
56          if (nodo == null) return -1;
57          else return 1 +
58                  max(altura(nodo.esquerda), altura(nodo.direita));
59      }
60
```

Referências

- SZWARCFITER, J. L., MARKEZON, L.. Estruturas de Dados e Seus Algoritmos. 3 Edição. 2015.

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Obrigado!

Prof. Vinicius Ramos
https://viniciusramos.pro.br
v.ramos@ufsc.br

Árvores AVL

* Testes das Implementações

Prof. Vinicius Ramos

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Obrigado!

Prof. Vinicius Ramos
https://viniciusramos.pro.br
v.ramos@ufsc.br