UNISINOS
DESAFIE
O AMANHÃ.

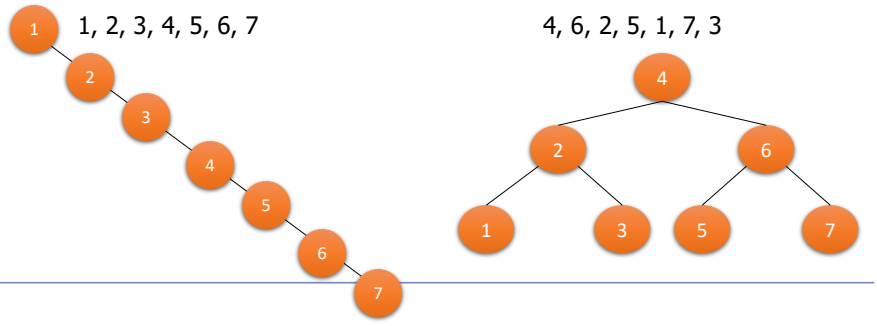
# Estruturas Avançadas de Dados I

Árvores AVL

1

## Introdução

- As Árvores Binárias de Pesquisa (ABP) estudadas têm uma **séria desvantagem** que pode afetar o tempo necessário para recuperar um item armazenado;
- A desvantagem é que o desempenho da ABP **depende da ordem em que os elementos são inseridos**.




UNISINOS
DESAFIE
O AMANHÃ.

2

# Introdução

- Idealmente, deseja-se que a árvore esteja **balanceada**, para qualquer nó **p** da árvore;
- Como saber se a árvore está **balanceada**?
- Para cada nó **p** da árvore a altura da sua **sub-árvore à esquerda** é **aproximadamente igual** à altura da sua **sub-árvore à direita**.

UNISINOS
DESAFIE
O AMANHÃ.

3

# Definição

- **Balanceamento Estático**

  Este balanceamento consiste em, depois de um certo tempo de uso da árvore, destruir sua estrutura, guardando suas informações em uma lista ordenada e reconstruí-la de forma balanceada.

- **Balanceamento Dinâmico**

  Tem por objetivo reajustar os nós de uma árvore sempre que uma inserção ou remoção provocar desbalanceamento.

UNISINOS
DESAFIE
O AMANHÃ.

4

# Definição

- O nome AVL vem de seus criadores Adelson Velsky e Landis (1962);
- Uma ABP **T** é denominada **AVL** se:
  - Para todos nós de **T**, as alturas de suas duas sub-árvores diferem **no máximo em uma unidade;**


5

# Definição

- Como saber se a árvore está **desbalanceada** ?
  - Verificando se existe algum nó “desregulado”.
- Como saber se um nodo está **desregulado** ?
  - Subtraindo-se as alturas das suas sub-árvores.
- Por questões de **eficiência**, estas diferenças são pré-calculadas e armazenadas nos nós correspondentes, sendo atualizadas durante as operações.


6

# AVL – Fator de Balanceamento

- Possíveis valores de diferença para cada nó em uma árvore balanceada: **-1**, **0**, **1**.
- Fator de Balanceamento (FB) de cada nó da árvore
  - **FB(p)** = h(sae(p)) – h(sad(p)) :

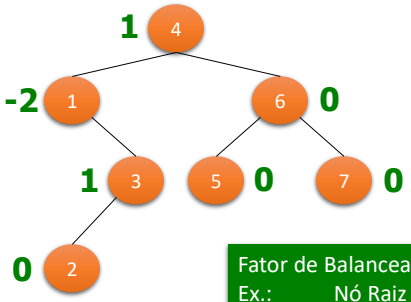
    h = altura
    p = nó
    sae = sub-árvore à esquerda
    sad = sub-árvore à direita
- Em uma árvore binária balanceada todos os FB de todos os nós estão no intervalo -1 ≤ FB ≤ 1.



# AVL

- Exemplo de uma ABP não AVL

Inserção: **4, 1, 3, 6, 5, 2 e 7**




Fator de Balanceamento
Ex.: Nó Raiz
FB("4") = 3 – 2
FB("4") = **1**

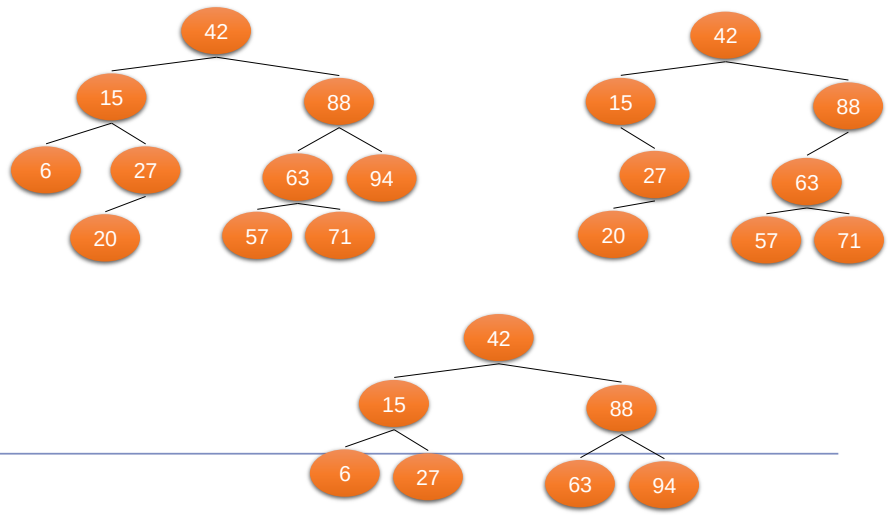
# AVL – Vantagem

- A vantagem de uma árvore balanceada com relação a uma degenerada está em sua eficiência;
- Ex.: numa árvore binária degenerada de 10.000 nós são necessárias, em média, 5.000 comparações (semelhança com arrays ordenados e listas encadeadas);
- Numa árvore balanceada com o mesmo número de nós essa média reduz-se a 14 comparações.

UNISINOS
DESAFIE
O AMANHÃ.

9

# AVL – Identifique quais ABPs são AVLs




UNISINOS
DESAFIE
O AMANHÃ.

10

# AVL – Operações

- A inserção ou remoção de um nó em uma árvore AVL **pode** ou **não** provocar seu desbalanceamento;
- Se a árvore AVL ficar desbalanceada, a restauração do seu balanceamento é realizado através de **ROTAÇÕES**.

Rotação Simples:
Rotação à Esquerda
Rotação à Direita

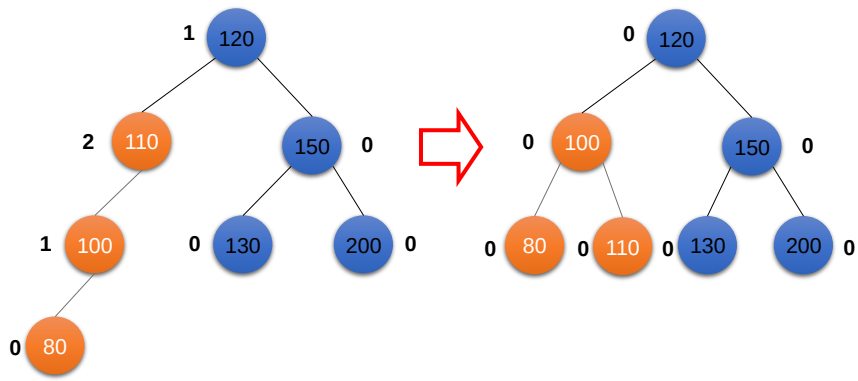
Rotação Dupla:
Rotação à Esquerda
Rotação à Direita

UNISINOS
DESAFIE
O AMANHÃ.

11

# AVL – Rotação Simples à Direita

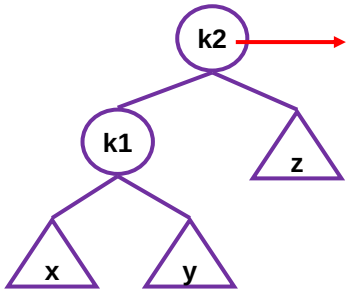
- Toda vez que uma sub-árvore fica com um fator:
  - **positivo** e sua sub-árvore da esquerda também tem um fator **positivo**




UNISINOS
DESAFIE
O AMANHÃ.

13

# AVL – Rotação Simples à Direita

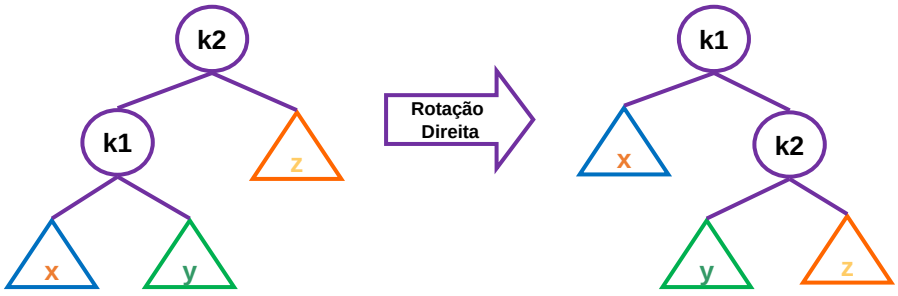



Nó k2 é a raiz de transformação

X, Y e Z são subárvores (vazias ou não)

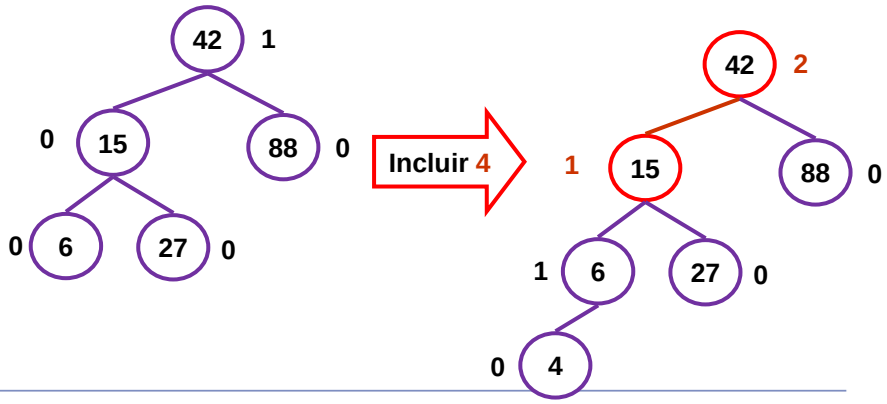
14

# AVL – Rotação Simples à Direita



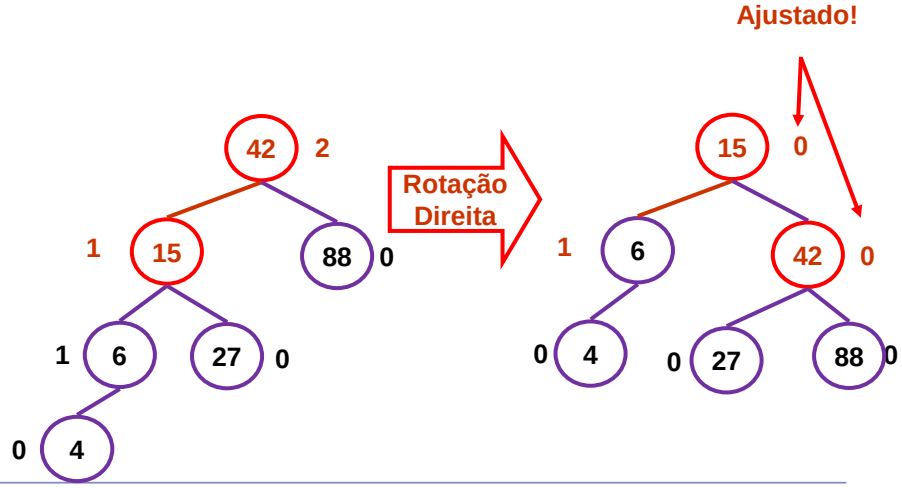

17

# AVL – Rotação Simples à Direita




18

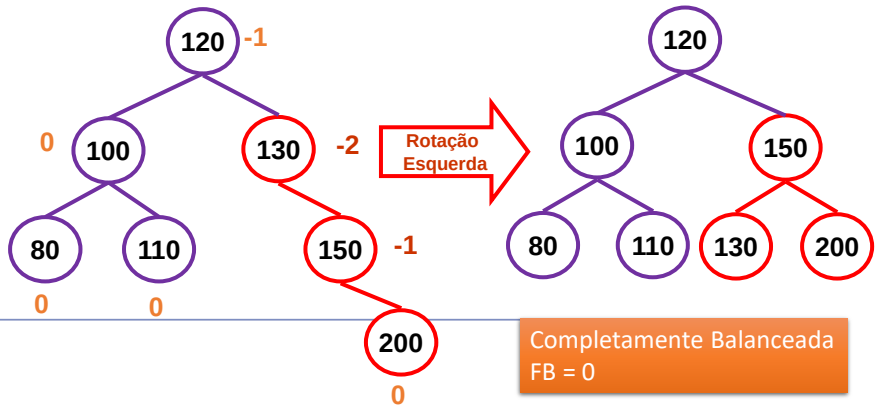
# AVL – Rotação Simples à Direita




19

# AVL – Rotação Simples à Esquerda

- Toda vez que uma sub-árvore fica com um fator:
  - negativo e sua sub-árvore da direita também tem um fator negativo

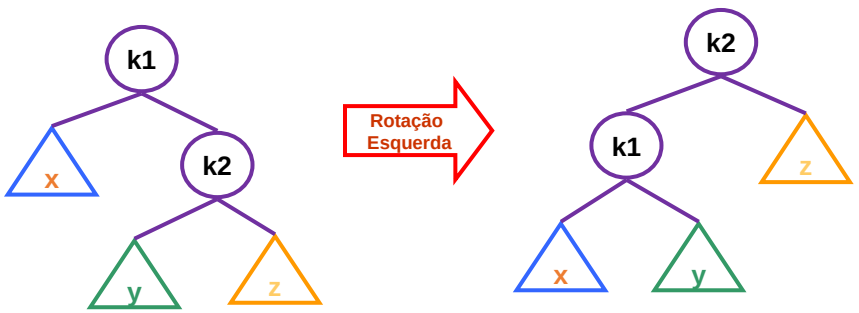



Completamente Balanceada
FB = 0

UNISINOS
DESAFIE
O AMANHÃ.

20

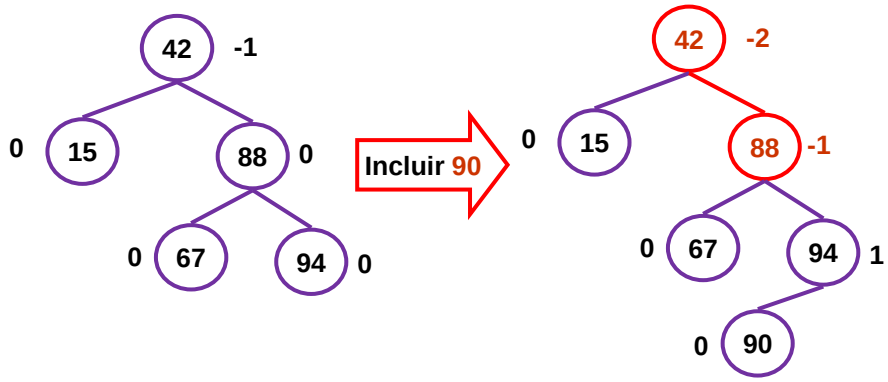
# AVL – Rotação Simples à Esquerda




UNISINOS
DESAFIE
O AMANHÃ.

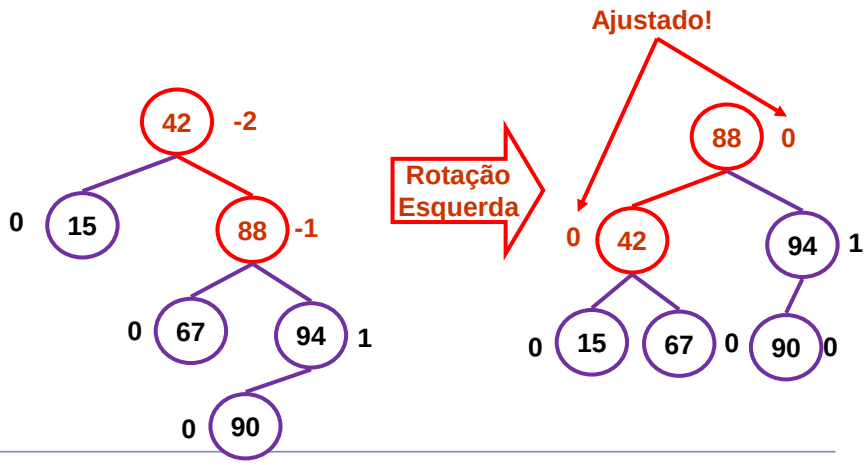
22

# AVL – Rotação Simples à Esquerda




23

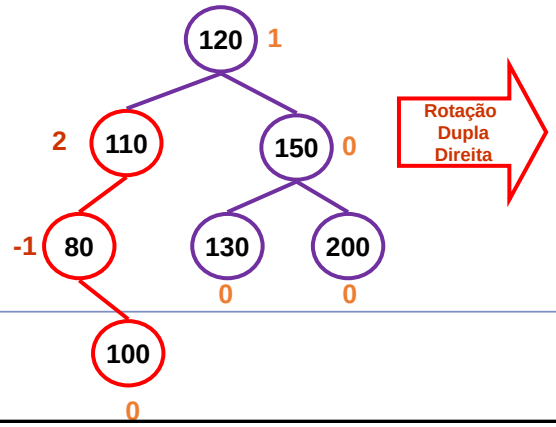
# AVL – Rotação Simples à Esquerda




24

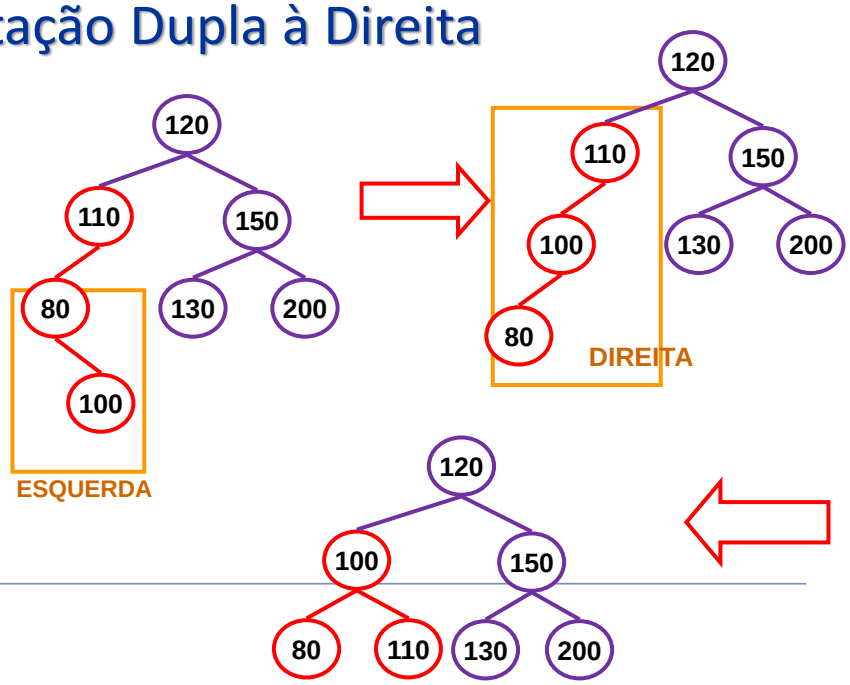
# AVL – Rotação Dupla à Direita

- Toda vez que uma sub-árvore fica com um fator:
  - positivo e sua sub-árvore da esquerda tem um fator negativo



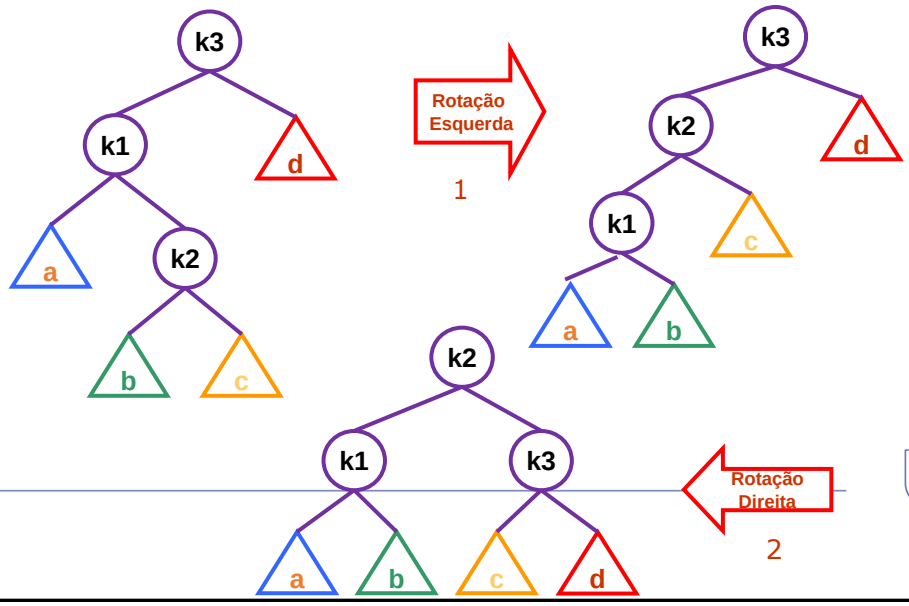

25

# AVL – Rotação Dupla à Direita



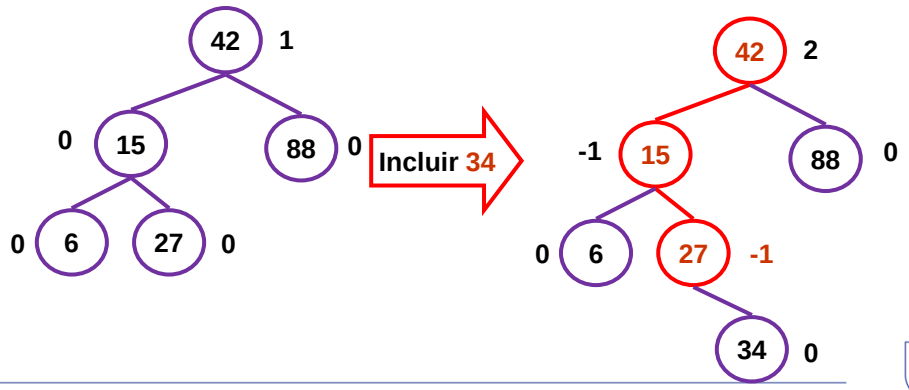

26

# AVL – Rotação Dupla à Direita



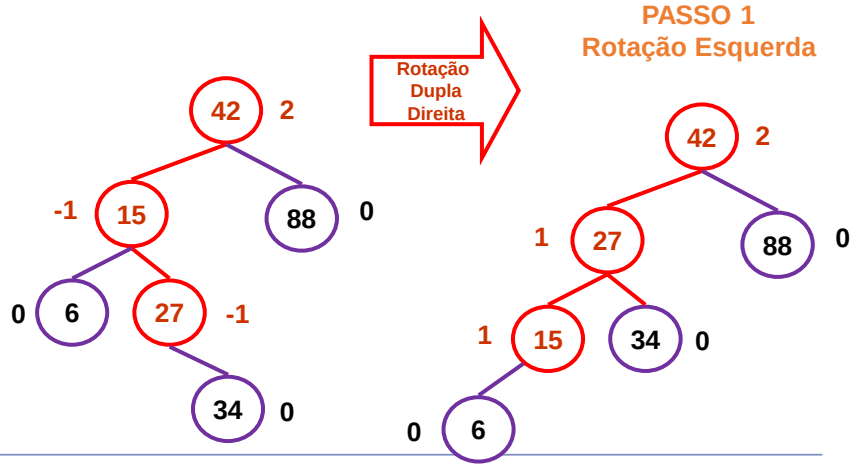

27

# AVL – Rotação Dupla à Direita



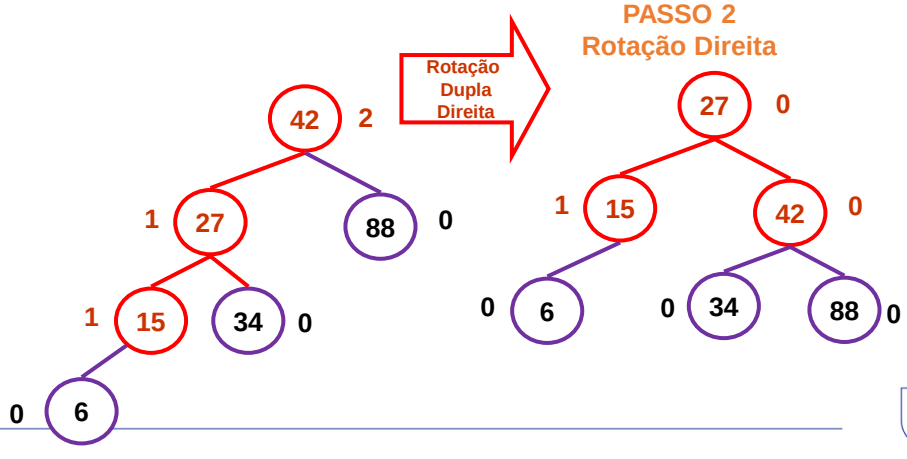

29

# AVL – Rotação Dupla à Direita




30

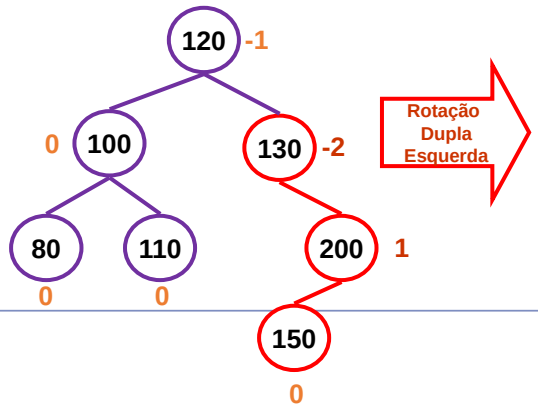
# AVL – Rotação Dupla à Direita




31

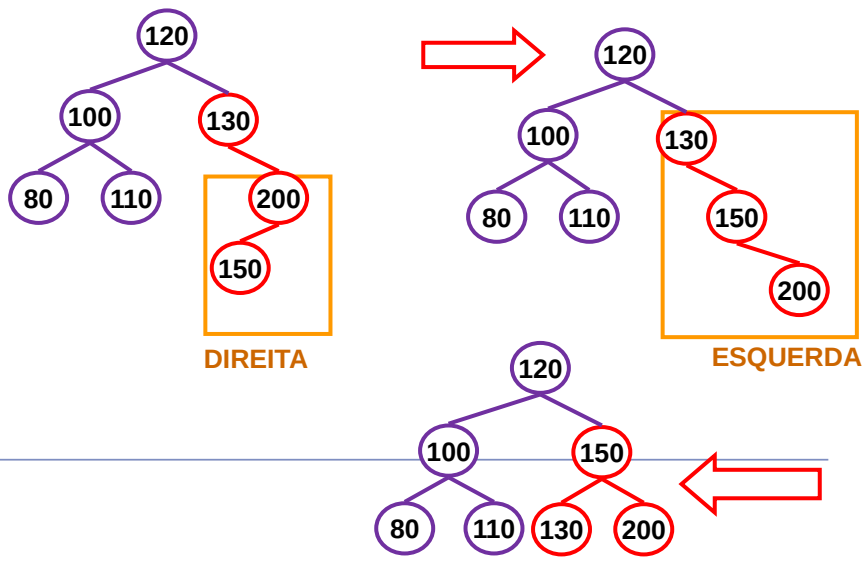
# AVL – Rotação Dupla à Esquerda

- Toda vez que uma sub-árvore fica com um fator:
  - negativo e sua sub-árvore da direita tem um fator positivo



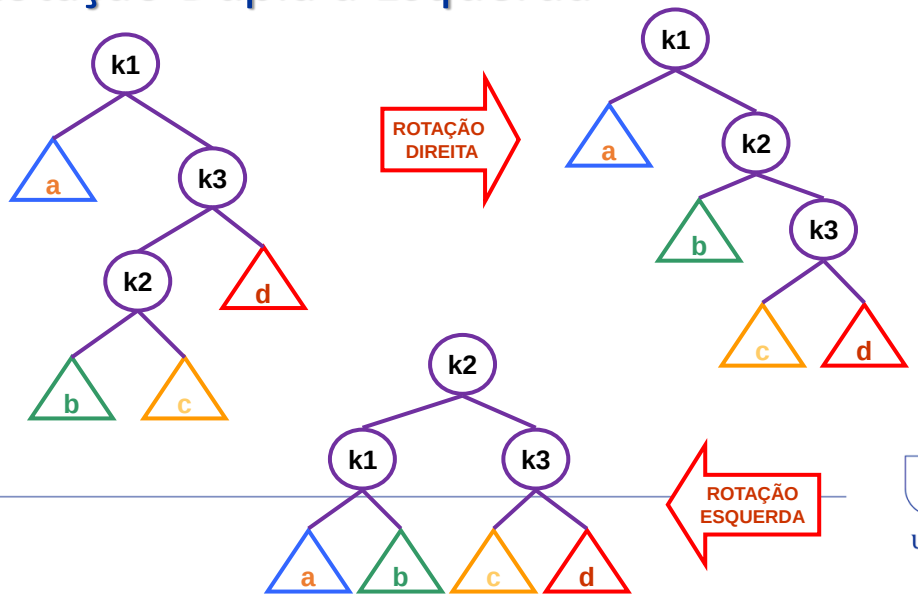

32

# AVL – Rotação Dupla à Esquerda




33

# AVL – Rotação Dupla à Esquerda




34

# AVL – Inserção

- Percorre-se a árvore verificando se a chave já existe ou não
  - Em caso positivo, encerra a tentativa de inserção
  - Caso contrário, a busca encontra o local correto de inserção do novo nó
- Verifica-se se a inclusão tornará a árvore desbalanceada
  - Em caso negativo, o processo termina
  - Caso contrário, deve-se efetuar o balanceamento da árvore
    - Descobre-se qual a operação de rotação a ser executada
    - Executa-se a rotação

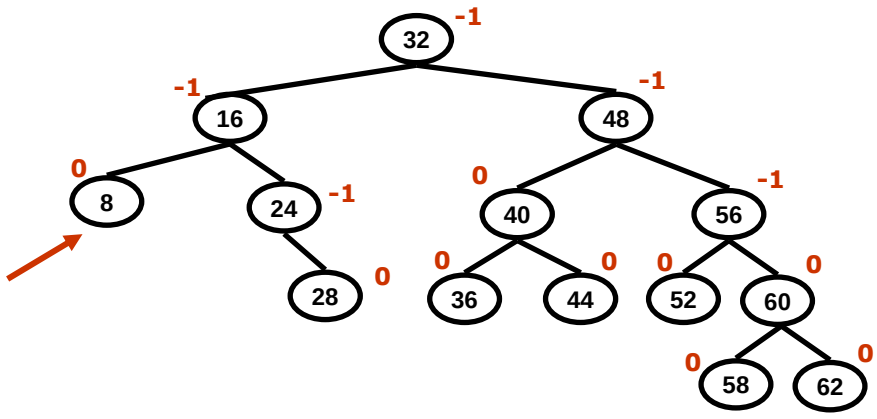
36

# AVL – Exclusão

- Caso parecido com as inclusões;
- No entanto, nem sempre se consegue solucionar com uma única rotação;
- Remover elemento e retornar do pai do nó removido até a raiz, verificando se cada nó do caminho precisa ser balanceado.

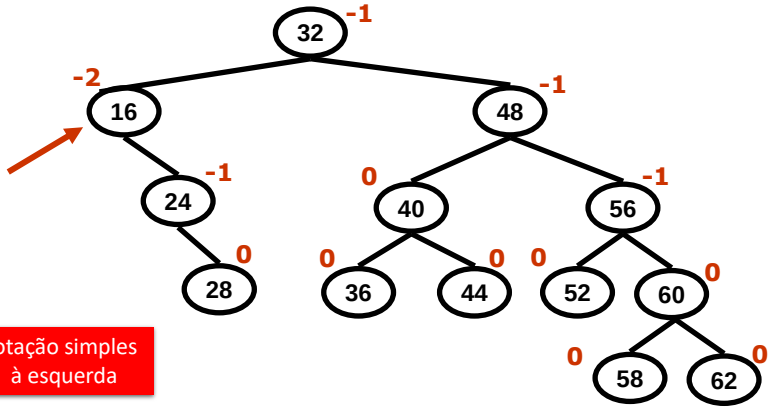


# AVL – Exclusão




Excluindo 8

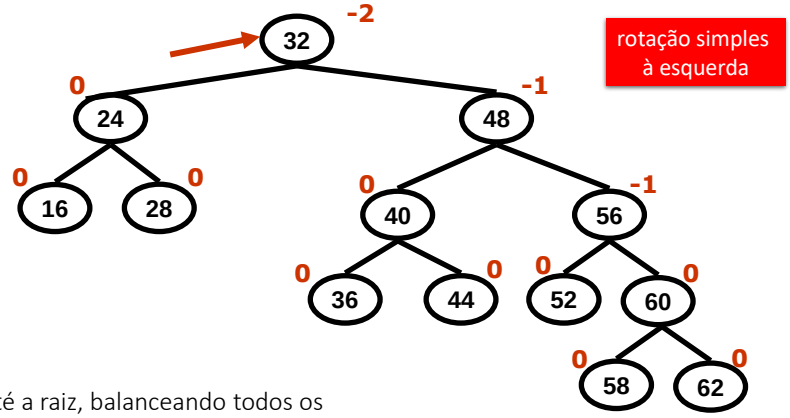
# AVL – Exclusão




rotação simples à esquerda



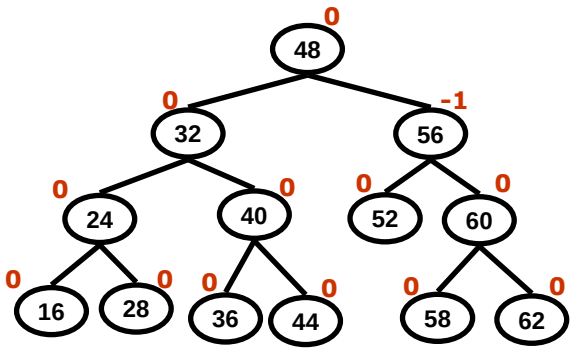
# AVL – Exclusão




rotação simples à esquerda

volta recursivamente até a raiz, balanceando todos os nós que se encontrarem desbalanceados.

# AVL – Exclusão




41

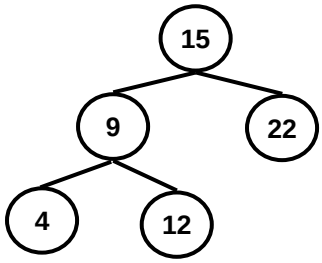
# AVL - Complexidade

- Uma única reestruturação é O(1)
  - usando uma árvore binária implementada com estrutura ligada
- Pesquisa é O(log n)
  - altura de árvore é O(log n), não necessita reestruturação
- Inserir é O(log n)
  - busca inicial é O(log n)
  - reestruturação para manter balanceamento é O(log n)
- Remover é O(log n)
  - busca inicial é O(log n)
  - reestruturação para manter balanceamento é O(log n)

42

# Exercícios

1. Considere a árvore abaixo, no qual 12 está entre 9 e 15.
   - Fazendo a rotação direita em 9, onde ficará 12?
   - Terminada a rotação à direita, tente agora a rotação à esquerda em 15.

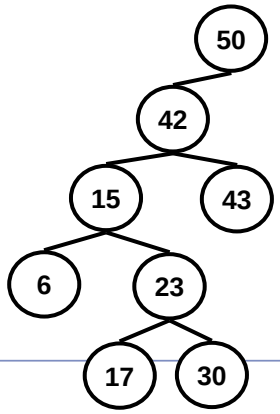



UNISINOS
DESAFIE
O AMANHÃ.

44

# Exercícios

2. Considere a árvore abaixo:
   - Execute a rotação em 15. O que acontece com o nó 42 (que é pai de 15)?
   - Execute agora a operação oposta em 42. O que acontece com 15 (que é filho de 42)?




UNISINOS
DESAFIE
O AMANHÃ.

45

# Exercícios

3. Monte a árvore AVL (passo a passo) para as seguintes chaves: 10, 20, 30, 40 e 45 (nesta ordem) indicando em cada passo:
   1) Qual elemento foi inserido
   2) O grau de balanceamento de cada nó
   3) Qual rotação foi realizada, se necessária.

4. Monte a **árvore AVL** (passo a passo) para as seguintes inserções de chaves 10, 15, 7, 25, 30, 27, 49 (nesta ordem), indicando a cada passo qual elemento foi inserido e qual rotação foi realizada.


46

# Referências Bibliográficas


- ASCENCIO, A. F. G; ARAÚJO, G. S. **Estruturas de Dados**. São Paulo: Pearson Prentice Hall, 2010. 432 p.
- CORMEN, Thomas H. et al. **Introduction to algorithms.** 3. ed. Cambridge: MIT, 2009. xix. 1292 p.
- JAQUES, Patrícia. **Lâminas Árvores Binárias – Programação II**, Unisinos.



47

**Prof. Márcio Garcia Martins**
**marciog@unisinos.br**

Para anotar: *ao enviar e-mail sempre coloque o seguinte prefixo no assunto*
**[EADI-ano-semestre] – Nome do aluno**


48