# AVL

SCC-502 – Algoritmos e Estruturas de Dados I

Thiago A. S. Pardo

## Árvores binárias de busca (ABB)

- Muito boas para busca
  - O(log n)
- Sabe-se que
  - Lista encadeada
    - Eficiente para inserção e remoção dinâmica de elementos, mas ineficiente para busca
  - Lista seqüencial (ordenada)
    - Eficiente para busca, mas ineficiente para inserção e remoção de elementos

mas... ABBs: solução eficiente para inserção, remoção e busca



## ABB

- Contra-exemplo
  - Inserção dos elementos na ordem em que aparecem
    - A, B, C, D, E, ..., Z
    - 1000, 999, 998, ..., 1



## ABB

- O desbalanceamento da árvore pode tornar a busca tão ineficiente quanto a busca seqüencial (no pior caso)
  - O(N)
- Solução?



## ABB

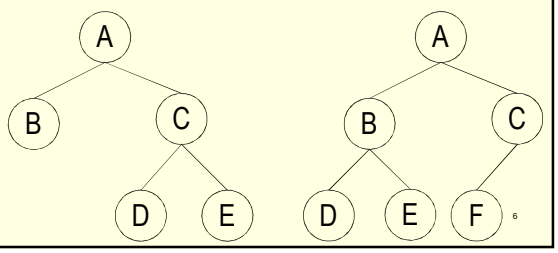
- O desbalanceamento da árvore pode tornar a busca tão ineficiente quanto a busca seqüencial (no pior caso)
  - O(N)
- Solução?

Balanceamento da árvore!



## Árvores balanceadas

- Uma árvore binária é dita balanceada se, para cada nó, as alturas de suas duas subárvores diferem de, no máximo, 1

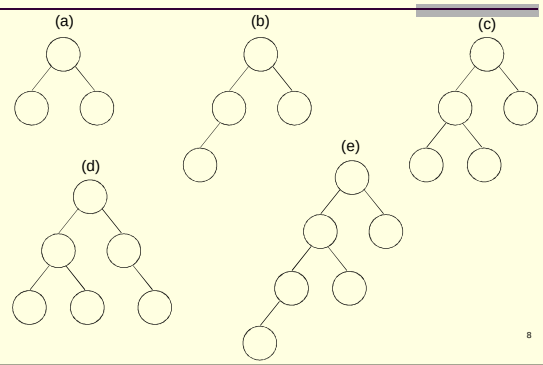


## AVL

- Árvore binária de busca balanceada
  - Para cada nó, as alturas das subárvores diferem em 1, no máximo
  - Proposta em 1962 pelos matemáticos russos G.M. Adelson-Velskki e E.M. Landis
    - Métodos de inserção e remoção de elementos da árvore de forma que ela fique balanceada

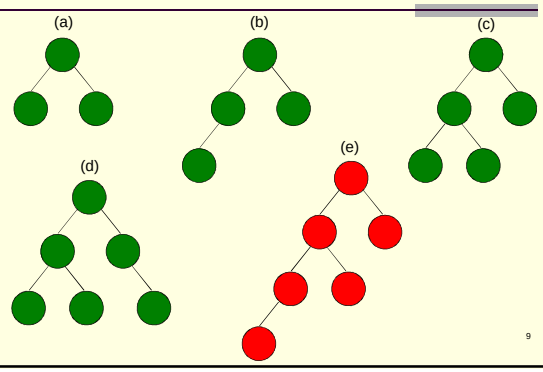
7

## AVL: quem é e quem não é?



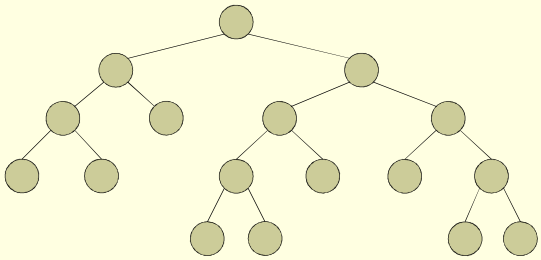

8

## AVL: quem é e quem não é?



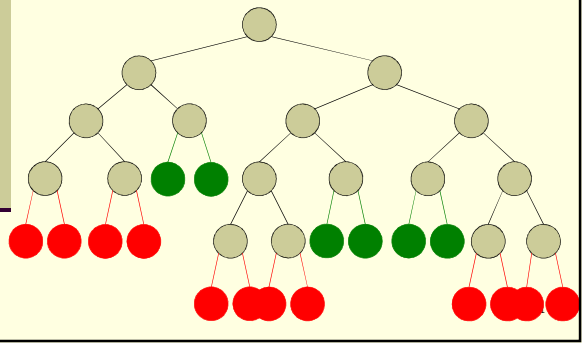

9

## Pergunta: a árvore abaixo é AVL?



10

## Exercício: onde se pode incluir um nó para a AVL continuar sendo AVL?



## AVL

- Como é que se sabe quando é necessário balancear a árvore?
  - Se a diferença de altura das subárvores deve ser 1, no máximo, então temos que procurar diferenças de altura maior do que isso
  - Possível solução: cada nó pode manter a diferença de altura de suas subárvores
    - Convencionalmente chamada de fator de balanceamento do nó

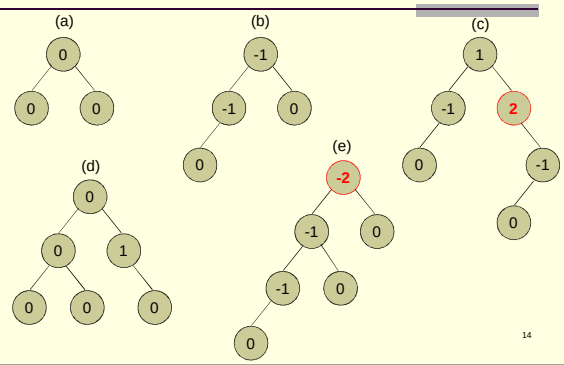
12

## AVL

- Fatores de balanceamento dos nós
  - Altura da subárvore direita menos altura da subárvore esquerda
    - Hd-He
  - Atualizados sempre que a árvore é alterada (elemento é inserido ou removido)
  - Quando um fator é 0, 1 ou -1, a árvore está balanceada
  - Quando um fator se torna 2 ou -2, a árvore está desbalanceada
    - Operações de balanceamento!

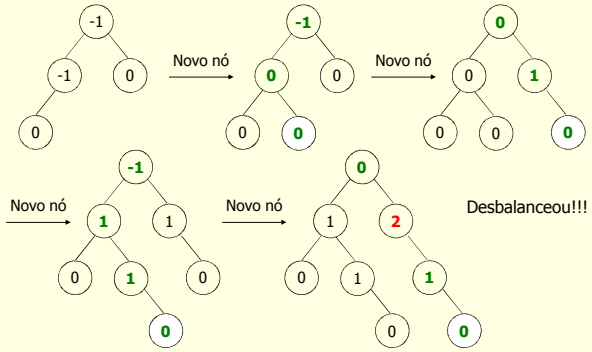
13

## AVL: quem é e quem não é




14

## AVL: exemplo de desbalanceamento



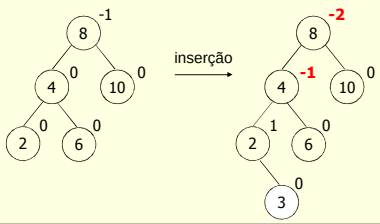

## AVL

- Controle do balanceamento
  - Altera-se o algoritmo de inserção para balancear a árvore quando ela se tornar desbalanceada após uma inserção (nó com FB 2 ou -2)
    - Rotações
      - Se árvore pende para esquerda (FB negativo), rotaciona-se para a direita
      - Se árvore pende para direita (FB positivo), rotaciona-se para a esquerda
    - 2 casos podem acontecer

16

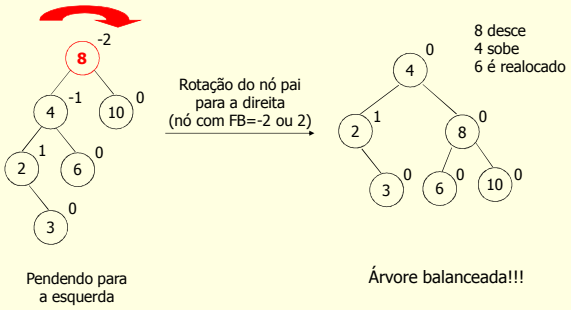
## AVL: primeiro caso

- Raiz de uma subárvore com FB -2 (ou 2) e um nó filho com FB -1 (ou 1)
  - Os fatores de balanceamento têm sinais iguais: subárvores de nó raiz e filho pendem para o mesmo lado

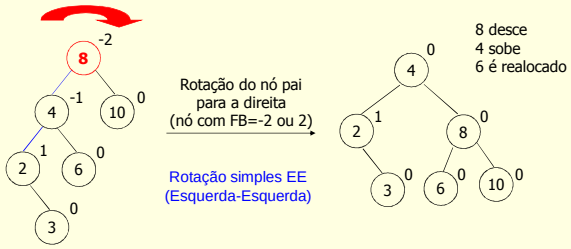



17

## AVL: primeiro caso



## AVL: primeiro caso




Pendendo para a esquerda

Árvore balanceada!!!

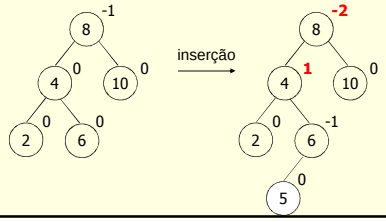
## AVL: primeiro caso

- Quando subárvores do pai e filho pendem para um mesmo lado
  - Rotação simples para o lado oposto
    - EE ou DD (Direita-Direita, com raciocínio inverso)
  - Às vezes, é necessário realocar algum elemento, pois ele perde seu lugar na árvore

20

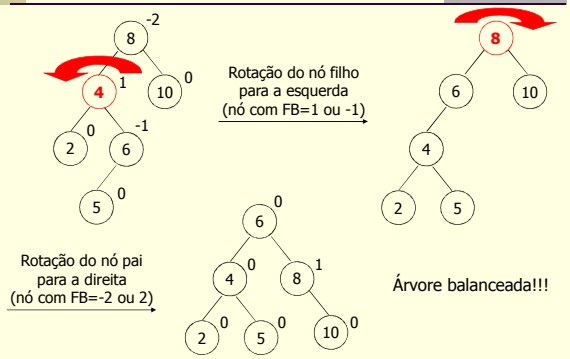
## AVL: segundo caso

- Raiz de uma subárvore com FB -2 (ou 2) e um nó filho com FB 1 (ou -1)
  - Os fatores de balanceamento têm sinais opostos: subárvore de nó raiz pende para um lado e subárvore de nó filho pende para o outro (ou o contrário)



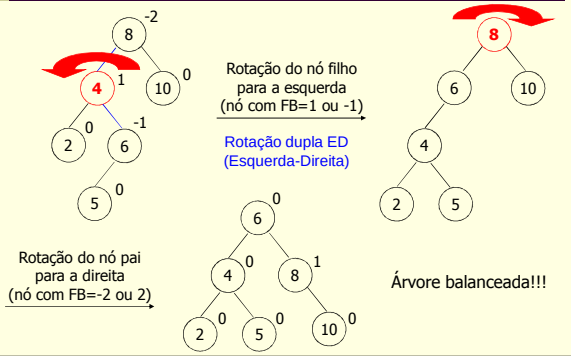

21

## AVL: segundo caso




Árvore balanceada!!!

## AVL: segundo caso




Árvore balanceada!!!

## AVL: segundo caso

- Quando subárvores do pai e filho pendem para lados opostos
  - Rotação dupla
    - Primeiro, rotaciona-se o filho para o lado do desbalanceamento do pai
    - Em seguida, rotaciona-se o pai para o lado oposto do desbalanceamento
      - ED ou DE (com raciocínio inverso)
  - Às vezes, é necessário realocar algum elemento, pois ele perde seu lugar na árvore

24

## AVL

- As transformações dos casos anteriores diminuem em 1 a altura da subárvore com raiz desbalanceada p
- Assegura-se o rebalanceamento de todos os ancestrais de p e, portanto, o rebalanceamento da árvore toda

25

## AVL

- Novo algoritmo de inserção
  - A cada inserção, verifica-se o balanceamento da árvore
    - Se necessário, fazem-se as rotações de acordo com o caso (sinais iguais ou não)
  - Em geral, armazena-se uma variável de balanceamento em cada nó para indicar o FB

26

## AVL

- Declaração

```
typedef int elem;

typedef struct no {
        elem info;
        struct no *esq, *dir;
        int FB;
} no;

typedef struct {
      no *raiz;
} AVL;
```

27

## AVL

- Exercício
  - Inserir os elementos 10, 3, 2, 5, 7 e 6 em uma árvore e balancear quando necessário

28

## AVL

- Exercício
  - Inserir os elementos A, B, C, ..., J em uma árvore e balancear quando necessário

29

## AVL

- Os percursos em-ordem da árvore original e da balanceada permanecem iguais
  - Exercício: prove para um dos exemplos anteriores!

30

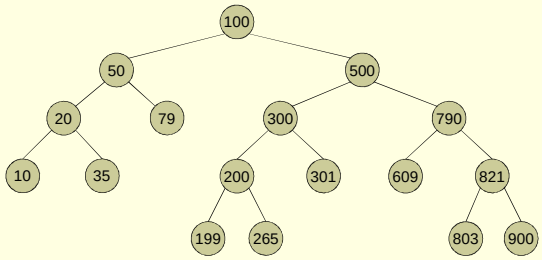
## AVL

- Implementar sub-rotinas de inserção de elementos na AVL

31

## AVL

- Exercício: teste a sub-rotina anterior inserindo alguns elementos na árvore abaixo




## AVL

- Exercício
  - Pense no procedimento de remoção de um elemento da AVL e responda
    - Quais são os casos de desbalanceamento?
    - Esboce um método de remoção que balanceie a árvore se necessário

33