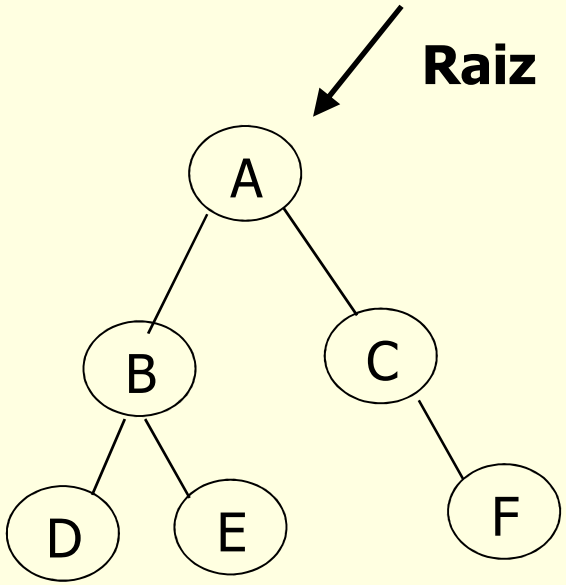
Árvore AVL: teoria e prática

Animated AVL TREE (mostra inserções)
http://www.cs.jhu.edu/~goodrich/dsa/trees/avltree.html

Material de Thiago A. S. Pardo/Revisado por Roseli Romero em nov/2011

Árvores binárias de busca (ABB)

- Árvores de grau 2, isto é, cada nó tem dois filhos, no máximo



Terminologia:

- filho esquerdo
- filho direito
- informação

ABB

- Também chamadas “árvores de pesquisa” ou “árvores ordenadas”

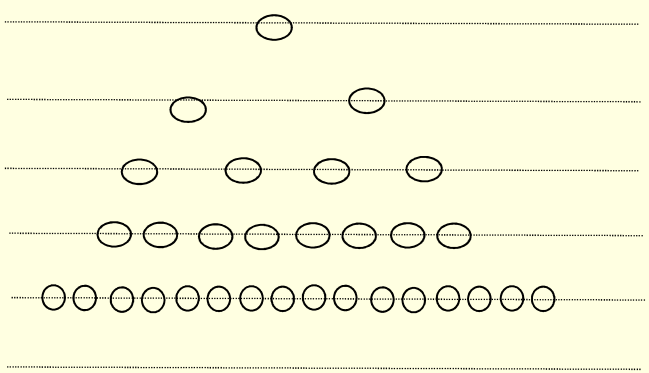
- Definição
  - Uma árvore binária com raiz R é uma ABB se:
    - a chave (informação) de cada nó da subárvore esquerda de R é menor do que a chave do nó R (em ordem alfabética, por exemplo)
    - a chave de cada nó da subárvore direita de R é maior do que a chave do nó R
    - as subárvores esquerda e direita também são ABBs

ABB

- Muito boa para busca
  - Em uma árvore de altura A, visitam-se, no máximo, A nós
  - Grande quantidade de informação em relativamente poucos níveis

ABB

- Quantidade de informação



Nível	Quantos cabem
1	1
2	3
3	7
4	15
...	...
N	**$2^N - 1$**
10	1.024
13	8.192
16	65.536
18	262.144
20	1 milhão
30	1 bilhão
	...

ABB

- Vantagens
  - Se nós espalhados uniformemente, consulta rápida para grande quantidade de dados
    - Divide-se o espaço de busca restante em dois em cada passo da busca
    - O(Log N)

ABB

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

ABB

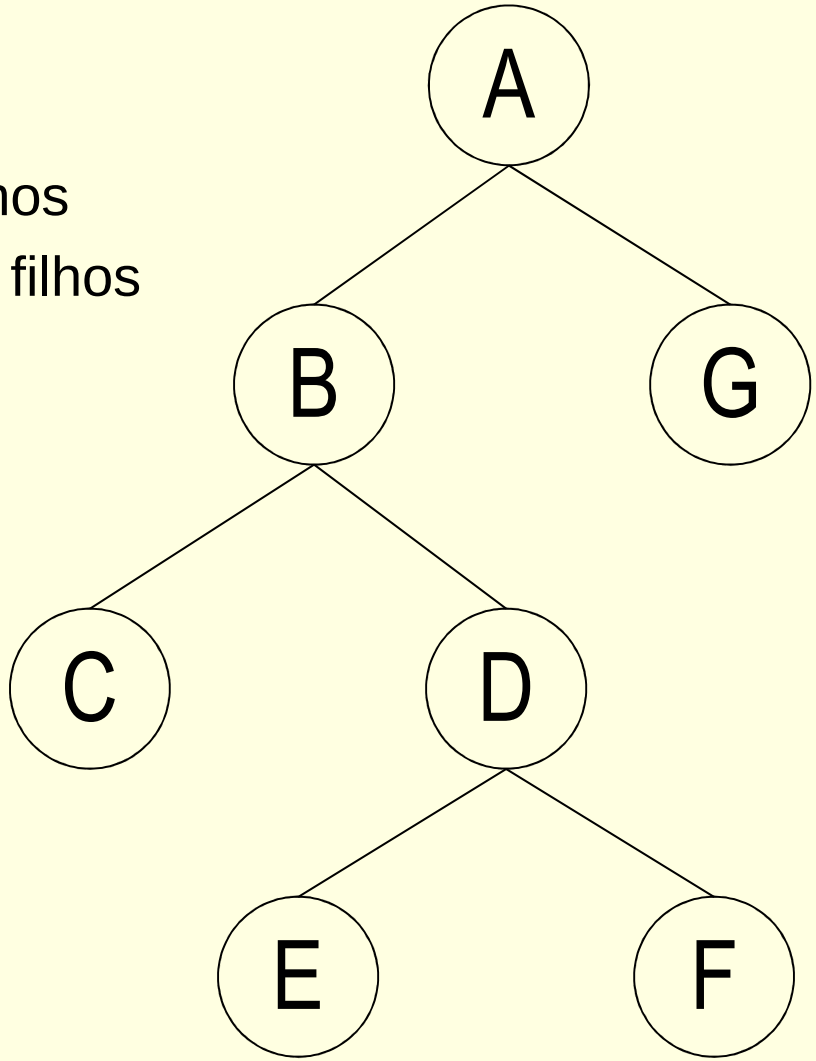
- O desbalanceamento da árvore pode tornar a busca tão ineficiente quanto a busca seqüencial (no pior caso)
  - O(N)

- Solução?

  Balanceamento da árvore quando necessário!

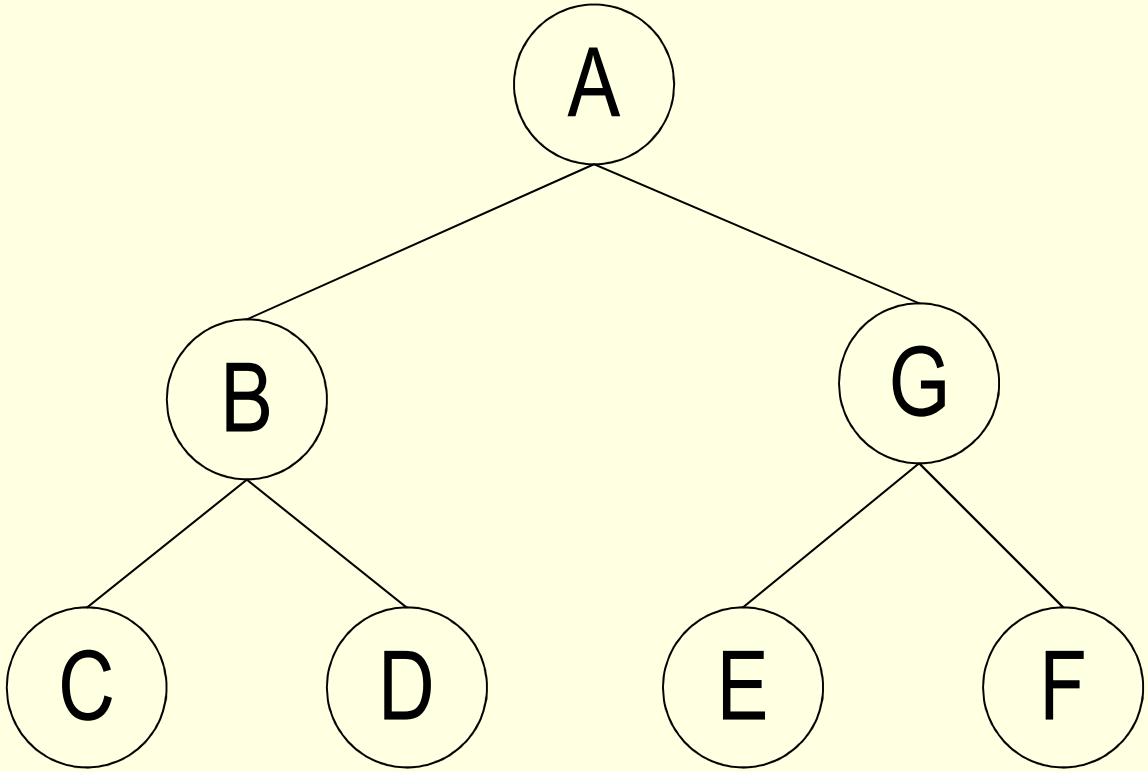
Conceitos

- Árvore estritamente binária
  - Os nós tem 0 ou 2 filhos
    - Todo nó interno tem 2 filhos
    - Somente as folhas têm 0 filhos



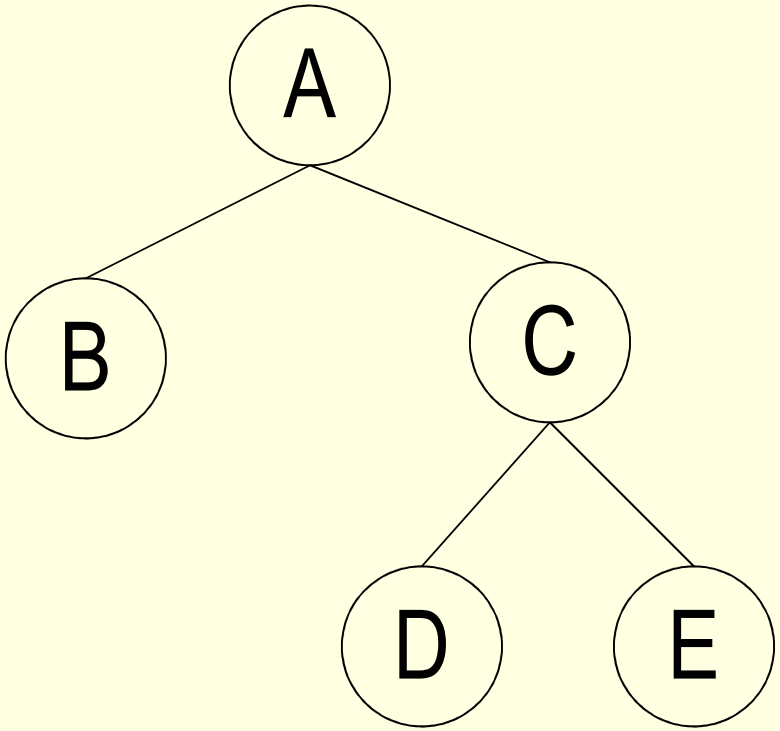
Conceitos

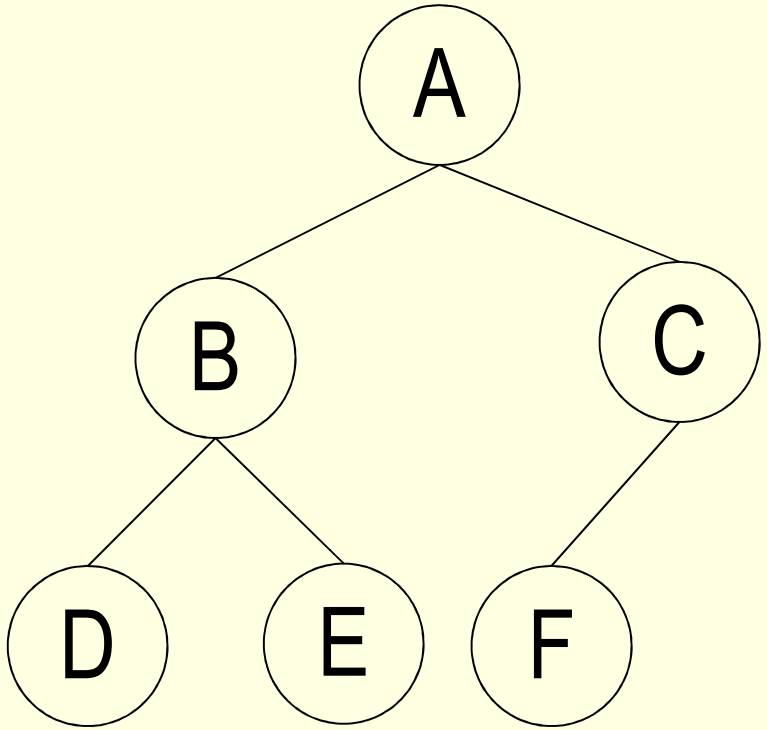
- Árvore binária completa (ou cheia)
  - Árvore estritamente binária
  - Todos os nós folha no mesmo nível



Conceitos

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





Conceitos

- Uma árvore binária perfeitamente balanceada é aquela cujo número de nós de suas subárvores esquerda e direita diferem em 1, no máximo
  - Toda árvore binária perfeitamente balanceada é balanceada
    - Vale o inverso?

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

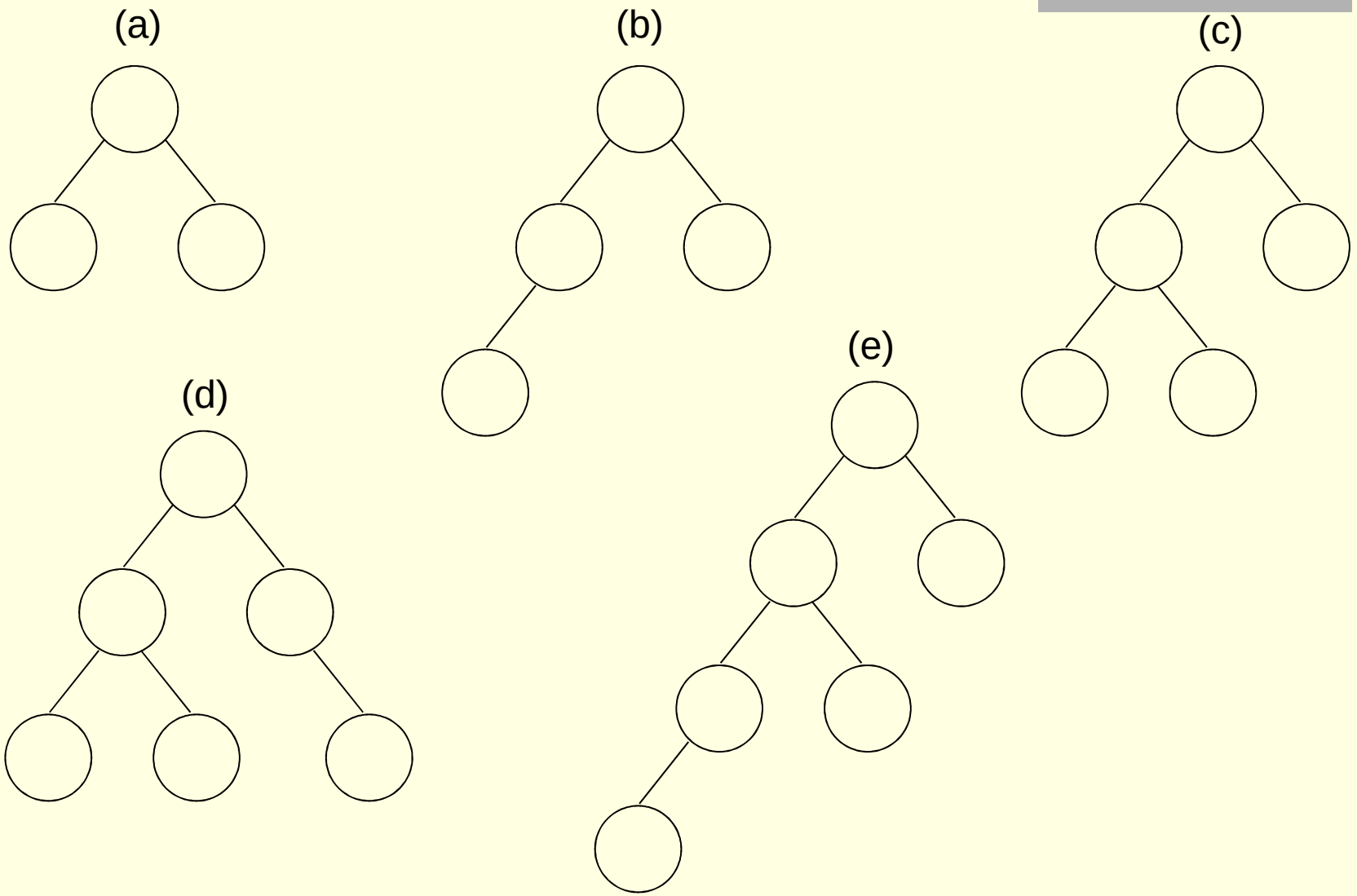
Por que não se exige que seja perfeitamente balanceada?
Custo alto...certos casos exigem movimentar a árvore inteira

Métodos de Balanceamento

- Há duas categorias: dinâmico e global (ou estático).
- O rebalanceamento **dinâmico** mantém a árvore balanceada toda vez que é um nó é inserido ou removido.
  - AVL é o melhor exemplo
- O **global** permite a árvore crescer sem limites e somente faz o balanceamento quando tal necessidade é acionada, externamente.
  - Há vários métodos. Vale a pena ver o de Chang & Iyengar, de 1984.
  - Códigos destes rebalanceamentos são mostrados em:

  Binary Search Tree Balancing Methods: A Critical Study (http://paper.ijcsns.org/07_book/200708/20070834.pdf)

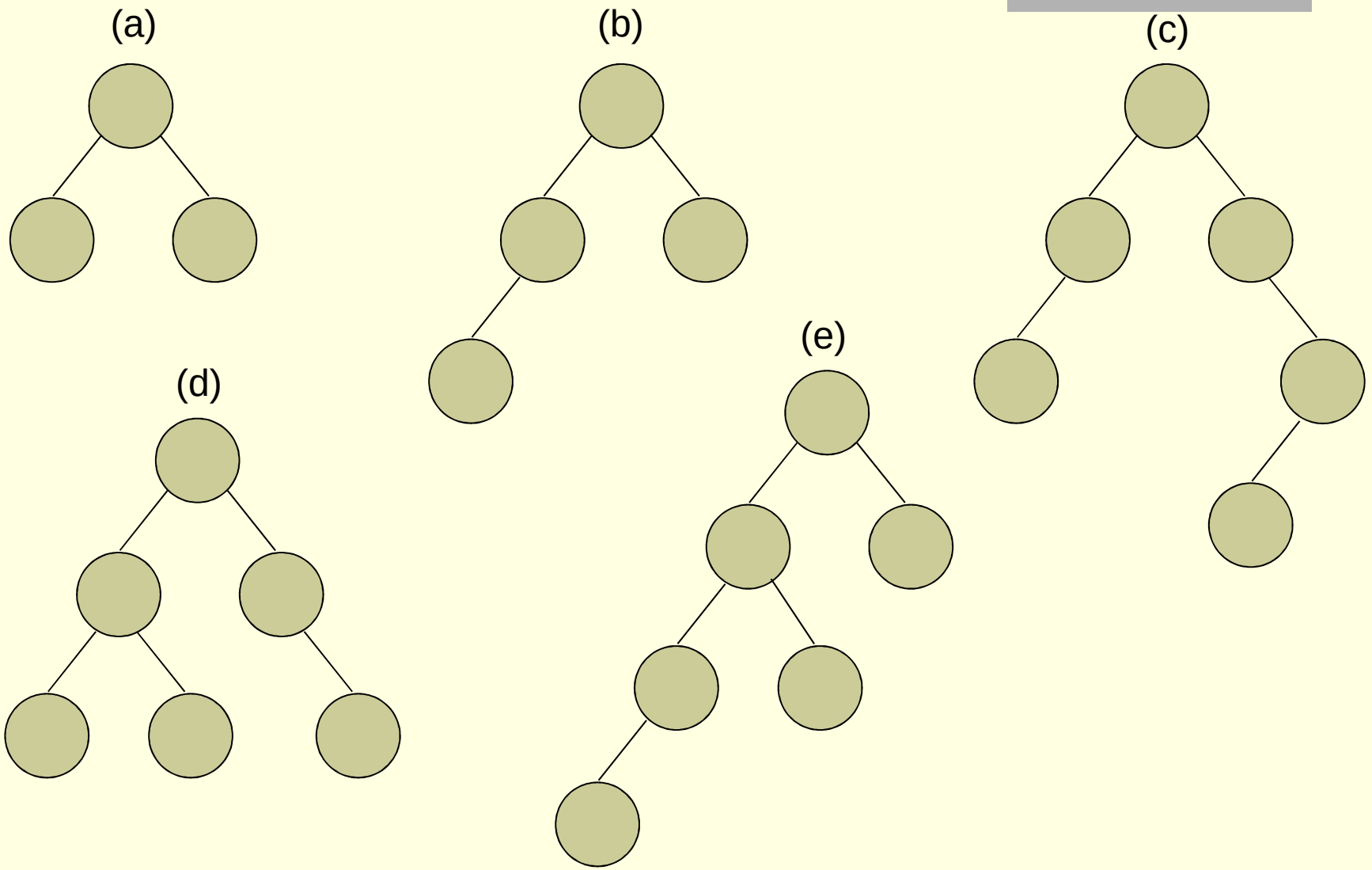
AVL: quem é e quem não é?



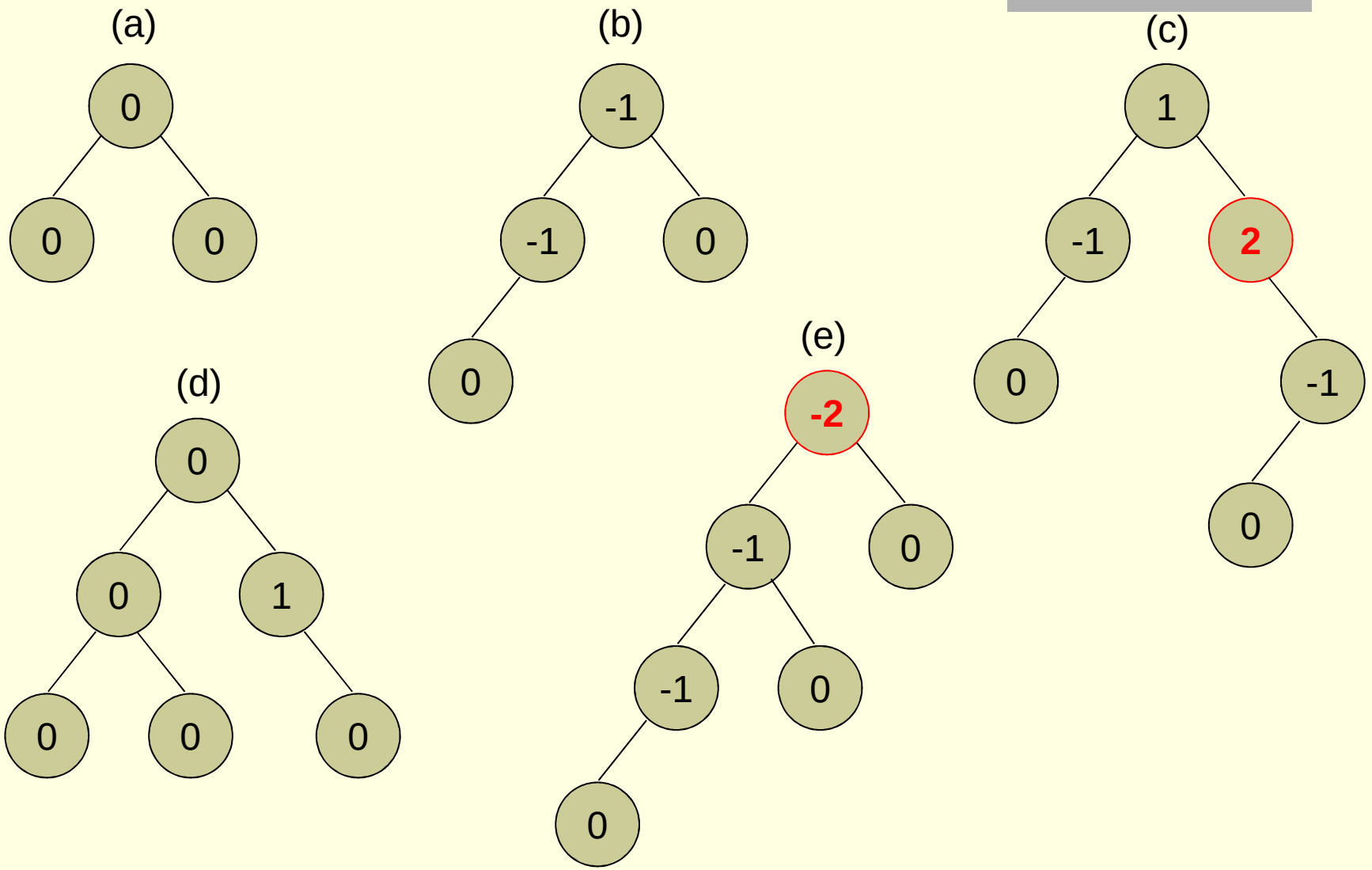
AVL

- 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!

AVL: quem é e quem não é



AVL: quem é e quem não é



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

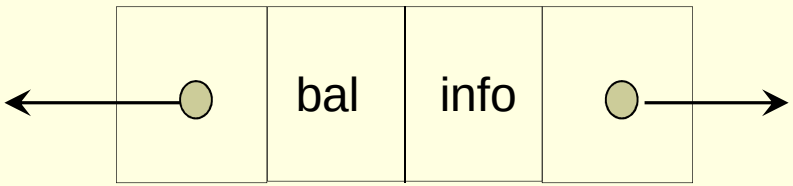
AVL

- 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

AVL – Estrutura de Dados

C

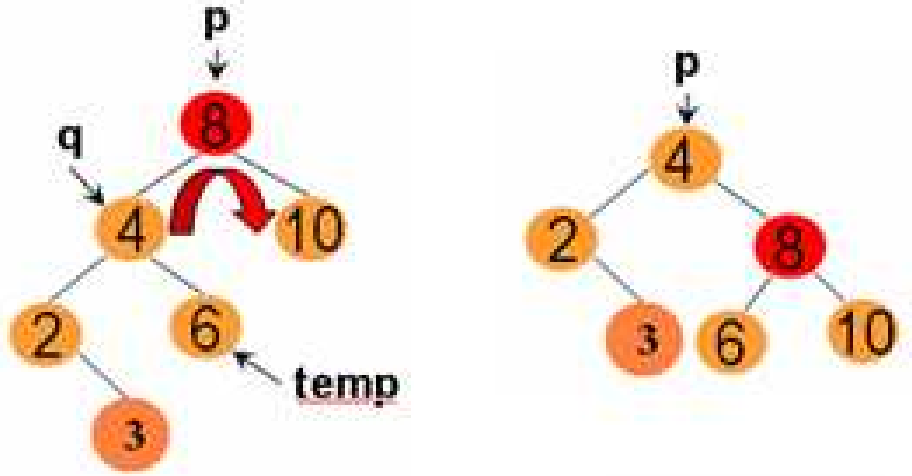
```
typedef  int tipo_elem;
typedef char *tipo_chave;

struct nodetype {
        tipo_chave chave;
        tipo_elem info;
        struct nodetype *left;
        struct nodetype *right;
        int bal;
}

typedef struct nodetype *NODEPTR;
```



AVL: Rotação simples à direita

```
procedure rot_dir(var p: ^no);
var q, temp: ^no;
begin
        q:=p^.esq;
        temp:=q^.dir;
        q^.dir:=p;
        p^.esq:=temp;
        p:=q;
end;
```

p^.bal := 0

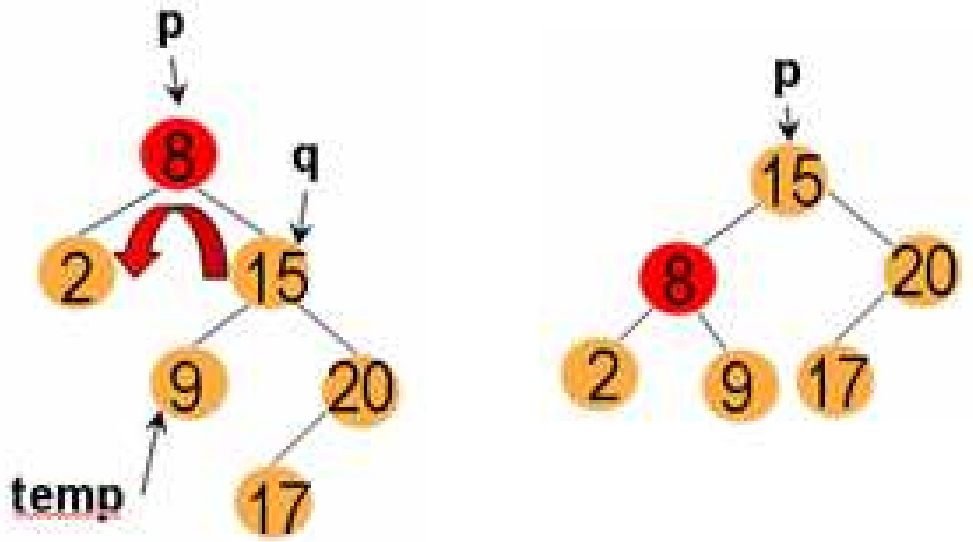



```
void rot_dir (NODEPTR p){
NODEPTR q, temp;
        q = p->esq;
         temp = q->dir;
          q->dir = p;
          p->esq = temp; p->bal = 0;
          p = q;
}
```

AVL: Rotação simples à esquerda

```
procedure rot_esq(var p: ^no);
var q, temp: ^no;
begin
        q:=p^.dir;
        temp:=q^.esq;
        q^.esq:=p;
        p^.dir:=temp;
        p:=q;
end;
```

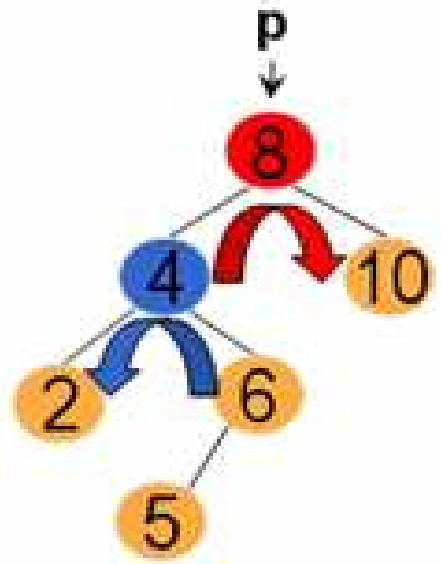
p^.bal := 0




```
void rot_esq (NODEPTR p){
NODEPTR q, temp;
        q = p->dir;
        temp = q->esq;
        q->esq = p;
        p->dir = temp; p->bal = 0;
        p = q;
}
```

Rotação dupla esquerda e direita

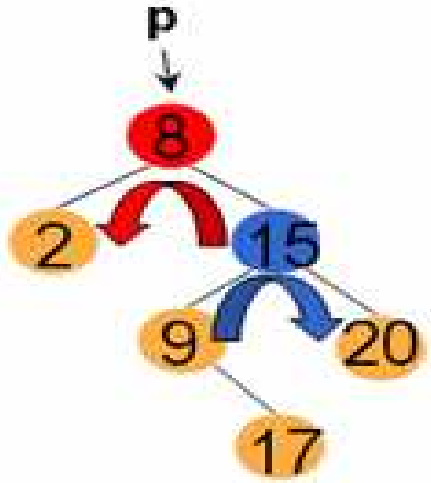
```
procedure rot_esq_dir(var p: ^no);
begin
    rot_esq(p^.esq);
    rot_dir(p);
end;
```




```
void rot_esq_dir(NODEPTR p){
  rot_esq(p->esq);
  rot_dir(p);
}
```

Rotação dupla direita e esquerda

```
procedure rot_dir_esq(var p: ^no);
begin
    rot_dir(p^.dir);
    rot_esq(p);
end;
```




```
void rot_dir_esq(NODEPTR p){
  rot_dir(p->dir);
  rot_esq(p);
}
```

AVL

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

AVL

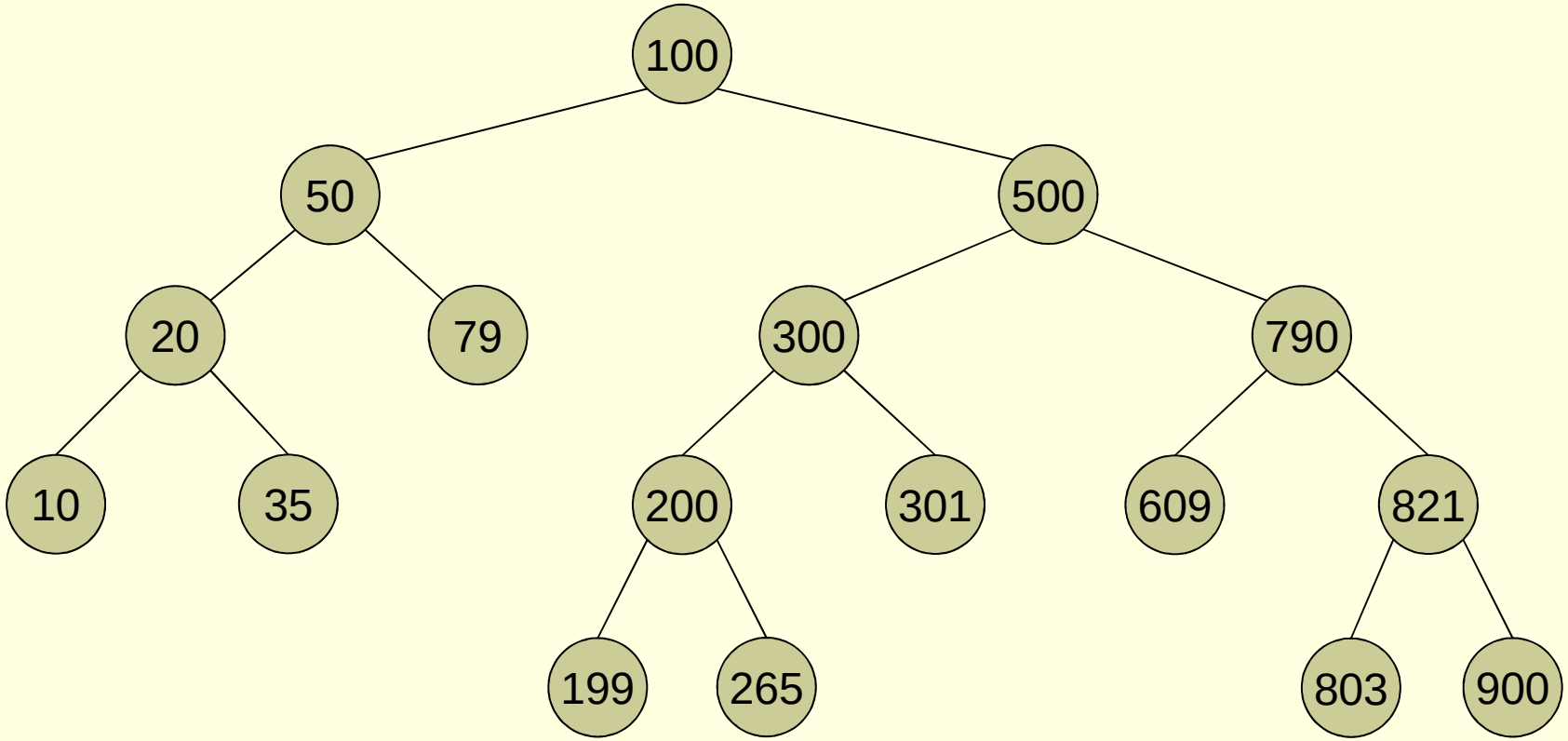
- Exercício
  - Inserir os elementos A, B, C, ..., J em uma árvore e balancear quando necessário

AVL

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

AVL

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



Remoção em AVL

- Remoção é um pouco mais complexo e se divide em duas partes:
- Em primeiro lugar, a remoção de um nó qualquer é substituída pela remoção de uma folha. Para tanto, existem 3 casos possíveis:

1. o nó tem grau zero e, portanto, já é uma folha;

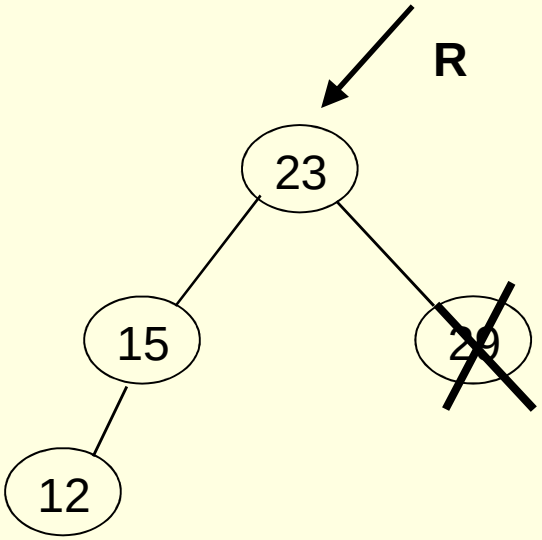
2. o nó tem grau um – pela propriedade AVL, a sua única subárvore é necessariamente constituída por uma folha, cujo valor é copiado para o nó pai; o nó a ser eliminado é a folha da subárvore;

3. o nó tem grau dois – o seu valor é substitído pelo maior valor contido na sua subárvore esquerda (ou o menor valor contido na sua subárvore direita); o nó que continha o menor (ou maior) valor copiado tem necessariamente grau zero ou um, recaindo num dos casos anteriores.

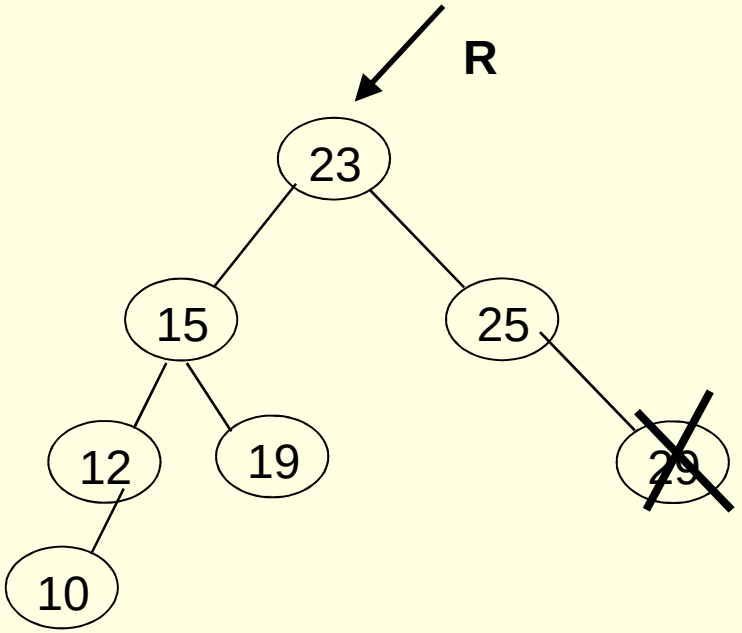
- A segunda parte do algoritmo consiste, portanto, na remoção de uma folha. O processo é semelhante à inserção.

AVL: remoção

- Exemplos



remoção de 29

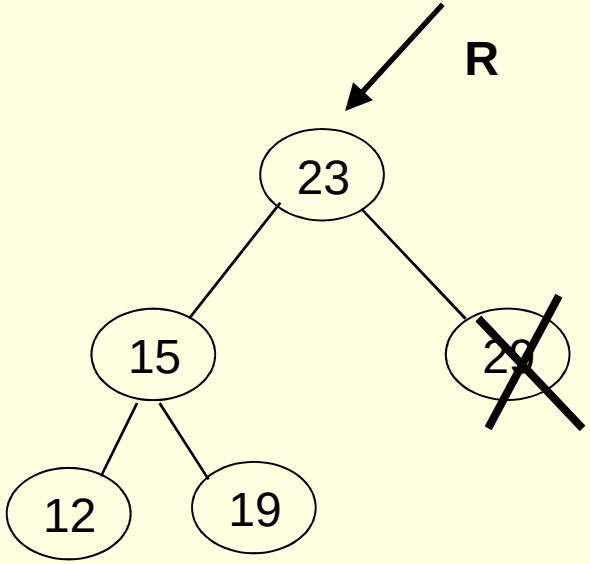


remoção de 29

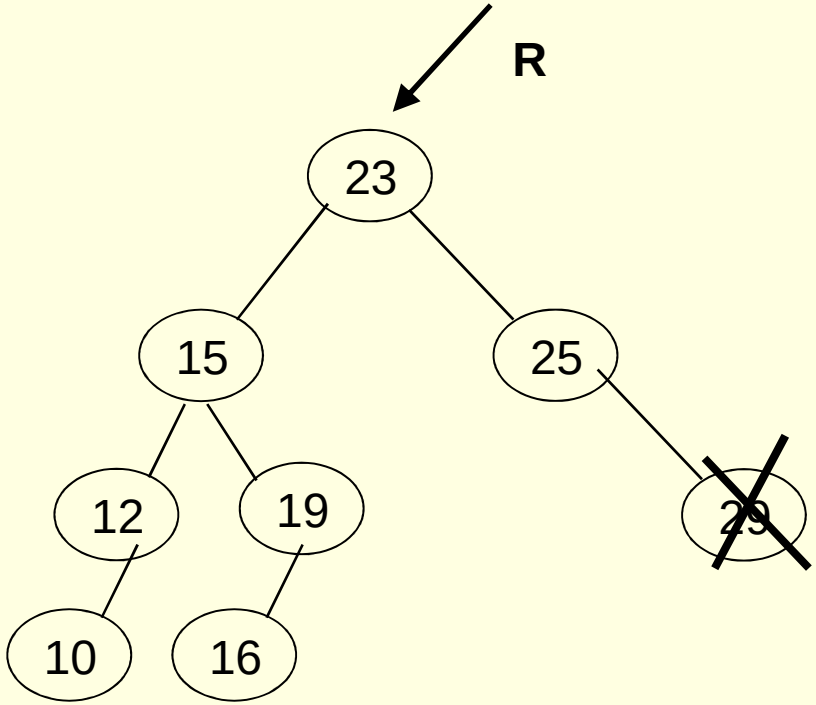
Como balancear?

AVL: remoção

- Exemplos



remoção de 29



remoção de 29

Como balancear?

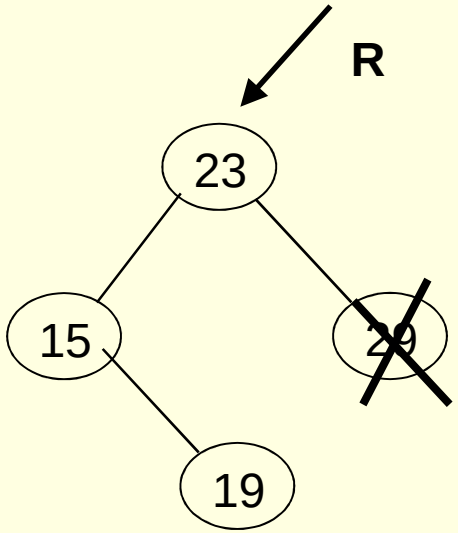
Rotação simples em R

AVL: remoção do nó folha

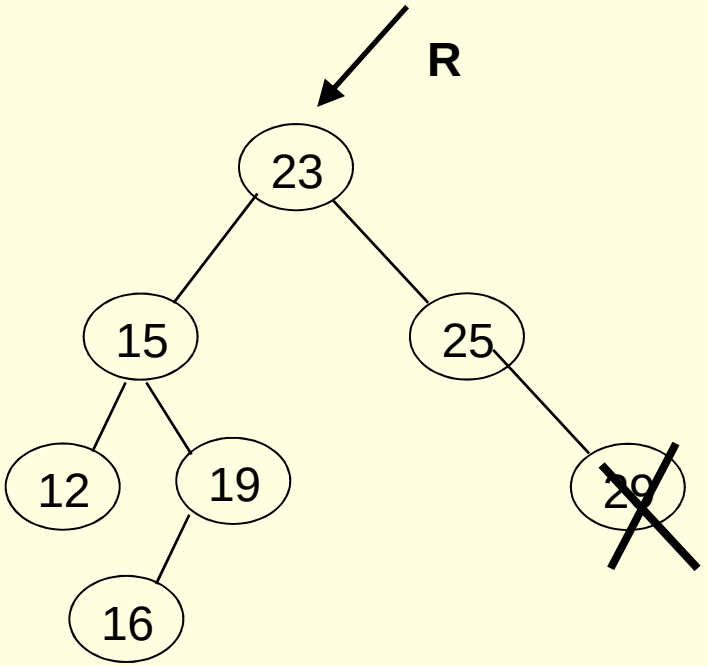
- Primeiro caso
  - Rotação simples em R (FB=2 ou -2) com filho com fator de balanceamento de mesmo sinal (1 ou -1) ou zero
    - Se R negativo, rotaciona-se para a direita; caso contrário, para a esquerda

AVL: remoção do nó folha

- Exemplos



remoção de 29

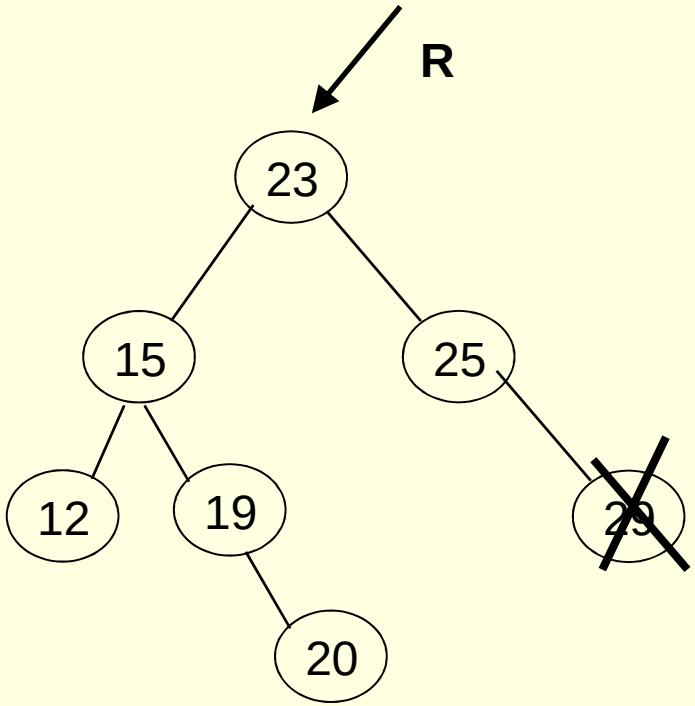


remoção de 29 = inserção de 16

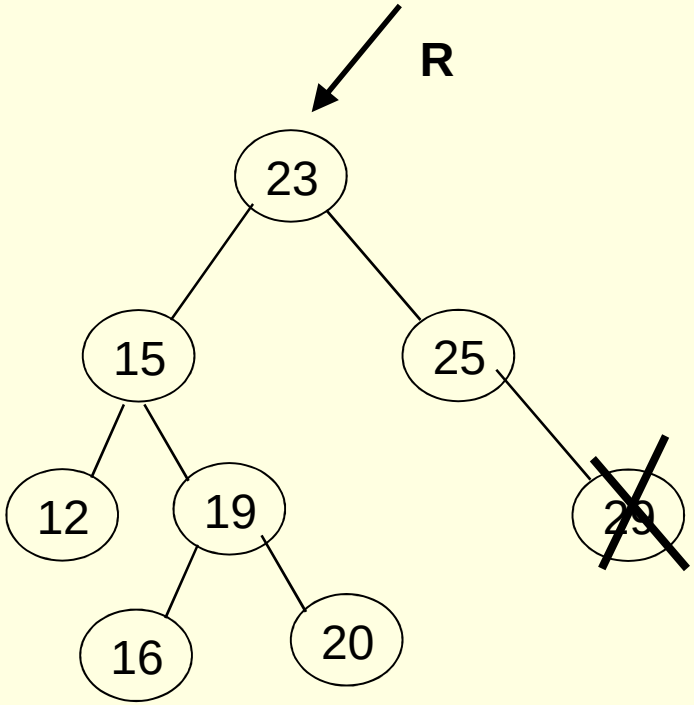
Como balancear?

AVL: remoção nó folha

- Exemplos



remoção de 29 = inserção de 20



remoção de 29 = inserção de 16 ou 20

Como balancear?

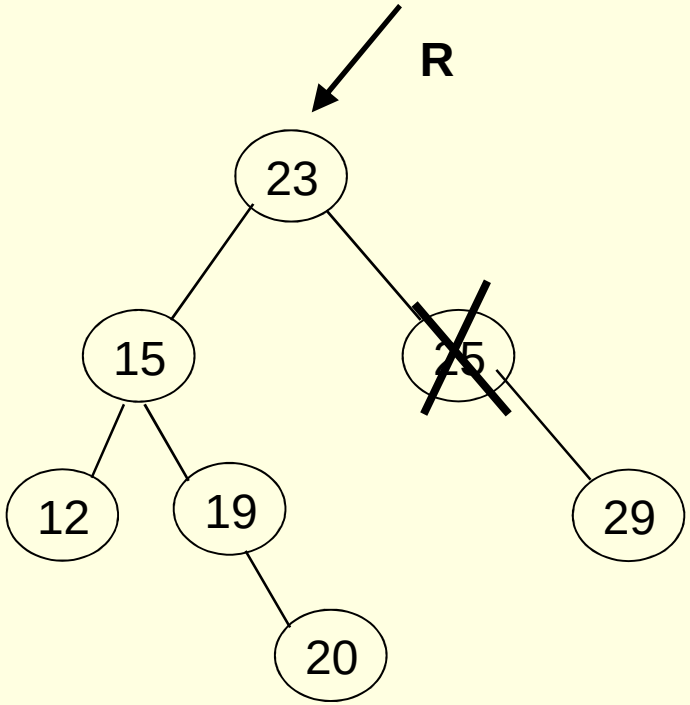
Rotação dupla: filho de R e R

AVL: remoção nó folha

- Segundo caso
  - Rotação dupla quando R (FB=2 ou -2) e seu filho (-1 ou 1) tem fatores de balanceamento com sinais opostos
    - Rotaciona-se o filho para o lado do desbalanceamento do pai
    - Rotaciona-se R para o lado oposto do desbalanceamento

AVL: remoção

- Questão: como remover um nó intermediário em vez de um nó folha?
  - É necessário balancear?



Se nó com grau 1 troca-se pela folha e remove-se a folha.

AVL: remoção

- Questão: como remover um nó intermediário em vez de um nó folha?
  - É necessário balancear?

Se nó com grau 2, troca-se pelo maior da sub-árvore esquerda ou menor da sub-árvore direita e depois remove-se a folha trocada.

AVL

- Exercício para casa
  - Implementar sub-rotina de remoção de elemento de uma AVL

Resumo ABB

- Boa opção como ED para buscas de chaves, SE a árvore é balanceada => tempo proporcional a $\log_2 n$.
- Inserções (como Folhas) e Eliminações (mais complexas) causam desbalanceamento.
- Inserções: melhor se aleatórias (não ordenadas) para evitar linearizações.
- Para manter o balanceamento:
  - Balanceamento global
  - Árvores AVL

Exercícios

1. Duas ABBs são SIMILARES se possuem a mesma distribuição de nós (independente dos valores nos mesmos). Em uma definição mais formal, duas ABBs são SIMILARES se são ambas vazia, ou se suas subárvores esquerdas são similares e suas subárvores direitas também são similares
   - Implemente a sub-rotina que verifica se duas ABBs são similares

Exercícios

2. Duas ABBs são IGUAIS se são ambas vazias ou então se armazenam valores iguais em suas raízes, suas subárvores esquerdas são iguais e suas subárvores direitas são iguais
   - Implemente a sub-rotina que verifica se duas ABBs são iguais

Exercícios

3. Uma ABB é estritamente binária se todos os nós da árvore tem 2 filhos ou nenhum filho
   - Implemente uma sub-rotina que verifica se uma ABB é estritamente binária

Exercícios

4. Implemente uma sub-rotina para verificar se uma árvore binária é uma ABB