Árvores AVL

Algoritmos e Estruturas de Dados

Prof. Luciano Demétrio Santos Pacífico
{luciano.pacifico@ufrpe.br}

Conteúdo

- Introdução
- Árvores Balanceadas
- Árvores AVL
- Busca, Inserção e Remoção

Introdução

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Árvores Degeneradas

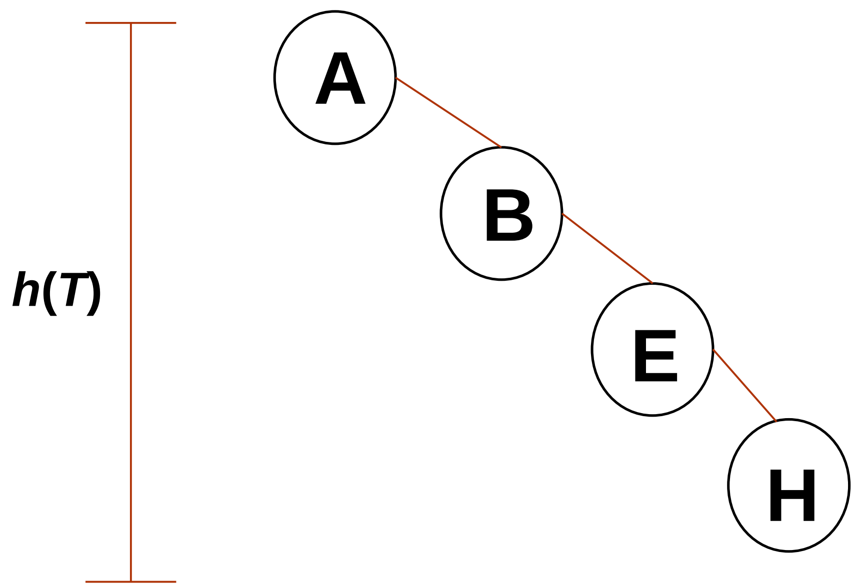
- Um aspecto fundamental do estudo de árvores de busca é, naturalmente, o custo de acesso a uma chave desejada.

- Com o intuito de minimizar esse custo, foram desenvolvidas as árvores binárias.

- Em uma árvore binária completa não vazia, o custo de acesso a uma chave está relacionado à altura da árvore, sendo o mesmo, no pior caso, da ordem de O(log *n*).

Árvores Degeneradas

- Porém, como visto na aula anterior, dependendo da ordem na qual remoções e inserções ocorrem em uma árvore binária, a mesma pode degenerar-se.



Árvores Degeneradas

- Deve-se fazer uso de mecanismos que mantenham o custo de acesso de uma árvore binária na mesma ordem de grandeza de uma árvore ótima, ou seja, O(log *n*).

- Esse custo deve ser mantido ao longo de toda a utilização da estrutura, mesmo após operações de inserção e remoção.

- Para isso, deve-se alterar periodicamente a estrutura, de forma que os custos de operações na mesma mantenham-se na ordem O(log *n*).

- Uma estrutura com essas características é dita **balanceada**.

Árvores Balanceadas

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Árvores Balanceadas

- Árvores completas são aquelas que minimizam o número de operações efetuadas no pior caso para uma busca.

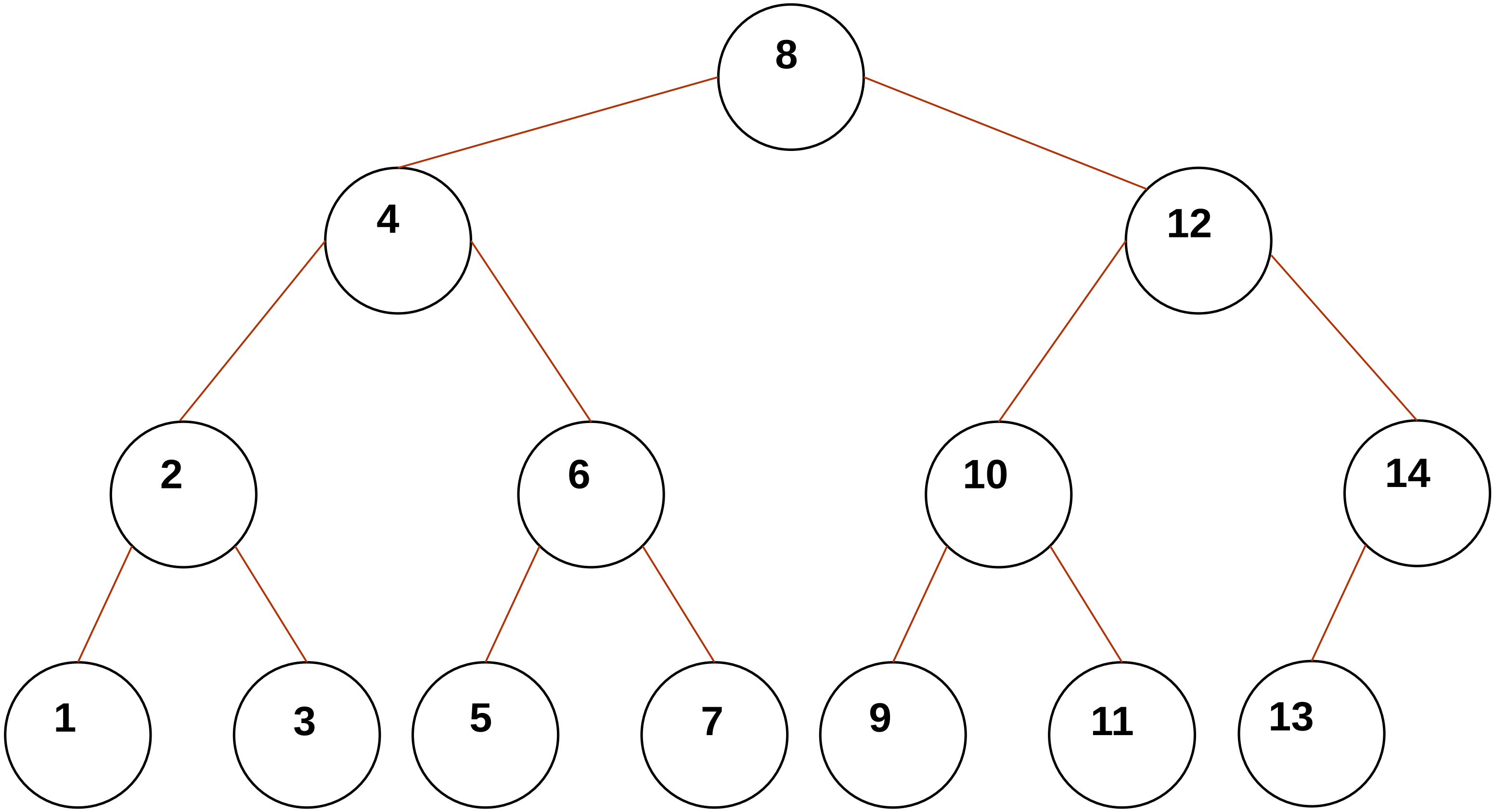
- Do ponto de vista de estruturas dinâmicas, o uso de árvores completas é desaconselhado.

- Após um número de inserções ou remoções, a árvore poderia degenerar-se.

- Um algoritmo poderia ser elaborado para tentar tornar a árvore novamente completa após inserções ou remoções.

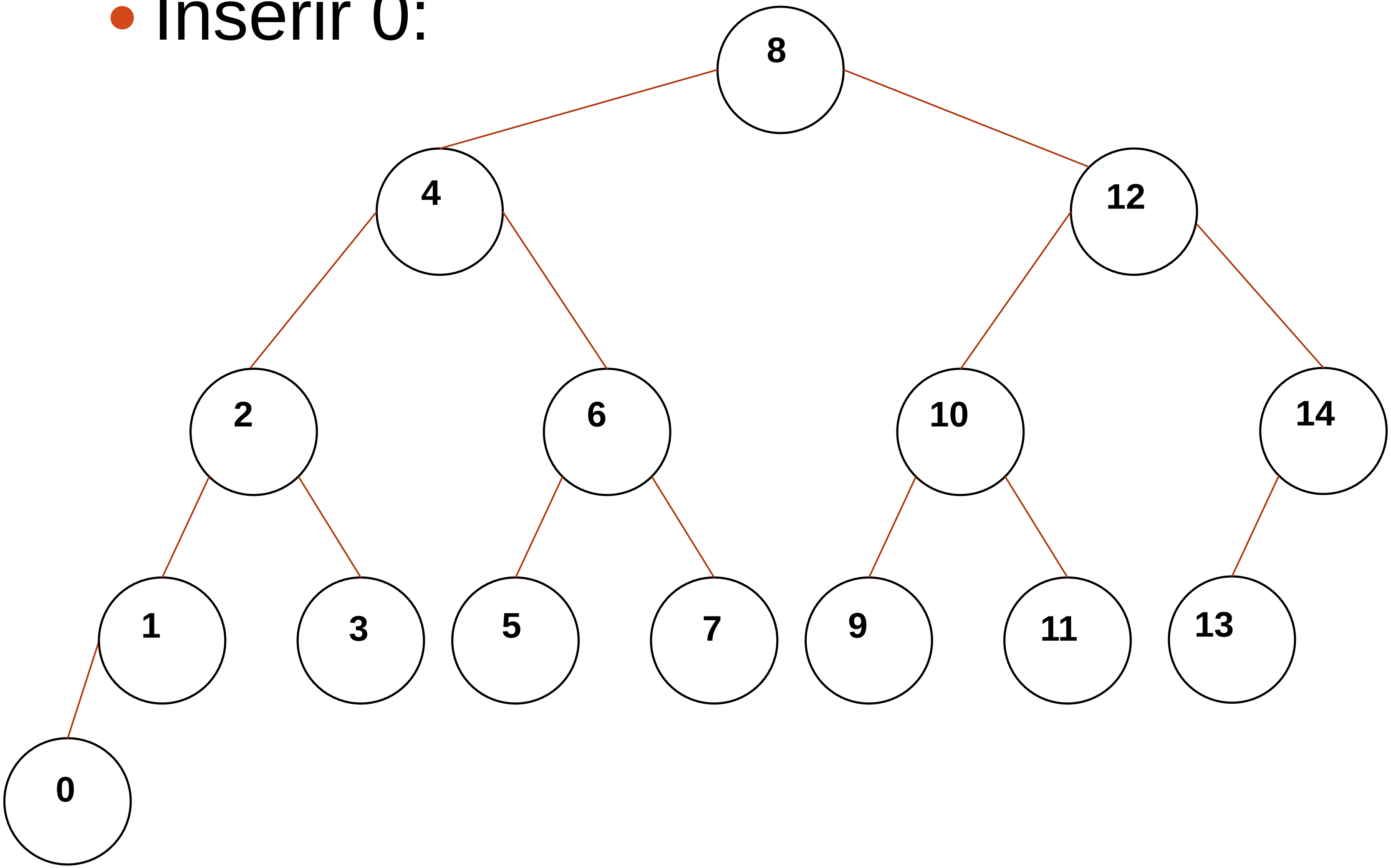
Árvores Balanceadas

- Inserir 0:



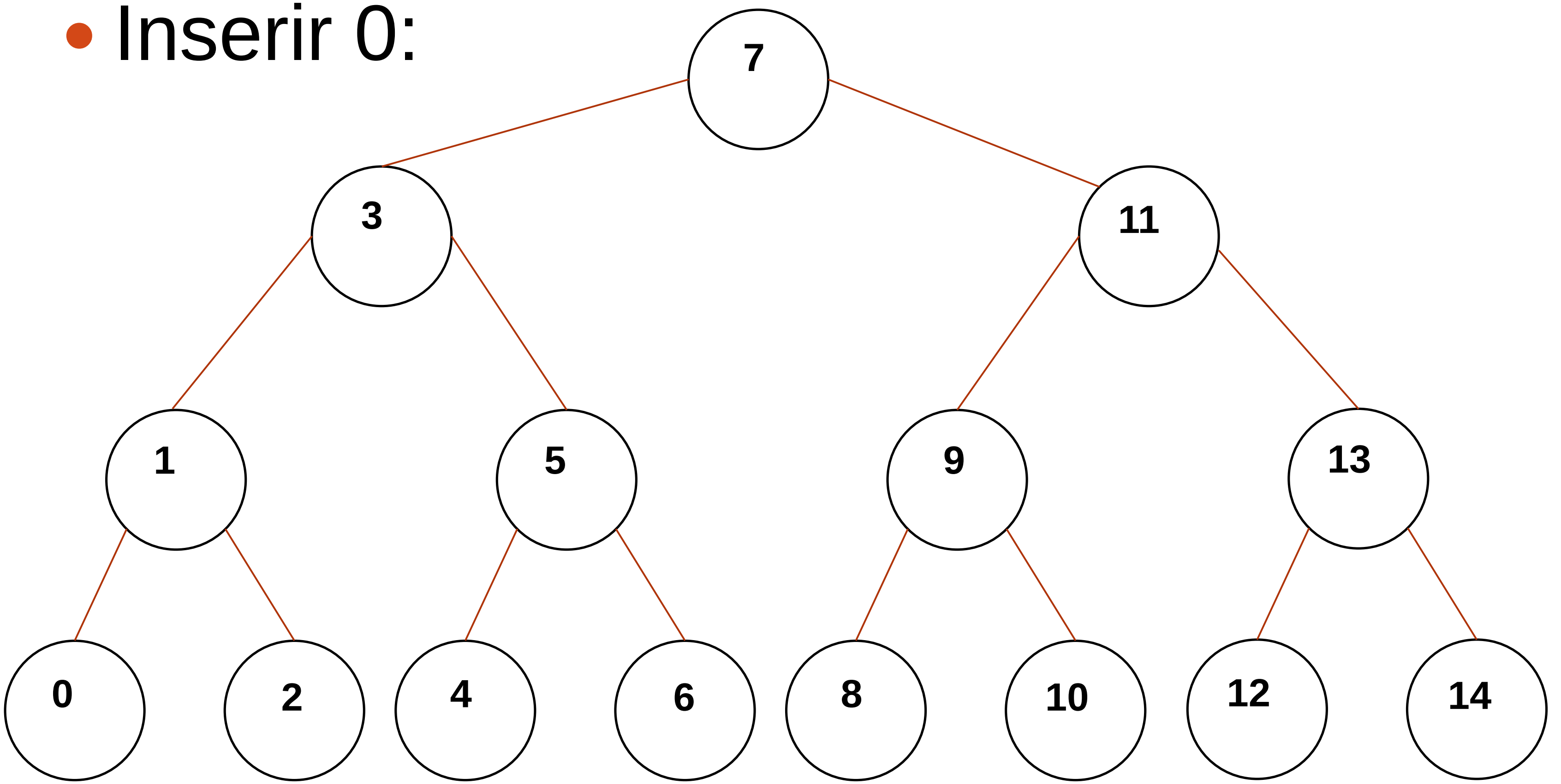
Árvores Balanceadas

- Inserir 0:



Árvores Balanceadas

- Inserir 0:



Árvores Balanceadas

- O custo de $\Omega(n)$ para o reestabelecimento da árvore é excessivo, considerando que operações como inserções e remoções seriam realizadas em $O(\log n)$ passos.

- Por esse motivo, árvores completas e busca binária não são recomendadas para aplicações dinâmicas.

Árvores Balanceadas

- Uma alternativa é exigir que a altura da árvore seja da mesma ordem de grandeza que a de uma árvore completa com o mesmo número de nós, ou seja, O(log *n*).

- Além disso, é desejável que esta propriedade se estenda a todas as subárvores: cada subárvore que contém *m* nós deve possuir altura igual a O(log *m*).

- Uma árvore que satisfaça essa condição é denominada **balanceada**.

Árvores Balanceadas

- Intuitivamente, a ideia é utilizar árvores cuja altura, embora possa ser maior que a mínima $1 + \lfloor \log n \rfloor$, ainda assim não ultrapasse O(log *n*).

- Como a forma de uma árvore balanceada é menos rígida do que a de uma árvore completa, seu rebalanceamento, ou seja, o restabelecimento das condições de balanceamento, torna-se mais fácil.

Árvores AVL

UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Árvores AVL

- O fator de balanceamento (ou fator de equilíbrio) de um nó em uma árvore binária *T* é definido como sendo igual a diferença entre as alturas de suas subárvores esquerda e direita.

- Para um nó *v* qualquer, seu fator de balanceamento é dado pela equação abaixo:

$$\boldsymbol{fb(v) = \text{altura}(v.dir) - \text{altura}(v.esq)}$$

- **O fator de balanceamento de uma folha é zero**.

Árvores AVL

- Uma árvore binária *T* é denominada AVL quando, para qualquer nó de *T*, as alturas de suas subárvores esquerda e direita diferem, em módulo, de até uma unidade.

- Nesse caso, o nó é dito **regulado**.

- Um nó que não satisfaça essa condição de altura é dito *desregulado*, e uma árvore que contenha um nó nessas condições é também **desregulada**.

- Toda árvore completa é AVL, mas a recíproca nem sempre é verdadeira.

Árvores AVL

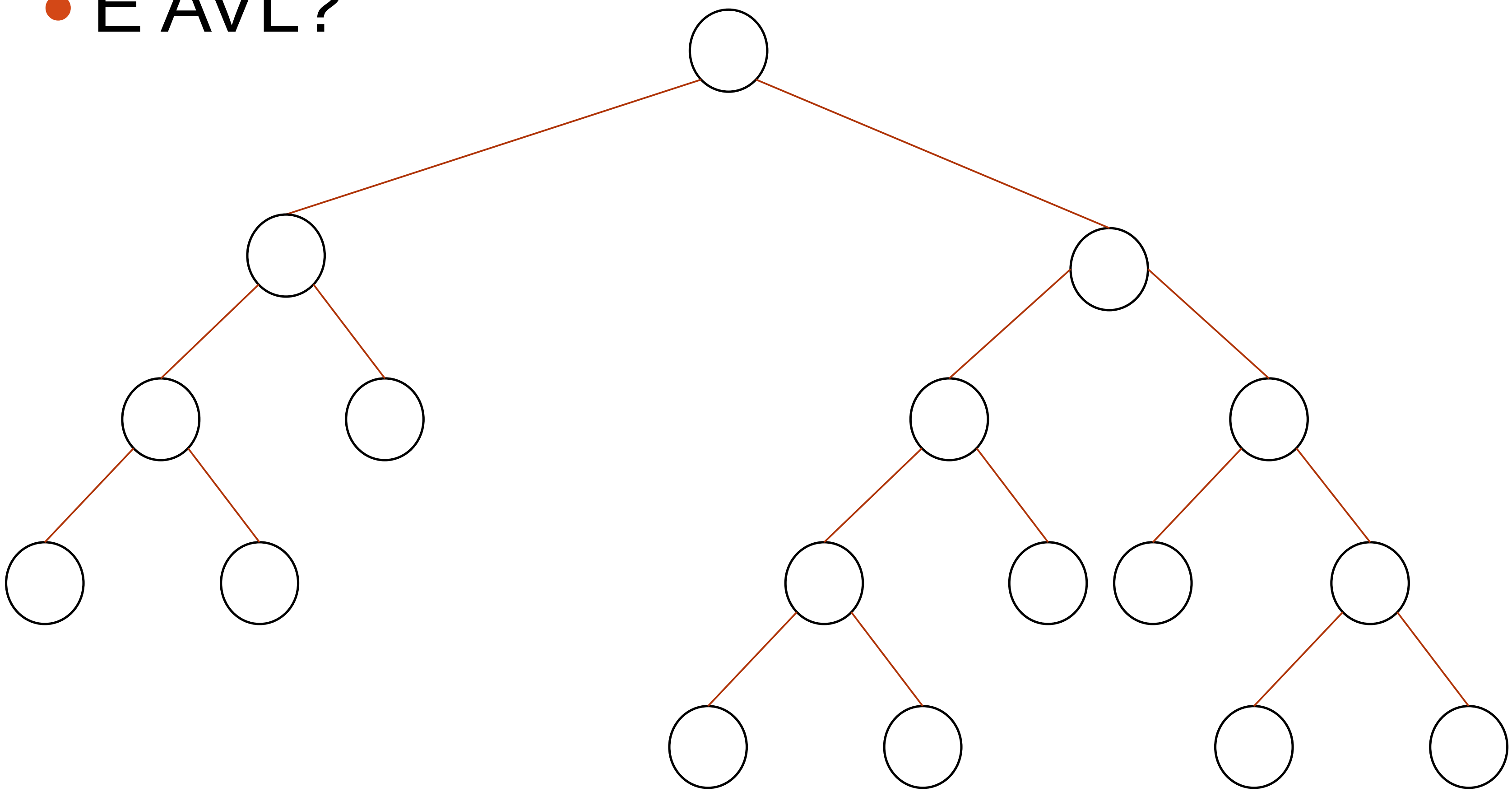
- Em outras palavras:

Para todo nó *v* em uma árvore AVL, a altura das duas subárvores, esquerda e direita, satisfazem:

$$\boldsymbol{fb(v) = |\mathrm{altura}(v.dir) - \mathrm{altura}(v.esq)| \leq 1}$$

Exemplo

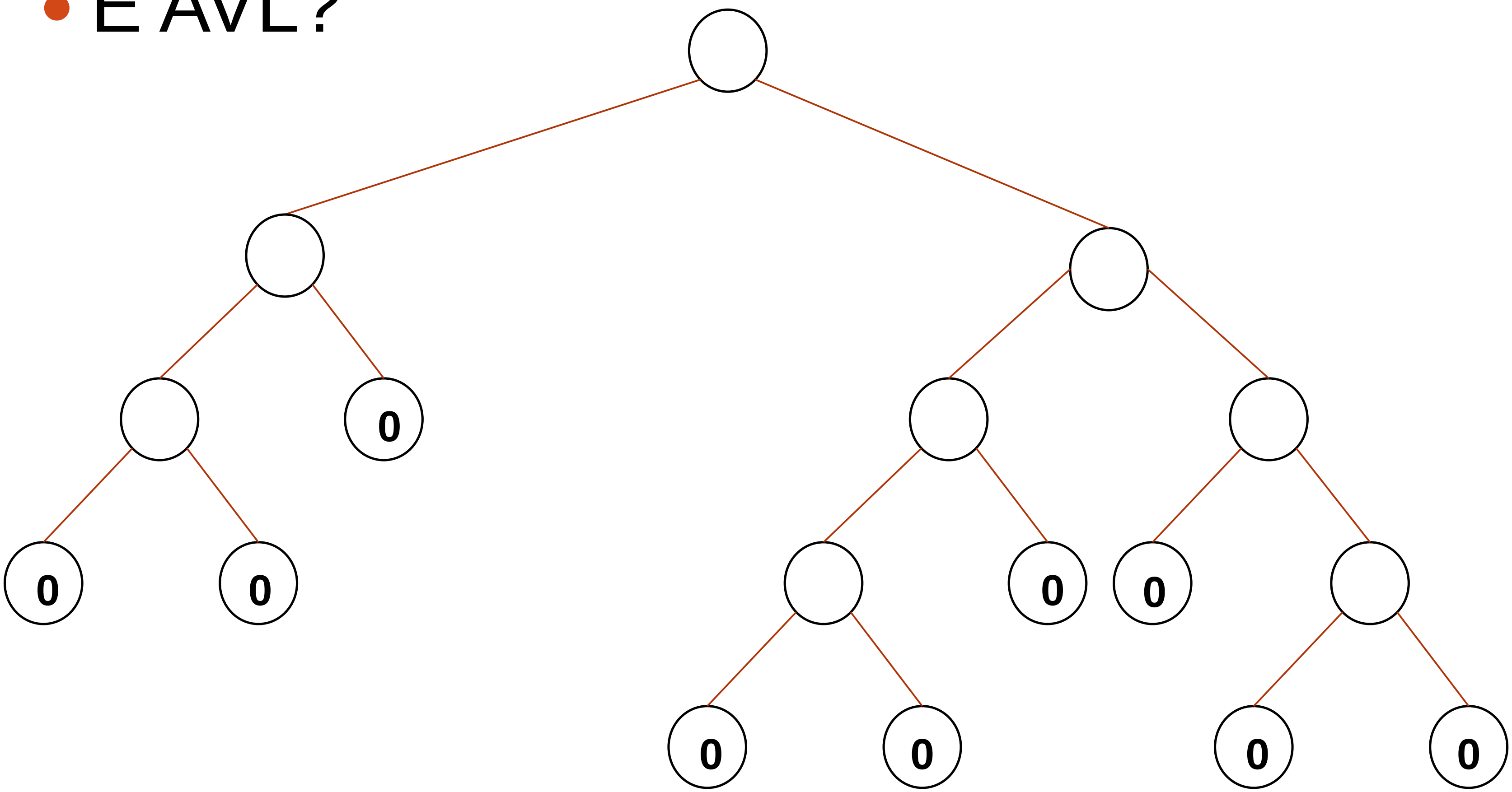
- É AVL?



Exemplo

- É AVL?

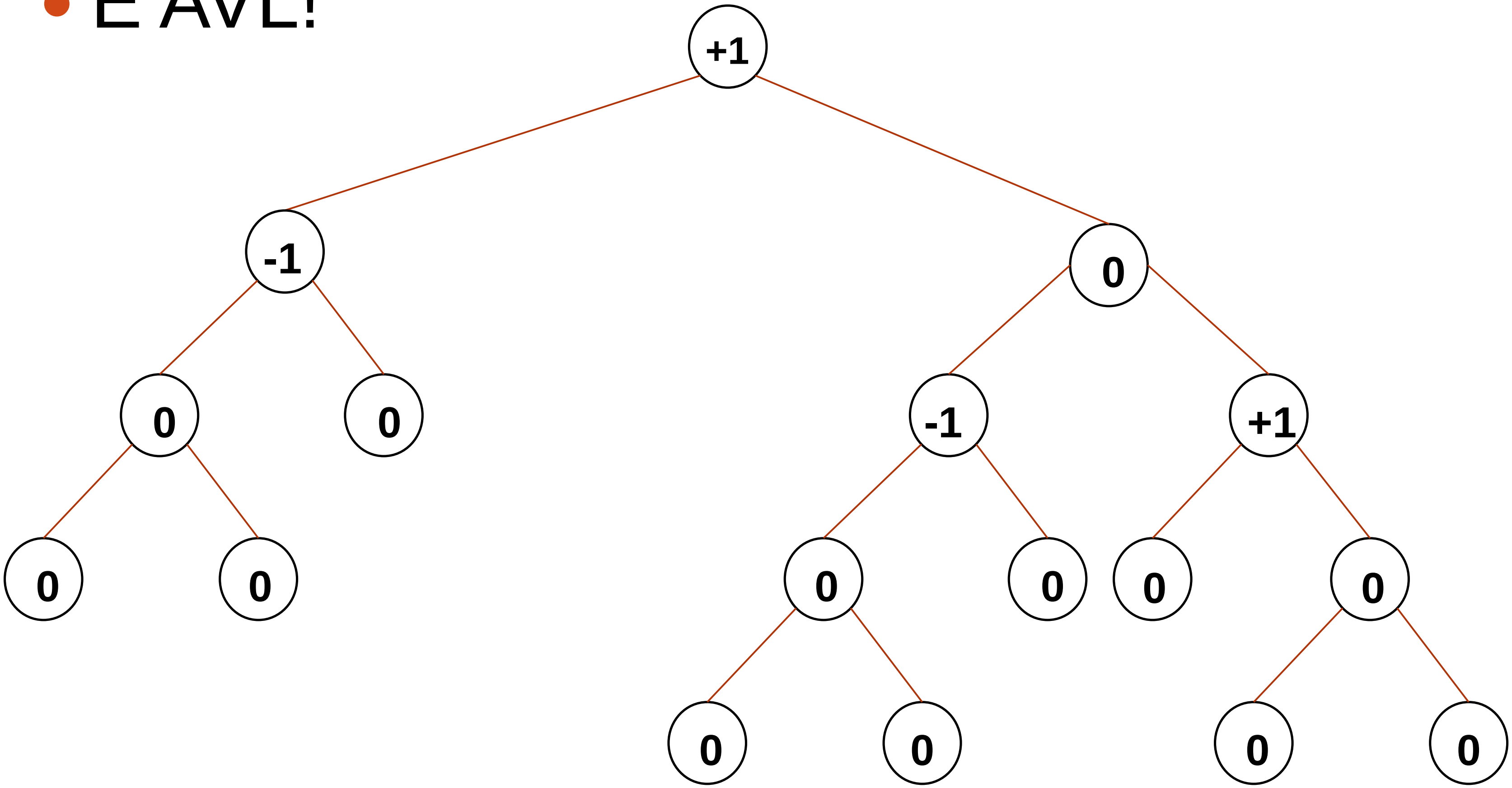
$fb(v) = |altura(v.dir) - altura(v.esq)| \leq 1$



Exemplo

$fb(v) = |altura(v.dir) - altura(v.esq)| \leq 1$

- É AVL!



Busca, Inserção e Remoção

UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Árvores AVL – Estruturas Básicas

- Por uma questão de facilidade de acesso, armazenaremos em um nó de uma árvore AVL informações sobre a altura de suas subárvores.

```
registro NoArvoreAVL
    chave:inteiro,
    altura:inteiro,
    alturaEsquerda:inteiro,
    alturaDireita:inteiro,
    fatorBalanceamento:inteiro,
    pai:NoArvoreAVL,
    esquerda:NoArvoreAVL,
    direita:NoArvoreAVL
```

```
registro ArvoreAVL
    raiz:NoArvoreAVL
```

Busca em Árvores AVL

- O procedimento de **busca** em árvore AVL é idêntico ao procedimento usado em árvores binárias de busca não-balanceadas.

- A diferença é que no caso das árvores AVL, dado o fato de que **a estrutura estará balanceada**, garante-se que o custo do procedimento de busca será da ordem de O(log *n*).

Inserção em Árvores AVL

- Seja *T* uma árvore AVL, na qual serão efetuadas inclusões de nós.

- Para que *T* se mantenha AVL após as inclusões, ou seja, mantenha-se balanceada, é preciso efetuar operações de restabelecimento da regulagem de seus nós, quando necessário.

- É preciso verificar após cada inserção se algum nó tornou-se desregulado, ou seja, se a diferença de altura entre suas subárvores tornou-se maior que 1.

- Em caso afirmativo, transformações serão aplicadas com o intuito de regular tal nó.

Inserção em Árvores AVL

- Suponha que o nó *q* foi incluído na árvore *T*.
- Se após a inclusão todos os nós mantiveram-se regulados, então *T* manteve-se AVL.
- Caso contrário, seja *p* o nó mais próximo às folhas de *T* que se tornou desregulado.
- Observe que não há ambiguidade na escolha de *p*, pois qualquer subárvore de *T* que se tornou desregulada após a inclusão de *q* deve necessariamente conter *p*.

Inserção em Árvores AVL

- Então *p* se encontra no caminho de *q* à raiz de *T*, e sua escolha é única.

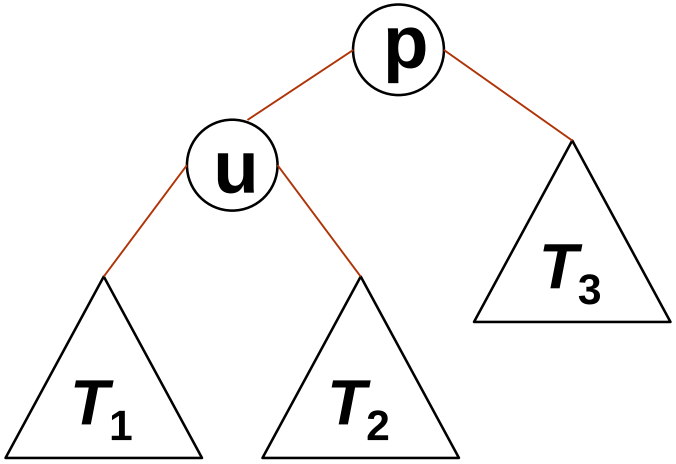
- Sejam h(*p*.*esq*) e h(*p*.*dir*) as alturas das subárvores esquerda e direita de *p*, respectivamente, temos que |h(*p*.*dir*)-h(*p*.*esq*)|>1, de onde concluímos que |h(*p*.*dir*)-h(*p*.*esq*)|=2, tendo em vista que *T* era AVL antes da inclusão de *q*.

- Os casos de estudo possíveis neste contexto serão apresentados.

Caso 1

- Caso 1: h(*p.esq*)>h(*p.dir*)
- *q* pertence à subárvore esquerda de *p*. Além disso, *p* possui o filho esquerdo $u \neq q$.
- Caso contrário, *p* não estaria desregulado.
- Sabe-se ainda que h(*u.esq*) ≠ h(*u.dir*).
- O caso acima pode ser ramificado em dois subcasos.

Caso 1.1



Caso 1.1

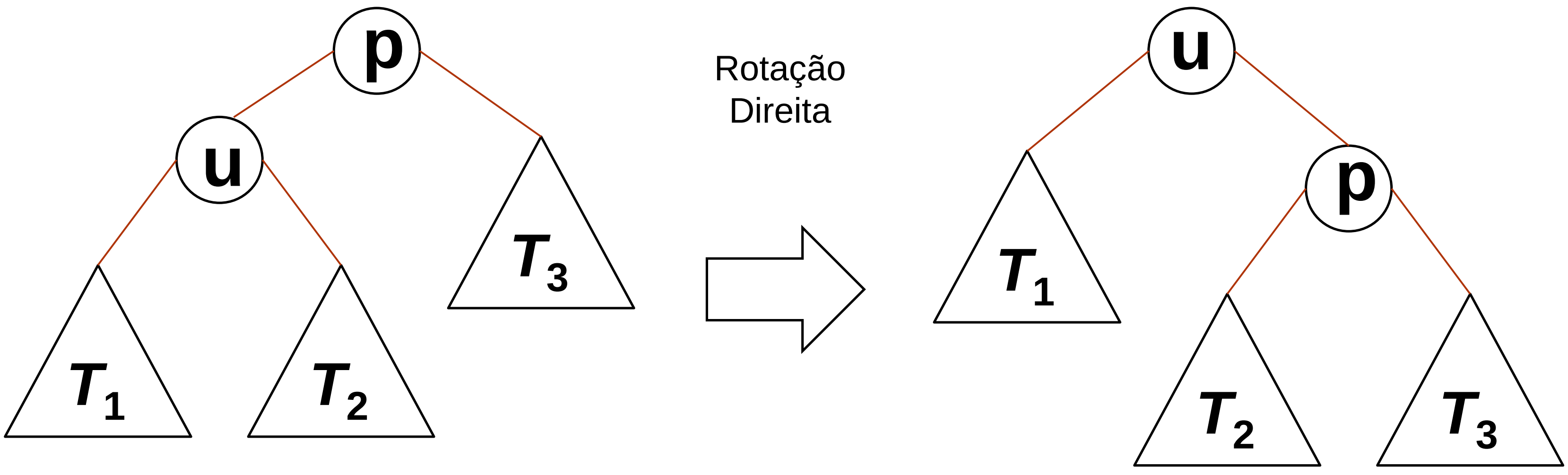
- Caso 1.1: h(*p.esq*)>h(*p.dir*) e h(*u.esq*)>h(*u.dir*)

- $q \in T_1$

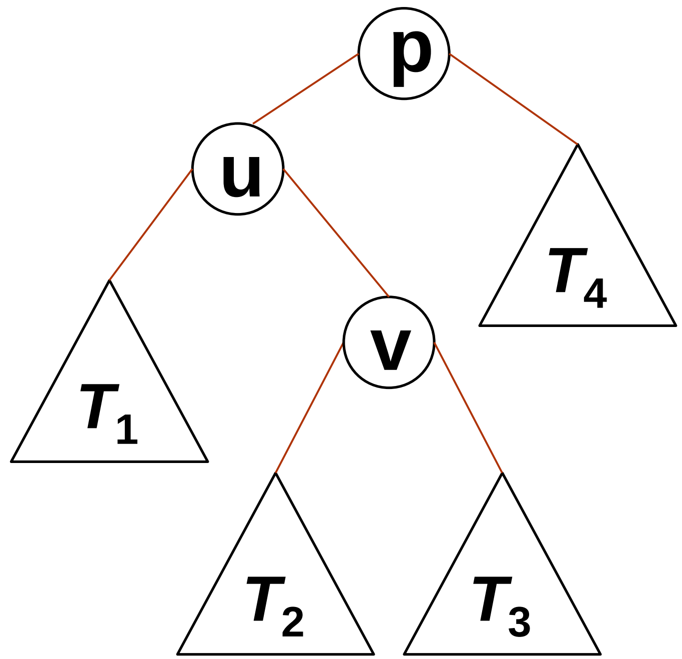
- $h(T_1) - h(T_2) = 1$ e $h(T_2) = h(T_3)$

- A aplicação da **rotação direita** à raiz *p* transformará a subárvore considerada em uma árvore novamente regulada.

Rotação Direita



Caso 1.2



Caso 1.2

- Caso 1.2: h(*p.esq*)>h(*p.dir*) e h(*u.esq*)<h(*u.dir*)

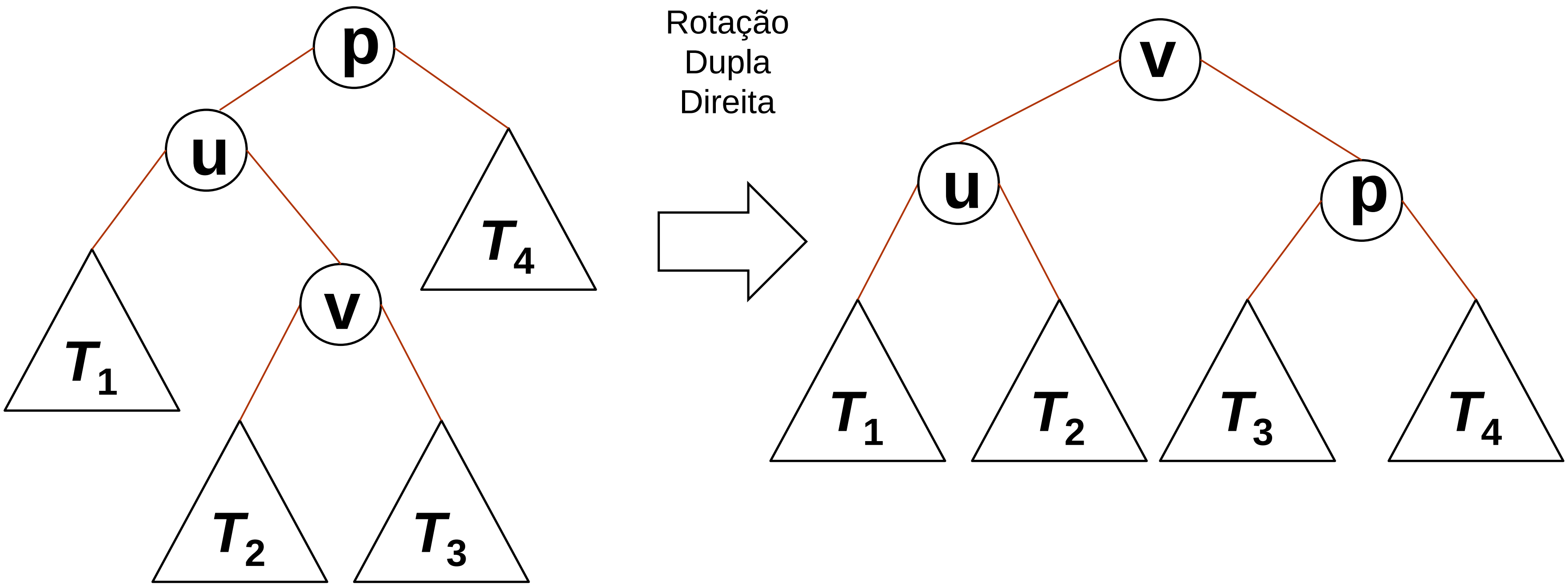
- *u* possui o filho direito *v*.

- Temos que:

$$|h(T_2) - h(T_3)| \leq 1 \text{ e } \max\{h(T_2), h(T_3)\} = h(T_1) = h(T_4)$$

- Aplica-se a **rotação dupla direita** à raiz *p*, restabelecendo sua regulagem.

Rotação Dupla Direita



Caso 2

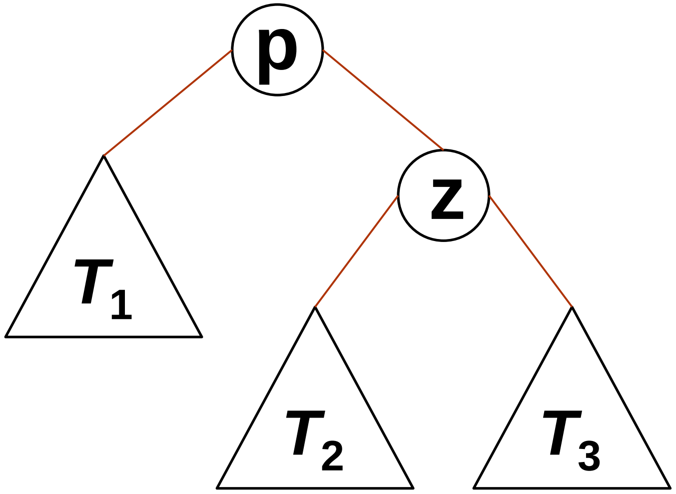
- Caso 2: h(*p.esq*)<h(*p.dir*)

- Desta vez, p possui o filho direito $z \neq q$.

- Sabe-se ainda que h(*z.esq*) ≠ h(*z.dir*)

- O caso acima pode ser ramificado em dois subcasos.

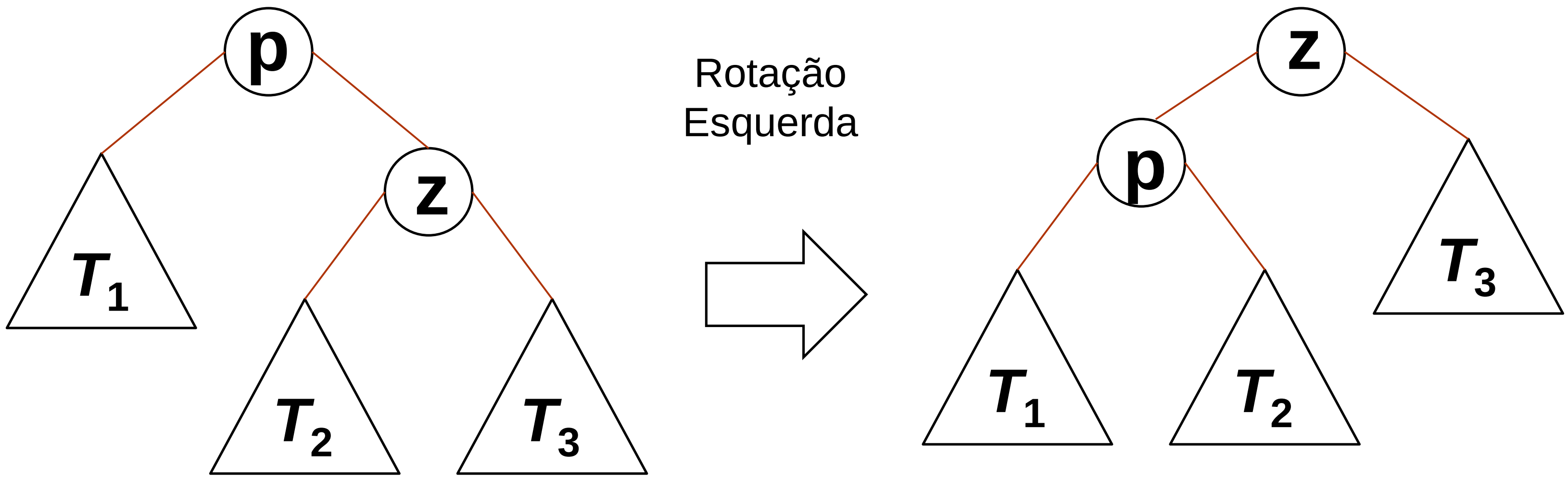
Caso 2.1



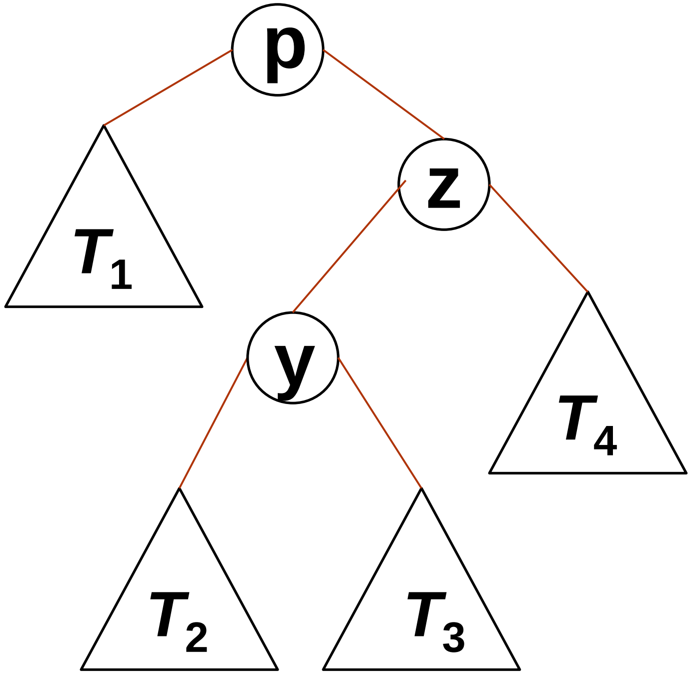
Caso 2.1

- Caso 2.1: h(*p.esq*)<h(*p.dir*) e h(*z.esq*)<h(*z.dir*)
- $q \in T_3$
- $h(T_3) - h(T_2) = 1$ e $h(T_2) = h(T_1)$
- A aplicação da **rotação esquerda** à raiz *p* transformará a subárvore considerada em uma árvore novamente regulada.

Rotação Esquerda



Caso 2.2



Caso 2.2

- Caso 2.2: h(*p.esq*)<h(*p.dir*) e h(*z.esq*)>h(*z.dir*)

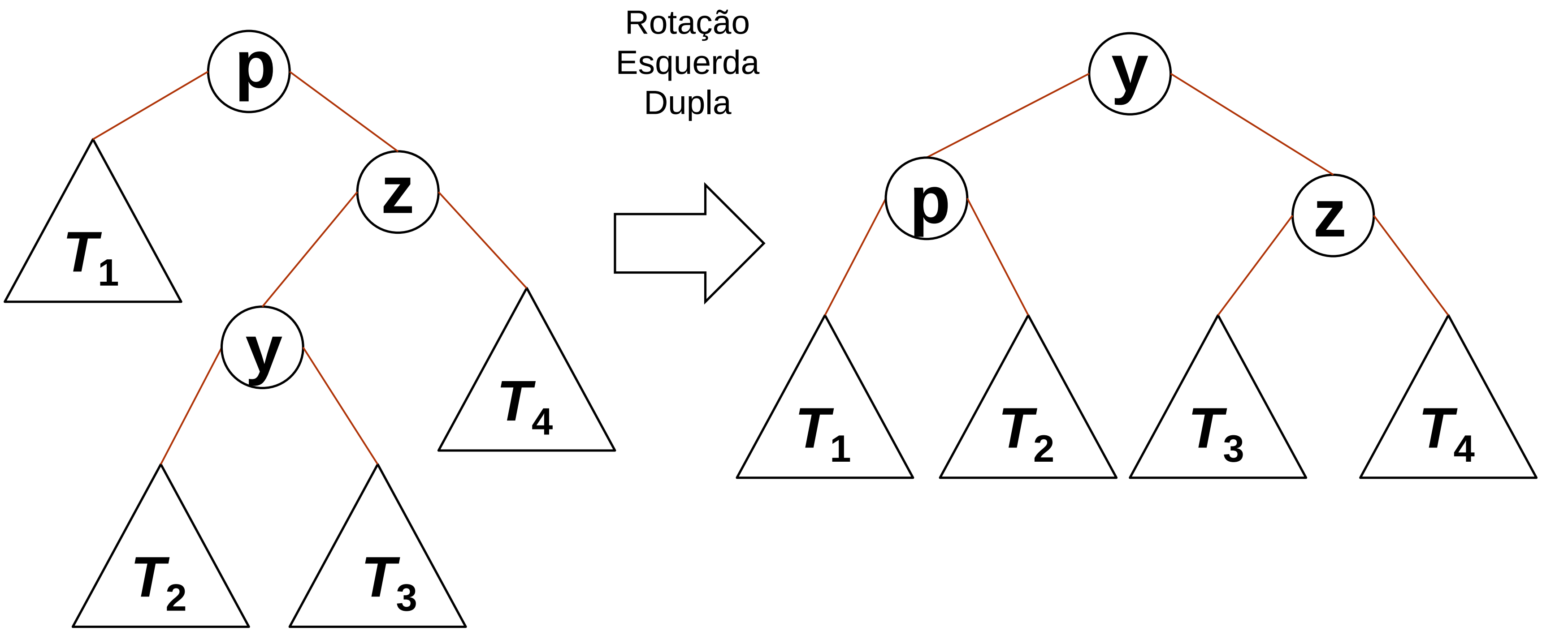
- *z* possui o filho esquerdo *y*.

- Temos que:

$$|h(T_2) - h(T_3)| \leq 1 \text{ e } \max\{h(T_2), h(T_3)\} = h(T_1) = h(T_4)$$

- Aplica-se a **rotação dupla esquerda** à raiz *p*, restabelecendo sua regulagem.

Rotação Esquerda Dupla



Inserção em Árvores AVL

```
//T -> AVL
//X -> nó a ser inserido na AVL
procedimento inserirArvoreAVL(X, T)
    inserirArvoreBin(X,T) //vide Aula de Árvores Binárias
    eAVL = verificarBalanceamentoNo(X.pai)
  se eAVL
      retorne T
  senão
      ...
      //balancear T de acordo com um dos casos estudados
```

Cálculo do Fator de Balanceamento

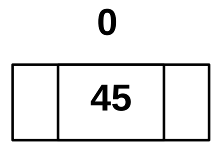
```
//pt -> NoArvoreAVL a ter seu balanceamento checado
procedimento verificarBalanceamentoNo(pt)
  se pt == NIL
      retorne falso
  senão
      calcularFatorBalanceamento(pt)
      se (pt.fatorBalanceamento < -1) ou (pt.fatorBalanceamento > 1)
          retorne verdadeiro //pt está desbalanceado
      senão //checa todo o ramo na hierarquia do nó avaliado
          pt = pt.pai //pt armazenará o último nó analisado
          retorne verificarBalanceamentoNo(pt);
```

Cálculo do Fator de Balanceamento

```
procedimento calcularFatorBalanceamento(pt)
   se pt.direita != NIL
       pt.alturaDireita = pt.direita.altura
   senão
       pt.alturaDireita = 0
   se pt.esquerda != NIL
       pt.alturaEsquerda = 1
   senão
      pt.alturaEsquerda = 0
   pt.fatorBalanceamento = pt.alturaDireita - pt.alturaEsquerda
   //max retorna o maior valor entre dois valores parâmetros
   pt.altura = max(pt.alturaDireita, pt.alturaEsquerda) + 1
```

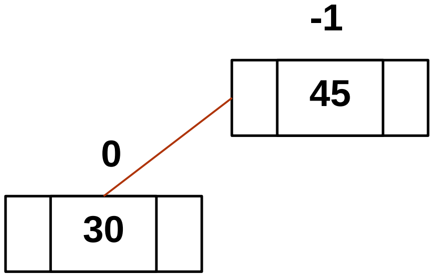
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



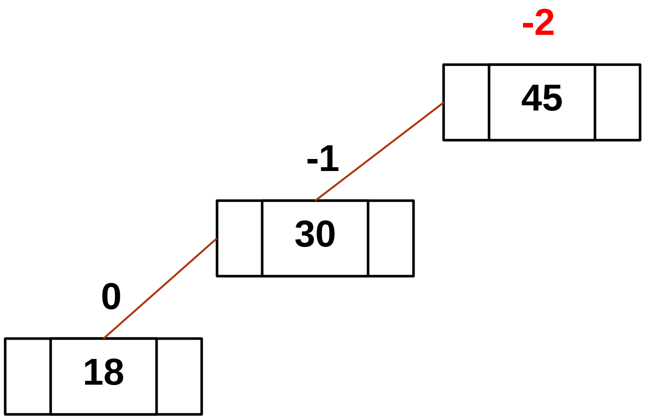
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



Exemplo

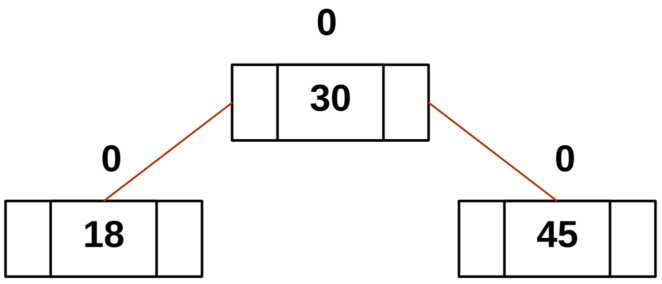
- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



Caso 1.1:
Rotação
Direita

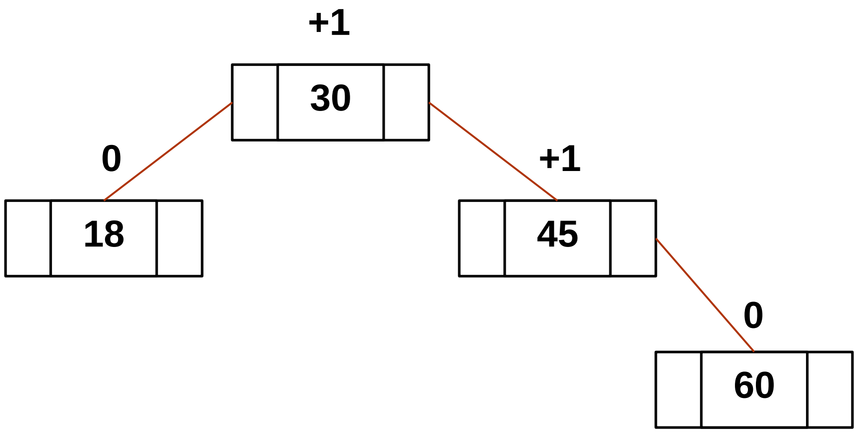
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



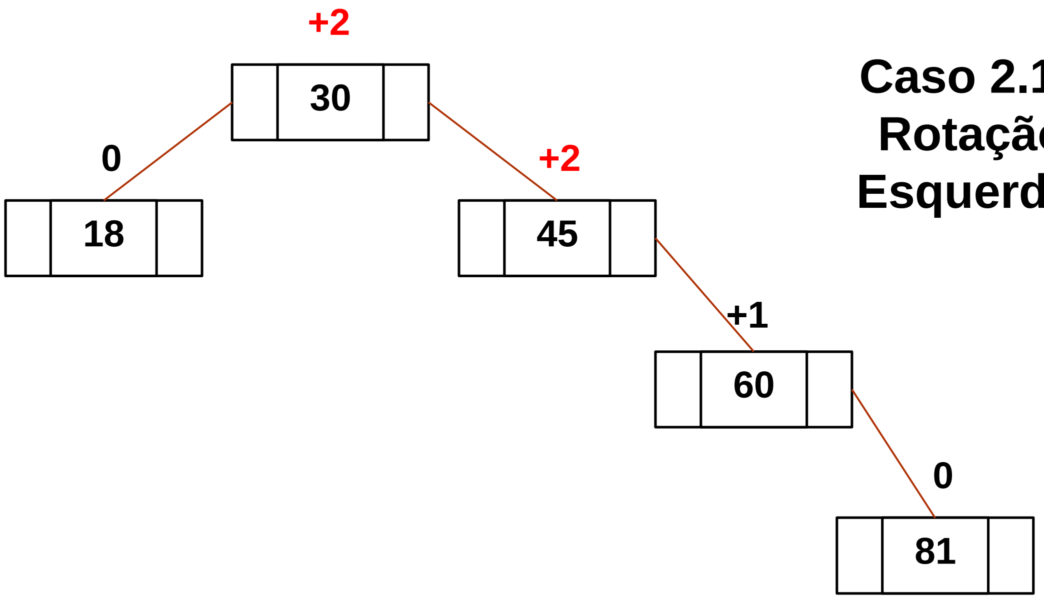
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



Exemplo

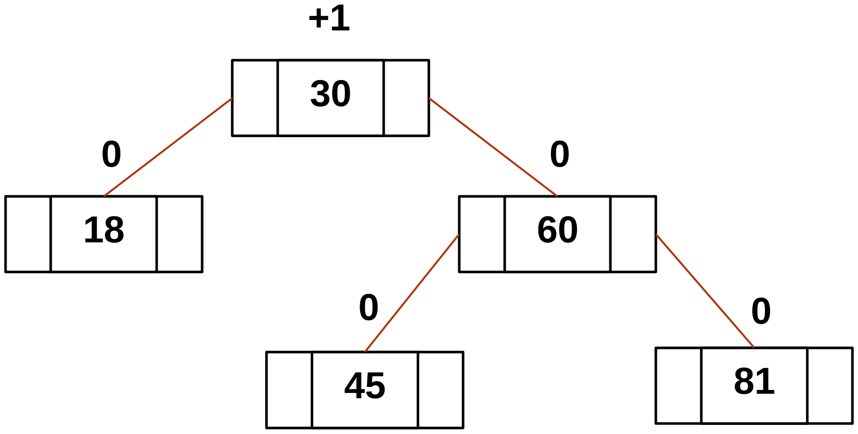
- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



Caso 2.1: Rotação Esquerda

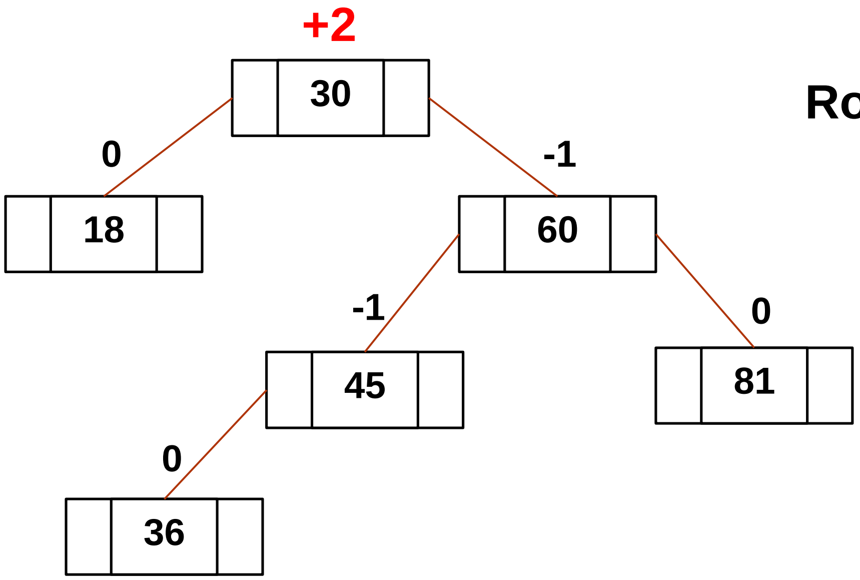
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



Exemplo

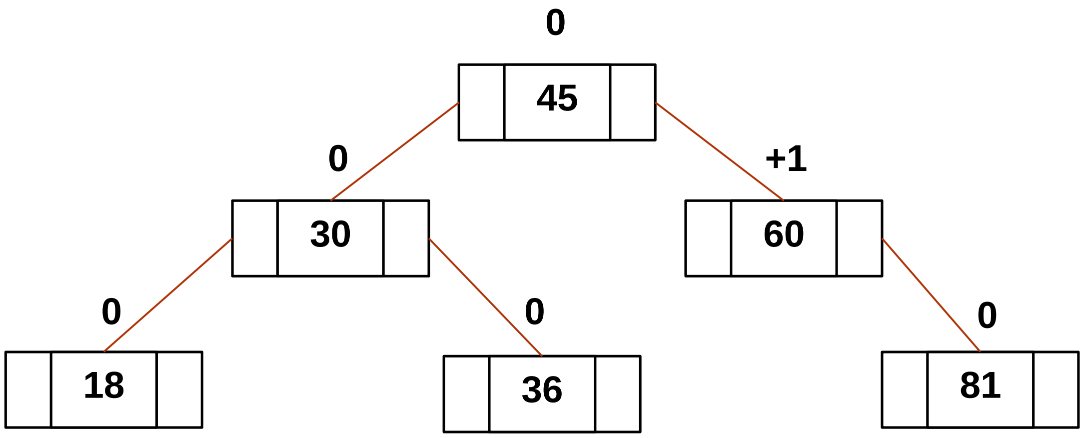
- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



Caso 2.2: Rotação Dupla Esquerda

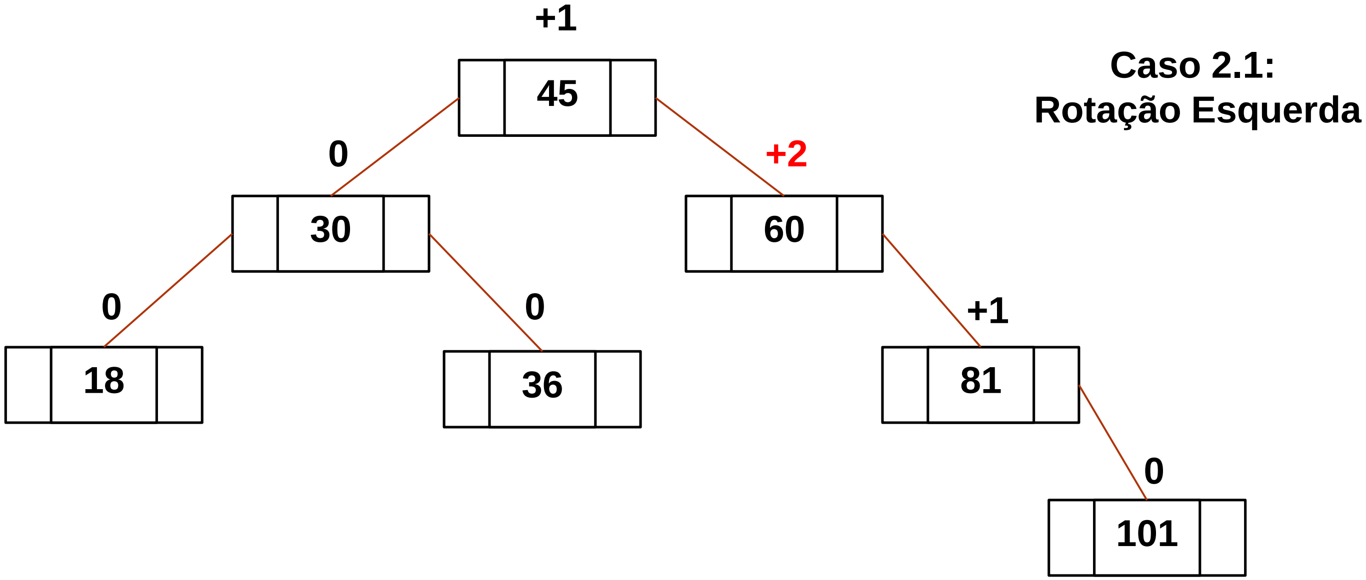
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



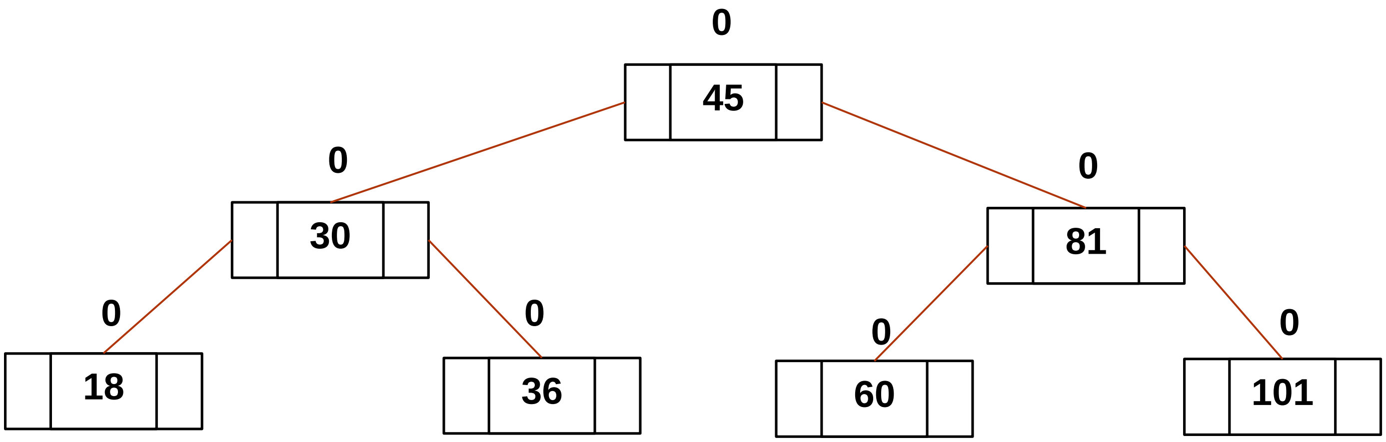
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



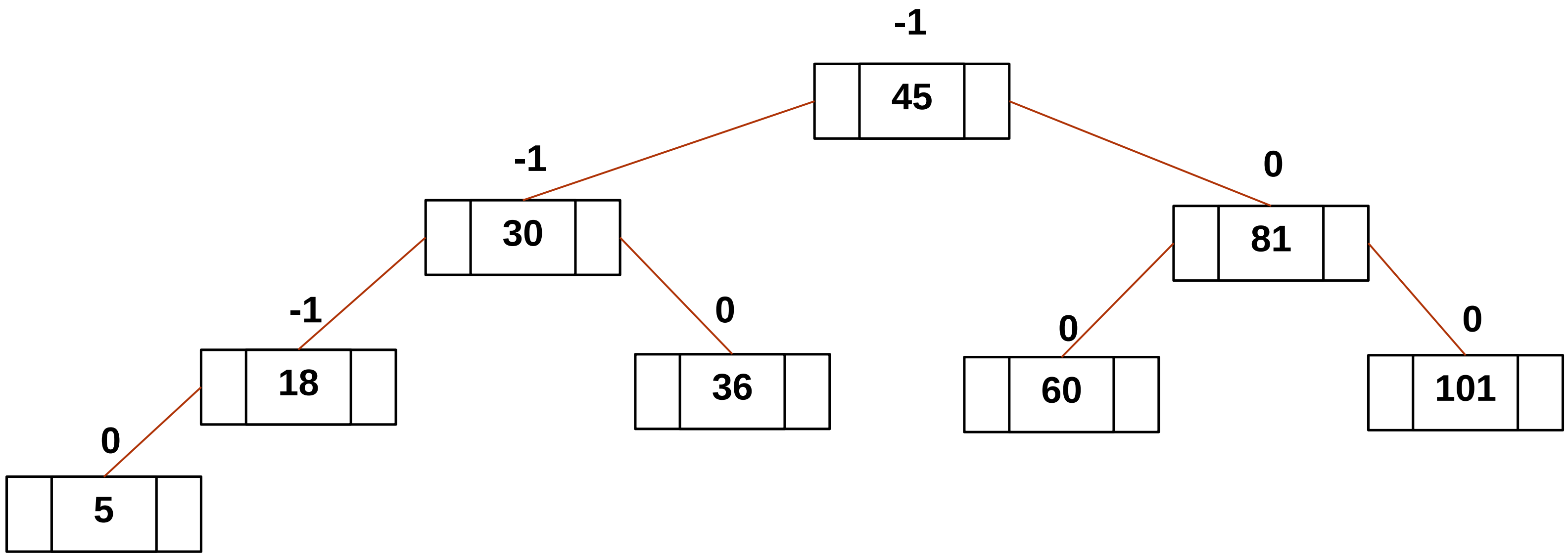
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



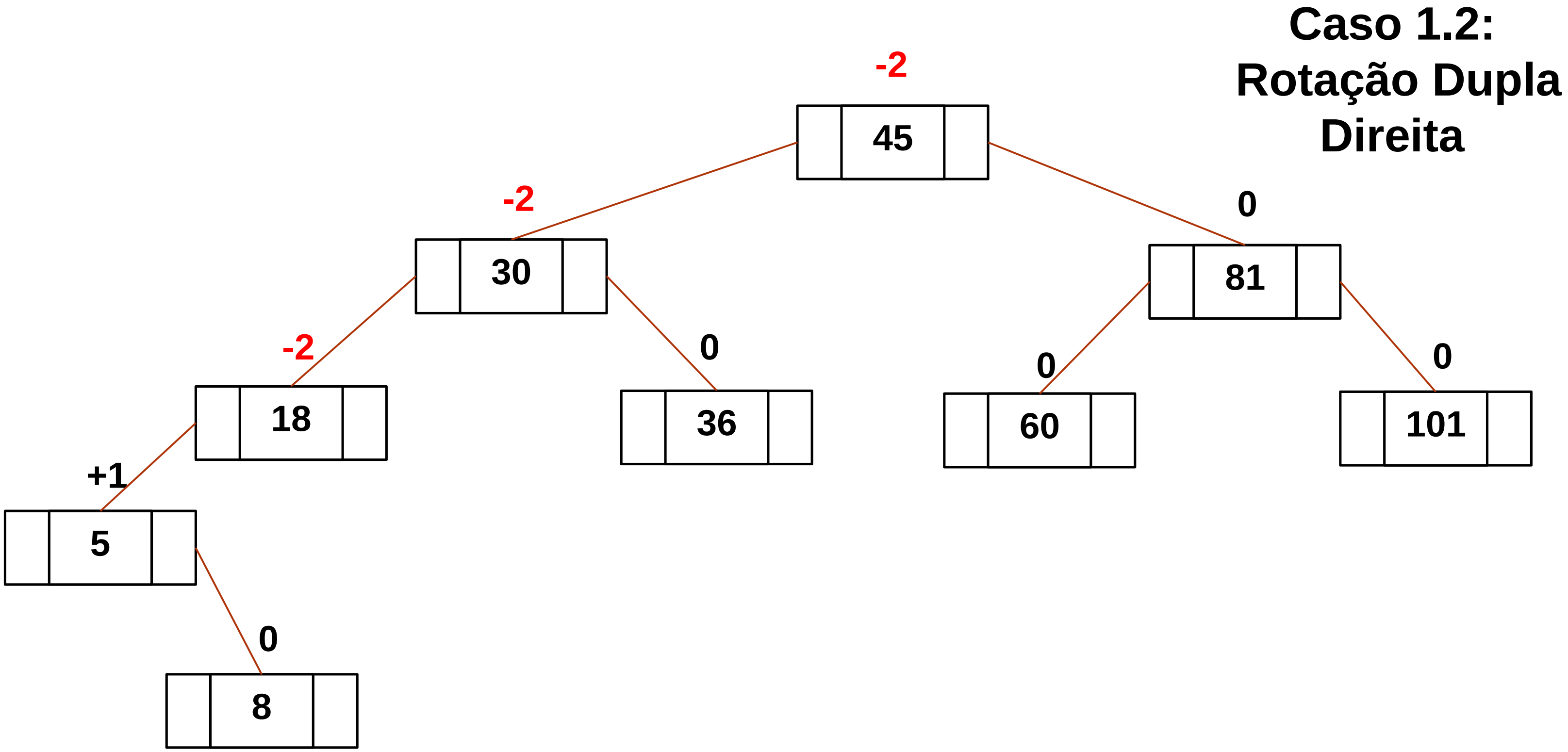
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



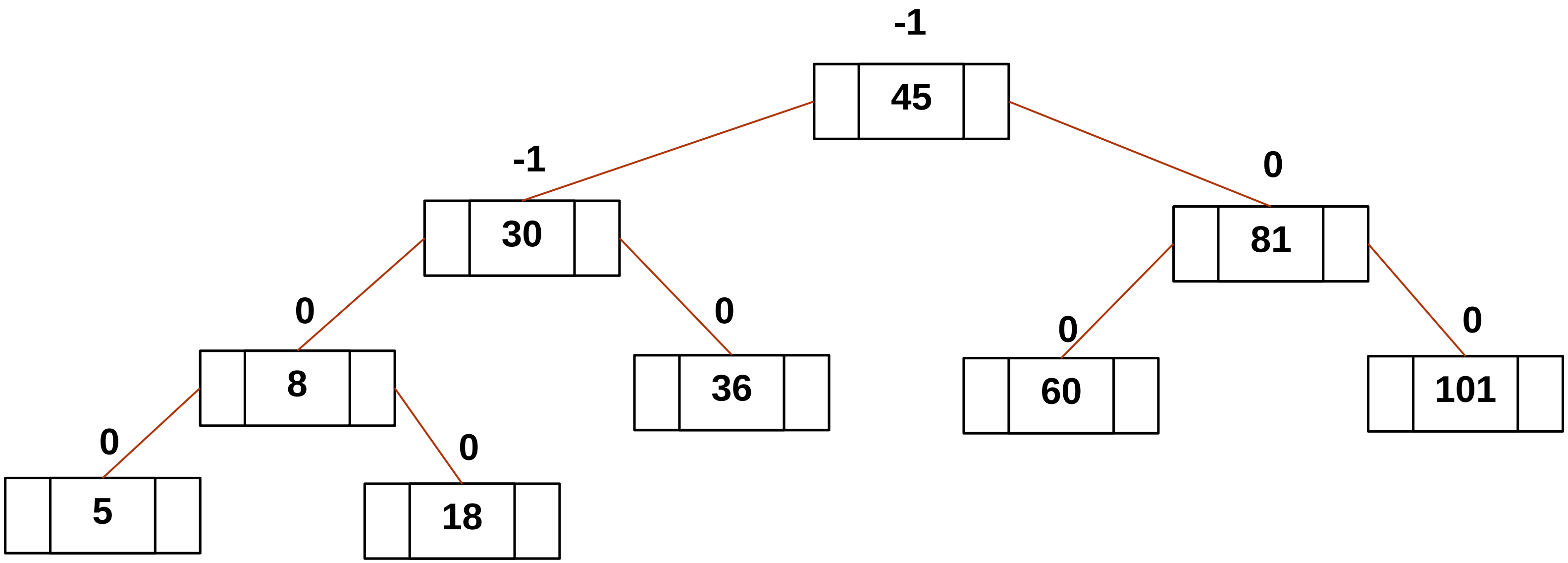
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



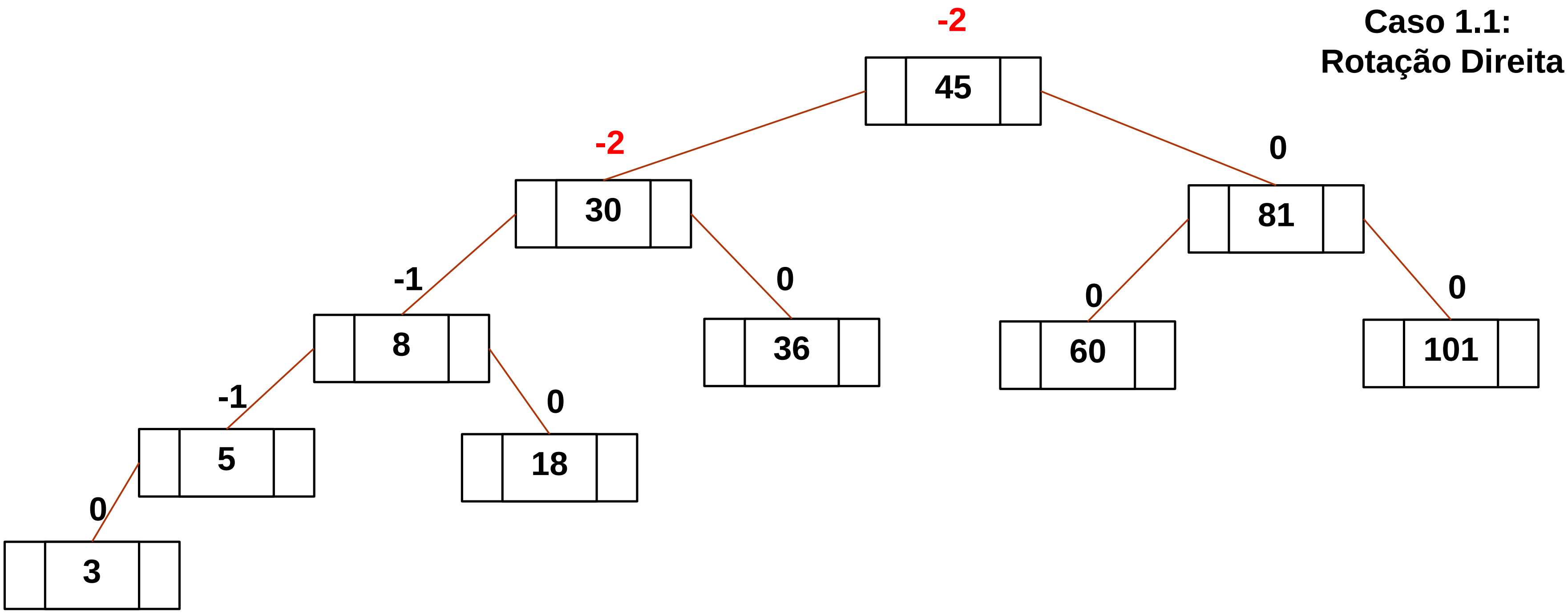
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



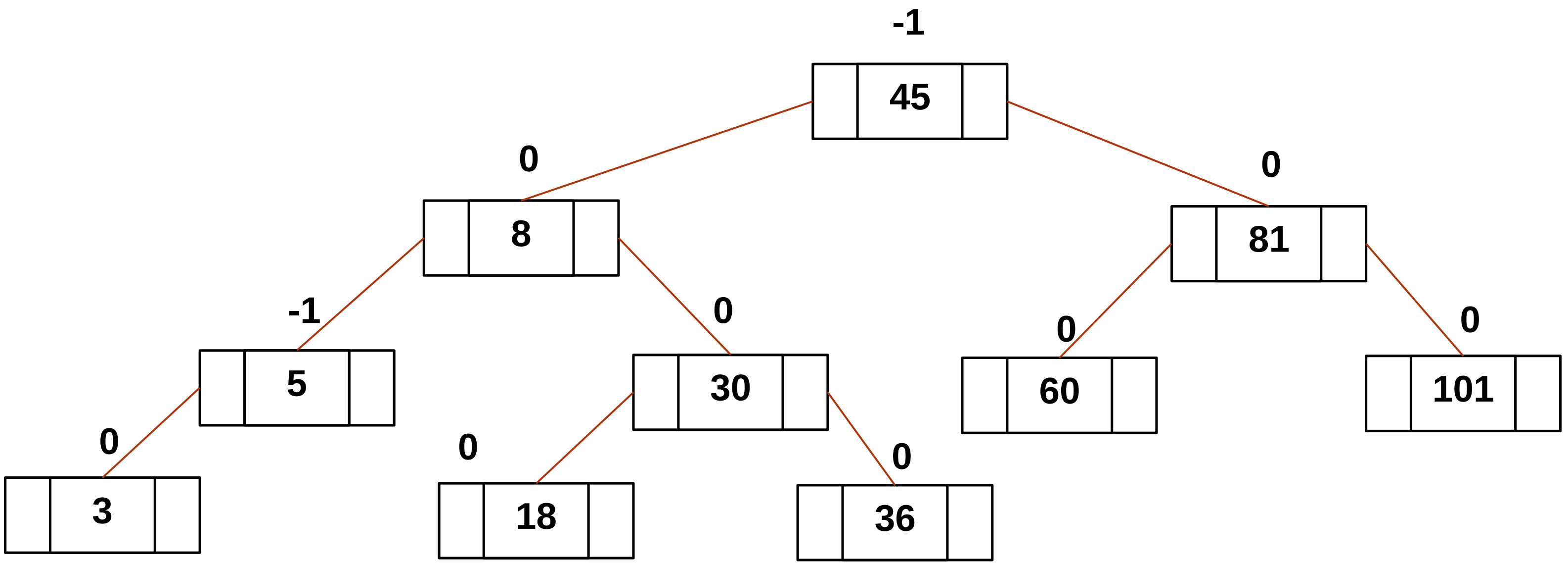
Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



Exemplo

- Inserir: 45, 30, 18, 60, 81, 36, 101, 5, 8, 3



Remoção em Árvores AVL

- A remoção em árvores AVL também pode ser executada em O(log *n*) passos.

- Para verificar se a árvore tornou-se desbalanceada, basta checar os nós no caminho até alguma folha.

- Contudo, podemos precisar de até O(log *n*) rotações para restabelecer o balanceamento da árvore.

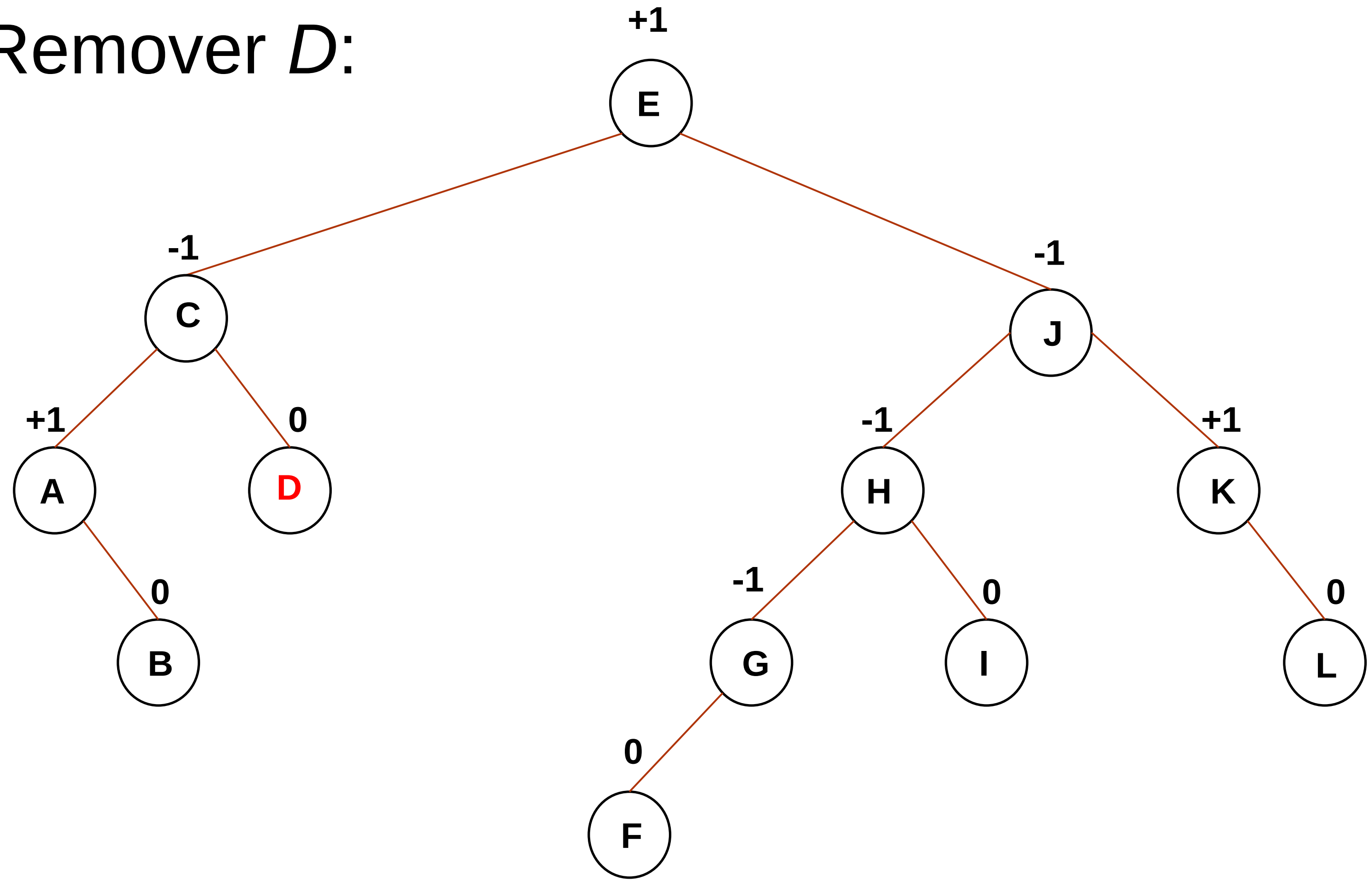
Remoção em Árvores AVL

- **OBS.:** **Não estudaremos a remoção em AVLs em detalhes, em decorrência do grande número de casos a serem avaliados.**
  - Estudaremos aprofundadamente apenas a **correção da AVL na inserção**, como já exemplificado (a busca já foi vista na Aula sobre Árvores Binárias).

```
//T -> AVL
//x -> chave do nó a ser removido da AVL
procedimento removerArvoreAVL(X, T)
    removerArvoreBin(X,T) //vide Aula de Árvores Binárias
    eAVL = verificarBalanceamentoAVL(T)
  se eAVL
      retorne T
  senão
      ...
  //balancear T de acordo com um dos casos estudados
  //enquanto houver necessidade
```

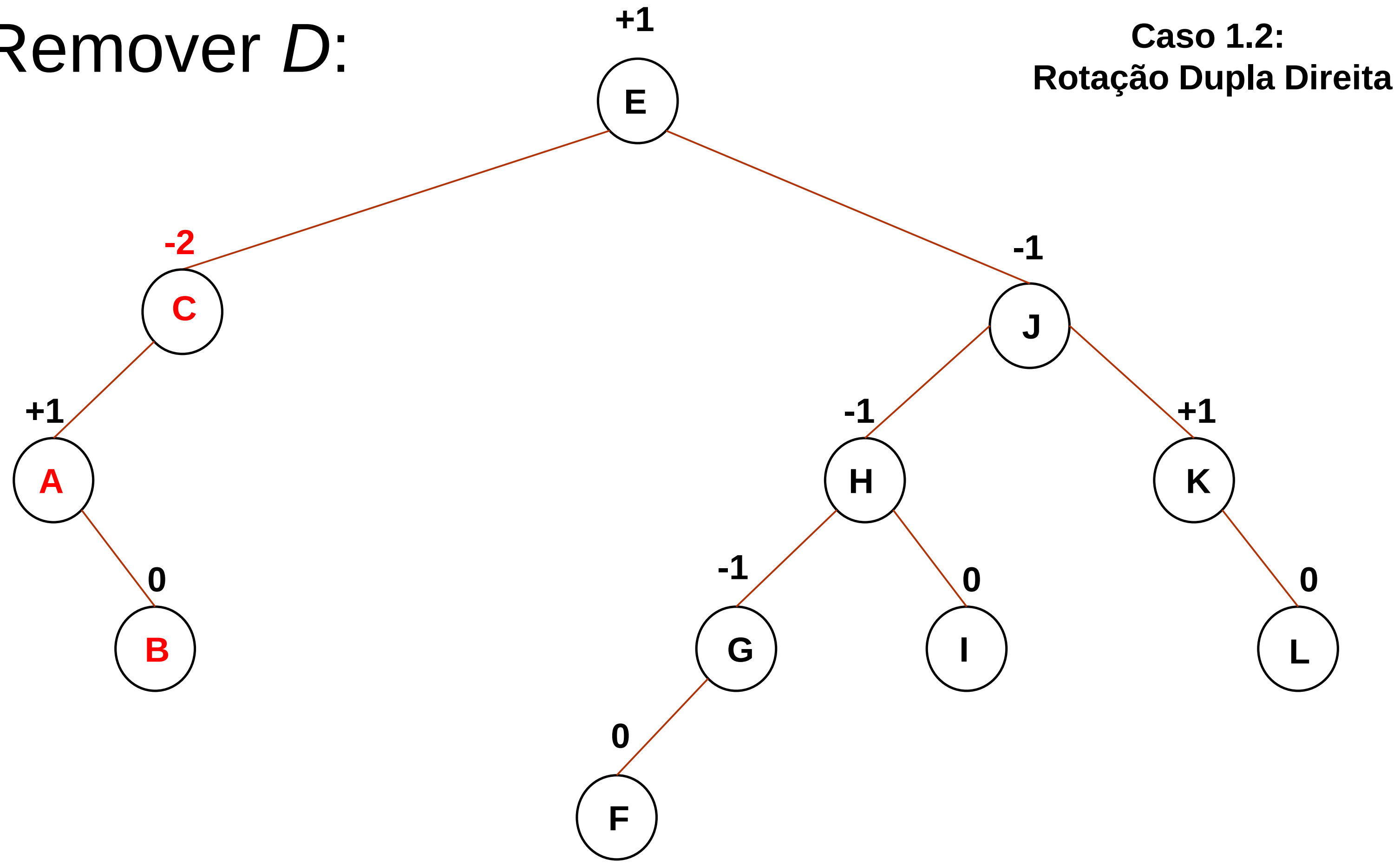
Exemplo

- Remover *D*:



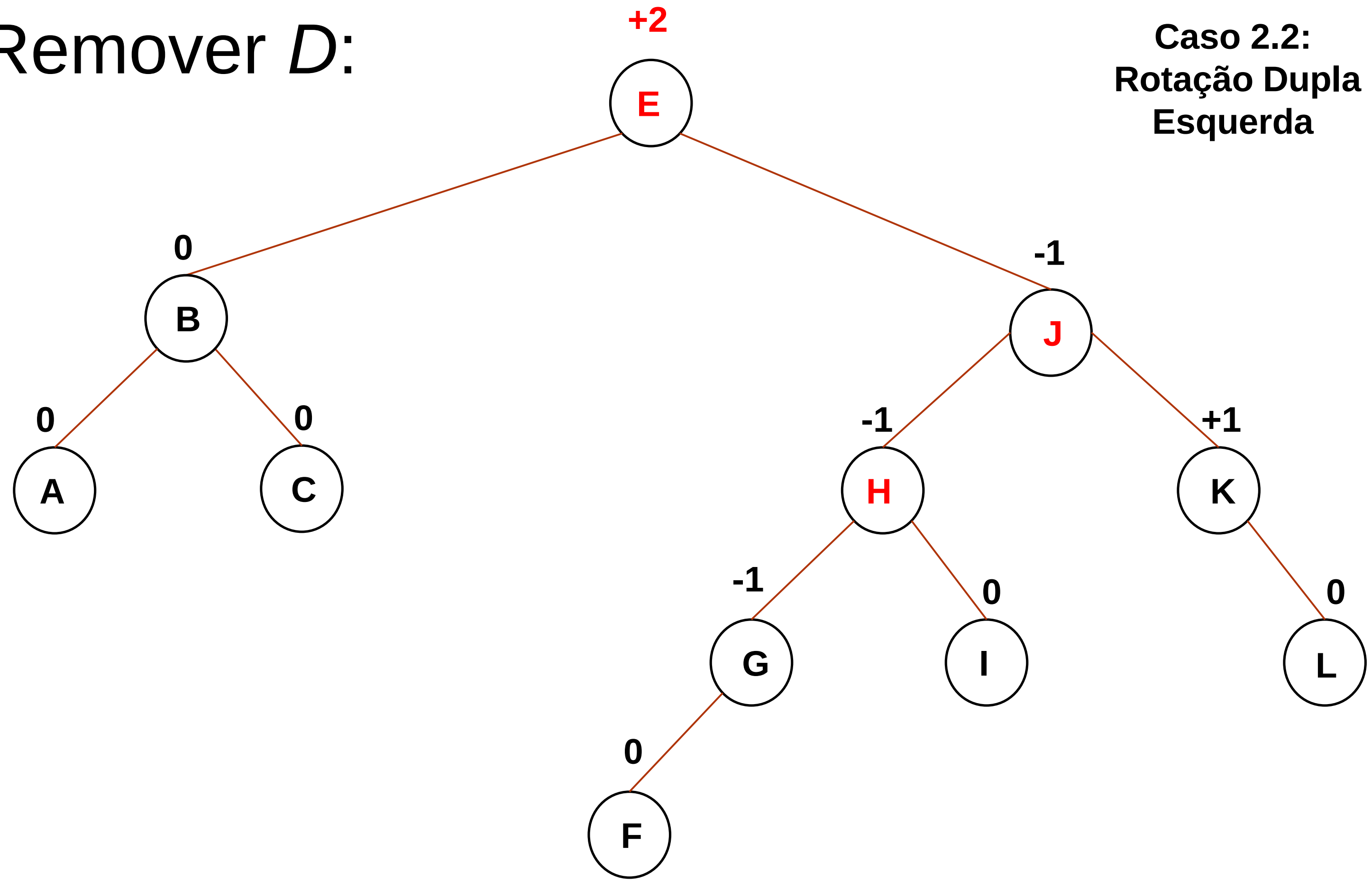
Exemplo

- Remover *D*:



Exemplo

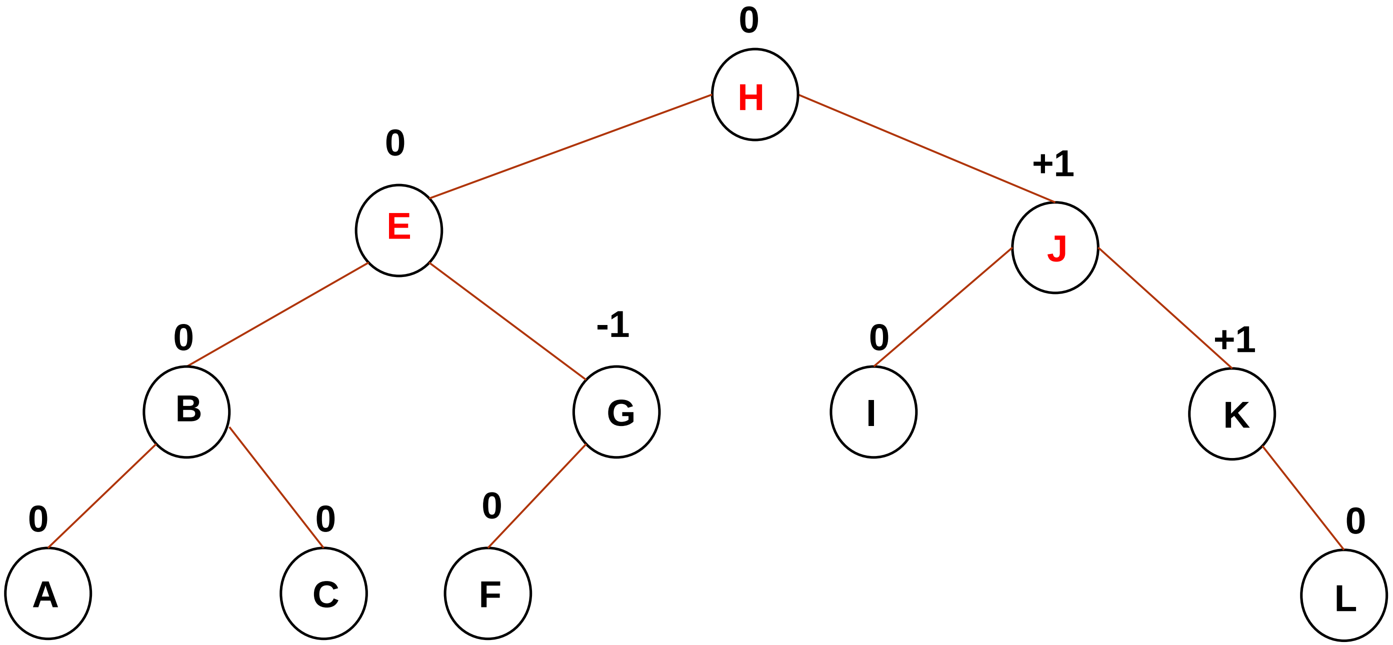
- Remover *D*:



Caso 2.2:
Rotação Dupla Esquerda

Exemplo

- Remover *D*:



Referências

- SZWARCFITER, J.; MARKENZON, L. Estruturas de Dados e seus Algoritmos, 3ª ed. Rio de Janeiro: LTC, 2010.

- CORMEN, H. T.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introduction to Algorithms, 3rd ed., *Boston: MIT Press*, 2009.
  - **OBS.:** O Cormen apresenta os detalhes de outros tipos de árvores balanceadas, porém fala brevemente apenas sobre as AVLs.

- FEOFILOFF, Paulo. Algoritmos em Linguagem C. Editora Campus/Elsevier, 2009.

Árvores AVL

Algoritmos e Estruturas de Dados

Prof. Luciano Demétrio Santos Pacífico
{luciano.pacifico@ufrpe.br}

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO