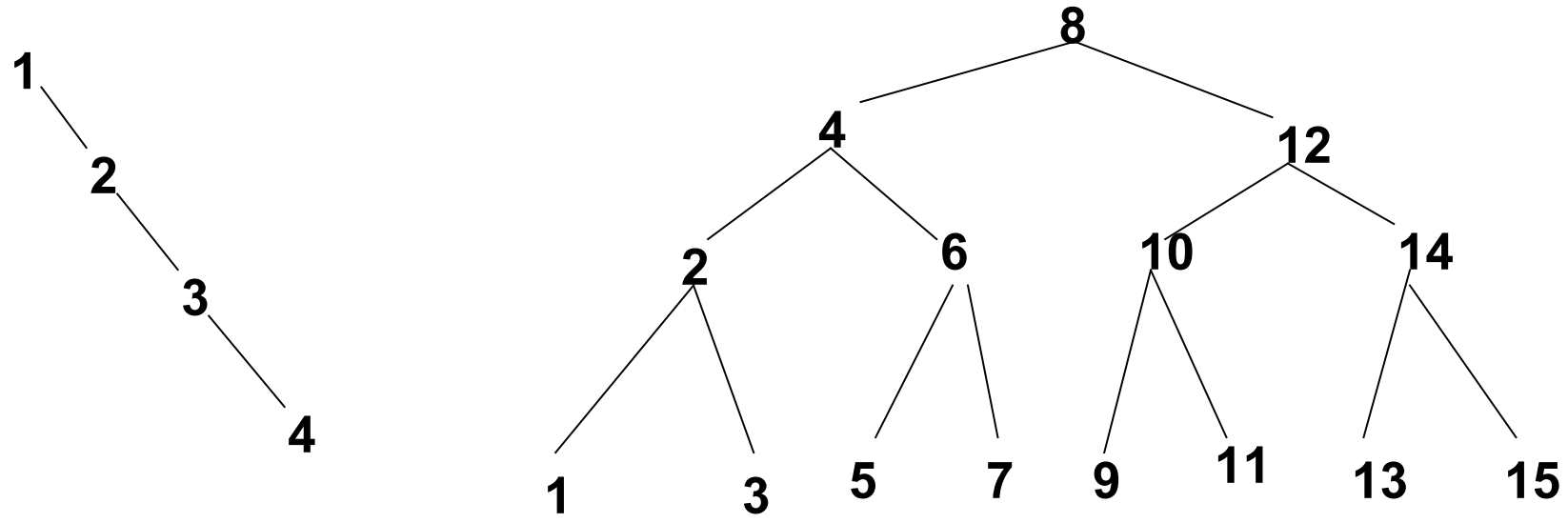
Árvores Balanceadas

Tempo de busca numa Árvore Binária de Busca depende da profundidade da árvore



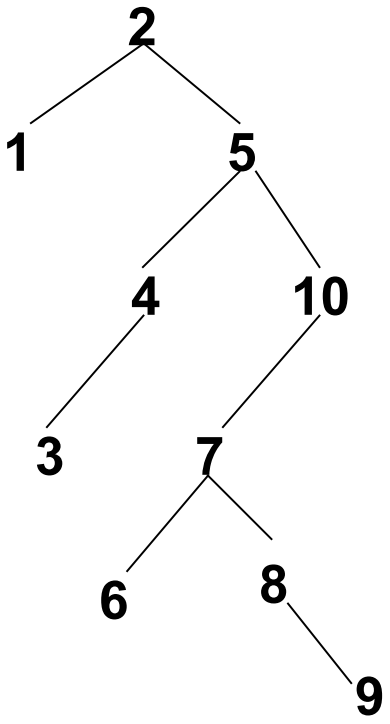
Com 4 nós precisa de 4 comparações

Com 15 nós precisa de 4 comparações

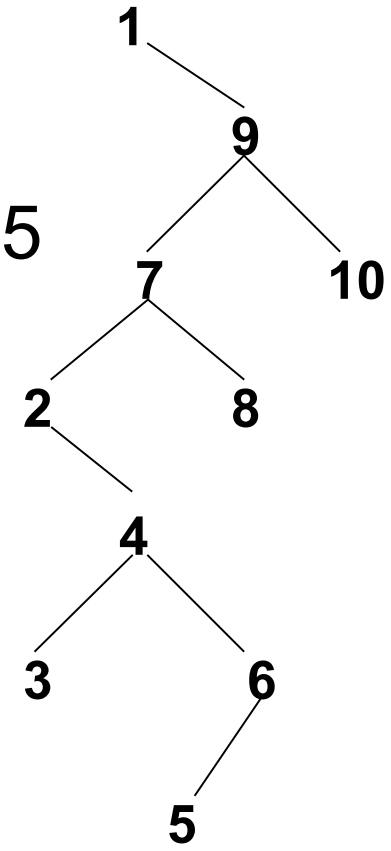
Árvores Balanceadas

A geração de uma Árvore Binária de Busca não garante que o resultado seja eficiente para a busca

2 5 10 7 1 6 8 4 3 9

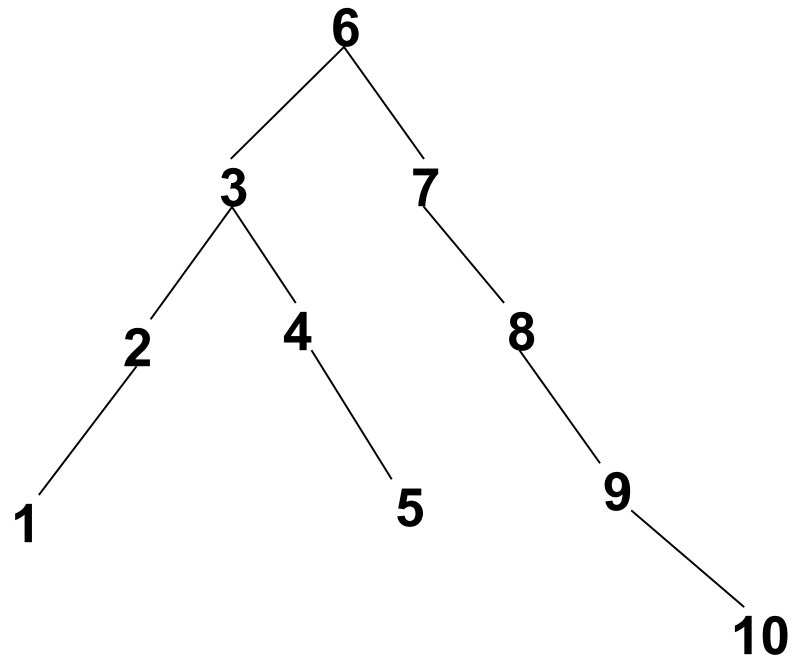


1 9 7 2 8 10 4 3 6 5

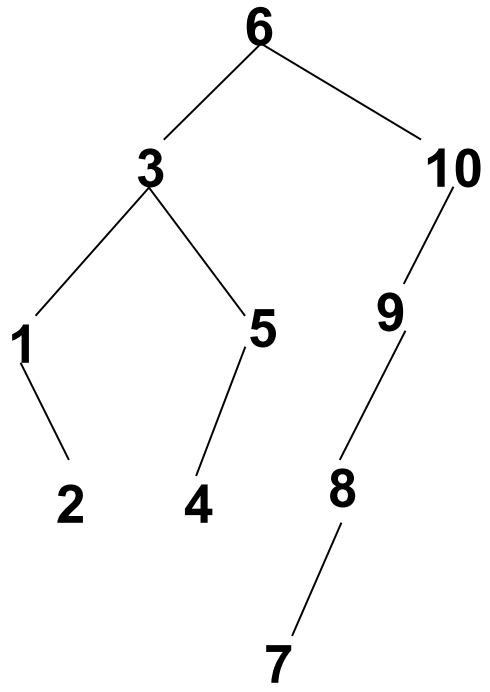


Árvores Balanceadas

6 7 8 3 4 9 2 5 10 1



6 10 3 5 9 1 4 2 8 7



Árvores Balanceadas

O ideal é sempre ter uma Árvore Binária completa.
Mas nem sempre isso é possível.

Árvore não é balanceada

A melhor estrutura, para uma árvore parcialmente preenchida, é aquela onde TODOS OS NÍVEIS estão preenchidos exceto o último nível que pode não ter algumas folhas

Árvores Balanceadas

Conhecimento a priori dos dados é uma boa ajuda.
Não é sempre o caso.

Mas nas operações de inserção e remoção, a eficiência na busca deve ser mantida. COMO?

Duas categorias de árvores:

- Árvores *Height-Balanced* (1962)
  - ✓ AVL (G. Adel'son-Vel'skii e E. Landis)

- Árvores *Multiway*
  - ✓ B-trees e sua generalização a-b trees

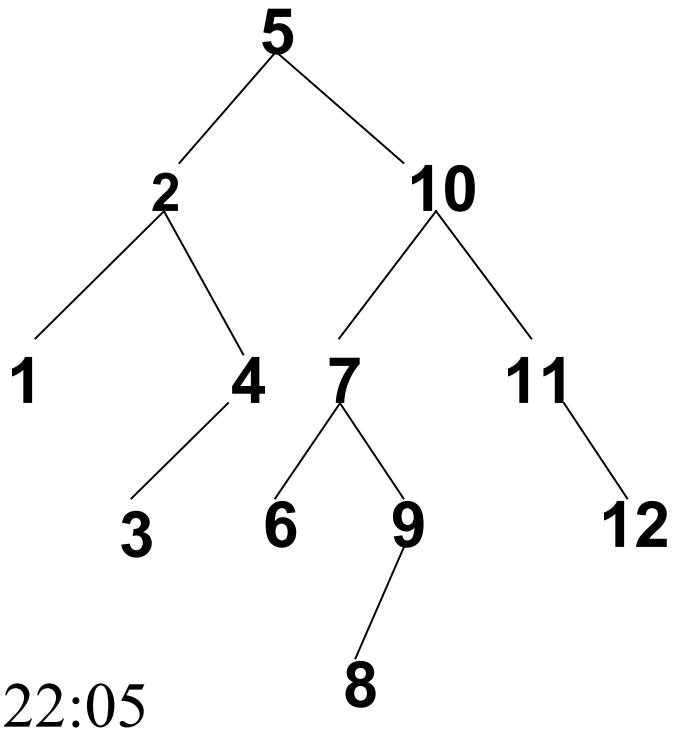
Árvores Balanceadas

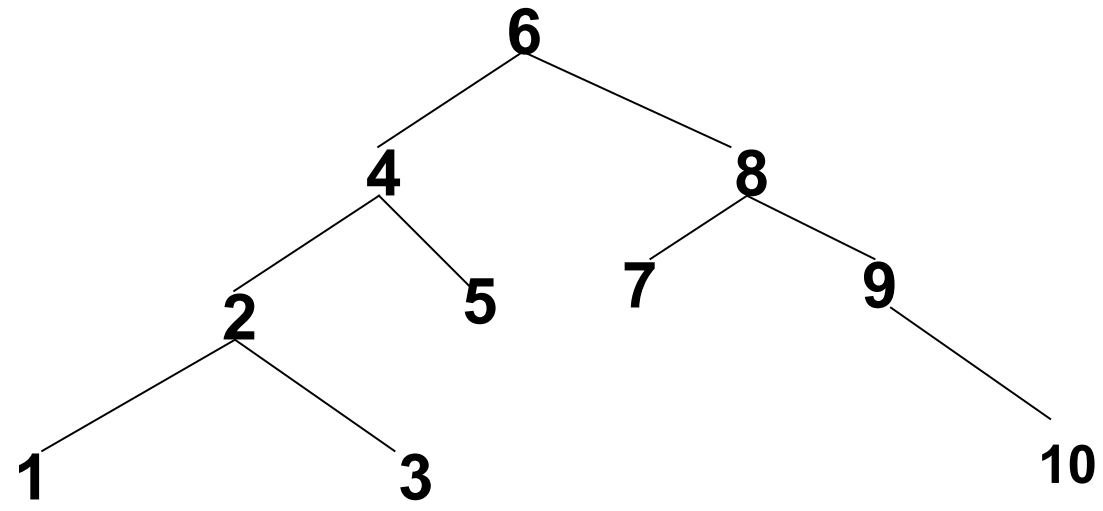
Altura de um nó

Comprimento do caminho mais longo do nó até folha

Height-Balanced trees

Para cada nó, as alturas das suas sub-árvores diferem no máximo em 1. Também são chamadas de árvores AVL e esta diferença é o fator de balanceamento.

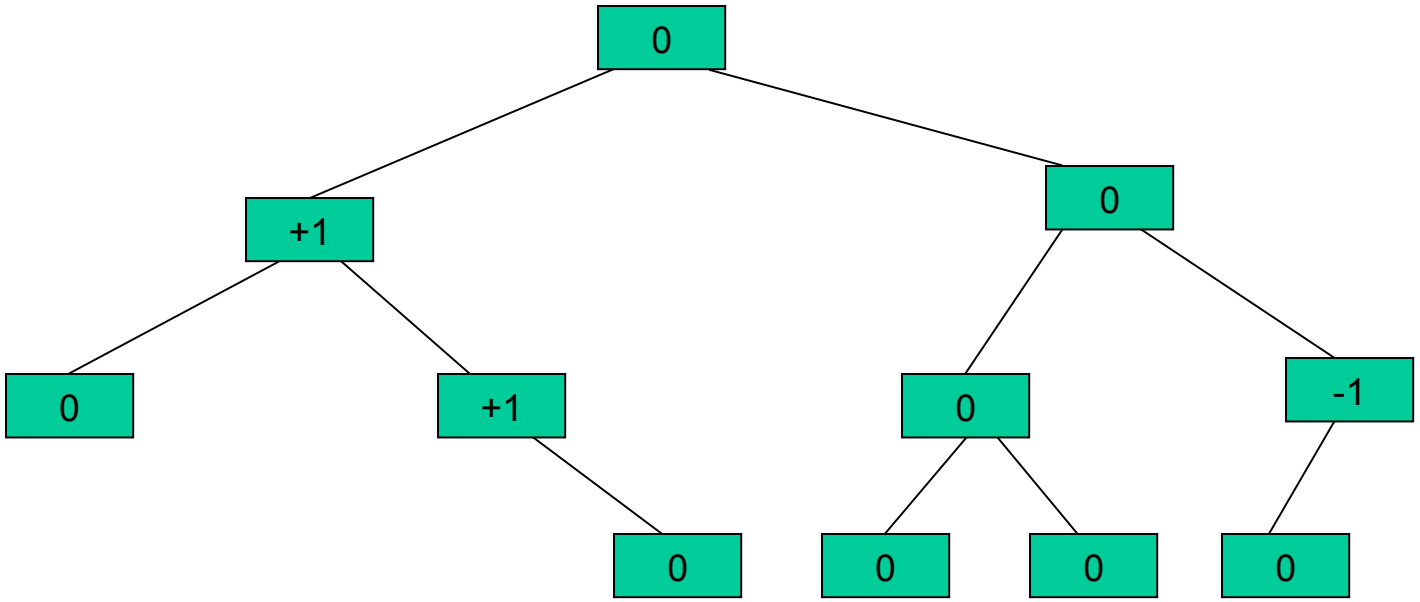




Height-Balanced trees

Convenções usadas para o Fator de Balanceamento

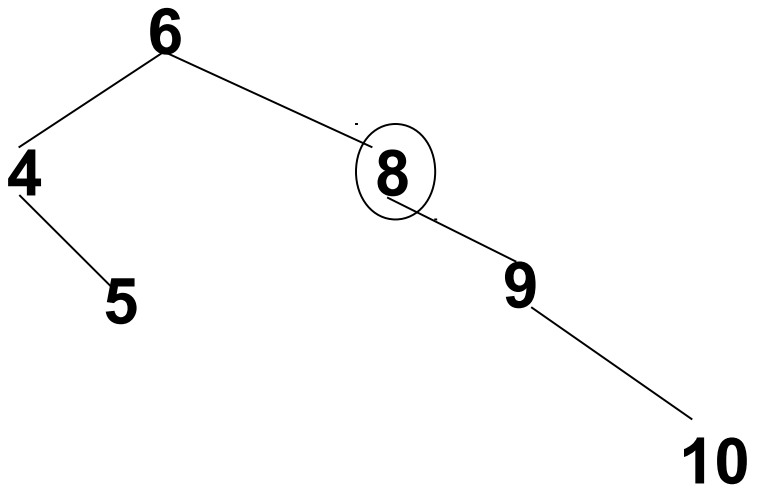
[/] ou [-1], sub-árvore à esquerda tem altura maior
[\] ou [+1], sub-árvore à direita tem altura maior
[-] ou [0], sub-árvores possuem a mesma altura



Height-Balanced trees

Exemplo de uma árvore que não está balanceada pela altura.

O nó marcado viola a condição de balanceamento



Height-Balanced trees

Operações aplicadas para reestruturar AVLs depois de serem “bagunçadas” devido à inserção e remoção

- Rotação (*rotation*)
- Rotação Dupla (*double rotation*)

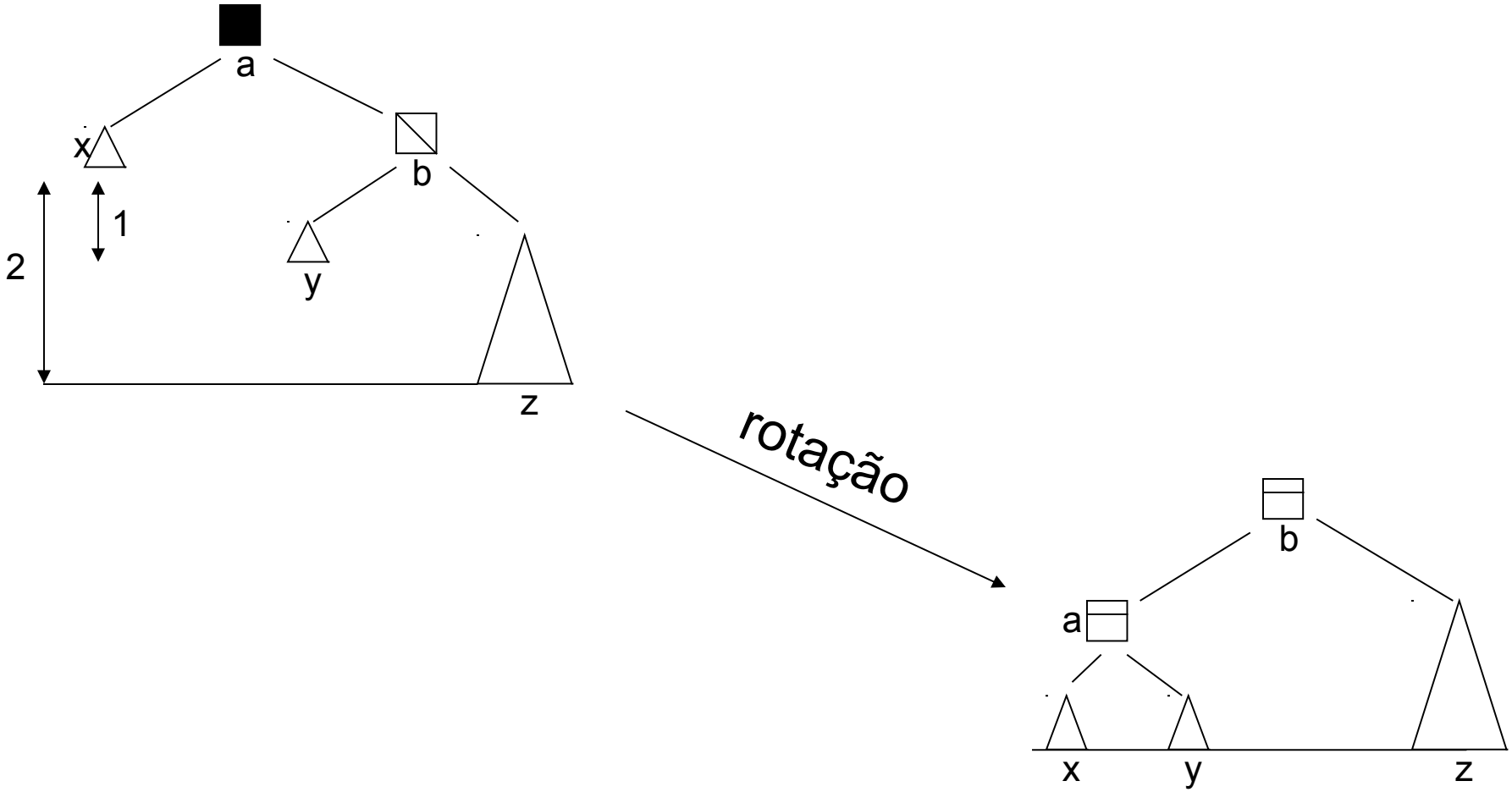
Árvore dividida em sub-árvores e reconstrui numa maneira diferente

Height-Balanced trees

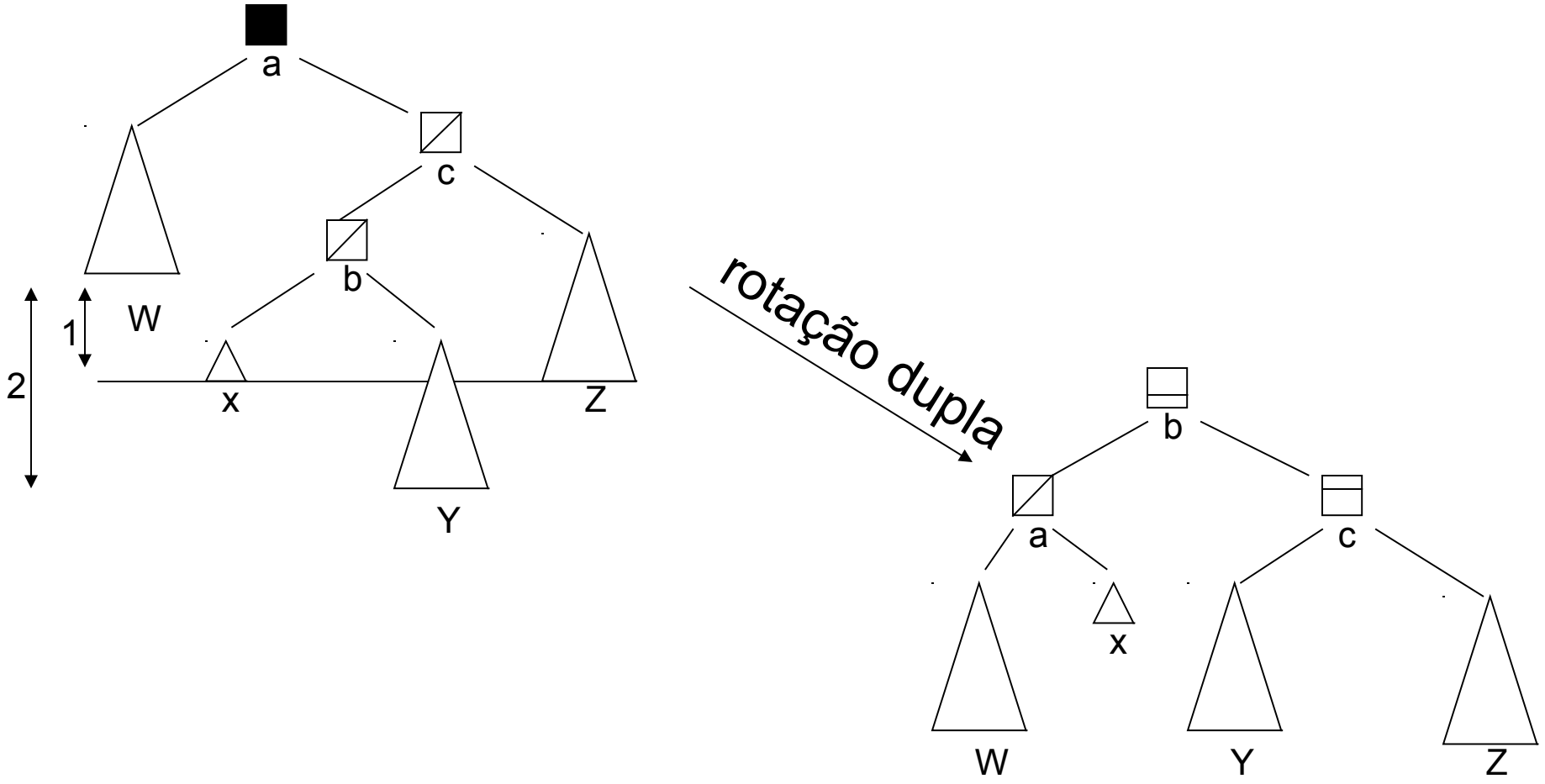
Três propriedades de balanceamento:

- [-]horizontal
- [/]inclinação à esquerda (*left-leaning*)
- [\]inclinação à direita (*right-leaning*)

Height-Balanced trees



Height-Balanced trees



Height-Balanced trees

Processo de Inserção

* novo elemento inserido na árvore
* condição de balanceamento do novo nó -
* a partir do novo nó
  - caminhe até a raiz passando a informação que a altura da sub-árvore do nó inserido aumentou em 1
* em cada nó no caminho uma operação baseada em *algumas regras* é realizada

Height-Balanced trees

As regras dependem de:

→condição de balanceamento do nó (encontrado no caminho) antes da inclusão do novo elemento

→direção do acesso a esse nó (esquerda ou direita)

Height-Balanced trees

Regra I_1

SE (nó atual tem a condição de balanceamento
horizontal[-])
{
mude para [\] se acessou pelo lado direito

mude para [/] se acessou pelo lado esquerdo

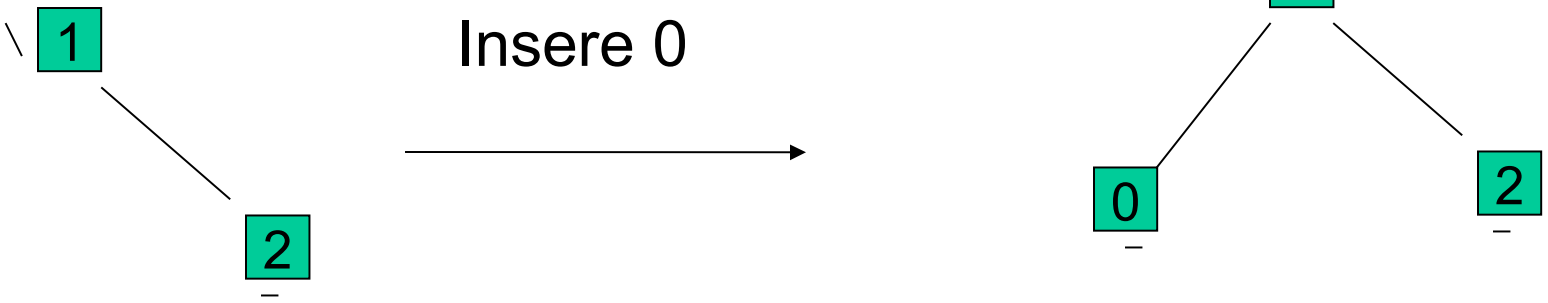
SE (nó é raiz) FIM
SENÃO continue subir
}

Height-Balanced trees

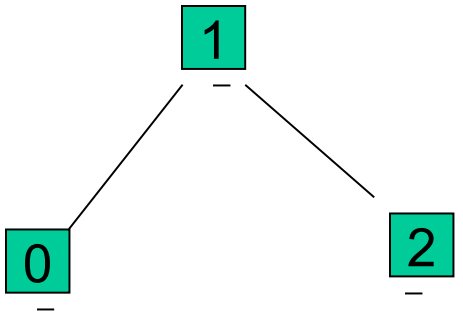
Regra I_2

```
SE ( nó atual tem a condição de balanceamento
       [/] ou [\] )
{
    SE ( acesso via sub-árvore que era
                mais baixa antes da inserção do nó)
    {
        mude para [-]
        FIM
    }
}
```

Height-Balanced trees



Antes da inserção do novo nó 0
a sub-árvore à esquerda era mais baixa que o da direita (Não havia sub-árvore à esquerda)

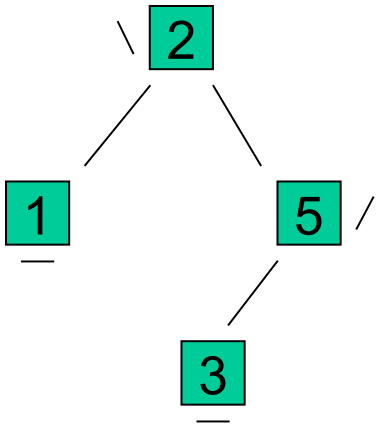


Height-Balanced trees

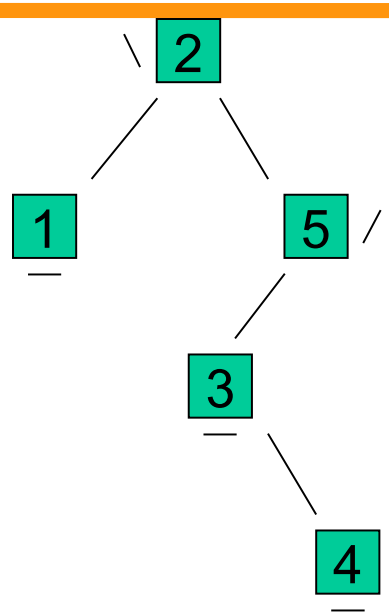
Regra I_3

```
SE ( nó atual tem a condição de balanceamento
       [/] ou [\] )
{
    SE ( acesso via sub-árvore que era
                mais alta antes da inserção do nó)
    {
        // condição de balanceamento violada
        restaura o balanceamento
    }
}
```

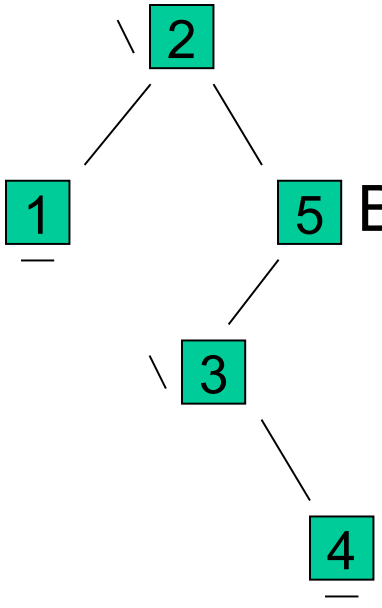
Height-Balanced trees



Insere 4



Antes da inserção do novo nó 4 a sub-árvore à esquerda era mais alta que o da direita (Não havia sub-árvore à direita)



BALANCEAR

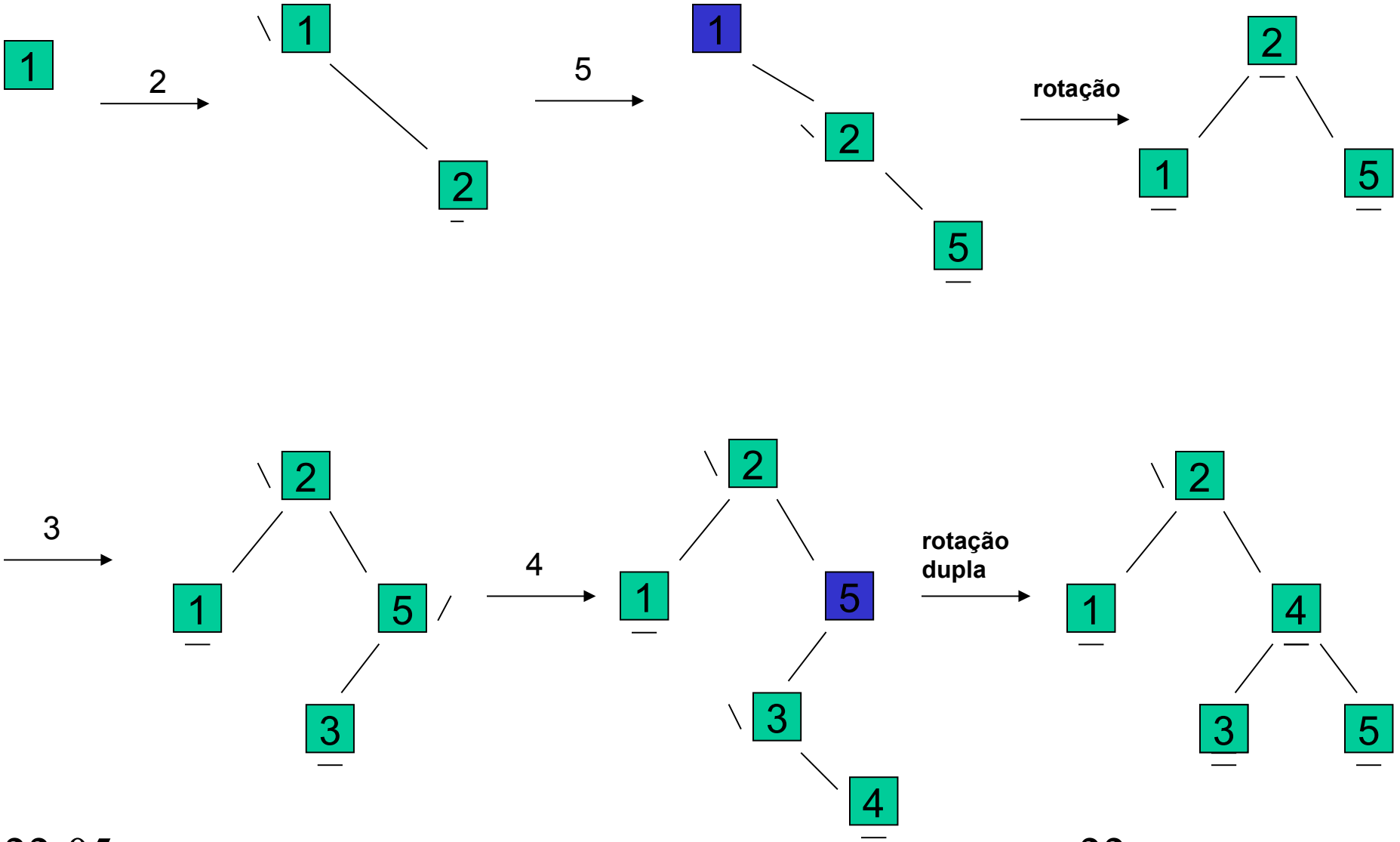
Height-Balanced trees

Como restaurar a condição de balanceamento?

SE os últimos dois passos vieram da mesma direção (os dois de esquerda ou os dois de direita) ENTÃO aplica *rotação*

SE os últimos dois passos vieram de direções opostas (um de esquerda e outro de direita OU um de direita e outro de esquerda) ENTÃO aplica *rotação dupla*

Height-Balanced trees



Height-Balanced trees

Processo de Remoção

* um elemento é removido da árvore

* a partir do nó removido
  - caminhe até a raiz passando a informação que a altura da sub-árvore do nó removido diminuiu em 1

* em cada nó no caminho uma operação baseada em *algumas regras* é realizada

Height-Balanced trees

As regras dependem de:

→condição de balanceamento do nó (encontrado no caminho) antes da remoção do elemento

→direção do acesso a esse nó (esquerda ou direita)

Height-Balanced trees

Regra D_1

SE (nó atual tem a condição de balanceamento
horizontal[-])
{

mude para [\] se acessou pelo lado direito

mude para [/] se acessou pelo lado esquerdo

}

Height-Balanced trees

Regra D_2

SE (nó atual tem a condição de balanceamento
[/] ou [\])
{
SE (acesso via sub-árvore que era
mais alta antes da remoção do nó)
{
mude para [-]
continue a caminhar para cima com a
mensagem que a sub-árvore do nó
encurtou
}
}

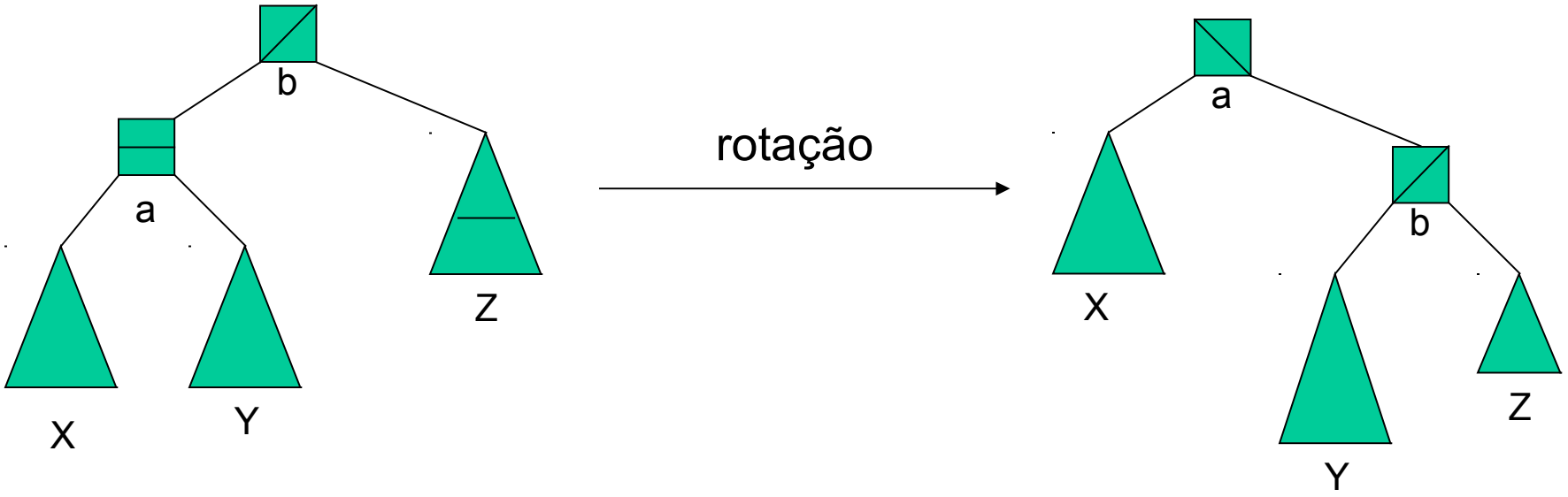
Height-Balanced trees

Regra D_3

```
SE ( nó atual tem a condição de balanceamento
     [/] ou [\] )
{
   SE ( acesso via sub-árvore que era
              mais baixa antes da remoção do nó)
   {
      // condição de balanceamento violada
      restaura o balanceamento (3 casos)
   }
}
```

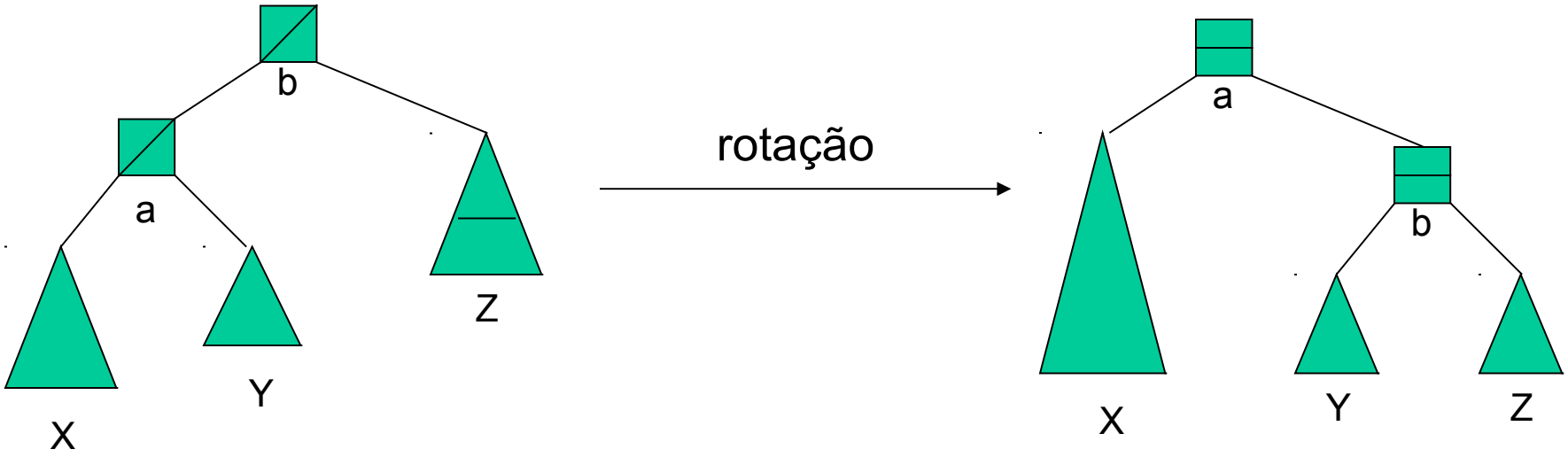
Height-Balanced trees

Caso 1



Height-Balanced trees

Caso 2



Height-Balanced trees

Caso 3