Estruturas de Dados - T.332

Capítulo 8

Árvores

8.2 Árvores de Busca

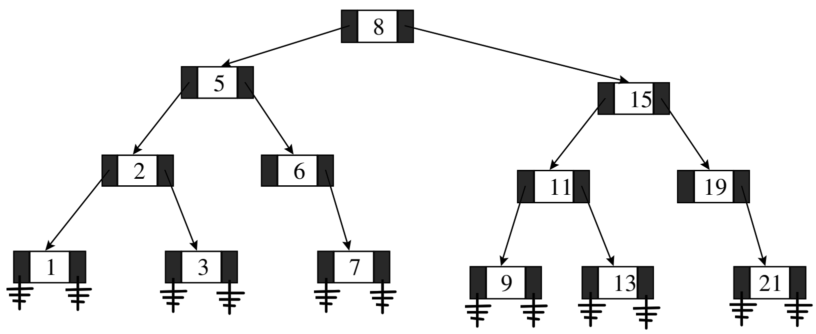
8.2.1. Árvores (binárias) de Busca

- Árvores (binárias) são muito utilizadas para se representar uma grande conjunto de dados onde se deseja encontrar um elemento de acordo com a sua chave.
- **Definição**: Árvore Binária de Busca (Niklaus Wirth): Uma que árvore se encontra organizada de tal forma que, para cada nodo t_i, todas as chaves (*info*) da subárvore à esquerda de t_i são menores que (*ou iguais a*) t_i e à direita são maiores que t_i.
- Termo em Inglês: Search Tree

Características de Árvores de Busca

- Em uma árvore de busca é possível encontrar-se qualquer chave existente descendo-se a árvore
  - sempre **à esquerda** toda vez que a chave procurada for **menor** do que a chave do nodo visitado e
  - sempre **à direita** toda vez que for **maior**.
  - A escolha da direção de busca só depende da chave que se procura e da chave que o nodo atual possui.
- A busca de um elemento em uma árvore balanceada com n elementos toma tempo médio < log(n), tendo a busca então O(log n).
  - Graças à estrutura de árvore a busca poderá ser feita com apenas log(n) comparações de elementos.

Exemplo de Árvore de Busca Binária



• Um exemplo de uma árvore de busca balanceada.

Busca em Árvores

```
tipo nodo {
        info : tipo info
        dir : *nodo
        esq : *nodo
}
```

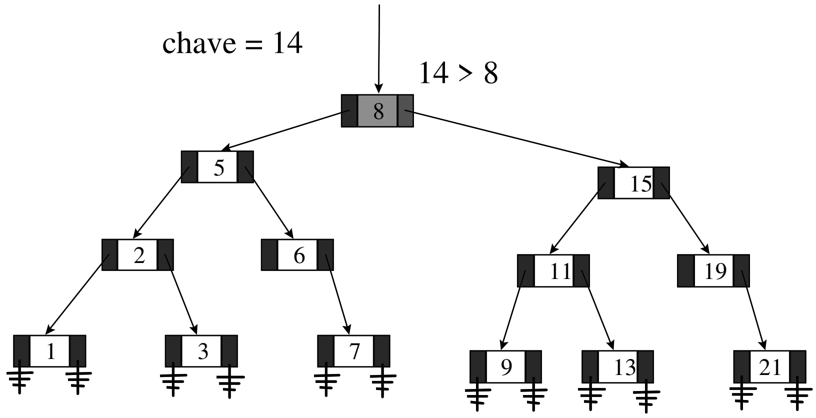
- Considerando-se a estrutura de dados da aula passada, temos como algoritmo de busca:

```
nodo *localize ( chave : info, ptr : *nodo)
        início
            enquanto (ptr ~= nulo) e
                       (ptr->info ~= chave) faça
                se (ptr->info < chave) então
                    ptr <- ptr->dir;
                senão
                    ptr <- ptr->esq;
                fim se
            fim enquanto
            retorne ptr;
        fim localize
```

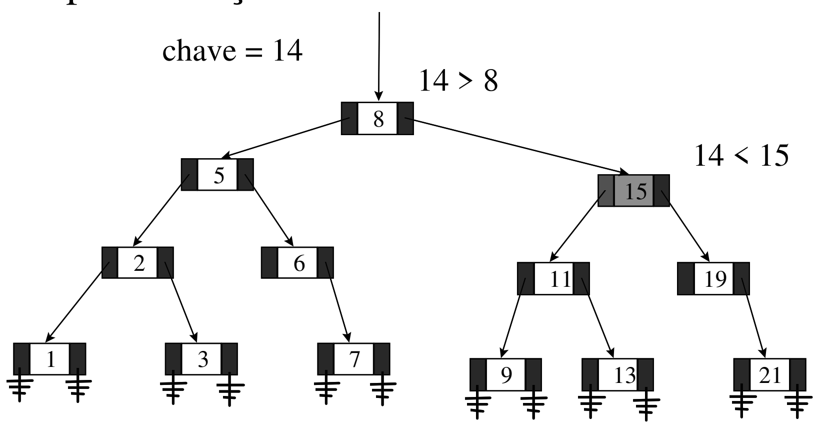

Construção: Inserção simples em árvore binária

```
inserção ( chave : info, ptr : *nodo)
    oNovo : *nodo;
    início
        se (chave < ptr->info) então
            se (ptr->esq = nulo) então
                oNovo <- aloca novo nodo;
                oNovo->info <- chave;
                oNovo->esq <- nulo; oNovo->dir <- nulo;
                ptr->esq <- oNovo;
            senão
                se (ptr->dir = nulo) então
                    oNovo <- aloca novo nodo;
                    oNovo->info <- chave;
                    oNovo->esq <- nulo; oNovo->dir <- nulo;
                    ptr->dir <- oNovo;
                senão
                    inserção ( chave, ptr->dir );
                fim se
            fim se
        fim se
    fim inserção
```

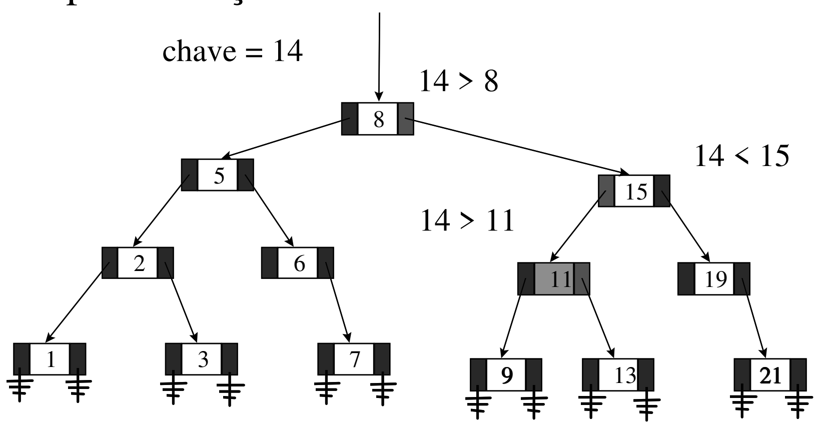
Exemplo: Inserção de um elemento com chave 14



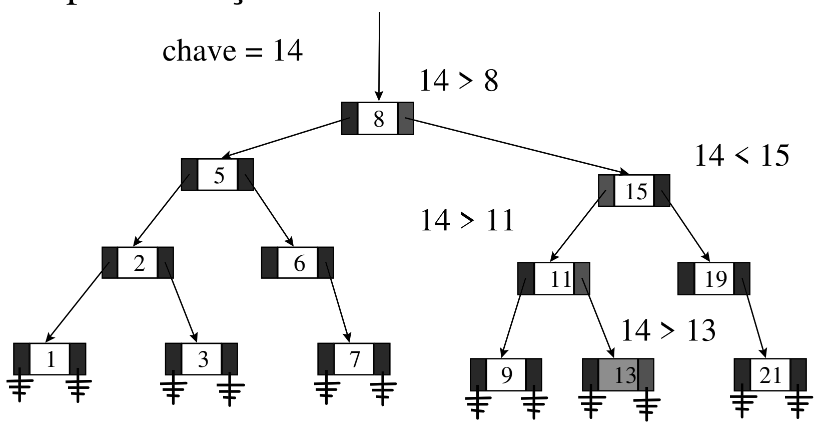
Exemplo: Inserção de um elemento com chave 14



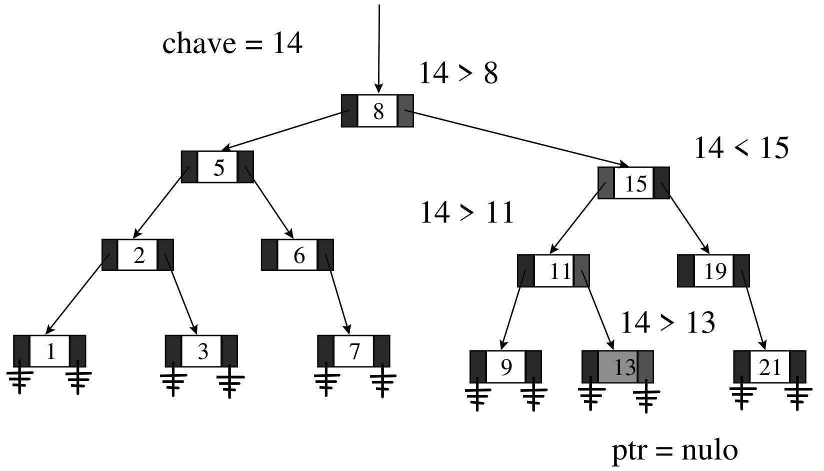
Exemplo: Inserção de um elemento com chave 14



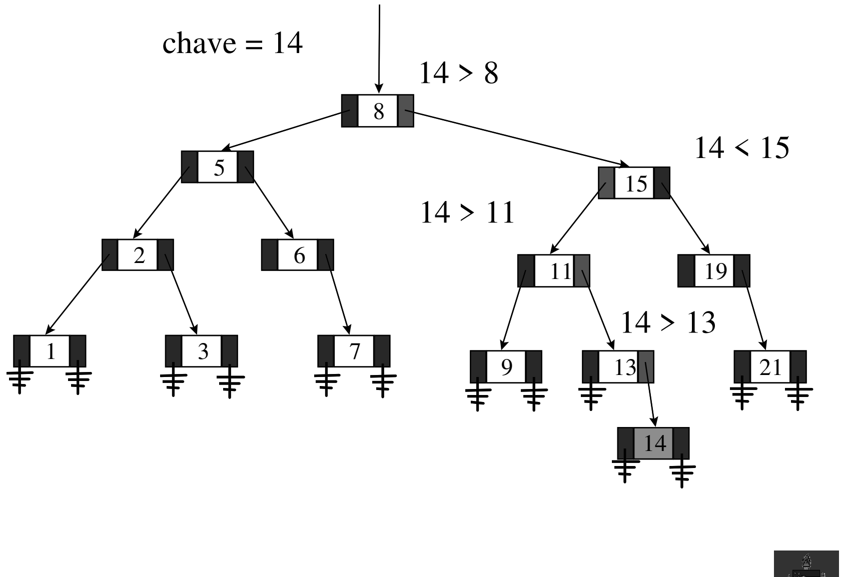
Exemplo: Inserção de um elemento com chave 14



Exemplo: Inserção de um elemento com chave 14



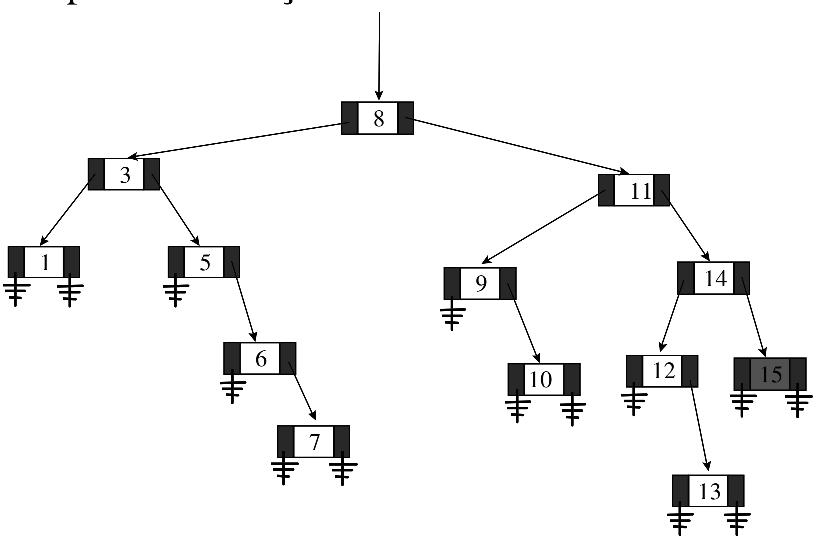
Exemplo: Inserção de um elemento com chave 14



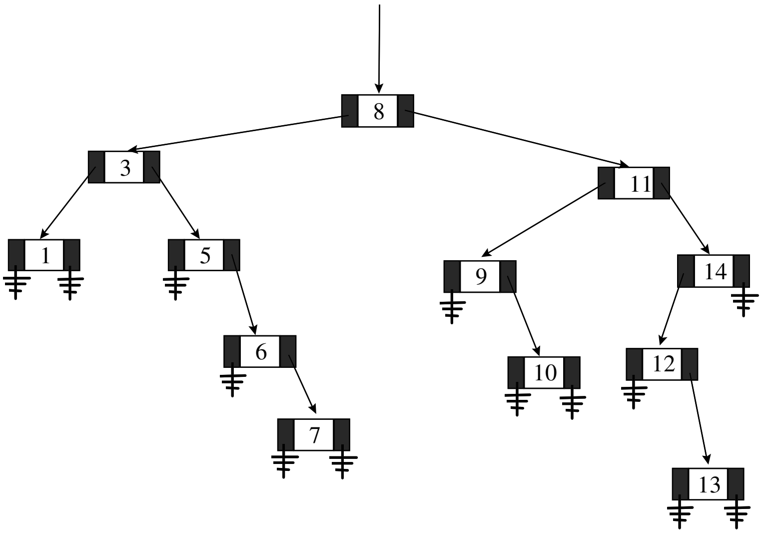
Eliminação em uma árvore de busca binária

- A eliminação é mais complexa do que a inserção
- A razão básica é que a característica organizacional da árvore não deve ser quebrada:
  - A subárvore da direita de um nodo não deve possuir chaves menores do que o pai do nodo eliminado.
  - A subárvore esquerda do nodo eliminado não deve possuir chaves maiores do que o pai do nodo eliminado.
- Para garantir isto, o algoritmo de exclusão deve remanejar os nodos.

Exemplo: Eliminação de elemento com chave 15



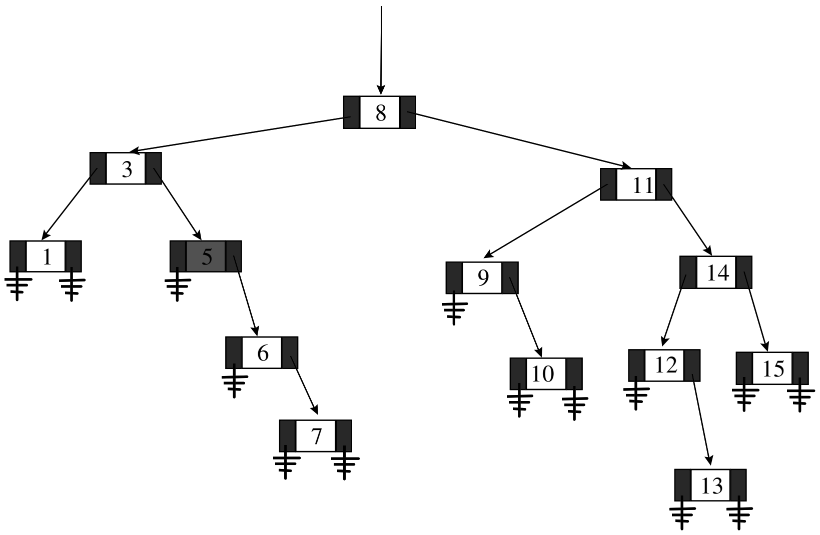
Exemplo: Eliminação de elemento com chave 15



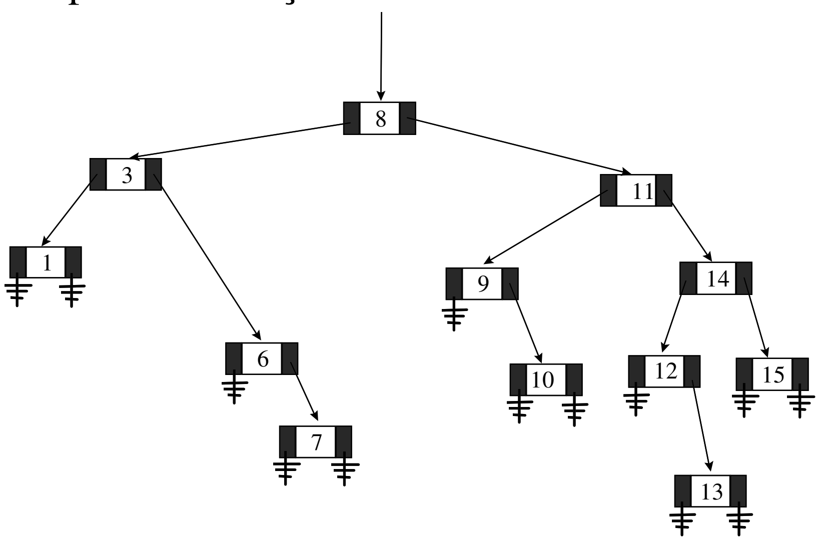
Eliminação em uma árvore de busca binária

- Se o elemento possuir somente uma subárvore filha:
  - Podemos simplesmente mover esta subárvore toda para cima.
  - O único sucessor do nodo a ser excluído será um dos sucessores diretos do pai do nodo a ser eliminado.
  - Se o nodo a ser excluído é filho esquerdo de seu pai, o seu filho será o novo filho esquerdo deste e vice versa.

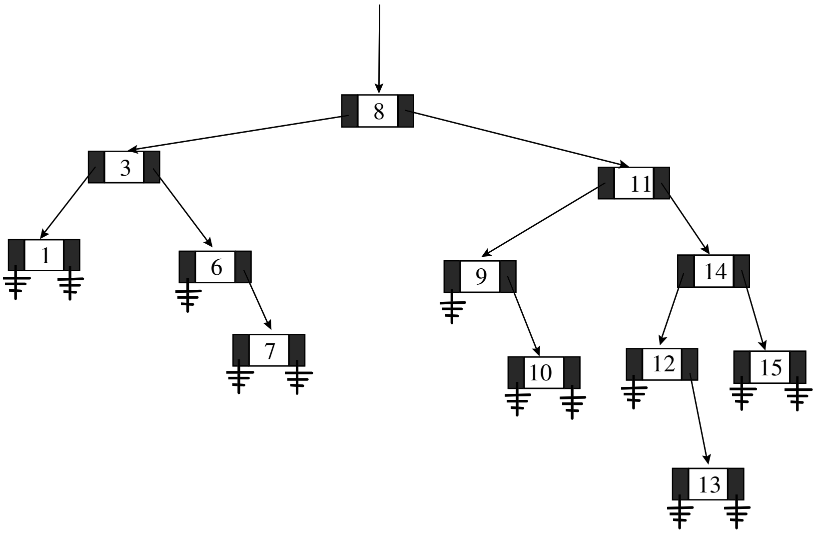
Exemplo: Eliminação de elemento com chave 5



Exemplo: Eliminação de elemento com chave 5



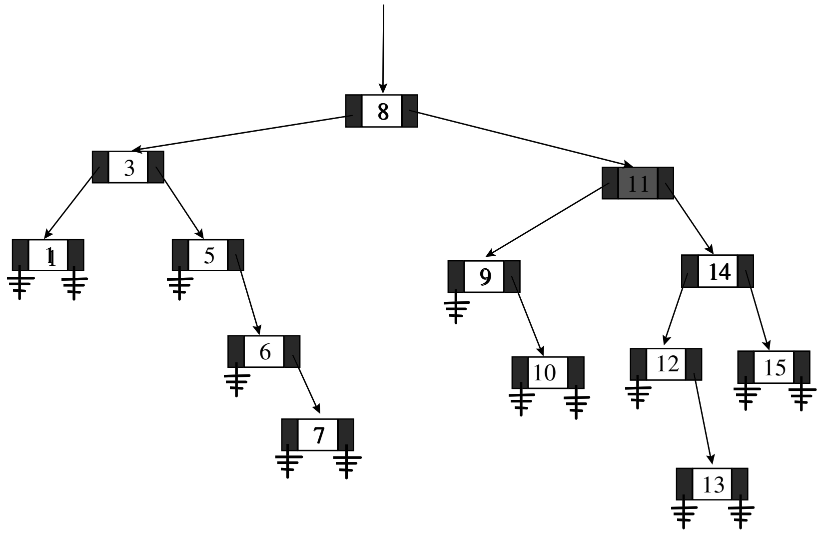
Exemplo: Eliminação de elemento com chave 5



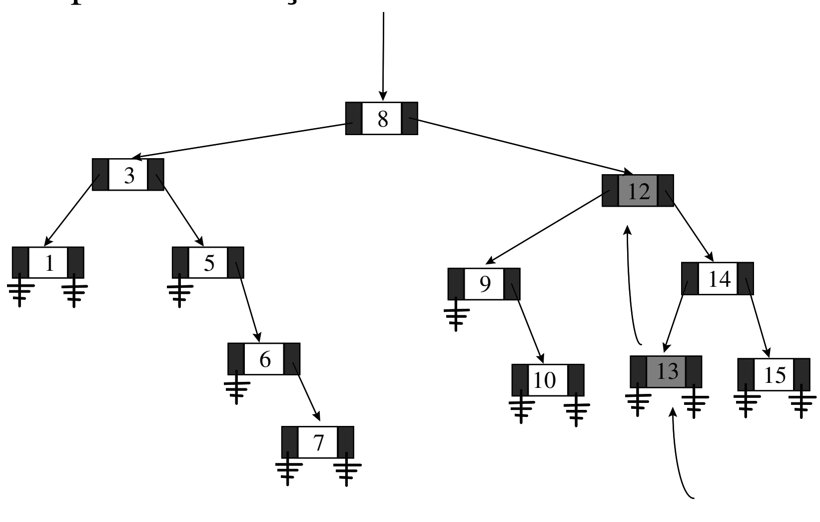
Eliminação em uma árvore de busca binária

- Se o elemento possuir duas subárvores filhas:
  - Se o sucessor direito não possui subárvore esquerda, é ele que ocupa o seu lugar.
  - Se possuir uma subárvore esquerda, a raiz desta será movida para cima e assim por diante.
  - A estratégia geral (Mark Allen Weiss) é sempre substituir a chave retirada pela menor chave da subárvore direita
    - Pode facilmente ser encontrada.

Exemplo: Eliminação de elemento com chave 11



Exemplo: Eliminação de elemento com chave 11

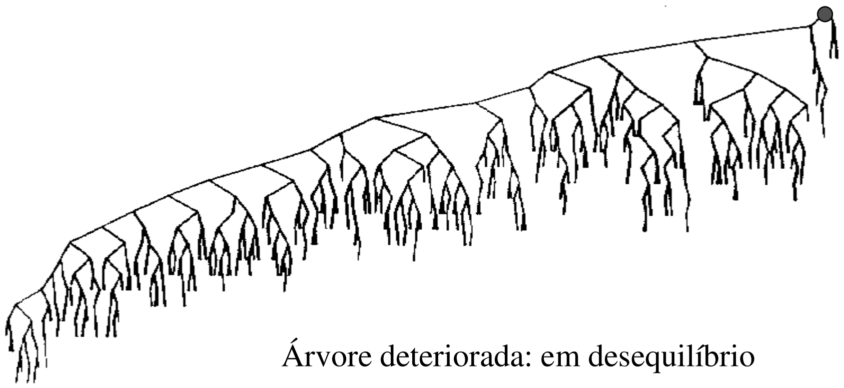


Algoritmo de Deleção em Árvore de Busca Binária

```
nodo *delete(info: tipo info, arv: *nodo)
 início
   tmp, filho: *nodo;
   se arvore == nulo então
     retorne erro;
   senão se (info < arv ->info) // vá a esq.
      arv->esq <- delete(info, arv->esq);
   senão se (info > arv ->info) // vá a dir.
      arv->dir <- delete(info, arv->dir);
   senão // encontrei elemento que quero deletar.
     se (arv->dir ~= nulo E arv->esq ~= nulo)// 2 fil
        tmp <- minimo(arv->dir);
        arv->info <- tmp->info;
        arv->dir <- delete(arv->info, arv->dir);
     senão // um filho só
        tmp <- arv;
        se (arv->esq = nulo) então // um filho à dir.
            filho <- arv->dir;
        senão se (arv->dir = nulo) então // filho esq.
            filho <- arv->esq;
        fim se fim se
        retorne filho;
     fim se
   fim se fim se
fim
```


Problemas com Árvores Binárias

- Deterioração
  - **quando inserimos utilizando a inserção simples, dependendo da distribuição de dados, pode haver deterioração.**
  - Árvores deterioradas perdem a característica de eficiência de busca.



Árvore deteriorada: em desequilíbrio

8.2.2 Árvores AVL

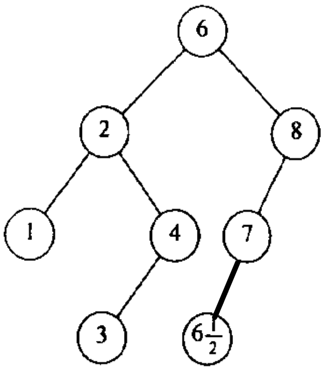
- Manter uma árvore de busca balanceada sob a presença de constantes inserções e deleções é ineficiente.
- Para contornar este problema foi criada a árvore AVL (*Adelson-Velskii e Landis*).
- A árvore AVL é uma árvore binária com uma condição de balanço, porém não completamente balanceada.
- Árvores AVL permitem inserção/deleção e rebalanceamento aceitavelmente rápidos.

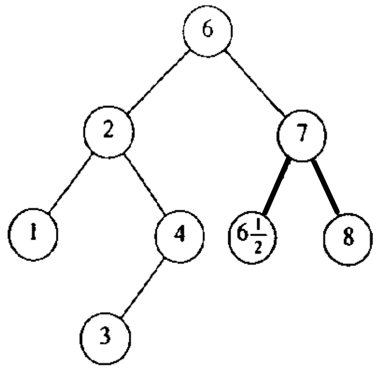
Árvores AVL

- Critério de balanceamento:
  Dado um nodo qualquer, uma árvore está AVL-balanceada se as alturas das duas subárvores deste nodo diferem de, no máximo, 1.
- Para rebalancear uma árvore após uma inserção, são utilizadas **rotações** de subárvores:
  - Rotações simples em muitos casos
  - Rotações duplas em alguns.

Árvores AVL: Inclusão com Rotação

- Exemplo de rebalanceamento:
  - O nodo com chave 6.5 desequilibrou a árvore. Com a rotação da subárvore em torno do nodo 7, rebalanceamos.



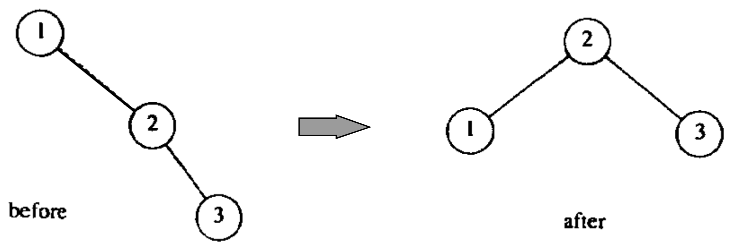


Árvores AVL: Inclusão com Rotação

- Algoritmo básico:
  - Partimos do nodo inserido e subimos a árvore.
  - Atualizamos a informação do balanceamento em cada nodo (na árvore AVL , cada nodo conhece a sua altura).
  - Se chegamos à raiz sem encontrar nada errado, terminamos.
  - Caso encontremos um nodo desbalanceado (corresponde à $|H_{esq} - H_{dir}| < 2$ ferida), realizamos a rotação no primeiro nodo desbalanceado encontrado.
    - No exemplo anterior, isto significa que, depois da inserção de 6.5, o nodo 8 ficou desbalanceado.
- Exemplo: Criação de uma árvore AVL com os n^{os} de 1 a 7

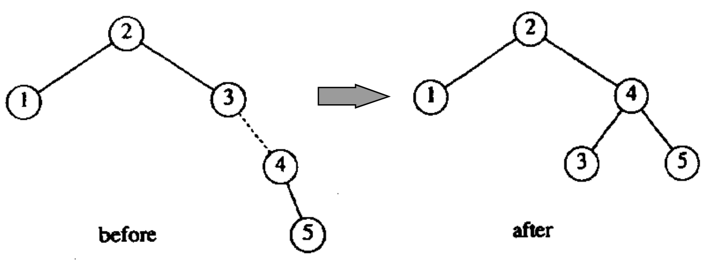
Árvores AVL: Inclusão com Rotação

- O primeiro problema ocorre quando inserimos o 3. A condição-AVL foi violada.
  - Executamos uma rotação simples entre a raiz (cuja condição-AVL está violada) e seu filho da direita.



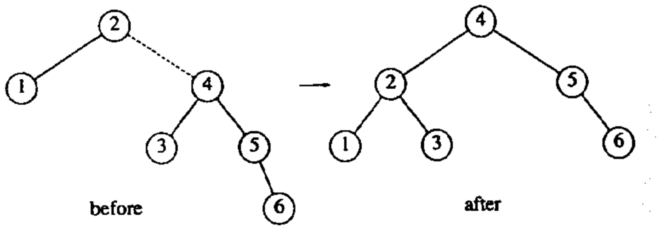
Árvores AVL: Inclusão com Rotação Simples

- Inserimos 4 sem problemas.
- Inserimos 5: violação em 3.
  - Mesmo processo da rotação anterior será seguido.
  - Importantíssimo: Observe-se que 2 precisa ser notificado que seu filho agora é o nodo com chave 4



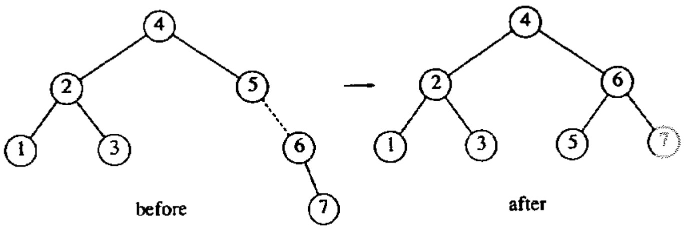
Árvores AVL: Inclusão do 6

- O 6 desequilibra a raiz:
  - Subárvore direita tem altura 0
  - Subárvore esquerda tem altura 2.
- Rotacionamos na raiz entre 2 e 4:
  - Transformamos 2 em filho de 4
  - Subárvore esquerda original de 4 é agora subárvore direita de 2. Condição de ordem da árvore de busca é mantida.



Árvores AVL: Inclusão do 7

- A inclusão de um nodo com chave 7 causa uma rotação como já havíamos visto antes:
  - 5 fica desequilibrado.
  - Rotacionamos em 6.

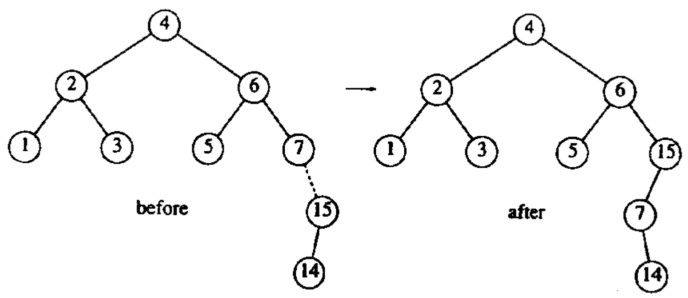


Árvores AVL: Inclusão com Rotação Dupla

- O algoritmo descrito até agora tem um problema:
  - Existem casos onde ele não basta para consertar a árvore após uma inclusão ou excliusão.
- Exemplo:
  Inserimos as chaves 8 a 15, em ordem inversa, na árvore anterior
  - A inserção de 15 não provoca problemas, nem desbalanceia a árvore.
  - A inserção de 14, por sua vez, faz o rebalanceamento falhar.

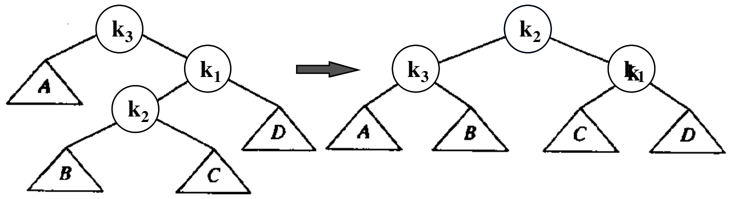
Árvores AVL: Inclusão com Rotação Dupla

- O problema que surge aqui, é que tanto 7 como 14 são candidatos à subárvore esquerda de 15.
- Neste caso, necessitamos de uma dupla rotação.



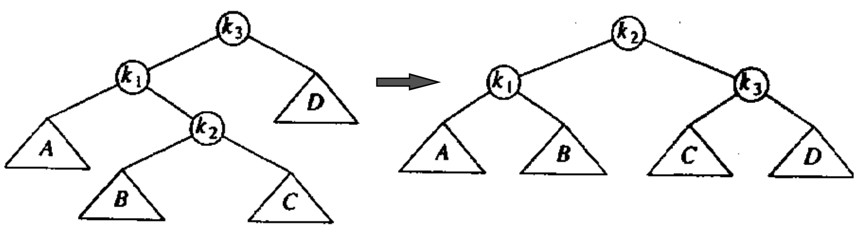
Árvores AVL: Inclusão com Rotação Dupla

- Semelhante à rotação simples, só que envolve 4 subárvores:
  - **Corresponderia a uma rotação efetuada primeiro entre k1 e k2 e depois entre k2 e k3.**



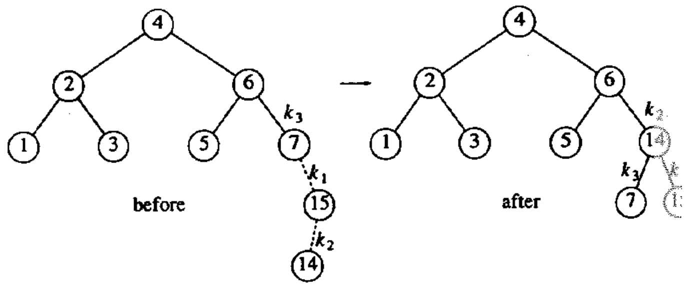
Árvores AVL: Inclusão com Rotação Dupla

- **Exemplo simétrico ao anterior**



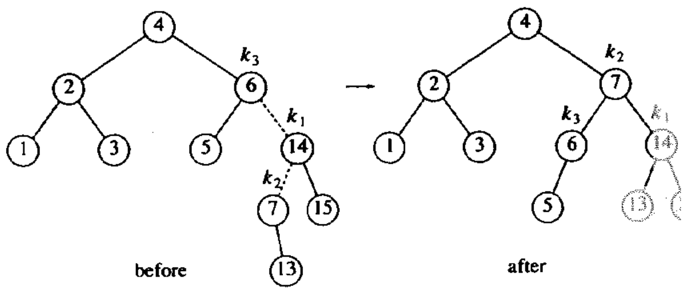
Árvores AVL: Inclusão do 14 com Rotação Dupla

- No exemplo, a rotação direita-esquerda envolve o 7, o 15 e o 1:
  - k3 é o nodo de chave 7, k1 o de chave 15 e k2 o de chave 14
  - Primeiro rotacionamos 14-15 a direita, depois 7-14 a esquerda.



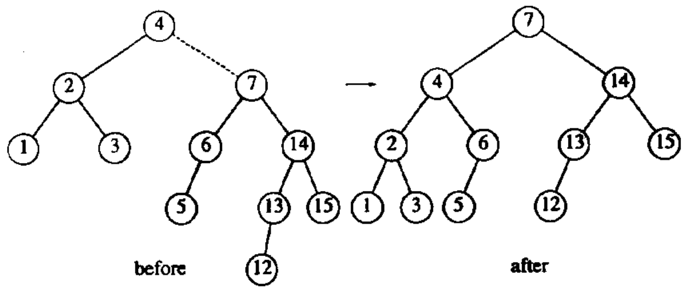
Árvores AVL: Inclusão do 13 (rot.dupla)

- Novamente uma dupla rotação direita-esquerda:
  - Desta vez a rotação envolverá 6 (k3), 14 (k1) e 7 (k2).



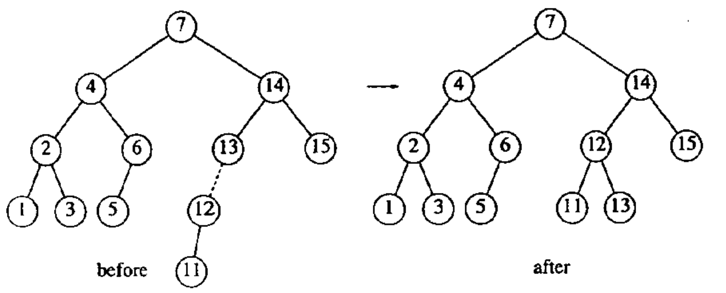
Árvores AVL: Inclusão do 12

- A inclusão do 12 provoca um desequilíbrio na raiz:
  - Como 12 não está entre 4 e 7, sabemos que uma rotação simples vai funcionar.



Árvores AVL: Inclusão do 11

- A inclusão do 11 provoca um desequilíbrio logo embaixo:
  - Como 11 não está entre 12 e 13, sabemos que uma rotação simples vai funcionar.



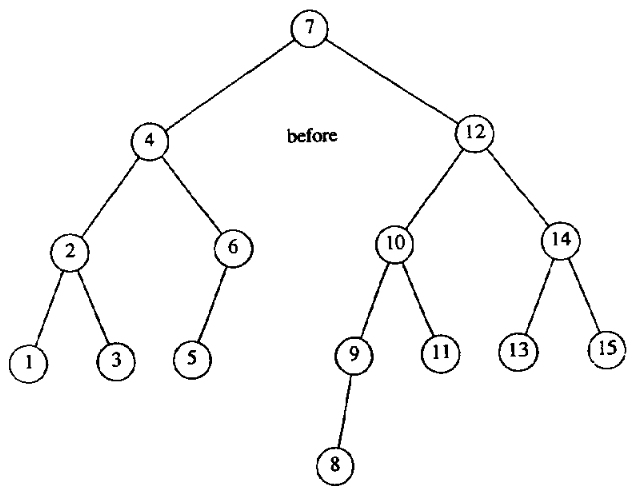
Árvores AVL: Inclusão do 10,9,8

- 10 e 9 serão inseridos com rotação simples, 8 sem rotação alguma.
  - Tente em casa.

- A árvore resultante veremos adiante.

Árvores AVL: Inclusão com Rotação Dupla

- Interessante é o caso da inserção de 8.5:



Árvores AVL: Inclusão com Rotação Dupla

- Temos uma dupla rotação que parece ser uma simples: