Remoção em Árvores Binárias de Busca

Algoritmos e Estruturas de Dados - 2025/01

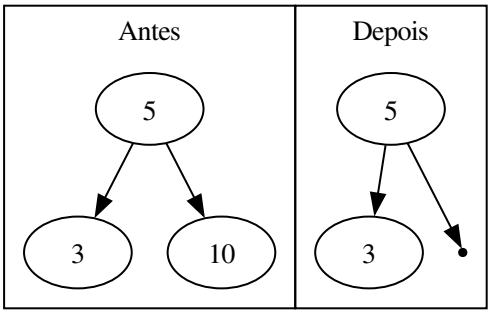
Última atualização: 27/12/2024 13:57

Remover elementos de árvores é uma operação mais complicada que inserção e busca. Esta aula será auto-guiada: os exemplos e exercícios devem levá-los à construção de um algoritmo para remoção de nós de uma árvore usando operações de rotação (que definiremos a seguir).

Os algoritmos serão comentados na próxima aula.

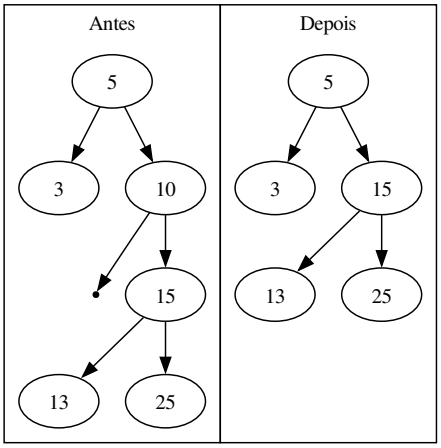
Casos fáceis da remoção

O caso mais fácil é quando **o elemento a ser removido é uma folha**.

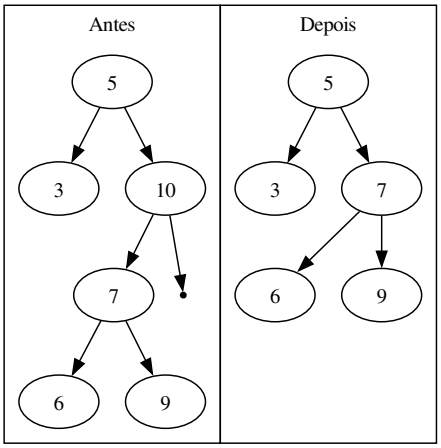


Remoção do elemento 10 - folha

Outro caso fácil é quando **o elemento a ser removido não tem um dos filhos**. Veja abaixo os dois casos possíveis.



Remoção do elemento 10 - sem filho esquerdo

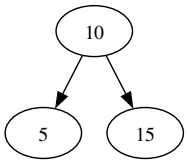


Remoção do elemento 10 - sem filho direito

O caso complicado: nó tem dois filhos

Agora falta só descobrir o que fazer quando o nó a ser removido tem ambos os filhos. Apesar de isso parecer complicado. Veremos como aplicar operações na árvore de maneira que sempre chegaremos em um caso fácil!

O caso mais simples com dois filhos é a árvore balanceada com 3 elementos e fazendo a remoção do elemento raiz (`10` no exemplo abaixo)



Para conseguirmos chegar em um caso fácil precisaríamos fazer o `10` virar

1. ou um nó com filho direito e sem filho esquerdo
2. ou um nó com filho esquerdo e sem filho direito
3. ou uma folha

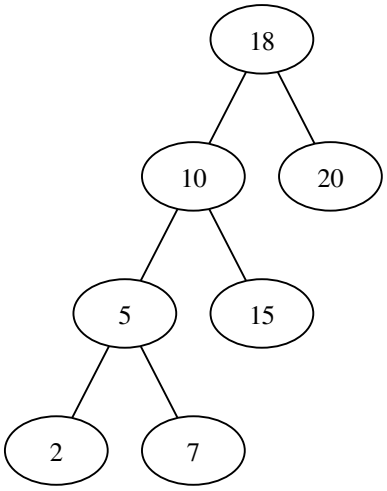
Exercício: Redesenhe a árvore acima fazendo com que o nó `10` esteja em cada uma das três posições acima.

Note que nos 3 casos o nó `10` *desceu de nível na árvore.* Ele inicia na raiz (nível 0) e desce para o nível 1 (nos casos 1 e 2) e nível 2 (no caso 3).

> Atenção
>
> Se o seu nó `10` não está nessas posições chame o professor ou valide com algum colega que já fez esse exercício

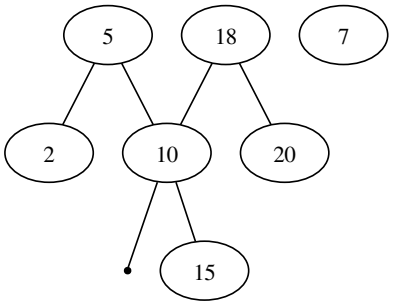
Exercício: Em uma árvore com altura $h = 10$, qual é o número máximo de rotações necessárias para fazer um nó qualquer da árvore se tornar uma folha?

Conseguimos descer um nó de nível fazendo uma **rotação** na árvore. Usaremos como referência a árovre abaixo e rodaremos o nó `10` à direita.
Iremos agora nomear alguns nós (e subárvores) para facilitar nossa vida.

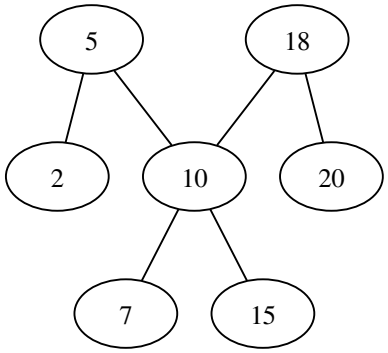


- o nó `10` será a raiz antiga
- o nó `5` será o pivô
- a subárvore enraizada no nó `7` será chamada de β

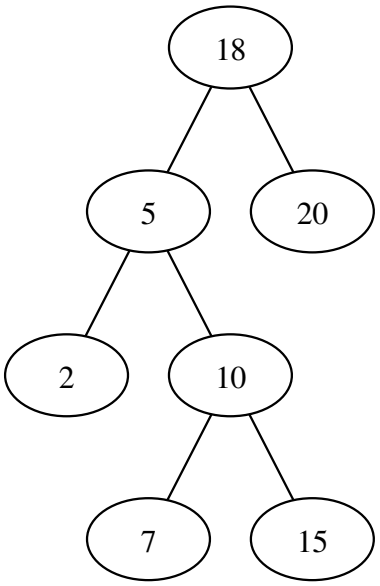
Passo 1: conectar o nó `10` (raiz antiga) à *direita* do nó `5` (pivô). Isso implica em desconectar a subárvore enraizada no nó `7` (β), que por enquanto está desconectado do resto da árvore.



Passo 2: Agora conectamos a subárvore enraizada em `7` (β) à *esquerda* do nó `10` (antiga raiz).



Passo 3: agora tornamos o nó `5` (pivô) a raiz dessa árvore, conectando-o ao nó `18` (antigo pai do `10`).



Importante

Em todos os passos a árvore fica "torta" e tem problemas.

1. No passo 1, a subárvore enraizada no nó `7` (β) ficou desconectada do resto da árvore.
2. No passo 2, o nó `10` tem dois predecessores, enquanto o nó `5` não tem nenhum.

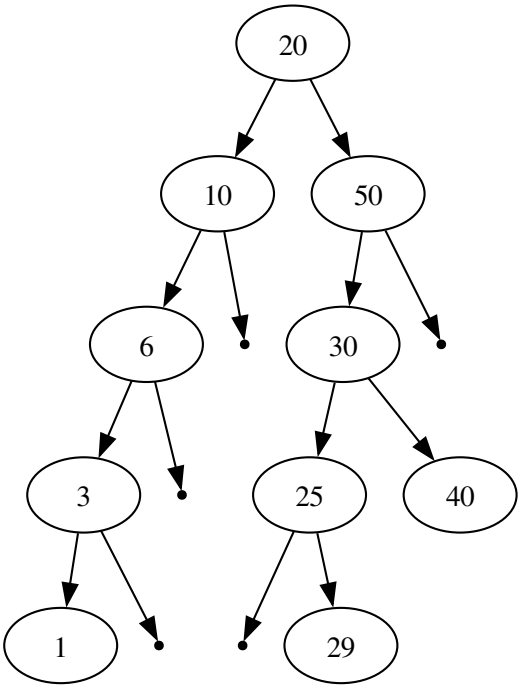
Isso é normal. Nosso objetivo é deixá-la perfeita somente após o passo 3.

Algoritmo

Formalize a explicação acima em um algoritmo chamado `ROTAÇÃO-DIREITA(R)` que o executa e devolve a nova raiz após a rotação

Exercício: Use a árvore abaixo para testar seu algoritmo e tentar realizar rotações nos seguintes nós:

- `6, 50, 30`.

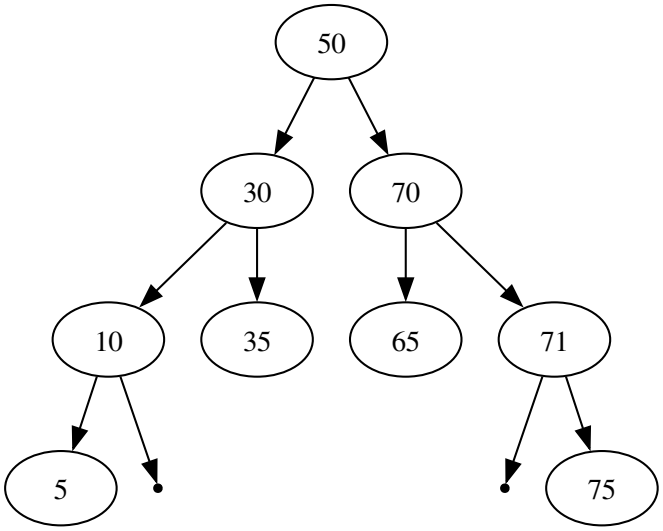


Algoritmo para remoção

Agora que conseguimos "descer" um elemento de nível, temos um algoritmo de remoção baseado em rotações que é bem simples:

1. encontre o nó a ser removido
2. rotacione a árvore para a direita até que esse nó esteja em um dos 3 casos fáceis acima.
3. remova o nó de acordo com as explicações dos 3 casos fáceis
4. devolva a nova raiz da subárvore que iniciava no elemento removido

Vamos simular essa ideia para o grafo abaixo. Todas as rotações serão feitas para a direita.



Exercício: Simule a remoção dos elementos `10, 70, 50` da árvore acima. Em cada simulação, sempre comece da árvore acima.

Notem que em alguns casos a remoção de um elemento fez a altura da árvore aumentar! Esse é uma das principais dificuldades em criar árvores balanceadas: quanto mais mexemos na árvore maior a chance de criarmos árvores altas.

Algoritmo Avançado

Existem diversas árvores que se autobalanceiam como parte das operações de inserção e remoção. O vídeo abaixo explica brevemente o funcionamento da *AVL*, uma árvore balanceada relativamente simples.

O algoritmo `REMOVE`

Agora que já simulou o algoritmo algumas vezes está na hora de formalizá-lo.

Exercício: Escreva um algoritmo `REMOVE(R, K)` que remove o nó `K` da árvore `R`, devolvendo uma nova raiz da árvore se necessário. Você pode supor que as seguintes funções existem:

- `REMOVE-RAIZ(R)` remove a raiz de uma árvore, devolvendo a nova raiz.

Algoritmo `REMOVE(R, K)`

Exercício: Escreva um algoritmo recursivo `REMOVE-RAIZ(R)` que remove a raiz `R` da árvore e devolve a nova raiz. Use rotações para a direita como foi feito no exercício anterior.

Algoritmo `REMOVE-RAIZ(R)`

Desafio

É possível fazer a remoção usando só o caso do nó escolhido ser folha. Você consegue?