Professor
Marcio Feitosa

CURSO: Estruturas de Dados

TAD ÁRVORE B

-- TEORIA --

Introdução

A Árvore B, assim como a Árvore Binária, possui um nó raiz, que é representado em cima, e os ramos, compostos por outros nós, que se espalham para baixo.

As diferenças básicas são duas:

1. Cada nó pode armazenar N elementos (na Binária apenas um)[1]
2. Cada nó pode ter até $N + 1$ filhos (na Binária no máximo dois)

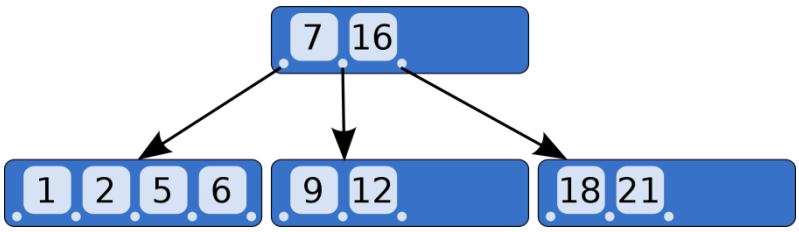

Figura 1

(Imagem extraída de: https://pt.wikipedia.org/wiki/%C3%81rvore_B)

Por ocasião da implementação de uma **Árvore B**, define-se N, o que vai determinar a chamada **ordem** da árvore.

Entre os diversos autores[2], há mais de uma definição para a **ordem** da árvore:

Tomando-se o exemplo da Figura 1, observamos que a quantidade máxima de elementos que um nó pode conter é de 4.

a) Ordem = 4 (quantidade máxima de elementos que o nó pode conter).

[1] Via de regra esse número genérico N costuma ser representado pela letra K de *key* (chave, em inglês).
[2] A Árvore B foi inventada por Rudolf Bayer e Edward Meyers McCreight em 1971. Dois cientistas que trabalhavam no Laboratório de Pesquisas da Boeing.

b) Ordem = 5 (quantidade máxima de elementos que um nó pode ter)
c) Ordem = 2 (quantidade mínima de elementos que um nó pode conter; vide explicação a seguir)

Como se pode observar, na árvore da Figura 1 o nó comporta até 4 elementos. Pela definição original da Árvore B, um nó, excetuando-se o nó raiz, não pode conter menos que $N \div 2$ elementos com arredondamento para baixo. Portanto, no exemplo, a quantidade mínima é $4 \div 2 = \mathbf{2}$ (que embasa a definição de **ordem**, letra c).

Para compatibilizar com o conceito da Árvore Binária, onde o nó armazena 1 elemento e pode ter até 2 filhos, vamos adotar a definição de **ordem** da letra **b**, onde a quantidade máxima de filhos de um nó é a quantidade máxima de elementos mais 1.

Critérios para armazenamento na árvore

- Os elementos (ou as chaves) em um nó são dispostos em sequência ordenada da esquerda para a direita.
- Os ponteiros para os filhos de um nó ficam entre os elementos, de forma que se um elemento procurado não estiver no nó e for menor que determinado elemento e maior que seu antecessor, segue-se à esquerda (para o nó abaixo), do maior elemento, ou à direita do menor.
- Novos nós são criados pela divisão de nós que estejam lotados e recebam um novo elemento.
- Nós existentes são extintos à medida que a quantidade de elementos fique abaixo da mínima permitida. Os elementos do nó extinto são realocados em outros nós.

Critério de busca

a) Toda busca se inicia pelo primeiro elemento do nó raiz
b) Corre-se a lista até encontrar-se o elemento procurado ou um elemento maior ou ainda se chegar ao final da lista[3].
c) Desce-se ao nível inferior pelo ponteiro à esquerda do maior ou, sendo maior que todos, à direita do último.
d) Entrando em um novo nó, a busca se inicia pelo primeiro elemento.
e) Volta-se ao passo (b) repetindo-se até encontrar o elemento ou constatar que não se encontra na árvore.

[3] Mais adiante vamos ver a técnica conhecida como “Busca Binária”.

Professor
Marcio Feitosa

A seguir vamos ver como movimentar os elementos em uma Árvore B.

Regras para inserção:

1) No início, os elementos devem ser inseridos na raiz até completá-la (número máximo de chaves).

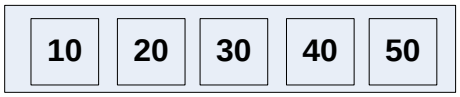

Figura 2

(supondo-se um nó de 5 chaves)

2) Próximo elemento => será necessário dividir o nó raiz em dois e promover um dos elementos para integrar um novo nó raiz. Exemplo: inserir 35.

Se N for ímpar, o elemento central (o novo elemento ainda está de fora) será eleito para compor o novo nó raiz e o novo elemento será posicionado em um dos nós recém-criados na posição correspondente.

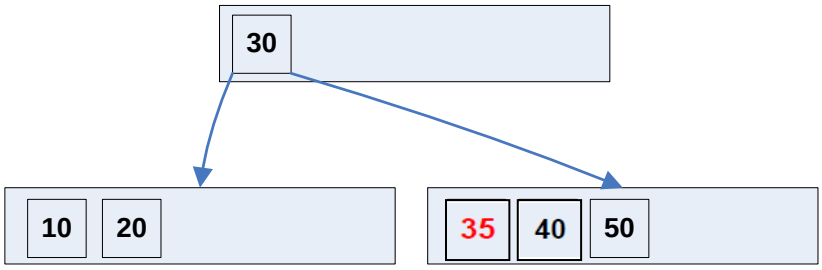

Figura 3

Se N for par, encontra-se a posição do novo elemento no nó, que neste momento ficaria com overflow de 1 elemento, e determina-se o elemento central que será promovido (que pode, inclusive, ser o próprio elemento novo).

Exemplo: inserir 35.

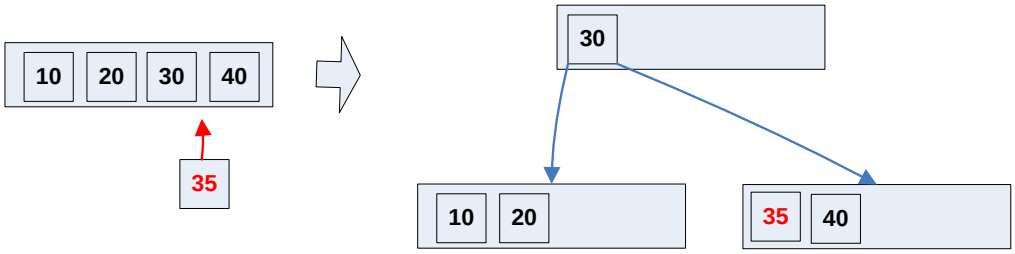

Figura 4

Exemplo: inserir 25.

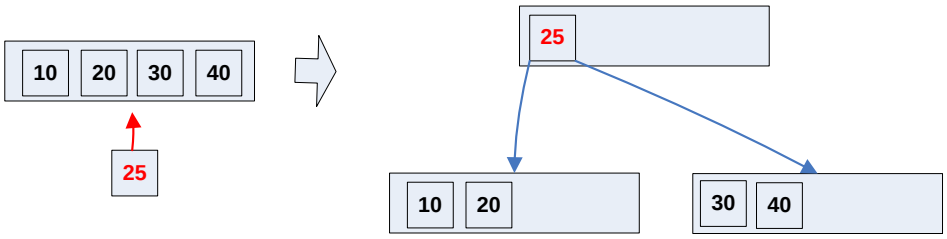

Figura 5

3) O nó folha está lotado e o nó pai tem espaço => divide-se o nó seguindo a mesma regra do item 2.

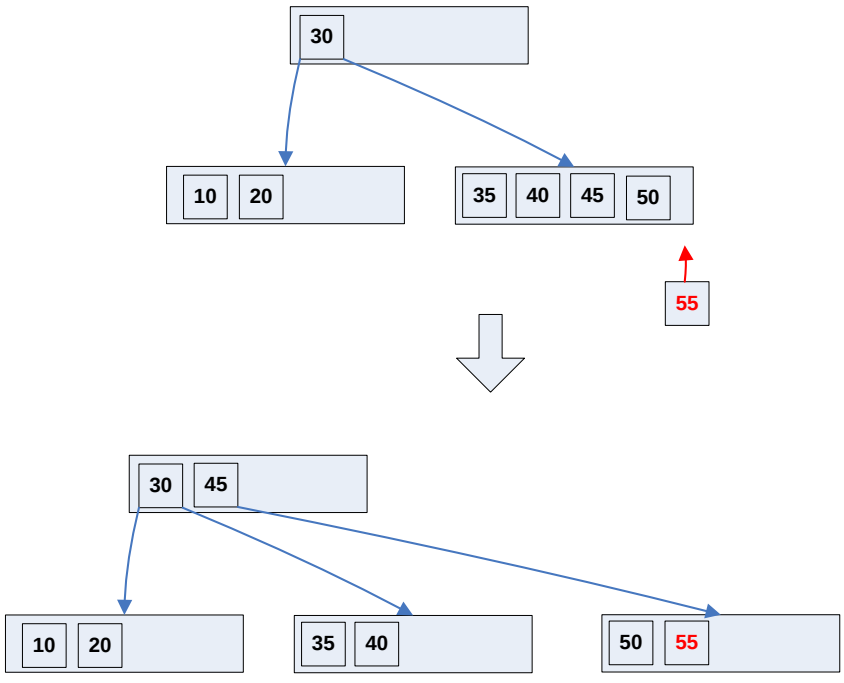

Figura 6

4) O nó folha está lotado e o nó pai também está lotado => propaga-se a divisão para cima.

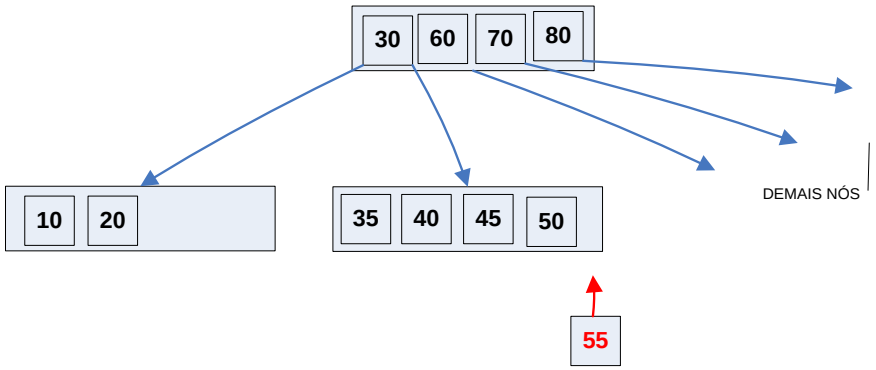

Figura 7

a) O nó que sofrerá a inserção deve ser dividido.
b) O elemento promovido será inserido no nó superior.
c) O nó superior deverá ser dividido.
d) O nó seguinte a ser promovido será inserido no seu superior. Caso não exista, será a nova raiz.

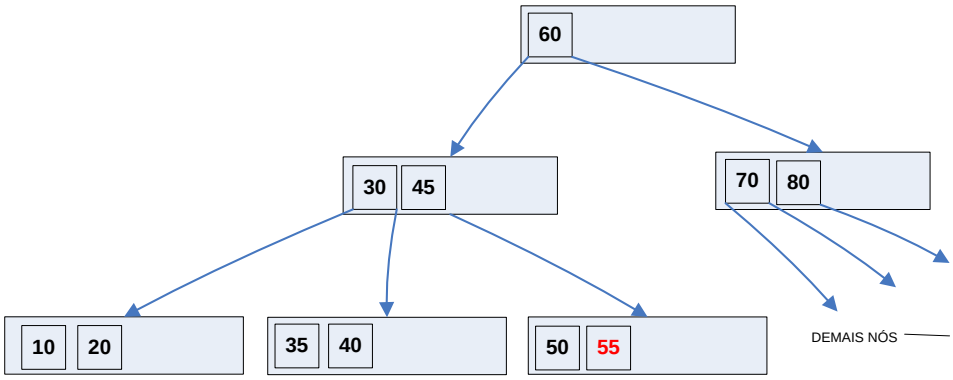

Figura 8

Regras para remoção:

1) Remover um elemento de um nó folha que tem ocupação acima do mínimo obrigatório (não é >=, é >). Neste caso é só remover e nada mais se faz necessário.

2) Remover um elemento de um nó folha que está com a lotação mínima obrigatória, ou seja, uma vez removido o elemento, o nó ficará com lotação abaixo.

a) Um dos seus nós vizinhos está com lotação acima da mínima exigida => faz-se uma rotação subindo-se um elemento e baixando-se o pai.
   a. se for o vizinho da direita, traz-se o elemento pai para o nó que sofreu a exclusão e sobe-se o elemento de **menor** valor.
   b. se for o vizinho da esquerda, traz-se o elemento pai para o nó que sofreu a exclusão e sobe-se o elemento de **maior** valor.

b) Seus dois vizinhos estão com lotação mínima => neste caso será necessário fazer um merge dos dois nós.

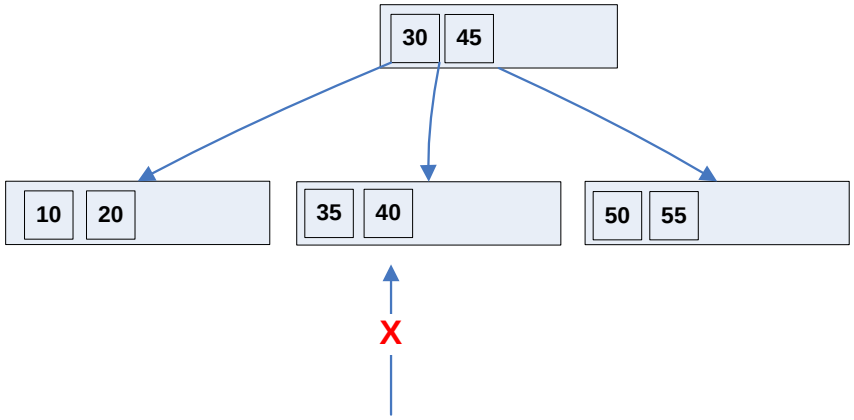

Figura 9

Se o nó tiver vizinhos de ambos os lados, pode-se escolher com qual nó será feito o merge. Um nó será eliminado, portanto o pai dos dois (pai de valores inferiores do nó esquerdo e pai de valores superiores do nó direito) deverá descer e integrar no novo nó.

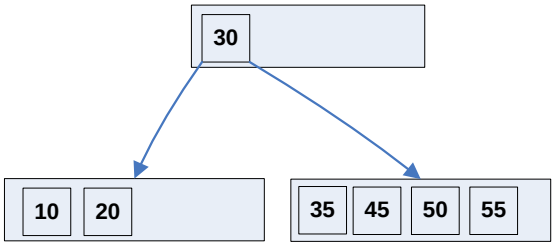

Figura 10

Observe que o 40 sumiu e o 45, antes pai dos dois nós, desceu e agora faz parte do novo nó "mergeado".

3) Remover um elemento de um nó intermediário => a regra geral é trocá-lo pelo seu antecessor.

O antecessor é o maior elemento do lado esquerdo. Ou seja, desce-se um nó à esquerda e em seguida toma-se a rota à direita até o final (deverá estar em nó folha, não importando a quantidade de níveis que se irá descer).

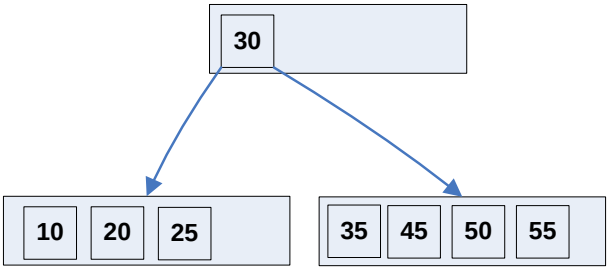

Figura 11

Na Figura 11, para eliminarmos o elemento 30, localizamos seu antecessor e o colocamos no lugar.

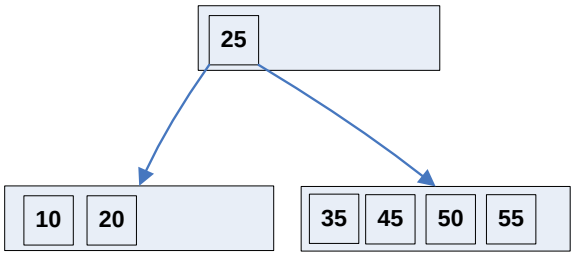

Figura 12

Agora, para eliminar o 25, substitui-se pelo 20. Porém o nó à esquerda ficará com lotação abaixo da mínima. Como o nó à direita tem lotação acima da mínima, pode-se fazer a rotação à esquerda.

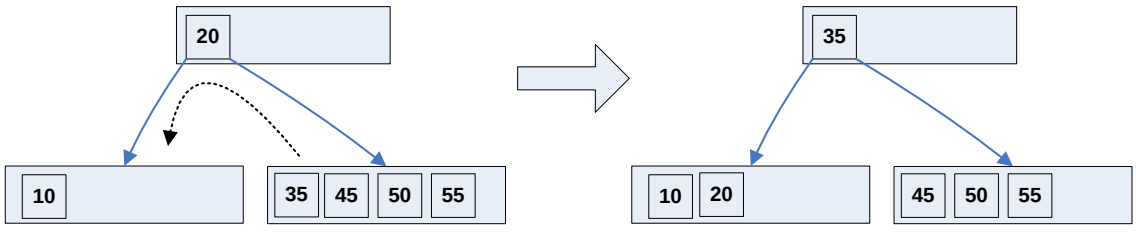

Figura 13

Caso os nós vizinhos tenham lotação mínima, deve-se fazer o procedimento de merge conforme explicado anteriormente.

Professor
Marcio Feitosa

Finalização

As maiores aplicações da Árvore B fogem um pouco do nosso escopo, que se concentra em estruturas de dados em memória.

Por comportar quantidades massivas de dados com bom desempenho nas inserções e remoções, assim como na busca, além do que já é uma árvore naturalmente balanceada, é extensamente utilizada para estruturar o armazenamento de dados em mídias persistentes, como, por exemplo, os HD e atuais SSD.

Por razões diversas, incluindo análises teóricas e experimentações práticas, um número de **ordem** frequentemente encontrado nas implementações de Árvores B gira em torno de 100. Pode-se, no entanto, encontrar casos de 1000 assim como de 5.

É possível se implementar um Árvore B de ordem 2, assemelhando-se a uma Árvore Binária, mas entende-se que os benefícios inerentes da Árvore B são minimizados, de forma que é raro se encontrar uma implementação de ordem menor que 11.

-o-o-o-o-o-o-o-o-o-o-o-