Árvores B
Parte 1

Jander Moreira*

25 de Agosto de 2017

Conteúdo

Palavras-chaves: árvore, árvore B, organização e recuperação da informação

Tempo de leitura estimado: 1.1 hora(s)

1 Introdução

Árvores de busca (ou árvores de pesquisa) são estruturas para armazenamento de dados organizadas para que seja viável a recuperação eficiente de um item armazenado. Além das pesquisas, outras operações podem ser usadas nessas árvores: inserção, remoção, máximo, mínimo, predecessor e sucessor. (CORMEN et al., 2002)

As árvores mais comumente vistas como estruturas de dados são as árvores binárias, especificamente as árvores binárias de busca. Buscando-se eficiência, essas árvores também existem em versões balanceadas, como AVL e vermelho-preto (CORMEN et al., 2002; DROZDEK, 2010).

Há, entretanto, árvores de busca que, diferentemente das árvores binárias, permitem a existência de mais de dois filhos para cada nó. Essas árvores são chamadas árvores multicaminhos (LANGSAM; AUGENSTEIN; TENENBAUM, 1990) ou árvores múltiplas (DROZDEK, 2010).

Uma árvore multicaminho é uma árvore de busca, de forma que as mesmas operações citadas anteriomente encontram suporte nela. Sua organização usa critérios de ordem para as chaves armazenadas. São, assim, mantidas as chaves de menor valor "à esquerda" e as de maior valor "à direita". Há, porém, valores em intervalos intermediários, para os demais nós filhos existentes.

A Figura 1 mostra um exemplo de árvore multicaminho. Cada nó consiste de um conjunto de chaves (dados) e de ponteiros. Os ponteiros para os nós filhos seguem a lógica da ordenação. Por exemplo, o nó que contém as chaves 22 e 36 possui ponteiros para nós com valores menores que 22 (à direita do 22), nós com valores entre 22 e 36 (ponteiro entre essas duas chaves) e para nós com valores maiores que 36 (à direita do 36). No caso de se admitirem chaves repetidas, pode-se escolher um dos caminhos, desde que mantida a coerência para toda a árvore. Os filhos inexistentes são indicados por um ponteiro nulo (representado pela linha sem a ponta de seta na figura). A Figura 1 é a única que mostra os ponteiros nulos; nas demais figuras eles serão omitidos, mas é importante que se lembre que eles estão lá.

Um requisito importante da árvore é que, havendo um filho para um nó, necessariamente deve haver uma chave que contenha o critério de descida que permita chegar ao nó.

Na prática, como cada nó de uma árvore multicaminho pode conter várias chaves, há a tendência de que sua altura seja menor que a de uma árvore binária com os mesmos valores armazenados. O número de nós consultados, então, em uma busca na árvore tende também a ser menor. Existe, por outro lado, a necessidade de pesquisa da chave dentro do nó, o que também tem custo computacional.

2 Árvores B

Uma categoria das árvores multicaminho engloba as chamadas árvores B. Uma árvore B possui restrições tanto de estrutura quanto de organização dos dados, que definem suas características e suas aplicações.

As árvores B são, assim, árvore multicaminho com as seguintes características:

*Universidade Federal de São Carlos – Departamento de Computação – *Rodovia Washington Luis, km 235 – 13565-905 - São Carlos/SP – Brasil* – jander@dc.ufscar.br

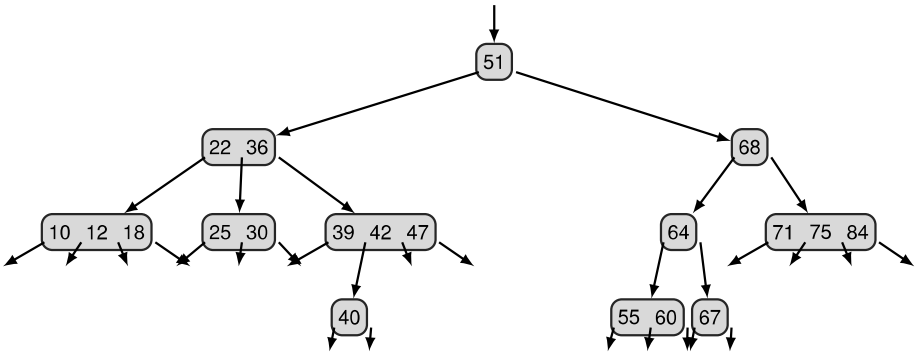

Figura 1: Exemplo de árvore multicaminho de busca. As linhas sem pontas de seta indicam ponteiros nulos.

- É mantida balanceada permanentemente;
- Se um nó possui filhos e contém k chaves, então necessariamente terá $k+1$ filhos;
- É definido um valor máximo n para o número de filhos, o que define, por consequência, o número máximo de chaves por nó em $n-1$;
- É definido um número mínimo de chaves para cada nó.

As árvores B são árvores balanceadas. O critério que define o balanceamento da árvore é que todos os nós folhas[1] sempre estejam no mesmo nível da árvore.

Ao se estabelecer que, existindo k chaves em um nó interno necessarimente existem $k+1$ nós filhos, fica definido que a árvore é sempre completa. Isso torna todas as chaves existentes nas árvores "úteis". Em outras palavras, se uma chave existe em um nó interno, então ela tem a função de ser o critério de descida para um nó filho a sua direita e a sua esquerda.

A estruturação da árvore B define um valor n, que é o número máximo de filhos que qualquer nó da árvore pode ter. Esse valor n é chamado ordem da árvore[2]. Desta forma, uma árvore de ordem 15 é aquela que possui, no máximo, 15 filhos por nó. Considerando-se que todas as chaves de nós intermediários devem ser critérios para separar dois nós filhos, tem-se como consequência que o número máximo de chaves que um nó da árvore pode ter é $n-1$. Em uma árvore de ordem 15, o número máximo de chaves por nó fica restrito a 14, por exemplo.

O número mínimo de chaves que um nó pode ter é um critério para que os nós da árvore não fiquem subutilizados. De forma geral, considera-se que pelo menos metade da capacidade de um nó esteja ocupada. Assim, sendo n a ordem de uma árvore, a metade do número de chaves é $\lfloor \frac{n-1}{2} \rfloor$ (função piso[3].). Por exemplo, se a ordem for 15, então o número mínimo de chaves por nó é 7; se for 6, então o mínimo é 2. A raiz da árvore é o único nó que é exceção a essa regra. Se não fosse assim, algumas configurações de árvores não seriam possíveis.

A Figura 2 mostra um exemplo de uma árvore B de ordem 5, sem repetição de chaves. Para a ilustração, todos os nós foram desenhados com tamanho igual para indicar que podem ser armazenadas até 4 chaves em cada um deles. Somente os ponteiros usados foram representados e os nulos, omitidos.

Nessa árvore podem ser observadas as características da árvore B:

- Todas os nós folhas estão no mesmo nível, caracterizando a árvore como balanceada;
- Possuindo ordem 5, o número máximo de filhos por nó é 5, o que ocorre apenas com um dos nós da árvore;

[1] Em contraposição aos *nós folhas* (ou seja, que não têm filhos), os nós com filhos são chamados *nós internos*.

[2] Não há uma definição precisa para a ordem da árvore B. Alguns autores consideram como ordem o número máximo de filhos, enquanto outros o número máximo de chaves. Deve-se, portanto, sempre haver atenção ao que um dado texto efetivamente considera como ordem da árvore.

[3] A função piso $\lfloor x \rfloor$, com $x \in \mathbb{R}$, é definida como o maior valor inteiro que seja menor ou igual a x

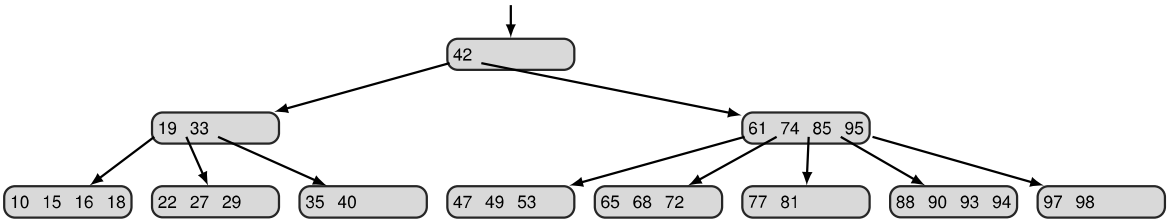

Figura 2: Exemplo de uma árvore B de ordem 5.

- A raiz, como exceção ao número mínimo de chaves por nó, possui apenas uma chave;
- Todos os demais nós possuem pelo menos duas chaves, dado que é o número mínimo possível para essa ordem;
- Todos os nós internos possuem número de filhos uma unidade maior que o número de chaves, não havendo nenhuma chave que não seja critério de descida para dois nós filhos;
- O número de nós que devem ser passados até se chegar a qualquer uma das folhas é o mesmo (neste caso, três nós incluindo-se a própria folha).

Na árvore B ilustrada é possível verificar que qualquer chave pode ser recuperada em um máximo de três consultas a nós.

Todas as operações sobre as árvores B baseiam-se no conceito de minimizar a modificação da estrutura da árvore. Para isso, os algoritmos são projetados para alterar o menor número de nós. Os efeitos das modificações feitas na árvore, sejam por inserção ou por remoção, têm seu raio de ação tão limitado quanto possível.

As próximas subseções apresentam os algoritmos para busca e inserção na árvore B.

2.1 Busca

O procedimento de recuperação de um dado na árvore, dada sua chave, usa a lógica da organização de árvore de busca.

Toda busca se inicia na raiz e, a partir dela, decide-se por qual dos ramos da árvore a pesquisa deve continuar. O limite da busca é, naturalmente, determinado ao se chegar a um nó folha.

Na Figura 3 é apresentado o algoritmo de busca. A primeira chamada deve passar o ponteiro para o nó raiz. A verificação para o ponteiro nulo é necessária para o caso da árvore estar vazia[4]. A cada chamada recursiva, procura-se pela chave no nó corrente e, caso ela não seja encontrada, o ramo adequado da árvore é escolhido para a descida. A recursão termina sem sucesso quando se chega a um nó folha sem que a chave buscada tenha sido localizada.

2.2 Inserção

O procedimento de inserção requer maior complexidade, pois é preciso acrescentar uma nova chave à árvore e ainda mantê-la balanceada.

Para conseguir esse objetivo, a inserção sempre coloca uma nova chave em um nó folha. Se há espaço no nó folha para acomodar a nova chave, então ela é inserida nesse nó e o processo termina. Porém, é possível que se chegue a um nó folha que tenha atingido o número máximo de chaves. Nesse caso, o nó sofre um processo de divisão (*split*).

A divisão consiste em se criar um novo nó no mesmo nível do nó que foi dividido. As chaves são distribuídas entre os dois nós (o original e o novo). Criando-se o novo nó à direita do original, a divisão da chaves leva a deixar no nó original as chaves menores, enquanto as maiores são transferidas para o novo nó. Como esse novo nó precisa fazer parte da árvore, é necessário que uma das chaves seja selecionada como critério de separação. O valor da mediana das chaves é a melhor opção. Essa chave escolhida é dita *promovida* para o nível imediatamente superior.

As etapas para a divisão de um nó são exemplificadas na Figura 4. Deve-se observar que, das cinco chaves que deveriam ficar no nó original,

[4]Em algumas versões, a árvore vazia pode ser alternativamente indicada por uma árvore somente com o nó raiz, o qual não contém nenhuma chave, como em Cormen et al. (2002).

Descrição: Busca em profundidade na árvore B para localização de uma dada chave
Entrada: O ponteiro para um nó da árvore B (inicialmente a raiz) e o valor da chave a ser localizada
Saída: O valor da chave encontrada, por referência, ou um valor indeterminado se a chave não existir
Retorno: `Verdadeiro` caso a a chave seja localizada; `Falso` caso contrário

```
1  função BUSQUE(nodu atual, chave)
2      se o nodu atual for um ponteiro nulo
3          # Caso da árvore vazia
4          retorne Falso
5      senão
6          se a chave está no nodu atual
7              # Localização da chave procurada
8              Armazene em chave o valor encontrado      # retorno por referência
9              retorne Verdadeiro
10         senão
11             # Necessidade de busca nos níveis inferiores
12             se o nodu atual é do tipo folha
13                 # Não há como descer mais
14                 retorne Falso
15             senão
16                 # Busca no próximo nível da árvore
17                 Determine o nodu filho para descer
18                 retorne BUSQUE(nodu filho, chave)
```

Figura 3: Algoritmo de busca na árvore B.

duas ficam no próprio nó, duas ficam no novo nó (criado à direita do original) e a mediana é destacada para promoção.

O algoritmo de divisão (Figura 5) mostra, em mais alto nível, como o processo de divisão ocorre. O algoritmo faz com que a coleção total de chaves (as já existentes no nó mais a nova que está sendo inserida) seja dividida em três grupos. Um grupo contém apenas a mediana, enquanto os outros dois mantém as chaves menores que a mediana e o outro, as maiores.

No contexto da árvore toda, a inserção segue a mesma lógica da busca, descendo a árvore a partir da raiz para inserir a nova chave no nó folha apropriado.

O algoritmo de inserção é dividido em duas partes: a primeira trata do nó raiz e a segunda realiza a busca descendente na árvore para determinar o nó folha no qual haverá a inserção da nova chave.

O tratamento do nó raiz é feito pelo algoritmo apresentado na Figura 6 e corresponde à primeira parte. Ele cuida da primeira inserção, quando a árvore está vazia e, caso a árvore já tenha nós, executa a segunda parte do algoritmo de inserção.

A segunda parte do algoritmo, que é faz a descida na árvore recursivamente, usa a lógica de organização da árvore para localizar o nó folha em que a nova chave deve ser inserida. Toda nova inserção ocorre no nível das folhas. A sub-rotina que realiza esta etapa é apresentada na Figura 7.

Depois de cada chamada recursiva é verificado se houve uma divisão de nós no nível imediatamente inferior e, tendo havido, a chave que foi promovida é tratada. Uma divisão em um nó pode implicar em uma cadeia de divisões dos nós no caminho de subida, se todos estiverem com a capacidade máxima de chaves.

Para ilustrar a inserção são apresentados três exemplos.

O primeiro considera a árvore apresentada na Figura 2 com a inserção da chave 20. A função INSIRA é chamada, verifica que a árvore não está vazia e aciona a busca recursiva por meio da função BUSQUEEINSIRA. Na primeira instância de cha-

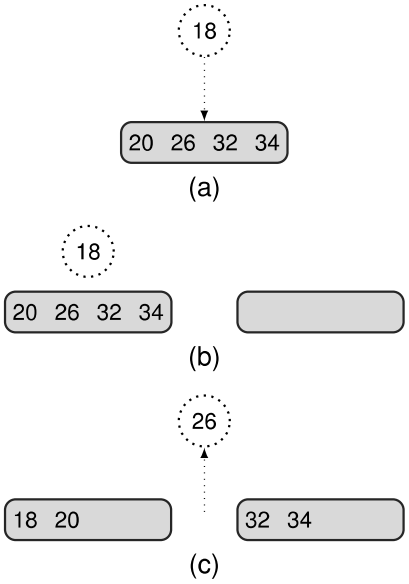

Figura 4: Etapas para a divisão de um nó, considerando-se ordem 5: (a) Tentativa de inserção da chave 18 em um nó já completo; (b) Criação do novo nó à direita; (c) Distribuição das chaves – 18 e 20 no nó original, 32 e 34 no novo nó e escolha da mediana (26) para promoção

mada de BUSQUEEINSIRA, o $nodu\ atual$ é a raiz da árvore que, não sendo uma folha, determina que a inserção se dará à esquerda da chave 42 ($20 < 42$). Na nova instância da função, o nó contendo as chaves 19 e 33 é verificado e, não sendo também folha, opta-se por descer pelo filho entre o 19 e o 33 ($19 < 20 < 33$). Na última chamada recursiva, chega-se ao nó folha contendo as chaves 22, 27 e 29. Nele há espaço para a nova chave, que é inserida mantendo-se a ordenação interna do nó. O valor `Verdadeiro` é retornado por cada uma das instâncias e o parâmetro $houve\ promocao$ fica com valor `Falso` desde o nível das folhas até retornar à raiz. O resultado da inserção é apresentado na Figura 8.

O segundo exemplo mostra a inserção da chave 11 na árvore da Figura 8. A primeira chamada é para a função INSIRA, que verifica que a árvore já existe e repassa a raiz para a função BUSQUEEINSIRA.

Na primeira instância de chamada de BUSQUEEINSIRA, opta-se pelo nó à esquerda da chave 42; na segunda, escolhe-se o ramo à esquerda da chave 19, chegando-se, na terceira instância, ao nó contendo as chaves 10, 15, 16 e 18.

No nó folha atingido, não há espaço para a nova chave, o que implica a chamada à função DIVIDA. O resultado dessa divisão é mostrado na Figura 9(a). O destaque é para a promoção da chave 15, que é a mediana. O algoritmo faz com que os parâmetros de saída $houve\ promocao$, $chave\ promovida$ e $novo\ nodu$ tenham seus valores ajustados, respectivamente, para `Verdadeiro`, para o valor 15 e para o ponteiro para o nó criado à direita do que foi dividido.

Terminada essa instância da chamada, a chamada anterior verifica que $houve\ promocao$ é `Verdadeiro` e, assim, trata a divisão que houve no nível imediatamente inferior. No caso do exemplo, a chave 15 é acomodada no nó (mantendo-se a ordenação interna) e o ponteiro a direita do 15 é ajustado para apontar para $novo\ nodu$, acrescentando o nó recém-criado à árvore. Com o tratamento da chave que foi promovida, $houve\ promocao$ é alterada para `Falso`, pois não há mais nada a ser tratado nesta inserção. O resultado final da inserção é apresentado na Figura 9(b).

É interessante notar que, quando uma chave qualquer é deslocada de local dentro da árvore, juntamente com ela é copiado o ponteiro a sua direita.

No terceiro exemplo são ilustradas duas divisões de nós sucessivas. A árvore inicial é mostrada na Figura 10(a), na qual é feita a inserção da chave 68. A busca pelo nó folha para a inserção chega ao nó com as chaves 61, 64, 70 e 71. Como não há espaço no nó para mais uma chave, é feita a divisão pelo procedimento DIVIDA (Figura 10(b)). A chave promovida pela divisão, neste caso, é 68 (mediana), que é repassada para o nível superior.

Não há espaço novamente para acomodar a chave 68, o que implica em nova divisão de nó. Um novo nó interno é criado (Figura 10(c)), as chaves são distribuídas entre eles e se determina uma nova chave para promoção. Neste caso, a mediana é a chave 60.

Em particular, neste exemplo, houve a divisão do nó raiz, o que é tratada pela função INSIRA. O procedimento é usar a chave promovida para criar uma nova raiz, que é critério para discriminar entre os dois nós (o original à esquerda e o recém-criado à direita).

Descrição: Divisão de um nó de uma árvore B
Entrada: O ponteiro $nodu$ para um nó já completo de chaves, uma $nova\ chave$ a ser inserida e o ponteiro $nodu\ direita$
Saída: O ponteiro para o $novo\ nodu$ criado, por referência, e o valor da chave mediana $chave\ promovida$, por referência

1 **procedimento** DIVIDA($nodu$, $nova\ chave$, $nodu\ direita$, $novo\ nodu$, $chave\ promovida$)
2 *# Criação do novo nó*
3 Aloque o novo nó $novo\ nodu$
4 Ajuste o tipo de $novo\ nodu$ para o mesmo de $nodu$
5 *# Distribuição dos nós entre os dois nós*
6 Defina m com o valor $\left\lfloor \frac{ordem}{2} \right\rfloor$
7 **se** o valor de $nova\ chave$ for menor que a chave na posição m de $nodu$
8 Transfira de $nodu$ para $novo\ nodu$ as chaves a partir da posição m, juntamente com respectivos ponteiros à direita
9 Insira a $nova\ chave$ em $nodu$
10 **senão**
11 Transfira de $nodu$ para $novo\ nodu$ as chaves a partir da posição $m+1$, juntamente com respectivos ponteiros à direita
12 Insira a $nova\ chave$ em $novo\ nodu$
13 Ajuste o ponteiro à direita $nova\ chave$ para apontar para $nodu\ direita$
14 *# Definição da mediana como chave promovida e ajuste de ponteiros*
15 Transfira o último ponteiro de $nodu$ para o primeiro ponteiro de $novo\ nodu$
16 Remova a maior chave de $nodu$ e a armazene em $chave\ promovida$

Figura 5: Algoritmo de divisão de um nó.

Quando há a divisão da raiz, a árvore aumenta sua altura.

3 Comentários finais

A forma para manter a árvore sempre balanceada é nunca criar um nó acima ou abaixo de outro. Sempre que se cria um novo nó, ele fica no mesmo nível de um já existente. Assim, ao se criar um nó folha, ele sempre estará no mesmo nível de outro nó folha e, em consequência, de *todos* os outros nós folhas.

Um nó, depois de criado, não muda sua situação de folha para interno ou *vice-versa*. Um nó criado como folha é folha até que seja removido. O mesmo acontece para um nó interno.

O crescimento da árvore é *bottom-up*, ou seja, de baixo para cima. A árvore somente cresce em altura quando sua raiz é dividida, sendo uma nova raiz criada um nível acima da anterior.

Quando ocorre divisão, há sempre uma chave que é promovida para o nível superior. Nessa situação, é possível que haja uma nova divisão no nó superior, bastando que não exista espaço para a abrigar esse chave. Esse processo pode desencadear divisões até o nível da raiz. Porém, quando uma chave promovida pode ser inserida no nó pai, o processo de divisões termina e nenhuma outra divisão ocorre nos níveis superiores.

Na inserção em árvores B, somente os nós no caminho de descida podem ser modificados. Os ramos que foram descartados ao se buscar o nó folha não são consultados em nenhuma hipótese.

Descrição: Inserção de uma nova chave na árvore B
Entrada: O ponteiro que indica a raiz da árvore e a nova chave
Saída: O ponteiro modificado, por referência, no caso da inserção na árvore vazia
Retorno: `Verdadeiro` se a inserção for bem sucedida ou `Falso` caso já exista uma ocorrência da chave na árvore

```
1  função INSIRA(raiz, nova chave)
2      se raiz é o ponteiro nulo        # árvore vazia
3          # Criação da raiz
4          Crie o nó raiz
5          Altere o tipo de raiz para FOLHA
6          Insira a nova chave como único dado de raiz
7          Ajuste os ponteiros à esquerda e à direita de nova chave para o valor nulo
8          retorne Verdadeiro
9      senão
10         # Inserção no caso da árvore não estar vazia
11         status ← BUSQUEEINSIRA(nodu atual, nova chave, houve promocao, chave promovida,
               novo nodu)
12         se houve promocao é Verdadeiro
13             # Tratamento a divisão da raiz
14             Crie um novo nó nova raiz
15             Altere o tipo de raiz para INTERNO        # nó não-folha
16             Insira a chave promovida em nova raiz
17             Ajuste o ponteiro esquerdo dessa chave para apontar para raiz
18             Ajuste o ponteiro direito dessa chave para apontar para novo nodu
19             Ajuste o ponteiro raiz para apontar para nova raiz
20         retorne status
```

Figura 6: Algoritmo de inserção: tratamento da raiz.

4 Material para leitura

Esta seção descreve material para leitura e estudo. O presente texto não é completo e o conhecimento deve ser complementado depois da aula.

4.1 Leituras complementares

Os livros de Cormen et al. (2002) e Drozdek (2010) são boas referências para leitura. Nestes materiais devem ser estudados os algoritmos de busca e inserção, com a ressalva que há diferenças entre os algoritmos dos livros e os deste texto.

Além desses livros, outros materiais podem ser consultados. Seguem algumas sugestões:

- Langsam, Augenstein e Tenenbaum (1990);
- Garcia-Molina, Ullman e Widom (2001);
- Animação da University of San Francisco (https://www.cs.usfca.edu/~galles/visualization/BTree.html).

4.2 Leituras suplementares

Para quem desejar se aprofundar no tema, podem ser lidos, nas mesmas referências anteriores, textos sobre outras ávores da família das árvores B. Um destaque pode ser dado às árvores B+ e às árvores B*.

Referências

CORMEN, T. et al. *Algoritmos: teoria e prática*. Rio de Janeiro: Campus, 2002.

DROZDEK, A. *Estrutura de dados e algoritmos em C++*. São Paulo: Cengage Learning, 2010.

GARCIA-MOLINA, H.; ULLMAN, J.; WIDOM, J. *Implementação de sistemas de banco de dados*. New Jersey: Campus, 2001.

LANGSAM, Y.; AUGENSTEIN, M.; TENENBAUM, A. *Data structures using C and C++*. New Jersey: Prentice-Hall, 1990.

Descrição: Inserção de uma nova chave na árvore B, buscando descendentemente o nó folha correto para inserção da nova chave

Entrada: O ponteiro para um $nodo$ da árvore que está no caminho de descida e a $nova\ chave$ que será inserida

Saída: O valor lógico $houve\ promocao$, passado por referência, que indica que houve uma divisão de nós no nível inferior. Caso esse valor seja `Verdadeiro` , os parâmetros (ambos por referência $chave\ promovida$ conterá a mediana da divisão das chaves e $novo\ nodu$ conterá o ponteiro para o novo nó criado pela divisão; esses dois últimos parâmetros conterão valores indeterminados se $houve\ promocao$ for `Falso`

Retorno: `Verdadeiro` se a inserção for bem sucedida ou `Falso` caso já exista uma ocorrência da chave na árvore

```
1  função BUSQUEEINSIRA(nodu atual, nova chave, houve promocao,
           chave promovida, novo nodu)
2      Verifique se a nova chave já existe nodu atual
3      se a nova chave foi localizada no nodu atual
4          # Indica que há duplicidade de chaves
5          retorne Falso
6      senão
7          se nodu atual for do tipo FOLHA
8              # Realização da inserção no folha nodu atual
9              se nodu atual não está cheio
10                 Insira a nova chave no nodu atual
11                 Ajuste o ponteiro à direita da nova chave para nulo
12                 houve promocao ← Falso
13             senão
14                 DIVIDA(nodu atual, nova chave, NIL, novo nodu, chave promovida)
15                 houve promocao ← Verdadeiro
16             retorne Verdadeiro
17         senão
18             # Procura pelo caminho para inserir
19             Determine, verificando o valor das chaves, o nodu filho correto
20             status ← BUSQUEEINSIRA(nodu filho, nova chave, houve promocao,
                       chave promovida, novo nodu)
21             # Verificação de divisão no nível inferior
22             se status e houve promocao são Verdadeiro
23                 se nodu atual não está cheio
24                     Insira a chave promovida no nodo atual
25                     Ajuste o ponteiro à direita da chave promovida para novo nodu
26                     houve promocao ← Falso
27                 senão
28                     DIVIDA(nodu atual, chave promovida, novo nodu, novo filho,
                           chave promovida)
29                     novo nodu ← novo filho      # para retornar para a chamada anterior
30             retorne status
```

Figura 7: Algoritmo de inserção: busca descendente e inserção. O valor NIL indica o ponteiro nulo.

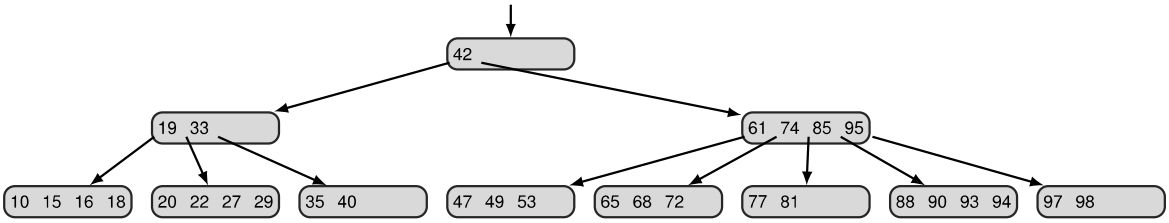

Figura 8: Exemplo de inserção da chave 20 na árvore B da Figura 2.

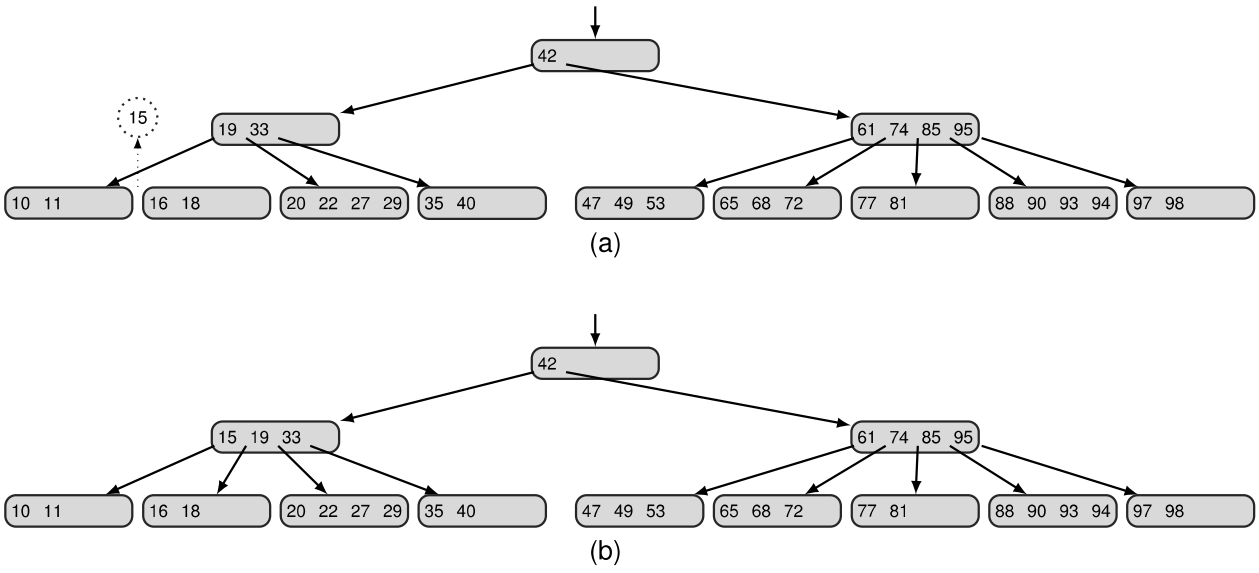

Figura 9: Exemplo de inserção da chave 11 na árvore B da Figura 8: (a) A inserção de 11 leva à divisão do nó, promovendo a mediana 15; (b) A chave promovida 15 é inserida no nível acima e o ponteiro a sua direita aponta para o novo nó criado.

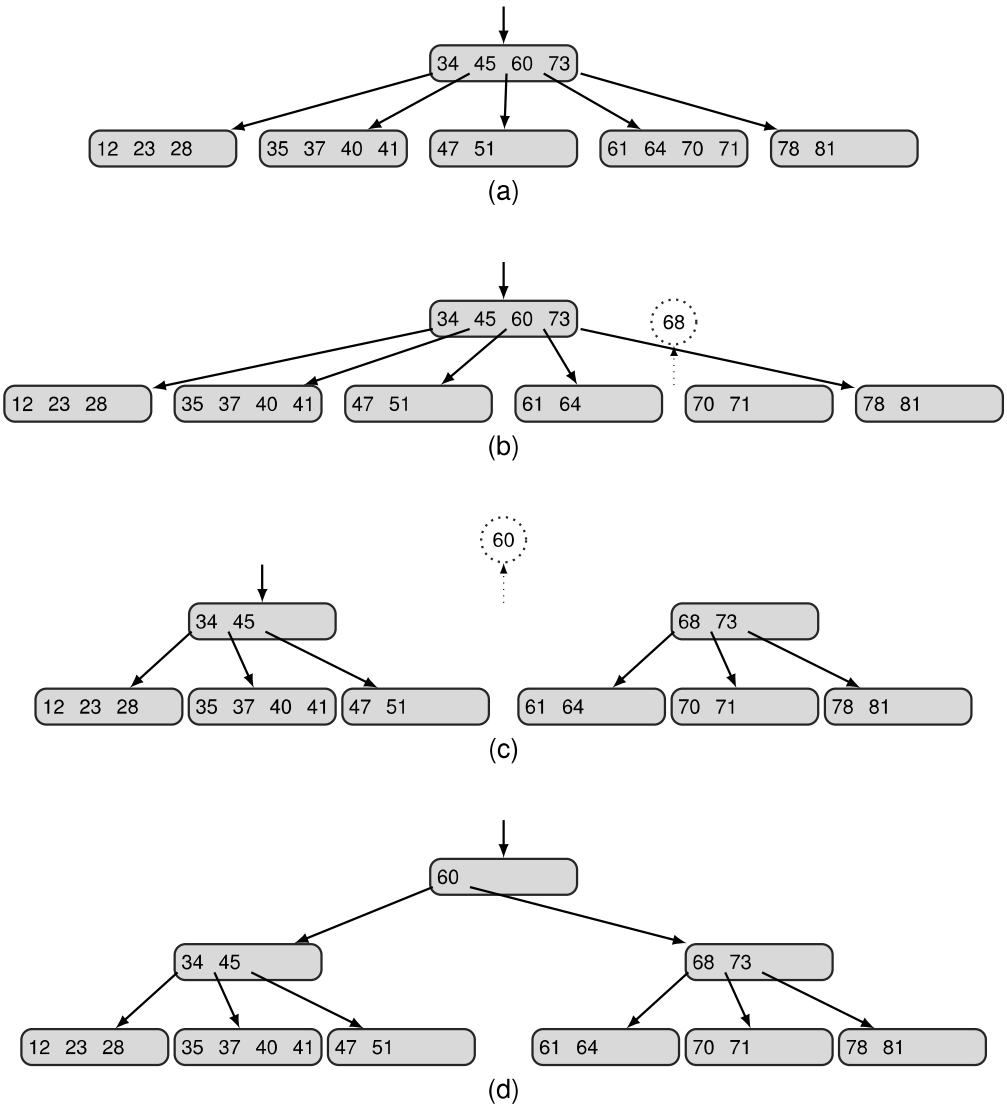

Figura 10: Exemplo de inserção em árvore B com duas divisões sucessivas: (a) Árvore B original; (b) Primeira divisão ocasionada pela inserção da chave 68; (c) Segunda divisão causada pela inserção da chave promovida do nível inferior; (d) Árvore final, com a criação de uma nova raiz.