Árvores B
Parte 2

Jander Moreira*

22 de Agosto de 2014

Conteúdo

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

Tempo de leitura estimado: 1 hora(s)

1 Introdução

As árvores B são árvores de busca em que, por princípio, procura-se reduzir o número de nós modificados para que se obtenha desempenho.

As características importantes de árvores B incluem:

- A estrutura da árvore é multicaminhos, possuindo vários filhos por nó;
- A estruturação da árvore é na forma de árvore de busca, ou seja, que permita uma busca eficiente por uma dada chave;
- A árvore é sempre balanceada, mantendo todas a folhas no mesmo nível;
- A ordem da árvore determina o número máximo e o número minimo de filhos por nó.

A operação de remoção de uma chave de uma árvore B usa procedimentos que garantem a manutenção dessas características.

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

2 Remoção

Para se realizar a remoção é preciso considerar como a estrutura da árvore é afetada, o que é visto em termos de eficiência e, depois, como satisfazer esses requisitos. Esta seção mostra uma visão geral dos requisitos da remoção (seção 2.1) e os algoritmos envolvidos na remoção, com exemplos (seção 2.2). A seção 2.3 descreve o tratamento exigido quando uma remoção causa a redução de chaves no nó abaixo do mímino.

2.1 Visão geral

A remoção de árvores B envolve, além de manter a árvore como árvore de busca, o cuidado com algumas características:

- Balanceamento;
- Número mínimo de chaves por nó;
- Redução do número de nós afetados pelas operações sobre a árvore.

O balanceamento é considerado nas remoções evitando-se que um nó seja simplesmente removido. Se, na inserção, nós são criados fazendo a divisão de um dado nó, a remoção trata os nós fazendo a fusão de nós, quando apropriado. A operação é, praticamente, a inversa da inserção.

Considera-se que há um número mínimo de chaves por nó, usualmente definido por $\lfloor \frac{n-1}{2} \rfloor$, sendo n a ordem da árvore. Se não houver violação desse critério, a estrutura da árvore não é modificada. Ainda que um nó fique com utilização abaixo do mínimo, esse problema é tratado de forma que se adie a modificação do número de nós, o que se dá pelo remanejamento de chaves entre os nós.

A eficiência das árvores B reside no fato de se tentar minimizar alterar a estrutura da árvore. Assim, como mencionado nos dois parágrafos anteriores, quanto menor o número de nós modificados na operação de remoção, melhor para a árvore.

2.2 Algoritmos e exemplos

O processo de remoção pode ser separado em dois casos: remoções de chaves que estão em nós folhas e de chaves em nós internos.

Quando uma chave de um nó folha é removida, a estrutura lógica da árvore não é modificada, visto que a chave removida não é critério para descida na árvore. No caso de chaves de um nó interno, toda chave é essencial, pois é critério para decidir se uma busca ou outra operação deve ir para um ramo ou para outro.

A estratégia para remoções de chaves de nós internos é não remover a chave do nó, mas substituí-la por outra. Essa outra chave deve servir como critério de descida equivalente àquela que foi eliminada. Para isso, uma alternativa é substituir a chave removida buscando-se a maior chave existente no ramo imediatamente à esquerda da primeira.

Na árvore ilustrada na Figura 1, por exemplo, a remoção da chave `42` que está na raiz implicaria em substituí-la pela `41`. A escolha é feita pela descida ao filho imediatamente à esquerda da raiz, para o nó com chaves `19` e `33` e, a partir daí, descer sempre pelo ponteiro mais à esquerda até encontrar um nó folha; a partir desse nó, a maior chave existente nele é a escolhida para substituição[1].

Outros exemplos: para remover a chave `74`, escolhe-se a `72` para substituí-la; para a `94`, escolhe-se a `94`; e para a `61`, `53` é a opção. Não são apresentadas figuras para esses exemplos.

Essa forma de tratamento para os nós internos tem como efeito final que o nó realmente afetado por uma remoção é sempre um nó folha, de modo que o tratamento é reduzido a um único caso.

Para o caso da remoção de uma chave de um nó folha, o procedimento inicial é simples: basta remover a chave do nó. Pode haver, porém, uma implicação posterior dessa remoção, que deixar o nó com número de chaves inferior ao mínimo.

O algoritmo de remoção é dividido em duas partes: a primeira faz o tratamento do nó raiz e a segunda realiza a busca na árvore e a remoção em si. A Figura 3 mostra a primeira parte (função REMOVA), enquanto a Figura 4 apresenta a segunda (função BUSQUEEREMOVA.

[1] A opção simétrica também pode ser adotada, buscando-se a menor chave a partir do ramo imediatamente à direita.

O algoritmo apresentado para REMOVA verifica se a árvore existe ou não. Caso a árvore esteja vazia, termina sumariamente; se existir pelo menos um nó na árvore, aciona a busca recursiva BUSQUEEREMOVA e, a seguir, verifica se a raiz ficou sem chaves (o que acontece se houver a fusão de nós no nível imediatamente inferior). Quando a raiz fica vazia, ela não é mais necessária e a árvore tem sua altura reduzida em uma unidade.

De forma mais genérica, o algoritmo BUSQUEEREMOVA (Figura 4) trata a remoção nos demais nós. Sua primeira tarefa é localizar a chave e, para isso, usa a lógica do algoritmo de busca BUSQUE[2]. Caso a chave não seja localizada, o valor `Falso` é retornado; caso contrário a remoção é realizada.

Quando a chave é localizada em um nó interno, o tratamento feito é substituir essa chave por outra que tenha a mesma função estrutural na árvore. A função LOCALIZESUBSTITUTA faz a localização dessa chave, a qual é sobreposta àquela que se quer remover. Logo em seguida é acionada a remoção da cópia de $chave\ substituta$ da folha em que foi encontrada (linha 33). Essa chamada reduz o problema à situação básica, na qual a remoção ocorre na folha.

O trecho do algoritmo que trata a remoção de uma chave de uma folha se inicia na linha 21. A única função deste segmento de código é eliminar a chave do nó e indicar se houve ou não violação do critério de ocupação mínima.

Em todas as chamadas onde se solicita recursivamente uma remoção, os próximos comandos correspondem a verificações de violação do critério de número mínimo de chaves. Isso ocorre nas linhas 34 e 11.

Árvores B tentam, dentro de suas premissas,alterar o mínimo possível a sua estrutura. Isso significa que, nas remoções, a alteração do número de nós é a última opção.

2.3 Tratamento dos nós com ocupação abaixo do mínimo

Um dos princípios da estrutura da árvore é que não haja nenhum nó com menos chaves que o mínimo determinado pela ordem. Em geral, esse valor é a metade da capacidade máxima, ou seja, $\lfloor \frac{n-1}{2} \rfloor$.

[2] Apresentado em outro texto.

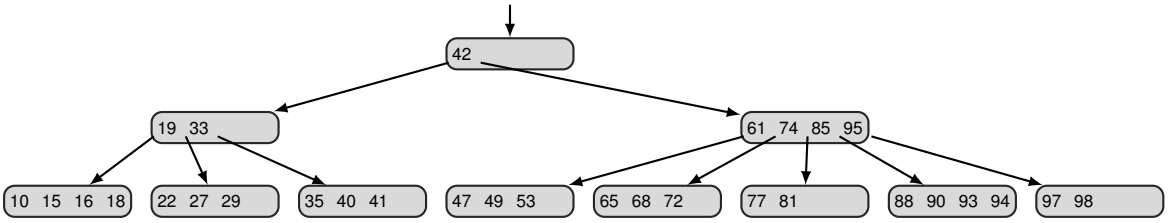

Figura 1: Exemplo de uma árvore B de ordem 5.

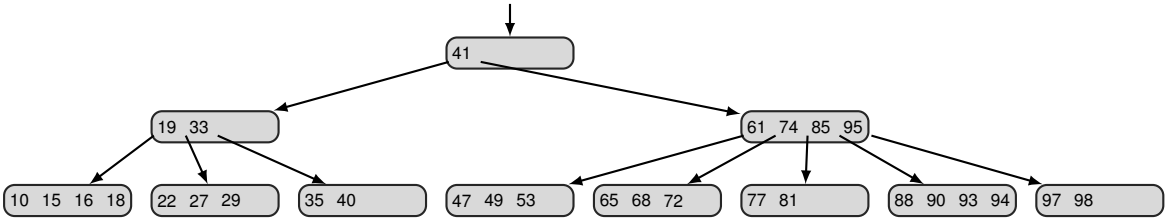

Figura 2: Árvore B da Figura 1 depois da remoção da chave `42`, substituída pela chave `41` encontrada na folha. Não houve, neste caso, nenhuma violação do número mínimo de chaves por nó.

Assim, definem-se duas estratégias para tratar esse problema: o empréstimo de uma chave e a fusão de dois nós em um só.

O empréstimo é, na prática, uma "rotação" de chaves que envolve o nó pai, o nó com problemas e um nó irmão[3]. Consiste na verificação se é possível transferir uma chave de um nó irmão para ele, por intermédio do nó pai. Nesta situação, apenas os nós irmãos imediatamente adjacentes são consultados, se existirem. Não há, portanto, verificação para mais que dois irmãos.

Um nó irmão pode fazer o empréstimo sempre que possuir pelo menos uma chave a mais que a quantidade mínima, preservando sua própria consistência. Se o empréstimo for do nó irmão à esquerda, a chave cedida para movimentação é sua maior chave; se for do nó irmão à direita, este cede sua menor chave. Nas duas situações, a chave do nó pai que separa os dois nós filhos envolvidos é usada para a troca.

O exemplo da Figura 5 mostra um empréstimo do irmão à esqueda. No caso, primeiro a chave 21 é copiada para o nó que ficou subutilizado e, em seguida, a maior chave (20) do irmão à esquerda é copiada para o nó pai.

Na Figura 6 mostra um empréstimo do irmão à esqueda. No caso, primeiro a chave 21 é copiada para o nó que ficou subutilizado e, em seguida, a maior chave (20) do irmão à esquerda é copiada para o nó pai.

Em geral, o empréstimo é tentado primeiro do nó irmão à esquerda. Se este não existir ou não tiver chaves suficientes, é tentado o empréstimo do nó irmão à direita.

Sempre que for possível fazer o empréstimo, o problema de subutilização é resolvido e não há necessidade de se alterar a estrutura da árvore.

Há situações, porém, que o empréstimo não é possível. A solução é, então, fazer a fusão de dois nós em um só, realizando um processo equivalente ao inverso da divisão da inserção. Quando há a fusão, o conteúdo dos dois nós é passado para apenas um deles e o outro nó, ficando vazio, é liberado. Como um dos nós deixa de existir, a chave do nó pai que os separava deixa de ter utilidade naquele nó e ela é transferida para o nó filho.

A fusão de um nó é feita com seu irmão imediatamente à esquerda. Quando o nó com problemas já é o filho mais à esquerda, então a fusão feita é a do irmão à direita com o nó faltoso.

A fusão é exemplificada na Figura 7, na qual a remoção da chave 55 causa uma subutilização na folha que a continha. Os empréstimos tanto do irmão à esquerda quanto do à direita não são possíveis, uma vez que ambos já estão com no limite

[3] Um nó é considerado *irmão* de outro se ambos estão como filhos de um mesmo nó.

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

```
1  função REMOVA(raiz, chave)
2      se o raiz for um ponteiro nulo
3          # Caso da árvore vazia
4          retorne Falso
5      senão
6          # Busca recursiva na árvore (ignorando o retorno em houve violacao)
7          status ← BUSQUEEREMOVA(raiz, chave, chave removida, houve violacao)
8          se a raiz ficou sem nenhuma chave
9              Faça nodu apontar para raiz
10             Altere raiz para apontar para o primeiro ponteiro do próprio nó raiz
11             Libere o nó nodu
12         retorne status
```

Figura 3: Algoritmo de remoção da árvore B, para o tratamento da raiz.

mínimo de chaves por nó. A fusão envolve o irmão à esquerda (que contém as chaves 40 e 48). Todas as chaves são mantidas no nó à esquerda, incluindo-se a chave no nó pai que servia como separação entre os dois nós. Na árvore final, o nó vazio é liberado.

O algoritmo TRATEVIOLAÇÃO é o que trata o problema de violação da quantidade mínima de chaves por nó (Figura 8). Nesse algoritmo, o parâmetro $nodu\ filho$ é o ponteiro para o nó que ficou com menos chaves que podia e $nodu\ pai$ é o nó imediatamente acima dele. A primeira tentativa é um empréstimo do nó à esquerda, que deve existir e possuir chaves suficientes para emprestar. Se não houver sucesso, então é tentado o empréstimo do nó irmão da direita (o qual também tem que existir e ter chaves suficientes). Somente no caso de nenhum empréstimo ser possível, há o acionamento da fusão dos nós pelo procedimento FAÇAFUSÃO.

O algoritmo para FAÇAFUSÃO é apresentado na Figura 9 o qual define quais os nós envolvidos no processo de fusão para, em seguida, transferir todas as chaves de um para outro, incluindo a chave do nó pai que os separava.

3 Comentários finais

Uma árvore B possui implementação que tenta minimizar alterações em sua estrutura como um todo. Assim, a prioridade do empréstimo sobre a fusão se torna claro. Ao se fazer um empréstimo, o problema de subutilização de um nó é resolvido e nenhum outro nó da árvore é modificao (ou sequer verificado sobre problemas).

Quando uma fusão ocorre, uma chave do nó pai é transferida ("rebaixada") para um nó filho, o que reduz de uma unidade seu número de chaves. Essa redução pode violar o critério de quantidade miníma de chaves que, por sua vez, pode gerar nova necessidade de tratamento.

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: Busca e remoção de uma chave da árvore B
Entrada: O ponteiro para um nó da árvore B (inicialmente a raiz) e o valor da chave a ser removida
Saída:
Retorno: `Verdadeiro` caso a a chave seja localizada; `Falso` caso contrário

```
1  função BUSQUEEREMOVA(nodu atual, chave, houve violacao)
2      Busque a chave em nodu atual
3      se a chave não foi localizada
4          se nodu atual é um nó do tipo FOLHA
5              # A chave não existe na árvore
6              retorne Falso
7          senão
8              # Busca recursiva
9              Determine o nó filho para descida na árvore
10             status ←BUSQUEEREMOVA(filho, chave, houve violacao)
11             se houve violacao for Verdadeiro
12                 TRATAVIOLAÇÃO(nodu atual, filho)
13             se nodu atual estiver abaixo do mínimo de chaves        # Houve nova violação?
14                 Atribua a houve violacao o valor Verdadeiro
15             senão
16                 Atribua a houve violacao o valor Falso
17             retorne status
18     senão
19         Atribua a chave o valor localizado        # retorno por referência
20         se nodu atual é um nó do tipo FOLHA
21             # Remoção simples do nó folha
22             Remova chave do nodu atual
23             se nodu atual estiver abaixo do mínimo de chaves        # Houve violação?
24                 Atribua a houve violacao o valor Verdadeiro
25             senão
26                 Atribua a houve violacao o valor Falso
27         senão
28             # Substituição de chave para nó interno
29             Defina filho como o ponteiro à esquerda da chave a ser removida
30             chave substituta ← LOCALIZESUBSTITUTA(filho)
31             Subsititua, em nodu atual, chave por chave substituta

32             # Remoção da chave copiada (o retorno status é ignorado – sempre Verdadeiro )
33             status ← BUSQUEEREMOVA(filho, chave substituta, houve violacao)
34             se houve violacao tiver valor Verdadeiro
35                 TRATAVIOLAÇÃO(nodu atual, filho)
36             se nodu atual estiver abaixo do mínimo de chaves        # Houve nova violação?
37                 Atribua a houve violacao o valor Verdadeiro
38             senão
39                 Atribua a houve violacao o valor Falso
40         retorne Verdadeiro
```

Figura 4: Algoritmo de remoção da árvore B, para o tratamento da raiz.

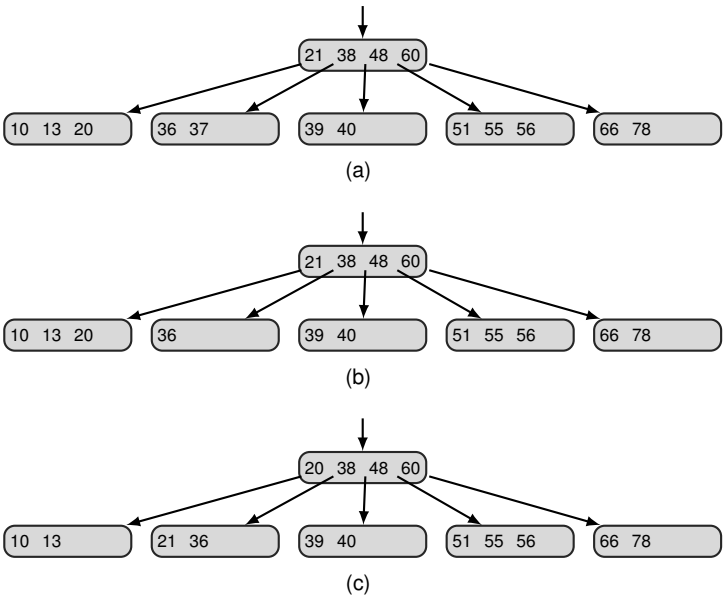

Figura 5: Exemplo de um empréstimo, considerando-se uma árvore de ordem 5: (a) Árvore original; (b) Resultado depois da remoção da chave 37, deixando no nó apenas a chave 36; (c) Resultado do empréstimo, fazendo a rotação das chaves 21 (do nó pai para o nó faltoso) e 20 (do irmão à esquerda para o pai).

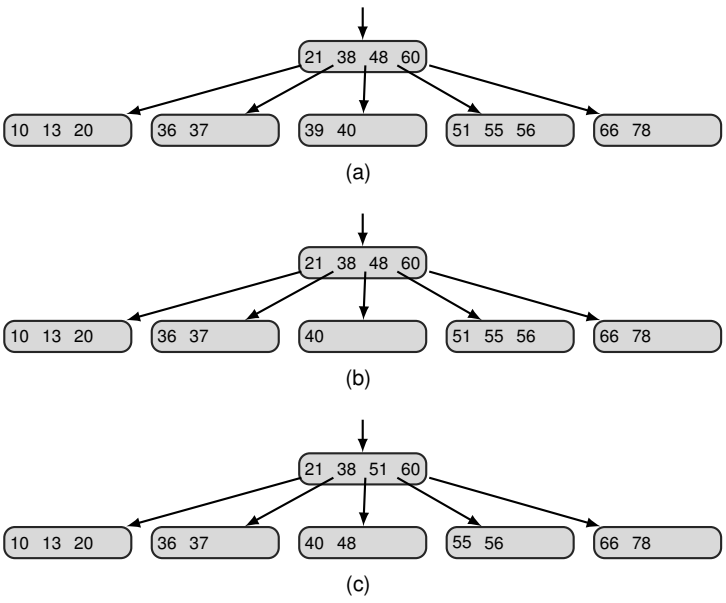

Figura 6: Exemplo de um empréstimo, considerando-se uma árvore de ordem 5: (a) Árvore original; (b) Resultado depois da remoção da chave 39, deixando no nó apenas a chave 40; (c) Resultado do empréstimo, fazendo a rotação das chaves 48 (do nó pai para o nó faltoso) e 51 (do irmão à direita para o pai).

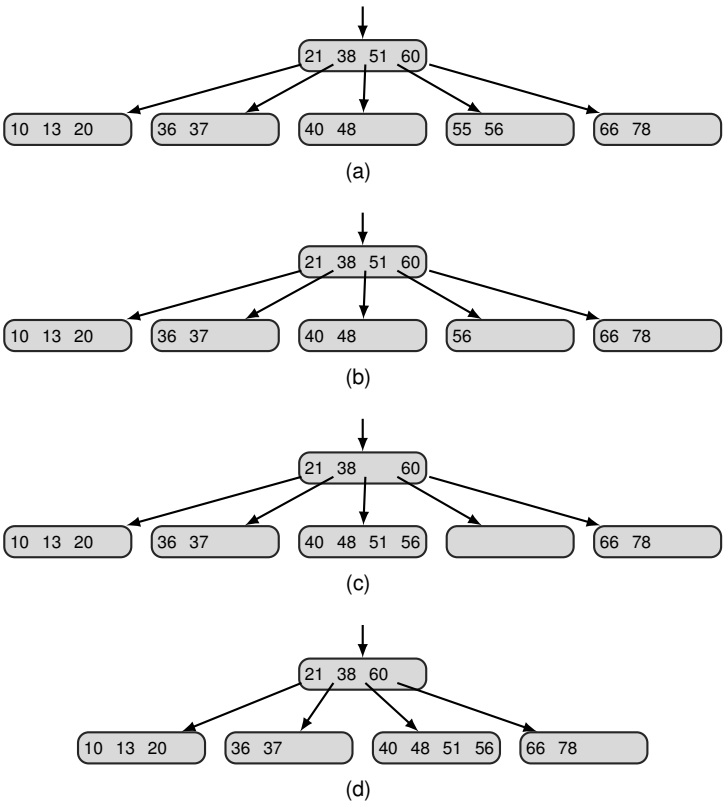

Figura 7: Exemplo de um fusão, considerando-se uma árvore de ordem 5: (a) Árvore original; (b) Resultado depois da remoção da chave 55, deixando no nó apenas a chave 56; (c) Resultado da fusão dos nós pela transferência das seguintes chaves para o nó irmão à esquerda: 51 (vinda do nó pai) e 56 (vinda do irmão). (d) Resultado final, depois da liberação do nó que ficou vazio.

Descrição: Tratamento da violação do número mínimo de chaves por nó, tentando inicialmente empréstimos e, então, a fusão.
Entrada: O ponteiro para o nó $nodu\ pai$ e o ponteiro para o nó $nodu\ filho$ no qual ocorreu a subutilização.

```
1  procedimento TRATEVIOLAÇÃO(nodu pai, nodu filho)
2      se existe um nó nodu esquerda imediatamente à esquerda de nodu filho e esse nó
3              possui mais de ⌊(n−1)/2⌋ chaves
4          # Empréstimo de nodu esquerda
5          Transfira a chave à esquerda do ponteiro nodu filho de nodu pai para nodu filho
6          Transfira a maior chave de nodu esquerda para nodu pai, na posi-
               ção à esquerda do ponteiro nodu filho
7      senão
8          se existe um nó nodu direita imediatamente à direita de nodu filho e esse nó
9                  possui mais de ⌊(n−1)/2⌋ chaves
10             # Empréstimo de nodu direita
11             Transfira a chave à direita do ponteiro nodu filho de nodu pai para nodu filho
12             Transfira a menor chave de nodu direita para nodu pai, na posição
                   à direita do ponteiro nodu filho
13         senão
14             # Fusão de nós
15             FAÇAFUSÃO(nodu pai, nodu filho)
```

Figura 8: Algoritmo de tratamento de subutilização de um nó.

Descrição: Fusão de dois nós da árvore.
Entrada: O ponteiro para o nó $nodu\ pai$ e o ponteiro para o nó $nodu\ filho$ no qual ocorreu a subutilização.

```
1  procedimento FAÇAFUSÃO(nodu pai, nodu filho)
2      # Definição de quais nós estão envolvidos na fusão
3      se existe um nó nodu esquerda imediatamente à esquerda de nodu filho
4          Defina nodu como o nó à esquerda de nodu filho
5          Defina nodu vazio igual a nodu filho
6      senão
7          Defina nodu igual a nodu filho
8          Defina nodu vazio como o nó à direita de nodu filho

9      # Fusão
10     Defina chave como a chave de nodu pai entre os ponteiros nodu e nodu vazio
11     Copie a chave para nodu
12     Copie o primeiro ponteiro de nodu vazio como ponteiro à direita de chave em nodu
13     Copie todas as chaves de nodu vazio para nodu, com respectivos ponteiros à direita
14     Libere o nó nodu vazio
15     Remova chave de nodu pai
```

Figura 9: Algoritmo fusão de dois nós da árvore.