Algoritmos e Estruturas de Dados II

Prof. Debora Medeiros

Árvores B – Parte IV

Algoritmos de Split e Procedimento inicial

Eliminação, Redistribuição & Concatenação

Adaptado dos Originais de:

Ricardo J. G. B. Campello
Thiago A. S. Pardo
Cristina D. A. Ciferri
Leandro C. Cintra
Maria Cristina F. de Oliveira

Algoritmo: Split

- Parâmetros
  - chave_a_inserir, filho_dir_chave_a_inserir
    - dados a serem inseridos
  - Pagina
    - página atual
  - chave_promovida, filho_dir_chave_promovida
    - parâmetros de retorno: novos dados promovidos para o nível superior
  - Pagina_nova
    - página criada

Algoritmo: Split

- Tratamento do overflow causado pela inserção de uma chave
  - cria uma nova página (i.e., Pagina_nova)
  - distribui as chaves o mais uniformemente possível entre Pagina e Pagina_nova
  - determina qual chave e qual RRN serão promovidos
    - chave_promovida e filho_dir_chave_promovida

Algoritmo: Split

SPLIT(*chave_a_inserir*, *filho_dir_chave_a_inserir*, *Pagina*, *chave_promovida*, *filho_dir_chave_promovida*, *Pagina_nova*)
copie todas as chaves e ponteiros de *Pagina* para uma página temporária com um espaço extra para uma nova chave e um novo ponteiro
inseria *chave_a_inserir* e *filho_dir_chave_a_inserir* no lugar apropriado na página temporária
crie *Pagina_nova*
chave_promovida ← chave do meio da página temporária
filho_dir_chave_promovida ← RRN de *Pagina_nova*
copie as chaves e os ponteiros que precedem *chave_promovida* na página temporária para *Pagina*
copie as chaves e os ponteiros que seguem *chave_promovida* na página temporária para *Pagina_nova*

Algoritmo: Split

- Observações
  - somente uma chave é promovida e sai da página de trabalho atual
  - todos os RRN dos nós filhos são transferidos de volta para Pagina e Nova_pagina
  - o RRN promovido é o de Nova_pagina
    - Nova_pagina é a descendente direita da chave promovida
- Note que a função *split move os dados!*

Procedimento inicial

- Rotina inicializadora e de tratamento da raiz
  - abre ou cria o arquivo de índice (árvore-B)
  - identifica ou cria a página da raiz
  - lê chaves para serem armazenadas na árvore-B e chama função de inserção de forma apropriada
  - cria uma nova raiz quando a função de inserção particionar a raiz atual

Algoritmo: procedimento inicial

PROCEDIMENTO INICIAL
se arquivo com árvore-B existe
  abra arquivo
senão crie arquivo e coloque a primeira chave na raiz
recupere *RRN* da página raiz
leia uma chave
enquanto há chaves **faça**
  se FUNÇÃO-INSERÇÃO(*raiz*, *chave*, *chave_promovida*, *filho_dir_chave_promovida*) = HÁ_PROMOÇÃO
    crie nova página raiz com *chave* = *chave_promovida*, *filho_esquerdo* = *raiz*, *filho_direito* = *filho_dir_chave_promovida*
    raiz ← RRN da nova página raiz
  leia próxima chave
escreva no arquivo o RRN de raiz
feche arquivo

Propriedades

Definição

- A **ordem** de uma árvore B é dada pelo número máximo de descendentes que uma página/nó, pode possuir
- Em uma árvore B de ordem **m**, o número máximo de chaves em uma página é **m – 1**
- **Exemplo:**
  - Uma árvore B de ordem 8 tem, no máximo, 8 descendentes e 7 chaves por página

Propriedade (No. Mín. de Chaves)

- Quando uma página é sub-dividida na inserção, as chaves são distribuídas "igualmente" entre as páginas resultantes:
  - ...

Propriedade (No. Mín. de Chaves)

- Quando uma página é sub-dividida na inserção, as chaves são distribuídas "igualmente" entre as páginas resultantes:
  - Deste modo, o número mínimo de chaves em uma página/nó é dado por $\lceil m/2 \rceil - 1$ (exceto para a raiz)
- **Exemplos:**
  - árvore B de ordem 8: armazena no máximo 7 chaves por página e no mínimo 3 chaves por página
  - árvore B de ordem 7: armazena no máximo 6 chaves por página e no mínimo 3 chaves por página

Propriedades Gerais

- Para uma árvore B de ordem **m**:
  - Cada página tem:
    - no máximo, **m** descendentes
    - no mínimo $\lceil \mathbf{m/2} \rceil$ descendentes (exceto a raiz e as folhas)
  - A raiz tem, no mínimo, dois descendentes
    - a menos que seja uma folha
  - Todas as folhas estão no mesmo nível
  - Uma página não folha com **k** descendentes contém **k – 1** chaves
  - Uma página contém no mínimo $\lceil \mathbf{m/2} \rceil - \mathbf{1}$ chaves (exceto a raiz) e, no máximo, **m – 1** chaves

Altura de Pior Caso

- Qual a altura máxima que uma árvore com **N** chaves e ordem **m** pode atingir?
- Pior caso ocorre quando cada página tem apenas o número mínimo de descendentes, e a árvore possui, portanto, altura máxima e largura mínima

Altura de Pior Caso

- O número mínimo de descendentes para a raiz é 2
- Cada uma das 2 páginas descendentes da raiz possui no mínimo $\lceil \mathbf{m/2} \rceil$ descendentes
  - Logo, o 2º nível possui no mínimo $2^* \lceil \mathbf{m/2} \rceil$ descendentes
- Cada uma das $2^* \lceil \mathbf{m/2} \rceil$ páginas descendentes do 2º nível possui no mínimo $\lceil \mathbf{m/2} \rceil$ descendentes
  - Logo, o 3º nível possui no mínimo $2^* \lceil \mathbf{m/2} \rceil^2$ descendentes
- Em geral, para um nível **d** da árvore, o número mínimo de descendentes é dado por $\mathbf{2^* \lceil m/2 \rceil^{(d-1)}}$

Altura de Pior Caso

- Em Resumo:
  - Altura de pior caso ocorre quando cada página tem apenas o no. mínimo de descendentes
  - O no. mínimo de descendentes para um nível **d** da árvore de ordem **m** é dado por $\mathbf{2^* \lceil m/2 \rceil^{(d-1)}}$
  - Qual o nível mais profundo **d** de pior caso para uma árvore B com **N** chaves e ordem **m** ?

Altura de Pior Caso

- Qual o nível mais profundo **d** de pior caso para uma árvore B com **N** chaves e ordem **m** ?
  - O no. de descendentes que existiriam abaixo do nível **d** mais profundo (nível das folhas) se a árvore possuísse mais um nível é $\leq \mathbf{N + 1}$, já que a árvore comporta **N** chaves
  - Mas, no pior caso, sabemos que o no. de descendentes em um dado nível **d** da árvore é $\mathbf{2^* \lceil m/2 \rceil^{(d-1)}}$
  - Logo, no pior caso, tem-se $\mathbf{2^* \lceil m/2 \rceil^{(d-1)}} \leq \mathbf{N} + 1$ para o nível **d** mais profundo, o que resulta:

$$\mathbf{d} \leq 1 + \log_{\lceil \mathbf{m}/2 \rceil} [\, (\mathbf{N}+1)/2 \,]$$

Altura de Pior Caso

- Exemplo (**m** = 512 e **N** = 1.000.000):
  - $\mathbf{d} \leq 1 + \log_{256} (500.000{,}5) = 3{,}37$
  - Logo, a árvore terá no máximo 3 níveis
  - No pior caso 3 acessos serão necessários para localizar uma chave dentre 1.000.000

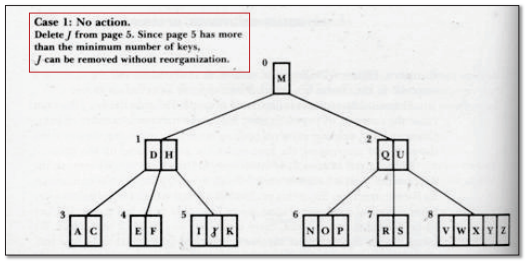
Eliminação de Chaves

- O ***split*** garante a manutenção das propriedades da árvore B durante a inserção
- Essas propriedades precisam ser mantidas, também, durante a eliminação de chaves

Eliminação: Caso 1

- **Caso 1:** eliminação de uma chave em uma página folha, sendo que o número mínimo de chaves na página é respeitado
- **Solução:** chave é retirada e os registros internos à página são reorganizados

Eliminação: Caso 1



Remover J (árvore com **m** = 6)

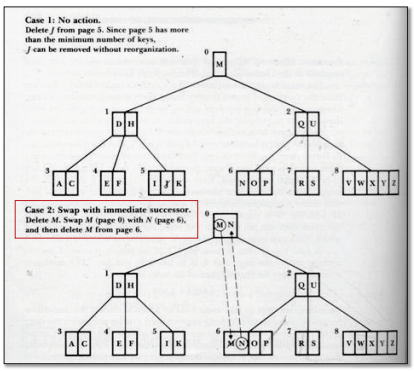
Eliminação: Caso 2

- **Caso 2:** eliminação de uma chave que não está em um nó folha
  - ...

Eliminação: Caso 2

- **Caso 2:** eliminação de uma chave que não está em um nó folha
- **Solução:** sempre eliminamos de páginas folha
  - para tanto, troca-se a chave com sua sucessora imediata (ou predecessora imediata), que está numa folha. A seguir, elimina-se a chave da folha (Caso 1)
    - sucessora imediata: 1ª chave da folha descendente mais à esquerda da página/nó descendente direito
    - predecessora imediata: ... ?

Eliminação: Caso 2



Remover M

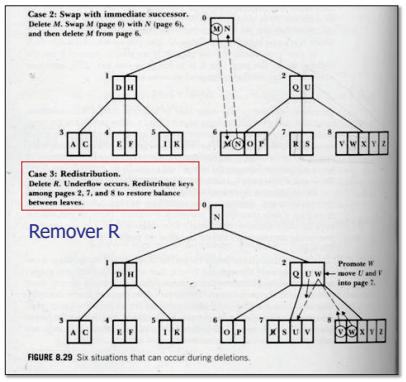
Eliminação: Caso 3

- **Caso 3:** eliminação causa *underflow* na página
  - no. mín. de $\lceil \mathbf{m/2} \rceil - \mathbf{1}$ chaves em pág. não raiz violado
  - ...

Eliminação: Caso 3

- **Caso 3:** eliminação causa *underflow* na página
  - no. mín. de $\lceil \mathbf{m/2} \rceil - \mathbf{1}$ chaves em pág. não raiz violado
- **Solução: Redistribuição**
  - procura-se uma página irmã direta (mesmo nó pai e chave separadora comum) que contenha mais chaves do que o mínimo:
    - se existir, redistribui-se as chaves entre essas páginas
    - senão, vide Caso 4...

Eliminação: Caso 3



Notas:

1. Redistribuição pode provocar uma alteração na chave separadora, que está no nó pai, mas não se propaga !
2. Redistribuição só pode ser aplicada para solução de underflows em páginas folha

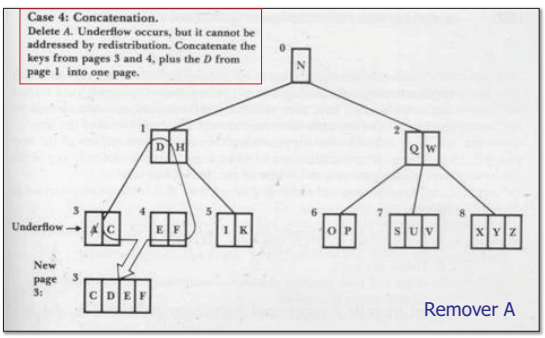
Eliminação: Caso 4

- **Caso 4:**
  - ocorre underflow e redistribuição não pode ser aplicada
    - implicaria underflow em qualquer das páginas irmãs diretas
  - ...

Eliminação: Caso 4

- **Caso 4:**
  - ocorre underflow e redistribuição não pode ser aplicada
    - implicaria underflow em qualquer das páginas irmãs diretas
- **Solução: Concatenação**
  - combina-se o conteúdo da página com o de uma irmã direta e adiciona-se a chave separadora da página pai para formar uma única página
  - concatenação é o inverso do processo de split
    - rebaixamento de chave da página pai ao invés de promoção
    - como conseqüência, pode ocorrer underflow da página pai

Eliminação: Caso 4



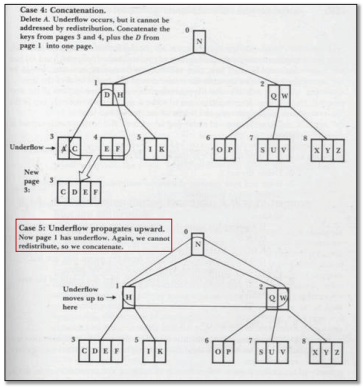
Eliminação: Caso 5

- **Caso 5:** underflow da página pai
- ...

Eliminação: Caso 5

- **Caso 5:** underflow da página pai
- **Solução:**
  - utiliza-se concatenação novamente

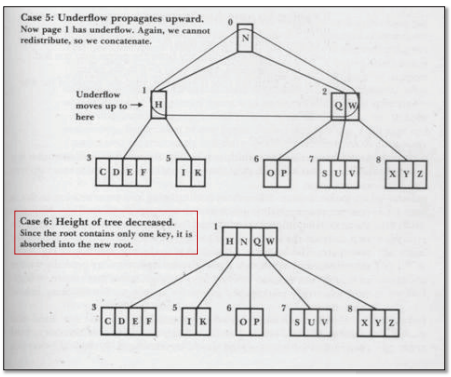
Eliminação: Caso 5



Eliminação: Caso 6

- **Caso 6:** diminuição da altura da árvore
  - ocorre quando o nó raiz tem uma única chave e aplica-se a concatenação nos seus nós filhos

Eliminação: Caso 6



Eliminação (Resumo)

1. Se a chave não estiver numa folha, troque-a com sua sucessora

2. Elimine a chave da folha

3. Se a página continuar com o número mínimo de chaves, **FIM**

4. Senão (underflow):

4.1. Se uma das páginas irmãs diretas (à esquerda ou direita) tiver mais que o número mínimo de chaves, aplique redistribuição e **FIM**

4.2. Senão:

4.2.1. Concatene a pág. com uma das irmãs e a chave separadora do nó pai

4.2.2. Se nó pai for raiz e sua última chave foi rebaixada, elimine a raiz e **FIM**

4.2.3. Senão, se nó pai continuar com o número mínimo de chaves, **FIM**

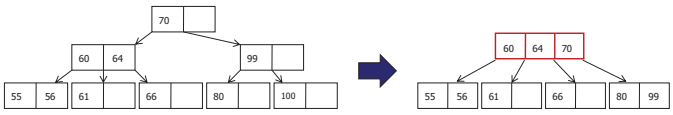
4.2.4. Senão (underflow no pai), volte ao passo 4.2.1 para o nó pai

Eliminação (Nota 1)

- Embora esta hipótese não esteja contemplada no algoritmo de eliminação anterior, a operação de concatenação pode não apenas causar um underflow na página pai, mas pode também causar um overflow na própria página concatenada
- Basta que a página irmã já contenha m – 1 chaves
- Exemplo:
  - Árvore com **m** = 3 e chaves 55, 60, 61, 56, 70, 80, 64, 99, 100 e 66 inseridas nesta ordem
  - Remover chave 100...

Eliminação (Nota 1 – cont.)

- Exemplo (**m** = 3 e chaves 55, 60, 61, 56, 70, 80, 64, 99, 100 e 66)
  - Remover chave 100:
    - rebaixamento de 99 causa **underflow**
    - rebaixamento de 70 para corrigir underflow (concatenação) causa **overflow** !



  - Solução com split (no quadro...) !

Eliminação (Nota 2)

- Na **redistribuição**, não existe regra obrigatória:
  - Estritamente, é necessário mover apenas 1 chave para a página com underflow para restabelecer as propriedades da árvore-B
  - Estratégia usual, no entanto, é redistribuir as chaves de forma equilibrada entre as páginas:
    - "Balanceamento" dos espaços disponíveis

Desempenho de Árvores B

- Qual a complexidade computacional de pior caso para as operações de busca, inserção e remoção de chaves?
  - Sabemos que, no pior caso, a altura da árvore é dada pelo maior inteiro **d** tal que: $\mathbf{d} \leq 1 + \log_{\lceil \mathbf{m}/2 \rceil} [\ (\mathbf{N}+1)/2\ ]$
  - ...

Desempenho de Árvores B

- Qual a complexidade computacional de pior caso para as operações de busca, inserção e remoção de chaves?
  - Sabemos que, no pior caso, a altura da árvore é dada pelo maior inteiro **d** tal que: $\mathbf{d} \leq 1 + \log_{\lceil \mathbf{m}/2 \rceil} [\ (\mathbf{N}+1)/2\ ]$
  - Ou seja, a altura é $O(\log_{\lceil m/2 \rceil} N)$
  - Logo, no pior caso, uma **busca** requer $O(\log_{\lceil m/2 \rceil} N)$ acessos

Desempenho de Árvores B

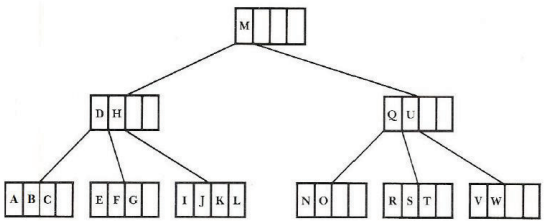
- Qual a complexidade computacional de pior caso para as operações de busca, inserção e remoção de chaves?
  - Toda inserção demanda uma busca ($O(\log_{\lceil m/2 \rceil} N)$ acessos)
  - Além disso, pode demandar operações de split
    - Cada split opera sobre um número fixo de páginas
      - Logo, cada split demanda $O(1)$ acessos
    - No pior caso, overflows se propagarão até a raiz
      - Nesse caso, teremos $O(\log_{\lceil m/2 \rceil} N)$ splits com $O(1)$ acessos cada
  - Logo, no pior caso, uma **inserção** requer $O(\log_{\lceil m/2 \rceil} N)$ acessos

Desempenho de Árvores B

- Qual a complexidade computacional de pior caso para as operações de busca, inserção e remoção de chaves?
  - Toda remoção demanda uma busca ($O(\log_{\lceil m/2 \rceil} N)$ acessos)
  - Além disso, pode demandar operações de concatenação
    - Cada concatenação opera sobre um número fixo de páginas
      - Logo, cada concatenação demanda $O(1)$ acessos
    - No pior caso, underflows se propagarão até a raiz
      - Nesse caso, teremos $O(\log_{\lceil m/2 \rceil} N)$ concatenações com $O(1)$ acessos cada
  - Logo, no pior caso, uma **remoção** requer $O(\log_{\lceil m/2 \rceil} N)$ acessos

Exercício

- Remova as chaves A, B, Q e R da árvore-B de ordem 5 abaixo



Bibliografia

- **M. J. Folk and B. Zoellick, *File Structures: A Conceptual Toolkit*, Addison Wesley, 1987.**