Algoritmos e Estruturas de Dados II - SCE-203

Arquivos: Árvores-B

Gustavo Batista

Histórico

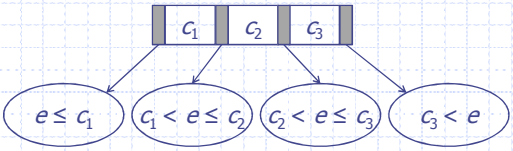
- Ao final dos anos 60 existia uma competição para encontrar uma estrutura que permitisse acesso rápido a um arquivo.
- Em 1972, Bayer e McCreigh publicaram o artigo "Organization and Maintence of Large Ordered Indexes", propondo as árvores-B.
- Em 1979, árvores-B já haviam se tornado o padrão de organização de índices em sistemas de banco de dados.

Histórico

- Árvores-B permite a recuperação em tempo proporcional a $O(\log_m n)$, na qual m é o tamanho do nó.
- Portanto, se m=64 e n=1,000,000, é possível encontrar a chave em não mais do que 3 acessos.
- Além disso, as operações de inserção e remoção são tão rápidas quanto a busca.

Árvores Multi-caminho

- Árvores-B são árvores de busca multi-caminho.
- Exemplo:



Árvores-B

- O número de ponteiros em um nó excede o número de chaves em 1.
- O número máximo de ponteiros que podem ser armazenados em um nó é a **ordem** da árvore.
- O número máximo de ponteiros é igual ao número máximo de descendentes de um nó.

5

Árvores-B: *Splitting* e *Promoting*

- Árvores-B possuem um crescimento de baixo para cima (*bottom-up*).
- Para isso, quando um nó se torna cheio ele é dividido (*splitting*) e uma chave é promovida (*promoting*).
- Por exemplo, como inserir uma chave **H** em:

6

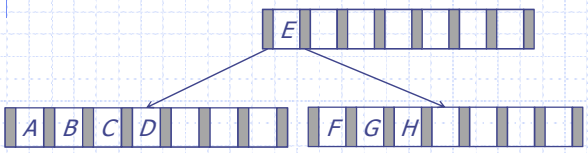
Árvores-B: *Splitting*

- Divide o nó em dois, distribuindo as chaves igualmente.
- Com dois nós, é necessário criar uma nova raiz.

7

Árvores-B: *Promoting*

- Promove uma das chaves que estão no limite da separação para um novo nó superior.



8

Árvores-B: Exemplo

- Vamos fazer um exemplo. Consiste na inserção das seguintes chaves:
  - C S D T A M P I B W N G U R K E H O L J Y Q Z F X V
- Em uma árvore-B de ordem 4.
- (Na lousa)

9

Árvores-B: Implementação

- Existem três operações principais sobre árvores-B:
  - Pesquisa;
  - Percurso;
  - Inserção e,
  - Remoção.

10

Árvores-B: Pesquisa

- A operação de pesquisa pode ser organizada em dois passos:
  - Pesquisa interna ao nó:
    - Realiza uma busca seqüencial ou binária entre as chaves dentro de um nó.
  - Pesquisa externa ao nó:
    - Desce pela árvore à procura da chave de interesse.

11

Árvores-B: Pesquisa

- Vamos assumir a existência de algumas funções de apoio:
  - PesqInterna(*p*, *chave*): realiza a pesquisa interna entre as chaves de um nó *p*. Retorna o maior índice da menor chave maior ou igual à chave de pesquisa.
  - NumChaves(*p*): retorna o número de chaves do nó *p*.

12

Árvores-B: Pesquisa

- Vamos assumir a existência de algumas funções de apoio:
  - Filho(*p*, *i*): retorna o ponteiro para o *i*-ésimo filho do nó *p*.
  - K(*p*, *i*): retorna a *i*-ésima chave do nó *p*.
  - Offset(*p*, *i*): retorna o offset (ou RRN) do registro indexado pela chave *k*(*p*,*i*).

13

Árvores-B: Pesquisa

- Implementação:
  - Pesquisa(Raiz: Ponteiro; Chave: TChave): TOffset;
  - (Na lousa)

14

Árvores-B: Percurso

- Útil para imprimir as chaves em ordem crescente.
- Implementação:
  - Percurso(Raiz: Ponteiro);
  - (Na lousa)

15

Árvores-B: Revisão

- Até o momento foi discutido como os algoritmos de pesquisa e percurso podem ser implementados.
- Nesta aula estudaremos o algoritmo de inserção.
- Antes vamos relembrá-lo, inserindo as chaves 1, 7, 5, 3, 9, 8, 10, 11, 2, 4, 0 em uma árvore de ordem 5.

16

Árvores-B: Inserção

- A função Insere(Chave: TChave; Offset: TOffset): lógico utiliza algumas funções de apoio:
  - PesquisaIns(Raiz: Ponteiro; Chave: TChave; var No: Ponteiro; var Pos: inteiro): lógico é uma variação da função de pesquisa implementada em aula. Ela retorna em No um ponteiro para o nó no qual a chave deve ser inserida e em Pos a posição de inserção nesse nó. O retorno lógico indica se a chave já existe na árvore-B.

17

Árvores-B: Inserção

- Outras funções de apoio:
  - Split(No: Ponteiro; Pos: inteiro; Chave: TChave; Offset: TOffset; NoDir: Ponteiro; var ChavePro: TChave; var OffSetPro: TOffset; var NovoNo: Ponteiro);

18

Árvores-B: Inserção

- Outras funções de apoio:
  - Pai(No: Ponteiro): Ponteiro retorna um ponteiro para o nó pai de No (implementação discutida posteriormente);
  - PosSubArvore(No: Ponteiro): inteiro retorna a posição de No entre as sub-árvores de Pai(No).

19

```
Função Insere(Raiz: Ponteiro; Chave: TChave; Offset: Offset): lógico;
variáveis
    P, NovoNo: Ponteiro;
    Pos: inteiro;
    ChavePro: TChave; OffsetPro: TOffset;
início
    se (PesquisaIns(Raiz, Chave, P, Pos)) retorna falso;
    NoDir := NULO;
    PPai := Pai(P);
    enquanto ((PPai != NULO) e (NumChaves(P) = Ordem-1)) faça
    início
        Split(P, Pos, Chave, Offset, NoDir, ChavePro, OffsetPro, NovoNo);
        NoDir := NovoNo;
        Pos := PosSubArvore(P);
        P := PPai; PPai := Pai(P);
        Chave := ChavePro; Offset := OffsetPro;
    fim;
    se (NumChaves(P) < Ordem-1)
        InsereChave(P, Pos, Chave, Offset, NoDir);
    senão início
        Split(P, Pos, Chave, Offset, NoDir, ChavePro, OffSetPro, NovoNo);
        Raiz := CriaRaiz(ChavePro, OffSetPro, P, NovoNo);
    fim;
    retorna verdade;
fim;
```

Como calcular Pai e PosSubArvore

- Para implementar as funções auxiliares Pai e PosSubArvore pode-se manter um ponteiro em cada nó que aponto o seu pai
  - Dificultaria a implementação;
  - Exigiria acessos extras ao disco atualizar essa informação.

21

Como calcular Pai e PosSubArvore

- Entretanto, como o PesquisaIns percorre a árvores desde a raiz, essa função poderia armazenar os nós antecessores:
  - Por meio de um ponteiro para cada um desses nós (Árvore-B em memória principal)
  - Por meio de buffers que armazenassem os blocos lidos do disco.

22

Definições

- A ordem de uma árvore-B é dada pelo número máximo de descendentes que um 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, 7 chaves por página.

23

Número mínimo de chaves por nó

- Quando um nó é sub-dividido na inserção, as chaves são divididas igualmente entre os nós velho e novo. Deste modo, o número mínimo de chaves em um nó é dado por $\lceil m/2 \rceil - 1$ (exceto para a raiz).
  - **Exemplo:** Uma árvore-B de ordem 8, que armazena no máximo 7 chaves por página, tem, no mínimo, 3 chaves por página.

24

Definição formal das propriedades de árvores-B

- Para uma árvore-B de ordem m:
  - **1.** Cada nó tem, no máximo, m descendentes;
  - **2.** Cada nó, exceto a raiz tem no mínimo $\lceil m/2 \rceil$ descendentes;
  - **3.** A raiz tem, no mínimo, dois descendentes (a menos que seja uma folha também);
  - **4.** Todas as folhas aparecem em um mesmo nível;
  - **5.** Um nó que não é folha e possui k descendentes, contém k-1 chaves
  - **6.** Um nó folha contém, no mínimo $\lceil m/2 \rceil$-1 e, no máximo, m-1 chaves, e nenhum descendente.

25

Árvores-B: Remoção

- Para remover de uma árvore-B deve-se preservar a exigência de $\lceil m/2 \rceil$-1 chaves por nó.
- Como regra geral, todas as chaves são removidas de nós folha:
  - Caso a remoção ocorra em um nó não-folha, a chave sucessora em nó folha deve ser movida para a posição vaga.

26

Árvores-B: Remoção

- Portanto, um *underflow* (nó com menos $\lceil m/2 \rceil$-1 de chaves) somente pode ocorrer em nós folha.

27

Árvores-B: Remoção

- Quando um underflow ocorre duas ações podem ser tomadas:
  - Se um dos irmãos possui mais do que $\lceil m/2 \rceil$-1 de chaves, as chaves podem ser re-distribuídas;
  - Se ambos os irmão possuem $\lceil m/2 \rceil$-1, o nó com underflow e um dos irmãos podem ser *concatenados* ou *consolidados* em um único nó.

28

Árvores-B: Remoção

- A concatenação de dois irmãos faz com que o nó pai tenha que ceder uma chave.
  - Por sua vez, o nó pai pode ser $\lceil m/2 \rceil - 1$ chaves, e requerer uma redistribuição entre os irmãos.
  - Eventualmente, os irmãos do nó pai possuem $\lceil m/2 \rceil - 1$ chaves e uma concatenação seja necessária.

29

Árvores-B: Remoção

- Em um caso extremo, as concatenações podem chegar ao nó raiz:
  - O nó raiz pode ter, no mínimo, uma única chave;
  - Se o nó raiz somente tiver uma chave, a concatenação dos seus nós filhos fará com que a árvore diminua a sua altura em uma unidade.

30

Árvores-B: Remoção

- Entretanto, a consolidação de nós faz com que o nó resultante fique com $m-1$ chaves (máximo permitido):
  - Uma futura inserção nesse nó irá causar uma divisão;
  - Portanto, pode-se permitir menos do que $\lceil m/2 \rceil - 1$, sobretudo quando m é grande, apesar de violar a definição formal de árvores-B.

31

Árvores-B: Remoção

- Vamos ver alguns casos de remoção:
  - Na lousa.

32

Profundidade da Busca no Pior Caso

- Armazena-se 1.000.000 chaves, utilizando uma árvore-B de ordem 512:
  - **Qual a altura máxima que a árvore pode atingir?**
- O pior caso ocorre quando cada página tem apenas o número mínimo de descendentes, e a árvore possui, portanto, altura máxima e largura mínima.

33

Profundidade da Busca no Pior Caso

- O número mínimo de descendentes para o nó raiz é 2,sendo que cada um destes nós, por sua vez, possui no mínimo $\lceil m/2 \rceil$ descendentes.
- O segundo nível, por sua vez, possui no mínimo $2 * \lceil m/2 \rceil$ descendentes.
- Em geral, para um nível d da árvore, o número mínimo de descendentes é dado por $(2 * \lceil m/2 \rceil)^{(d-1)}$.

34

Profundidade da Busca no Pior Caso

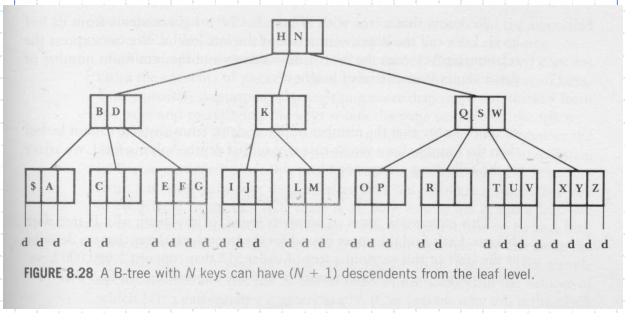


FIGURE 8.28 A B-tree with N keys can have $(N + 1)$ descendents from the leaf level.

35

Profundidade da Busca no Pior Caso

- Uma árvore de n chaves tem $n + 1$ descendentes a partir de seu nível mais inferior, das folhas.
- Seja d a profundidade da árvore no nível dos nós folhas. Podemos expressar a relação entre os $n + 1$ descendentes e o número mínimo de descendentes de uma árvore de altura d como:
  - $d \leq 1 + \log_{\lceil m/2 \rceil} ((n+1)/2)$

36

Profundidade da Busca no Pior Caso - Exemplo

- $d \leq 1 + \log_{\lceil m/2 \rceil} ((n+1)/2)$
- Para a árvore-B de ordem 512 com 1.000.000 de chaves, tem-se
  - $d <= 1 + \log_{256} (500.000,5) <= 3.37$.
  - A nossa árvore terá altura 3, no máximo, e no máximo 3 acessos serão necessários para localizar uma chave, o que é um desempenho muito bom.

37

Variações de Árvores-B

- Árvores-B possuem um problema de desperdício de espaço:
  - Em um caso extremo, aproximadamente 50% do espaço dos nós pode estar livre;
  - Na prática, a média de utilização do espaço é de aproximadamente 69%.

38

Variações de Árvores-B: Árvore-B*

- As árvores-B* tentam resolver esse problema postergando a divisão de um nó:
  - Ao invés de dividir um nó, as chaves são redistribuídas entre os irmãos do nó;
  - Quando o nó e seu irmão estão completos, os dois nós são divididos em 3 nós.
  - Essa modificação garante uma utilização mínima de 67% do espaço (versus 50%).

39

Variações de Árvores-B: Árvore-B Compacta

- As árvores-B compactas tem utilização máxima do armazenamento:
  - A utilização chega a 98%~99% do espaço;
  - Mas não há algoritmo eficiente conhecido para inserção;
  - A árvore-B compacta é criada em períodos de pouco uso a partir de uma árvore-B comum.

40

Variações de Árvores-B: Árvore-B Compacta

- Infelizmente as inserções são caras, pois os nós estão cheios e causam muitas divisões.
- Poucas chaves inseridas (~2%) causam uma rápida degeneração da árvore.
- Árvores-B compactas somente são utilizadas com conjuntos de chaves altamente estáveis.

41

Variações de Árvores-B: Árvore-B+

- Árvores-B+ são muito populares:
  - Permitem percorrer as chaves seqüencialmente de forma simples;
  - Todas as chaves são mantidas em nós folhas e repetidas em nós não-folha;
  - As folhas são ligadas de forma a permitir o acesso seqüencial.

42

Variações de Árvores-B: Árvore-B+

- A busca em uma árvore-B+ somente termina em um nó folha. Chaves em nós não-folha somente servem para direcionar a busca.
- A inserção é a mesma da árvore-B, mas as chaves promovidas mantém uma cópia no nó folha.

43

Bibliografia

- Tenenbaum, Langsam e Augenstein. Estruturas de Dados Usando C. Makron Books. 2005.

44