Algoritmos e Estruturas de Dados II -2022

Profa. Karina Valdivia Delgado

Especificação da segunda parte do segundo trabalho (EP2)

Composição dos grupos: de 2 a 4 alunos.

Problema a ser resolvido

Uma biblioteca deseja manter um cadastro em disco de seus livros. Para cada livro se deseja manter os seguintes atributos:

- código do livro (chave)
- título
- nome completo do primeiro autor e
- ano da publicação.

O objetivo do trabalho é desenvolver um sistema que permita realizar as operações de inserção, remoção e busca dos livros utilizando **uma árvore B a qual será usada como índice.** Os registros deverão ser armazenados em um arquivo e o índice (árvore B) também será guardado em um outro arquivo.

O código base que deverá ser usado para a implementação das operações na árvore B é aquele resultante da parte 1 do EP2.

Mudanças a serem feitas no código base

O objetivo da primeira parte do trabalho 2 foi ter uma implementação correta baseada no pseudocódigo do livro do Cormen das operações de busca, inserção e remoção de árvores B para qualquer valor de t. Nessa implementação a árvore B estava armazenada no arquivo binário tree.dat e usava as seguintes estruturas que representam um registro e um nó da árvore, respectivamente:

```
struct rec
{
  int key
  char country[5];
  char grate[10];
  int score;
  int rate;
}
typedef struct rec recordNode;


  struct bTreeNode
  {

   bool isLeaf;
   int pos;

  int noOfRecs;
```

```
 recordNode* recordArr[2 * t - 1];
 int children[2 * t]; //posições das páginas filhas no arquivo tree.dat, que são no máximo 2t
}
typedef struct bTreeNode bTreeNode;
```

Note que o nó da árvore tem um vetor de tamanho 2t-1 de ponteiros a registros e 2t posições das páginas filhas.

Entre as mudanças necessárias a serem feitas nesta segunda parte do trabalho estão:

1. O registro deverá ser mudado para:

```
struct rec
{
 bool valid; // indica se o registro foi eliminado ou não
 int codigoLivro; // chave
 char titulo[30];
 char nomeCompletoPrimeiroAutor[30];
 int anoPublicacao;
}
typedef struct rec recordNode;
```

2. Os registros completos (incluindo o valor da chave e dos demais campos) serão armazenados em um arquivo separado chamado **data.dat**.

3. Em cada nó da árvore não teremos mais 2t-1 ponteiros a registros. No lugar desses ponteiros, teremos 2t-1 chaves e 2t-1 posições em que estão armazenados os registros correspondentes no arquivo data.dat. Assim, o nó da árvore deve ser mudado para:

```
 struct bTreeNode
 {

 bool valid; //indica se o nó da árvore B foi eliminado ou não
 int noOfRecs;
 bool isLeaf;
 int pos;


 int keyRecArr[2 * t - 1]; //as chaves dos registros,  que são no máximo 2t-1
 int posRecArr[2 * t - 1]; //posições dos registros no arquivo data.dat, que são no máximo 2t-1
 int children[2 * t]; //posições das páginas filhas no arquivo tree.dat, que são no máximo 2t
}
typedef struct bTreeNode bTreeNode;
```

4. O arquivo de índice que usa uma árvore B com grau mínimo t será armazenado no arquivo tree.dat como era feito anteriormente, a única diferença é que a estrutura de cada nó desta árvore é diferente como comentado no item anterior.

5. Não será usado mais o arquivo dataset.csv no qual estavam os dados que seriam inseridos na árvore B. Agora os **dados serão inseridos manualmente no terminal**.

6. As páginas excluídas da árvore B devem ter o campo valid = false. O espaço da página excluída não

será reutilizado para novas inserções pois toda inserção é sempre feita no final do arquivo (isto já estava sendo feito pelo código da primeira parte do EP1).

7. Para remover um registro, o registro deverá ser localizado acessando primeiro o índice. A remoção deve colocar o campo valid=false do registro removido no arquivo data.dat. O espaço do registro removido não será reutilizado para novas inserções. Toda inserção será sempre no fim do arquivo (supõe-se, assim, que um processo de compactação eliminaria os espaços disponíveis em outro processo, o qual não deve ser implementado).

Assim, teremos apenas os seguintes arquivos:

a) um arquivo binário chamado tree.dat em que a árvore B é armazenada
b) um arquivo binário chamado data.dat em que são armazenados os registros (livros).

Sobre as operações permitidas no sistema

Os comandos permitidos para usar o sistema são representados por uma letra maiúscula. Toda linha de entrada obrigatoriamente inicia com um comando.

1. Insere livro atualizando o índice

I <código>;<título>;<autor>;<ano>

Insere um novo livro contendo todos os campos.

2. Remove livro a partir do código atualizando o índice

R <código>

Remove um livro a partir da sua chave.

3. Busca livro a partir de seu código utilizando o índice

B <código>

Busca um livro a partir da sua chave e imprime o registro da seguinte forma:

<código> <título> <autor> <ano>

4. Imprimir o índice (a árvore B)

P1

Imprime todas as páginas do índice, sem incluir as páginas que foram eliminadas

5. Imprimir o conteúdo do arquivo indice.dat

P2

Imprime todas as páginas que estão no arquivo indice.dat na ordem em que estão no arquivo, inclusive as páginas que foram eliminadas.

6. Imprimir o conteúdo do arquivo data.dat

P3

Imprime todos os registros que estão no arquivo data.dat na ordem em que estão no arquivo, inclusive os registros que foram eliminados.

7. Finaliza a execução fechando o arquivo de índice e o arquivo de dados

F

CASO DE TESTE 1

Exemplo de saída esperada

A seguir mostramos a saída esperada do programa em termos dos dois arquivos .dat criados para t=2.

Depois de aplicar o comando:
I 221549;Java: como programar; Deitel e Deitel; 2011

Deve ser criada uma página com a seguinte estrutura a qual deve estar armazenada em indice.dat

valid	noOfRecs	isLeaf	pos	keyRecArr			posRecArr			children			
true	1	true	0	221549			0			-1	-1	-1	-1

O arquivo dados.dat deve conter:

true	**221549**	**Java: como programar**	**Deitel e Deitel**	**2011**

Depois de aplicar o comando
I 210591;Projeto de algoritmos;Nivio Ziviani;2011

O arquivo indice.dat deve conter:

true	2	true	0	221549	**210591**		0	1		-1	-1	-1	-1

E o arquivo dados.dat deve conter :

true	**221549**	**Java: como programar**	**Deitel e Deitel**	**2011**	**true**	**210591**
Projeto de algoritmos		**Nivio Ziviani**		**2011**		

Depois de aplicar o comando
R 221549

O arquivo indice.dat deve conter:

true	1	true	0	**210591**			1			-1	-1	-1	-1

E o arquivo dados.dat deve conter :

false	**221549**	**Java: como programar**	**Deitel e Deitel**	**2011**	**true**	**210591**
Projeto de algoritmos	**Nivio Ziviani**	**2011**				

Depois de aplicar o comando:

B 221549

O seguinte texto é mostrado na tela:

"O livro com código 221549 não existe na biblioteca"

E os arquivos indice.dat e dados.dat permanecem iguais.

Depois de aplicar o comando:

B 210591

O seguinte texto é mostrado na tela:

"210591 Projeto de algoritmos Nivio Ziviani 2011"

E os arquivos indice.dat e dados.dat permanecem iguais.

CASO DE TESTE 2

Caso de teste com dados artificiais e com t=5

I 1;T;A;2001
I 2;T;A;2002
I 3;T;A;2003
I 4;T;A;2004
I 5;T;A;2005
I 6;T;A;2006
I 7;T;A;2007
I 8;T;A;2008
I 9;T;A;2009
I 10;T;A;2010
I 11;T;A;2011
I 12;T;A;2012
I 13;T;A;2013
I 14;T;A;2014
I 15;T;A;2015
I 16;T;A;2016
I 17;T;A;2017
I 18;T;A;2018

I 19;T;A;2019
I 20;T;A;2020
I 21;T;A;2021
I 22;T;A;2022
I 23;T;A;2023

I 24;T;A;2024
I 25;T;A;2025
I 26;T;A;2026
I 27;T;A;2027
I 28;T;A;2028
I 29;T;A;2029
I 30;T;A;2030
P1
P2
P3
B 50
R 15
R 13
R 8
R 30
R 12
I 32;T;A;2032
R 1
R 2
R 3
R 4
I 31;T;A;2031
B 15
B 7
P1
P2
P3
R 5
R 6
R 7
R 29
R 28
R 27
R 26
R 25
R 24
R 23
R 22
R 21
P1
P2

P3

O que deve ser entregue:

a) **Código desenvolvido pelo grupo:** o grupo deverá fazer o upload de todos os arquivos fonte no ambiente e-disciplinas.
b) **Vídeo de apresentação:** o grupo deverá gravar e enviar no ambiente e-disciplinas um vídeo com duração de 10 a 15 minutos. Todos os integrantes do grupo devem participar da apresentação no vídeo. Uma janela com a imagem do aluno realizando a explicação deverá fazer parte dos vídeos.
c) **Formulário:** O grupo precisa preencher o formulário que estará disponível no e-disciplinas a partir do dia 03/07.

Detalhes esperados no vídeo (listagem não exaustiva):
● Apresentação dos integrantes do grupo (nomes completos).
● Explicação detalhada do código, mostrando especialmente as principais mudanças feitas no código. No vídeo responder a seguinte pergunta: Quais métodos já existentes o grupo precisou modificar e/ou quais precisou criar para poder trabalhar com as duas novas estruturas (rec e bTreeNode)?
● Além de mostrar a execução dos algoritmos nos dois casos de teste do enunciado, o grupo deve elaborar e mostrar no vídeo um desenho do estado da árvore B final encontrada para os dois casos de teste.
● Descrição curta das dificuldades encontradas.

IMPORTANTE: o tempo de vídeo é um critério de estabelecimento de avaliação igualitária entre os alunos, ou seja, vídeos "acelerados" implicará em descontos na nota, podendo inclusive ser um fator determinante para a atribuição de nota 0 (zero) para esse quesito do trabalho. Uma possibilidade pode ser acelerar o vídeo enquanto mostra a execução dos casos de teste, mas a gravação da voz deve estar em velocidade normal.

IMPORTANTE: você pode disponibilizar o seu vídeo em qualquer repositório público ou no sistema e-disciplinas. Se você disponibilizar em um repositório público você deverá depositar no sistema e-disciplinas, até a data limite de entrega do trabalho, um arquivo .**pdf** com o link e a duração exata do vídeo.

Este é um documento preliminar e pode sofrer alterações ao longo do tempo no sentido de torná-lo mais didático e mais detalhado.