Algoritmos e Estruturas de Dados II -2022
Profa. Karina Valdivia Delgado

Especificação da primeira parte do segundo trabalho (EP2)

Composição dos grupos: de 2 a 4 alunos.

O objetivo dessa primeira parte do trabalho 2 é entender as operações de busca, inserção e remoção de árvores B para qualquer valor de **t** e ter uma implementação correta baseada no pseudocódigo do livro do Cormen. A implementação disponível no e-disciplinas é uma versão modificada do código que está disponível em: https://github.com/Rahul-RB/B-Tree/tree/master/btrees_files, essa última tinha alguns problemas. A implementação disponível no e-disciplinas pode ainda conter alguns erros de implementação, assim, a sua tarefa é entender, documentar, explicar e corrigir essa implementação.

A seguir alguns detalhes sobre essa implementação:

- Os dados (registros) que são inseridos na árvore são os que estão no arquivo https://github.com/Rahul-RB/B-Tree/blob/master/btrees_files/data/dataset.csv
- Cada registro tem os seguintes campos: key, country, grate, score e rate. A seguinte estrutura representa um registro:

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

- A árvore B é armazenada no arquivo binário tree.dat. Note que nessa implementação cada nó da árvore-B tem tamanho fixo. As seguintes estruturas representam a árvore e um nó da árvore:

```
struct tree
{
        char fileName[20];
        FILE* fp;
        int root;
        int nextPos;
};
typedef struct tree bTree;
```

```
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.

- No arquivo Makefile primeiro executar o clean e se for usado Windows substituir rm -rf *.o por del *.o

Coloque comentários em cada uma das funções do código e responda às seguintes perguntas:

1. Qual a diferença entre *pos* de *bTreeNode* e *nextPos* de *tree*?
2. Quais as funções relacionadas com a inserção e o que cada uma delas faz? A inserção dessa implementação funciona corretamente? Caso não, que mudanças foram feitas e em que funções para obter uma versão correta da inserção?

   Por exemplo, teste com
   make all t=3
   ./run -b
   How many records do you want to build from dataset? 28

   Faça um desenho da árvore B que deveria ser criada mostrando as chaves dos registros. A lista de links de todos os nós criados pelo algoritmo está correta?

3. Quais as funções relacionadas com a remoção e o que cada uma delas faz? A remoção dessa implementação funciona corretamente? Caso não, que mudanças foram feitas e em que funções para obter uma versão correta da remoção? Faça as remoções dos seguintes registros da árvore criada no item 2: 774597, 996522, 782891 e 676106.
   Faça um desenho do estado da árvore após essas remoções.
4. Quais as funções relacionadas com a busca e o que cada uma delas faz? A operação de busca está corretamente implementada? Buscar 132486 e 990171.
5. O que faz a função traverse?
6. Os registros criados do tipo *recordNode* estão sendo armazenados na memória principal ou em um arquivo? Depois de fechar o programa é possível recuperar esses registros criados?

O que deve ser entregue:

a) **Código modificado pelo grupo**: o grupo deverá fazer o upload de todos os arquivos fonte no ambiente e-disciplinas.
b) **Respostas para as perguntas:** o grupo deverá fazer o upload de um arquivo em formato pdf com as respostas para as perguntas.