Organização e Recuperação de Dados

2º Trabalho Prático

A especificação abaixo deverá ser implementada utilizando a linguagem Python e poderá ser realizada em equipes de até 2 alunos. A equipe deverá enviar o código fonte pelo Google Classroom (apenas arquivos .py). Posteriormente, a equipe deverá apresentar o trabalho em horário previamente agendado pela professora.

Especificação

O objetivo deste trabalho é criar um programa que, a partir de um arquivo de registros, construa e mantenha **um índice estruturado como uma árvore-B.**

O arquivo de registros conterá informações sobre jogos e estará armazenado no arquivo **games.dat** no mesmo formato do arquivo dados.dat utilizado no 1º trabalho prático. Para cada registro lido, a chave (o identificador do jogo) deverá ser inserida em uma árvore-B de determinada **ORDEM**, juntamente com o *byte-offset* do registro correspondente. Para facilitar a experimentação com árvores de diferentes ordens, **defina ORDEM como uma constante** e use-a ao longo do código sempre que precisar referenciar a ordem da árvore. **Esteja ciente de que o seu programa será testado com árvores de diferentes ordens.**

A estrutura interna das páginas da árvore-B será similar à vista em aula, porém deverá conter, adicionalmente, uma lista para o armazenamento dos *byte-offsets* dos registros. **A árvore-B deverá ser criada e mantida em arquivo**, logo, nunca estará **completamente** carregada na memória.

O programa deverá oferecer as seguintes funcionalidades:

- Criação do índice (árvore-B) a partir do arquivo de registros (opção -c);
- Execução de um arquivo de operações (apenas busca e inserção) (opção -e);
- Impressão das informações do índice, i.e., da árvore-B (opção -p).

Criação do índice (i.e., a árvore-B)

A funcionalidade de criação da árvore-B (`-c`) deverá ser acessada pela linha de comando, da seguinte forma:

```
$ programa -c
```

sendo `programa` o nome do arquivo executável do seu programa. Sempre que ativada, essa funcionalidade fará a leitura do arquivo **games.dat** para a construção do índice, i.e., inserção dos pares {chave, *byte-offset*} na árvore-B que deverá ser armazenada no arquivo **btree.dat**. Caso o arquivo btree.dat exista, ele deverá ser sobrescrito. Note que o formato do arquivo games.dat será sempre o mesmo, porém **o número de registros nesse arquivo pode variar**. Para simplificar o processamento do arquivo de registros, considere que ele sempre será fornecido corretamente (i.e., o seu programa não precisa verificar a integridade desse arquivo) e que ele armazenará **o total de registros do arquivo no cabeçalho** como um número inteiro de 4 bytes. **Neste trabalho, não serão feitas remoções** e, consequentemente, não haverá gerenciamento de espaços disponíveis.

Ao final da criação do índice, o programa deverá apresentar uma mensagem na tela indicando se essa operação foi concluída com sucesso ou não.

Execução do arquivo de operações

Dados os arquivos **games.dat** e **btree.dat**, o seu programa deverá executar as seguintes operações:

- Busca de um jogo pelo IDENTIFICADOR;
- Inserção de um novo jogo.

As operações a serem realizadas em determinada execução serão especificadas em um arquivo de operações no mesmo formato utilizado no 1º trabalho. Dessa forma, **o programa não possuirá interface com o usuário** e executará as operações na sequência em que estiverem especificadas no arquivo de operações.

A execução do arquivo de operações será acionada pela linha de comando, no seguinte formato:

```
$ programa -e nome_arquivo_operacoes
```

sendo `programa` o nome do arquivo executável do seu programa, `-e` a flag que sinaliza o modo de execução e `nome_arquivo_operacoes` o nome do arquivo que contém as operações a serem executadas. Para simplificar o processamento do arquivo de operações, considere que ele sempre será fornecido corretamente (i.e., o seu programa não precisa verificar a integridade desse arquivo).

Observe que, para esse tipo de execução, os arquivos **games.dat** e **btree.dat** devem existir. Caso eles não existam, o programa deve apresentar uma mensagem de erro e terminar.

Considere como exemplo o arquivo de operações abaixo:

```
b 22
i 147|Resident Evil 2|1998|Survival horror|Capcom|PlayStation|
b 95
b 230
i 181|Pac-Man|1980|Maze|Namco|Arcade|
i 147|Resident Evil 2|1998|Survival horror|Capcom|PlayStation|
```

Esse arquivo representa a execução consecutiva das seguintes operações:

- Busca pelo registro de chave 22
- Inserção do registro do jogo de identificador 147 (“Resident Evil 2”)
- Busca pelo registro de chave 95
- Busca pelo registro de chave 230
- Inserção do registro do filme de identificador 181 (“Pac-Man”)
- Inserção do registro do jogo de identificador 147 (“Resident Evil 2”)

As buscas deverão ser realizadas no índice, i.e., no arquivo btree.dat. Uma vez localizada a chave no índice, o *byte-offset* do registro correspondente deverá ser recuperado e **o registro deverá ser acessado de forma direta** no arquivo **games.dat**.

As inserções sempre acontecerão no fim do arquivo games.dat e, complementarmente, as informações do novo registro (chave e *byte-offset*) deverão ser inseridas no índice do arquivo **btree.dat**. Não será permitida a inserção de registros com chave duplicada. Dessa forma, **antes de realizar uma inserção, o índice deverá ser consultado** e uma mensagem de erro deverá ser dada caso se identifique a duplicação.

Utilizando como exemplo o arquivo de operações mostrado acima, o seu programa deverá apresentar:

```
Busca pelo registro de chave "22"
22|Tetris|1984|Puzzle|Elorg|Electronika 60| (43 bytes - offset 1293)

Insercao do registro de chave "147"
147|Resident Evil 2|1998|Survival horror|Capcom|PlayStation| (60 bytes - offset 6460)

Busca pelo registro de chave "95"
95|Braid|2008|Puzzle-platformer|Microsoft Game Studios|Xbox 360| (64 bytes - offset 6092)

Busca pelo registro de chave "230"
Erro: registro nao encontrado!

Insercao do registro de chave "181"
181|Pac-Man|1980|Maze|Namco|Arcade| (35 bytes - offset 6522)

Insercao do registro de chave "147"
Erro: chave "147" já existente!
```

Impressão das informações da árvore-B

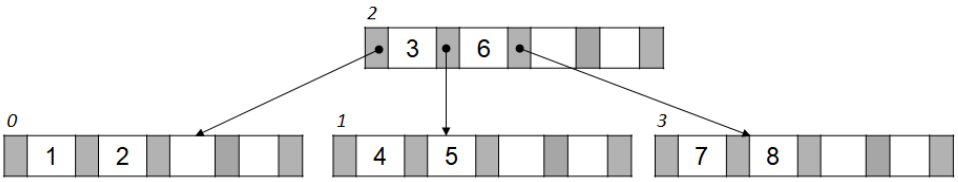
A funcionalidade de impressão das informações da árvore-B (`-p`) também será acessada via linha de comando, no seguinte formato:

```
$ programa -p
```

sendo `programa` o nome do arquivo executável do seu programa. Note que, para essa execução, o arquivo **btree.dat** deve existir. Caso o arquivo contrário, o programa deve apresentar uma mensagem de erro e terminar.

Sempre que ativada, essa funcionalidade apresentará na tela o conteúdo de todas as páginas da árvore-B armazenada no arquivo **btree.dat**. Para cada página da árvore deverá ser informado: (a) o seu RRN; (b) os valores do vetor de chaves; (c) os valores do vetor de *byte-offsets*; e (d) os valores do vetor de filhos. As páginas devem ser apresentadas pela ordem do seu RRN e a página raiz deve ser devidamente identificada.

Como um exemplo, considere uma árvore-B de ordem 5 que indexa um arquivo com os oito primeiros registros do arquivo games.dat utilizado no 1º trabalho.



Supondo a árvore-b acima, seu programa deverá apresentar as seguintes informações:

```
Página 0
Chaves: 1 | 2
Offsets: 4 | 86
Filhas: -1 | -1 | -1

Página 1
Chaves: 4 | 5
Offsets: 218 | 270
Filhas: -1 | -1 | -1

- - - - - - Raiz - - - - - -
Página 2
Chaves: 3 | 6
Offsets: 169 | 332
Filhas: 0 | 1 | 3
- - - - - - - - - - - - - -

Página 3
Chaves: 7 | 8 | 9
Offsets: 377 | 432 | 510
Filhas: -1 | -1 | -1 | -1
```

Qualquer dúvida referente a esta especificação deverá ser sanada com os respectivos professores.

BOM TRABALHO!