PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Trabalho Prático 4

Cursos	: *Engenharia de Computação*
	Sistemas de Informação
Disciplina	: *Algoritmos e Estruturas de Dados*
Professora	: *Eveline Alonso Veloso*

Regras Básicas:

1. Desenvolva esse trabalho prático a partir do que já foi implementado nos Trabalhos práticos 2 e 3.
2. Estude bastante cada par de entrada/saída.
3. Todos os programas deverão ser desenvolvidos na linguagem de programação Java.
4. Esse trabalho prático poderá ser desenvolvido em grupos de, no máximo, três integrantes.
5. Cópias de trabalho, se existirem, serão encaminhadas ao colegiado de coordenação didática do curso.
6. Fique atento ao *charset* dos arquivos de entrada e saída. Recomenda-se a utilização dos métodos da classe MyIO.java para leitura de dados do teclado. É necessário definir o *charset* a ser utilizado antes de começar a realizar a leitura de dados do teclado, da seguinte forma: MyIO.*setCharset*("UTF-8").
7. Para cada exercício, vocês devem submeter apenas um arquivo (.java) por grupo. Essa regra será necessária para a submissão de trabalhos no VERDE e no identificador de plágios utilizado na disciplina.
8. A resolução (código) de cada exercício deverá ser submetida ao VERDE.
9. A correção será realizada automaticamente pelo VERDE e validada por meio de apresentações durante as aulas práticas da disciplina.

Base de Dados:

Spotify é um serviço de *streaming* de música, *podcast* e vídeo lançado oficialmente em 7 de outubro de 2008. É atualmente o serviço de *streaming* mais popular e usado do mundo. Desenvolvido pela *startup* Spotify AB (Estocolmo, Suécia), fornece conteúdo protegido de restrições relacionadas a direitos digitais de gravadoras e empresas de mídia. O Spotify é um serviço *freemium*; com recursos básicos gratuitos, com propagandas ou limitações, enquanto recursos adicionais, como qualidade de transmissão aprimorada e *downloads* de música, oferecidos para assinaturas pagas.

Spotify está disponível na maior parte da Europa, parte da América, Austrália, Nova Zelândia e parte da Ásia. Disponível para a maioria dos dispositivos modernos,

incluindo computadores Windows, macOS e Linux, bem como *smartphones* e *tablets* com iOS, Windows Phone e Android. Pode-se encontrar as músicas desejadas por navegação ou pesquisas referentes a artista, álbum, gênero, lista de reprodução ou gravadora. Seus usuários podem criar, editar ou compartilhar *playlists*, compartilhar faixas em redes sociais ou fazer *playlists* com outros usuários. Fornece acesso a mais de 30 milhões de músicas e, em julho de 2019, contava com mais de 232 milhões de usuários ativos, incluindo 108 milhões de assinantes pagantes.

O arquivo dataAEDs.csv contém um conjunto de dados relacionados a mais de 175.000 músicas coletadas da plataforma Spotify Web API, que podem ser agrupadas por artista, ano ou gênero. Tal arquivo deve ser copiado para a pasta /tmp.

Exercícios:

Pesquisa sequencial e pesquisa binária:

1. **Pesquisa sequencial**
   Faça a inserção dos registros correspondentes a algumas músicas em um vetor e, em seguida, faça algumas pesquisas sequenciais. A chave primária de pesquisa será o atributo ***name***.
   A entrada padrão é dividida em duas partes. A primeira contém, em cada linha, uma *string* indicando o *id* da música cujos dados devem ser inseridos no vetor de músicas. Após a palavra FIM, inicia-se a segunda parte da entrada padrão, que também é composta por várias linhas, sendo que cada uma possui o *nome* de uma música que deve ser pesquisada no vetor de músicas. A última linha dessa parte também terá a palavra FIM.
   A saída padrão será composta por várias linhas contendo as palavras SIM/NAO para indicar se cada uma das músicas pesquisadas existe ou não no vetor de músicas.
   Além disso, crie um arquivo de *log* na pasta corrente com o nome matrícula_sequencial.txt com uma única linha contendo: seu número de matrícula, tempo de execução de seu algoritmo (em milissegundos) e número de comparações realizadas. Todas as informações desse arquivo de *log* devem ser separadas por uma tabulação '\t'.

2. **Pesquisa binária**
   Repita a questão **Pesquisa sequencial**, contudo, aplicando agora pesquisa binária implementada recursivamente.
   O nome do arquivo de *log* desta questão será matrícula_binaria.txt.
   A entrada desta questão não está ordenada.

Árvores:

3. **Árvore Binária de Busca**
   Crie uma árvore binária de busca. Em seguida, faça a inserção dos registros correspondentes a algumas músicas conforme a entrada padrão. A chave de pesquisa será o atributo ***name*** da música. Não insira uma música se sua chave

já estiver na árvore. Por fim, pesquise se alguns registros estão cadastrados na árvore, mostrando seus respectivos caminhos de pesquisa.
A entrada padrão é igual a da questão **Pesquisa sequencial em Java**.
A saída padrão será composta por várias linhas, uma para cada pesquisa realizada na árvore. Cada linha será composta pelo caminho de pesquisa, ou seja, os ***nomes*** das músicas que foram inspecionados durante o processamento da pesquisa; e, no final, pelas palavras SIM ou NAO, indicando se cada um dos nomes pesquisados existe ou não na árvore.
Além disso, crie um arquivo de *log* na pasta corrente com o nome matrícula_arvoreBinaria.txt com uma única linha contendo: seu número de matrícula, tempo de execução de seu algoritmo (em milissegundos) e número de comparações realizadas. Todas as informações desse arquivo de *log* devem ser separadas por uma tabulação '\t'.

4. ***Tree sort***
   Tree sort é um algoritmo de ordenação. Nesse algoritmo, constrói-se uma árvore binária de busca com os elementos a serem classificados e, em seguida, percorre-se a árvore (utilizando caminhamento em ordem) para que seus elementos sejam impressos em ordem de classificação.
   Utilizando uma árvore binária de busca, ordene os registros das músicas do Spotify aplicando o algoritmo de ordenação *tree sort*, considerando que a chave de pesquisa seja o atributo ***name***. Em caso de empate, o segundo critério de ordenação deve ser o atributo ***id*** da música.
   A entrada padrão é composta por várias linhas e cada uma contém uma *string* indicando o *id* da música que deve ser inserida na árvore binária de busca de músicas. A última linha da entrada contém a palavra FIM.
   A saída padrão corresponde aos registros ordenados, um por linha. Em cada linha da saída, escreva os dados do registro correspondente.
   Além disso, crie um arquivo de *log* na pasta corrente com o nome matrícula_treeSort.txt com uma única linha contendo: seu número de matrícula, tempo de execução de seu algoritmo (em milissegundos) e número de comparações realizadas para inserção de todas as músicas na árvore. Todas as informações desse arquivo de *log* devem ser separadas por uma tabulação '\t'.

Tabelas ***hash***:

5. **Tabela *hash* com endereçamento aberto e *rehashing***
   Crie uma tabela *hash* com endereçamento aberto e *rehashing*.
   Em seguida, faça a inserção dos registros correspondentes a algumas músicas conforme a entrada padrão.
   A função de transformação que deve ser aplicada é: **(duration mod tamanho da tabela *hash* + k * (duration mod 31)) mod tamanho da tabela *hash***, onde o tamanho da tabela *hash* é 839 e *k* indica o número de tentativas de se inserir o registro na tabela, começando com o valor 0.
   Por fim, pesquise se algumas músicas estão cadastradas na tabela *hash*, mostrando suas respectivas posições nessa tabela.
   A entrada padrão é igual a da questão **Pesquisa sequencial**.

A saída padrão será composta por várias linhas, uma para cada pesquisa realizada na tabela *hash*. Cada linha deve apresentar a posição de cada música procurada na tabela *hash*. Se o elemento desejado não estiver na tabela, escreva a palavra NAO.
Além disso, crie um arquivo de *log* na pasta corrente com o nome matrícula_hashRehashing.txt com uma única linha contendo: seu número de matrícula, tempo de execução de seu algoritmo (em milissegundos) e número de comparações realizadas. Todas as informações desse arquivo de *log* devem ser separadas por uma tabulação '\t'.

6. **Tabela *hash* com endereçamento em separado**

Crie uma tabela *hash* com endereçamento em separado.
Em seguida, faça a inserção dos registros correspondentes a algumas músicas conforme a entrada padrão.
A função de transformação que deve ser aplicada é: **(duration mod tamanho da tabela *hash*)**, onde o tamanho da tabela *hash* é 233.
Por fim, pesquise se algumas músicas estão cadastradas na tabela *hash*, mostrando suas respectivas posições nessa tabela.
A entrada padrão é igual a da questão **Pesquisa sequencial**.
A saída padrão será composta por várias linhas, uma para cada pesquisa realizada na tabela *hash*. Cada linha deve apresentar a posição de cada música procurada na tabela *hash*. Se o elemento desejado não estiver na tabela, escreva a palavra NAO.
Além disso, crie um arquivo de *log* na pasta corrente com o nome matrícula_hashSeparado.txt com uma única linha contendo: seu número de matrícula, tempo de execução de seu algoritmo (em milissegundos) e número de comparações realizadas. Todas as informações desse arquivo de *log* devem ser separadas por uma tabulação '\t'.