Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

Trabalho Prático II

Regras Básicas

1. extends Trabalho Prático 01

2. Fique atento ao Charset dos arquivos de entrada e saída.

Classe + Registro

A National Basketball Association (em português: Associação Nacional de Basquetebol; abreviação oficial: NBA) é a principal liga de basquetebol profissional da América do Norte. Com 30 franquias sendo membros da mesma (29 nos Estados Unidos e 1 no Canadá), a NBA também é considerada a principal liga de basquete do mundo. É um membro ativo da USA Basketball (USAB), que é reconhecida pela FIBA (a Federação Internacional de Basquetebol) como a entidade máxima e organizadora do basquetebol nos Estados Unidos. A NBA é uma das 4 'major leagues' de esporte profissional na América do Norte. Os jogadores da NBA são os mais bem pagos esportistas do mundo, por salário médio anual.

A liga foi fundada na cidade de Nova Iorque em 6 de Junho de 1946, como a Basketball Association of America (BAA). A liga adotou o nome de National Basketball Association em 1949 quando se fundiu com a rival National Basketball League (NBL). A liga tem diversos escritórios ao redor do mundo, além de vários dos próprios clubes fora da sede principal na Olympic Tower localizada na Quinta Avenida 645. Os estúdios da NBA Entertainment e da NBA TV são localizados em Secaucus, New Jersey.

O arquivo **players.csv** contém um conjunto de dados de jogadores da liga de basquete norte americana - NBA extraídos do site `https://www.kaggle.com/drgilermo/nba-players-stats`. Essa base contém registros de jogadores desde 1950. Um total de 67 temporadas da NBA. Este arquivo **sofreu algumas adaptações** para ser utilizado neste e nos próximos trabalhos práticos. Tal arquivo deve ser copiado para a pasta /tmp/. Quando reiniciamos o Linux, ele normalmente apaga os arquivos existentes na pasta /tmp/.

Implemente os itens pedidos a seguir.

1. **Classe em Java**: Crie uma classe *Jogador* seguindo todas as regras apresentadas no slide unidade01g_conceitosBasicos_introducaoOO.pdf. Sua classe terá os atributos privados `id`(int), `nome`(String), `altura`(int), `peso`(int), `universidade` (String), `anoNascimento` (int), `cidadeNascimento`(String), `estadoNascimento`(String). Sua classe também terá pelo menos dois construtores, e os métodos *gets*, *sets*, *clone*, *imprimir* e *ler*. O método *imprimir* mostra os atributos do registro (ver cada linha da saída padrão) e o *ler* lê os atributos de um registro. Atenção para o arquivo de entrada, pois em alguns registros faltam valores e esse foi substituído pelo valor "nao informado".

   A entrada padrão é composta por várias linhas e cada uma contém uma string indicando o id do jogador a ser lido. A última linha da entrada contém a palavra FIM. A saída padrão também contém várias linhas, uma para cada registro contido em uma linha da entrada padrão.

2. **Registro em C**: Repita a anterior criando o registro *Jogador* na linguagem C.

Pesquisa

3. **Pesquisa Sequencial em Java**: Faça a inserção de alguns registros no final de um vetor e, em seguida, faça algumas pesquisas sequenciais. A chave primária de pesquisa será o atributo **nome**. A entrada padrão é composta por duas partes onde a primeira é igual a entrada da primeira questão. As demais linhas correspondem a segunda parte. A segunda parte é composta por várias linhas. Cada uma possui um elemento que deve ser pesquisado no vetor. A última linha terá a palavra FIM. A saída padrão será composta por várias linhas contendo as palavras SIM/NAO para indicar se existe cada um dos elementos pesquisados. Além disso, crie um arquivo de log na pasta corrente com o nome matrícula_sequencial.txt com uma única linha contendo sua matrícula, tempo de execução do seu algoritmo e número de comparações. Todas as informações do arquivo de log devem ser separadas por uma tabulação '\t'.

4. **Pesquisa Binária em C**: Repita a questão anterior, contudo, usando a Pesquisa Binária. A entrada e a saída padrão serão iguais as da questão anterior. O nome do arquivo de log será matrícula_binaria.txt. A entrada desta questão **não** está ordenada.

Ordenação

Observação: ATENÇÃO para os algoritmos de ordenação que já estão implementados no Github!

5. **Ordenação por Seleção em Java**: Usando vetores, implemente o algoritmo de ordenação por seleção considerando que a chave de pesquisa é o atributo **nome**. A entrada e a saída padrão são iguais as da primeira questão, contudo, a saída corresponde aos registros ordenados. Além disso, crie um arquivo de log na pasta corrente com o nome matrícula_selecao.txt com uma única linha contendo sua matrícula, número de comparações (entre elementos do *array*), número de movimentações (entre elementos do *array*) e o tempo de execução do algoritmo de ordenação. Todas as informações do arquivo de log devem ser separadas por uma tabulação '\t'.

6. **Ordenação por Seleção Recursiva em C**: Repita a questão anterior, contudo, usando a Seleção Recursiva. A entrada e a saída padrão serão iguais as da questão anterior. O nome do arquivo de log será matrícula_selecaoRecursiva.txt.

7. **Ordenação por Inserção em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo de Inserção, fazendo com que a chave de pesquisa seja o atributo **anoNascimento**. O nome do arquivo de log será matrícula_insercao.txt.

   (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do jogador)

8. **Shellsort em C**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Shellsort, fazendo com que a chave de pesquisa seja o atributo **peso**. O nome do arquivo de log será matrícula_shellsort.txt.

9. **Heapsort em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Heapsort, fazendo com que a chave de pesquisa seja o atributo **altura**. O nome do arquivo de log será matrícula_heapsort.txt.

10. **Quicksort em C**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Quicksort, fazendo com que a chave de pesquisa seja o atributo **estadoNascimento**. O nome do arquivo de log será matrícula_quicksort.txt.

11. **Counting Sort em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Counting Sort, fazendo com que a chave de pesquisa seja o atributo **altura**. O nome do arquivo de log será matrícula_countingsort.txt.

12. **Bolha em C**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo da Bolha, fazendo com que a chave de pesquisa seja o atributo **anoNascimento**. O nome do arquivo de log será matrícula_bolha.txt.

13. **Mergesort em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Mergesort, fazendo com que a chave de pesquisa seja o atributo **universidade**. O nome do arquivo de log será matrícula_mergesort.txt.

14. **Radixsort em C**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Radixsort, fazendo com que a chave de pesquisa seja o atributo **id**. O nome do arquivo de log será matrícula_radixsort.txt.

15. **Ordenação PARCIAL por Seleção em Java**: Refaça a Questão "Ordenação por Seleção" considerando a ordenação parcial com k igual a 10.

16. **Ordenação PARCIAL por Inserção em C**: Refaça a Questão "Ordenação por Inserção" considerando a ordenação parcial com k igual a 10.

17. **Heapsort PARCIAL em C**: Refaça a Questão "Heapsort" considerando a ordenação parcial com k igual a 10.

18. **Quicksort PARCIAL em Java**: Refaça a Questão "Quicksort" considerando a ordenação parcial com k igual a 10.