Pontificia 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

Harry Potter é uma série de sete romances de fantasia escrita pela autora britÃ¢nica J.K. Rowling. A narrativa centraliza-se em Harry Potter, um jovem órfão que descobre, no seu décimo primeiro aniversário, que é um bruxo. Ele é convidado a estudar na Escola de Magia e Bruxaria de Hogwarts, onde aprende a prática da magia sob a orientação do gentil diretor Albus Dumbledore e de outros professores da escola. Harry descobre que é famoso no mundo dos bruxos por ter sobrevivido a um ataque letal do poderoso e maligno bruxo Lord Voldemort quando era apenas um bebê. Voldemort matou os pais de Harry, mas por algum motivo, sua maldição mortal não funcionou em Harry.

Ao longo dos livros, Harry é acompanhado por seus melhores amigos, Ronald Weasley e Hermione Granger. Juntos, eles enfrentam diversos desafios, incluindo o retorno de Voldemort, que busca não apenas conquistar o mundo bruxo, mas também destruir Harry, o único que pode impedir seus planos maléficos.

A série alcançou uma imensa popularidade, aclamação da crítica e sucesso comercial em todo o mundo. Além dos livros, a história de Harry Potter foi adaptada para uma série de filmes de sucesso,

peças de teatro, jogos e uma vasta gama de produtos. Tornou-se um significativo fenÃ´meno cultural e uma das séries de livros mais vendidas da história.

O arquivo **characters.csv** contém um conjunto de dados de personagens da série extraídos do site https://www.kaggle.com/. Essa base contém registros de 405 personagens. 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 *Personagem* seguindo todas as regras apresentadas no slide unidade00l_conceitosBasicos_introducaoOO.pdf. Sua classe terá os atributos privados `id` (String), `name` (String), `alternate_names` (Lista)[1], `house` (String), `ancestry` (String), `species` (String), `patronus` (String), `hogwartsStaff` (Boolean), `hogwartsStudent` (String), `actorName` (String), `alive` (Boolean), `dateOfBirth` (DateTime), `yearOfBirth` (int), `eyeColour` (String), `gender` (String), `hairColour` (String), `wizard` (Boolean). 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 0 (zero) ou vazio.

   A entrada padrão é composta por várias linhas e cada uma contém uma string indicando o id do Personagem 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 *Personagem* 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 **name**. 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

[1] Aqui você pode usar a sua classe Lista ou algum Collection nativo da linguagem.

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 **name**. 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 **dateOfBirth**. 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 Personagem)

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 **eyeColour**. 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 **hairColour**. 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 **house**. 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 **yearOfBirth**. 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 **hairColour**. 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 **actorName**. 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.