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

Pokémon é uma franquia de mídia criada pela Nintendo, Game Freak e Creatures, que começou como uma série de jogos de RPG para o Game Boy em 1996. Criada por Satoshi Tajiri, a série Pokémon rapidamente se tornou um fenômeno global, evoluindo para uma das franquias mais populares e lucrativas do mundo.

A premissa básica dos jogos gira em torno dos Pokémon (Pocket Monsters), criaturas fictícias que os jogadores, conhecidos como Treinadores, capturam e treinam para lutar contra outros Pokémon. Cada jogo geralmente começa com o jogador recebendo seu primeiro Pokémon de um professor e então viajando por várias regiões para capturar novos Pokémon, desafiar líderes de ginásio, e eventualmente competir na Liga Pokémon.

Um elemento central dos jogos é a Pokédex, uma enciclopédia eletrônica que registra informações sobre todos os Pokémon encontrados ou capturados. A Pokédex fornece detalhes como espécie, altura, peso, e uma descrição única para cada Pokémon. O objetivo de completar a Pokédex, capturando todos os Pokémon disponíveis, é uma das principais motivações dos treinadores.

Os Pokémon possuem individualidades marcantes, como tipos e habilidades específicas. Existem vários tipos, como Fogo, Água, Planta, Elétrico, Psíquico, Dragão, entre outros. Esses tipos determinam as fraquezas e resistências de um Pokémon em batalhas, criando uma dinâmica estratégica.

Por exemplo, um POKÉMON do tipo Fogo é forte contra POKÉMON do tipo Planta, mas fraco contra POKÉMON do tipo Água.

Além dos tipos, cada POKÉMON tem habilidades especiais que conferem vantagens em batalha. Por exemplo, a habilidade "Levitate" torna um POKÉMON imune a ataques do tipo Terra, enquanto "Intimidate" reduz o poder de ataque do oponente ao início de uma batalha.

Dentro do universo POKÉMON, também existem os POKÉMON lendários, que são raros, poderosos e fundamentais para a história dos jogos. Esses POKÉMON são únicos, com habilidades e estatísticas superiores em comparação com os POKÉMON comuns. Exemplos de POKÉMON lendários incluem Mewtwo, Zapdos, e Rayquaza, cada um com seu próprio lore e importância dentro da série.

POKÉMON se tornou um fenômeno cultural, com impacto significativo em várias gerações. Os jogos não apenas entretêm, mas também promovem valores como amizade, estratégia e a importância de cuidar de outras criaturas. A série de jogos POKÉMON continua a crescer, lançando novos títulos regularmente e mantendo sua relevância na cultura pop global.

O arquivo POKEMON.CSV contém um conjunto de dados dos pokémons do jogo extraídos do site Kaggle. Essa base contém 801 pokémons. 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 POKÉMON seguindo todas as regras apresentadas no slide unidade00l_conceitosBasicos_introducaoOO.pdf. Sua classe terá os atributos privado `id` (int), `generation` (int), `name` (String), `description` (String), `types` (Lista)[1], `abilities` (Lista), `weight` (double), `height` (double), `captureRate` (int), `isLegendary` (boolean), `captureDate` (Date). 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 um número inteiro indicando o *id* do POKÉMON 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, no seguinte formato: `[#id -> name: description - [types] - [abilities] - weight - height - captureRate - isLegendary - generation] - captureDate]`.

   Exemplo: `[#181 -> Ampharos: Light Pokémon - ['electric'] - ['Static', 'Plus'] - 61.5kg - 1.4m - 45% - false - 2 gen] - 25/05/1999`

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

2. **Registro em C**: Repita a anterior criando o registro POKÉMON 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 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 **captureDate**. 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 Pokémon)

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 **weight**. O nome do arquivo de log será matrícula_shellsort.txt. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

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 **height**. O nome do arquivo de log será matrícula_heapsort.txt. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

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 **generation**. O nome do arquivo de log será matrícula_quicksort.txt.

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

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 **captureRate**. O nome do arquivo de log será matrícula_countingsort.txt. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

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 **id**. O nome do arquivo de log será matrícula_bolha.txt. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

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 **types**. O nome do arquivo de log será matrícula_mergesort.txt. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

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 **abilities**. O nome do arquivo de log será matrícula_radixsort.txt. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

15. **Ordenação PARCIAL por Seleção em Java**: Refaça a Questão "Ordenação por Seleção" considerando a ordenação parcial com k igual a 10. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

16. **Ordenação PARCIAL por Inserção em C**: Refaça a Questão "Ordenação por Inserção" considerando a ordenação parcial com k igual a 10. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

17. **Heapsort PARCIAL em C**: Refaça a Questão "Heapsort" considerando a ordenação parcial com k igual a 10. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)

18. **Quicksort PARCIAL em Java**: Refaça a Questão "Quicksort" considerando a ordenação parcial com k igual a 10. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o nome do Pokémon)