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

Trabalho Prático III

Regras Básicas

1. extends Trabalho Prático 02
2. Fique atento ao Charset dos arquivos de entrada e saída.
3. **Nos exercícios de ordenação ou estruturas de dados, se dois objetos tiverem a mesma chave de pesquisa, eles serão ordenados pelo nome da série.**.

Você foi contratado para trabalhar em uma empresa que distribui *stream* de séries de TV na web. Sua tarefa é organizar as informações das séries disponíveis para exibição ao usuário. Entretanto, esses dados estão espalhados em vários arquivos no formato *html*, os quais foram obtidos através de consultas à base de dados Wikipedia. Todos esses arquivos estão agrupados no arquivo series.zip, e o mesmo deve ser descompactado na pasta /tmp/. [1] Para isso, você deve ler, organizar e armazenar os dados de cada série em memória, utilizando as estruturas de dados em aula (Lista, Pilhas e Filas). Em seguida executar as operações descritas nos arquivos de entrada. Muito cuidado ao realizar o *parser* do texto. Fique atento a descrição dos dados que serão lidos e manipulados pelo seu sistema.

Algoritmos de Ordenação

1. **Ordenação por Seleção em Java**: Na classe Lista, implemente o algoritmo de ordenação por seleção considerando que a chave de pesquisa é o atributo **paísDeOrigem**. A entrada e a saída padrão são iguais às da primeira questão do Trabalho Prático II, contudo, a saída corresponde aos objetos 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

[1] Quando reiniciamos o Linux, ele normalmente apaga os arquivos existentes na pasta /tmp/.

(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'.

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

3. **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 **Idioma**. O nome do arquivo de log será matrícula_insercao.txt.

4. **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 **Idioma**. O nome do arquivo de log será matrícula_shellsort.txt.

5. **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 **Formato**. O nome do arquivo de log será matrícula_heapsort.txt.

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

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

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

9. **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 **numeroDeEpisodios**. O nome do arquivo de log será matrícula_mergesort.txt.

10. **Radixsort em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Radixsort, fazendo com que a chave de pesquisa seja o valor doidão. O nome do arquivo de log será matrícula_radixsort.txt. O valor doidão é igual a **numeroDeEpisodios * 1000 + numeroDeTemporadas**.

Estrutras Flexíveis

11. **Lista com Alocação Flexível em Java**: Refazer a Questão "Lista com Alocação Sequencial" do Trabalho Prático II usando lista dinâmica simples.

12. **Pilha com Alocação Flexível em Java**: Refazer a Questão "Pilha com Alocação Sequencial" do Trabalho Prático II.

13. **Fila Circular com Alocação Flexível em Java**: Refazer a Questão "Fila Circular com Alocação Sequencial" do Trabalho Prático II. Lembre-se que essa fila terá tamanho máximo igual a cinco.

14. **Quicksort com LISTA DINÂMICA DUPLAMENTE ENCADEADA em Java**: Refazer a Questão "Quicksort" com lista duplamente encadeada. O nome do arquivo de log será matrícula_quicksort2.txt.

15. **Pilha com Alocação Flexível em C**: Refaça a questão 12 deste TP na linguagem C.

16. **Quicksort com LISTA DINÂMICA DUPLAMENTE ENCADEADA em C**: Refaça a questão 14 deste TP na linguagem C.

17. **Matriz Dinâmica em Java**: Complete o código da classe matriz dinâmica visto na sala de aula. A primeira tarefa consiste em, no construtor da classe Matriz, dados os números de linha e coluna, fazer as devidas alocações de células. As demais tarefas são as implementações dos métodos Matriz soma(Matriz), Matriz multiplicacao(Matriz), void mostrarDiagonalPrincipal() e void mostrarDiagonalSecundaria(). A entrada padrão é composta por vários casos de teste sendo que o número de casos é um inteiro contido na primeira linha da entrada. Em seguida, temos cada um dos casos de teste. Cada caso é composto por duas matrizes. Para cada caso de teste, temos que suas duas primeiras linhas contêm um número inteiro cada representando os números de linhas e de colunas da primeira matriz, respectivamente. Em seguida, temos os elementos da primeira matriz que estão representados nas próximas l linhas onde l é o número de linhas dessa matriz. Cada uma dessas linhas têm c colunas onde c é o número de colunas dessa matriz. Nas duas linhas seguintes, temos os números de linhas e colunas da segunda matriz do caso de teste. As $l2$ linhas seguintes têm $c2$ colunas contendo os elementos da segunda matriz. $l2$ e $c2$ correspondem aos números de linhas e colunas da segunda matriz do caso de teste, respectivamente. A saída padrão contém várias linhas para cada caso de teste. As duas primeiras linhas de saída de um caso de teste correspondem às diagonais principal e secundaria da primeira matriz, respectivamente. As demais ls linhas de um caso de teste correspondem as linhas matriz obtida pela soma das duas matrizes do caso de teste sendo que essas linhas contêm cs colunas referentes às colunas da matriz de soma. Da mesma forma, as linhas seguintes do caso

teste contêm lm linhas com cm colunas representando os elementos da matriz de multiplicação onde lm e cm são os números de linhas e colunas da matriz de multiplicação.