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

Trabalho Prático IV

Regras Básicas

1. extends Trabalho Prático 03
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.

Árvores

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

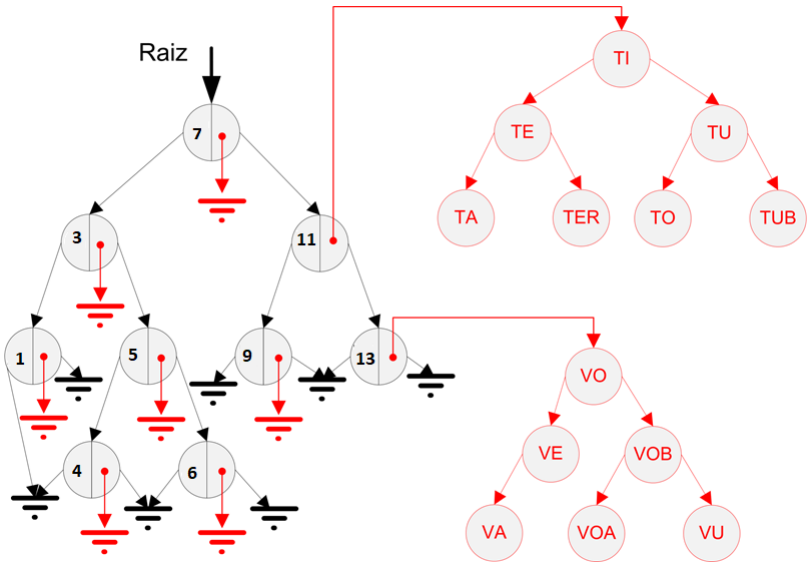
1. **Árvore Binária em Java**: Crie uma Árvore Binária, fazendo inserções de objetos conforme a entrada padrão. A chave de pesquisa é o atributo **nomeSerie**. Não insira um elemento se sua chave estiver na árvore. Em seguida, pesquise se alguns objetos estão cadastrados na Árvore,

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

mostrando seus respectivos caminhos de pesquisa. A entrada padrão contém três partes. As duas primeiras são iguais as duas partes da questão "TP02Q03 - Pilha Sequencial" do Trabalho Prático II, contudo, no caso dos comandos de remoção, após a letra R, temos um valor indicando a chave de um objeto a ser removido. A terceira parte contém vários valores, um por linha, indicando a chave de objetos a serem pesquisados. A saída padrão é composta por várias linhas, uma para cada pesquisa. Cada linha é composta pelo caminho ou sequência de ponteiros (raiz, esq ou dir) utilizados na pesquisa e, no final, pelas palavras SIM ou NÃO. Além disso, crie um arquivo de log na pasta corrente com o nome matrícula_arvoreBinaria.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'.

2. **Árvore Binária de Árvore Binárias em Java**: Crie uma árvore de árvore. Nesse caso, temos uma árvore binária tradicional na qual cada nó tem um ponteiro para outra árvore binária. Graficamente, a primeira árvore está no plano xy e a árvore de seus nós pode ser imaginada no espaço tridimensional. Temos dois tipos de nós. O primeiro tem um caractere como chave, os ponteiros esq e dir (ambos para nós do primeiro tipo) e um ponteiro para nós do segundo tipo. O outro nó tem um string como chave e os ponteiros esq e dir (ambos para nós do segundo tipo). A chave de pesquisa da primeira árvore é o primeiro caractere do atributo **nomeSerie** e, da outra, é o atributo **nomeSerie**. Nesta questão, vamos inserir e remover alguns objetos e, em seguida, pesquisar a existência de alguns **nomeSerie**.

   Destaca-se que nossa pesquisa faz um "mostrar" na primeira árvore e um "pesquisar" na segunda. Faremos um "mostrar" na primeira árvore porque ela é organizada pelo caractere do alfabeto, permitindo que o valor desejado esteja na segunda árvore de qualquer um de seus nós. Faremos o "pesquisar" na segunda porque ela é organizada pelo atributo **nomeSerie**. Antes de inserir qualquer elemento, crie a primeira árvore, inserindo todos seus nós e respeitando a ordem D, R, Z, X, V, B, F, P, U, I, G, E, J, L, H, T, A, W, S, O, M, N, K, C, Y, Q. A entrada e a saída padrão são iguais as da questão anterior e o arquivo de log será matrícula_arvoreArvore.txt. O "mostrar" da **primeira** árvore deve ser o central.



3. **Árvore AVL em C**: Refaça a primeira questão deste trabalho com Árvore AVL. O nome do arquivo de log será matrícula_avl.txt. Não insira um elemento se sua chave estiver na árvore.

4. **Árvore Alvinegra em Java**: Refaça a primeira questão deste trabalho com Árvore Alvinegra. O nome do arquivo de log será matrícula_avinegra.txt. Não insira um elemento se sua chave estiver na árvore. **Não será necessário implementar a opção de remoção.**

5. **Tree sort em Java**: Um tipo de árvore que é um algoritmo de ordenação. Este constrói uma árvore de busca binária dos elementos a serem classificados, em seguida, percorre a árvore (em ordem) para que os elementos sair em ordem de classificação. Efetue a ordenação das séries pelo **nome** usando o algoritmo `Tree sort`. Ao seu arquivo de log, contabilize a quantidade de comparações efetuadas para inserção de todas as séries na árvore. O nome do arquivo de log será matrícula_treesort.txt.

6. **Tabela *Hash* Direta com Reserva em Java**: Refaça a questão "TP02Q03 - Pesquisa Sequencial" do Trabalho Prático II com Tabela *Hash* Direta com Reserva, fazendo com que a função de transformação seja **valorASCIInome mod tamTab** onde tamTab (tamanho da tabela) é 21. A área de reserva tem tamanho 9, fazendo com que o tamanho total da tabela seja igual a 30. A entrada padrão será igual as demais deste trabalho. A saída será a posição de cada elemento procurado na tabela. Se ele estiver na área de reserva considere que o tamanho total da tabela é a área da *hash* mais a de reserva. Caso o elemento procurado não esteja na tabela, escreva a palavra NÃO. Além disso, o nome do arquivo de log será matrícula_hashReserva.txt. **Não será necessário implementar a opção de remoção.**

7. **Tabela *Hash* Direta com Rehash em Java**: Refazer a questão anterior com Tabela *Hash* Direta com *Rehash*, fazendo com que a função de rehash seja **(id mais um) mod tamTab** onde tamTab é 21. A entrada e saída são como na questão anterior. O nome do arquivo de log será matrícula_hashRehash.txt. **Não será necessário implementar a opção de remoção.**

8. **Tabela *Hash* Indireta com Lista Simples em C**: Refazer a questão anterior com Tabela *Hash* Indireta com Lista Simples, fazendo com que a função de transformação seja **valorASCII-nome mod tamTab** onde tamTab é igual a 21 e corresponde ao tamanho da tabela. A entrada e saída são como na questão anterior. O nome do arquivo de log será matrícula_hashIndireta.txt.