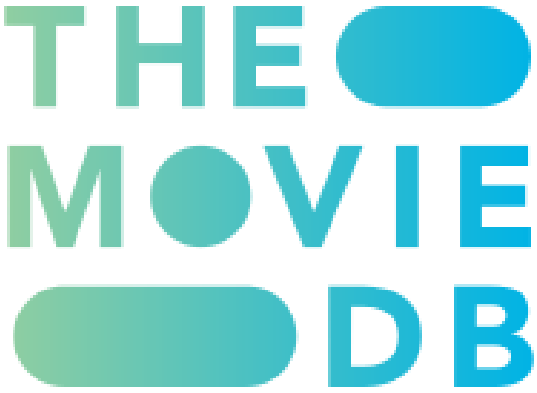
Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

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 do filme.**.



The Movie Database (O Banco de Dados de Filmes), mais conhecido pela sigla TMDb, é uma base de dados grátis e de código aberto sobre filmes e seriados (streaming, televisivas, cinematográficas), criado por Travis Bell em 2008. Atualizado constantemente através do apoio da comunidade. Era inicialmente apenas uma base de dados sobre filmes, mas em 2013 foi adicionado a seção de séries. O TMDb com o código aberto, significa que muitos sites de cinema e TV podem usar os seus dados, incluindo utilizadores individuais de Software de gerenciamento de mídia, como Plex, Kodi e outros.

Neste Trabalho Prático sua tarefa é organizar as informações dos filmes 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 TMDb. Todos esses arquivos estão agrupados no arquivo filmes.zip, e o mesmo deve ser descompactado na pasta /tmp/filmes/. [1] Para isso, você deve ler, organizar e armazenar os dados de cada filme 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.

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

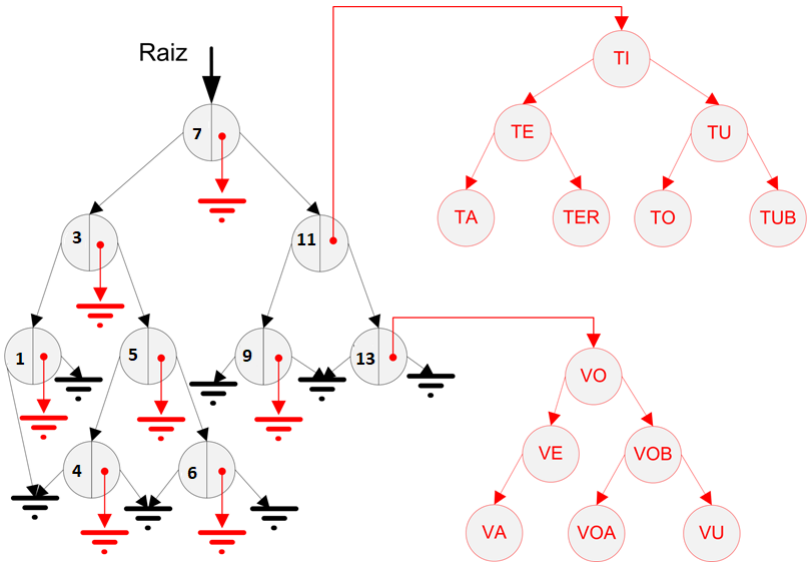
Árvores

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

1. **Árvore Binária**: Crie uma Árvore Binária, fazendo inserções de objetos conforme a entrada padrão. A chave de pesquisa é o atributo **tituloOriginal**. Não insira um elemento se sua chave estiver na árvore. Em seguida, pesquise se alguns objetos estão cadastrados na Árvore, 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**: 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 **tituloOriginal** e, da outra, é o atributo **tituloOriginal**. Nesta questão, vamos inserir e remover alguns objetos e, em seguida, pesquisar a existência de alguns **tituloOriginal**.

   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 **tituloOriginal**. 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**: 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**: 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. **Tabela *Hash* Direta com Reserva**: Refaça a questão "TP02Q03 - Pesquisa Sequencial" do Trabalho Prático II com Tabela *Hash* Direta com Reserva, fazendo com que a função de transformação seja **valorASCIItituloOriginal 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.**

6. **Tabela *Hash* Direta com Rehash**: Refazer a questão anterior com Tabela *Hash* Direta com *Rehash*, fazendo com que a função de rehash seja **(valorASCIItituloOriginal mod tamTab) 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.**

7. **Tabela *Hash* Indireta com Lista Simples**: Refazer a questão anterior com Tabela *Hash* Indireta com Lista Simples, fazendo com que a função de transformação seja **valorASCIItituloOriginal 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.

8. **Árvore Binária de Busca**

Em computação, a árvores binária de busca ou árvore binária de pesquisa é uma estrutura baseada em nós (nodos), onde todos os nós da subárvore esquerda possuem um valor numérico inferior ao nó raiz e todos os nós da subárvore direita possuem um valor superior ao nó raiz (e assim sucessivamente). O objetivo desta árvore é estruturar os dados de forma flexível, permitindo a busca binária de um elemento qualquer da árvore.

A grande vantagem das árvores de busca binária sobre estruturas de dados convencionais é que os algoritmos de ordenação (percurso infixo) e pesquisa que as utilizam são muito eficientes.

Para este problema, você receberá vários conjuntos de números e a partir de cada um dos conjuntos, deverá construir uma árvore binária de busca. Por exemplo, a sequência de valores: 8 3 10 14 6 4 13 7 1 resulta na seguinte árvore binária de busca:

Entrada

A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro C ($C \leq 1000$), indicando o número de casos de teste que virão a seguir. Cada caso de teste é composto por 2 linhas. A primeira linha contém um inteiro N ($1 \leq N \leq 500$) que indica a quantidade de números que deve compor cada árvore e a segunda linha contém N inteiros distintos e não negativos, separados por um espaço em branco.

Saída

Cada linha de entrada produz 3 linhas de saída. Após construir a árvore binária de busca com os elementos de entrada, você deverá imprimir a mensagem "Case n:", onde n indica o número do caso de teste e fazer os três percursos da árvore: prefixo, infixo e posfixo, apresentando cada um deles em uma linha com uma mensagem correspondente conforme o exemplo abaixo, separando cada um dos elementos por um espaço em branco.

Obs: Não deve haver espaço em branco após o último item de cada linha e há uma linha em branco após cada caso de teste, inclusive após o último.

Exemplo de Entrada	Exemplo de Saída
2 3 5 2 7 9 8 3 10 14 6 4 13 7 1	Case 1: Pre.: 5 2 7 In..: 2 5 7 Post: 2 7 5 Case 2: Pre.: 8 3 1 6 4 7 10 14 13 In..: 1 3 4 6 7 8 10 13 14 Post: 1 4 7 6 3 13 14 10 8

9. **Operações em ABP I**

Marcela recebeu como trabalho de Algoritmos a tarefa de fazer um programa que implemente uma Árvore Binária de Pesquisa (ou Busca). O Programa deve aceitar os seguintes comandos:

- I N: Insere na árvore binária de pesquisa o elemento n.
- INFIXA: lista os elementos já cadastrado segundo o percurso infixo
- PREFIXA: lista os elementos já cadastrado segundo o percurso prefixo
- POSFIXA: lista os elementos já cadastrado segundo o percurso posfixo
- P N: pesquisa se o elemento n existe ou não.

A qualquer momento pode-se inserir um elemento, visitar os elementos previamente inseridos na ordem infixa, prefixa ou posfixa ou ainda procurar por um elemento na árvore para saber se o elemento existe ou não.

Entrada

A entrada contém N operações utilizando letras (A-Z,a-z) sobre uma árvore binária de Busca, que inicialmente se encontra vazia. A primeira linha de entrada contém a inserção de algum elemento. As demais linhas de entrada podem conter quaisquer um dos comandos descritos acima, conforme exemplo abaixo. O final da entrada é determinado pelo final de arquivo (EOF).

Obs: Considere que não serão inseridos elementos repetidos na árvore.

Saida

Cada linha de entrada, com exceção das linhas que contém o comando "I", deve produzir uma linha de saída. A saída deve ser de acordo com o exemplo fornecido abaixo. Não deve haver espaço em branco após o último caractere de cada linha, caso contrário, sua submissão receberá *Presentation Error*.

Exemplo de Entrada	Exemplo de Saída
I c	a c f h
I f	c a f h
I a	a h f c
I h	z nao existe
INFIXA	h existe
PREFIXA	a c f g h
POSFIXA	
P z	
P h	
I g	
INFIXA	