Universidade Federal da Bahia
MATD04 - Estruturas de Dados
Segundo Trabalho Prático
Prof. Flávio Assis

Semestre 2023.2 - 8 de novembro de 2023

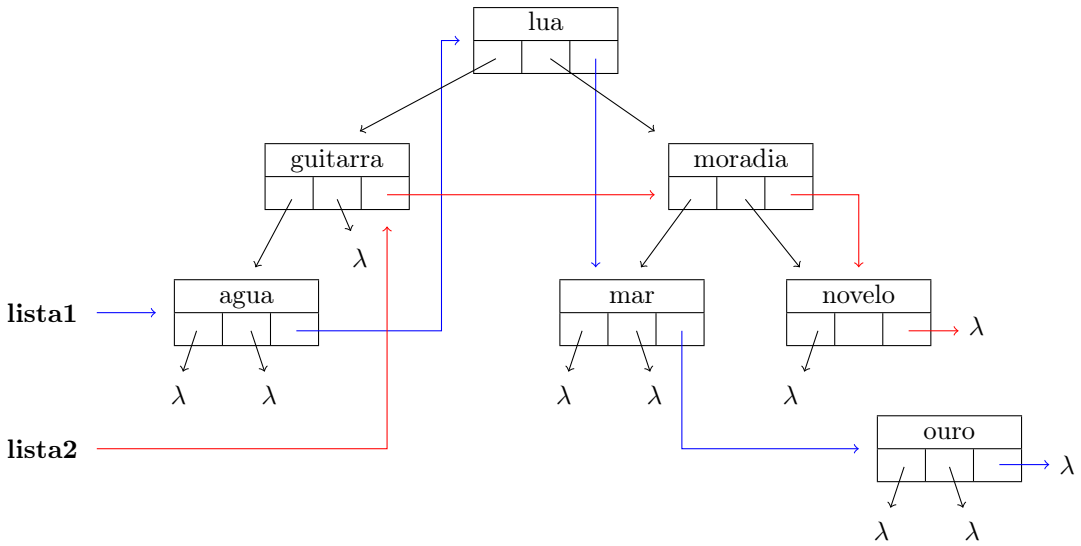
Árvore Binária de Busca com Palavras

# 1 Descrição Geral do Trabalho

Neste trabalho deverá ser implementado um programa que armazena palavras em uma árvore binária de busca e permite que operações sobre essa árvore sejam feitas. Nós dessa árvore estarão unidos em duas listas, que ligarão palavras com base no seu número de caracteres. O programa terá como entrada um conjunto de comandos a serem executados, como descrito na próxima seção. O programa deve ler da entrada padrão e escrever os resultados na saída padrão.

Cada nó da árvore armazenará uma palavra, que será a sua *chave*. Os nós da árvore serão armazenados de acordo com a ordem alfabética das palavras. As duas listas serão: uma, que unirá palavras com até 5 letras; e outra, que unirá palavras com 6 ou mais letras. Cada uma dessas listas deve ser implementada como uma *lista simplesmente encadeada* e deverá manter as palavras em ordem alfabética crescente (de A para Z). De forma análoga ao primeiro trabalho, estas listas ***não poderão ser implementadas de maneira separada dos nós da árvore!*** Ou seja, as listas unirão nós da árvore.

A figura abaixo ilustra um exemplo. As três listas básicas são: *lista1* (palavras com até cinco letras); e *lista2* (palavras com seis letras ou mais). Observe que cada nó terá, além da chave, dois apontadores para seus filhos e um apontador para referenciar o próximo elemento da lista à qual a palavra (chave do nó) pertence.



## 2 Formato da Entrada e Saída

A entrada constará de uma sequência de comandos. Nos comandos, uma palavra será formada apenas por letras minúsculas, sem acento e sem cedilha. Uma palavra não terá mais do que 29 caracteres.

O formato de cada comando está descrito a seguir.

1. **insere palavra:** este comando consistirá de duas linhas. A primeira conterá a letra 'i'. A segunda linha conterá uma palavra.

   Se a palavra já estiver na árvore, o programa deve gerar na saída: *'palavra ja existente:'*, seguida de um espaço, seguido da palavra digitada. Se não estiver, o programa deve inserir a palavra na árvore e na lista correspondente, considerando o número de caracteres que possuir. Em seguida, o programa deve gerar na saída a sequência: *'palavra inserida:'*, seguida de um espaço, seguido da palavra.

2. **consulta palavra:** este comando consistirá de duas linhas. A primeira conterá a letra 'c'. A segunda linha conterá uma palavra.

   Se a palavra estiver na árvore, o programa deve gerar na saída: *'palavra ja existente:'*, seguida de um espaço, seguido da palavra digitada. Se não estiver, o programa deve gerar na saída *'palavra inexistente:'*, seguida de um espaço, seguido da palavra.

3. **lista palavras em lista:** este comando consistirá de uma linha contendo a letra 'l', seguida por uma linha contendo um número. O comando listará as palavras de uma das listas, uma por linha, em ordem alfabética, de acordo com o número da segunda linha: se for 1, lista as palavras com cinco letras ou menos; se for 2, lista as palavras com seis ou mais letras.

   Se não houver palavras que atendam ao comando, o programa deve gerar na saída a sequência: *'lista vazia'*.

4. **lista palavras por número de letras:** este comando consistirá de uma linha contendo a letra 'x', seguida por uma linha contendo um número (qualquer valor inteiro positivo). O comando lista as palavras armazenas nas listas que possuem exatamente o número de letras indicado na segunda linha do comando.

   Este comando apresenta na saída uma palavra por linha, em ordem alfabética. Caso não haja palavras com o número de letras dado, o programa gera na saída a sequência *'lista vazia'*.

5. **lista palavras em ordem alfabética:** este comando será formado por três linhas. Na primeira linha ocorrerá a letra 'o'. Na segunda e na terceira linha ocorrerá, em cada uma, uma letra. Sejam $l_1$ e $l_2$, respectivamente, as letras que ocorrem na segunda e na terceira linha. O programa deve retornar na saída as palavras que começam com as letras que vão de $l_1$ a $l_2$ no alfabeto (incluindo $l_1$ e $l_2$), em ordem alfabética, uma por linha. Caso não haja palavras a serem listadas, o programa deve gerar na saída a sequência: *'lista vazia'*.

   Considere que $l_1$ é igual a ou vem antes de $l_2$ no alfabeto.

6. **remove palavra:** este comando consistirá de uma linha contendo a letra 'r', seguida por uma segunda linha, contendo uma palavra.

   Se não houver a palavra armazenada na árvore, o programa deve gerar na saída a sequência: *'palavra inexistente:'*, seguida de um espaço, seguido da palavra. Se houver, o nó contendo a palavra deve ser removido. Em seguida, o programa deve gerar na saída a sequência: *'palavra removida:'*, seguida de um espaço, seguido da palavra.

7. **imprime árvore:** este comando consiste apenas de uma linha, contendo a letra 'p'. O formato de impressão da árvore está descrito na seção 3.

8. **término da seqüência de comandos:** a seqüência de comandos será terminada por uma linha com a letra 'e'.

# 3 Impressão da Árvore de Busca Binária

Esta operação listará os valores referentes a cada nó, um por linha, seguindo a ordem de visita de um percurso em *pré-ordem* da árvore.

Para cada nó, os dados devem ser apresentados no seguinte formato: a sequência de caracteres *'palavra:'*, seguida de um espaço, seguido da palavra (chave) armazenada no nó, seguido de um espaço, seguido pela sequência de caracteres *'fesq:'*, seguida por um espaço, seguido pelo valor da palavra (chave) armazenada no filho à esquerda, seguido por um espaço, seguido pela sequência de caracteres *'fdir:'*, seguida por um espaço, seguido da palavra (chave) armazenada no filho à direita. Quando não houver um filho (à esquerda ou direita), a sequência de caracteres *nil* deve ser gerada após a sequência *'fesq:'* ou *'fdir:'* correspondente.

Se a árvore estiver vazia, o programa deve mostrar na saída a sequência *'arvore vazia'*.

# 4 Observações

- A implementação terá que, necessariamente, usar a estrutura de árvore descrita acima.
- Os comandos devem gerar as saídas **somente** através do acesso à estrutura de árvore e listas descrita acima. Portanto, não poderão ser usadas listas de Python, nem métodos de ordenação existentes. **O uso desses mecanismos será considerado erro de programação (não atendimento à especificação) e, portanto, o trabalho será parcial ou totalmente desconsiderado.**
- Somente podem ser usados recursos da própria linguagem. Recursos adicionais somente podem ser utilizados se autorizados pelo professor. Em particular, nenhum recurso que não seja de biblioteca padrão da linguagem poderá ser usado.
- O programa não deve gerar nenhum caractere a mais na saída, além dos indicados acima. Em particular, **o programa não deve conter menus**.
- Não pode haver espaço entre linhas na saída. A saída deve apresentar os caracteres em letras minúsculas e sem acento.
- Trabalho individual.
- Linguagens de programação permitida: Python. A versão utilizada de Python deve ser indicada.
- Data de entrega: **01/12/2023 - prazo inadiável! Somente serão aceitos trabalhos entregues até essa data, até 23:59h!**
- O(A) aluno(a) deverá submeter seu trabalho através do *moodle*.