

Tries

1 Descrição Geral do Trabalho

Neste trabalho deverá ser implementada uma *trie*. O programa a ser implementado permitirá a inserção de palavras na *trie* e a execução de operações sobre essa estrutura.

As palavras serão armazenadas em uma *árvore k-ária*, em que cada nó armazena um caractere e referências (apontadores) para os seus nós filhos, como apresentado em sala de aula. **Considere que cada palavra possui um caractere final especial, que indica o término da palavra.** Informação sobre implementação de *tries* pode ser encontrada em A. L. Tharp. "File Organization and Processing". John Wiley & Sons. 1988, disponível no sistema de bibliotecas da UFBA. Na implementação nesse trabalho, **a estrutura deve também armazenar quantas vezes cada palavra foi consultada.**

Cada palavra será formada apenas por letras (incluindo *k*, *y* e *w*), minúsculas, sem acento e sem cedilha.

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.

2 Formato da Entrada e Saída

A entrada será formada por uma sequência de operações, terminada pela operação de *término de sequência*. As operações que devem ser suportadas pelo programa são:

1. **insere palavra:** esta operação terá duas linhas. Na primeira linha haverá a letra 'p'. Na segunda linha haverá uma palavra.

Se a palavra já estiver na *trie*, 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 *trie* e gerar na saída a sequência: *'palavra inserida:'*, seguida de um espaço, seguido da palavra.

Este comando não altera o contador de nenhum dos nós da árvore.

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 *trie*, esse comando incrementa o contador do número de vezes em que a palavra foi consultada e gera na saída a sequência *'palavra existente:'*, seguida de um espaço, seguido da palavra digitada, seguida de um espaço, seguido do número de vezes em que a palavra foi consultada até então (incluindo essa operação). **Se não estiver,** o programa deve gerar na saída *'palavra inexistente:'*, seguida de um espaço, seguido da palavra.

Este comando altera, se for o caso, apenas o contador correspondente à palavra consultada. Ele **não altera** o contador de outros nós.

3. **palavra(s) mais consultada(s):** este comando consistirá de uma única linha contendo a letra 'f'.

Se a *trie* estiver vazia, este comando gera na saída a sequência: *'trie vazia'*. **Se não estiver,** o comando gera na saída a sequência *'palavras mais consultadas:'*, seguida, a partir da próxima

linha, das palavras que tiveram o maior número de acessos. As palavras devem ser impressas uma por linha. Ao final, o comando deve gerar, em uma linha adicional, a sequência '*numero de acessos:*' seguida de um espaço, seguido do número de vezes em que as palavras foram consultadas. As palavras devem ser listadas em ordem alfabética.

Este comando **não altera** o contador de nenhum dos nós da árvore.

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

Este comando **não altera** o contador de acesso de nenhuma das palavras.

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

3 Impressão da *Trie*

Esta operação **listará os valores referentes a cada nó, um por linha**, seguindo a ordem de visita de um percurso em *pré-ordem* (estendido) da árvore, considerando, para cada nó, os nós filhos em ordem alfabética.

Para cada nó, os dados devem ser apresentados no seguinte formato:

1. se o nó contiver uma letra, deve ser gerada na saída a sequência de caracteres '*letra:*', seguida de um espaço, seguido do caractere armazenado no nó, seguido de um espaço, seguido do caractere ('-'), seguido de um espaço, seguido da sequência de caracteres correspondentes aos nós filhos, separados por espaço, em ordem alfabética. **Se um dos filhos for um nó contendo o símbolo de fim de palavra, deve ser impresso um asterisco ('*')** no local correspondente ao caractere.
2. se o nó contiver um símbolo de fim de palavra, deve ser gerada na saída apenas a sequência de caracteres '*letra: **'.
3. se for o nó raiz da *trie*, a saída deve ser igual ao caso da letra (a) acima, com exceção de que a sequência de caracteres *raiz* deve ser gerada no local do caractere.

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

4 Observações

- A implementação terá que, necessariamente, usar a estrutura de árvore descrita acima.
- Somente podem ser usados recursos da própria linguagem. 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.
- Linguagem de programação permitida: somente Python. A versão utilizada de Python deve ser indicada.
- O programa pode ser organizado em diversos arquivos. No entanto, o programa principal deve estar em um arquivo de nome **trab01.py**.

- O(A) aluno(a) deverá submeter seu trabalho através do *moodle*, em um único arquivo *".zip"*, de nome **matd04-trab01.zip**, mesmo que o trabalho tenha sido feito na forma de um único arquivo Python.
- Data de entrega: **29/12/2024 - prazo inadiável! Somente serão aceitos trabalhos entregues até essa data, até 23:59h!**