INF1010 – ESTRUTURAS DE DADOS AVANÇADAS
TRABALHO 2 – SUGESTÃO DE SUFIXOS - DICIONÁRIO

Descrição:

O Trabalho 2 consiste em implementar uma árvore de prefixo (Trie) que pode ser utilizada para sugerir palavras de um dicionário. A implementação é semelhante a utilizada em programas editores de texto, ambientes de programação, sites de busca ou sugestões de palavras na digitação em celulares. Também pode ser utilizado em processadores de linguagem natural ou na identificação de substrings em uma string.

Há um projeto criado com o mesmo código apresentado nos slides e contém um programa de teste. O programa pode ser modificado para incluir outras palavras e já inclui as seguintes: {Bola, Bolao, Careca, Carro, Carroca, Carruagem, Homen[1], Homenagem, Joana, Joao}.

Para ajuda na implementação:

- Consulte a explicação de árvores de expressão (slides)

- Se preferir, utilize o ambiente no site https://www.onlinegdb.com/
    - Neste caso, envie o zip do seu projeto para a entrega do trabalho e indique o link para o fork - isso facilita a execução do projeto para correção.

Projeto fornecido contém:

- compile.sh (apenas para usuários Linux) – Script em bash para auxiliar na compilação usando o GCC

- main.c – Implementa um programa teste com o dicionário previamente fornecido

- trie.h – Interface para Árvores Trie

- trie.c – Implementação das funções

    - **Em trie.c devem ser implementadas as funções:**

        - Trie *criaNo(char v) – Cria nó da árvore

        - void inserePalavra(Trie *t, char *palavra) – Insere palavra

        - int buscarPalavra(Trie *t, char *palavra) – Busca por palavra (completa)

[1] Sim, homem está errado propositalmente.

- Trie* buscarPrefixo(Trie *t, char *palavra) – Busca por prefixo (palavra completa ou parte inicial)

- void removerPalavra(Trie *t, char *palavra) – Remove palavra Deve-se tomar cuidado para não remover outras palavras que compartilhem o mesmo prefixo.

- **Em trie.c as seguintes funções já estão implementadas:**

  - Trie *criaTrie() – Cria a árvore Trie

  - void alphabetize(Trie * t) – mostra todas as possibilidades a partir do nó **t** (que pode ser a raiz ou um nó intermediário)

  - void alphabetize2(Trie * t, char prefixo[]) – mesmo que alphabetize porém adiciona a string fornecida no parâmetro "prefixo" antes da sequências de **t**

  - void liberar(Trie *t) – libera a árvore trie da memória

Forma de Avaliação:

Será avaliado se:

(1) O trabalho atendeu a todos os requisitos especificados anteriormente;

(2) Os algoritmos foram implementados e aplicados de forma correta;

(3) O código foi devidamente organizado;

(4) Os resultados estão corretos;

- O resultado esperado é mostrado abaixo. Este resultado foi obtido utilizando o programa teste incluso no projeto compactado.

Resultado Esperado:

```
Programa Teste Árvore Trie
--------------------------------
Dicionário Trie:
bola
bolao
careca
carro
carroca
carruagem
homen
homenagem
joana
joao
--------------------------------
Buscar palavra: Carro
palavra 'Carro' encontrada.
--------------------------------
Buscar palavras que começam com: 'Ca'
..reca
..rro
..rroca
..rruagem
--------------------------------
Dicionário Trie Após Remoção:
bola
bolao
careca
carro
carruagem
homenagem

Buscar palavra: Carro
palavra 'Carro' encontrada.

Buscar palavra: Carroca
não encontrei a palavra 'Carroca'.
```