2º Trabalho Prático: GBC034 - Algoritmos e Estruturas de Dados 2

Instruções

Número de integrantes por grupo 3 pessoas.
Grupos com mais ou menos pessoas somente serão aceitos se não houver número suficiente de alunos para formar outro grupo.

Data de Entrega:

11/06/2020

Forma de Entrega:

O trabalho deverá se entregue por e-mail. No campo Assunto do e-mail, colocar: **Trabalho GBC034**

No arquivo principal (**main**) do programa deve haver o seguinte comentário com os nomes dos integrantes do grupo:

```
/*
Integrantes:
 Fulano de Tal - matrícula
 Ciclano de Tal - matrícula
 Beltrano de Tal - matrícula
*/
```

Problema

Dada a implementação da árvore binária vista em aula, implemente uma **árvore de prefixos**, também chamada ***trie***, para armazenar strings.

- Ao contrário de uma árvore de busca binária, nenhum nó nessa árvore armazena a chave associada a ele; ao invés disso, ela é determinada pela sua posição na árvore.

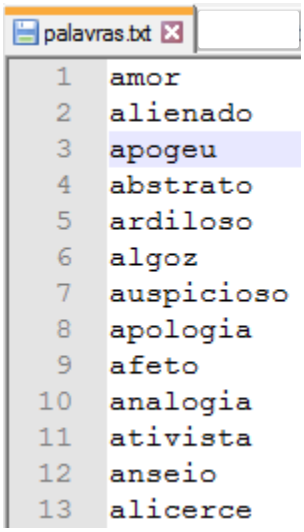
- Todos os descendentes de qualquer nó têm um prefixo comum com a cadeia associada com aquele nó, e a raiz é associada com a cadeia vazia. Normalmente, valores não são associados com todos os nós, apenas com as folhas e alguns nós internos que correspondem a chaves de interesse. [Fonte: wikipedia]

A implementação da **árvore de prefixos** deve seguir o seguinte protótipo de funções:

```
Trie* criaTrie();
void liberaTrie(Trie* tr);
int insereTrie(Trie* tr, char *str);
int buscaTrie(Trie* tr, char *str);
```

```
int removeTrie(Trie* tr, char *str);
//imprime todas as palavras armazenadas
void imprimeTrie(Trie* tr);
//imprime todas as palavras que começam com "prefixo"
void autocompletarTrie(Trie* tr, char *prefixo);
```

Em seguida, escreva um programa para ler um arquivo contendo uma lista de palavras. Cada linha do arquivo representa uma palavra que deve ser inserida na **árvore de prefixos**.



Instruções:

O trabalho será avaliado principalmente levando em consideração:

1) Realização das tarefas do trabalho.
2) Representação correta da entrada e saída dos dados.
3) Uso correto das variáveis e estruturas de dados.
4) Uso adequado dos conceitos aprendidos em sala (modularização, hash, árvores, etc, quando for o caso).
5) Boa identação e uso de comentários no código. Evite utilizar comentários excessivamente.

Observações:

- O professor em hipótese alguma irá verificar ou ajudará na construção do código.
- O professor poderá tirar dúvidas sobre o enunciado do problema em horário de aula ou por email.
- A interpretação do problema e a construção da solução fazem parte da avaliação e deverão ser resolvidos pelo aluno.