Pontifícia Universidade Católica do Rio Grande do Sul

Escola Politécnica

Algoritmos e Estrutura de Dados I | 4645G-04

Prof. Iaçanã Ianiski Weber

Árvore de Palavras - Trabalho III

Elaborado pela Professora Isabel H. Manssour

30 de maio de 2023

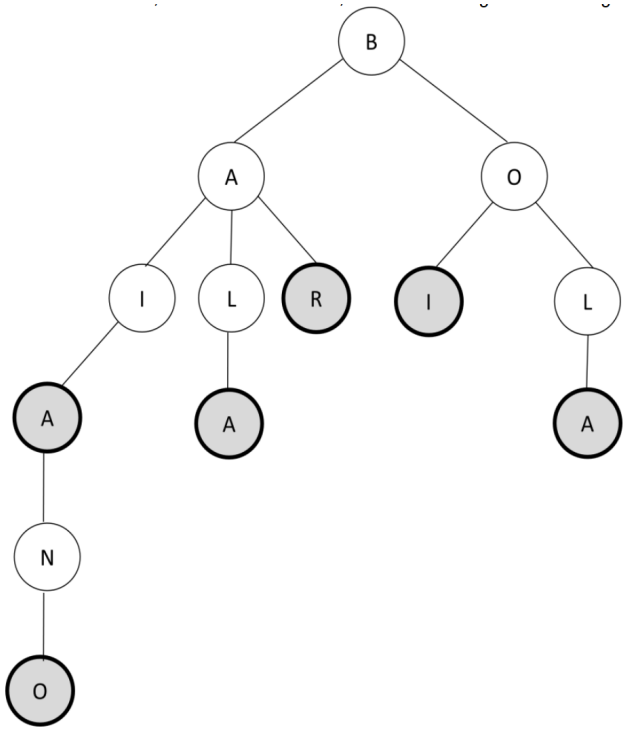
Deadline: 26/06/2023 - 23:59 - Moodle

1 Instruções

Neste trabalho, que pode ser desenvolvido individualmente ou em grupo de dois alunos, vocês desenvolverão uma solução para gerenciar uma árvore de palavras para um dicionário. Para isso, deverão ler um arquivo de palavras com seus respectivos significados, armazenando-os em uma árvore de palavras. Depois, ao inserir dois ou três caracteres, o programa deverá exibir todas as palavras que começam com esses caracteres.

Detalhes

As palavras devem ser armazenadas, caractere a caractere, em uma árvore genérica da seguinte maneira:



Deve haver uma maneira de indicar que um nodo é o último caractere que forma uma palavra e que conterá o significado da palavra. Assim, as palavras são compostas pelos caracteres armazenados nos

nodos visitados da raiz até o último caractere que forma a palavra. Para possibilitar uma consulta ao dicionário, a entrada deve ser um conjunto de caracteres, e a saída deve ser as possíveis palavras que podem ser formadas a partir deste conjunto de caracteres. Considerando a árvore exemplificada acima, se fosse digitado `ba`, a lista de palavras de saída seria: `baia, baiano, bala, bar`. A partir disso, é escolhida uma destas palavras para apresentar o seu significado.

Estrutura de Dados

A estrutura de dados a ser desenvolvida **obrigatoriamente** deve ser baseada em uma árvore genérica. Sugere-se ter como referência a implementação de árvore genérica estudada em aula. Neste caso, a classe `Node` deve armazenar um caractere e sempre que for o último caractere que forma uma palavra, deve-se armazenar também o seu significado (por exemplo, pode ser incluído um atributo `String significado`, se for `null` é porque não é o último caractere que forma a palavra). Além disso, os seus métodos devem ser alterados para darem suporte às funcionalidades necessárias. O código disponibilizado com o enunciado é um exemplo de como pode ser construída esta estrutura.

Aplicação

Para usar e testar a estrutura de dados desenvolvida deverá ser feita uma aplicação para um dicionário. O funcionamento deve ser o seguinte:

- É fornecido um conjunto de caracteres;
- Após a pesquisa na árvore é retornada uma lista de palavras que iniciam com os caracteres fornecidos;
- É escolhida uma palavra desta lista;
- É apresentado o significado desta palavra.

Formato do Arquivo de Entrada

O arquivo do dicionário deverá ter o seguinte formato:

```
Palavra 1; significado da palavra 1
Palavra 2; significado da palavra 2
Palavra 3; significado da palavra 3
Palavra 4; significado da palavra 4
                  ...
Palavra n; significado da palavra n
```

Importante: este arquivo não pode conter caracteres com acento! Como exemplo, e para testar os algoritmos desenvolvidos, será fornecido junto à especificação do trabalho, um arquivo que contém um conjunto de palavras com os seus significados bem como um código para leitura deste arquivo.

Tarefas

- Implementar a árvore e os algoritmos que permitam armazenar e consultar as informações de um dicionário;
- Implementar um programa que permita testar a estrutura de dados e os algoritmos implementados através das operações da aplicação com o arquivo de exemplo;
- Apresentar a solução proposta para o professor durante a aula do dia agendado (27/06/2023) para entrega do trabalho;

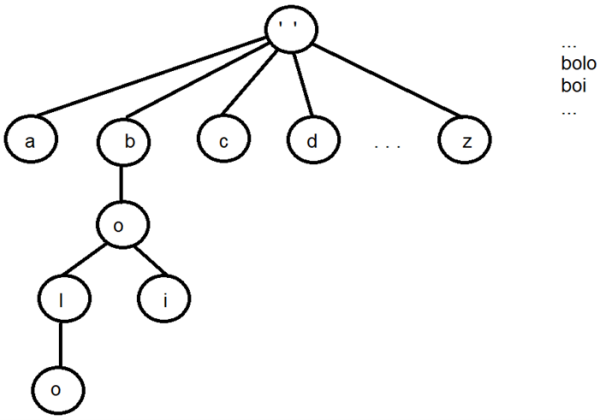
- Entregar no Moodle um arquivo .zip contendo apenas os arquivos do código fonte (somente os .cpp e os headers).

Observações

- Os trabalhos que NÃO FOREM ENTREGUES através do Moodle seguindo as regras estabelecidas, até o dia e horário especificado, não serão considerados!
- Trabalhos que apresentarem erro de compilação não serão considerados;
- Trabalhos que apresentarem cópias das soluções de outros colegas ou de outras fontes resultarão em nota zero para todos os alunos envolvidos;
- Todos os alunos devem apresentar o trabalho e estarem aptos a responder às perguntas sobre os algoritmos implementados.

Dicas

Seguem algumas ideias de procedimentos para inserir, ler e buscar na árvore.



Inserir

- Começa pela raiz (Node aux = root)
- Laço (para cada caractere da palavra)
  - Verifica se tem um filho com o caractere (pesquisa nos filhos de aux)
  - Se não tem, insere (se for "final" coloca o significado)
  - Se já tem, faz apontar para o filho com o caractere (aux = ...)

Ler

- Quando chegar a um nodo "final":
  - Cria uma pilha de caracteres
  - Laço do nodo "final" até a raiz:
    - ▷ Coloca o caractere na pilha

    ▷ Faz apontar para o pai

- Laço para esvaziar a pilha e concatenar os caracteres para formar a palavra.

Buscar

- Vai até o último nodo do prefixo (no exemplo acima, considerando “bo”, vai até o “o” do nível 2)
- Chama um método recursivo de caminhamento
- Este método recebe uma referência para “Node” e uma Lista por parâmetro
  - “Visita a raiz” (verifica se é “final” e se for adiciona a palavra e o significado na lista)
  - “Visita os filhos” (chama o método recursivo para cada filho)