PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
Algoritmos e Estruturas de Dados I
Trabalho 3

Trabalhando com Árvores

Nomes: Lennon Da Silva Rocha
William Antunes Rubert

Data: 04/12/2018

1 - INTRODUÇÃO

O trabalho consiste em desenvolver uma estrutura de arvore que armazena caracteres que formam palavras. O usuário digita uma palavra inicial ou parte de uma palavra e o programa apresenta outras palavras que iniciam com o input do usuário que estão presentes na arvore. A partir disto o usuário seleciona uma destas palavras e o programa lhe apresenta o seu significado.

Para tanto foi utilizada uma estrutura de arvore genérica que armazena os caracteres, e um dicionário que armazena as palavras contidas na arvore e seus significados.

2 - DESENVOLVIMENTO

Para implementarmos a solução do problema inicialmente foi desenvolvida uma classe Dicionario que faz a leitura do arquivo contendo as palavras e seu significo, adicionando as palavras como chave de uma estrutura de dicionário e seus significados como valores.

Foi também implementada uma classe de arvore genérica que recebe elementos de tipo genérico. A implementação desta arvore foi baseada no código desenvolvido em aula para uma arvore genérica de inteiros. A principal diferença implementação desta é arvore foi o método addArray, que recebe como parâmetro um array de objetos (no caso o nosso problema foi utilizado um array de caracteres que formam a palavra que desejamos acrescentar em nossa arvore). O método addArray basicamente itera pelos componentes do array passado como parâmetro e adiciona o caracter na arvore se caso este caracter já não esteja presente na arvore como filho direto do caracter anterior da palavra a ser adicionada.

Desta forma é possível construir uma arvore de caracteres que formam as palavras.

Outra diferença em relação a arvore desenvolvida em aula é que a classe interna nodo possui um outro argumento booleano que marca a fim de uma palavra “interna” em que o fim da palavra não necessariamente é uma folha da arvore.

Também foi implementado o método searchTree que recebe por parâmetro um array de objetos (no nosso caso caracteres) e retorna um array de objetos (palavras) com todas as palavras que podem ser formadas iniciando com os caracteres passado por parâmetro. Esse método utiliza um submetodo recursivo para percorrer todas as subarvores a partir do nodo em que se encontra o caracter final indicado pelo usuário.

A partir disso o usuário digita uma quantidade de caracteres e o método retorna todas as palavras iniciadas com este input. O usuário então seleciona uma dessas palavras que será utilizada como chave do dicionário apresentando o seu significado que está armazenado como valor dessa chave.

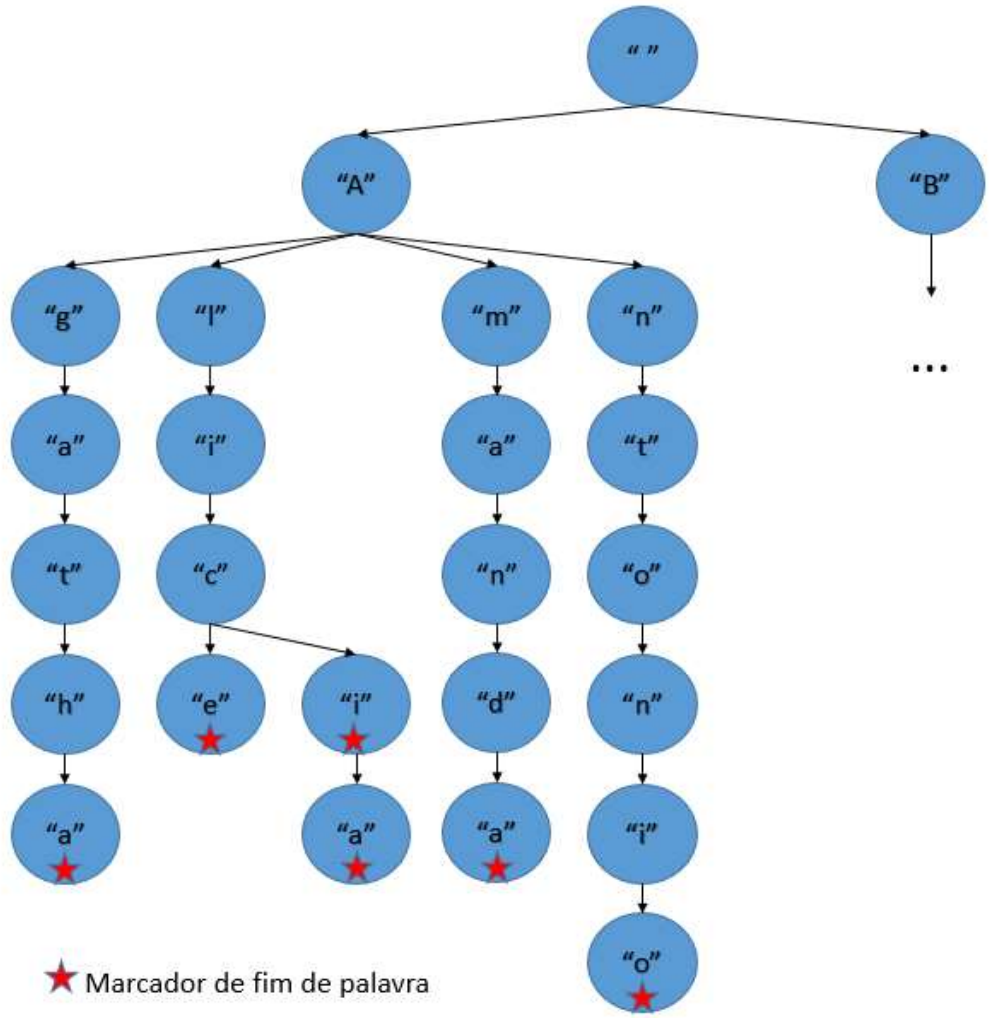

FIGURA 1 – Estrutura da árvore genérica.

FIGURA 2 – Estrutura do dicionário de palavras e significados.

2.1 – APRESENTAÇÃO DOS PRINCIPAIS ALGORITMOS

```
public void addArray(T[] elements)


Para cada caracter da palavra no parâmetro

      Itera por todas as subarvores a partir da raiz e verifica se o caracter
      em questão já se encontra neste nível

            Caso não, adiciona

                  Verifica se é o último caracter da palavra, se sim marca
                  nodo como final de palavra

            Caso sim, passa para o próximo caracter e verifica somente as
            subarvores a partir deste nodo
```

Ordem addArray: ~O(n^2)

```
public ArrayList<T[]> searchTree(T[] elements)

Para cada caracter do array passado por parâmetro
      Percorre arvore a partir da raiz e verifica se é igual ao caracter
            Se for igual, passa para o próximo caracter do array e busca a
            partir do nodo atual
            Se não encontra igual, sai do laço
Retorna um array com as palavras iniciados com os caracteres do array de
parâmetro


private ArrayList<T[]> getMarkedArrays(T[] elements, Node ref)

Cria array de resultados
Percorre todas as subarvores a partir do nodo ref
      Para cada nodo na subarvore verifica se é marcado como fim de palavra
            Se sim, adiciona a palavra ao array resultado
            Verifica se o próxima nodo tem subarvores
                  Se sim, chama método novamente
Retorna array com resultados
```

Ordem searchTree: ~O(n^3)

3 – DIFICULDADES

As principais dificuldades foram no desenvolvimento dos principais métodos.

No addArray encontramos um problema que acabava acrescentando caracteres em posições erradas na arvore, pois apesar de a subarvore no nível imediatamente inferior ao do pai não possuir esse caracter outros níveis inferiores poderiam ter, o problema foi resolvido verificando somente a ocorrência do carecter nos filhos imediatos do nodo.

No caso do searchTree o problema estava na elaboração do método em si pois para uma determinada palavra em um nodo com uma marcação de final poderia ter diversos outros galhos que seguiam para outras palavras. O problema foi resolvido criando um submetodo recursivo que devolve todas as palavras finalizadas a partir de determinado nodo.

4 – RESULTADOS

Exemplos de resultados obtidos pelo programa desenvolvido:

Ex 1)

```
Digite a palavra a ser pesquisada:
Hel
Palavras encontradas:
Helena
Heloisa

Digite uma palavra da lista para ver significado:
Helena
Significa a reluzente, a resplandecente.
```

Ex 2)

```
Digite a palavra a ser pesquisada:
Al
Palavras encontradas:
Alice
Alicia

Digite uma palavra da lista para ver significado:
Alice
Significa de qualidade nobre, de linhagem nobre.
```

Ex 3)

```
Digite a palavra a ser pesquisada:
C
Palavras encontradas:
Caio
Calebe
Carolina
Catarina
Caua
Cecilia
Clara

Digite uma palavra da lista para ver significado:
Carolina
Significa mulher do povo, mulher doce.
```

Ex 4)

```
Digite a palavra a ser pesquisada:
Q

Palavra nao encontrada
```