Dictionary_indexation

Generated by Doxygen 1.8.13

Contents

Chapter 1

Dictionary_indexation

Objetivo

Este trabalho consiste na construção e utilização de estrutura hierárquica denominada trie (do inglês "retrieval", sendo também conhecida coo árvore de prefixos ou ainda árvore digital) para a indexação e recuperação eficiente de palavras em grandes arquivos de dicionários (mantidos em memória secundária). A implementação deverá resolver dois problemas (listados a seguir), e os resultados deverão ser formatados em saída padrão de tela de modo que possam ser automaticamente avaliados no VPL.

A figura trie exemplifica a organização de um arquivo de dicionário. Cada linha apresenta a definição de uma palavra, sendo composta, no início, pela própria palavra com todos os caracteres em minúsculo (somente entre 'a' (97) e 'z' (122) da tabela ASCII) e envolvida por colchetes, seguida pelo texto de seu significado. Não há símbolos especiais, acentuação, cedilha, etc, no arquivo.

Primeiro problema: identificação de prefixos

Construir a trie, em memória principal, a partir das palavras (definidas entre colchetes) de um arquivo de dicionário, conforme o exemplo acima. A partir deste ponto, a aplicação deverá receber uma série de palavras quaisquer (pertencentes ou não ao dicionário) e responder se trata de um prefixo (a mensagem 'is prefix' deve ser produzida) ou não (a mensagem 'is not prefix' deve ser produzida na saída padrão). Sugestão de nó da trie:

NoTrie { char letra; //opcional NoTrie ∗filhos[26]; //pode ser uma 'LinkedList' de ponteiros unsigned long posição; unsigned long comprimento; //se maior que zero, indica último caracter de uma palavra } Segundo problema↩ : indexação de arquivo de dicionário

A contrução da trie deve considerar a localização da palavra no arquivo e o tamanho da linha que a define. Para isto, ao criar o nó correspondente ao último caracter da palavra, deve-se atribuir a posição do caracter inicial (incluindo o abre-colchetes '['), seguida pelo comprimento da linha (não inclui o caracter de mudança de linha) na qual esta palavra foi definida no arquivo de dicionário. Caso a palavra recebida pela aplicação exista no dicionário, estes dois inteiros devem ser produzidos. Importante: uma palavra existente no dicionário também pode ser prefixo de outra; neste caso, o caracter final da palavra será encontrado em um nó não-folha da trie e também deve-se produzir os dois inteiros (posição e comprimento) na saída padrão.

Exemplo:

Segue uma entrada possível para a aplicação, exatamente como configurada no VPL, contendo o nome do arquivo de dicionário a ser considerado, cuja a trie deve ser construída (no caso para 'dicionario1.dic' da figura acima), e uma sequência de palavras, separadas por um espaço em branco e finalizada por '0' (zero); e a saída que deve ser produzida neste caso.

Entrada: dicionario1.dic bear bell bid bu bull buy but sell stock stop 0

Saída: 0 149 150 122 273 82 is prefix 356 113 470 67 is not prefix 538 97 636 79 716 92

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Trie Class Reference

Public Member Functions

- Trie ()
  Constructor.
- void insert (string word, unsigned long position, unsigned long length)
  Inserts a word into the Trie.
- void search (string word)
  Searchs for a word in the Trie.

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Trie()

```
Trie::Trie ( )  [inline], [explicit]
```

Constructor.

4.1.2 Member Function Documentation

4.1.2.1 insert()

```
void Trie::insert (
            string word,
            unsigned long position,
            unsigned long length )  [inline]
```

Inserts a word into the Trie.

Parameters

word	the word to be inserted
position	the first character's position of the word
length	the length of the word's meaning

4.1.2.2 search()

```
void Trie::search (
             string word )  [inline]
```

Searchs for a word in the Trie.

Parameters

word	the word to be search on the Trie

The documentation for this class was generated from the following file:

- trie_implementation.cpp

Chapter 5

File Documentation

5.1 README.md File Reference

5.2 trie_implementation.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <vector>
```

Classes

- class Trie

Functions

- int main ()

5.2.1 Function Documentation

5.2.1.1 main()

```
int main ( )
```

Populates the dic_words and positions vectors

Populates the length vector

Inserts the words into the Trie

Expected results:

Index