Trabalho de Implementação II

Alek Frohlich & Nicolas Goeldner
Versão v1.0
Quinta, 20 de Junho de 2019

"Computers are good at following
instructions, but not at reading your mind."
— *Donald Knuth*

 VOLTAR À TABELA

Trabalho de Implementação II - Identificação de prefixos e Indexação de dicionários

Objetivo

Este trabalho consiste na construção e utilização de estrutura hierárquica denominada ***trie*** (do inglês "re**trie**val", sendo também conhecida como **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.

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).

Segundo problema: indexação de arquivo de dicionário

A construção 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 carácter da palavra, deve-se atribuir a **posição do carácter inicial** (incluindo o abre-colchetes '['), seguida pelo **comprimento da linha** (não inclui o carácter 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 carácter 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.

 VOLTAR À TABELA

Tabela de Desvios

 VOLTAR À TABELA

Documentação dos namespaces

Referência ao namespace structures

Este namespace contém a declaração e definição da estrutura TrieNode, usada como dicionário para o propósito proposto deste trabalho prático. Sem contar à definição da estrutura TrieNode, o namespace structures também possui uma interface para a mesma, oferecendo métodos como o initNode (que inicia um nodo sem filhos e com (pos,len) = (0,0)) e métodos de inserção e busca: insert() e search(). Esses três métodos somados à estrutura TrieNode completam toda a API necessária para resolver os problemas propostos pelo trabalho e portanto não foram implementados métodos de remoção.

Componentes

- struct TrieNode

Árvore de prefixos.

Funções

- struct TrieNode * initNode ()
  Aloca memória para um novo TrieNode.

- void insert (struct TrieNode *root, std::string word, int pos, int len)
  Adiciona uma chave na árvore de prefixos.

- std::pair< int, int > search (struct TrieNode *, std::string word)
  Procura por uma palavra na árvore de prefixos.

Documentação das funções

struct TrieNode * structures::initNode ()

 VOLTAR À TABELA

Aloca memória para um novo TrieNode.

A posição e o comprimento de cada TrieNode começam zerados. Além disso, todos as posições do vetor de nodos filhos começam nulas.

Retorna:

struct TrieNode* Ponteiro para o novo TrieNode criado.

Definido na linha 12 do arquivo trie.h.

void structures::insert (struct TrieNode * *root*, std::string *word*, int *pos*, int *len*)

Adiciona uma chave na árvore de prefixos. Os inteiros 'pos' e 'len' especificam, respectivamente, onde esta chave se encontra no dicionário e qual o tamanho de sua definição.

Parâmetros:

root	Root da árvore.
word	Palavra para inserir.
pos	Posição no dicionário da palavra a ser inserida.
len	Comprimento da linha no dicionário que possui a palavra a ser inserida.

Definido na linha 24 do arquivo trie.h.

std::pair< int, int > structures::search (struct TrieNode * *root*, std::string *word*)

Procura por uma palavra na árvore de prefixos.

Parâmetros:

word	Palavra a ser procurada na trie.
root	Raíz da árvore.

Retorna:

std::pair<int,int> Par que indica se a palavra pertence ao dicionário, é apenas um prefixo ou que não pertence ao dicionário. Se a palavra pertencer ao dicionário, a primeira entrada representa a posição da palavra enquanto a segunda, o comprimento da linha.

Definido na linha 39 do arquivo trie.h.

 VOLTAR À TABELA

Documentação da classe

Referência à estrutura structures::TrieNode

structures::TrieNode
Árvore de prefixos.

Atributos Públicos

- struct TrieNode * children [ALPHABET_SIZE]
- int pos
- int len

Descrição detalhada

Árvore de prefixos.

A árvore de prefixos é uma estrutura de dados eficiente no que diz respeito à recuperação de informações. Por isso ela também é conhecida como Trie, de reTRIEval. Usando uma Trie bem organizada, pode-se alcançar complexidade de busca O(M), onde M representa o tamanho máximo de suas chaves. Ao passo que, uma BST balanceada teria tempo proporcional a MlogN, em que N representa o número de chaves na árvore. Essas vantagens todavia não são sem seus custos, uma vez que a Trie perde para a BST em espaço ocupado em memória.

Definido na linha 30 do arquivo trie.h.

Documentação dos dados membro

TrieNode* structures::TrieNode::children[ALPHABET_SIZE]

Definido na linha 31 do arquivo trie.h.

int structures::TrieNode::len

Definido na linha 32 do arquivo trie.h.

int structures::TrieNode::pos

Definido na linha 32 do arquivo trie.h.

 VOLTAR À TABELA

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

1 src/trie.h

Documentação dos arquivos

Referência ao arquivo src/main.cpp

src/main.cpp
Código do programa principal.

Funções

ƺ int main ()

Descrição detalhada

Na main, primeiramente, criamos nossa Trie (que representará nosso dicionário). Para montar a nossa estrutura de dados lemos uma entrada que indica o nome do arquivo que utilizamos para obter as chaves que compõem a Trie e os dados do comprimento da linha e da posição da chave. Depois, lemos as entradas até chegar no "0", cada palavra lida é buscada na Trie. Se a palavra é uma chave da árvore, imprimimos o comprimento da linha e sua posição. Se a palavra é apenas um prefixo, imprimimos "is prefix". Se a palavra não for nenhuma das opções acima, imprimimos "is not prefix".

Autor:
Alek Frohlich, Nícolas Goeldner

Versão:
1.0

Data:
2019-06-20

Copyright:

Documentação das funções

int main ()

 VOLTAR À TABELA

Programa principal, realiza a leitura e processamento dos dicionários e indica o que as palavras da entrada são, se a palavra pertence ao dicionário é impresso a sua posição e o comprimento da linha em que a palavra está.

Definido na linha 24 do arquivo main.cpp.

Referência ao arquivo src/trie.h

src/trie.hsrc/trie.h

Declaração da estrutura TrieNode, além dos métodos para sua manipulação. Para a implementação, dirigir-se à seção seguinte. Para conferir o código fonte, dirigir-se ao VPL ou ao Github.

Componentes

- struct structures::TrieNode

Árvore de prefixos.

Namespaces

- structures

Macros

- #define ALPHABET_SIZE 26

Funções

- struct TrieNode * structures::initNode ()

Aloca memória para um novo TrieNode.

- void structures::insert (struct TrieNode *root, std::string word, int pos, int len)

Adiciona uma chave na árvore de prefixos.

- std::pair< int, int > structures::search (struct TrieNode *, std::string word)

Procura por uma palavra na árvore de prefixos.

Descrição breve

Declaração da estrutura TrieNode.

Autor:

Alek Frohlich, Nícolas Goeldner

 VOLTAR À TABELA

Versão:
1.0

Data:
2019-06-20

Copyright:
Copyright (c) 2019

Documentação das macros

trie.h:ALPHABET_SIZE

#define ALPHABET_SIZE 26

Definido na linha 15 do arquivo trie.h.

Referência ao src/trie.inc

src/trie.inc

Este arquivo contém as implementações das funções declaradas no arquivo-cabeçalho “trie.h”. Para mais informações, dirigir-se à seção de funções do namespace structures. Para conferir o código fonte, dirigir-se ao VPL ou Github.

Funções

- struct TrieNode * initNode ()
- void insert (struct TrieNode *root, std::string key, int pos, int len)
- std::pair< int, int > search (struct TrieNode *root, std::string key)

Descrição breve

Implementação do arquivo-cabeçalho “trie.h”.

Autor:
Alek Frohlich, Nícolas Goeldner

Versão:
1.0

Data:
2019-06-20

 VOLTAR À TABELA