Projeto Estrutura de Dados

Árvore de Prefixos

Thiago Z L Chaves (19100547)
Samuel Cardoso (19100544)
Professor: Alexandre e Aldo
Quinta, 15 de Outubro de 2020

Sumário

Índice dos Arquivos

Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Arquivos

Referência do Arquivo txtread.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include "Lista.cpp"
```

Funções

- int **txt_read** (string arquivo, Branch *root)

Funções

int txt_read (string arquivo, Branch *root)

Função serve para extrair os dados do objeto.dic:

É recebido o Objeto.dic e a Branch Root como parâmetros da função e então o programa inicia sua execução realizando os passos a seguir:
1)Abre o arquivo Dicionário.
2)Cria a variavel count_char, criada a fim de contar a posição do ultimo char da linha.
3)Cria a variavel count_global, criada a fim de contar o caractere atual globalmente.
4)Cria a variavel count_line, criada a fim de conter o comprimento da linha atual.
5) Percorre as linhas procurando um "[". Quando encontra, define a variável onWord como ‘1’, está variável que é responsável por quando dos elementos a seguir devem ser adicionados na String word.
6) Quando encontra um “]” setamos o onWord como ‘0’ para retratar que acabou a palavra, portanto não adicionamos mais caracteres.
7) Na segunda vez em diante quando achamos um ‘[‘ chamamos a função insert() do arquivo trie.struct passando a Branch, a word atual, count_char e count_line para que se insira a palavra e seus dados na Árvore de Prefixos.

```
void txt_read(string arquivo, Branch *root) {
    ifstream arquivotxt;
    string word = "";
    bool onWord = 0;
    int count_char = 0;
    int count_line = 0;
    int count_global = 0;
    arquivotxt.open(arquivo);

    if (arquivotxt.is_open()) {
        string linha;
        while(getline(arquivotxt, linha)) {
            for (int i = 0; i < linha.size(); i++) {
                if (linha[i] == '[') {
                    if (count_global != 0) {
                        insert(root, word, count_char, count_line);
                        count_global++;
                    }
                    count_char = count_global;
                    word = "";
                    count_line = linha.size();
                    onWord = 1;
                }
                if (linha[i] == ']') {
                    onWord = 0;
                }
                if (onWord == 1) {
                    if (linha[i] != '[' && linha[i] != ']') {
                        word += linha[i];
                    }
                }
                ++count_global;
            }
        }
    }
    insert(root, word, count_char, count_line);
```

Referência do Arquivo trie_struct.cpp

```
#include <iostream>
#include <bits/stdc++.h>
```

Struct Branch

- Struct Branch *lista_de_letras[26]
- Int forks
- bool isOneWord
- int count_char
- int count_line

Funções

- struct Branch* GetNode(void)
- void insert(struct Branch *root, string data, int count_char, int count_line)
- int order(struct Branch *current)
- int search(struct Branch *root, string data)
- struct Branch *isOneWord(struct Branch *root, string data)

Funções

struct Branch* GetNode (void)

Função basicamente cria um novo objeto Branch e seu argumento lista_de_palavras como um vetor de 26 posições com todas setadas como NULL.

void Insert (struct Branch *root, string data, int count_char, int count_line)

Verifica se o Node da árvore já contém o caractere, caso não o tenha a função GetNode é chamada e insere o mesmo na árvore. Além disso quando o caractere é o último caractere da palavra o Node tem o parâmetro isOneWord setado para True, e os parâmetros count_char e count_line recebem os dois últimos valores da entrada respectivamente.

void order (struct Branch *current)

É uma função recursiva, a qual percorre todos os Nodes não nulos depois do Node do último caractere da palavra usada como pesquisa de prefixo, e tendo percorrido retorna a quantidade de Nodes que possuem o atributo isOneWord setado como True.

void search (struct Branch *root, string data)

Função verifica se a palavra existe na árvore e retorna a quantidade de prefixos que a palavra atual possui. Caso não exista retorna:
“Palavra” is not a prefix.

struct Branch *isOneWord (struct Branch *root, string data)

Função verifica se é uma palavra já definida na árvore.

Referência do Arquivo main.cpp

```
#include <iostream>
#include "trie_struct.cpp"
#include "txtread.cpp"
```

Funções

- int main ()

Funções

int main( )

Funcão cria variavel filename.

Recebe o .dic do terminal e é alocado na variável string filename.

Chama a função txt_read.

Através da função search() verificamos se a palavra é ou não um prefixo, e caso seja, retornamos também a quantidade de prefixos através do retorno da função junto com o número do caractere '[' da linha do prefixo em relação ao texto como um todo e o tamanho da respectiva linha.

```cpp
#include <iostream>
#include <fstream>
#include "XMLValidation.h"

using namespace std;

int main() {

    char xmlfilename[100];

    cin >> xmlfilename;  // Entra (Nome do arquivo)

    validation(xmlfilename); // Função responsável por validar e iniciar a segunda etápa

    return 0;
}
```

Dificuldades Apresentadas

Neste trabalho tivemos como principal complicação durante seu desenvolvimento a montagem da Árvore de Prefixos (Trie) e seus respectivos métodos. A principal dificuldade apresentada no trabalho anterior que era o Parsing do arquivo nesse já foi mais fácil uma vez que já obtivemos experiência com isso.