CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA

Organização de Estrutura de Arquivos
Segunda Avaliação – Árvore Trie.

RIO DE JANEIRO, 13 DE JUNHO DE 2016.

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA

ALUNO:
SOLON CANTO DE OLIVEIRA NETO

CURSO:
BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

UNIDADE:
MARACANÃ

RIO DE JANEIRO, 13 DE JUNHO DE 2016.

Sumário

Objetivo

O seguinte trabalho tem como objetivo acompanhar a implementação de uma árvore Trie em arquivo utilizando a linguagem de programação C.

Introdução

Como descrito em aula a árvore Trie é uma árvore onde são inseridas palavras e cada nó representa uma letra. Desta forma tendo a raiz como ponto de partida ela pode possuir até 26 nós filhos (um para cada letra), cada um representando a primeira letra das palavras inseridas.
E cada nó pode ter também 26 nós filhos para as letras seguintes das palavras inseridas.

Implementação

A implementação da árvore Trie foi feita com 3 funções principais para a inserção das palavras que são:

- Criação de um novo nó;
- Inserção da palavra;
- Leitura das palavras do arquivo fonte;

Criação de um novo nó:

```
int createNode(FILE *trieFile){
	Node n;
	int reg;
	memset(&n, 0, sizeof(Node));
	fseek(trieFile, 0, SEEK_END);
	fwrite(&n, sizeof(Node), 1, trieFile);
	fseek(trieFile, 0, SEEK_END);
	reg = (int)((ftell(trieFile)/sizeof(Node))-1);

	return reg;
}
```

A criação dos nós é feita da seguinte forma:

Uma nova estrutura nó é alocada em memória, logo depois ela é escrita no final do arquivo de saída e então é calculado o número de registro desse novo nó, ou seja, quantos nós já foram escritos no arquivo para que esse valor possa ser retornado para a função que chama esta.

Essa função é chamada uma vez no início do programa para criar a raiz da árvore e depois sempre que for necessária para a inserção de uma palavra.

Inserção de palavras

A inserção das palavras é feita em 3 partes. Na primeira vai ser verificado se a primeira letra da palavra é um filho da raiz, caso não seja a palavra toda é inserida.

```
if(n.next_char[str[0] - 'a'] == 0){
        for(i = 0; i< strlen(str); i++){
                newReg = createNode(trieFile);
                fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
                fread(&n, sizeof(Node), 1, trieFile);
                n.next_char[str[i] - 'a'] = newReg;
                fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
                fwrite(&n, sizeof(Node), 1, trieFile);
                actReg = newReg;
                }
        fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
        fread(&n, sizeof(Node), 1, trieFile);
        n.is_end = 1;
        fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
        fwrite(&n, sizeof(Node), 1, trieFile);
        printf("nova palavra\n");//db

        return 0;
}
```

Na segunda é verificado se a palavra já não está inserida dentro da Trie porém não marcada como palavra pois foi inserida anteriormente outra palavra que à continha.

```
else{
	while(n.next_char[str[j] - 'a'] != 0){
		if(n.is_end == 1 && j == (strlen(str)-1)){
			printf("palavra ja inserida\n");//db
			return 1;
		}
		if(n.is_end == 0 && j == (strlen(str)-1)){
			fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
			fread(&n, sizeof(Node), 1, trieFile);
			n.is_end = 1;
			fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
			fwrite(&n, sizeof(Node), 1, trieFile);
			printf("nova palavra\n");//db
			return 0;
		}
		actReg = n.next_char[str[j] - 'a'];
		fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
		fread(&n, sizeof(Node), 1, trieFile);
		j++;
	}
```

E na terceira etapa caso somente algumas letras dessa palavra já tenham sido inseridas as outras serão todas inseridas.

```
for(i = j; i<strlen(str); i++){
	newReg = createNode(trieFile);
	fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
	fread(&n, sizeof(Node), 1, trieFile);
	n.next_char[str[i] - 'a'] = newReg;
	fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
	fwrite(&n, sizeof(Node), 1, trieFile);
	actReg = newReg;

}
fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
	fread(&n, sizeof(Node), 1, trieFile);
	n.is_end = 1;
	fseek(trieFile, actReg*sizeof(Node), SEEK_SET);
	fwrite(&n, sizeof(Node), 1, trieFile);
	printf("nova palavra\n");//db
	return 0;
	}
```

Leitura das palavras do arquivo fonte

O programa tem como objetivo inserir palavra de um arquivo de texto na árvore Trie e para isso foi feita a função que lê cada palavra do arquivo e chama a função de inserção para cada uma.

```
void getWords(FILE *textFile, FILE *trieFile){
int c;
	int i = 0;
	char str[99];
	c = fgetc(textFile);
	while(c != EOF){
		if(isalpha(c)){
			str[i] = c;
			i++;
			c = fgetc(textFile);
		}
		else{
			if(str){
				insertWord(str, trieFile);
				//printf("%s", str);//db
				//printf("...\n");//db
				memset(str, 0, 99);
				i = 0;
				c = fgetc(textFile);
			}
			else{
				memset(str, 0, 99);
				i = 0;
				c = fgetc(textFile);
			}
		}
	}

}
```

Considerações Finais

O arquivo de texto utilizado como base para o desenvolvimento do trabalho foi o e-book The Adventures of Sherlock Holmes retirado do Projeto Gutenberg alterado para as necessidades. O arquivo original assim como o editado e o código do programa utilizado para a edição constam junto ao projeto.

Todos os arquivos com códigos e texto se encontram no repositório no Git Hub. https://github.com/OliveiraNt/trie-Tree