Trabalho - Índice Remissivo

Professora: Maria Adriana Vidigal

Alunos: Lorenzo Machado Burgos - 12011BCC005
Marcus Vinícius Torres Silva - 12011BCC025

Link para repositório no Github: https://github.com/lorenzo-burgos/Indice-Remissivo

O trabalho tem como objetivo praticar os conceitos vistos em aulas sobre Árvore AVL. O projeto aborda a criação de um Índice Remissivo tendo em base um documento, utilizando árvore binária AVL. No nó de uma árvore, deve conter uma palavra e um vetor de inteiros para armazenar a linha correspondente de determinada palavra. Ao final, deve ser retornado em um arquivo texto, as palavras que estão contidas no documento de referência, as linhas em que tais palavras aparecem e as informações de quantas palavras há no documento, o total de palavras distintas e o tempo de construção da árvore.

Em relação a estrutura presente no trabalho, utilizamos uma “struct” chamada indice, que contém os campos palavra e linha. Além disso, temos uma “struct” No, que contém as informações do nó de uma árvore. As estruturas estão em um arquivo chamado “arvore.h”.

Exemplo:
```
struct indice{
    char pal[30];
    int linha;
};
typedef struct indice Ind;

struct NO{
    Ind info;
    int altura;
    struct NO *esq;
    struct NO *dir;
};
typedef struct NO* ArvAVL;
```

Sobre as principais funções que estão presentes no arquivo “arvore.c”, tem-se a função de criação de uma árvore AVL, a qual aloca um espaço da memória para a criação da árvore e faz a raiz da árvore apontar para NULL.

Exemplo:
```
ArvAVL* cria_ArvAVL(){
    ArvAVL* raiz = (ArvAVL*) malloc (sizeof(ArvAVL));
    if(raiz != NULL)
        *raiz = NULL;
    return raiz;
}
```

A função de inserção na árvore, tem como objetivo pegar as informações que são passadas por parâmetro, palavra e linha, e colocar no campo info dentro do nó da árvore.

```
int insere_ArvAVL(ArvAVL *raiz, Ind valor){
    int r;
    if(*raiz == NULL){
        struct NO *novo;
        novo = (struct NO*)malloc(sizeof(struct NO));
        if(novo == NULL)
            return 0;

        novo->info = valor;
        novo->altura = 0;
        novo->esq = NULL;
        novo->dir = NULL;
        *raiz = novo;
        return 1;
    }

    struct NO *atual = *raiz;

    valor.linha = atual->info.linha;

    if(strcoll(valor.pal, atual->info.pal)<0){
        r = insere_ArvAVL(&(atual->esq), valor);
        if(r == 1){
            if(fatorBalanceamento_NO(atual) >= 2){
                if(strcoll(valor.pal, (*raiz)->esq->info.pal)<0){
                    RotacaoDireita(raiz);
                }
                else{
                    RotacaoDuplaDireita(raiz);
                }
            }
        }
    }else{
        if(strcoll(valor.pal, atual->info.pal)>0){
            r = insere_ArvAVL(&(atual->dir), valor);
            if(r == 1){
                if(fatorBalanceamento_NO(atual) >= 2){
                    if(strcoll((*raiz)->dir->info.pal,valor.pal) < 0){
                        RotacaoEsquerda(raiz);
                    }
                    else{
                        RotacaoDuplaEsquerda(raiz);
                    }
```

```
                }
            }
        }else{
            return 0;
        }
    }

     atual->altura = maior(altura_NO(atual->esq),altura_NO(atual->dir)) +
1;

    return r;
}
```

No código da função, são chamadas as funções de rotação, caso precise, e para isso temos a função de balanceamento para ter esse controle.

Exemplo:

```
int fatorBalanceamento_NO(struct NO* no){
    return labs(altura_NO(no->esq) - altura_NO(no->dir));
}

void RotacaoDireita(ArvAVL *A){
    struct NO *B;
    B = (*A)->esq;
    (*A)->esq = B->dir;
    B->dir = *A;
    (*A)->altura = maior(altura_NO((*A)->esq),altura_NO((*A)->dir)) + 1;
    B->altura = maior(altura_NO(B->esq),(*A)->altura) + 1;
    *A = B;
}

void RotacaoEsquerda(ArvAVL *A){
    struct NO *B;
    B = (*A)->dir;
    (*A)->dir = B->esq;
    B->esq = (*A);
    (*A)->altura = maior(altura_NO((*A)->esq),altura_NO((*A)->dir)) + 1;
    B->altura = maior(altura_NO(B->dir),(*A)->altura) + 1;
    (*A) = B;
}

void RotacaoDuplaDireita(ArvAVL *A){
    RotacaoEsquerda(&(*A)->esq);
    RotacaoDireita(A);
}
```

```
void RotacaoDuplaEsquerda(ArvAVL *A){
    RotacaoDireita(&(*A)->dir);
    RotacaoEsquerda(A);
}
```

Em relação a função main do programa, é nela que ocorre a impressão das palavras e linhas, que é feita fazendo a gravação em um arquivo texto. O código usado para fazer a leitura do conteúdo de um arquivo texto, tem como referência o código deixado pela a professora na página da disciplina do moodle.

Exemplo:

```
ArvAVL *avl;
    clock_t tempo;
    struct NO *res;
    Ind t;
    FILE *arq = fopen(filename, "r");
    FILE *arq2 = fopen("gravacao.txt", "wt");
    char *conteudo_linha = malloc(MAX);
    char *palavra = malloc(50);
    int linha = 1;
    int qtd = 0, dist = 0;

    setlocale (LC_COLLATE, "");

    avl = cria_ArvAVL();

    while(fscanf(arq, "%[^\n] ", conteudo_linha) != EOF){
        palavra = strtok(conteudo_linha," ,.;!?/");
        while(palavra != NULL){
            minusculo(palavra);
            strcpy(t.pal,palavra);
            t.linha = linha;
            insere_ArvAVL(avl,t);
            fprintf(arq2,"%s %d\n", t.pal, t.linha);
            palavra = strtok(NULL," ,.;!?/");
            qtd++;
        }
        linha++;
    }
    fclose(arq);

    fprintf(arq2,"\nNumero total de palavras: %d",qtd);
    fprintf(arq2,"\nNumero de palavras distintas: %d",dist);
    fprintf(arq2,"\nTempo de construcao do indice usando arvore AVL:
%lds\n", tempo);
```

Exemplo do programa em execução: