UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA - DEPARTAMENTO DE INFORMÁTICA APLICADA

INF 01203 – ESTRUTURA DE DADOS

RELATÓRIO DE IMPLEMETAÇÃO

TRABALHO PRÁTICO FINAL

LEONARDO MARQUES RODRIGUES – 00213751

MARCOS VINICIOS DA SILVA – 00261600

PORTO ALEGRE, DEZEMBRO DE 2015

FUNCIONAMENTO DO ALGORITMO:

O algoritmo desenvolvido para solução da tarefa consiste em 3 partes, listadas as seguir:

- Leitura do arquivo de texto de entrada (corpus) e leitura do arquivo de texto com as consultas solicitadas.
- Geração de estatísticas.
- Consulta das estatísticas na estrutura de dados e escrita no arquivo de saida.

ESTRUTURAS DE DADOS:

As estruturas de dados utilizadas no desenvolvimento da solução foram as seguintes:

- Lista Simplesmente Encadeada:
  - Utilizada para armazenamento do corpus, e das palavras a serem consultadas.

    CÓDIGO EM C PARA ESTA ESTRUTURA:

```
typedef struct word{
      char p[TAM_PALAVRA];
}WORD;

typedef struct nodo{
      WORD inf;
      struct nodo *next;
}NODO;
```

- Árvore Binaria de Pesquisa (AVL):
  - Foram utilizadas duas árvores AVL neste trabalho, uma, para armazenar as palavras de interesse (consultadas) e suas frequências, e outra para armazenar as palavras de co-ocorrência para cada palavra consultada.

    CÓDIGO EM C PARA ESTRUTURAS:

  - Estrutura para palavras consultadas:

```
typedef struct p_cons{
     char palavra[TAM_PALAVRA];
     int  freqA;
}P_CONS;

typedef struct cons{
      struct cons *esq;
      struct cons *dir;
      int FB;
      P_CONS inf;
      NO_PROX *viz;
}NO_CONS;
```

  - Estrutura para palavras de co-ocorrência:
    typedef struct prox_palavra{

```
typedef struct prox_palavra{
  char palav[TAM_PALAVRA];
  int  freqAB;
  int  freqB;
  double stat;
}PROX_PALAVRA;

typedef struct prox{
 struct prox *esq;
 struct prox *dir;
 int FB;
 PROX_PALAVRA inf;
}NO_PROX;
```

Leitura do Corpus:

A leitura do arquivo de texto funciona da seguinte maneira:

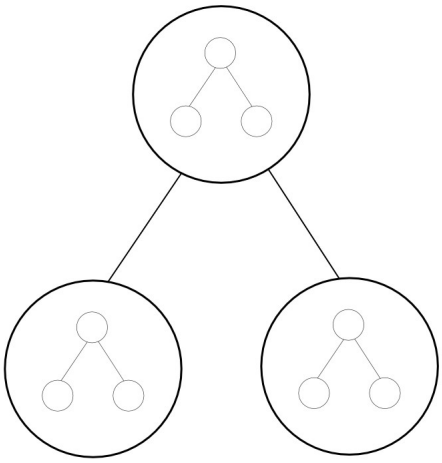
O programa chama a *ARQ_LE_TEXTO* do TAD fileread.h. Esta função le um arquivo de texto com seu caminho passado por parâmetro, separa as suas palavras de acordo com os seguintes separadores: *{'SPC','!','"','#','$','%','&','\'','(',')','*','+','"',',','-','.','/','0','1','2','3','4','5','6','7','8','9',':',';','<','=','>','?','@','[','\\',']','^','_','`','{','|','}','~','\n','\t','\r'}*, após isso, insere cada palavra separada em uma LSE. Ao final, retorna um ponteiro para o primeiro elemento da lista contendo as palavras do texto.

Esta função foi usada para ler o corpus e das palavras a serem consultadas, e optou-se por uma lista simplesmente encadeada pela simplicidade de verificar a co-ocorrência das palavras do corpus.

Geração de Estatísticas:

Após a leitura do corpus, o programa principal chama a função, *busca_palavras* do TAD consultadas.h. Esta função recebe o corpus, a lista de palavras a serem consultadas e procura cada ocorrencia das palavras consultadas no corpus.

Sempre que encontra no corpus uma palavra consultada, insere esta palavra (consultada) numa AVL do tipo NO_CONS, incrementando um contador da frequência desta palavra, além disto, extrai a informação de qual é a próxima palavra (proxima) a esta no corpus, e insere a *próxima* na arvore de co-ocorrencias da *consultada*, que é uma árvore do tipo NO_PROX, então é realizada uma contagem de frequencia da *proxima* no corpus, e após, é incrementado um contador de co-ocorrencia dessas duas palavras. Abaixo, segue uma ilustração da estrutura de dados retornado pela função *busca_palavras:*



Após a geração da árvore acima, o programa principal, chama a função *calcula_estatCONS,* que percorre a árvore principal, consultando a frequencia de ocorrência de cada uma das palavras a ser consultada, e após, usa esse valor para calcular as estatisticas de co-ocorrência para todas a palavras armazenadas na árvore de co-ocorrência de cada palavra consultada.

Após o cálculo dessas estatisticas, é retornado para o programa principal, a arvore das palavras consultadas, já com todas as estatísticas de co-ocorrência calculadas, estando assim, pronta para consulta e gravação no arquivo para gerar as sugestões.

Optou-se por usar duas AVLs para construir esta estrutura, pois, embora o custo de inserção em uma AVL seja alto, o tempo de consulta em uma AVL é baixo, assim, conseguimos realizar o cálculo das estátisticas e a consulta nesta mesma árvore para gravação no arquivo posteriormente, em tempos satisfatórios, acelerando a execução do programa.

Escrita no Arquivo:

A escrita no arquivo trabalha basicamente com a composição de 3 funções:

- grava_arquivo
- imprimeCONS
- imprimeSUG

A função “grava_arquivo” serve como o ponto de partida do processo, onde é realizada a criação de arquivo para armazenar as informações. Após sua abertura, a função “imprimeCONS” é acionada dando inicio ao caminhamento das árvores.

A função “imprimeCONS” trabalha apenas na árvore principal, das palavras consultadas, onde são acessados todos seus nodos sem restrição. O caminhamento percorrido é o central à esquerda. Cada acesso a um nodo é gravado no arquivo a palavra que foi pesquisada através do fprintf, em seguida é chamada a função “imprimeSUG”.

A função “imprimeSUG” percorre a árvore de co-ocorrências através do caminhamento central à direita, buscando primeiramente os nodos com maiores estatísticas. Esta por sua vez possui uma restrição, será percorrido e armazenados apenas n-nodos,onde n é um número inteiro informado na chamada do programa em linha de comando através de um parâmetro.