UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Disciplina: Algoritmos e Estrutura de Dados 2
Aluno: Jean Carlos Martins Miguel R.A: 1640593
Professor: Dr. Juliano Henrique Foleis

Trabalho 1 - Dicionário de Anagramas

O objetivo deste trabalho é de que, dada uma palavra, o algoritmo deve encontrar todos os anagramas possíveis para a palavra fornecida, sendo que, o professor forneceu um arquivo *.txt* que contém as palavras do dicionário da língua portuguesa. Para o trabalho utilizei um vetor de estrutura,por ser mais fácil manipular vetor, tanto na questão de ordenação e na busca das palavras, o qual está definida da seguinte forma:

```
typedef struct palavra{
   char original[30];
   char ordenada[30];
}palavra;
```

O algoritmo está dividido em 4 arquivos, sendo: utils.h, com a estrutura principal **palavra**, e com os protótipos das seguintes funções:

```
palavra* Palavra_Criar();
void ler_arquivo();
int partition(char* v, int p, r);
void quickSort(char*v, int e, int d);
void ordenacao(palavra* palavra, char* buscar);
int partition_palavra(char*v, int p, int r);
void quickSort_palavra(char* v,int e , int d);
int busca_binaria(palavra* v, int e , int d, char*x);
void merge(palavra* v,int p, int q , int r);
int mergeSort(palavra *v, int e, int d);
int novo_dicionario(palavra* palavra);
void minusculo(char* s1);
```

O arquivo utils.c, contém as implementações das funções.
O main.c é o arquivo principal para a chamada das funções
Makefile, criado para facilitar a compilação do algoritmo:
> Makefile
> ./main

A estrutura do tipo palavra foi alocada dinamicamente na função *palavra* Palavra_criar()* .O tamanho previamente definido é o tamanho de linhas contidas no arquivo,então utilizei já que sabia de antemão . A estrutura palavra,contém dois campos, o primeiro é um vetor de char, que conforme é feito a leitura do arquivo na função void *ler arquivo(),*cada palavra será inserida em uma posição do vetor da estrutura no campo original, o mesmo acontece com o campo ordenada. Após feita a leitura do arquivo e deixado o arquivo em minúsculo, para facilitar a comparação das palavras, com a função *void_minusculo* e inserindo as palavras do arquivo nos respectivos campos da estrutura, utilizei o algoritmo **quicksort** para fazer a ordenação do vetor pelo campo *ordenada*, então cada campo do vetor *ordenada* tem a palavra do vetor original só que de forma ordenada, *ex: aarao - aaaor,* feito essa ordenação de cada palavra, utilizei outro algoritmo estável (n(Log)n), o **mergeSort** , agora para ordenar o vetor pelo campo das palavras ordenadas, (pois assim, cada palavra que for anagrama da outra ficará uma ao lado da outra, facilitando a busca). Depois de ter ordenado o vetor pelo campo das palavras ordenadas, foi utilizado a **busca binária,** pois é o método de busca mais eficiente para casos em que temos um vetor ordenado por alguma chave, então quando o usuário digitar a palavra e essa palavra for ordenada, a busca binária irá procurar no campo das palavras ordenadas se a palavra informada de forma ordenada corresponde com alguma palavra do vetor, caso sim, precisou-se criar um laço de repetição para encontrar onde começa a palavra que contém o anagrama e onde termina, para fazer a impressão correta, já que o algoritmo poderia retornar qualquer posição onde contém a palavra ordenada.

Exemplos de entrada e saída do algoritmo

ENTRADA	SAÍDA
mouse	mouse usemo suemo ousem
chinelo	chinelo encolhi chinelo chileno
pedra	pedra perda pedra padre