ACH2023 - Algoritmos e Estruturas de Dados I

Primeiro Exercício Programa

Indexador e Buscador de Palavras

Objetivos

Neste EP vocês deverão implementar um indexador e buscador de palavras para arquivos texto. Ao executar, o programa deve carregar o conteúdo de um arquivo texto, armazená-lo em memória (de forma que se possa ter acesso eficiente a uma linha qualquer em um momento posterior) e também indexar suas palavras (isto é, criar uma estrutura que permita verificar a existência ou não de uma palavra sem que seja necessário verificar o texto linha a linha). Na sequência, o programa fica à disposição do(a) usuário(a) para que ele(a) possa realizar algumas buscas. Caso a palavra procurada exista no texto, o programa deve exibir as linhas do texto nas quais a palavra ocorre.

O propósito de se criar um índice, antes da realização das buscas em si, é acelerar o processo de busca. Para isso vocês deverão empregar as estruturas de dados estudadas ao longo do semestre (listas sequenciais, listas ligadas, árvores, etc). Vocês deverão implementar dois tipos diferentes de índice (o tipo a ser usado será definido no ato da execução do programa através de um parâmetro de linha de comando):

- baseado em lista
- baseado em árvore

Ambos os índices deverão manter uma coleção de entradas, onde cada entrada contém uma string (ou seja, representa uma palavra do texto) e outras informações eventualmente relevantes para gerar a saída esperada (total de ocorrências da palavra, linhas em que a palavra aparece).

Para desenvolver este EP de modo que o programa cumpra seu objetivo, vocês podem usar as implementações de estruturas de dados estudadas e disponibilizadas como base, fazendo as alterações necessárias.

O EP desenvolvido também deve indicar o tempo que o programa leva para realizar cada operação: carga inicial do arquivo/construção do índice, busca de palavras.

Funcionamento detalhado do EP

O programa, ao ser executado na linha de comando, deve receber dois parâmetros. O primeiro define o nome do arquivo texto a ser indexado, e sobre o qual as buscas serão feitas. O segundo parâmetro define o tipo de estrutura a ser empregada para representar o índice na execução atual. As únicas opções válidas para este segundo parâmetro são: **lista** ou **arvore**. Se a primeira opção for escolhida, a estrutura na qual as entradas do índice serão inseridas deve ser uma lista (o tipo exato a ser usado fica a critério de vocês, mas não esqueçam de ponderar a respeito da eficiência esperada da busca para o tipo de lista empregada). Já se a segunda opção for escolhida, as entradas do índice deverão ser organizadas em uma árvore binária de busca.

Para entender melhor o funcionamento do programa, considere o seguinte trecho de texto, extraído do capítulo 1 do livro *Introduction to Algorithms* (Cormen et al, 2009):

```
Informally, an algorithm is any well-defined computational procedure that takes
some value, or set of values, as input and produces some value, or set of values,
as output. An algorithm is thus a sequence of computational steps that transform
the input into the output.

We can also view an algorithm as a tool for solving a well-specified computational
problem. The statement of the problem specifies in general terms the desired
input/output relationship. The algorithm describes a specific computational
procedure for achieving that input/output relationship.

For example, we might need to sort a sequence of numbers into nondecreasing order.
This problem arises frequently in practice and provides fertile ground for introducing
many standard design techniques and analysis tools.
```

Suponha que este trecho esteja salvo em um arquivo chamado `texto.txt`, e que gostaríamos de realizar buscas sobre o conteúdo deste texto. O trecho abaixo ilustra como o programa deve ser executado, assim como a saída gerada pelo mesmo. Vocês devem seguir à risca a formatação de saída do programa conforme ilustrada no exemplo abaixo:

```
$./ep1 texto.txt arvore
Tipo de indice: 'arvore'
Arquivo texto: 'texto.txt'
Numero de linhas no arquivo: 13
Tempo para carregar o arquivo e construir o indice: XXXXX ms
> busca algorithm
Existem 4 ocorrências da palavra 'algorithm' na(s) seguinte(s) linha(s):
00001: Informally, an algorithm is any well-defined computational procedure that takes
00003: as output. An algorithm is thus a sequence of computational steps that transform
00006: We can also view an algorithm as a tool for solving a well-specified computational
00008: input/output relationship. The algorithm describes a specific computational
Tempo de busca: XXXXX ms
> busca set
Existem 2 ocorrências da palavra 'set' na(s) seguinte(s) linha(s):
00002: some value, or set of values, as input and produces some value, or set of values,
Tempo de busca: XXXXX ms
> busca quicksort
Palavra 'quicksort' nao encontrada.
Tempo de busca: XXXXX ms
```

```
> buska quicksort
Opcao invalida!
> fim
$
```

Notem que, nas linhas em que é impresso o tempo que uma determinada operação levou para ser realizada, a string ‘XXXXX’ deve ser substituída pelo tempo real medido (em milissegundos) que aquela operação levou.

Observem também que após a carga do arquivo de texto e construção do índice, o programa deve imprimir a sequência de caracteres `'> '`, indicando que está esperando por algum tipo de entrada do(a) usuário(a). O(a) usuário(a) pode então executar dois “comandos” possíveis: o comando “busca” (que deve ser seguido da palavra a ser buscada) ou o comando “fim” (que causa a finalização do programa). Enquanto a opção “fim” não for digitada, o programa continua esperando comandos do(a) usuário(a).

Observações

Seu programa deve ignorar por completo qualquer sinal de pontuação. Palavras compostas com o uso do hífen, devem ser tratadas como duas palavras separadas. Assuma também que a busca é sempre feita considerando palavras inteiras (por exemplo, se no arquivo texto existem as palavras “algorithm” e “algorithms”, e uma busca é feita pela palavra “algorithm”, as ocorrências da palavra no plural não devem ser consideradas. Por fim, a busca deve ser realizada de tal modo que versões maiúsculas e minúsculas de um caracteres sejam consideradas iguais.

Entrega

Além do(s) código(s) fonte(s), entregue também um arquivo README.TXT com as instruções de como seu EP deve ser compilado. Também deve ser entregue um pequeno relatório no qual devem ser relatados o resultado de alguns experimentos em que sejam variados o tamanho dos arquivos de texto, assim como o tipo do índice empregado. Apresente os resultados em relação ao consumo de tempo das operações (carga do arquivo/construção do índice e realização das buscas) e discuta os resultados obtidos.

Este EP pode ser feito em grupos de **2 ou 3 pessoas** (não serão aceitos grupos com mais de 3 componentes). Para realizar a entrega, crie um arquivo no formato ZIP contendo os arquivos fonte do EP, o README.TXT e o relatório em formato PDF. Entregue esse arquivo ZIP na atividade aberta no eDisciplinas referente a este EP até o final do dia **21 de dezembro** (apenas um membro do grupo deve fazer o envio).

Boa diversão! :)