UNIVERSIDADE MACKENZIE – SISTEMAS DE INFORMAÇÃO

ESTRUTURA DE DADOS – TURMA 3H – 2025/1

Profa. Solange Barros

Prof. Jefferson Zanutto

PROJETO 2 – Análise de discurso

A depressão é uma condição séria que afeta muitas pessoas, muitas vezes de forma silenciosa. Nem sempre quem está passando por isso consegue pedir ajuda diretamente — mas algumas pistas podem aparecer na forma como a pessoa se comunica. Palavras carregadas de tristeza, cansaço ou desânimo, por exemplo, podem indicar que algo não vai bem.

Este projeto propõe usar a programação para analisar discursos e identificar possíveis sinais de depressão. A ideia é ler um texto, guardar suas palavras em uma estrutura de árvore binária de busca (ABB) e verificar se ele contém termos ligados a esse tipo de sofrimento emocional. Assim, conseguimos mostrar como a tecnologia pode ajudar a perceber quando alguém pode estar precisando de apoio.

Se você ou alguém próximo estiver enfrentando momentos difíceis, lembre-se: você não está sozinho. Falar com alguém de confiança e buscar ajuda profissional pode fazer toda a diferença.

OS ARQUIVOS TXT

1. Discurso.TXT

O arquivo chamado **Discurso.txt** contem o discurso ou fala de uma pessoa qualquer. Ele deve conter apenas palavras em português, separadas por espaços em branco. O conteúdo pode conter várias linhas, mas a separação entre as palavras deve seguir o padrão de espaço simples.

Exemplo:

me sinto cansado sem vontade de sair de casa os dias parecem iguais e nada me anima

Observações:

- Todas as palavras devem ser convertidas para letras minúsculas pelo sistema antes de ser armazenadas na ABB;

- Palavras com acento serão tratadas normalmente.
- Pontuações como vírgulas, pontos e acentos especiais devem ser removidas do texto.
- É importante submeter um arquivo .TXT de discurso que não contenha palavras associadas à depressão para efeito de teste.

2. PalavrasDepressao.txt

Este arquivo contém uma lista de palavras associadas à depressão, como sentimentos negativos, emoções ligadas à tristeza, apatia, angústia, etc. Cada palavra deve estar em uma linha separada. O arquivo .TXT está disponível no Moodle e as palavras nele contidas foram extraídas de pesquisas que a Clinical Psychological Science fez a partir da análise de fóruns online de saúde mental e relatos de pacientes.

Exemplo de conteúdo:

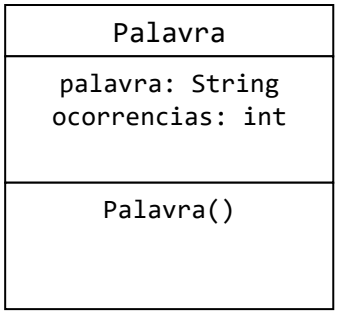
triste
cansado
vazio
dor
sofrimento
choro
solidão
desesperança etc.

Observações:

- As palavras devem estar no singular e em letras minúsculas, sem pontuação.
- O sistema fará a comparação exatamente com os termos dessa lista, portanto é importante manter a formatação simples.

DEFINIÇÃO DO PROJETO

Objetivo: Fazer a implementação de uma árvore binária de busca, que armazena objetos do tipo Palavra, que são originadas de um arquivo TXT que contém o discurso de uma pessoa qualquer. A ideia é armazenar cada uma das palavras que compõe o texto lido. Se ocorrer repetição de uma palavra no texto, deverá ser feita a contagem, a partir desta classe:



O método Construtor dessa classe deve receber uma palavra do texto como parâmetro e armazenar o valor 1 nas ocorrências (qtde de vezes que a palavra apareceu no discurso). Quando uma palavra é inserida na árvore, significa que é a primeira vez que ela foi encontrada no texto.

Menu Principal: seu programa deverá iniciar, a partir da exibição do seguinte menu na tela:

1. Carregar discurso
2. Contador de palavras
3. Buscar palavra
4. Exibir as palavras do discurso em ordem alfabética
5. Verificar sinais de depressão
6. Estatísticas sobre o texto
7. Sair

Detalhamento das opções:

1- CARREGAR DISCURSO - Deve ler o arquivo Discurso.txt e armazenar todas as palavras na árvore binária de busca (ABB). Antes de inserir cada palavra, deve verificar se ela já existe na árvore. Se existir, apenas atualize o contador de ocorrências. Se não, insira a nova palavra. Durante a leitura, todo o texto deve ser convertido para letras minúsculas. Palavras separadas por espaço devem ser inseridas uma a uma na árvore. Ao final, exiba a mensagem: **"Discurso carregado com sucesso"**.

2- CONTADOR DE PALAVRAS – Exibe o número total de palavras no discurso (ou seja, a soma das ocorrências de todas as palavras armazenadas na árvore).

3- BUSCAR PALAVRA - Solicita ao usuário que digite uma palavra e é feita a busca essa palavra na árvore. Se encontrada, exibe o número de ocorrências. Se não encontrada, exibe a mensagem: **"Palavra não encontrada no discurso"**.

4- EXIBIR PALAVRAS EM ORDEM ALFABÉTICA – Exibe o conteúdo da árvore em ordem alfabética. Para cada palavra, mostre também o número de vezes que ela apareceu no discurso.
5- VERIFICAR SINAIS DE DEPRESSÃO – Lê o arquivo PalavrasDepressao.txt contendo uma lista de palavras associadas à depressão. Compara essas palavras com as que estão presentes na árvore. Deve exibir: Quantas palavras da lista foram encontradas no discurso; quais foram essas palavras; Quantas vezes cada uma apareceu.

   Exemplo:

   Sinais de alerta identificados: 3 palavras
   Palavras encontradas:
   - cansado (3) - vazio (1) - dor (2)

6- ESTATÍSTICAS – cada grupo deve criar duas estatísticas a respeito do texto lido e apresentar nessa opção. Por exemplo: Número de palavras distintas; Palavra mais frequente; Quantidade de palavras que apareceram apenas uma vez, etc.
7- SAIR - exibir o nome dos integrantes, link do vídeo de apresentação e encerrar a aplicação.

Regras de implementação:

1- Todos os dados de entrada devem ser validados.
2- As opções 2 a 6 só podem ser executadas se a carga do arquivo TXT já tiver sido realizada (opção 1).
3- Use a implementação da ABB realizada em sala de aula e acrescente novos métodos, se necessário.
4- Devem existir, no mínimo, 4 classes: Palavra, Node, BinarySearchTree e a classe principal.

Entrega e apresentação:

Os projetos deverão ser enviados pelo Moodle (apenas uma entrega por grupo).

Adicionalmente, cada grupo deverá gravar um vídeo de no máximo, 5 minutos, apresentando a solução e o funcionamento do código (todos os integrantes devem participar da apresentação e deixar claro a sua contribuição no projeto). O link do vídeo deve ser inserido na opção 7 do menu.

Comparação lexicográfica

compareTo(String)

Este método recebe como parâmetro um dado tipo String que será comparado com outro dado do tipo String e retorna um valor inteiro: **0 (zero)**, se as strings forem exatamente iguais; **um número negativo**, se a String passada como parâmetro for maior (lexicograficamente) que a string que chamou o método ou **um número positivo**, se a String passada como parâmetro for menor (lexicograficamente) que a string que chamou o método. Este método considera as diferenças existentes entre letras maiúsculas e minúsculas. Veja o exemplo abaixo, o resultado é um valor positivo porque na ordem alfabética, `Anacleto` é menor que `Zé`.

```
String nome1 = "Zé";
String nome2 = "Anacleto";
int result = nome1.compareTo(nome2);
System.out.println("Resultado = " + result);
```

compareToIgnoreCase(String)

Este método funciona como o anterior, porém, desconsideras as diferenças entre letras maiúsculas e minúsculas.

Critérios de avaliação:

Carregar dados do arquivo TXT – até 1,5 ponto
Contador de palavras - até 1 ponto
Busca de palavra - até 1,5 ponto
Exibir palavras em ordem alfabética – até 1 ponto
Verificar sinais de depressão – até 2 pontos
Estatísticas – até 1,5 ponto
Legibilidade do código – até 0,5 ponto
Vídeo do projeto – até 1 ponto

Itens para redução da nota:

O projeto é cópia de outro projeto ➔ Projeto é zerado

O projeto não usa a linguagem Java ➔ Projeto é zerado

O projeto usa recursos de programação/Java ainda não estudados ➔ Projeto é zerado

Não seguiu alguma definição do projeto ➔ -1 ponto para cada definição não seguida

Integrante do grupo que não está presente no vídeo ➔ Projeto é zerado para o integrante que não apresentou.

O projeto usa estruturas prontas do Java para implementação das estruturas e/ou para a ordenação dos dados ➔ -5,0 pontos

Há erros de compilação e/ou o programa trava/"quebra" durante a execução ➔ -1,0 ponto

Atraso na entrega do projeto via Moodle ➔ -2,0 ponto por dia de atraso.