Algoritmos e Estruturas de Dados I – 2023/2

Trabalho II

Entrega em 09/11/2023 até às 11:15, via Moodle

Prof. Marcio Sarroglia Pinho

Este trabalho **pode ser feito em dupla ou individualmente**.

A ideia é criar uma estrutura que mantenha um conjunto ordenado de palavras, agrupando-as com base na letra inicial das palavras.

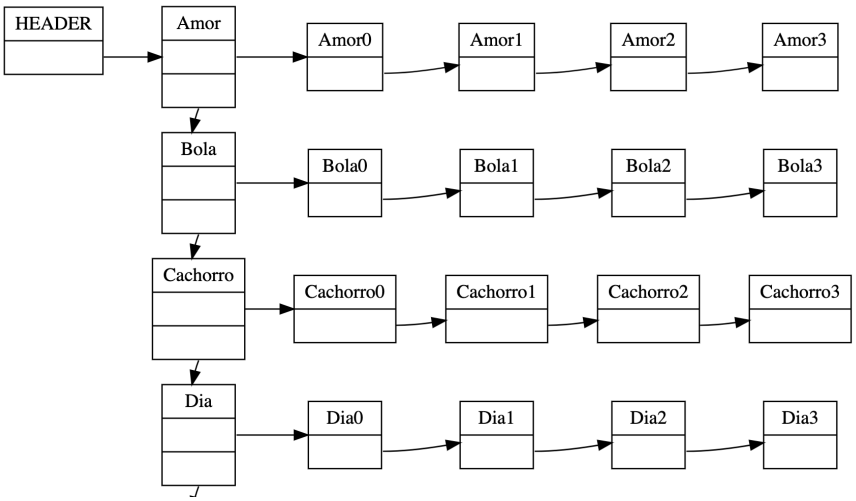
Está sendo disponibilizado um código base, que é de **uso obrigatório.**

Para compilar este código, inclua os fontes **DoubleLinkedListOfHeaders.java**, **HashCreator.java** e **Main.java** em seu projeto.

Garanta ainda que o arquivo de dados (*.txt) a ser lido pelo programa esteja na mesma pasta onde o programa for executado. A definição de qual arquivo será lido está na classe Main.java.

O programa deve realizar a leitura de um arquivo texto contento uma palavra em cada linha e, na medida em que for lendo, deve inserir a palavra na estrutura.

A estrutura se assemelha a imagem a seguir.



A parte referente à leitura do arquivo e à montagem da lista com a primeira palavra de cada letra já está disponível no código fornecido pelo professor e **é de uso obrigatório**.

Este código contém as classes **DoubleLinkedListOfHeaders.java** e **Main.java** que **são de uso obrigatório**. Os comentários contidos nos fontes são parte integrante da definição do trabalho.

O ponto de partida para a execução do trabalho é o entendimento deste fontes.

Toda a manipulação da classe **DoubleLinkedListOfHeaders** deverá ser feita a partir da classe **Main**, através da chamada de métodos.

O professor irá disponibilizar códigos com versões diferentes da classe Main para teste e geração do relatório final. Já estão disponíveis arquivos com listas de palavras.

A parte principal do código está na classe **DoubleLinkedListOfHeaders**.

Esta classe implementa uma lista duplamente encadeada de nodos da classe **Node**, apresentada a seguir.

```
private class Node {
    public String element; // primeira palavra que inicia com uma certa letra
    NodePalavra PalavrasComMesmaInicial; // Lista com as demais palavras da letra
    public Node next;
    public Node prev;

    public Node(String e) {
        element = e;
        next = null;
        prev = null;
        PalavrasComMesmaInicial = null;
    }
}
```

A classe **Node** armazena, no atributo **element**, a primeira palavra que inicia com uma certa letra do alfabeto. Já o atributo **PalavrasComMesmaInicial** é uma referência para a próxima palavra que inicia com a mesma letra que a palavra armazenada em **element**.

Esta referência deve ser entendida como o início da uma lista de nodos da classe **NodePalavra**. A manipulação dos elementos desta lista é um dos pontos principais do trabalho.

```
private class NodePalavra {
     public String element;
     public NodePalavra next;

     public NodePalavra(String e) {
         element = e;
         next = null;
     }
}
```

Para a execução do trabalho, as alterações devem ser feitas a partir do método **void addIncreasingOrder(String palavra)**, conforme as instruções contidas no código do método.

O código fornecido já insere uma palavra em ordem na lista de nodos da classe **Node**. Entretanto a inserção de uma nova palavra que inicie com a mesma letra ainda não está implementada.

Deverá ser adicionado ainda um método de remoção de uma palavra, conforme o exemplo a seguir.

```
void Remove(String palavra)
{

}
```

Métodos de Impressão da Estrutura

O código disponibilizado já contém dois métodos para imprimir a estrutura.

O método **void ImprimeLista()** faz a impressão conforme o exemplo a seguir.

```
A Lista:
**  Amor, Amor0, Amor1, Amor2, Amor3
**  Bola, Bola0, Bola1, Bola2, Bola3
**  Cachorro, Cachorro0, Cachorro1, Cachorro2, Cachorro3
**  Dia, Dia0, Dia1, Dia2, Dia3
....
HASH: c5cf5298936687b7faa39f2b7b3369ee
```

Já o método **void GeraDOT()** cria uma descrição no formato DOT que pode ser visualizado no site http://www.webgraphviz.com/. Também é possível instalar o software **GraphViz** localmente em um computador. Esta descrição foi usada para gerar a imagem apresentada no início deste documento.

Entrega do Trabalho

Para a entrega do trabalho deve ser gerado um relatório contendo as saídas gerada pelo programa.

Uma das saídas é a imagem gerada pelo WebGraphViz. A outra é a lista gerada pelo método **void ImprimeLista()**. Esta saída deve conter o **hash** da estrutura. A geração do código hash já é feita no método **void ImprimeLista()** na classe **DoubleLinkedListOfHeaders**, disponibilizada pelo professor.
A montagem da estrutura será considerada 100% correta somente se o código HASH gerado pelo programa for o mesmo gerado pelo código fornecido pelo professor.

Deve ser gerado também **um vídeo** de, **no máximo,** 2(dois) minutos com a explicação dos métodos que inserem em removem os dados da estrutura. Este vídeo deve ser gravado pelo Zoom, com narração feita pelos alunos. O vídeo deve ser colocado em uma plataforma de streaming, como Youtube ou Vimeo e **um link para o vídeo deve ser fornecido na primeira linha do relatório.**