ALGORITMOS E ESTRUTURAS DE DADOS // TABELA HASH (GUIA DE INTEGRAÇÃO)

-LEONARDO MAMEDE RODRIGUES
2022

Objetivo:

-Crie uma tabela hash para armazenar palavras de um arquivo

- Crie um método para carregar palavras de um arquivo fornecido por parâmetro. Este arquivo pode conter mais de 1 milhão de palavras.

• A partir do arquivo fornecido, a tabela deve armazenar somente as palavras que contenham exatamente 5 letras.
• Crie duas operações na sua tabela, com as assinaturas e atribuições especificadas a seguir : sortearPalavra() & existePalavra().

Visã Geral:

Classe principal

App:

*Carregar/armazenar palavras com 5 letras na tabela hash e exibir uma palavra aleatoria para o no console.

Métodos da classe:

+carregarPalavras(Hashtable) - return-> void
//ler arquivo e inserir na tabela

+mostrarPalavraDoDia(Hashtable) - return -> void
//exibir palavra aleatoria

Classes Auxiliares

StopWord:

*Classe que gera objeto para armazenar String(palavra).

Atributos: palavra(String).

Métodos da classe:

+construtor(String termo) - return-> void

+getPalavra() - return -> String
//retorna valor do atributo Termo

Entrada:

*Sentinelas/Entradas da tabela.

Atributos: Chave(String), palavra(StopWord), valido(boolean)

Métodos da classe:

+construtorVazio() →{

chave = " "

palavra = null

valido = false

}

+construtor(String palavra, StopWord termo) →{

chave = palavra

palavra = termo

valido = true

}

+isValido()→{
->retorna atributo "valido" do objeto
→ return: boolean

}

+setValido() →{
this.valido = true;
}

+unsetValido() →{
this.valido = false;
}

HashTable:

*Estrutura de dados que utiliza chave e código para funcionamento

Definição de Atributos da classe:

TAMANHO :

*TIPO INTEIRO QUE ARMAZENA O TAMANHO DA TABELA.

- TAMANHO = 124.000;

TAMANHO ESCOLHIDO APÓS VÁRIOS TESTES É UM NÚMERO RELATIVAMENTE BOM PARA O TAMANHO DE PALAVRAS QUE SERÃO CADASTRADA - > APROXIMADAMENTE(20.000)

***DADOS**:*

VETOR QUE ARMAZENA OBJETOS DA CLASSE *ENTRADA*.

- O TAMANHO DO VETOR É DEFINIDO APARTIR DO ATRIBUTO "TAMANHO" DA CLASSE HASHTABLE - >

-Em cada posição será armazenada um objeto Entrada que armazena a chave da palavra e a palavra.

→ *Entrada[] dados = new Entrada(tamanho);*

PESOS:

VETOR DE INTEIROS QUE ARMAZENA PESOS PARA SEREM UTILIZADOS NA HORA DE CALCULAR O HASH

- TAMNHO DO VETOR = 5;

Métodos da classe:

calcular Código(String chave) → {

***Receber chave(String)**

***Extrair cada char da chave e reitrando seu código equivalente na tebela ASC e multiplicando por um dos valores no vetor de pesos**

*** Somar todos os resultados das multiplicações**

***Retornar o código calculado**

hashCode += charCode * peso[x];

}

codigoHash2(int codigoHash)→{

***Receber código calculado do método anterior(codigo hash)**

***Realiza a operação :**

-> codigoHash / 12;

***Retornar o resultado da operação;**

}

mapear(int codigo)→{

***Receber código calculado do método anterior**

***Mapear a posição referente a tabela**

***Utilizando que posição mapeada será igual ao MOD da chave pelo tamanho da tabela**

***Retornar a posição mapeada(int)**

posMapeada = codigo % N -> n t=tamanho da tabela

}

localizar(String palavra)→{

***Localização da posição na tabela utilza um principio de tratamento de colisões e**

***Recebe parâmetro de String chave(palavra);**

Calcula seu *hash(codigo1) baseado na chave e armazena o valor;

****Mapeia e armazena o código gerado a cima ;***

****Enquanto a entrada da posição mapeada possuir seu estado de vazio equivalente a "true" no vetor de Entradas da tabela e a chave procurada for diferente da chave na posição, o programa***

continuar mapeando uma nova posição até encontrar um local apropriado na tabela baseado no indicie de colisões com o segundo hash

retorna a posição localizada;

pos = mapear(pos + (indiceSondagem *segundoHash(calcHash)));

}

inserir(String chave, StopWord objeto)→{

RECEBER CHAVE E OBJETO (VALOR) PARA INSERÇÃO NA TABELA;
CRIAR UMA NOVA ENTRADA PASSANDO A CHAVE E O OBJETO

LOCALIZAR A CHAVE

iNSERIR NOVA ENTRADA NA POSIÇÃO LOCALIZADA

RETURN: VOID;

}

buscarPalavra(String posição)→{

RECEBE A CHAVE COMO PARÂMETRO (PALAVRA/ STRING)
LOCALIZA A POSIÇÃO

SE O VETOR DE DADOS NAQUELA POSIÇÃO LOCALIZADA FOR VALIDO

RETORNO DA VARIAVEL DE CONTROLE COM VALOR TRUE SE NÃO RETORNA FALSO

RETURN: BOOLEAN

}

gerarPalavra()→{

UTILIZA A CLASSE RANDOM DO JAVA PARA GERAR VALORES ALEATORIOS QUE SÃO USADOS EM UM CALCULO O RESULTADO DESSE CALCULO É PASSADO COMO POSIÇÃO DO VETOR DE DADOS DA TABELA
SE O VALOR REFERENTE A POSIÇÃO DA TABELA FOR VALIDO / ESTIVER EM USO O MÉTODO RETORNA O VALOR REFERENTE(STRING) / PALAVRA DA POSIÇÃO

SE NÃO A CONTA CONTINUA SENDO REALIZADA GERANDO NOVO NÚMEROS RANDOMICOS

RETURN:STRING

}

preencherPesos(int[] pesos)→{

RECEBE O ENDEREÇO DE MEMORIA DO VETOR DE PESOS(ATRIBUTO DA CLASSE)
***LAÇO DE REPETIÇÃO PARA PREENCHER O VETOR COM VALORES DE -> I+2 NA POSIÇÃO I -> I=0; I<TAMANHO DO VETOR; I++;**

RETURN: VOID

}