

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO
FACULDADE DE CIÊNCIAS EXATAS E ENGENHARIA

RUAN SOARES DA SILVA FONSECA
GUILHERME DE OLIVEIRA VIEIRA BISPO
DANYLO HENRIQUE DA SILVA DOS SANTOS

TRABALHO DE ESTRUTURA DE DADOS 1

RIO DE JANEIRO 2023

INTRODUÇÃO

O consiste em uma aplicação escrita em java , que utiliza a estrutura de lista para armazenar palavras buscadas em um arquivo TXT . Ordenando – as com o auxílio do algoritmo de ordenação quicksort.

Listas essas segmentadas em duas abordagens de implementação diferentes, uma duplamente encadeadas e outra simplesmente encadeadas. Nesse trabalho, foram abordadas técnicas de programação apresentadas nas aulas da Disciplina de Estrutura de Dados 1 , pelo professor Dr Denis Cople.

CÓDIGO CORRESPONDENTE A LISTA SIMPLEMENTE ENCADEADA

```
package listaEncadeada;

public class listaEncadeada {
Node head;

public listaEncadeada() {
head = null;
}

public void insert(String word, int count) {
Node newNode = new Node(word, count);
newNode.next = head;
head = newNode;
}

public Node search(String word) {
Node current = head;
int aux = 0;
String palavra;
while (current != null) {
if (current.word.equals(word)) {
aux++;
}
current = current.next;
}
if(aux!=0)
    aux = aux - 1;
System.out.println("A Palavra : " + word + " tem "+ aux +" ocorrencias no arquivo ");
```

```
return null;
```

```
}
```

```
public void printList() {
```

```
Node current = head;
```

```
while (current != null) {
```

```
System.out.println(current.word + ": " + current.count);
```

```
current = current.next;
```

```
}
```

```
}
```

```
public String[] getWords() {
```

```
// Conta o número de palavras na lista
```

```
int count = 0;
```

```
Node current = head;
```

```
while (current != null) {
```

```
    count++;
```

```
    current = current.next;
```

```
}
```

```
// Cria um array de palavras
```

```
String[] words = new String[count];
```

```
// Preenche o array com as palavras da lista
```

```
current = head;
```

```
int i = 0;
```

```
while (current != null) {
```

```
    words[i] = current.word;
```

```
    i++;
```

```
    current = current.next;
```

```

    }

    return words;
}

/*****
***

public void quickSort() {
    head = quickSortRecursive(head, null);
}

private Node quickSortRecursive(Node start, Node end) {
    // caso base: lista vazia ou com um elemento
    if (start == end || start.next == end) {
        return start;
    }

    Node pivot = start;
    Node current = start.next;
    Node previous = start;

    while (current != end) {
        if (current.word.compareTo(pivot.word) < 0) {
            // move o nó atual para o início da lista
            previous.next = current.next;
            current.next = start;
            start = current;
            current = previous.next;
        } else {
            // avança para o próximo nó
            previous = current;
            current = current.next;
        }
    }
}

```

```
}  
}
```

```
// divide a lista em duas partes e as ordena recursivamente
```

```
start = quickSortRecursive(start, pivot);
```

```
pivot.next = quickSortRecursive(pivot.next, end);
```

```
return start;
```

```
}
```

```
/*
```

```
public static void quickSort(String[] words, int left, int right) {
```

```
    if (left < right) {
```

```
        int pivotIndex = partition(words, left, right);
```

```
        quickSort(words, left, pivotIndex);
```

```
        quickSort(words, pivotIndex + 1, right);
```

```
    }
```

```
}
```

```
private static int partition(String[] words, int left, int right) {
```

```
    String pivot = words[right];
```

```
    int i = left - 1;
```

```
    for (int j = left; j < right; j++) {
```

```
        if (words[j].compareTo(pivot) < 0) {
```

```
            i++;
```

```
            String temp = words[i];
```

```
            words[i] = words[j];
```

```
            words[j] = temp;
```

```
        }
```

```
}
```

```

        String temp = words[i + 1];
        words[i + 1] = words[right];
        words[right] = temp;
        return i + 1;
    }
}

/**
/*****
***

public void quickSort2() {
    head = quickSortRecursive2(head, null);
}

private Node quickSortRecursive2(Node start, Node end) {
    // caso base: lista vazia ou com um elemento
    if (start == end || start.next == end) {
        return start;
    }

    Node pivot = start;
    Node current = start.next;
    Node previous = start;

    while (current != end) {
        if (current.count < pivot.count) {
            // move o nó atual para o início da lista
            previous.next = current.next;
            current.next = start;
            start = current;
            current = previous.next;
        } else {
            // avança para o próximo nó

```

```

previous = current;
current = current.next;
}
}

// divide a lista em duas partes e as ordena recursivamente
start = quickSortRecursive2(start, pivot);
pivot.next = quickSortRecursive2(pivot.next, end);

return start;
}

}

package listaEncadeada;

public class Node {
String word;
int count;
Node next;

public Node(String word, int count) {
this.word = word;
this.count = count;
this.next = null;
}
}

package listaEncadeada;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;

```



```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        listaEncadeada list = new listaEncadeada();
        File file = new File(
            "C:\\Users\\Ruan\\eclipse-
            workspace\\TrabalhoAv1\\src\\listaDuplamenteEncadeada\\ListadePalavras.txt");

        BufferedReader reader = null;

        int qtd = 0;

        try {
            reader = new BufferedReader(new FileReader(file));
            String line;

            while ((line = reader.readLine()) != null) {

                list.insert(line, 0);

            }
        } catch (IOException e) {

            e.printStackTrace();
        } finally {
```

```
try {
    if (reader != null) {
        reader.close();
    }
} catch (IOException e) {
    e.printStackTrace();
}

}

countWords(
"C:\\Users\\Ruan\\eclipse-
workspace\\TrabalhoAv1\\src\\listaDuplamenteEncadeada\\ListadePalavras.txt",
list);

Menu(list);
}

public static void Menu(listaEncadeada list) {
    Double op;
    String opcao;
    Scanner input2 = new Scanner(System.in);
    String palavraEscolhida;
    Scanner input = new Scanner(System.in);

    System.out.println("Escolha uma opcao abaixo : ");
    System.out.println("Ver Lista [1] ");
    System.out.println("Pesquisar na Lista [2] ");
    System.out.println("Sair [0] : ");
    opcao = input.nextLine();
```

```

if (opcao.equals("1")) {
    verLista(list);
} else if (opcao.equals("2")) {
    System.out.println("Digite a palavra : ");
    palavraEscolhida = input2.nextLine();
    pesquisarPalavra(palavraEscolhida, list);
} else if (opcao.equals("3")) {
    System.exit(0);
}

}

/*****
*****/

public static void verLista(listaEncadeada list) {

    System.out.println("*****Ordenada      por      Ordem      Alfabetica
*****");

    list.quickSort();

    list.printList();

    System.out.println("*****Ordenada por Ordem Quantidade Crescente
*****");

    list.quickSort2();

    list.printList();

    System.out.println("Voltando para funcao principal\n");

    Menu(list);

}

public static void countWords(String filename, listaEncadeada list) {
    Map<String, Integer> wordCounts = new HashMap<>();

```

// Inicializa o mapa com as palavras da lista e contagens iniciais de 0

```
for (String word : list.getWords()) {  
wordCounts.put(word, 0);  
}
```

File file = new File(filename);

BufferedReader reader = null;

try {

reader = new BufferedReader(new FileReader(file));

String line;

while ((line = reader.readLine()) != null) {

String[] words = line.split(" ");

for (String word : words) {

if (wordCounts.containsKey(word)) {

int count = wordCounts.get(word);

wordCounts.put(word, count + 1);

}

}

}

// Imprime as contagens de palavras

for (Map.Entry<String, Integer> entry : wordCounts.entrySet()) {

list.insert(entry.getKey(), entry.getValue());

// System.out.println(entry.getKey() + ": " + entry.getValue());

}

} catch (IOException e) {

e.printStackTrace();

} finally {

```

try {
if (reader != null) {
reader.close();
}
} catch (IOException e) {
e.printStackTrace();
}
}
}

public static void pesquisarPalavra(String nome, listaEncadeada list) {

list.search(nome);
System.out.println("Voltando para funcao principal\n");
Menu(list);

}

}

```

CÓDIGO CORRESPONDENTE A LISTA DUPLAMENTE ENCADEADA

```

package listaDuplamenteEncadeada;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

```

```
import java.util.Scanner;
```

```
import listaEncadeada.listaEncadeada;
```

```
public class Principal {
```

```
public static void main(String[] args) {
```

```
ListaDuplamenteEncadeada lista = new ListaDuplamenteEncadeada();
```

```
File file = new File(
```

```
"C:\\Users\\Ruan\\eclipse-  
workspace\\TrabalhoAv1\\src\\listaDuplamenteEncadeada\\ListadePalavras.txt");
```

```
BufferedReader reader = null;
```

```
int qtd = 0;
```

```
try {
```

```
reader = new BufferedReader(new FileReader(file));
```

```
String line;
```

```
while ((line = reader.readLine()) != null) {
```

```
lista.adicionarNoInicio(line, 0);
```

```
}
```

```
} catch (IOException e) {
```

```
e.printStackTrace();
```

```
} finally {
```

```
try {
```

```
if (reader != null) {
```

```

reader.close();
}
} catch (IOException e) {
e.printStackTrace();
}

}

countWords(
"C:\\Users\\Ruan\\eclipse-
workspace\\TrabalhoAv1\\src\\listaDuplamenteEncadeada\\ListadePalavras.txt",
lista);

menu(lista);

/*****
*****/

}

public static void menu(ListaDuplamenteEncadeada lista) {
Double op;
String opcao;
Scanner input2 = new Scanner(System.in);
String palavraEscolhida;
Scanner input = new Scanner(System.in);

System.out.println("Escolha uma opcao abaixo : ");
System.out.println("Ver Lista [1] ");
System.out.println("Pesquisar na Lista [2] ");
System.out.println("Sair [0] : ");
opcao = input.nextLine();

if (opcao.equals("1")) {
verLista(lista);

```

```

} else if (opcao.equals("2")) {
    System.out.println("Digite a palavra : ");
    palavraEscolhida = input2.nextLine();
    pesquisarPalavra(palavraEscolhida, lista);
} else if (opcao.equals("3")) {
    System.exit(0);
}

}

public static void pesquisarPalavra(String nome, ListaDuplamenteEncadeada lista) {

    lista.pesquisarPalavra(nome);
    System.out.println("Voltando para funcao principal\n");
    menu(lista);

}

public static void verLista(ListaDuplamenteEncadeada lista) {

    System.out.println("*****Ordenada      por      Ordem      Alfabetica
*****");
    lista.ordenarPorOrdemAlfabetica();
    lista.mostrarLista();

    System.out.println("*****Ordenada por Ordem Quantidade Crescente
*****");
    lista.ordenarPorValor();
    lista.mostrarLista();

    System.out.println("Voltando para funcao principal\n");

```



```
menu(lista);
```

```
}
```

```
public static void countWords(String filename, ListaDuplamenteEncadeada lista) {
```

```
    Map<String, Integer> wordCounts = new HashMap<>();
```

```
    File file = new File(filename);
```

```
    BufferedReader reader = null;
```

```
    try {
```

```
        reader = new BufferedReader(new FileReader(file));
```

```
        String line;
```

```
        while ((line = reader.readLine()) != null) {
```

```
            String[] words = line.split(" ");
```

```
            for (String word : words) {
```

```
                if (wordCounts.containsKey(word)) {
```

```
                    int count = wordCounts.get(word);
```

```
                    wordCounts.put(word, count + 1);
```

```
                } else {
```

```
                    wordCounts.put(word, 1);
```

```
                }
```

```
            }
```

```
        }
```

```
// Adiciona as contagens de palavras na lista duplamente encadeada
```

```
for (Map.Entry<String, Integer> entry : wordCounts.entrySet()) {
```

```
    lista.adicionarNoInicio(entry.getKey(), entry.getValue());
```

```

        // System.out.println(entry.getKey() + ": " + entry.getValue());
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        if (reader != null) {
            reader.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
}

package listaDuplamenteEncadeada;

public class Node {
    String palavra;
    int valor;
    Node anterior;
    Node proximo;

    public Node(String palavra, int valor, Node anterior, Node proximo) {
        this.palavra = palavra;
        this.valor = valor;
        this.anterior = anterior;
        this.proximo = proximo;
    }
}

```

```

package listaDuplamenteEncadeada;

```

```
public class ListaDuplamenteEncadeada {  
    Node primeiro;  
    Node ultimo;  
  
    public void adicionarNoInicio(String palavra, int valor) {  
        Node novoNo = new Node(palavra, valor, null, primeiro);  
        if (primeiro == null) {  
            ultimo = novoNo;  
        } else {  
            primeiro.anterior = novoNo;  
        }  
        primeiro = novoNo;  
    }  
  
    public int contarPalavras() {  
        int contagem = 0;  
        Node atual = primeiro;  
        while (atual != null) {  
            if (atual.palavra != null && !atual.palavra.isEmpty()) {  
                contagem++;  
            }  
            atual = atual.proximo;  
        }  
        return contagem;  
    }  
}
```

```
public void mostrarLista() {  
    Node atual = primeiro;  
    int qtd = 0;  
    while (atual != null) {
```

```

        System.out.println(atual.palavra + ": " + atual.valor);
        atual = atual.proximo;
    }
}

public Node pesquisarPalavra(String palavra) {
    Node atual = primeiro;
    int qtd = 0;
    while (atual != null) {
        if (atual.palavra != null && atual.palavra.equalsIgnoreCase(palavra)) {
            qtd++;
        }
        atual = atual.proximo;
    }
    if(qtd!=0)
        qtd = qtd - 1;
    System.out.println("A Palavra : " + palavra + " tem "+ qtd +" ocorrencias no arquivo
");
    return null;
}

public void adicionarNoFinal(String palavra, int valor) {
    Node novoNo = new Node(palavra, valor, ultimo, null);
    if (ultimo == null) {
        primeiro = novoNo;
    } else {
        ultimo.proximo = novoNo;
    }
    ultimo = novoNo;
}

```

```
public void removerDoInicio() {  
    if (primeiro != null) {  
        primeiro = primeiro.proximo;  
        if (primeiro == null) {  
            ultimo = null;  
        } else {  
            primeiro.anterior = null;  
        }  
    }  
}
```

```
public void removerDoFinal() {  
    if (ultimo != null) {  
        ultimo = ultimo.anterior;  
        if (ultimo == null) {  
            primeiro = null;  
        } else {  
            ultimo.proximo = null;  
        }  
    }  
}
```

```
public void ordenarPorValor() {  
    // Verifica se a lista possui pelo menos dois nós  
    if (primeiro != null && primeiro.proximo != null) {  
        quickSort(primeiro, ultimo);  
    }  
}
```

```
private void quickSort(Node inicio, Node fim) {  
    if (inicio != null && fim != null && inicio != fim && inicio != fim.proximo){
```

```
Node pivo = particionar(inicio, fim);
quickSort(inicio, pivo.anterior);
quickSort(pivo.proximo, fim);
}
}
```

```
private Node particionar(Node inicio, Node fim) {
    int valorPivo = fim.valor;
    Node i = inicio.anterior;
    for (Node j = inicio; j != fim; j = j.proximo) {
        if (j.valor < valorPivo) {
            i = (i == null) ? inicio : i.proximo;
            int valor = i.valor;
            String palavra = i.palavra;
            i.valor = j.valor;
            i.palavra = j.palavra;
            j.valor = valor;
            j.palavra = palavra;
        }
    }
    i = (i == null) ? inicio : i.proximo;
    int valor = i.valor;
    String palavra = i.palavra;
    i.valor = fim.valor;
    i.palavra = fim.palavra;
    fim.valor = valor;
    fim.palavra = palavra;
    return i;
}
```

```
public void ordenarPorOrdemAlfabetica() {
```

```
// Verifica se a lista possui pelo menos dois nós
if (primeiro != null && primeiro.proximo != null) {
    quickSort2(primeiro, ultimo);
}
}
```

```
private void quickSort2(Node inicio, Node fim) {
    if (fim == null) {
        // A lista possui apenas um elemento ou está vazia, não precisa fazer nada
        return;
    }
    if (inicio != fim && inicio != fim.proximo) {
        Node pivo = particionar2(inicio, fim);
        quickSort2(inicio, pivo.anterior);
        quickSort2(pivo.proximo, fim);
    }
}
```

```
private Node particionar2(Node inicio, Node fim) {
    if (inicio == null) {
        // A lista possui apenas um elemento ou está vazia, não tem o que particionar
        return null;
    }
    String palavraPivo = fim.palavra;
    Node i = inicio.anterior;
    for (Node j = inicio; j != fim; j = j.proximo) {
        if (j.palavra.compareTo(palavraPivo) < 0) {
            i = (i == null) ? inicio : i.proximo;
            int valor = i.valor;
            String palavra = i.palavra;
            i.valor = j.valor;
```

```
i.palavra = j.palavra;  
j.valor = valor;  
j.palavra = palavra;  
}  
}  
i = (i == null) ? inicio : i.proximo;  
int valor = i.valor;  
String palavra = i.palavra;  
i.valor = fim.valor;  
i.palavra = fim.palavra;  
fim.valor = valor;  
fim.palavra = palavra;  
return i;  
}  
}
```

PRINT'S REFERENTE AO SEU FUNCIONAMENTO

Lista encadeada


```
Console X Problems Debug Shell TCP/IP Monitor
Main (1) [Java Application] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe (11 de jan. de 2023 15:59:19) [pid: 16564]
abacaxi: 1
abacates: 1
abacateiros: 1
abacateiro: 1
abacate: 1
aara: 1
a: 1
: 2
gamer: 2
lala: 2
lua: 6
aba: 8
Voltando para funcao principal

Escolha uma opcao abaixo :
Ver Lista [1]
Pesquisar na Lista [2]
Sair [0] :
2
Digite a palavra :
lua
A Palavra : lua tem 6 ocorrencias no arquivo
Voltando para funcao principal

Escolha uma opcao abaixo :
Ver Lista [1]
Pesquisar na Lista [2]
Sair [0] :
```

Lista Duplamente Encadeada

```
Console X Problems Debug Shell TCP/IP Monitor
Principal [Java Application] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe (11 de jan. de 2023 16:02:48) [pid: 2316]
Escolha uma opcao abaixo :
Ver Lista [1]
Pesquisar na Lista [2]
Sair [0] :
1
*****Ordenada por Ordem Alfabetica *****
: 0
: 2
: 0
a: 0
a: 1
aara: 0
aara: 1
aba: 0
aba: 0
aba: 0
aba: 0
aba: 0
aba: 0
aba: 0
aba: 0
aba: 8
aba: 0
abacate: 1
abacate: 0
abacateiro: 1
abacateiro: 0
abacateiros: 1
abacateiros: 0
abacates: 1
```

```
Principal [Java Application] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe (11 de jan. de 2023 16:02:48) [pid: 2316]
villipendiosas: 1
villipendiosas: 1
villipendiosas: 0
villipendiosas: 0
villipendioso: 1
*****Ordenada por Ordem Quantidade Crescente *****
villipendioso: 0
: 0
: 0
a: 0
aaraao: 0
aba: 0
aba: 0
aba: 0
aba: 0
aba: 0
aba: 0
aba: 0
abacate: 0
abacateiro: 0
abacateiros: 0
abacates: 0
abacaxi: 0
abacaxis: 0
abaciais: 0
abacial: 0
abaco: 0
abacos: 0
abade: 0
```

```
Principal [Java Application] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe (11 de jan. de 2023 16:02:48) [pid: 2316]
gasometro: 1
vilanias: 1
gelasseis: 1
vilao: 1
gelassem: 1
vilarejo: 1
garoaremos: 1
vilarejos: 1
vilas: 1
garimpamos: 1
vileza: 1
gelassemos: 1
vilezas: 1
gasometros: 1
gelasses: 1
villipendiosa: 1
villipendiosas: 1
gelaste: 1
: 2
gemer: 2
lala: 2
lua: 6
aba: 8
Voltando para funcao principal

Escolha uma opcao abaixo :
Ver Lista [1]
Pesquisar na Lista [2]
Sair [0] :
```

```
Principal [Java Application] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe (11 de jan. de 2023 16:06:41) [pid: 1224]
villipendiosas: 1
gelaste: 1
: 2
gemer: 2
lala: 2
lua: 6
aba: 8
Voltando para funcao principal

Escolha uma opcao abaixo :
Ver Lista [1]
Pesquisar na Lista [2]
Sair [0] :
2
Digite a palavra :
lua
A Palavra : lua tem 6 ocorrencias no arquivo
Voltando para funcao principal

Escolha uma opcao abaixo :
Ver Lista [1]
Pesquisar na Lista [2]
Sair [0] :
```

BIBLIOGRAFIA

<https://www.ime.usp.br/~cosen/verao/alg.pdf>

ACESSADO EM 09/01/2023