UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO

FACULTAD DE INGENIERIA Y CIENCIAS DE LA COMPUTACION Y TELECOMUNICACIONES

ESTRUCTURA DE DATOS II

CONTENIDO:

LAB.5. ABB,LISTA DE ABBs

PORCENTAJE TERMINADO: 100%

GRUPO: 14

INTEGRANTES	DT	HG	HI	EVAL
Flores Veizaga Eudenia Gandira	1	1	1	100
Garcia Taboada Brayan Albaro	1	1	1	100
Gedalge Cayhuara Cristian Gabriel	1	1	1	100
Haquin Serrano Rodrigo	1	1	1	100
Hernandez Lijeron Roly	1	1	1	100

//DT, días trabajados

//HG, horas grupo

//HI, horas individual

// Eval

Fecha de presentación : martes, 23 de mayo de 2024

Fecha Presentada : martes, 23 de mayo de 2024

Días de Atraso : 0

LISTA DE ÁRBOLES.

I. Arbol Binario de Búsqueda de Palabras.

Hacer un programa para leer un Archivo de Texto "Startup.txt", manipular las palabras, todos los caracteres en minúsculas. (Convertir todas las cadenas en minúsculas para tener mayor claridad en el análisis de los datos). Encontrar las siguientes salidas.

1. *Mostrar las palabras de menor a mayor con sus respectivas frecuencias.*
2. *Mostrar las palabras de mayor a menor con sus respectivas frecuencias.*
3. *Mostras las palabras de menor a mayor frecuencia.*
4. *Mostrar las palabras de mayor a menor frecuencia.*

```
public class Nodo {
    public String elem;
    public int frec;
    public Nodo izq;
    public Nodo der;

    public Nodo(String elem) {
        this.elem = elem;
        this.frec = 1;
        this.izq = this.der = null;
    }

    public Nodo(String elem, int frec) {
        this.elem = elem;
        this.frec = frec;
        this.izq = this.der = null;
    }
}
```

```
//-----------------------------------------
import java.io.*;
import java.util.StringTokenizer;

public class Arbol {
    public Nodo raiz;

    public Arbol() {
        raiz = null;
    }

    public Arbol(Arbol A1) {
        raiz = copiar(A1.raiz);
    }

    private Nodo copiar(Nodo p) {
        if (p == null) return null;
        Nodo q = new Nodo(p.elem, p.frec);
        q.izq = copiar(p.izq);
        q.der = copiar(p.der);
        return q;
    }

    public void insertar(String elem) {
        raiz = insertarRec(raiz, elem);
    }

    private Nodo insertarRec(Nodo raiz, String elem) {
```

```
    if (raiz == null) {
        raiz = new Nodo(elem);
        return raiz;
    }
    if (elem.compareTo(raiz.elem) < 0) {
        raiz.izq = insertarRec(raiz.izq, elem);
    } else if (elem.compareTo(raiz.elem) > 0) {
        raiz.der = insertarRec(raiz.der, elem);
    } else {
        raiz.frec++;
    }
    return raiz;
}

public void inOrden(Nodo nodo) {
    if (nodo != null) {
        inOrden(nodo.izq);
        System.out.println(nodo.elem + ": " + nodo.frec);
        inOrden(nodo.der);
    }
}

public void inOrdenDesc(Nodo nodo) {
    if (nodo != null) {
        inOrdenDesc(nodo.der);
        System.out.println(nodo.elem + ": " + nodo.frec);
        inOrdenDesc(nodo.izq);
    }
```

```
    }

    public void leerDesdeArchivo(String nombreArchivo) {
        try (BufferedReader br = new BufferedReader(new
FileReader(nombreArchivo))) {
            String linea;
            while ((linea = br.readLine()) != null) {
                StringTokenizer st = new StringTokenizer(linea.toLowerCase());
                while (st.hasMoreTokens()) {
                    insertar(st.nextToken());
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

//----------------------------------------
public class ArbolFrecuencia {
    public Nodo raiz;

    public ArbolFrecuencia() {
        raiz = null;
    }

    public void insertar(String elem, int frec) {
        raiz = insertarRec(raiz, elem, frec);
    }
```

```
private Nodo insertarRec(Nodo raiz, String elem, int frec) {
    if (raiz == null) {
        raiz = new Nodo(elem, frec);
        return raiz;
    }
    if (frec < raiz.frec || (frec == raiz.frec && elem.compareTo(raiz.elem) < 0)) {
        raiz.izq = insertarRec(raiz.izq, elem, frec);
    } else {
        raiz.der = insertarRec(raiz.der, elem, frec);
    }
    return raiz;
}

public void inOrden(Nodo nodo) {
    if (nodo != null) {
        inOrden(nodo.izq);
        System.out.println(nodo.elem + ": " + nodo.frec);
        inOrden(nodo.der);
    }
}

public void inOrdenDesc(Nodo nodo) {
    if (nodo != null) {
        inOrdenDesc(nodo.der);
        System.out.println(nodo.elem + ": " + nodo.frec);
        inOrdenDesc(nodo.izq);
    }
```

```
    }

    public void copiarDesdeArbol(Arbol arbol) {
        copiarRec(arbol.raiz);
    }

    private void copiarRec(Nodo nodo) {
        if (nodo != null) {
            insertar(nodo.elem, nodo.frec);
            copiarRec(nodo.izq);
            copiarRec(nodo.der);
        }
    }
}
```

En base a las fecuencias de palabras, concluir un análisis de contenido del archivo de texto.

II. Lista de Árboles Binarios de Búsqueda.

Encontrar un reporte de palabras de menor a mayor con sus respectivas frecuencias de ocurrencias ordenadas por la longitud de las palabras de menor a mayor.

Escribir un Análisis de Contenido del Archivo de Texto, en base a las frecuencias de palabras.

```
public class Nodo {
    public String cad;
    public int frec;
    public Nodo izq;
    public Nodo der;
    public Nodo(String cad )
    {
        this.cad=cad;
```

```
        this.frec=1;
        this.izq=this.der=null;
    }
    public Nodo(String cad,int frec){
        this.cad=cad;
        this.frec=frec;
        this.izq=this.der=null;
    }

}
public class Arbol {

    public Nodo raiz;
    public File archivo;

    public Arbol() {
        raiz = null;
    }

    public Arbol(Arbol A1) {
        raiz = copiar(A1.raiz);
    }

    public Nodo copiar(Nodo p) {
        if (p == null) {
            return null;
        }
        Nodo q = new Nodo(p.cad, p.frec);
```

```
        p.izq = copiar(p.izq);
        p.der = copiar(p.der);
        return q;
    }


    public void insertar(String x) {
          leerArchivodeTexto(x);
    }

    private Nodo insertar(Nodo p, String x) {
        if (p == null) {
            return new Nodo(x);

        }
        if (p.cad.equals(x)) {
            p.frec++;
        } else {
            if (x.length() > p.cad.length()) {
                p.izq = insertar(p.izq, x);
            } else {

                p.der = insertar(p.der, x);

            }
        }

        return p;
```

```
    }

    public void inordenMenMay() {
        inordenMenMay(raiz);
    }

    private void inordenMenMay(Nodo p) {
        if (p == null) {
            return;
        }
        inordenMenMay(p.izq);
        System.out.println(p.cad + " | " + p.frec);
        inordenMenMay(p.der);
    }

    public void inOrdenMayMen() {
        inOrdenMayMen(this.raiz);
    }

    private void inOrdenMayMen(Nodo p) {
        if (p == null) {
            return;
        }
        inOrdenMayMen(p.der);
        System.out.println(p.cad + " | " + p.frec);
        inOrdenMayMen(p.izq);
```

```
    }

    public void crearArchivoTexto() {
        this.archivo = new File("archivo.txt");

        try {
            if (archivo.createNewFile()) {
                System.out.println("Archivo creado");
            } else {
                System.out.println("Error al crear Archivo,ya hay uno creado");
            }
        } catch (Exception e) {

        }
    }

    public void eliminarArchivo() {
    }

    public void escribirArchivo(String texto) {
        try {
            FileWriter escritura = new FileWriter(this.archivo);
            escritura.write(texto);
            //escritura.write("\n estimado !!!");
            escritura.close();
        } catch (Exception e) {
        }
    }
```

```
    public void leerArchivodeTexto(String nameArchivo) {
        try {
            FileReader lector = new FileReader(nameArchivo);
            BufferedReader lectura = new BufferedReader(lector);
            String linea;

            while ((linea = lectura.readLine()) != null) {
                linea = linea.toLowerCase();
                StringTokenizer tokenizer = new StringTokenizer(linea);

                while (tokenizer.hasMoreTokens()) {
                    String palabra = tokenizer.nextToken();
                    if (palabra.matches("[a-zA-Z]+")) {
                        raiz = insertar(raiz, palabra);
                    }
                }
            }
            lectura.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    public static void main(String[] args) {
        Arbol A1 = new Arbol();
//        A1.generarElem(10, 1, 5);
```

```
//        System.out.println("--------------------------------");
//        A1.inOrdenMayMen();
//        A1.crearArchivoTexto();
//        A1.escribirArchivo("hola como estas mi amigo el amigo de mi amigo es mi enemigo");
        A1.insertar("Startup.txt");
        A1.inOrdenMayMen();
    }
}
```