*UNIVERSIDAD AUTÓNOMA GABRIEL RENE MORENO*
*FACULTAD DE INGENIERÍA Y CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES*

# ESTRUCTURAS DE DATOS 2

**CONTENIDO:** *// Breve Descripción del Contenido*

# TAREA-4. ABB. FRECUENCIA DE ELEMENTOS

**PORCENTAJE TERMINADO : 75%.**

**GRUPO:** 14



| Nombre | Registro |
|---|---|
| Cristian Gabriel Gedalge Cayhuara | 219022062 |



**Fecha de Presentación:** Lunes ,20 de mayo de 2024

**COMENTARIO:**
Fue un reto hacer el 2do ejercicio pero se pudo ,no entedi muy bien eso de lousar compareTo haci que no lo hice ,lo que hice fue un árbol de cadenas (palabras) que lee las palabras de un archivo.txt y lo ordeno de acuerdo a la primera letra de la cadena a partir de su valor ascci.

**Ejercicio-1.**
**Dado una secuencia de elementos, se desea encontrar la frecuencia de cada elementos.**

- ***Mostrar los elementos de menor a mayor con sus respectivas frecuencias de ocurrencias.***
- ***Mostrar los elementos de mayor  a menor con sus respectivas frecuencias de ocurrencias.***

CLASE NODO

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

CLASE ARBOL

```
public class Arbol {

    public Nodo raiz;

    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.elem, p.frec);
    p.izq = copiar(p.izq);
    p.der = copiar(p.der);
    return q;
}

public void generarElem(int n, int a, int b) {
    for (int i = n; i > 0; i--) {
        int x = (int) (Math.random() * (b - a)) + a;
        System.err.println(x);
        insertar(x);
    }
}

public void insertar(int x) {
    raiz = insertar(raiz, x);
}

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

    }
    if (p.elem == x) {
        p.frec++;
    } else {
        if (x < p.elem) {
            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.elem + " | " + 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.elem + " | " + p.frec);
    inOrdenMayMen(p.izq);

}
```

```
    public static void main(String[] args) {
        Arbol A1=new Arbol();
        A1.generarElem(15, 1, 7);
        System.out.println("--------------------------------");
        A1.inOrdenMayMen();
    }

}
```

**Crear otro ABB (nombre de clase diferente al anterior, e incluir métodos propios, que realicen lo requerido), con los elementos del árbol anterior, organizados por frecuencia de ocurrencia y elemento. (copiar los datos del árbol anterior, a este nuevo árbol)**

- ***Mostrar los elementos, asociados con sus frecuencias de menor a mayor.***
- ***Mostrar los elementos, asociados con sus frecuencias de mayor a menor.***

**Para los ejercicios anteriores, ejecutar los Algoritmos, generando n-elementos enteros entre a y b inclusive. Ejecutar para valores de n-grande.**

```
public class ABB {


    private Nodo raiz;

    public ABB() {
        raiz = null;
    }

    public ABB(ABB otroArbol) {
        raiz = copiar(otroArbol.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 generarElementos(int n, int a, int b) {
    for (int i = n; i > 0; i--) {
        int x = (int) (Math.random() * (b - a)) + a;
        insertar(x);
    }
}

public void insertar(int x) {
    raiz = insertar(raiz, x);
}

private Nodo insertar(Nodo p, int x) {
    if (p == null) {
        return new Nodo(x);
    }
    if (p.elem == x) {
        p.frec++;
    } else if (x < p.elem) {
        p.izq = insertar(p.izq, x);
    } else {
        p.der = insertar(p.der, x);
    }
    return p;
}

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

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

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

private void mostrarPorFrecuenciaDescendente(Nodo p) {
    if (p == null) {
        return;
```

```
        }
        mostrarPorFrecuenciaDescendente(p.der);
        System.out.println(p.elem + " | " + p.frec);
        mostrarPorFrecuenciaDescendente(p.izq);
    }

    public static void main(String[] args) {
        ABB arbol = new ABB();
        arbol.generarElementos(10, 1, 7);
        System.out.println("Elementos por frecuencia
ascendente:");
        arbol.mostrarPorFrecuenciaAscendente();
        System.out.println("-------------------------------
--");
        System.out.println("Elementos por frecuencia
descendente:");
        arbol.mostrarPorFrecuenciaDescendente();
    }

}
```

**Implementar el Ejercicios-1, con elementos Strings. Es decir; los elementos ya no son enteros, sino Cadenas de Caracteres, utilizar s1.compareTo(s2), para comparar dos cadenas de caracteres.**

**Para ejecutar con n-cadenas, para n-grande. Leer los datos de un Archivo de Texto y utilizar StringTokeizer(), para encontrar cada palabra del Archivo de Texto, para facilitar las consultas manipular todas las palabras en minúsculas.**

CLASE NODO

```
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;
    }
}
```

CLASE ARBOL

```
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 generarElem(int n, int a, int b) {
//        for (int i = n; i > 0; i--) {
//            int x = (int) (Math.random() * (b - a)) + a;
//            System.err.println(x);
//            insertar(x);
//        }
//    }
//    public void insertar(String x) {
//        raiz = insertar(raiz, x);
//    }
    public void insertar(String x) {
//        crearArchivoTexto();
//        escribirArchivo(x);
//        String linea = leerArchivodeTexto();
//        linea = linea.toLowerCase();
//        StringTokenizer tokenizer = new
StringTokenizer(linea);
//
//        while (tokenizer.hasMoreTokens()) {
//            //System.out.println(tokenizer.nextToken());
//            raiz = insertar(raiz, tokenizer.nextToken());
//        }
        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.charAt(0) < p.cad.charAt(0)) {
                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) {
        String linea = "";
        try {
            FileReader lector = new FileReader(nameArchivo);
            BufferedReader lectura = new
BufferedReader(lector);
            linea = lectura.readLine();

            while (linea != null) {
                //System.out.println(contenido);

                linea = linea.toLowerCase();
                StringTokenizer tokenizer = new
StringTokenizer(linea);

                while (tokenizer.hasMoreTokens()) {

//System.out.println(tokenizer.nextToken());
                    raiz = insertar(raiz,
tokenizer.nextToken());
                }
                linea=lectura.readLine();
            }
        } catch (Exception e) {
        }
        //return contenido;
    }

    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("archivo.txt");
        A1.inOrdenMayMen();
    }
}
```