UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO

FACULTAD DE INGENIERIA Y CIENCIAS DE LA COMPUTACION Y TELECOMUNICACIONES

ESTRUCTURA DE DATOS II

CONTENIDO:

TAREA-3. ABB CON LIBRERÍA DE LISTAS

PORCENTAJE TERMINADO: 100%

Grupo 14

Garcia Taboada Brayan Albaro

Fecha de presentación : Lunes , 20 de mayo de 2024

Fecha Presentada : Lunes , 20 de mayo de 2024

Días de Atraso : 0

FRECUENCIA DE ELEMENTOS.

Trabajo Individual.

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.*

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 Arbol {
   public Nodo raiz;
   public Arbol() {
      raiz = null;
   }
   public void insertar(int x){
      raiz = insertar(raiz,x);
   }
   private Nodo insertar (Nodo p, int x){
      if (p == null) {
         return new Nodo (x);
      }
      if (x < p.elem) {
         p.izq = insertar(p.izq,x);
      }else{
         p.der = insertar (p.der,x);
      }
      return p;
   }
   //Primera parte
   private void menorMayor(Nodo p) {
      if (p == null) {
```

```
        return;
    }
    menorMayor(p.izq);
    if (p.frec == 1) {
        System.out.println(p.elem + " | " + p.frec);
    } else {
        System.out.println(p.elem + " | " + p.frec);
    }
    menorMayor(p.der);
}
public void mayorMenor() {
    mayorMenor(raiz);
}
private void mayorMenor(Nodo p) {
    if (p == null) {
        return;
    }
    mayorMenor(p.der);
    if (p.frec == 1) {
        System.out.println(p.elem + " | " + p.frec);
    } else {
        System.out.println(p.elem + " | " + p.frec);
    }
    mayorMenor(p.izq);
}
//Segunda parte
public void ordFrecAsc(Arbol ar) {
    ordFrecAsc(ar, raiz);
    ordFrecAsc(ar.raiz);
}
private void ordFrecAsc(Arbol ar, Nodo p) {
    if (p == null) {
        return;
    }
    ordFrecAsc(ar, p.izq);
    ar.insertar(p.elem, p.frec);
    ordFrecAsc(ar, p.der);
}
private void ordFrecAsc(Nodo p) {
    if (p == null) {
        return;
    }
    ordFrecAsc(p.izq);
    System.out.println("[" + p.elem + ", " + p.frec + "]");
    ordFrecAsc(p.der);
```

```
    }
    public void ordFrecDes(Arbol ar) {
        ordFrecDes(ar, raiz);
        ordFrecDes(ar.raiz);
    }
    private void ordFrecDes(Arbol ar, Nodo p) {
        if (p == null) {
            return;
        }
        ordFrecDes(ar, p.der);
        ar.insertar(p.elem, p.frec);
        ordFrecDes(ar, p.izq);
    }
    private void ordFrecDes(Nodo p) {
        if (p == null) {
            return;
        }
        ordFrecDes(p.der);
        System.out.println("[" + p.elem + ", " + p.frec + "]");
        ordFrecDes(p.izq);
    }
}

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

    public Nodo(int x) {
        elem = x;
        izq = der = null;
        frec = 1;
    }
}
```

Ejercicio-2.

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.

```
public class ArbolString {
```

```
    public NodoString raiz;

    public ArbolString() {
        raiz = null;
    }

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

    private NodoString insertarRec(NodoString nodo, String x) {
        if (nodo == null) {
            return new NodoString(x);
        }
        if (x.compareTo(nodo.elem) < 0) {
            nodo.izq = insertarRec(nodo.izq, x);
        } else if (x.compareTo(nodo.elem) > 0) {
            nodo.der = insertarRec(nodo.der, x);
        } else {
            nodo.frec++;
        }
        return nodo;
    }

    public void mostrarAscendente() {
        mostrarAscendenteRec(raiz);
    }

    private void mostrarAscendenteRec(NodoString nodo) {
        if (nodo != null) {
            mostrarAscendenteRec(nodo.izq);
            System.out.println("Elemento: " + nodo.elem + ", Frecuencia: "
                    + nodo.frec);
            mostrarAscendenteRec(nodo.der);
        }
    }

    public void mostrarDescendente() {
        mostrarDescendenteRec(raiz);
    }

    private void mostrarDescendenteRec(NodoString nodo) {
        if (nodo != null) {
            mostrarDescendenteRec(nodo.der);
            System.out.println("Elemento: " + nodo.elem + ", Frecuencia: "
                    + nodo.frec);
            mostrarDescendenteRec(nodo.izq);
        }
    }
}

public class NodoString {
```

```
    public NodoString izq;
    public String elem;
    public NodoString der;
    public int frec;

    public NodoString(String x) {
        elem = x;
        izq = der = null;
        frec = 1;
    }
}
```