UNIVERSIDAD AUTÓNOMA “GABRIEL RENÉ MORENO”

FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

ESTRUCTURA DE DATOS II – SX(INF-310)

Laboratorio #4

PORCENTAJE TERMINADO: 100%.
GRUPO: 14, 20, 22

Integrantes	DT	HG	HI	EVAL
SANDOVAL PEREZ MADDY	1	1	1	100
CRISTIAN GEDALGE	1	1	1	100
RUBEN CANO	1	1	1	100

Fecha de Presentación: Jueves, 16 de Mayo del 2024
Fecha Presentada: Jueves, 16 de Mayo del 2024
Días de Retraso: 0

SANTA CRUZ – BOLIVIA

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, con los elementos del árbol anterior, organizados por frecuencia de ocurrencia y elemento.

- ***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.

```
package lab4;
import java.util.Random;

public class Arbol {
   public NodoArbol raiz;

   public Arbol() {
      this.raiz = null;
   }

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

   private NodoArbol insertar(NodoArbol p, int x) {
      if (p == null) return new NodoArbol(x);
      else 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 frecuencias() {
      frecuencias(raiz);
   }
```

```
private void frecuencias(NodoArbol p) {
   if (p == null) return;
   System.out.println(p.elem + "  " + p.frec);
   frecuencias(p.izq);
   frecuencias(p.der);
}

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

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

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

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

public void insertarElementosAleatorios(int n, int a, int b) {
   Random rand = new Random();
   for (int i = 0; i < n; i++) {
      int num = rand.nextInt((b - a) + 1) + a;
      insertar(num);
   }
}

public Arbol crearABBPorFrecuencia() {
   ListaArray lista = new ListaArray();
   llenarLista(raiz, lista);

   Arbol nuevoArbol = new Arbol();
   for (int i = 0; i < lista.cantElem; i++) {
      for (int j = 0; j < lista.frec[i]; j++) {
         nuevoArbol.insertar(lista.elem[i]);
      }
   }
   return nuevoArbol;
}
```

```
    private void llenarLista(NodoArbol p, ListaArray lista) {
        if (p == null) return;
        llenarLista(p.izq, lista);
        lista.insertarOrdenado(p.elem, p.frec);
        llenarLista(p.der, lista);
    }
}
```

```
package lab4;


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

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

```
package lab4;


public class ListaArray {
    public int max;
    public int cantElem;
    public int[] elem;
    public int[] frec;

    public ListaArray(int max) {
        this.max = max;
        this.cantElem = 0;
        this.elem = new int[max];
        this.frec = new int[max];
    }

    public ListaArray() {
        this(100);
    }
```

```
    public boolean seEncuentra(int x) {
        for (int i = 0; i < cantElem; i++) {
            if (elem[i] == x) return true;
        }
        return false;
    }

    public int indexOf(int x) {
        for (int i = 0; i < cantElem; i++) {
            if (elem[i] == x) return i;
        }
        return -1;
    }

    public void insertarSinOrden(int x) {
        if (!seEncuentra(x)) {
            elem[cantElem] = x;
            frec[cantElem] = 1;
            cantElem++;
        } else {
            frec[indexOf(x)]++;
        }
    }

    public void insertarOrdenado(int x, int f) {
        int i = 0;
        while (i < cantElem && (frec[i] < f || (frec[i] == f && elem[i] < x))) i++;
        insertarIesimo(x, f, i);
    }

    private void insertarIesimo(int x, int f, int i) {
        for (int j = cantElem; j > i; j--) {
            elem[j] = elem[j - 1];
            frec[j] = frec[j - 1];
        }
        elem[i] = x;
        frec[i] = f;
        cantElem++;
    }

    public void frecuencias() {
        for (int i = 0; i < cantElem; i++) {
            System.out.println(elem[i] + "  " + frec[i]);
        }
    }
}
```

```
package lab4;


public class Lab4 {
   public static void main(String[] args) {
      Arbol arbol = new Arbol();
      arbol.insertarElementosAleatorios(100, 1, 50); // Insertar 100 elementos aleatorios entre
1 y 50

      System.out.println("Elementos de menor a mayor:");
      arbol.mostrarMenorAMayor();

      System.out.println("\nElementos de mayor a menor:");
      arbol.mostrarMayorAMenor();

      Arbol arbolPorFrecuencia = arbol.crearABBPorFrecuencia();

      System.out.println("\nElementos en nuevo ABB (por frecuencia, menor a mayor):");
      arbolPorFrecuencia.mostrarMenorAMayor();

      System.out.println("\nElementos en nuevo ABB (por frecuencia, mayor a menor):");
      arbolPorFrecuencia.mostrarMayorAMenor();
   }
}
```