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-1. ARBOLES BINARIOS DE BÚSQUEDA.

PORCENTAJE TERMINADO : 100%.

GRUPO: 14

Nombre	Registro
Cristian Gabriel Gedalge Cayhuara	219022062

Fecha de Presentación: Sábado ,04 de mayo de 2024

COMENTARIO:
Los ejercicios propuestos me parecieron interesantes resolverlos, el que me complico un poco fue el ejercicio 15 el inmediato Superior no sabía como pasarlo el mismo elemento si no se encontraba ,así que hice uso de la función se encuentra, con eso pude resolverlo .

CLASE ARBOL

```
public class Arbol {

    public Nodo raiz;

    public Arbol() {
        this.raiz = null;
    }
//1.  A1.generarElem(n, a, b) : Método que genera n elementos
aleatorios enteros diferentes entre a y b inclusive.

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

//2.  A1.insertar(x) : Método que inserta el elemento x, en
el árbol A1 en su lugar correspondiente.
    public void insertar(int x) {
        this.raiz = insertar(x, raiz);
    }

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

//3.  A1.preOrden() : Método que muestra los elementos del
árbol A1 en preOrden.
    public void preOrden() {
        preOrden(raiz);
    }

    public void preOrden(Nodo p) {
        if (p == null) {
            return;
        }
        System.out.println(p.elem);
        preOrden(p.izq);
```

```
        preOrden(p.der);
    }

//4.  A1.inOrden() : Método que muestra los elementos del
árbol A1 en inOrden.
    public void inOrden() {
        inOrden(raiz);
    }

    private void inOrden(Nodo p) {
        if (p == null) {
            return;
        }
        inOrden(p.izq);
        System.out.println(p.elem);
        inOrden(p.der);
    }

//5.  A1.postOrden() : Método que muestra los elementos del
árbol A1 en postOrden.
    public void postOrden() {
        System.out.print("[");
        postOrden(raiz);
        System.out.print("]");
    }

    public void postOrden(Nodo p) {
        if (p == null) {
            return;
        }
        postOrden(p.izq);
        postOrden(p.der);
        System.out.print(p.elem + ",");
    }

//6.  A1.desc(): Método que muestra los elementos del árbol
A1 de mayor a menor.
    public void desc() {
        desc(this.raiz);
    }

    private void desc(Nodo p) {
        if (p == null) {
            return;
        }
        desc(p.der);
        System.out.print(p.elem + " ,");
        desc(p.izq);
```

```
}
//7.  A1.seEncuentra(x) : Métodos lógico que devuelve True,
si el elemento x, se encuentra en el árbol A1.

    public boolean seEncuentra(int x) {
        return seEncuentra(x, raiz);
    }

    private boolean seEncuentra(int x, Nodo p) {
        if (p == null) {
            return false;
        }
        if (x == p.elem) {
            return true;
        }
        if (x < p.elem) {
            return seEncuentra(x, p.izq);
        } else {
            return seEncuentra(x, p.der);
        }

    }

//8.  A1.cantidad() : Método que devuelve la cantidad de
nodos del árbol A1.
    public int cantidad() {
        return cantidad(raiz);
    }

    private int cantidad(Nodo p) {
        int cant;
        if (p == null) {
            return 0;
        } else {
            cant = 1 + cantidad(p.izq) + cantidad(p.der);
        }
        return cant;
    }

//9.  A1.suma() : Método que devuelve la suma de los
elementos del árbol A1.
    public int suma() {
        return suma(raiz);
    }

    private int suma(Nodo p) {
        int sum;
        if (p == null) {
```

```
            return 0;
        } else {
            sum = p.elem + (suma(p.izq) + suma(p.der));
        }
        return sum;
    }

//10.  A1.menor() : Método que devuelve el elemento menor del
árbol A1.
    public int menor() {
        return menor(raiz);
    }

    private int menor(Nodo p) {
        if (p.izq == null) {
            return p.elem;
        } else {
            return menor(p.izq);
        }
    }

//11.  A1.mayor() : Método que devuelve el elemento mayor del
árbol A1.
    public int mayor() {
        return mayor(raiz);
    }

    private int mayor(Nodo p) {
        if (p.der == null) {
            return p.elem;
        } else {
            return menor(p.der);
        }
    }

//12.  A1.mostrarTerm(): Método que muestra los elementos de
los nodos terminales del árbol A1. Mostrar los elementos de
menor a mayor.
    public void mostrarTerm() {
        mostrarTerm(this.raiz);
    }

    private void mostrarTerm(Nodo p) {
        if (p == null) {
            return;
        } else if (p.izq == null && p.der == null) {
            System.out.println(p.elem);
        } else {
```

```
            mostrarTerm(p.izq);
            mostrarTerm(p.der);
        }
    }
//13.  A1.cantidadTerm(): Método que devuelve la cantidad de
nodos terminales del árbol A1.

    public int cantidadTerm() {
        return cantidadTerm(this.raiz);
    }

    private int cantidadTerm(Nodo p) {
        int cantTerm;
        if (p == null) {
            return 0;
        } else if (p.izq == null && p.der == null) {
            return 1;
        } else {
            cantTerm = cantidadTerm(p.izq) +
cantidadTerm(p.der);
        }
        return cantTerm;
    }

//14.  A1.lineal() : Método lógico que devuelve True, si el
árbol A1 es un árbol degenerado o lIneal. (Se puede dar
cuando se genera el árbol con una secuencia ordenada de
elementos)
    public boolean lineal() {
        return lineal(this.raiz);
    }

    private boolean lineal(Nodo p) {
        if (p == null) {
            return false;
        } else {
            return (lineal(p.der) && lineal(p.izq));
        }
    }
//15. A1.inmediatoSup(x) : Método que devuelve el elemento
inmediato superior a x, si x se encuentra en A1, caso
contrario devuelve el mismo elemento.

    public int inmediatoSup(int x) {
        if(seEncuentra(x))
            return inmediatoSup(x, this.raiz);
        else
            return x;
```

```
    }

    private int inmediatoSup(int x, Nodo p) {
        int res;
        if (p == null) {
            return x;
        } else if (x == p.elem) {
            return p.elem;
        } else {
            if (x < p.elem) {
                res = inmediatoSup(x, p.izq);
            } else {
                res = inmediatoSup(x, p.der);
            }

            if (res == x) {
                res = p.elem;
            }

        }
        return res;
    }
//16. A1.inmediatoInf(x) : Método que devuelve el elemento
inmediato inferior a x, si x se encuentra en A1, caso
contrario devuelve el mismo elemento.
     public int inmediatoInf(int x) {
        if(seEncuentra(x))
            return inmediatoInf(x, this.raiz);
        else
            return x;
    }

    private int inmediatoInf(int x, Nodo p) {
        int res;
        if(x==p.elem){
            if(p.izq==null&&p.der==null)
                return x;
            if(p.izq!=null)
                return p.izq.elem;
            else
                return  p.der.elem;
        }else
        {
            if(x<p.elem)
                res=inmediatoInf(x,p.izq);
            else
                res=inmediatoInf(x,p.der);
        }
```

```
        return res;
    }
//17.  Implementar al menos 5 Ejercicios adicionales
cualesquiera, de consultas sobre uno o más árboles binarios
de búsqueda. Citar fuentes.

//1. Implementar una funcion para determinar la altura de un
arbol
    public int  altura()
    {
        return altura(this.raiz);
    }
    private int altura(Nodo p)
    {
        int x;
        if(p==null)return -1;

        else{
            if(altura(p.izq)>altura(p.der))
            {
                x=1+altura(p.izq);
            }else{
                x=1+altura(p.der);
            }
        }
        return x;
    }

//2. implementar una funcion que sume los elementos pares de
un arbol
    public int sumaPares()
    {
        return  sumaPares(this.raiz);
    }

    private int sumaPares(Nodo p)
    {
        int sum;
        if(p==null)return 0;
        else{
            sum=sumaPares(p.izq)+sumaPares(p.der);
            if(p.elem%2==0)
                sum=sum+p.elem;
        }
        return sum;
    }
```

```
// 3. Devolver true si exiten mas elemenos pares que impares
en el arbol
    public boolean masPares()
    {
        return (cantPares()>cantImpares());
    }
    private int  cantPares()
    {
        return cantPares(this.raiz);
    }
    public int  cantPares(Nodo p)
    {
        int cantPares;
        if(p==null)return 0;
        else{
            cantPares=cantPares(p.izq)+cantPares(p.der);
            if(p.elem%2==0)
            {
                cantPares=cantPares+1;
            }
        }
        return cantPares;

    }
// 4. mostrar la cantidad de elementos impares del arbol
     public int  cantImpares()
     {
         return  cantImpares(this.raiz);
     }

    private int  cantImpares(Nodo p)
    {
        int cantImpares;
        if(p==null)return 0;
        else{

cantImpares=cantImpares(p.izq)+cantImpares(p.der);
            if(p.elem%2!=0)
            {
                cantImpares=cantImpares+1;
            }
        }
        return cantImpares;

    }
```

```
// 5. Insertar el nodo Izq(RAMA IZQ) de A1 en  A2
    public void insertarenA2(Arbol A1)
    {
        insertarenA2(A1.raiz.izq);
    }
    private void insertarenA2(Nodo p)
    {
        if(p!=null)
        {
            insertar(p.elem);
            insertarenA2(p.izq);
            insertarenA2(p.der);
        }
    }
    public static void main(String[] args) {
        Arbol A1 = new Arbol();
        A1.insertar(45);
        A1.insertar(23);
        A1.insertar(2);
        A1.insertar(38);
        A1.insertar(7);
        A1.insertar(65);
        A1.insertar(52);
        A1.insertar(96);
        A1.insertar(8);
        Arbol A2=new Arbol();

        A2.insertarenA2(A1);
        A2.inOrden();
        //A1.mostrarTerm();
    }
}
```

BIBLIOGRAFIA

Ejercicios sobre Árboles Generales (ugr.es)
Ejercicios Arboles 1920 Soluciones(1).pdf (cartagena99.com)