Algoritmos
y
Estructuras de Datos

Curso 2014

Ejercicio 1-Anexo 2

Había una vez un chimpancé llamado *Luchu Bandor,* cuyo significado era “Mono Playboy”. *Luchu* estaba infelizmente casado con *Bunty Mona,* una chimpancé muy bonita pero de baja estatura. *Luchu* era alto y guapo, se sentía incómodo cuando estaba con *Bunty* en lugares públicos, ya que la gente los miraba a ellos continuamente. En un momento dado, *Luchu* no pudo soportar más esta situación y decidió hacer justicia a su nombre. Él comenzó a buscar una nueva esposa en el “Colegio Nacional de Señoritas Chimpancés”. Cada día *Luchu* se subía a unas cañas de bambú y esperaba a que el ejercicio matutino empezara. Desde allí podía ver a todas las chimpancés haciendo su rutina de ejercicio diario. Ahora, *Luchu* estaba buscando a una chimpancé más alta pero que sea más baja que él, y también estaba interesado en aquella chimpancé un poco más alta que él. Sin embargo, alguien de su misma altura no la consideraba.

Luchu pudo modelar la situación descripta a través de una subclase de árbol AVL, la cuál contenía todas las alturas de las señoritas chimpancés que él había observado durante un cierto período de tiempo.

Su trabajo consiste en ayudar a *Luchu* para encontrar a las dos mejores chimpancés de acuerdo al criterio de selección establecido: la chimpancé más alta de las más bajas que él y la más baja entre las más altas que él.

Usted debe implementar un método en la subclase **ArbolAVLAlturasLuchus** de **ArbolAVL<Integer>**, considerando que recibe como parámetro la altura de *Luchu* y debe devolver las alturas de las dos chimpancés buscadas ordenados de manera creciente. En el caso que sea imposible encontrar alguna de estas dos alturas devuelva un valor igual a 0 para la menor y 999 para la mayor. (0 y 999 no son alturas válidas, no están en el árbol)

Importante: considere que en el árbol existe una altura igual a la altura de *Luchu*.

Ejercicio 1-Anexo. Escenarios

Escenarios a considerar:

i) La altura de Luchus sea la raíz y las alturas de las monas descendientes.

ii) La altura de Luchus no sea la raíz, pero las alturas de las monas sigan siendo descendientes.

iii) Luchus no tenga alguna mona (menor o mayor).

iv) Alguna o ambas de las alturas de las monas sean ancestros.

Ejercicio 1-Anexo. Solución:

```
public class ArbolAVLAlturasLuchus extends ArbolAVL<Integer> {

public Alturas buscarAlturas (Integer alt){
    Alturas alturas=new Alturas(0,999); //La clase Altura contendra el resultado con las alturas buscadas
    if (!this.esVacio()) {
        if (alt.compareTo(this.getDatoRaiz()) == 0)  {  // encontré la altura de Luchu !
                    Integer altMenor=0,  altMayor=999;
                    if( ! this.getHijoIzquierdo().esVacio()){
                              altMenor  = ((ArbolAVLAlturasLuchus) this.getHijoIzquierdo()).buscarMax();// busco la altura mayor del sub. izquierdo
                     }
                     if( ! this.getHijoDerecho().esVacio()){
                             altMayor =  ((ArbolAVLAlturasLuchus) this.getHijoDerecho()).buscarMin();  // busco la altura menor del sub. Derecho
                     }
                     alturas = new Alturas(altMenor,altMayor);
           }else {
                     if (alt.compareTo(this.getDatoRaiz())  > 0) {
                          alturas = ((ArbolAVLAlturasLuchus) this.getHijoDerecho()).buscarAlturas (alt); // busco en la rama derecha
                          if( alturas.getAlturaMenor() == 0) alturas.setAlturaMenor( this.getDatoRaiz() ); //Corrige por si no encontró
                     }else {
                           alturas = ((ArbolAVLAlturasLuchus) this.getHijoIzquierdo()).buscarAlturas (alt);         // busco en la rama izquierda
                           if( alturas.getAlturaMayor() == 999) alturas.setAlturaMayor( this.getDatoRaiz() ); //Corrige por si no encontró
                     }
           }
        }
    }
    return alturas;
}
```

Ejercicio 1-Anexo. Solución(cont.):

```
private Integer buscarMax() {
    Integer alt;
    if (this.getHijoDerecho().esVacio()) {
        alt = this.getDatoRaiz();
    } else {
        alt = ((ArbolAVLAlturasLuchus) this.getHijoDerecho()).buscarMax();
    }
    return alt;
}

private Integer buscarMin() {
    Integer alt;
    if (this.getHijoIzquierdo().esVacio()) {
        alt = this.getDatoRaiz();
    } else {
        alt =((ArbolAVLAlturasLuchus) this.getHijoIzquierdo()).buscarMin();
    }
    return alt;
}

}
```

Ejercicio 1-Anexo. Solución(cont.):

```
public class Alturas {
private Integer alturaMayor, alturaMenor;
public Alturas(Integer alturaMayor, Integer alturaMenor) {
    this.alturaMayor = alturaMayor;
    this.alturaMenor = alturaMenor;
}

public Integer getAlturaMayor() {
    return alturaMayor;
}

public void setAlturaMayor(Integer alturaMayor) {
    this.alturaMayor = alturaMayor;
}

public Integer getAlturaMenor() {
    return alturaMenor;
}

public void setAlturaMenor(Integer alturaMenor) {
    this.alturaMenor = alturaMenor;
}
}
```