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 un árbol AVL, el 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 clase árbol AVL, 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. Solución:

```
private void  find (Integer altLuchu,) {
if (!this.esVacio()) {
    if (altLuchu.compareTo(this.getDatoRaiz()) == 0)  {  // encontré la altura de Luchu !
                if( ! this.getHijoIzquierdo().esVacio())
                        altMenor  = this.getHijoIzquierdo().findMax();    // busco la altura mayor del sub. izquierdo
                if( ! this.getHijoDerecho().esVacio())
                        altMayor = this.getHijoDerecho().findMin();  // busco la altura menor del sub. derecho
          }else {
       if (altLuchu.compareTo(this.getDatoRaiz())  > 0) {
                this.getHijoDerecho().find ();                    // busco en la rama derecha
                    if (altMenor. == 0)                 // verifica si se enc. la mas chica (en el ej casos : 107,  118, 150)
                        altMenor  = this.getDatoRaiz();         //sino encontro, la mas chica inmediata es la 1er encontrada al
       volver del llamado recursivo
            }else {
                    this.getHijoIzquierdo().find ();                // busco en la rama izquierda
                    if (altMayor == 999)                            // verifica si se enc. la mas gde (en el ej casos : 95,107,118
                        altMayor  = this.getDatoRaiz();   //sino encontro, la mas gde. inmediata es la 1er encontrada al volver
       del llamado recursivo
             }
        }
  }
}
```

Ejercicio 1-Anexo. Solución(cont.):

```
//busca la altura mayor del sub. izquierdo
private Integer findMax() {
 Integer alt;
     if( ! this.getHijoDerecho().esVacio())
         alt = this.getHijoDerecho().findMax();
    else {
       alt = this.getDatoRaiz();
   }
    return alt;
}
// busca la altura menor del sub. Derecho
private Integer findMin() {
 Integer alt;
    if( ! this.getHijoIzquierdo().esVacio())
         alt = this.getHijozquierdo( ). findMin();
    else {
       alt = this.getDatoRaiz();
   }
    return alt;
}
```

Ejercicio 1-Anexo. Solución(cont.):

```
private Integer altMayor  = 999; //variable de instancia
private Integer altMenor = 0;  //variable de instancia
ListaGenerica< Integer > buscar (Integer alt){
    ListaGenerica< Integer> l = new     ListaEnlazadaGenerica< Integer >();
    find(alt );
    l.agregar( altMenor, 1);    // agrega a la lista l los valores hallados, ordenados de manera  creciente
    l.agregar( altMayor, 2);

    return l;
}
```