Árboles AVL: Funciones AVL InsAVL(Elem e, AVL a) y AVL HazAVL(DicBin a)

En la última clase presencial, terminamos de revisar los casos en los que debemos hacer cada una de las 4 rotaciones para re-balancear un árbol que era AVL, y dejó de serlo después de una inserción.

Lurgo, en la clase en vídeo, reduje las rotaciones a dos: izquierda y derecha, y expliqué los tres casos que se pueden presentar al insertar un elemento que causa que el árbol se desbalancee.

También, en el último vídeo, me referí todo el tiempo a la función "InsAVL", que se encargaba de insertar y, si fuera necesario, re-balancear el árbol. Para efectos de implementación, la función "InsAVL" invocará, cuando el árbol se desbalancee, a la función "HazAVL", que se encargará de re-balancear al árbol.

Ahora, debemos definir las funciones de inserción (AVL InsOrd(Elem e, AVL a) y de re-balanceo (AVL HazAVL(DicBin a) para crear y mantener un árbol AVL.

I.- La función de inserción para árboles AVL, debe, después de insertar un elemento 'e' en un árbol AVL 'a', mediante la función InsOrd(Elem e, AVL a), verificar si el árbol de búsqueda binaria, o diccionario binario, continúa siendo o no un árbol AVL. Si el árbol 'a' sigue siendo AVL, no es necesario hacer rotación alguna; en caso contrario, debemos de re-balancear el árbol para devolverle su calidad de árbol AVL. Esto último, lo haremos invocando a la función "AVL HazAVL(DicBin a)", la que describo en pseudo-código, después de presentar el pseudo-código de la función AVL InsAVL(Elem a, AVL a).

1.- AVL InsAVL(Elem e, AVL a)= a=InsOrd(e,a);
Si EsAVL(a)
return a;
Sino
return HazAVL(a);

II. Como la función "AVL HazAVL(Elem e, AVL a)" contempla todas las posibilidades por la cuales un árbol AVL deja de serlo, mismas que expliqué en el vídeo, después de haberle insertado un elemento, la describo con más detalle en pseudo-código, de la siguiente manera:

2.- El árbol dejo de ser AVL, esto es: EsAVL(a))==Falso, y debe ser balanceado nuevamente. En este caso, tenemos las siguientes posibilidades:

2.1.- Si (Absoluto(FactBal(a))>1
2.1.1 Si (EsAVL(izquierdo(a)) && EsAVL(derecho(a))) == Cierto
Entonces, ambos sub-árboles son AVL, y tenemos algunos de los dos siguientes casos:

2.1.1.1- Si FactBal(a)>1: se desbalanceo por la izquierda, y existen dos posibilidades:
2.1.1.1.1 Si FacBal(izquierod(a))>0
return RotaDer(a).
2.1.1.1.2 Sino, entonces se cumple que FacBal(izquierod(a))<0
return RotaDerIzq(a).
2.1.1.2.- Sino, entonces se cumple que FactBal(a)<-1: se desbalanceos por la derecha, y existen dos posibilidades:
2.1.1.2.1 Si FacBal(derecho(a))<0
return RotaIzq(a).
2.1.1.2.2 Sino, entonce se cumple que FacBal(derechod(a))>0
return a=RotaIzqDer(a).
2.1.2 Sino, entonces alguno de los subarboles dejó de ser AVL, y tenemos dos posibilidades:
2.1.2.1 Si EsAVL(izquierdo(a))==Cierto
return cons(raiz(a), izquierdo(a), HazAVL(derecho(a)))
2.1.2.2 Sino, entonces se cumple que EsAVL(derecho(a))==Cierto
return cons(raiz(a),HazAVL(izquierdo(a), derecho(a))

2.2.- Sino, evidentemente se cumple que Absoluto(FactBal(a))<=1, implicando que alguno de los sub-árboles dejo de ser AVL.
En este caso, tenemos alguno de los dos casos:
2.2.1 Si EsAVL(izquierdo(a))=Cierto
return cons(raiz(a), izquierdo(a), HazAVL(derecho(a))
2.2.2 Sino, entonces se cumple que EsAVL(derecho(a))==Cierto
return cons(raiz(a), HazAVL(izquierdo(a), derecho(a)).

La funciones de doble rotación, funcionan de la siguiente forma:

AVL RotaIzqDer(DicBin a){ return RotaIzq(consAB(raiz(a),izquierdo(a),RotaDer(derecho(a))));
primero rota hacia la derecha el sub-árbol derecho, y luego rota hacia la izquierda todo el árbol, es decir desde la raíz.

AVL RotaDerIzq(DicBin a){ return RotaDer(consAB(raiz(a),RotaIzq(izquierdo(a)),derecho(a)));
primero rota hacia la izquierda el sub-árbol izquierdo, y luego rota hacia la derecha todo el árbol, es decir desde la raíz.