ESTRUCTURA DE DATOS

Ing. Cívil Informática

Prof. Nibaldo Rodriguez A.

ARBOLES DE BÚSQUEDA BALANCEADOS

Prof. Nibaldo Rodriguez A.

DEFINICIÓN

- Se dice que un árbol binario está Balanceado SSI
  - –En cada nodo las alturas de sus dos subárboles defieren como máximo en **Una Unidad**.

AVL

- Este tipo de árbol son conocidos con el nombre de **ÁRBOLES AVL**
- En honor a sus inventores
  - –Adelson-Velskii y Landis, 1962

FACTOR DE BALANCE

- **FB=Hd-Hi**
  - Altura subárbol derecho-altura subárbol izquierdo)
- **FB=0**, significa que las alturas de los subárboles son iguales
- **FB>0**, significa que el subárbol derecho es más grande que el izquierdo
- **FB<0**, significa que el subárbol izquierdo es más grande que le derecho.

FACTOR DE BALANCE

- **FB=Hi-Hd**
  - altura subárbol izquierdo-altura subárbol derecho
- **FB=0**, significa que las alturas de los subárboles son iguales
- **FB>0**, significa que el subárbol izquierdo es más grande que el derecho
- **FB<0**, significa que el subárbol izquierdo es más pequeño que le derecho.

ESTRUCTURA de Datos

```
typedef struct nodoAVL {
   int  data,Altura ;
   struct nodoAVL *izq,*der;
}AVL;
```

Operaciones Básicas:

- **Insertar**
- **Buscar**
- **Eliminar**
- **etc**

INSERTAR NODO en AVL

- Es idéntico al de un ABB
- El nuevo dato se inserta como nodo Hoja
- Se actualiza su altura (Altura=0)
- Después de la Inserción se deberá verificar si el árbol mantiene la condición de árbol AVL.
  - **Si la mantiene**, entonces actualizar sólo FB o Alturas
  - **Sino** Realizar proceso de Rebalanceo
    - **Rotaciones Simples (RS) Izq o Derecha**
    - **Rotaciones Dobles (RDI o RDD).**

NODO PIVOTE

- En qué caso se Hará o no Balanceo en el AVL?
  - **Identificar Nodo PIVOT**
- **Nodo Pivote**:
  - **Es el Nodo Ancestral más cercano al Nodo Nuevo con FB distinto de Cero.**

ROTACIONES

- **Suponer que el Nodo cuyo Equilibrio debemos ajustar es K**
- **Si diferencia de las profundidades de los subárboles de K es (+-2).**
- **Entonces:**
  - **Ocurre 4 casos:**

ROTACIONES

- **1.- Un insert en el SubArbol Izq. del hijo Izq. de K**
- **2.- Un insert en el SubArbol Derecho del hijo Izq. de K**
- **3.- Un insert en el SubArbol Izq. del hijo Derecho de K**
- **4.- Un insert en el SubArbol Derecho del hijo Derecho de K**

ROTACIONES

- Los casos 1 y 4 son conocidos como:
  - Rotación Simples con Hijo Izquierdo de **K** (RSI)
  - Rotación Simple con Hijo Derecho de **K** (RSD)
- Los casos 2 y 3 son conocidos como Rotaciones Dobles (RD):
  - RD Izquierdo-Derecho
  - RD Derecho-Izquierdo

ROTACIONES

- **CASO 2:Rotación dobles**
- **Se realiza usando 2 Rotaciones Simples:**
  - **RS con el Hijo de K y su Nieto**
  - **RS entre K y su nuevo Hijo**
- **CASO 3:Rotación Doble**
  - **Simétrico.**

REBALANCE

Si No Existe Nodo Pivot:

Modificar FB desde **NodoNuevo** Hasta la **Raíz**

Si Existe Nodo Pivot && NodoNuevo insertado en Subárbol Menor:

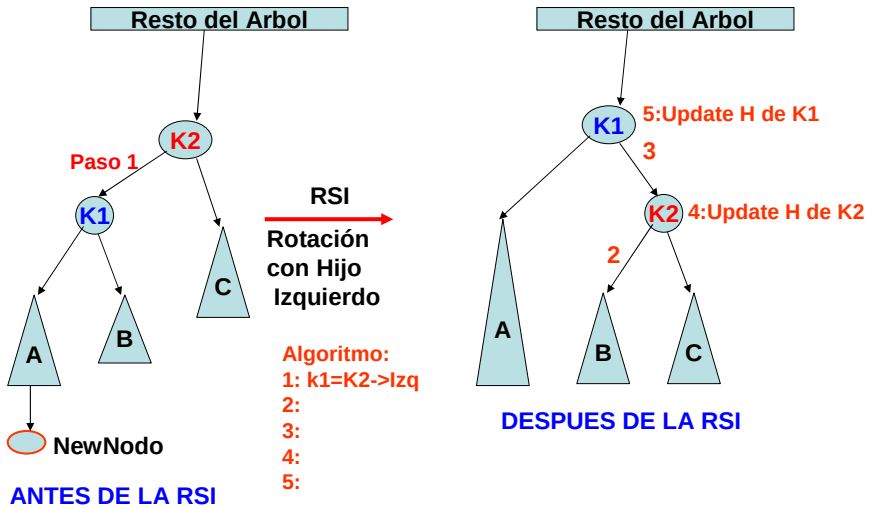
Modificar FB desde **NodoNuevo** Hasta el **Pivot**

SI Existe Nodo Pivot && NodoNuevo Insertado en Subárbol Mayor:

Realizar una de las 4 Rotaciones

R O T A C I O N E S
AVL

ROTACIÓN SIMPLE IZQUIERA



ROTACIÓN SIMPLE IZQUIERDA (RSI)

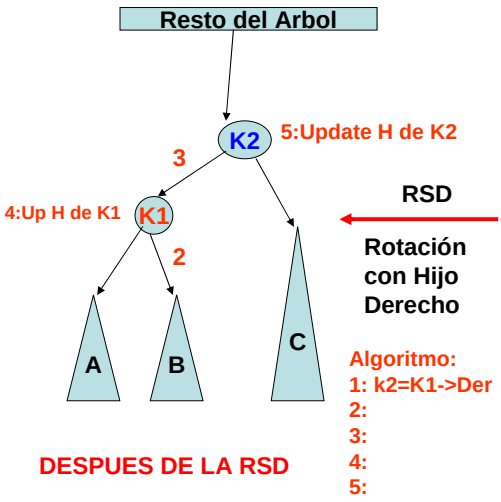
K2: Nodo Pivot **K1: Hijo izquierdo de K2**

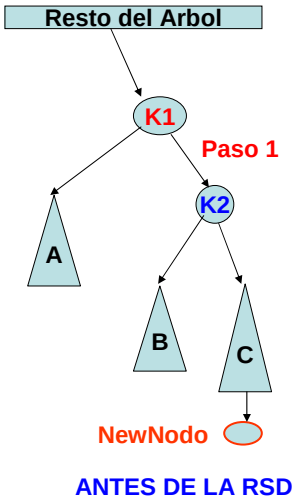
```
Avl * RSI( Avl *K2 ) {
Avl *K1;
K1 = K2->Left;
K2->Left = K1->Right;
K1->Right = K2;
K2->Altura=Max( Altura( K2->Left ), Altura( K2->Right ) ) + 1;
K1->Altura = Max( Altura( K1->Left ), K2->Altura ) + 1;
return K1;  /* Nuevo  Nodo Raíz del Subárbol */
}
```

```
int Altura(Avl *P ){
return(P==NULL ? -1 : P->Altura);
}
```

```
int Max( int L, int R ){
   return (L > R ? L: R);
}
```

ROTACIÓN SIMPLE DERECHA





ROTACIÓN SIMPLE DERECHA (RSD)

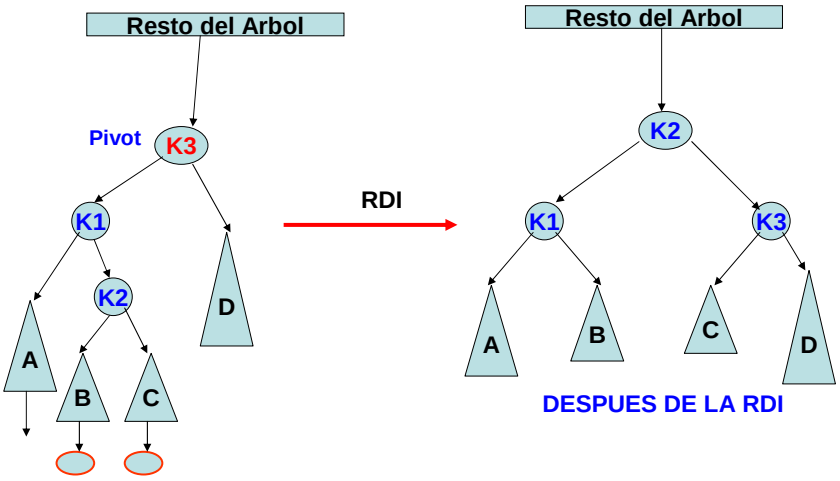
K1: Nodo Pivot K2: Hijo derecho de K1

```
Avl *RSD(Avl *K1 ) {
Avl *K2;
K2 = K1->Right;
K1->Right = K2->Left;
K2->Left = K1;
K1->Altura = Max( Altura( K1->Left ), Altura( K1->Right ) )
+ 1;
K2->Altura = Max( Altura( K2->Right ), K1->Altura ) + 1;
return K2;  /* Nuevo Nodo Raíz del subárbol */
}
```

```
int Altura(Avl *P ){
return(P==NULL ? -1 : P->Altura);
}
```

```
int Max( int L, int R ){
    return (L > R ? L: R);
}
```

ROTACIÓN DOBLE HIJO IZQUIERA



ANTES DE LA RDI

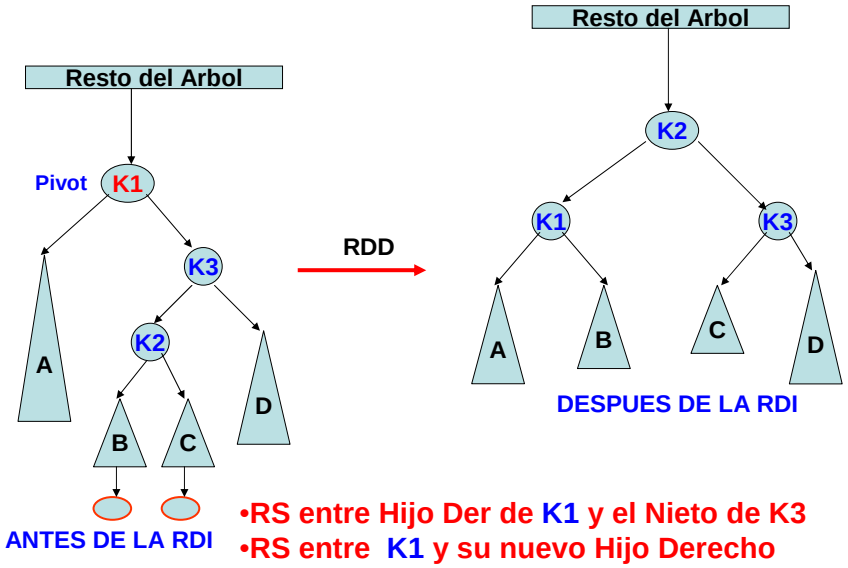
- RS entre Hijo Izq de K3 y el Nieto de K3
- RS entre K3 y su nuevo Hijo Izquierdo

ROTACIÓN DOBLE CON HIJO IZQUIERDO

K3: Nodo Padre
K1: Hijo Izquierdo de K3
K2: Hijo Derecho de K1

```
Avl * RD_Izquierda(Avl *K3 ){
// RS entre Hijo Izq de K3 y Nieto de K3
K3->Left = RSD( K3->Left );
// RS de K3 con su nuevo Hijo Izquierdo
return RSI( K3 );
}
```

ROTACIÓN DOBLE DERECHA



- RS entre Hijo Der de K1 y el Nieto de K3
- RS entre K1 y su nuevo Hijo Derecho

ROTACIÓN DOBLE CON HIJO DERECHO

K1: Nodo Pivot
K3: Hijo derecho de K1
K2: Hijo Izquierdo de K3

```
Avl * RD_H_Derecha(Avl *K1 ) {
// RS entre Hijo Der de K1 y Nieto de K1
K1->Right =RSI( K1->Right );
// RS entre K1 y su nuevo Hijo derecho
return RSD( K1 );
}
```

Altura de un Árbol Binario

```
int Altura(Avl *P ){
return(P==NULL ? -1 : P->Altura);
}
```

```
int Max( int L, int R ){
    return (L > R ? L: R);
}
```

WEB

http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm

ESTRUCTURA DE DATOS

Ing. Cívil Informática