

Universidad de las Fuerzas Armados "ESPE"

Maryely Simbaña

10/12/2025

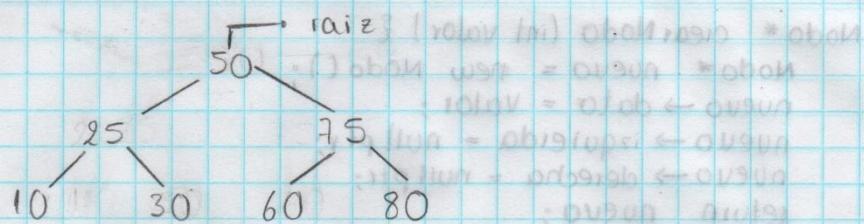
NRC.

Matemática 1

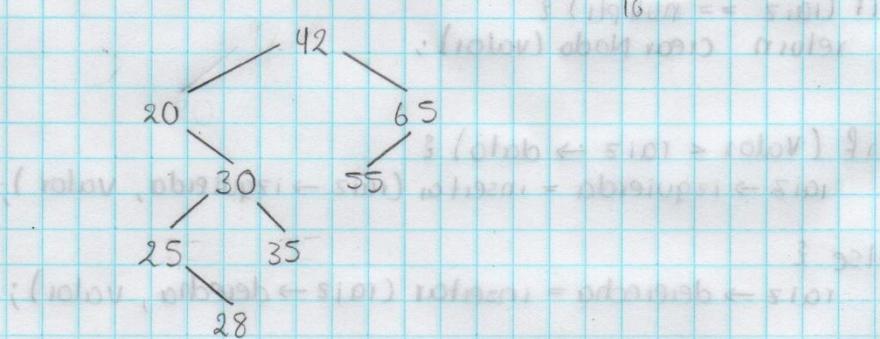
Algoritmos y Estructuras de Datos

Árbol Binario

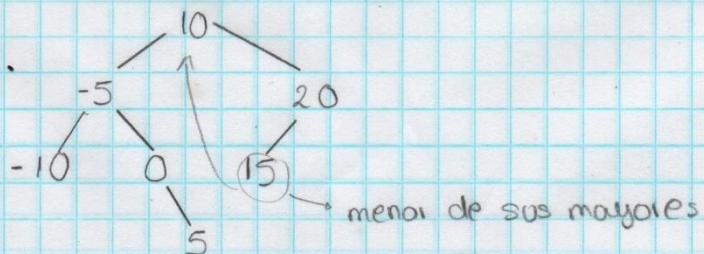
- ① Lista: 50, 25, 75, 10, 30, 60, 80



- ② Lista: 42, 20, 65, 30, 55, 25, 35, 28

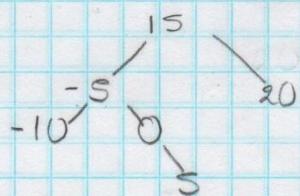


- ③ Lista: 10, -5, 20, 0, -10, 15, 5

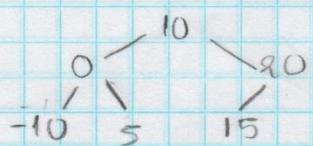


Si se elimina la raíz se busca al número más parecido, al menor de sus mayores

si se elimina 10 =>



Si se elimina -5



Código: "3923" elaborado por mi en F del año 2016

```
#include <iostream>
using namespace std;
```

```
struct Nodo {
```

```
    int dato;
    Nodo* izquierda;
    Nodo* derecha;
```

```
};
```

```
Nodo* crearNodo (int valor) {
```

```
    Nodo* nuevo = new Nodo();
    nuevo->dato = valor;
    nuevo->izquierda = nullptr;
    nuevo->derecha = nullptr;
    return nuevo;
```

```
}
```

```
Nodo* insertar (Nodo* raiz, int valor) {
```

```
    if (raiz == nullptr) {
        return crearNodo (valor);
```

```
}
```

```
    if (valor < raiz->dato) {
```

```
        raiz->izquierda = insertar (raiz->izquierda, valor);
```

```
}
```

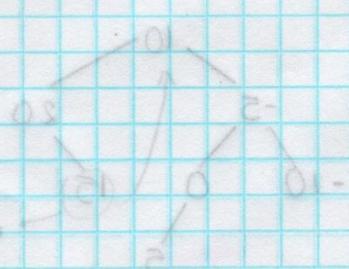
```
    else {
```

```
        raiz->derecha = insertar (raiz->derecha, valor);
```

```
}
```

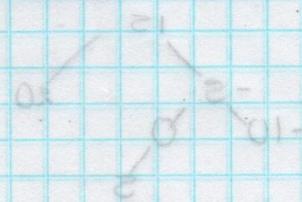
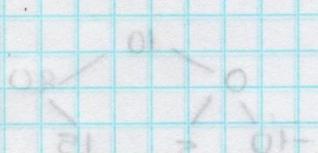
```
    return raiz;
```

```
}
```



Al momento de dibujar el diagrama me di cuenta que el orden de los nodos no es correcto.

El orden correcto es:



= O) orden de los

25 10 21 15 18 09 05 08 02 01 12

ARBOL AVL

AVL \rightarrow Por (Adelson-Velsky y Landis)

Dector de Inclinación (Factor de Equilibrio)

Izquierda - Derecha

Puede ser : -1, 0, 1

0: Equilibrado

1: Pesa más la izquierda

-1: Pesa más la derecha

Si el resultado es 2 o -2 ocurre la Rotación.

CASO A: Línea recta de 3 números
se toma el número del medio y lo sube

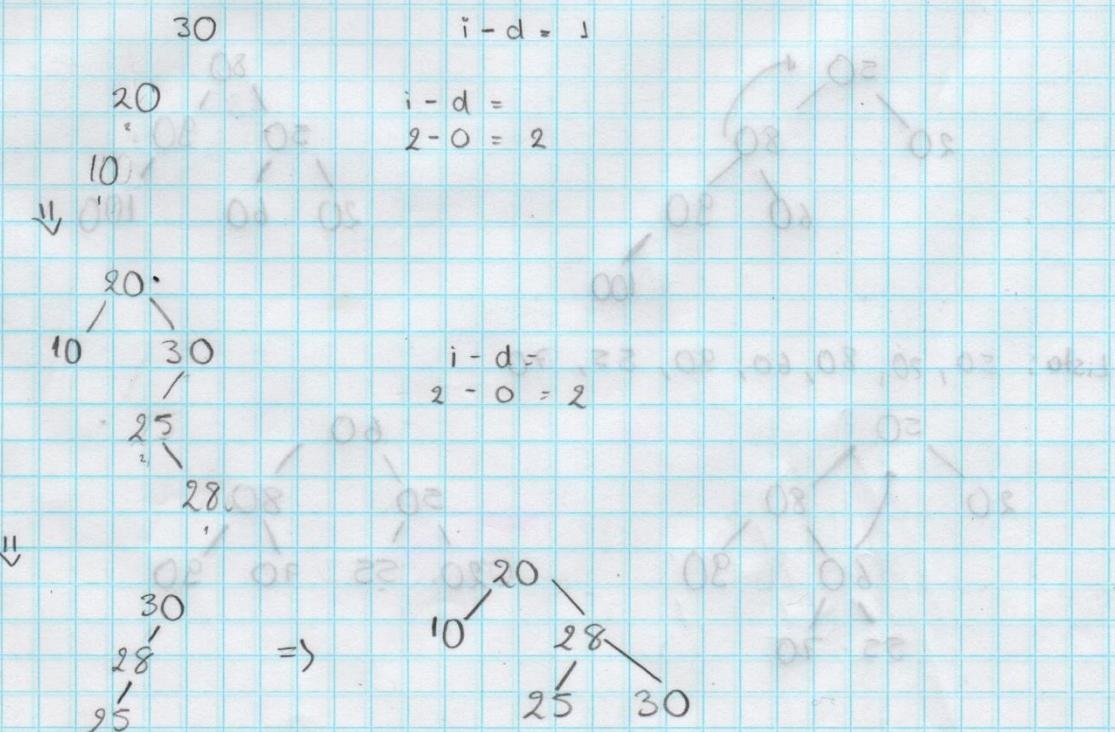
CASO B: zig-zag

1: Primero se convierte en una línea recta

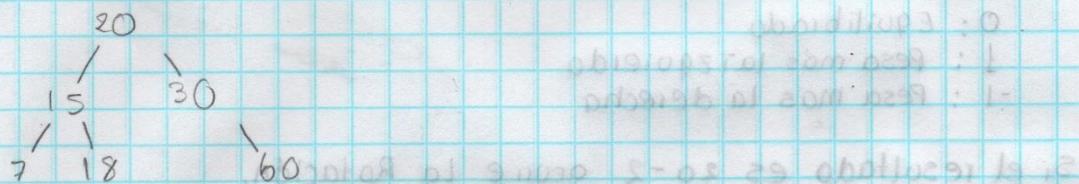
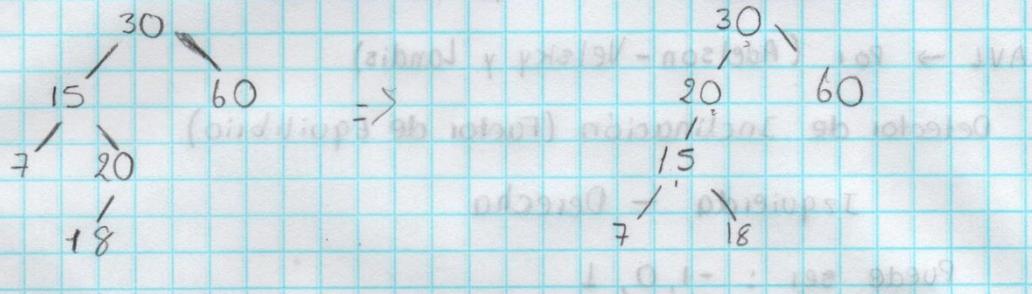
2: luego se hace lo del caso A (subir el de en medio)

Ejemplo

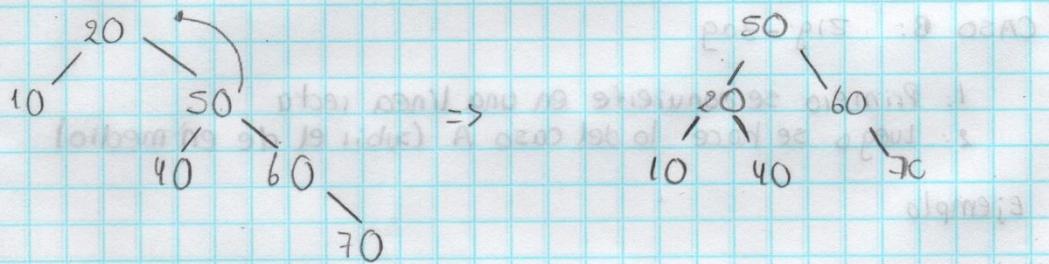
Lista: 30, 20, 10, 25, 28



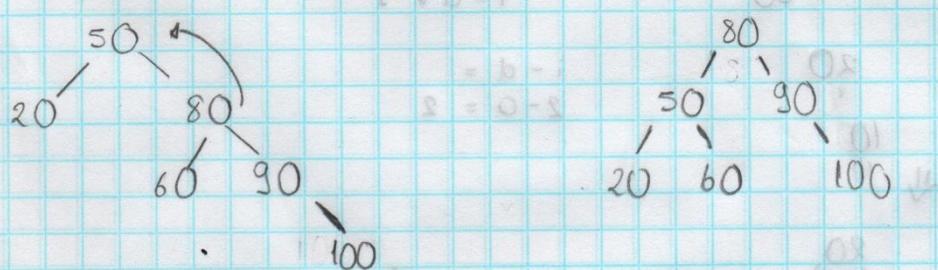
Lista: 30, 15, 60, 7, 20, 18



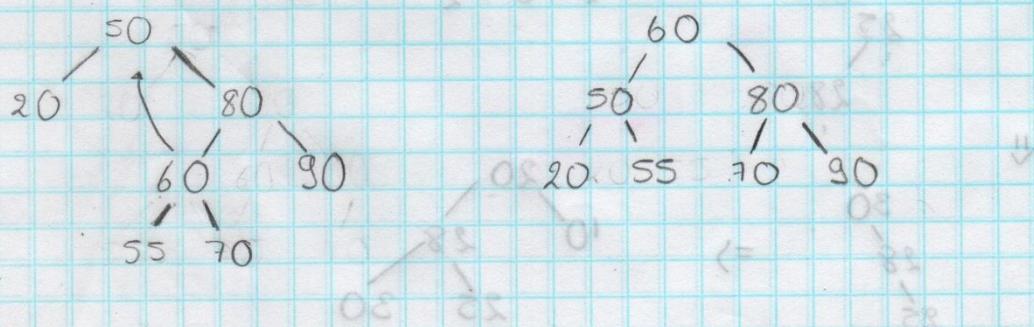
Lista: 20, 10, 50, 40, 60, 70



Lista: 50, 20, 80, 60, 90, 100



Lista: 50, 20, 80, 60, 90, 55, 70



Código:

3 (oblab tñi abon > obab) rotarizri *obab
; (obab) obbab rotarizri (lun = obab) tñi

struct Nodo {

int dato; (obab < abon > obab) ;

Nodo* izquierda; (obab < abon > obab) ;

Nodo* derecha; (obab < abon > obab) ;

int altura; (obab < abon > obab) ;

};

int obtenerAltura (Nodo* n) {

if (n == nullpti) return 0;

; (obab abon > abon) n->altura; (obab abon > abon) n->altura + 1 - obab abon

};

Nodo* crearNodo (int dato) {

Nodo* nodo = new Nodo (); (obab < abon > obab) ;

nodo->dato = dato; (obab) obbab rotarizri ;

nodo->izquierda = null;

nodo->derecha = null; (obab abon > abon > obab) ;

nodo->altura = 1; (obab abon > abon > obab) ;

return nodo; (obab abon > abon > obab) ;

};

int obtenerBalance (Nodo* n) { (obab > abon > obab) ;

if (n == null) return 0; (obab abon > abon) ;

return obtenerAltura (n->izquierda) - obtenerAltura (n->derecha); (obab abon > abon) ;

};

Nodo* rotarDerecha (Nodo* y) {

Nodo* x = y->izquierda

Nodo* T2 = x->derecha

x->derecha = y;

y->izquierda = T2;

y->altura = max (obtenerAltura (y->izquierda), obtenerAltura (y->derecha)) + 1;

x->altura = max (obtenerAltura (x->izquierda), obtenerAltura (x->derecha)) + 1;

return x;

};

Nodo* rotarIzquierda (Nodo* x) {

Nodo* y = x->derecha;

Nodo* T2 = y->izquierda;

y->izquierda = x;

x->derecha = T2;

x->altura = max (obtenerAltura (x->izquierda), obtenerAltura (x->derecha)) + 1;

y->altura = max (obtenerAltura (y->izquierda), obtenerAltura (y->derecha)) + 1;

return y;

};

```

Nodo* insertar (Nodo* nodo, int dato) {
    if (nodo == null) return crearNodo (dato);
    if (dato < nodo->dato)
        nodo->izquierda = insertar (nodo->izquierda, dato); z apunta
    else if (dato > nodo->dato)
        nodo->derecha = insertar (nodo->derecha, dato); obligarse a rotar
    else
        return nodo;
    nodo->altura = 1 + max (obtenerAltura(nodo->izquierdo), obtenerAltura (nodo->derecha));
    int balance = obtenerBalance (nodo);
    if (balance < -1 & dato > nodo->derecha->dato)
        return rotarIzquierda (nodo); rotar izquierdo
    if (balance > 1 & dato > nodo->izquierda->dato)
        nodo->izquierda = rotarIzquierda (nodo->izquierda); rotar izquierdo
    return rotarDerecha (nodo); rotar derecho
}
3
if (balance < -1 & dato < nodo->derecha->dato) { 3
    nodo->derecha = rotarDerecha (nodo->derecha); dato = n - 1
    return rotarIzquierda (nodo); dato = n + 1
}
3
return nodo;
3

```

$\{ \text{if} (\text{dato} < \text{nodo} \leftarrow \text{R}) \text{ o } \text{nullAlt}(\text{nodo}) \text{ o } \text{obligarse} \leftarrow \text{R} \} \text{ o } \text{if} (\text{dato} > \text{nodo}) \text{ x } \text{dato} = \text{dato} \leftarrow \text{R}$

$\{ \text{if} (\text{dato} > \text{nodo} \leftarrow \text{x}) \text{ o } \text{nullAlt}(\text{nodo}) \text{ o } \text{obligarse} \leftarrow \text{x} \} \text{ o } \text{if} (\text{dato} > \text{nodo}) \text{ x } \text{dato} = \text{dato} \leftarrow \text{x}$

$\{ \text{x } \text{dato} \}$

$\{ \text{x } \text{dato} \} \text{ o } \text{obligarse}(\text{nodo}) \text{ o } \text{dato}$

$\{ \text{obligarse} \leftarrow \text{x} \} \text{ o } \text{dato}$

$\{ \text{x } \text{dato} \} \text{ o } \text{obligarse}(\text{nodo}) \text{ o } \text{dato}$

$\{ \text{obligarse} \leftarrow \text{x} \} \text{ o } \text{dato}$

$\{ \text{x } \text{dato} \} \text{ o } \text{obligarse}(\text{nodo}) \text{ o } \text{dato}$

$\{ \text{obligarse} \leftarrow \text{x} \} \text{ o } \text{obligarse}(\text{nodo}) \text{ o } \text{dato}$

$\{ \text{x } \text{dato} \}$