

# PRIMER PARCIAL

## INF310 SX- Estructuras de Datos II. Gestión 1-2018.

### Subgrupo: P-Z

#### Árbol Binario

**1.** La class Árbol presentada aquí, permite crear un árbol desordenado, pero evitando la duplicación. Su constructor crea el primer nodo del árbol (ver código fuente). Escriba usted el algoritmo de inserción de este árbol:


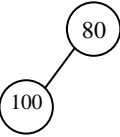
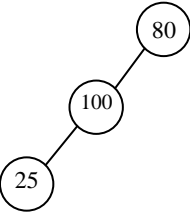
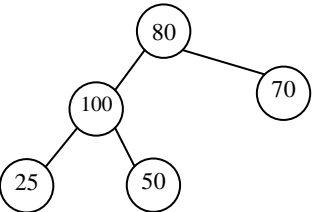
**public void insertar(int padre, int h)**

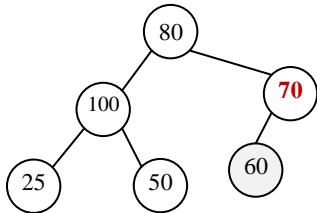
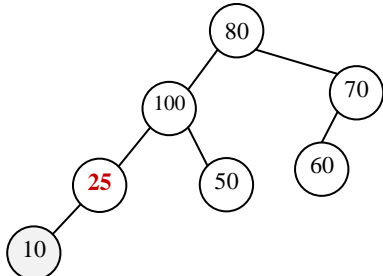
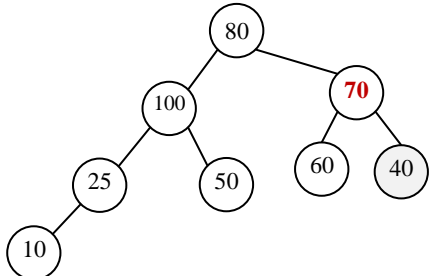
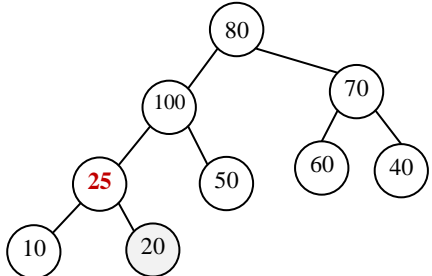
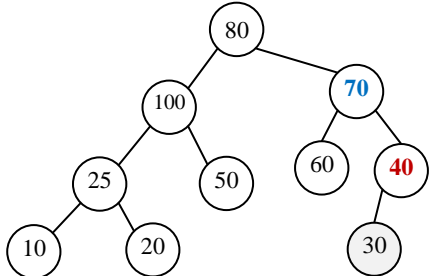
El cual básicamente inserta a **h** como hijo de **padre**. Si **padre** no existe o el nodo **h** ya existe en el Árbol, este algoritmo no realiza ninguna acción ("no pasa nada").

Si **padre** existe y **h** no existe en el Árbol, siga los siguientes pasos:

- a) Si **padre** no tiene hijos, **h** se insertará como su hijo izquierdo. Fin de la inserción.
- b) Si **padre** tiene un hijo, **h** se insertará como el hijo faltante. Fin de la inserción.
- c) Si **padre** tiene 2 hijos, entonces **h** toma como padre al hijo menor, es decir **padre=hijo\_menor\_de\_padre**. Continuar en (a) .

Por ejemplo:

<b>Arbol A = new Arbol(80) ;</b>	
<b>A.insertar(80, 100) ;</b> //100 se inserta a la izq del 80, porque el 80 no tiene hijos.	
<b>A.insertar(100, 25) ;</b> //25 se inserta a la izq del 100, porque el 100 no tiene hijos.	
<b>A.insertar(80, 70) ;</b> //70 se inserta a la der del 80, porque el 80 ya tiene hijo izq.  <b>A.insertar(100, 50) ;</b> //50 se inserta a la der del 100, porque el 100 ya tiene hijo izq.	
<b>A.insertar(200, 60) ;</b> //El 200, el <b>padre</b> , <b>no existe</b> . El árbol permanece igual.	
<b>A.insertar(50, 70) ;</b> //El <u>70</u> , el <b>hijo</b> , <b>ya existe</b> . El árbol permanece igual, porque no se //permiten duplicados.	

<p><b>A.insertar(80, 60);</b> /* Como el 80 tiene dos hijos, escogemos al menor de ellos (o sea al <b>70</b>) como padre del 60. El 70 no tiene hijos, por tanto el 60 se inserta a la izquierda del 70 */</p>	
<p><b>A.insertar(100, 10);</b> /* Como el 100 tiene dos hijos, escogemos al menor de ellos (o sea al <b>25</b>) como padre del 10. El 25 no tiene hijos, por tanto el 10 se inserta a la izquierda del 25. */</p>	
<p><b>A.insertar(80, 40);</b> /* Como el 80 tiene dos hijos, escogemos al menor de ellos (o sea al <b>70</b>) como padre del 40. El 70 ya tiene hijo izquierdo, por tanto el 40 se inserta a la derecha del 70. */</p>	
<p><b>A.insertar(100, 20);</b> /* Como el 100 tiene dos hijos, escogemos al menor de ellos: al <b>25</b>, como padre del <b>20</b>. El 25 ya tiene hijo izquierdo, por tanto el 20 se inserta a la derecha del 25. */</p>	
<p><b>A.insertar(80, 30);</b> /* Como el 80 tiene dos hijos, escogemos al menor de ellos: al <b>70</b>, como padre del <b>30</b>. Pero el <b>70</b> ya tiene dos hijos, por tanto, escogemos al menor de los hijos del 70 como padre del 30: El <b>40</b> será el padre del <b>30</b>. Como el <b>40</b> no tiene hijos, el <b>30</b> se inserta a la izquierda del 40. */</p>	

## Listas

**2.** La clase Lista de enteros, presentada aquí, se mantiene **ordenada** pero permite la duplicación de valores. En esta class, escriba el procedimiento

```
public void add(int x)
```

el cual inserte x a la Lista, manteniéndola **ordenada**. Sin embargo, tome en cuenta que si x ya aparece dos veces en la Lista, x es eliminado de la misma.

### Por ejemplo:

```
Lista p = new Lista();          //p=[ ].
p.add(9);                       //p=[9].
p.add(5);                       //p=[5, 9]. (Note que la Lista se mantiene ordenada)
p.add(1);                       //p=[1, 5, 9].
p.add(5);                       //p=[1, 5,5, 9].
p.add(9);                       //p=[1, 5,5, 9, 9].
p.add(5);                       //Como el 5 ya está dos veces en la Lista p, el 5 es eliminado de la Lista. p=[1, 9, 9].
p.add(12);                      // p=[1, 9, 9, 12].
p.add(1);                       // p=[1, 1, 9, 9, 12].
p.add(1);                       // Como el 1 ya está dos veces en la Lista p, el 1 es eliminado de la Lista. p=[9, 9, 12].
```