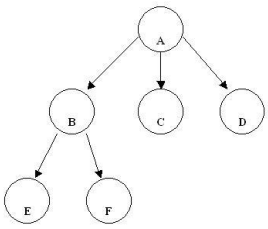
ARBOLES

ESTRUCTURAS DINÁMICAS: CONCEPTOS DE ÁRBOLES

Igual que la lista, el árbol es una estructura de datos. Son muy eficientes para la búsqueda de información. Los árboles soportan estructuras no lineales.



Algunos conceptos de la estructura de datos tipo árbol:

Nodo hoja: Es un nodo sin descendientes (Nodo terminal)

Ej. Nodos E – F – C y D.

Nodo interior: Es un nodo que no es hoja.

Ej. Nodos A y B.

Nivel de un árbol: El nodo A está en el nivel 1 sus descendientes directos están en el nivel 2 y así sucesivamente. El nivel del árbol está dado por el nodo de máximo nivel.

Ej. Este árbol es de nivel 3.

Grado de un nodo: es el número de nodos hijos que tiene dicho nodo (solo se tiene en cuenta los nodos interiores)

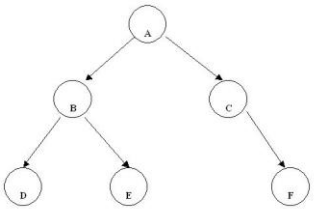
Ej. El nodo A tiene grado 3.
El nodo B tiene grado 2.

Los otros nodos no tienen grado porque no tienen descendientes.

Grado de un árbol: Es el máximo de los grados de todos los nodos de un árbol.

Ej. El grado del árbol es 3.

Longitud de camino del nodo x: Al número de arcos que deben ser recorridos para llegar a un nodo x, partiendo de la raíz.
La raíz tiene longitud de camino 1, sus descendientes directos tienen longitud de camino 2, etc. En forma general un nodo en el nivel i tiene longitud de camino i.
Árbol binario: Un árbol es binario si cada nodo tiene como máximo 2 descendientes.



Para cada nodo está definido el subárbol izquierdo y el derecho.

Para el nodo A el subárbol izquierdo está constituido por los nodos B, D y E. Y el subárbol derecho está formado por los nodos C y F.

Lo mismo para el nodo B tiene el subárbol izquierdo con un nodo (D) y un nodo en el subárbol derecho (E).

El nodo D tiene ambos subárboles vacíos.

El nodo C tiene el subárbol izquierdo vacío y el subárbol derecho con un nodo (F).

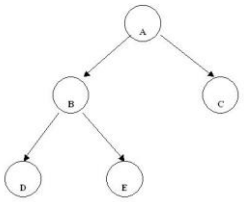
Árbol binario perfectamente equilibrado: Si para cada nodo el número de nodos en el subárbol izquierdo y el número de nodos en el subárbol derecho difiere como mucho en una unidad.

Hay que tener en cuenta todos los nodos del árbol.

El árbol de más arriba es perfectamente equilibrado.

Ej. árbol que no es perfectamente equilibrado:

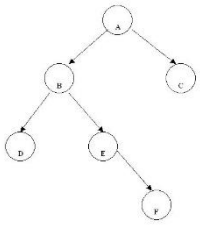
El nodo A tiene 3 nodos en el subárbol izquierdo y solo uno en el subárbol derecho, por lo que no es perfectamente equilibrado.



Árbol binario completo: Es un árbol binario con hojas como máximo en los niveles n-1 y n (Siendo n el nivel del árbol)

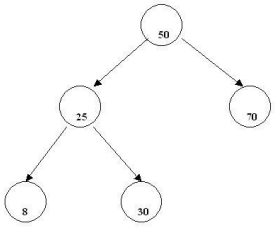
Los dos árboles graficados son completos porque son árboles de nivel 3 y hay nodos hoja en el nivel 3 en el primer caso, y hay nodos hoja en los niveles 3 y 2 en el segundo caso.

Ej. Árbol binario no completo:



Hay nodos hoja en los niveles 4, 3 y 2. No debería haber nodos hojas en el nivel 2.

Árbol binario ordenado: Si para cada nodo del árbol, los nodos ubicados a la izquierda son inferiores al que consideramos raíz para ese momento y los nodos ubicados a la derecha son mayores que la raíz.



Ej. Analicemos si se trata de un árbol binario ordenado:

Para el nodo que tiene el 50:

Los nodos del subárbol izquierdo son todos menores a 50? 8, 25, 30 Si

Los nodos del subárbol derecho son todos mayores a 50? 70 Si.

Para el nodo que tiene el 25:

Los nodos del subárbol izquierdo son todos menores a 25? 8 Si

Los nodos del subárbol derecho son todos mayores a 25? 30 Si.

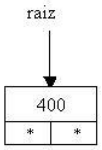
No hace falta analizar los nodos hoja. Si todas las respuestas son afirmativas podemos luego decir que se trata de un árbol binario ordenado.

INSERCIÓN DE NODOS Y RECORRIDO DE UN ÁRBOL BINARIO

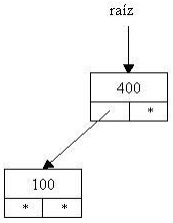
Para administrar un árbol binario ordenado debemos tener especial cuidado en la inserción.

Inicialmente el árbol está vacío, es decir raíz apunta a null:

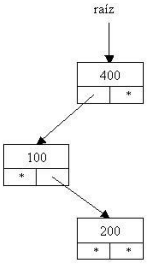
Insertamos el 400



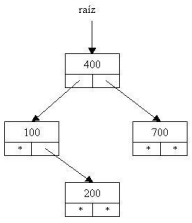
Insertamos el valor 100. Debemos analizar si raíz es distinto a null verificamos si 100 es mayor o menor a la información del nodo apuntado por raíz, en este caso es menor y como el subárbol izquierdo es null debemos insertarlo allí.



Insertamos el 200. Hay que tener en cuenta que siempre comenzamos las comparaciones a partir de raíz. El 200 es menor que 400, descendemos por el subárbol izquierdo. Luego analizamos y vemos que el 200 es mayor a 100, debemos avanzar por derecha. Como el subárbol derecho es null lo insertamos en dicha posición.



Insertamos el 700 y el árbol será:



Como podemos observar si cada vez que insertamos un nodo respetamos este algoritmo siempre estaremos en presencia de un árbol binario ordenado. Posteriormente veremos el algoritmo en java para la inserción de información en el árbol.

Búsqueda de información en un árbol binario ordenado.

Este es uno de los principales usos de los árboles binarios.
Para realizar una búsqueda debemos ir comparando la información a buscar y descender por el subárbol izquierdo o derecho según corresponda.

Ej. Si en el árbol anterior necesitamos verificar si está almacenado el 700, primero verificamos si la información del nodo apuntado por raíz es 700, en caso negativo verificamos si la información a buscar (700) es mayor a la información de dicho nodo (400) en caso afirmativo descendemos por el subárbol derecho en caso contrario descendemos por el subárbol izquierdo.

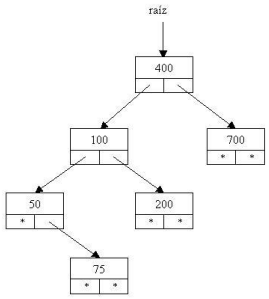
Este proceso lo repetimos hasta encontrar la información buscada o encontrar un subárbol vacío.

Recorridos de árboles binarios.

Recorrer: Pasar a través del árbol enumerando cada uno de sus nodos una vez.
Visitar: Realizar algún procesamiento del nodo.
Los árboles pueden ser recorridos en varios órdenes:

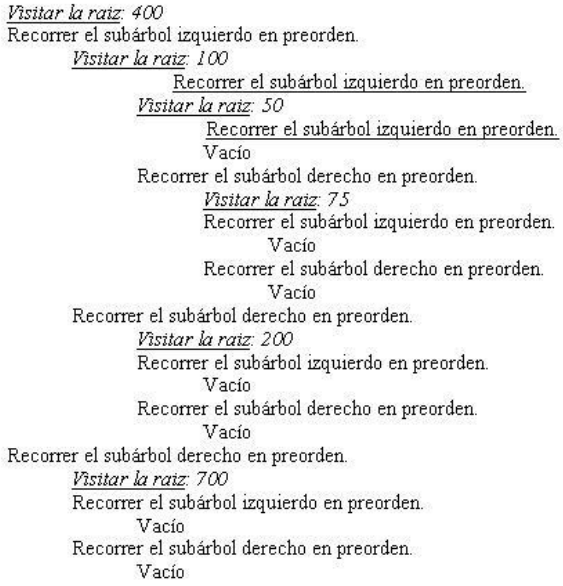
```
Pre-orden:
           - Visitar la raíz.
           - Recorrer el subárbol izquierdo en pre-orden.
           - Recorrer el subárbol derecho en pre-orden.

Entre-orden
          - Recorrer el subárbol izquierdo en entre-orden.
          - Visitar la raíz.
          - Recorrer el subárbol derecho en entre-orden.

Post-orden
          - Recorrer el subárbol izquierdo en post-orden.
          - Recorrer el subárbol derecho en post-orden.
          - Visitar la raíz.
```

Ejemplo:



Veamos cómo se imprimen las informaciones de los nodos según su recorrido:

Recorrido preorden:



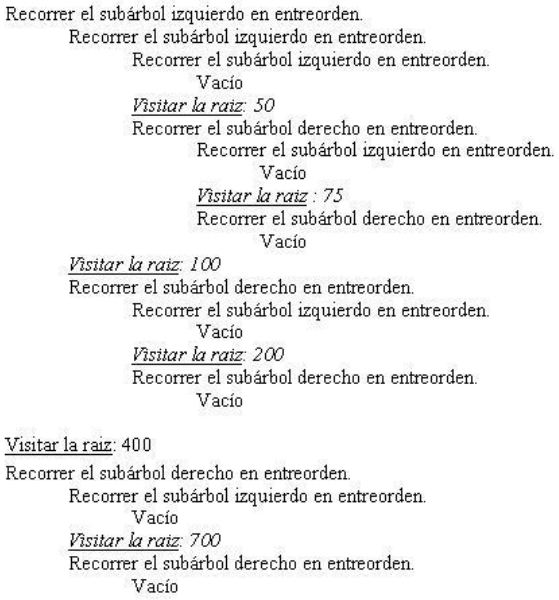
Es decir que el orden de impresión de la información es:

`400 – 100 –50 – 75 –200 – 700`

Es importante analizar que el recorrido de árboles es recursivo. Recorrer un subárbol es semejante a recorrer un árbol.

Es buena práctica dibujar el árbol en un papel y hacer el seguimiento del recorrido y las visitas a cada nodo.

Recorrido entreorden:



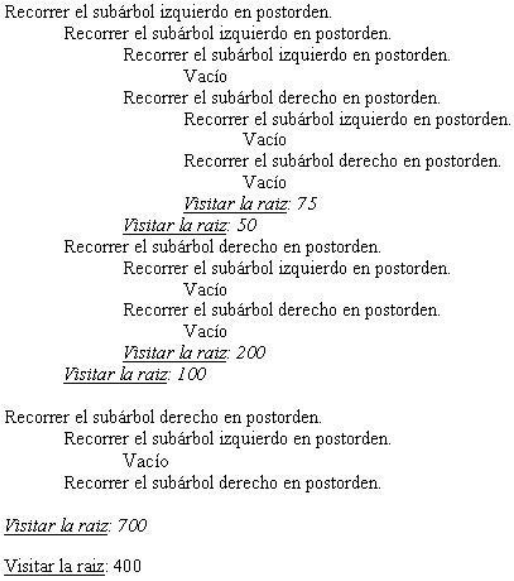
Es decir que el orden de impresión de la información es:

`50 –75 – 100 –200 – 400 – 700`

Si observamos podemos ver que la información aparece ordenada.

Este tipo de recorrido es muy útil cuando queremos procesar la información del árbol en orden.

Recorrido postorden:



Es decir que el orden de impresión de la información es:

```
75 – 50 – 200 – 100 – 700 – 400
```

INSERCIÓN DE NODOS Y RECORRIDO DE UN ÁRBOL BINARIO

Problema 1: (Ejercicio 12)

A continuación desarrollamos una clase para la administración de un árbol binario ordenado.

Problema 2: (Ejercicio 13)

Confeccionar una clase que permita insertar un entero en un árbol binario ordenado verificando que no se encuentre previamente dicho número.
Desarrollar los siguientes métodos:

1 - Retornar la cantidad de nodos del árbol.
2 - Retornar la cantidad de nodos hoja del árbol.
3 - Imprimir en entre orden.
4 - Imprimir en entre orden junto al nivel donde se encuentra dicho nodo.
5 - Retornar la altura del árbol.
6 - Imprimir el mayor valor del árbol.
7 - Borrar el nodo menor del árbol.

IMPLEMENTACIÓN EN C# DE UN ÁRBOL BINARIO ORDENADO

Problema 1: (Ejercicio 14)

A continuación desarrollamos una clase para la administración de un árbol binario ordenado.

Problema 2: (Ejercicio 15)

Confeccionar una clase que permita insertar un entero en un árbol binario ordenado verificando que no se encuentre previamente dicho número.

Desarrollar los siguientes métodos:

1 - Retornar la cantidad de nodos del árbol.
2 - Retornar la cantidad de nodos hoja del árbol.
3 - Imprimir en entre orden.
4 - Imprimir en entre orden junto al nivel donde se encuentra dicho nodo.
5 - Retornar la altura del árbol.
6 - Imprimir el mayor valor del árbol.
7 - Borrar el nodo menor del árbol.