	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	90/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

# Guía Práctica de Estudio 8


## Árboles parte 1

Elaborado por:

M.I. Elba Karen Sáenz García

Revisión:

Ing. Laura Sandoval Montaña

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	91/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Guía Práctica 8

### Estructura de datos y Algoritmos II

### Árboles. Parte 1.

**Objetivo:** El estudiante conocerá e identificará las características de la estructura no lineal árbol.

#### Actividades

Implementar una estructura de datos- árbol binario- así como su recorrido en algún lenguaje de programación.

#### Antecedentes

- Análisis previo del concepto de árbol y su representación visto en clase teórica.
- Manejo de listas, diccionarios, estructuras de control, funciones y clases en Python 3.
- Conocimientos básicos de la programación orientada a objetos.


#### Introducción

En las estructuras de datos lineales como las pilas o las colas, los datos se estructuran en forma secuencial es decir cada elemento puede ir enlazado al siguiente o al anterior. En las estructuras de datos no lineales o estructuras multi-enlazadas se pueden presentar relaciones más complejas entre los elementos; cada elemento puede ir enlazado a cualquier otro, es decir. puede tener varios sucesores y/o varios predecesores. Ejemplos de estructuras de datos no lineales son los grafos y árboles.

Un árbol es una colección de elementos llamados nodos, uno de los cuales se distingue como raíz, junto con una relación(rama) que impone una estructura jerárquica entre los nodos. Los árboles genealógicos y los organigramas son ejemplos de árboles.

Un árbol puede definirse formalmente de forma recursiva como sigue:

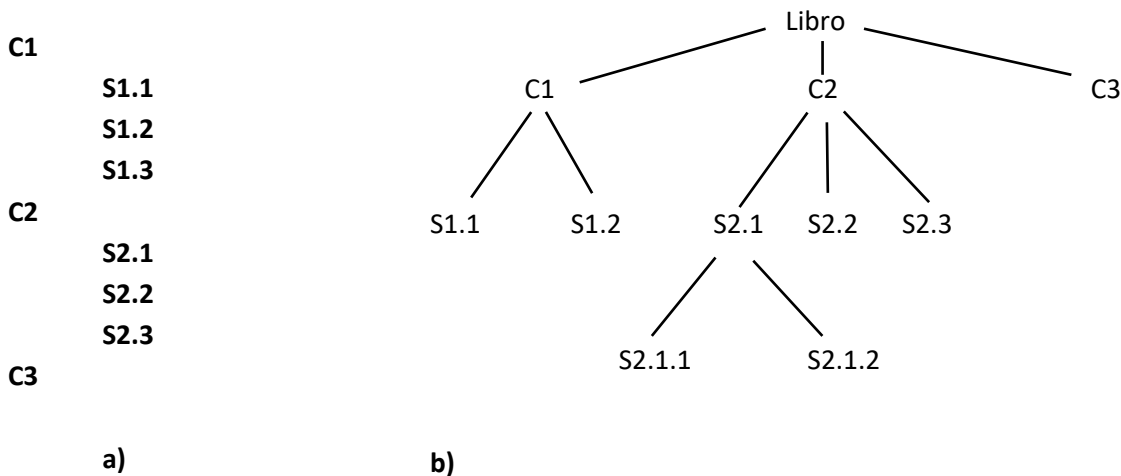
- 1- Un solo nodo es, por sí mismo un árbol. Ese nodo es también la raíz de dicho árbol.
- 2- Suponer que  $n$  es un nodo y que  $A_1, A_2, \dots, A_k$  son árboles con raíces  $n_1, n_2, \dots, n_k$ , respectivamente. Se puede construir un nuevo árbol haciendo que  $n$  se constituya en el padre de los nodos  $n_1, n_2, \dots, n_k$ . En dicho árbol,  $n$  es la raíz y  $A_1, A_2, \dots, A_k$  son subárboles de la raíz. Los nodos  $n_1, n_2, \dots, n_k$ , reciben el nombre de hijos del nodo  $n$ .

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	92/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

La definición implica que cada nodo del árbol es raíz de algún subárbol contenido en el árbol principal.

Un árbol vacío o nulo es aquel que no tiene ningún nodo.

**Ejemplo:** El índice general de un libro Figura 8.1 a), es un árbol y se puede dibujar como en 8.1 b) donde la relación padre hijo se representa con una línea que los une. Los árboles normalmente se dibujan de arriba hacia abajo y de izquierda a derecha como en el ejemplo.



**Figura 8.1**


Se puede observar en la figura 8.2 b) que la raíz del árbol es libro y este tiene 3 subárboles C1, C2 y C3 que a su vez son hijos del nodo o padre libro.

## Términos asociados con el concepto de árbol.

**Grado de un nodo:** Es el número de subárboles o hijos que tienen como raíz ese nodo. En la figura 8.2 b) la raíz del árbol es libro y este tiene 3 subárboles C1, C2 y C3, el grado de libro es tres.

**Nodo terminal u hoja:** Nodo con grado 0, no tiene subárboles, por ejemplo, el nodo C3.

**Grado de un árbol:** Grado máximo de los nodos de un árbol.

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	93/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

**Hijos de un nodo:** Nodos que dependen directamente de ese nodo, es decir, las raíces de sus subárboles. Si un nodo  $x$  es descendiente directo de un nodo  $y$ , se dice que  $x$  es hijo de  $y$ .

**Padre de un nodo:** Antecesor directo de un nodo, nodo del que depende directamente. Si un nodo  $x$  es antecesor directo de un nodo  $y$ , se dice que  $x$  es padre de  $y$ .

**Nodos hermanos:** Nodos hijos del mismo nodo padre.

**Camino:** Sucesión de nodos del árbol  $n_1, n_2, n_3, \dots, n_k$ , tal que  $n_i$  es el padre de  $n_{i+1}$ .

**Antecesoros de un nodo:** Todos los nodos en el camino desde la raíz del árbol hasta ese nodo.

**Nivel de un nodo:** Es el número de arcos que deben ser recorridos para llegar a un determinado nodo o la longitud del camino desde la raíz hasta el nodo.

**Altura:** La altura de un nodo en un árbol es la longitud del mayor de los caminos del nodo a cada hoja. La altura de un árbol es la altura de la raíz.

**Longitud de camino de un árbol:** Suma de las longitudes de los caminos a todos sus componentes.


**Bosque:** Conjunto de uno o más árboles.

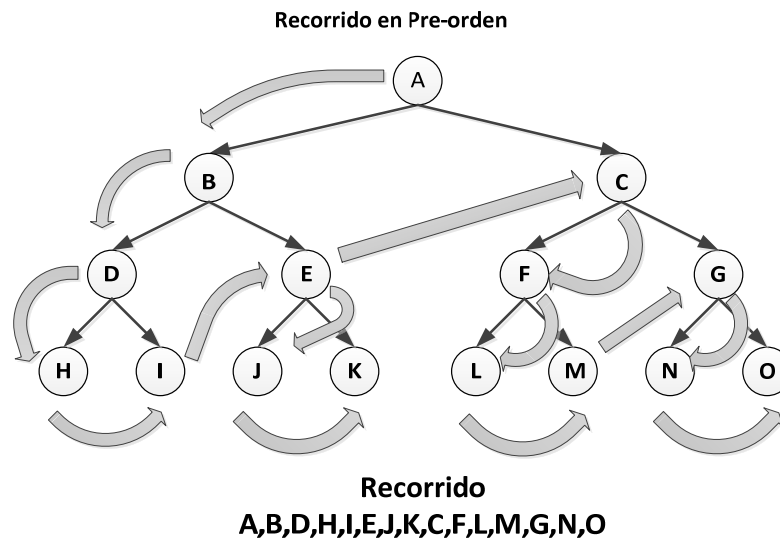
## Recorridos

El recorrido de una estructura de datos es el acceso sistemático a cada uno de sus miembros. Hay dos formas básicas de recorrer un árbol; en profundidad y en amplitud

**Recorrido en profundidad:** Siempre empieza accediendo a la raíz. Cuando es en profundidad se trata de alejarse en todo lo posible de la raíz hasta alcanzar un nodo hoja. Una vez alcanzado se da un paso atrás para intentar alejarse por un camino alternativo. Este esquema implica que por nodo se pasa varias veces: cuando se accede por primera vez y cuando se regresa de cada uno de sus hijos para acceder al siguiente o volver al padre. El tratamiento del nodo se puede hacer en cualquiera de esas ocasiones, lo que da lugar, según el momento que se elija a tres variantes: los recorridos en preorden, inorden(simétrico) y postorden[1]. El nombre dado a los recorridos es de acuerdo a la posición de la raíz.

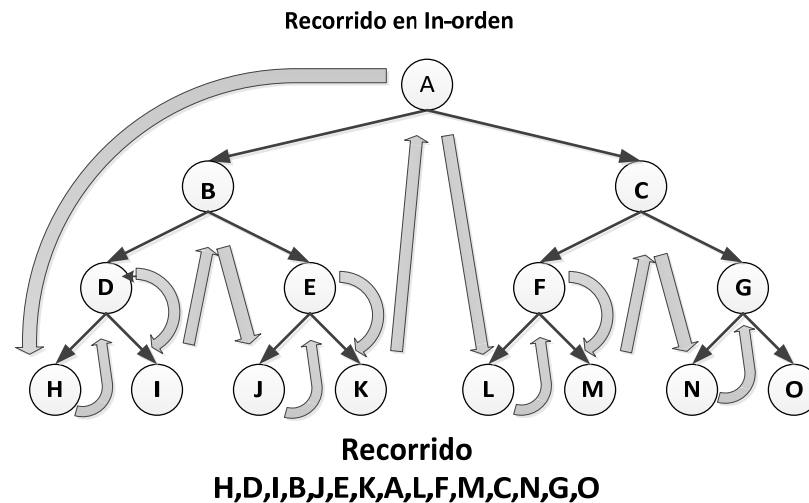
**Preorden:** Se trata primero la raíz del árbol  $y$ , a continuación, se recorren en preorden sus subárboles de izquierda a derecha (raíz - subárbol izq. - subárbol der). Figura 8.2.

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	94/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			




**Figura 8.2**

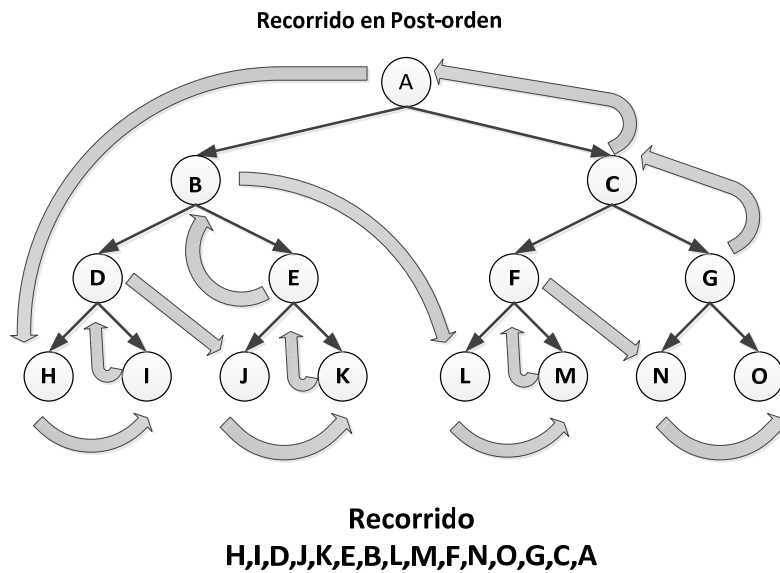
Inorden : Se recorre en inorden el primer subárbol , luego se trata de la raíz y, a continuación se recorre en inorden el resto de los subárboles (subárbol izq. - raíz - subárbol der). Figura 8.3.



**Figura 8.3**

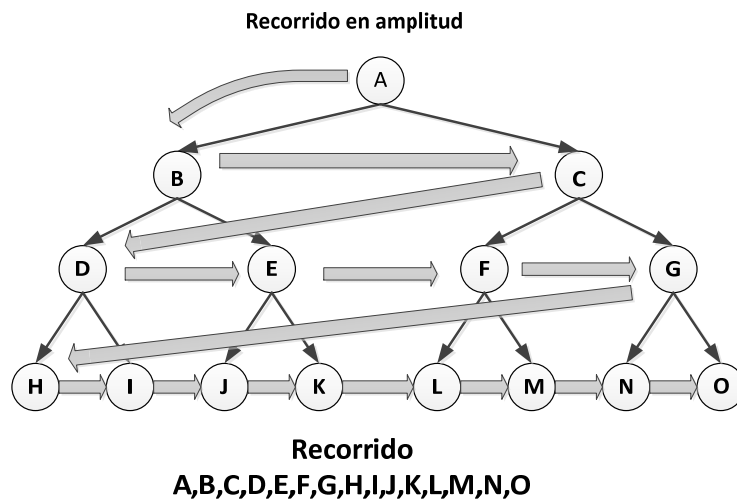
	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	95/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Postorden: Se recorren primero en postorden todos los subárboles, de izquierda a derecha, y finalmente se trata la raíz ( subárbol izq. - subárbol der. -raíz). Figura 8.4.




**Figura 8.4**

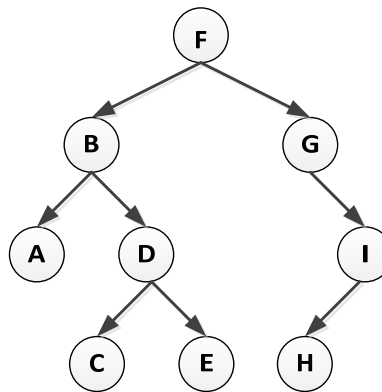
**Recorrido en anchura:** También conocida como recorrido por niveles o en amplitud. Se explora el árbol nivel a nivel, de izquierda a derecha y del primero al último. Figura 8.5.



**Figura 8.5**

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	96/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Ejemplo. Para el árbol de la figura 8.6, sus recorridos son:



**Figura 8.6**

#### **Recorrido en Profundidad:**

Secuencia de recorrido de preorden: F, B, A, D, C, E, G, I, H (raíz, izquierda, derecha)

Secuencia de recorrido de inorden: A, B, C, D, E, F, G, H, I (izquierda, raíz, derecha).


Secuencia de recorrido de postorden: A, C, E, D, B, H, I, G, F (izquierda, derecha, raíz)

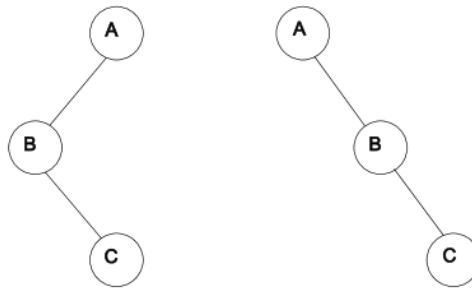
#### **Recorrido en Anchura:**

Secuencia de recorrido de orden por nivel: F, B, G, A, D, I, C, E, H

### **Árboles Binarios**

Un árbol binario es un árbol de grado dos, esto es, cada nodo puede tener dos, uno o ningún hijo. En los árboles binarios se distingue entre el subárbol izquierdo y el subárbol derecho de cada nodo. De forma que, por ejemplo, los dos siguientes árboles (Figura 8.7), a pesar de contener la misma información son distintos por la disposición de los subárboles:

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	97/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			



**Figura 8.7**

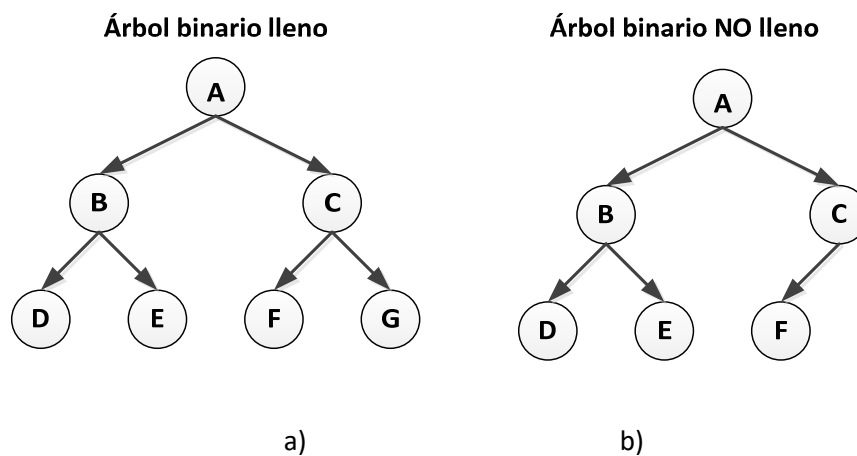
Se puede definir al árbol binario como un conjunto finito de  $m$  nodos ( $m \geq 0$ ), tal que:

- 1- Si  $m = 0$ , el árbol está vacío
- 2- Si  $m > 0$ 
  - a. Existe un nodo raíz
  - b. El resto de los nodos se reparte entre dos árboles binarios, que se conocen como subárbol izquierdo y subárbol derecho de la raíz.

A continuación, se describen algunas características de los árboles binarios.


### Árbol binario lleno

Es un árbol binario lleno si cada nodo es de grado cero o dos. O bien, si es un árbol binario de profundidad  $k$  que tiene  $2^k - 1$  nodos (es el número máximo de nodos).



**Figura 8.8[4]**



	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	98/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

### Árbol binario completo

Un árbol binario es completo si todos los nodos de grado cero o uno están en los dos últimos niveles, de forma que las hojas del último nivel ocupan las posiciones más a la izquierda de dicho nivel. En un árbol binario completo de altura  $h$ , todos los niveles, excepto posiblemente el nivel  $h$  están completamente llenos.

Si un árbol está lleno también está completo.

### Árboles equilibrados

Un árbol es equilibrado cuando la diferencia de altura entre los subárboles de cualquier nodo es máxima 1. Un árbol binario es equilibrado totalmente cuando los subárboles izquierdo y derecho de cada nodo tienen las mismas alturas, es decir, es un árbol lleno.

Se dice que un árbol completo es equilibrado y un árbol lleno es totalmente equilibrado.


**Operaciones en árboles binarios:** Un árbol binario se puede ver como un tipo de dato abstracto y algunas de las operaciones que se pueden realizar sobre él son:

- Crear y eliminar el árbol
- Verificar el estado del árbol, si está vacío o no.
- Crear y remover nodos
- Saber si el nodo es hoja o no
- Búsqueda por contenido
- Recorrer el árbol
- Obtener el padre, hijo izquierdo o hijo derecho, dado un nodo.

Una de las operaciones básicas es la creación del árbol, una vez que ya se tiene, se pueden realizar las operaciones sobre sus elementos como recorrerlo, insertar un nuevo nodo, eliminar uno existente o buscar un valor determinado. Es importante mencionar que el proceso de generación de un árbol dependerá de las reglas impuestas por una aplicación particular y el procedimiento de generación deberá, por tanto, reproducir de la manera más eficiente posible esas reglas de generación.

### Representación de los árboles binarios

Si el árbol es lleno o árbol completo, es posible encontrar una buena representación secuencial del mismo. En esos casos, los nodos pueden ser almacenados en un arreglo unidimensional,  $A$ , de manera que el nodo

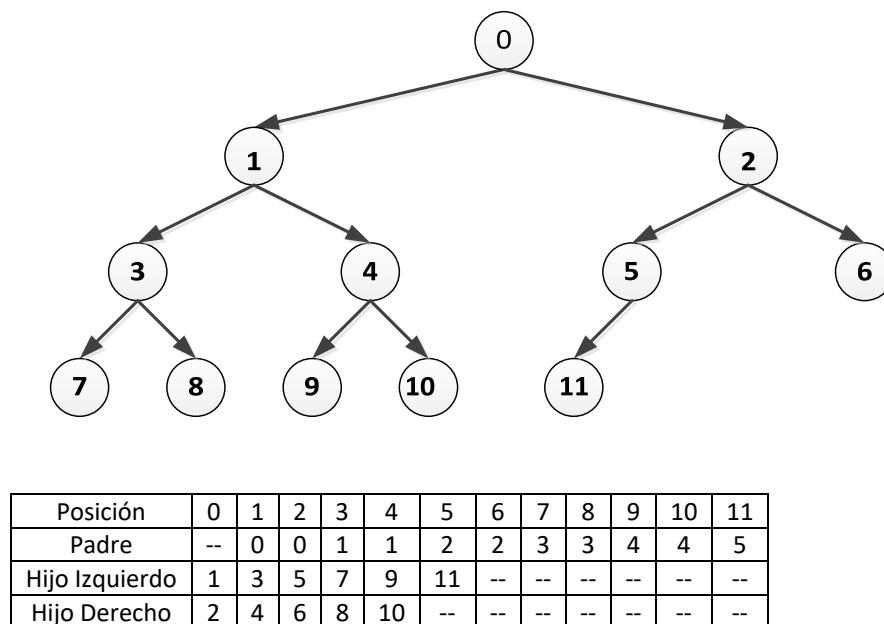
	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	99/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

numerado como  $i$  se almacena en  $A[i]$ . Esto permite localizar fácilmente las posiciones de los nodos padre, hijo izquierdo e hijo derecho de cualquier nodo  $i$  en un árbol binario arbitrario que cumpla tales condiciones.


Entonces, si un árbol binario completo con  $n$  nodos se representa secuencialmente, se cumple que, para cualquier nodo con índice  $i$ ,  $0 \leq i \leq (n - 1)$ , y se tiene que:

- El padre del nodo  $i$  estará localizado en la posición  $(i - 1)/2$  si  $i > 0$ . Si  $i = 0$ , se trata del nodo raíz y no tiene padre.
- El hijo izquierdo del nodo  $i$  estará localizado en la posición  $2 * (i + 1) - 1$  si  $2 * (i + 1) - 1 < n$ . Si  $2 * (i + 1) - 1 \geq n$ , el nodo no tiene hijo izquierdo.
- El hijo derecho del nodo  $i$  estará localizado en la posición  $2 * (i + 1)$  si  $2 * (i + 1) < n$ . Si  $2 * (i + 1) \geq n$ , el nodo no tiene hijo derecho.

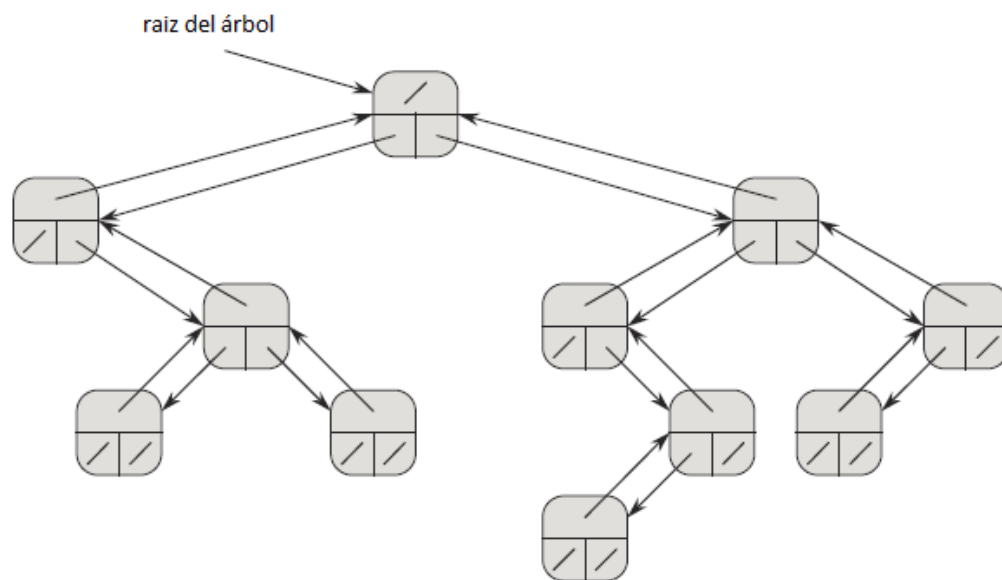
Ejemplo en la figura 8.9 se tienen tres arreglos, padre, hijo izquierdo e hijo derecho, el índice de cada arreglo representa al nodo y el contenido de cada arreglo la posición de su padre, hijo izquierdo e hijo derecho.



**Figura 8.9**

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	100/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

También es posible representar cada nodo de un árbol como un objeto o estructura, donde cada nodo contiene como atributos el valor, y las referencias a los nodos hijos su padre. En árboles binarios se pueden tener los atributos, padre, hijo izquierdo e hijo derecho los cuales pueden hacer referencia al nodo raíz, nodo izquierdo o nodo derecho respectivamente.




**Figura 8.10[2]**

### **Aplicaciones de los árboles binarios.**

Una aplicación es la representación de otro tipo de árboles. Esto es importante porque resulta más complejo manipular nodos de grado variable (número variable de relaciones) que nodos de grado fijo. Entonces es posible establecer una relación de equivalencia entre cualquier árbol no binario y un árbol binario, es decir obtener un árbol binario equivalente.

Otra aplicación de los árboles binarios es hallar soluciones a problemas cuyas estructuras son binarias, por ejemplo, las expresiones aritméticas y lógicas.

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	101/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Árboles binarios de búsqueda.

Son árboles binarios donde los nodos se identifican por una llave  $k$  y no existen dos elementos con la misma, además sus nodos están ordenados de manera conveniente para la búsqueda de alguna llave como se indica:

- Todos los elementos almacenados en el subárbol izquierdo de un nodo con valor o llave  $k$ , tienen valores menores a  $k$ .
- Todos los elementos almacenados en el subárbol derecho de un nodo con valor o llave  $k$ , tiene valores mayores o iguales a  $k$ .

Tienen la característica de que sus subárboles izquierdo y derecho son también árboles binarios de búsqueda.

Un algoritmo en pseudocódigo propuesto en [2] para realizar la búsqueda en este tipo de árboles es:

Busqueda (nodoX, llave)

Inicio

```

Si (nodoX==Ninguno) o ( valor==nodoX.llave)
    Retornar nodoX
Fin Si
Si llave < nodoX.llave
    Retornar Busqueda(nodoX.hijoIzq, llave)
En otro caso
    Retornar Busqueda(nodoX.hijoDer, llave)
Fin Si

```

Fin

El procedimiento recibe información del nodo y la llave para buscar de forma recursiva en el subárbol izquierdo o derecho según el valor de la llave.

En este tipo de árboles la obtención de las llaves se puede realizar en un orden dependiendo del recorrido que se realice (*preorden*, *in-orden*, *post orden*) y lo anterior se consigue con un procedimiento recursivo. Por ejemplo, para el recorrido en *inorden* se tiene:

*In\_Orden(Nodo)*


*inicio*

```

Si es Nodo valido
    In_Orden(Nodo.hijoIzquierdo)
    Imprimir Nodo.llave
    In_Orden(Nodo.derecho)
Fin Si

```

*Fin*

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	102/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

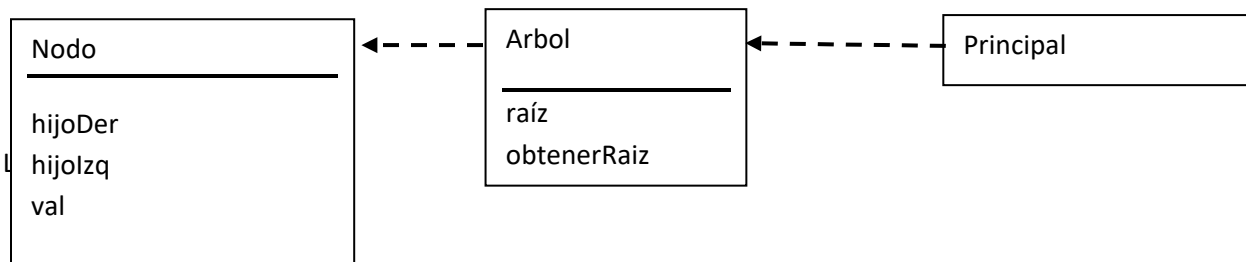
Se pueden entonces tener diferentes operaciones a realizar en esta estructura de datos, como crear nodo, insertar nodo, borrar, recorrido, buscar etc.

## Desarrollo

### Actividad 1:

Realizar un programa que forme un árbol binario de búsqueda, para lo cual se seguirá el diseño e implementación sugerido abajo.

Primero se formará el árbol binario de acuerdo al siguiente diagrama de clases y después se agregarán algunos métodos propios de un árbol binario de búsqueda.



Las implementaciones de la clase *Nodo* y *Arbol* en Python son las siguientes:

```


class Nodo:
    def __init__(self, valor):
        self.hijoIzq = None
        self.hijoDer = None
        self.val = valor
  
```

```

class Arbol:
    def __init__(self):
        self.raiz = None

    def obtenerRaiz(self):
        return self.raiz
  
```

Después, para insertar nodos se utilizará el criterio de árboles binarios de búsqueda, donde para mantener un orden entre los elementos, cualquier elemento menor está en la rama izquierda, mientras que cualquier


	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	103/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

elemento mayor está en la rama derecha. Entonces se colocan los métodos *agregar()* y *agregarNodo()* a la clase *Arbol* para ir formando el árbol binario de acuerdo al criterio mencionado.

```
def agregar(self, val):
    #Si árbol vacío, agregar nodo raíz
    if(self.raiz == None):
        self.raiz = Nodo(val)
    else:
        #Si el árbol tiene raíz
        self.agregarNodo(val, self.raiz)

def agregarNodo(self, val, nodo):
    #Si el valor a introducir es menor al valor que se encuentra
    #en el nodo actual se revisa el hijo izquierdo
    if(val < nodo.val):
        #Si hay hijo izquierdo
        if(nodo.hijoIzq != None):
            self.agregarNodo(val, nodo.hijoIzq)
        else:
            #Si no hay hijo izquierdo se crea un nodo con el valor
            nodo.hijoIzq = Nodo(val)
    #Si el valor a agregar es mayor al valor que tiene el nodo actual
    #Se revisa hijo derecho
    else:
        if(nodo.hijoDer != None):
            self.agregarNodo(val, nodo.hijoDer)
        else:
            nodo.hijoDer = Nodo(val)
```

Ahora para poder visualizar al árbol se adicionan los métodos *preorden()* e *imprimePreorden()* que permiten recorrerlo en preorden.

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	104/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```

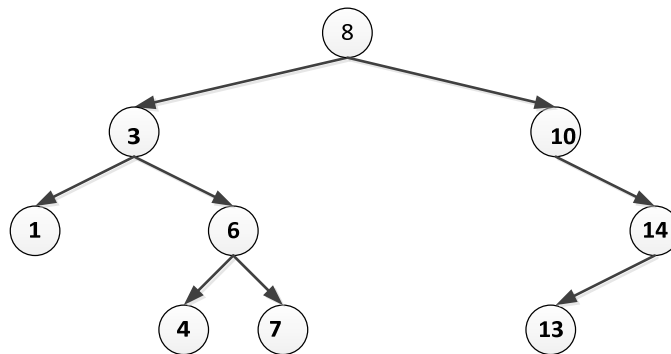
def preorden(self,nodo):
    if(nodo != None):
        print (str(nodo.val))
        if nodo.hijoIzq !=None:
            self.preorden(nodo.hijoIzq)

        if nodo.hijoDer != None :
            self.preorden(nodo.hijoDer)

def ImprimePreorden(self):
    if(self.raiz != None):
        self.preorden(self.raiz)

```


Finalmente con las clases *Nodo* y *Arbol* completas, se tiene que realizar una clase controladora que forme el siguiente árbol y se liste en preorden.



## Actividad 2

Modificar el programa de la actividad 1 para que se realice la búsqueda de un valor dado en el árbol binario.

Para ello se puede implementar el algoritmo en pseudocódigo propuesto en [2] y que se menciona en esta guía en el apartado de árboles binarios

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos II</b>	Código:	MADO-20
		Versión:	01
		Página	105/180
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

### Actividad 3

Ejercicios sugeridos por el profesor

### Referencias

[1]Zenón José

Fundamentos de Estructura de Datos  
Thomson

[2]CORMEN, Thomas, LEISERSON, Charles,et al.

Introduction to Algorithms

3rd edition

MA, USA

The MIT Press, 2009

[3]Ziviani, Nivio

Diseño de algoritmos con implementaciones en Pascal y C

Ediciones paraninfo /Thomson Learning

2007

[5] Alfred V. Aho, Jeffrey D. Ullman y John E. Hopcroft

Estructura de datos y algoritmos

S.A. ALHAMBRA MEXICANA