

Estructuras de datos no lineales

Ing Esp Alvaro Hernan Pito Burbano



SISTEMA DE ASEGURAMIENTO
INTERNO DE LA CALIDAD

Estructuras de datos no lineales

- Son aquellas que ocupan bloques de memoria no continuos / lineales. Para lidiar con el problema de la fragmentación y, sobre todo del crecimiento dinámico
- A las estructuras de datos no lineales se les llama también estructuras de datos multienlazadas y se caracteriza por no existir una relación de sus elementos es decir que un elemento puede estar con cero uno o mas elementos.
- Se trata de estructuras de datos en las que cada elemento puede tener varios sucesores y/o varios predecesores

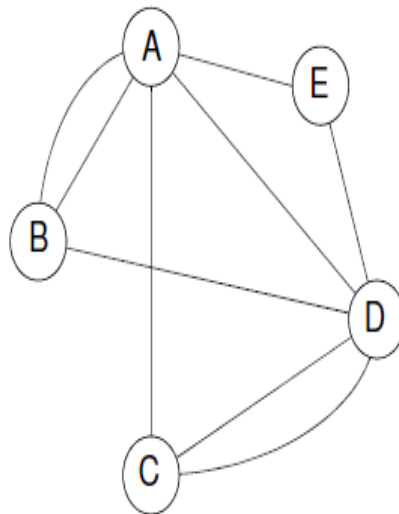
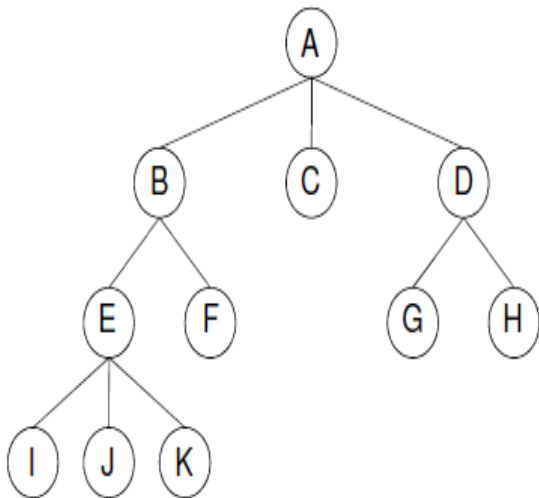
Tipos de Estructuras de datos no lineales



SISTEMA DE ASEGURAMIENTO
INTERNO DE LA CALIDAD

Tipos de Estructuras de datos no lineales

- Árboles: Cada elemento sólo puede estar enlazado con su predecesor y sus sucesores.
- Grafos: Cada elemento puede estar enlazado a cualquier otro.

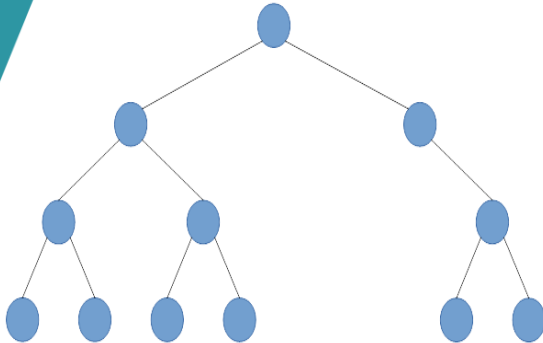


Arboles

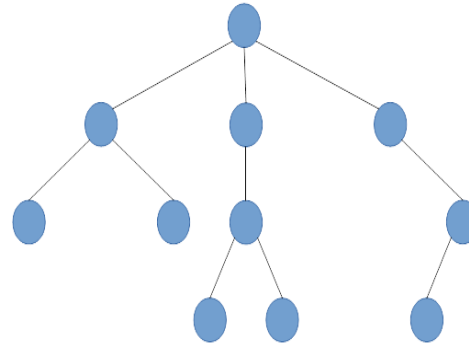
- Un árbol es una estructura de datos homogénea, dinámica y no lineal, en la que cada nodo(elemento) puede tener varios nodos posteriores, pero solo puede tener un nodo interior.
- -Un árbol es dinámico por que su estructura puede cambiar durante la ejecución de un programa, y no lineal, ya que cada nodo de árbol puede contener varios nodos que dependen de él, Se usan para todo tipo de jerarquías.
- Aplicaciones algorítmicas(ordenación, búsqueda).

Clasificación

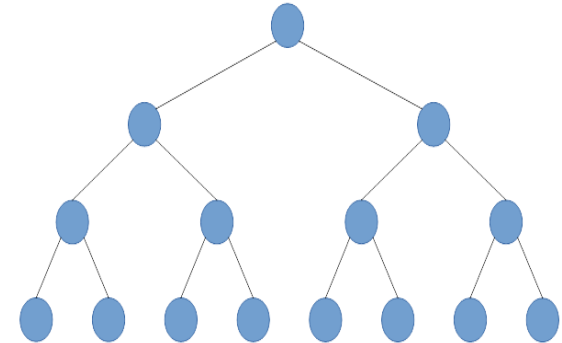
Ejemplos gráficos de árboles binarios



Árbol equilibrado



Árbol desequilibrado



Árbol completo

Árboles



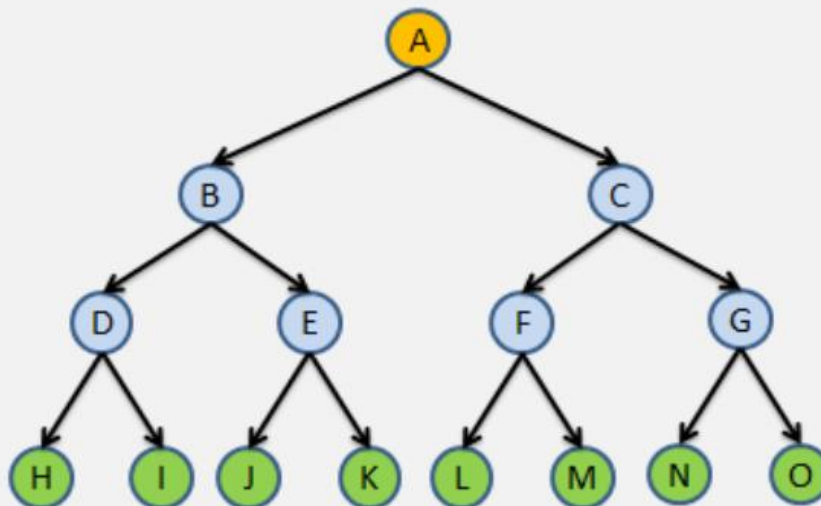
SISTEMA DE ASEGURAMIENTO
INTERNO DE LA CALIDAD



Árboles

Los Árboles son las estructuras de datos mas utilizadas, pero también una de las mas complejas, Los Árboles se caracterizan por almacenar sus nodos en forma jerárquica y no en forma lineal como las Listas enlazadas, Colas, Pilas.

Estructura Jerárquica



V.S.

Estructura Lineal



Fig. 1: La imagen muestra la diferencia entre las estructuras de datos lineales y las no lineales como lo son los Árboles.



SISTEMA DE ASEGURAMIENTO
INTERNO DE LA CALIDAD



INSTITUCIÓN UNIVERSITARIA
COLEGIO MAYOR DEL CAUCA



Datos importantes de los Árboles

- **Nodos:** Se le llama Nodo a cada elemento que contiene un Árbol.
- **Nodo Raíz:** Se refiere al primer nodo de un Árbol, Solo un nodo del Árbol puede ser la Raíz.
- **Nodo Padre:** Se utiliza este termino para llamar a todos aquellos nodos que tiene al menos un hijo.
- **Nodo Hijo:** Los hijos son todos aquellos nodos que tiene un padre.
- **Nodo Hermano:** Los nodos hermanos son aquellos nodos que comparte a un mismo padre en común dentro de la estructura.
- **Nodo Hoja:** Son todos aquellos nodos que no tienen hijos, los cuales siempre se encuentran en los extremos de la estructura.
- **Nodo Rama:** Estos son todos aquellos nodos que no son la raíz y que además tiene al menos un hijo.

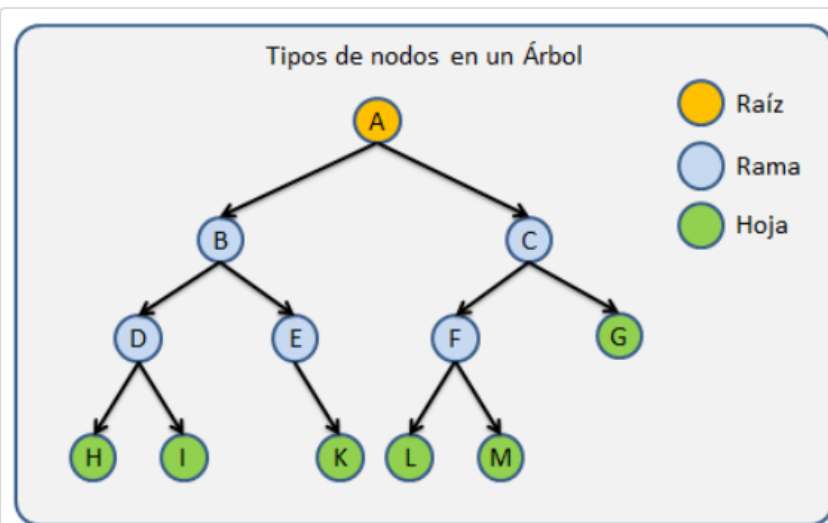


Fig. 2: La imagen muestra de forma gráfica cuales son los nodos Raíz, Rama, Hoja.

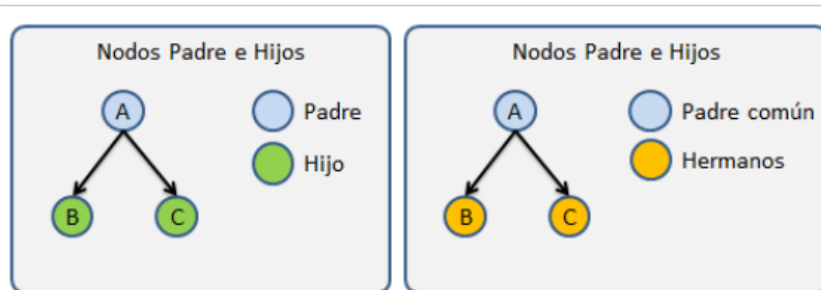


Fig.3: La siguiente imagen muestra de forma gráfica los nodos Padre, Hijo y Hermanos

Los arboles a demás de los nodos tiene otras propiedades importantes que son utilizadas en diferente ámbitos los cuales son:

Nivel: Nos referimos como nivel a cada generación dentro del árbol. Por ejemplo, cuando a un nodo hoja le agregamos un hijo, el nodo hoja pasa a ser un nodo rama pero a demás el árbol crece una generación por lo que el Árbol tiene un nivel más. Cada generación tiene un número de Nivel distinto que las demás generaciones.

- Un árbol vacío tiene 0 niveles
- El nivel de la Raíz es 1
- El nivel de cada nodo es calculado contando cuantos nodos existen sobre el, hasta llegar a la raíz + 1, y de forma inversa también se podría, contar cuantos nodos existes desde la raíz hasta el nodo buscado + 1.

Altura: Le llamamos Altura al número máximo de niveles de un Árbol.

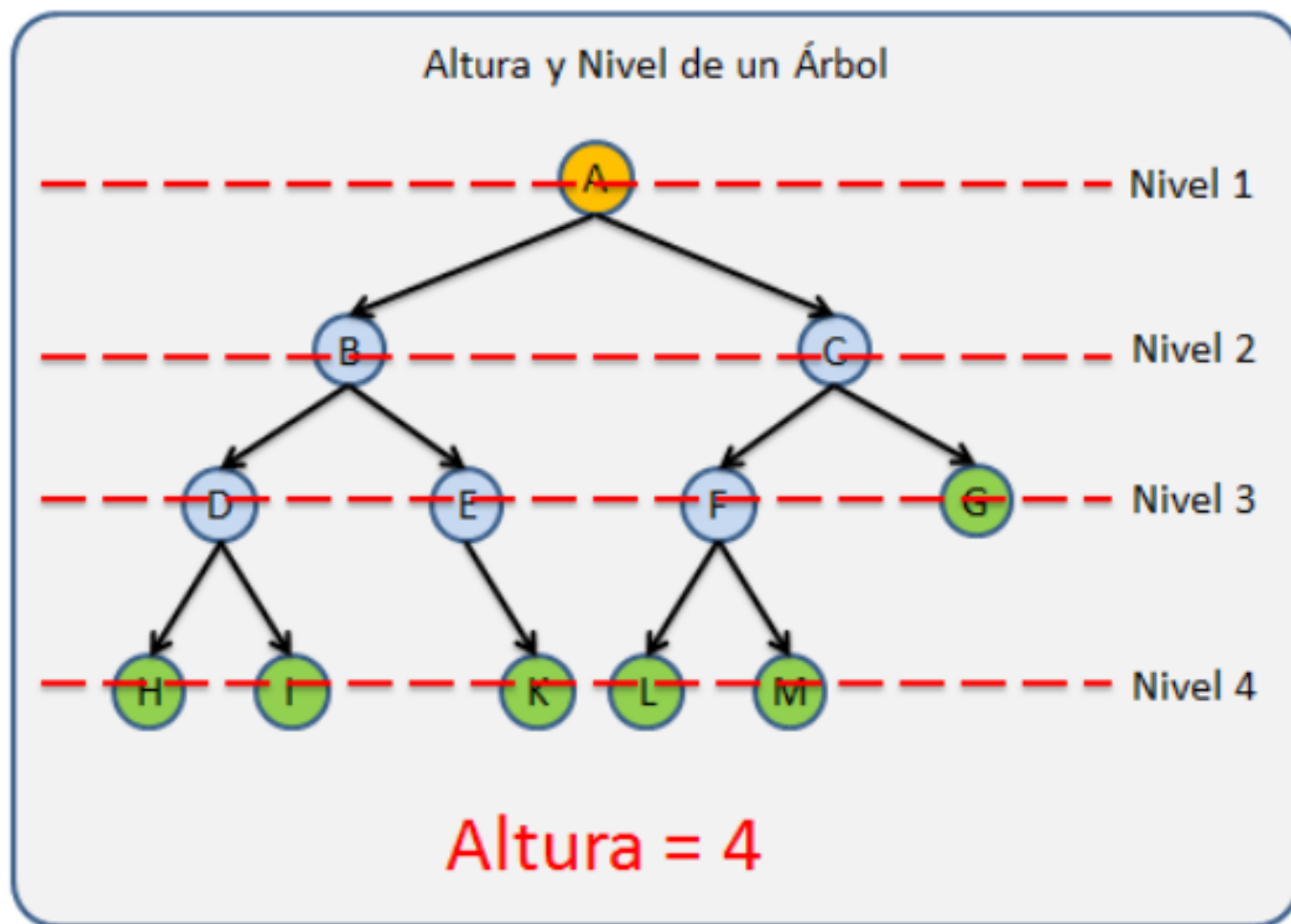
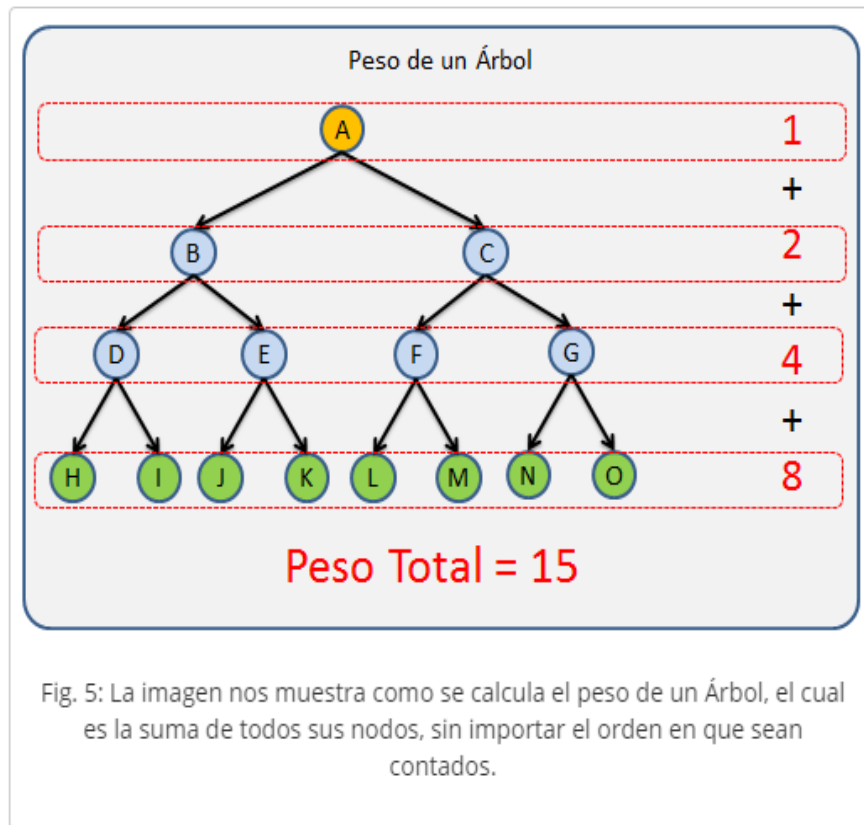


Fig. 4: En la imagen se muestran los Niveles y la Altura de un Árbol.

La **altura** es calculado mediante recursividad tomando el nivel mas grande de los dos sub-árboles de forma recursiva de la siguiente manera:

altura = max(altura(hijo1), altura(hijo2), altura(hijoN)) + 1

Peso: Conocemos como peso a el número de nodos que tiene un Árbol. Este factor es importante por que nos da una idea del tamaño del árbol y el tamaño en memoria que nos puede ocupar en tiempo de ejecución(Complejidad Espacial en análisis de algoritmos.)

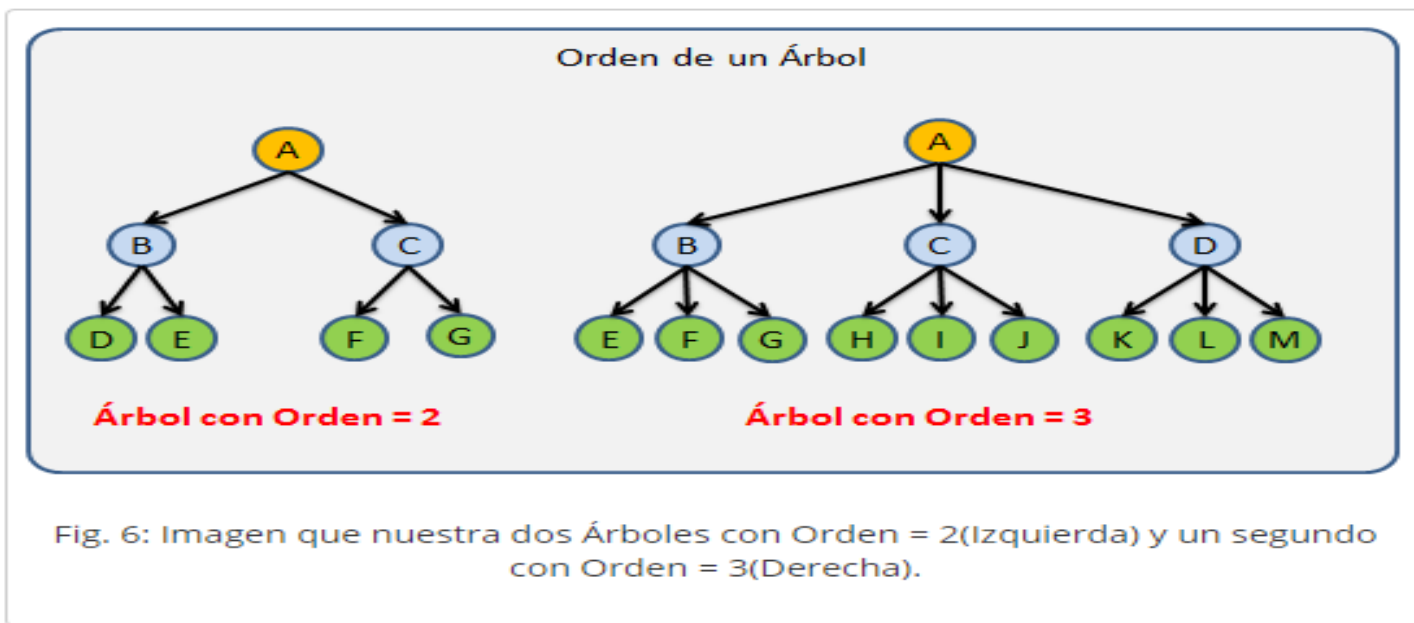


El peso se puede calcular mediante cualquier tipo de recorrido el cual vaya contando los nodo a medida que avanza sobre la estructura. El peso es un árbol es igual a la suma del peso de los sub-árboles hijos + 1

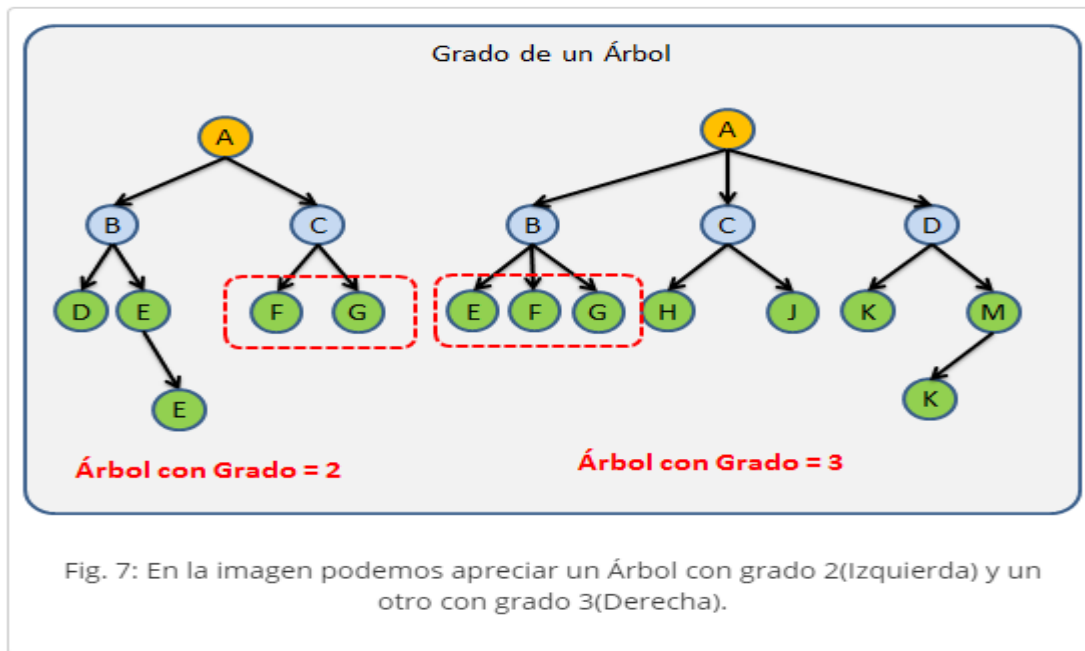
$\text{peso} = \text{peso}(\text{hijo1}) + \text{peso}(\text{hijo2}) + \text{peso}(\text{hijoN}) + 1$

Nota: Los tipos de recorridos los veremos mas adelante.

Orden: El Orden de un árbol es el número máximo de hijos que puede tener un Nodo.



Grado: El grado se refiere al número mayor de hijos que tiene alguno de los nodos del Árbol y esta limitado por el Orden, ya que este indica el número máximo de hijos que puede tener un nodo.



El grado se calcula contando de forma recursiva el número de hijos de cada sub-árbol hijo y el número de hijos del nodo actual para tomar el mayor, esta operación se hace de forma recursiva para recorrer todo el árbol.

grado =

max(contarHijos(hijo1), contarHijos(hijo2),
contarHijos(hijoN), contarHijos(this))

Sub-Árbol: Conocemos como Sub-Árbol a todo Árbol generado a partir de una sección determinada del Árbol, Por lo que podemos decir que un Árbol es un nodo Raíz con N Sub-Árboles.

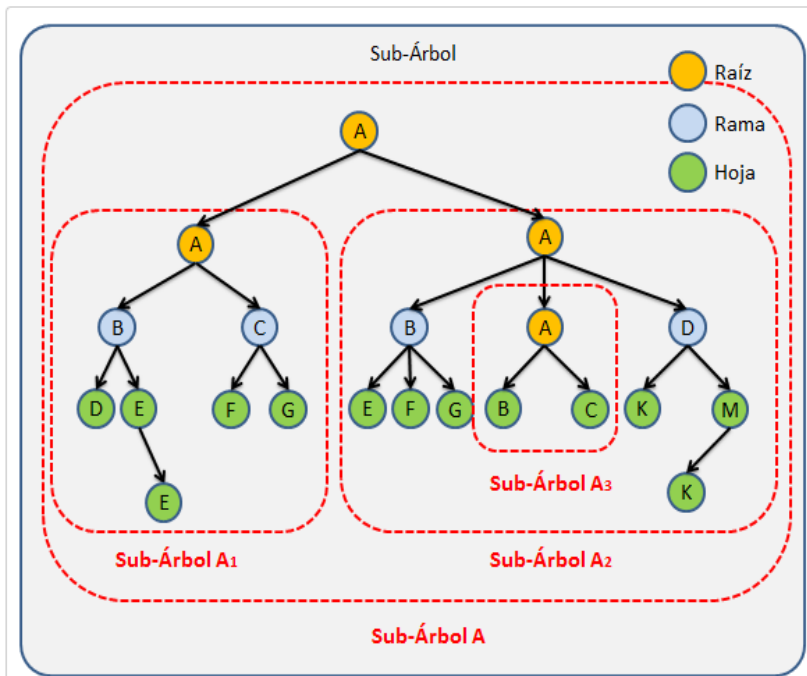
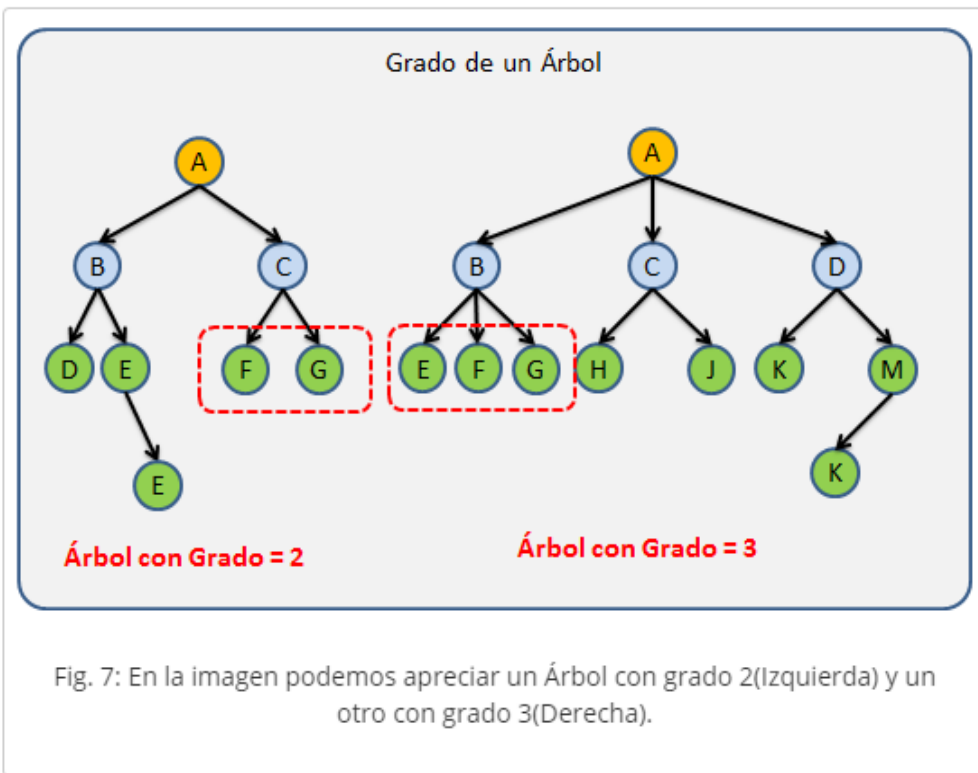


Fig. 8: En la imagen se puede apreciar que un Árbol está compuesto por una serie de Sub-Árboles los cuales conforman toda la estructura.

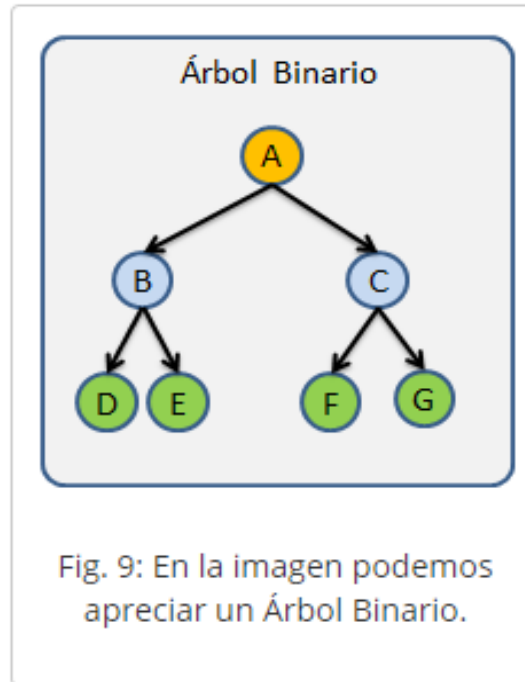
Existen escenarios donde podemos sacar un Sub-Árboles del Árbol para procesarlo de forma separada, de esta forma el Sub-Árboles pasa a ser un Árbol independiente, También podemos eliminar Sub-Árboles completos, Agregarlos, entre otras operaciones.

Árbol n-ario

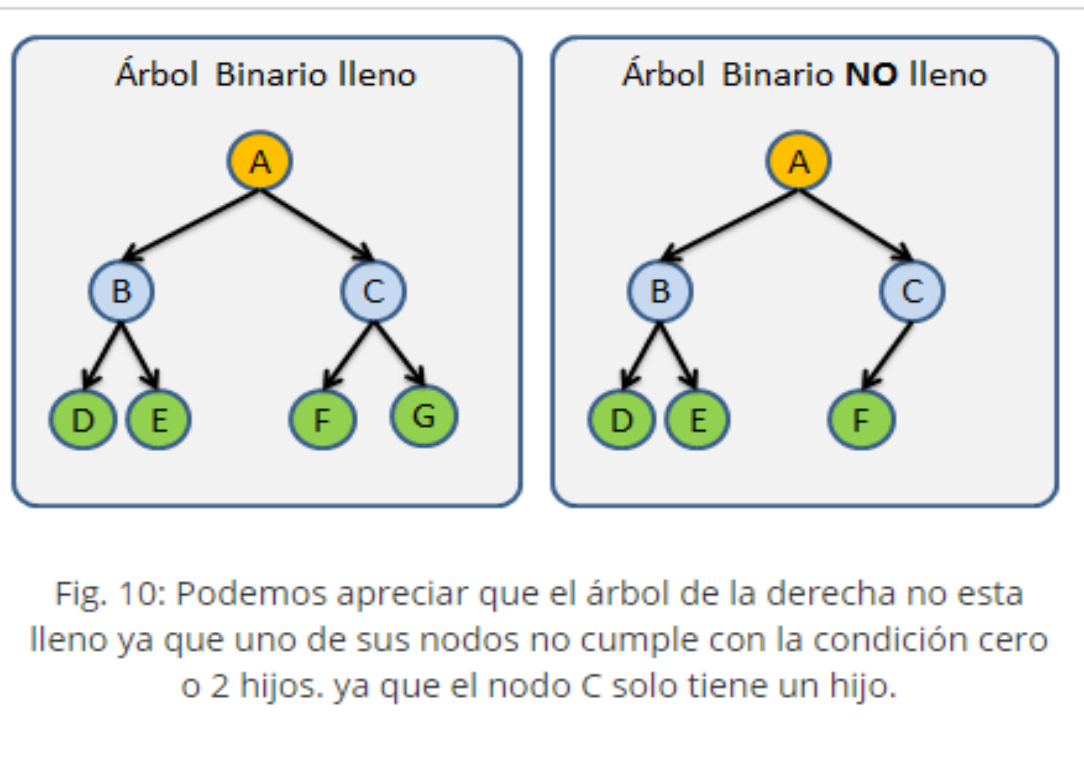
Los arboles **n-arios** son aquellos arboles donde el **número máximo de hijos por nodo** es de **N**, en la figura 7 podemos apreciar dos árboles con grado 2 y grado 3, estos dos arboles también los podemos definir como Árbol n-ario con $n = 2$ y $n=3$ respectivamente.



Árboles binarios: Esta estructura se caracteriza por que cada nodo solo puede tener máximo 2 hijo, dicho de otra manera es un Árbol n-ario de Grado 2.



**Árbol binario
lleno:** Es aquel
que el que todos
los nodos tiene
cero o 2 hijos
con excepción
de la Raíz.



Árbol binario perfecto: Es un Árbol lleno en donde todas las Hojas están en el mismo **Nivel**.



INTO
AD

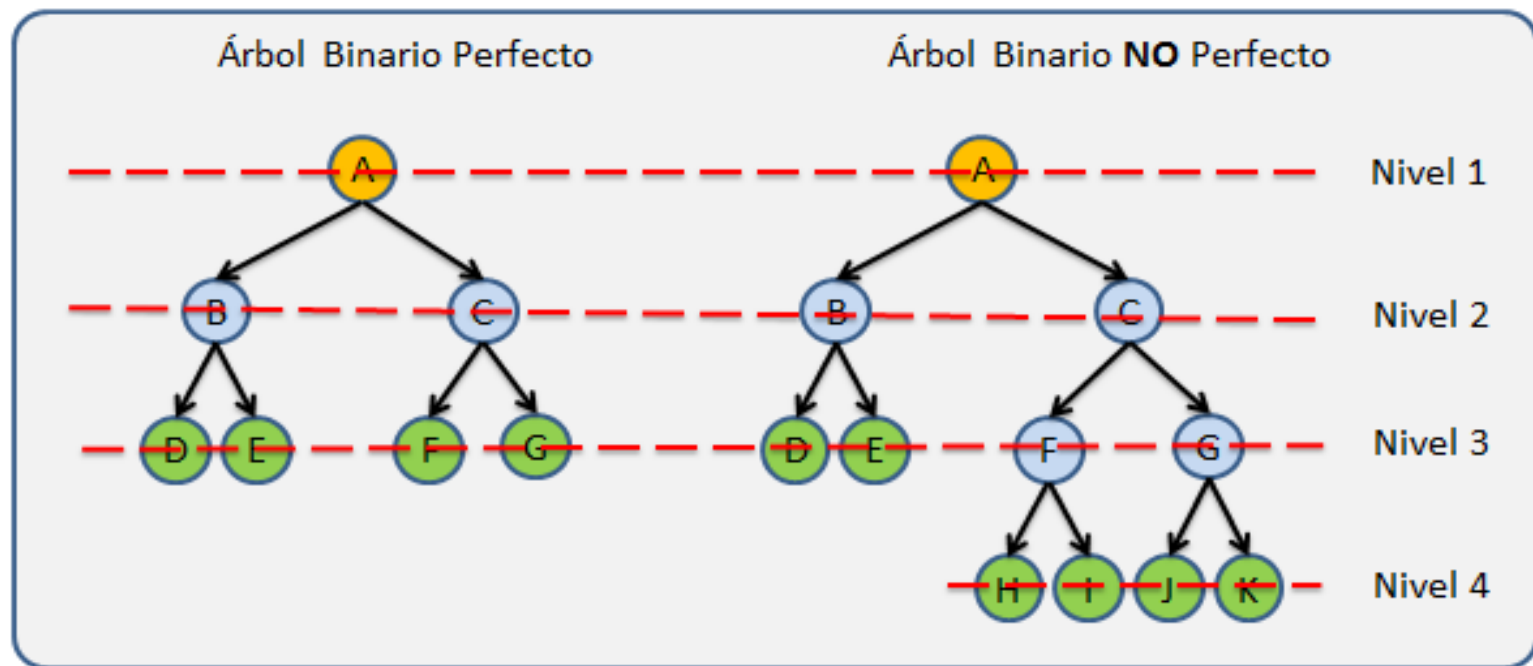


Fig. 11: En la imagen podemos apreciar que el árbol de la izquierda tiene todas sus hojas al mismo nivel y que además está lleno, lo que lo convierte en un árbol binario perfecto. Sin embargo, del lado derecho podemos ver que aunque el árbol está lleno no tiene todas las hojas al mismo nivel lo que hace que no sea un árbol binario perfecto pero sí lleno.

Recorrido sobre Árboles

Los recorridos son algoritmos que nos permiten recorrer un árbol en un orden específico, los recorridos nos pueden ayudar encontrar un nodo en el árbol, o buscar una posición determinada para insertar o eliminar un nodo.

Básicamente podemos catalogar las búsquedas en dos tipos, las búsquedas en profundidad y las búsquedas en amplitud.

UNIMAYOR

Recorrido Pre-orden:

El recorrido inicia en la Raíz y luego se recorre en pre-orden cada uno de los sub-árboles de izquierda a derecha.

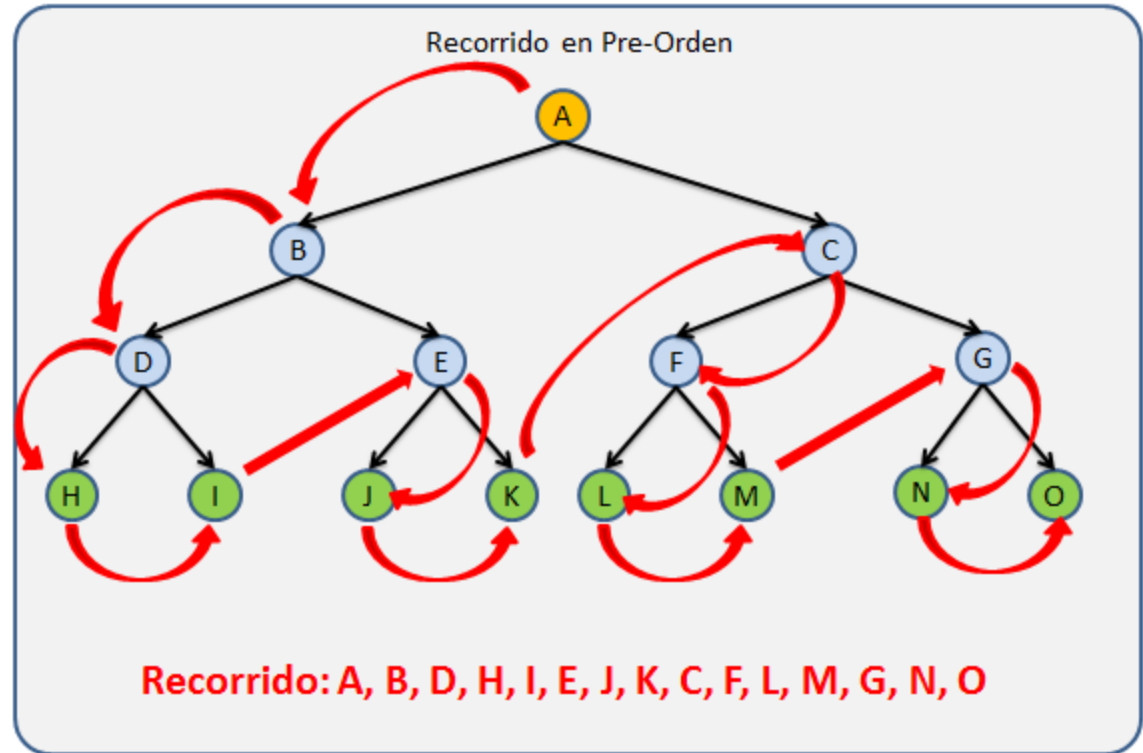


Fig. 12: En la imagen podemos ver el orden en que es recorrido el árbol iniciando desde la Raíz.

Recorrido orden:

Pos-

Se recorre el pos-orden cada uno de los sub-árboles y al final se recorre la raíz.

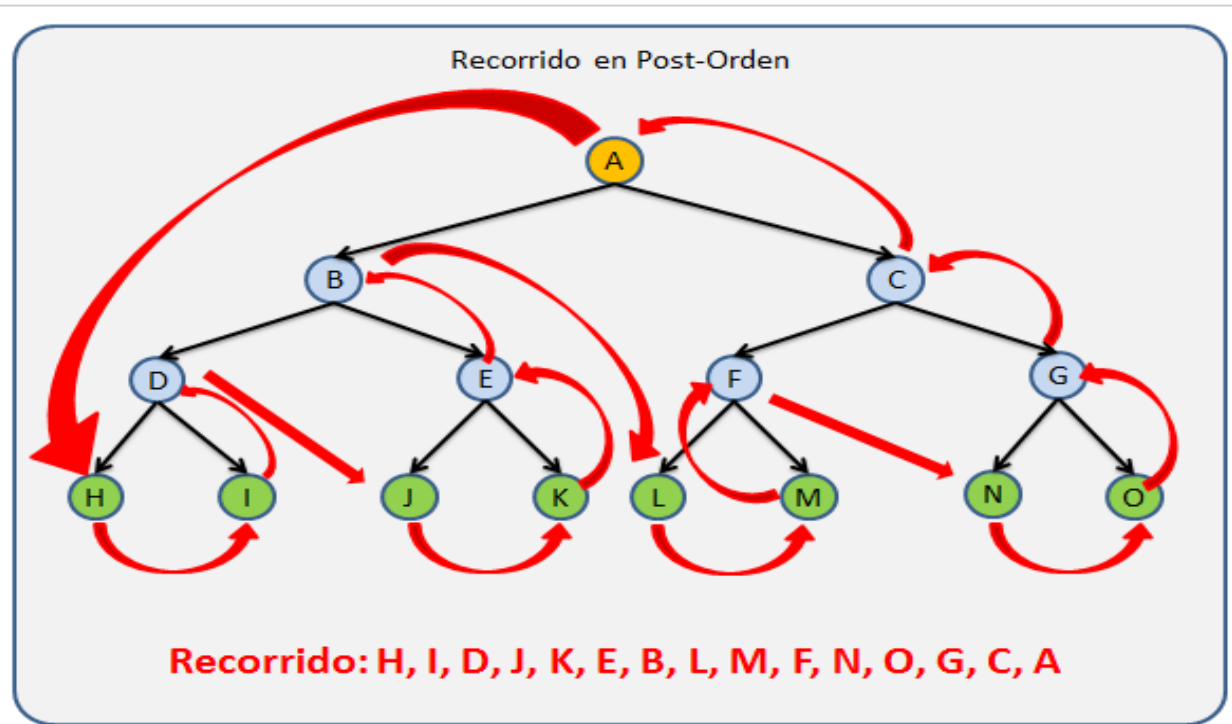


Fig. 14: En la imagen podemos observar como se realiza el recorrido en Pos-Orden, Sin embargo es importante notar que el primer nodo que se imprime no es la Raíz pues en este recorrido la Raíz de cada Sub-Árbol es procesado al final, ya que toda su descendencia ha sido procesada.

Recorrido in-orden:

Se recorre en in-orden el primer sub-árbol, luego se recorre la raíz y al final se recorre en in-orden los demás sub-árboles

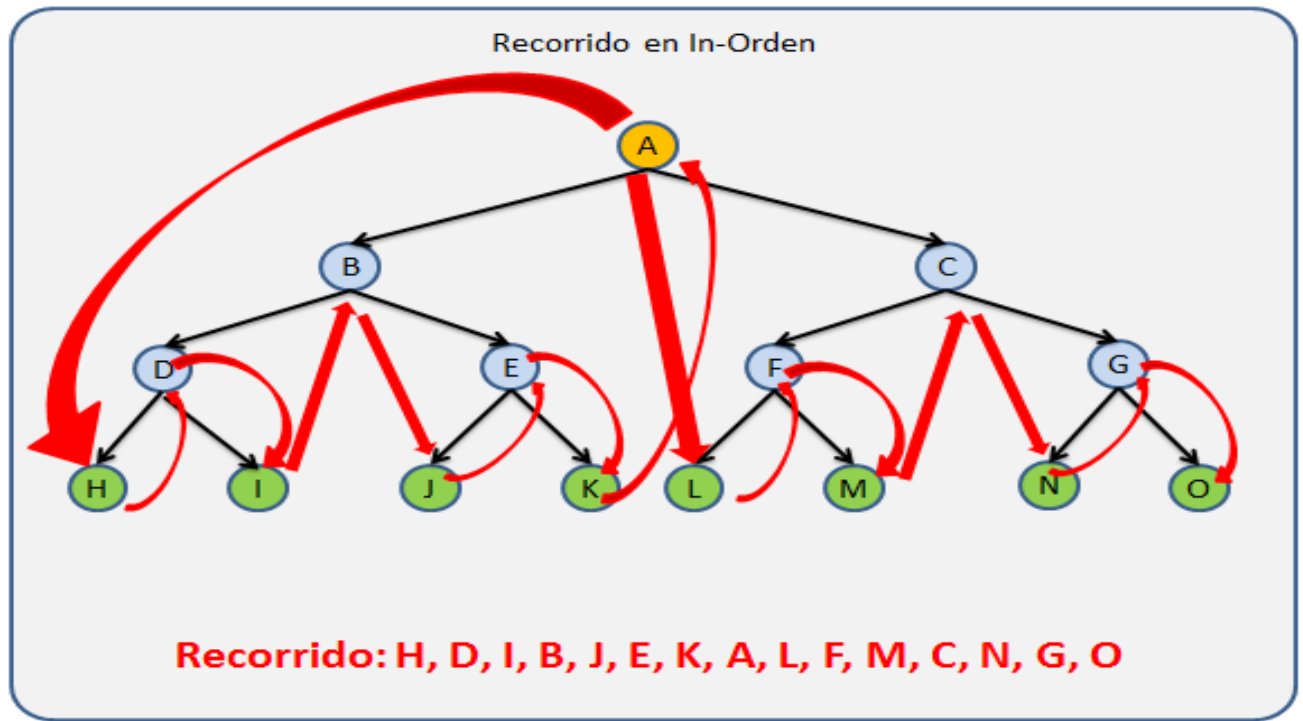
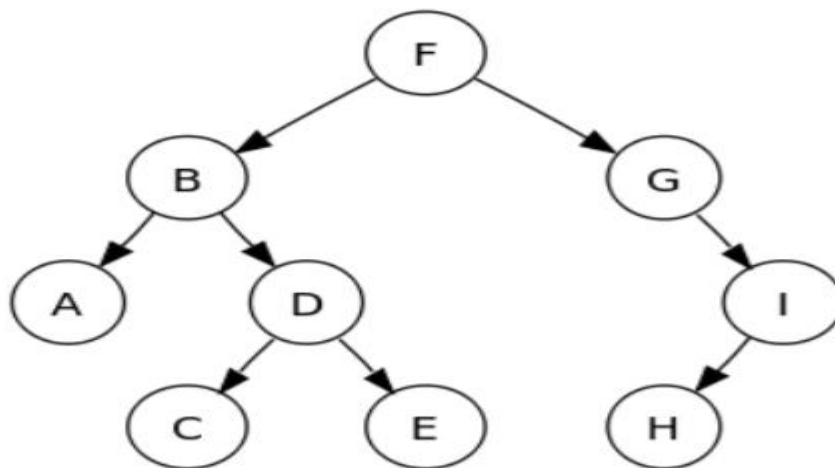


Fig. 16: En la imagen se muestra como es el recorrido In-Orden, Podemos apreciar que la Raíz no es el primero elemento en ser impreso pues este recorrido recorre su rama izquierda, luego la raíz del sub-árbol y luego la rama derecha.

Ejemplo: Aquí podemos observar un árbol y el recorrido de el.



Solución:

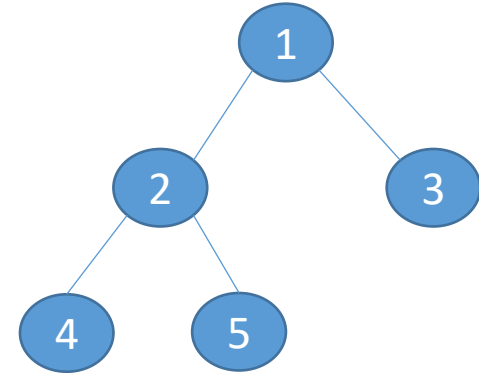
Profundidad-primero

- 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); note cómo esto produce una secuencia ordenada
- Secuencia de recorrido de postorden: A, C, E, D, B, H, I, G, F (izquierda, derecha, raíz)

Recorrido en Pre-orden de un árbol Raíz Izquierda Derecha

Preorden.py > ...

```
1 class Nodo:
2     def __init__(self, valor):
3         self.valor = valor
4         self.izquierdo = None
5         self.derecho = None
6
7 def preorden(raiz):
8     if raiz:
9         print(raiz.valor)
10        preorden(raiz.izquierdo)
11        preorden(raiz.derecho)
12
13 # Ejemplo de uso
14 raiz = Nodo(1)
15 raiz.izquierdo = Nodo(2)
16 raiz.derecho = Nodo(3)
17 raiz.izquierdo.izquierdo = Nodo(4)
18 raiz.izquierdo.derecho = Nodo(5)
19
20 print("Recorrido en preorden del árbol:")
21 preorden(raiz)
```

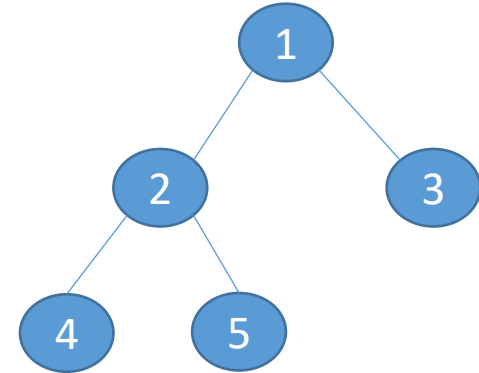


1,2,4,5,3

Recorrido en Pos-orden de un árbol Izquierda Derecha Raíz


posorden.py > ...

```
1 class Nodo:
2     def __init__(self, valor):
3         self.valor = valor
4         self.izquierdo = None
5         self.derecho = None
6
7 def posorden(raiz):
8     if raiz:
9         posorden(raiz.izquierdo)
10        posorden(raiz.derecho)
11        print(raiz.valor)
12
13 # Ejemplo de uso
14 raiz = Nodo(1)
15 raiz.izquierdo = Nodo(2)
16 raiz.derecho = Nodo(3)
17 raiz.izquierdo.izquierdo = Nodo(4)
18 raiz.izquierdo.derecho = Nodo(5)
19
20 print("Recorrido en posorden del árbol:")
21 posorden(raiz)
```

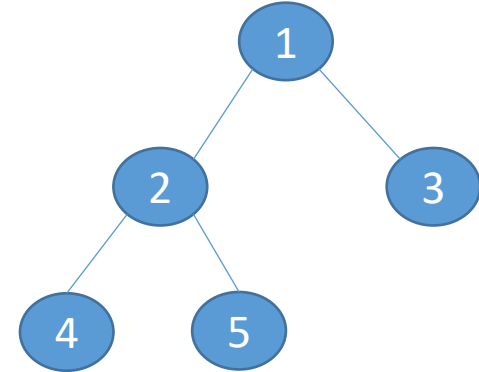


4,5,2,3,1

Recorrido en In-orden de un árbol Izquierda Raíz Derecha

 Inorden.py > ...

```
1 class Nodo:
2     def __init__(self, valor):
3         self.valor = valor
4         self.izquierdo = None
5         self.derecho = None
6
7 def inorden(raiz):
8     if raiz:
9         inorden(raiz.izquierdo)
10        print(raiz.valor)
11        inorden(raiz.derecho)
12
13 # Ejemplo de uso
14 raiz = Nodo(1)
15 raiz.izquierdo = Nodo(2)
16 raiz.derecho = Nodo(3)
17 raiz.izquierdo.izquierdo = Nodo(4)
18 raiz.izquierdo.derecho = Nodo(5)
19
20 print("Recorrido en inorden del árbol:")
21 inorden(raiz)
```



4,2,5,1,3

PREGUNTAS



SISTEMA DE ASEGURAMIENTO
INTERNO DE LA CALIDAD





SISTEMA DE ASEGURAMIENTO
INTERNO DE LA CALIDAD



Gracias



**SISTEMA DE ASEGURAMIENTO
INTERNO DE LA CALIDAD**

Somos Institución de Educación Superior Pública sujeta a inspección y vigilancia por MinEducación

Claustro de la Encarnación, Carrera 5 # 5 - 40 / Edificio Bicentenario, Carrera 7 # 2-34
Casa Obando, Calle 3 # 6-52 / Sede Zona Norte, Barrio La Ximena, Carrera 6 # 46N-44 /
Sede Administrativa, Carrera 7 # 3-60 - Cuarto Piso
Teléfono: PBX: (602) 8274178- Línea Nacional Gratuita 018000931018
www.unimayor.edu.co / Popayán.