



UNIVERSIDAD NACIONAL DE CAJAMARCA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS



ALGORITMOS Y ESTRUCTURA DE DATOS II

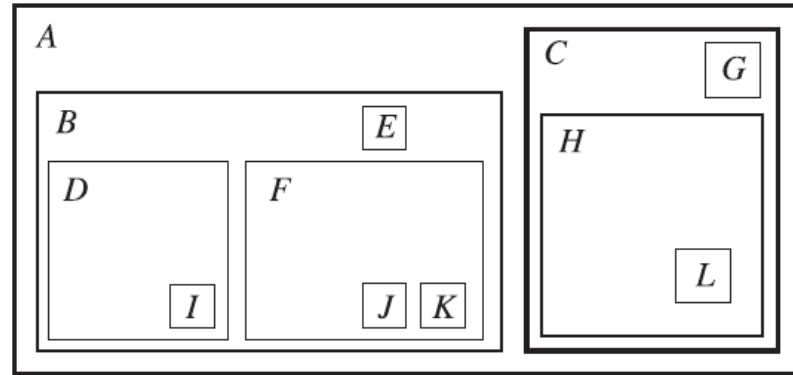
13: ÁRBOLES

INTRODUCCIÓN

Semana 13 - Árboles



QUÉ PUEDE OBSERVAR EN LA FIGURA

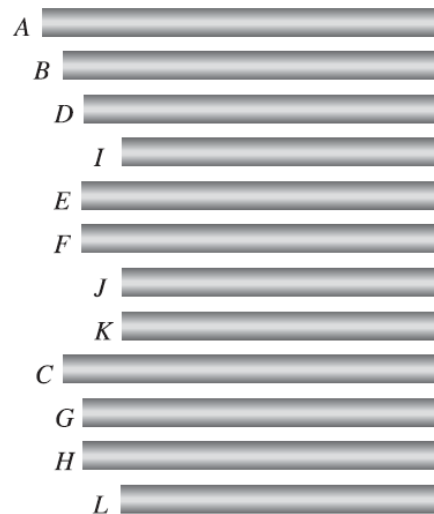


a)

$(A (B (D (I), E, F, (J, K)), C (G, H (L))))$

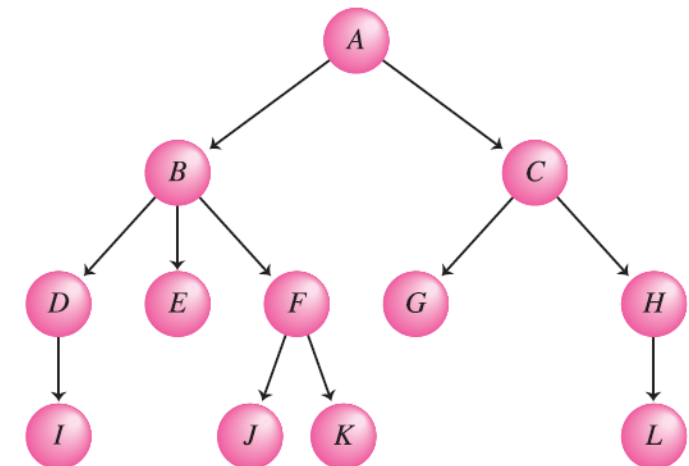
b)

1.A, 1.1.B, 1.1.1.D, 1.1.1.1.I, 1.1.2.E, 1.1.3.F, 1.1.3.1.J, 1.1.3.2.K, 1.2.C, 1.2.1.G, 1.2.2.H, 1.2.2.1.L



d)

c)



e)

RECORDEMOS



- En una pila cuál es el objetivo del método top o peek
- De acuerdo a lo revisado que utilidad encuentra en una pila
- ¿Qué notaciones conoces y cuáles son sus características?
- ¿Qué tipos de colas conocemos?
- A parte delas colas estudiadas identifique otro tipo de colas y cómo son implementadas.

¿QUÉ PASARÍA SI...

- Me piden recorrer un árbol de tal manera que sus elementos se muestren ordenados?



LOGRO ESPERADO



- **Al término de la sesión, el estudiante resuelve dos ejercicios de ejemplos o aplicaciones de los árboles a problemas de la vida real, lo sustenta a través de su código java, explicando su solución propuesta.**

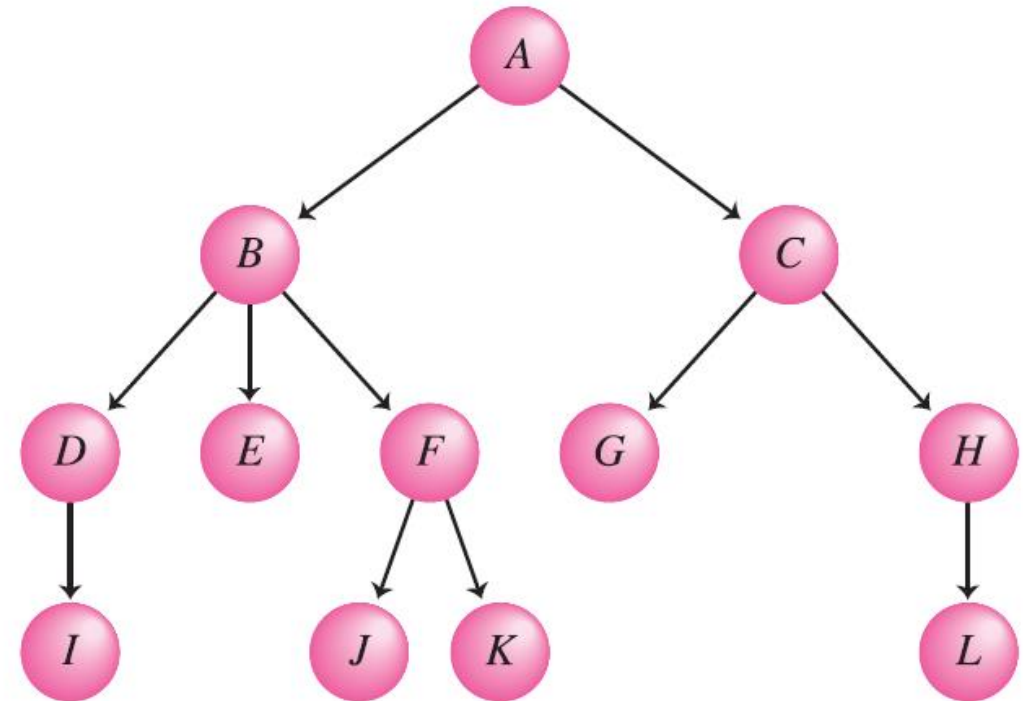
DESARROLLO DEL TEMA

Estructura de Datos



ARBOLES

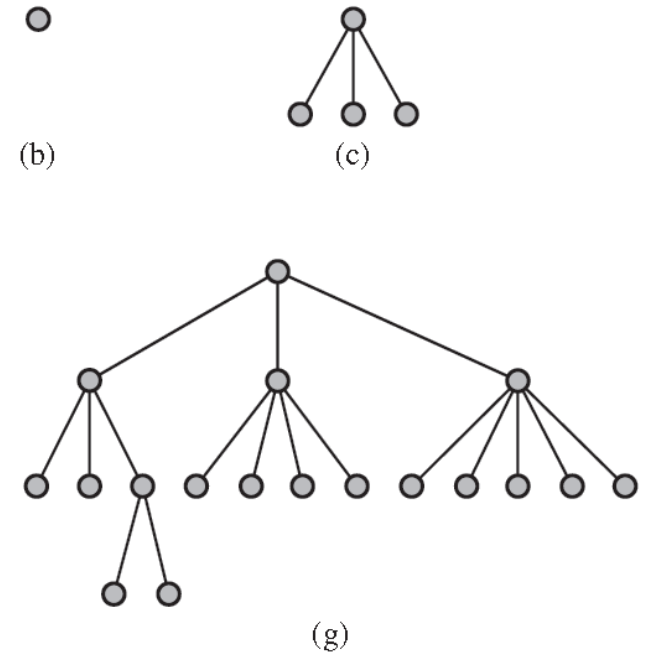
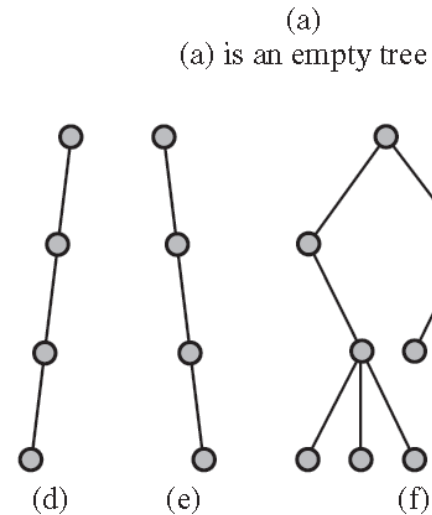
- Un árbol es una estructura no lineal en la que cada nodo puede apuntar a uno o varios nodos, puesto que cada elemento apunta a uno o varios elementos del mismo tipo; esto es, dado un elemento, no hay un único camino a seguir
- También se suele dar una definición recursiva: un árbol es una estructura compuesta por un dato y varios árboles.



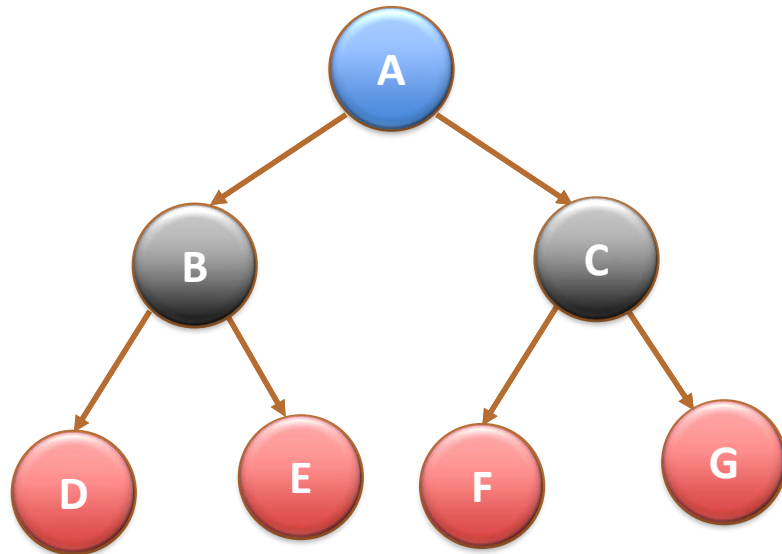
ÁRBOLES



- Una estructura vacía es un árbol vacío.
- En esta estructura existe sólo un nodo sin padre, que es la raíz del árbol.
- Todo nodo, a excepción del nodo raíz, tiene uno y sólo un padre.



EJEMPLOS



- Padre de C:



- Padre de E:



- Padre de G



- Padre de A:

NO

- Hijos de A:



- Hijos de C:



- Hijos de F:

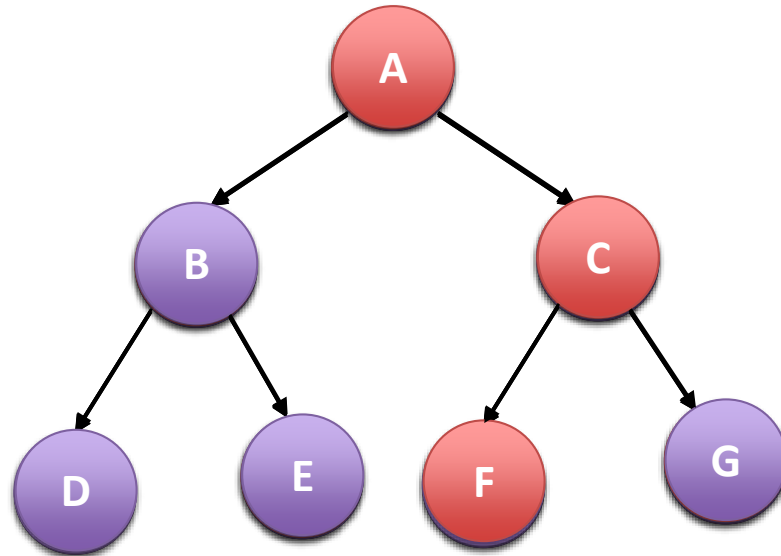
NO

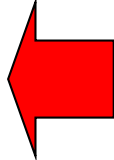
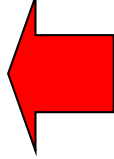
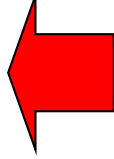
OBSERVACIÓN

- Para todo nodo k , distinto de la raíz, existe una única secuencia de la forma:
 - $k_0, k_1, k_2, k_3, \dots, k_n$, donde k_0 =raíz y $k_n=k$
 - Con $n \geq 1$, donde k_i es el sucesor de k_{i-1} , para $1 \leq i \leq n$, o sea, cada nodo k_i de la secuencia es la raíz de otro subárbol.



EJEMPLOS



- Secuencias
 - de A a G 
 - de A a E 
 - de A a F 
- C es sucesor de A y F es sucesor de C

DEFINICIONES

Grado de un nodo: cantidad de hijos de un nodo.

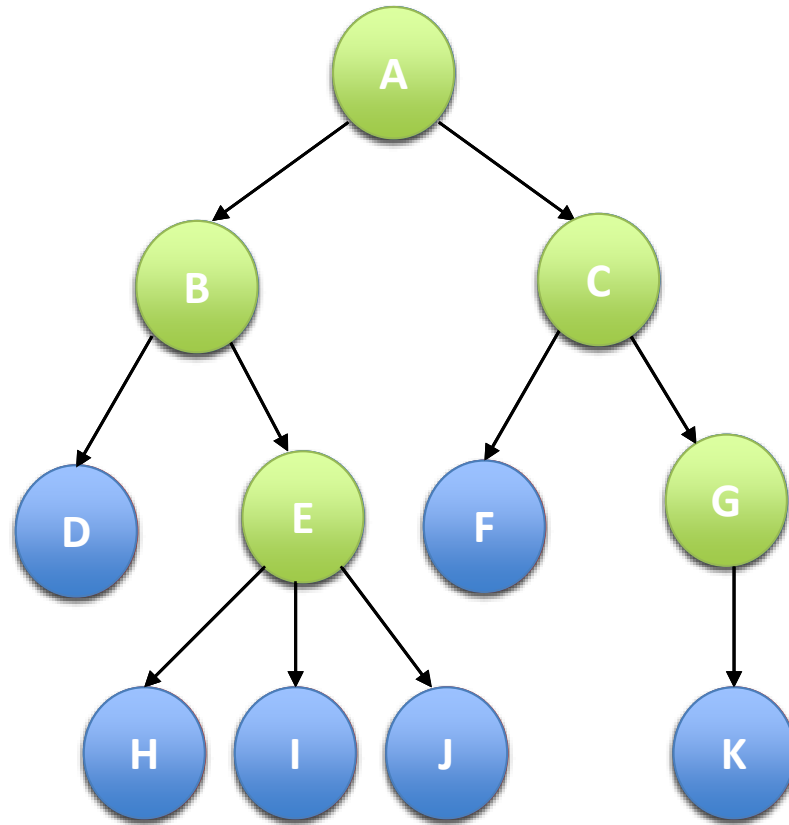
Grado de un árbol al mayor de los grados de todos sus nodos.

Nodo hoja a un nodo sin hijos o con grado = 0.

Nodo rama a un nodo que tiene hijos, o sea, a la raíz de un subárbol.



EJEMPLOS



- Grado

- de A:

2

- de E:

3

- de G:

1

- de J:

0

- Grado del árbol:

3

- Nodos hojas:

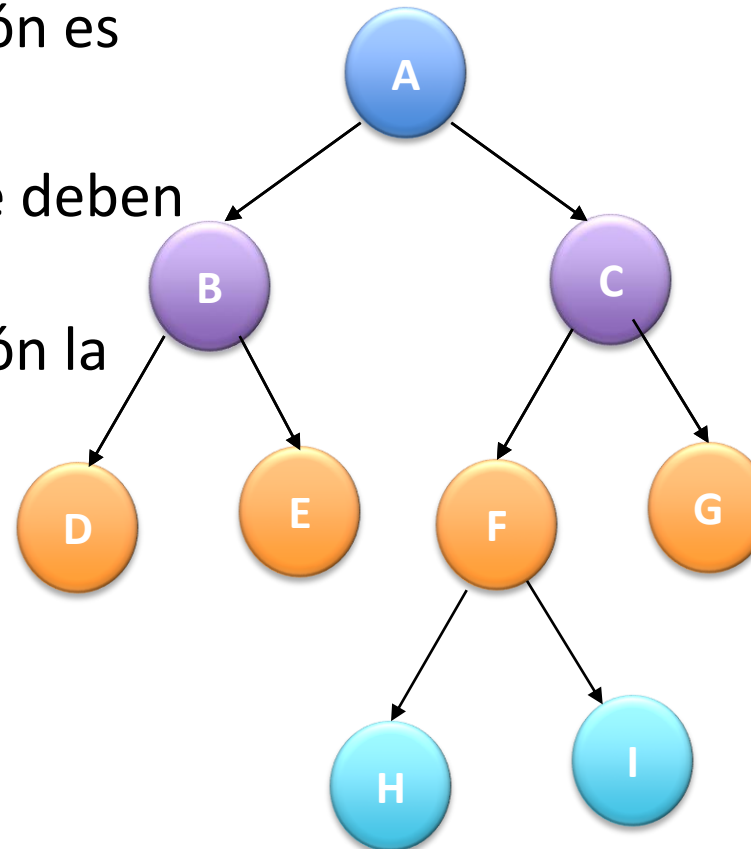
D, H, I, J, F, K

- Nodos ramas:

A, B, C, E, G

DEFINICIONES. EJEMPLOS

- **Nivel de un nodo** al nivel de su padre más uno. Por definición, la raíz del árbol tiene nivel 0. Esta definición es recursiva.
- **Nivel** es el número de arcos que deben ser recorridos para llegar a un determinado nodo. Por definición la raíz tiene nivel 1.



Nivel

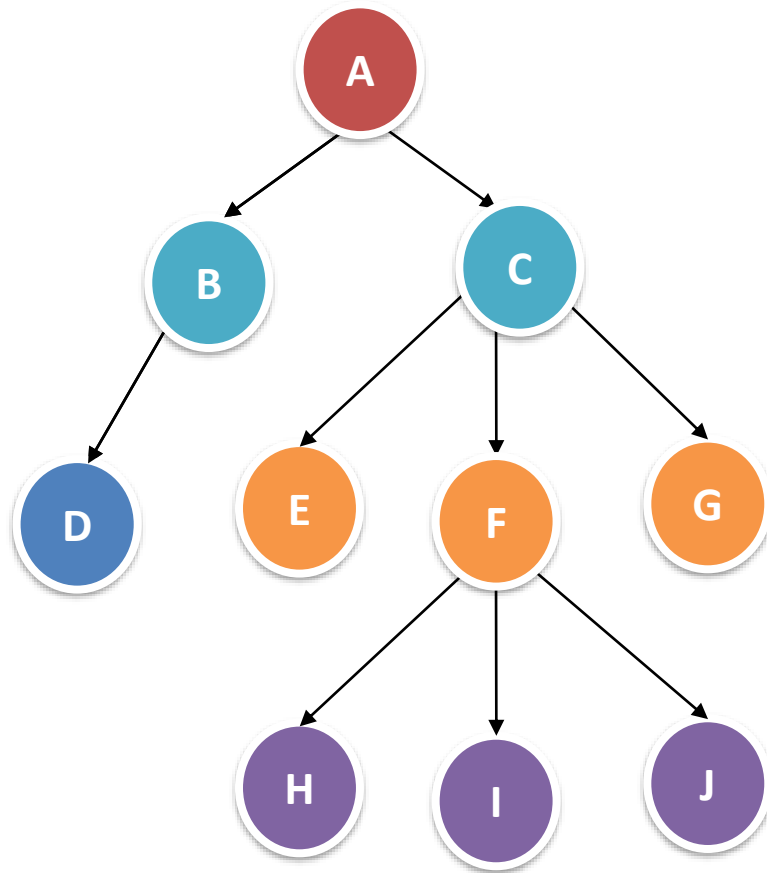
- de A: 0
- de E: 2
- de B: 1
- de I: 3
- de G: 2







DEFINICIONES



- Padre de un nodo, al nodo raíz del subárbol más pequeño que contiene a dicho nodo y en el cual él no es raíz.
- Hijo de un nodo, al (los) nodo(s) raíz(ces) de uno de sus subárboles.
- Predecesor de un nodo, al nodo que le antecede en un recorrido del árbol.
- Hermano de un nodo, a otro nodo hijo de su padre.

EJEMPLOS



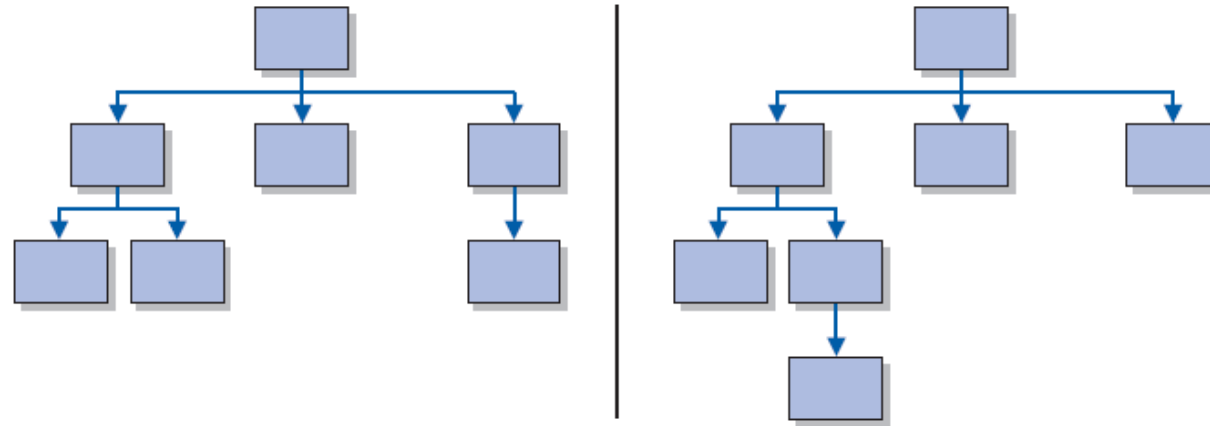
- Padre de G: 
- Hijos de C:   
- Hermanos de I:  

CLASIFICACIÓN

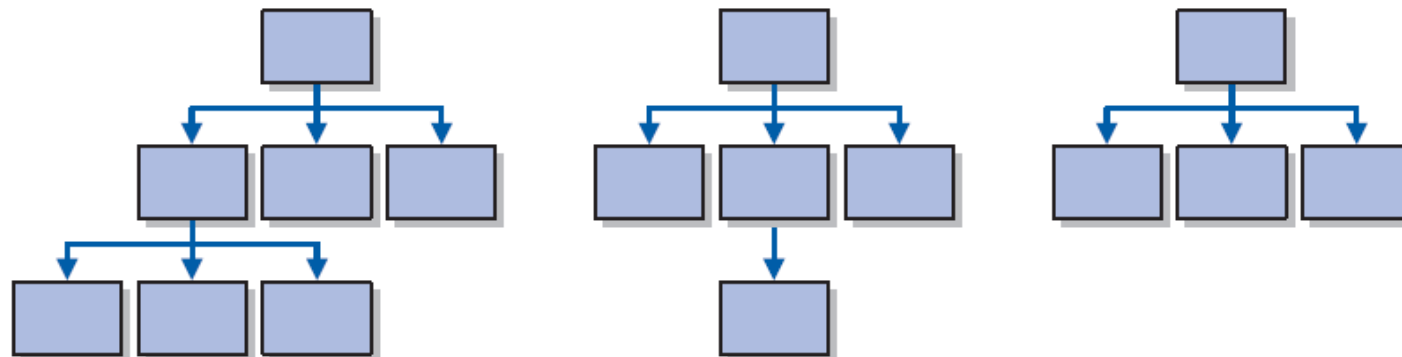


- El criterio más importante es el número máximo de hijos que cada nodo del árbol pueda tener. Este valor se denomina en ocasiones orden del árbol.
- Un árbol que no tenga ningún límite en lo que respecta al número de hijos que un nodo puede tener se denomina *árbol general*. Un árbol que limite cada nodo a no más de n hijos se denomina *árbol n -ario*.
- **Árboles binarios**, cuando cada nodo tiene como máximo dos hijos
- Otra forma de clasificar los árboles es verificando si son **equilibrados** o no.
- Un **árbol se considera equilibrado** si todas las hojas del árbol se encuentran en el mismo nivel o, como máximo, con un nivel de diferencia unas respecto de otras.
- **Árbol completo** si está equilibrado y todas las hojas situadas en el nivel inferior están en la parte izquierda del árbol

CLASIFICACIÓN



Árboles equilibrados y no equilibrados.



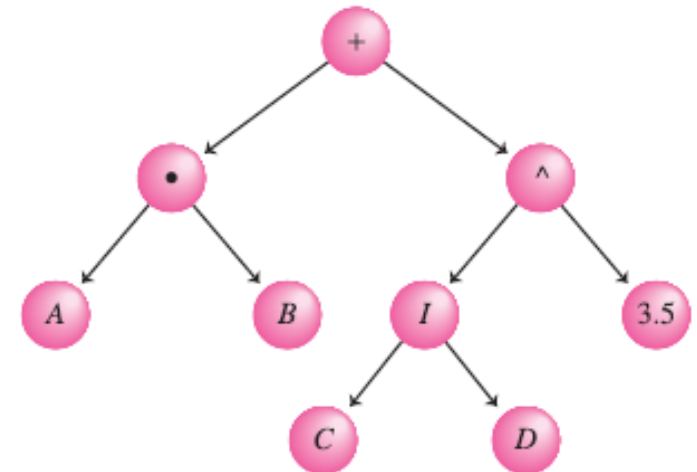
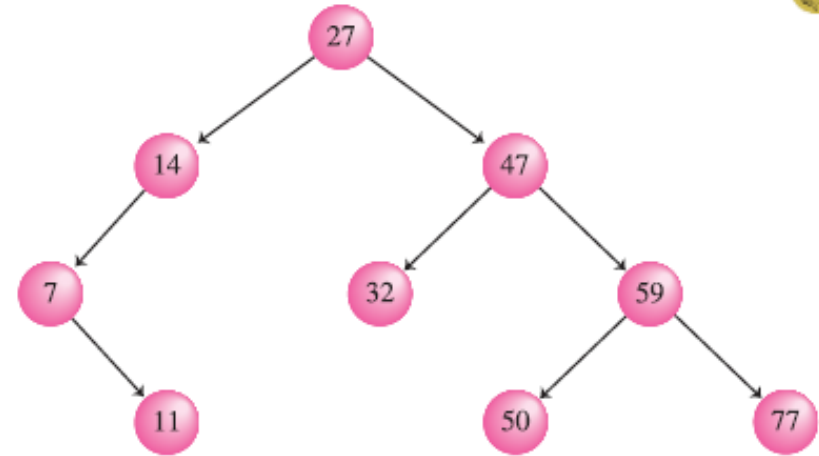
Algunos árboles completos.

DEFINICIÓN DE ÁRBOL BINARIO

Un **árbol binario** (en inglés **binary tree**) es un árbol ordenado de, a lo sumo, grado 2.

Aclaraciones:

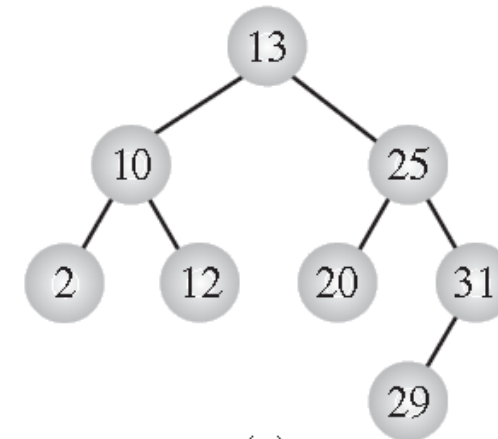
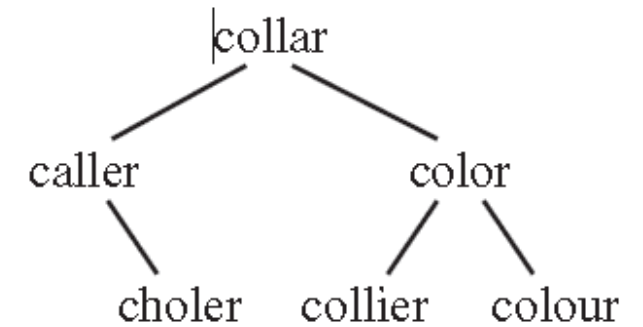
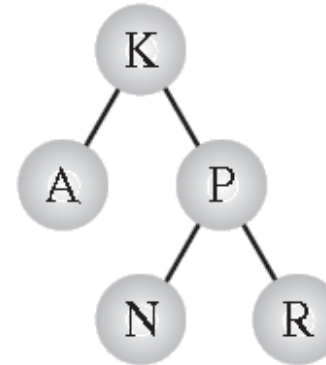
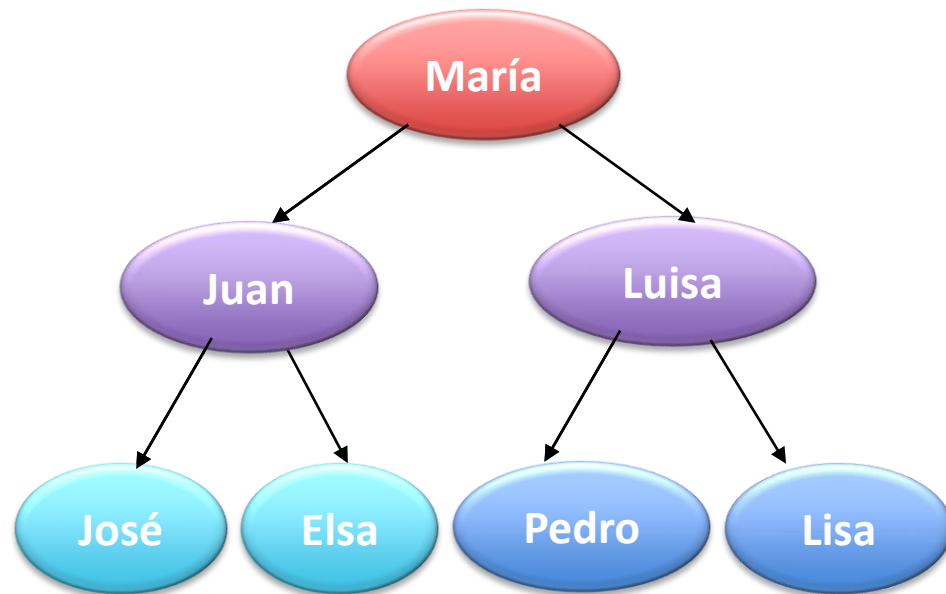
- A lo sumo grado 2 significa que cada nodo tiene como máximo dos hijos, o sea, dos subárboles.
- Al ser ordenado el árbol, importa el orden de los subárboles, es decir, que será necesario especificar de cada nodo cuál es el hijo izquierdo y cuál el hijo derecho.



EJEMPLOS

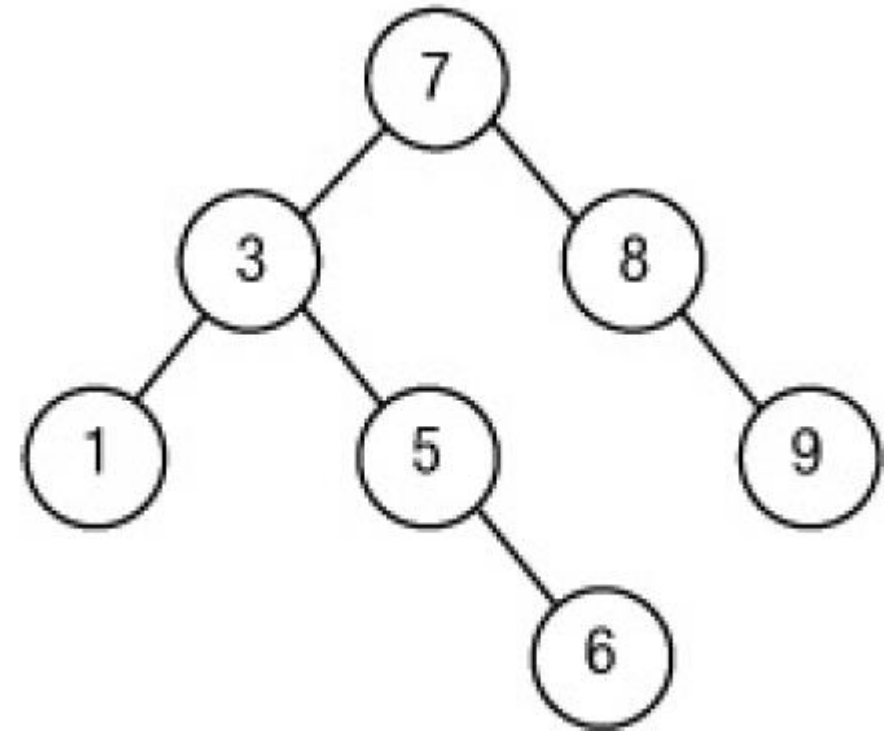


- El árbol genealógico es un árbol binario.
 - Cada nodo tiene dos hijos
 - Es significativo el orden de los subárboles.
 - Otros ejemplos se muestran a la derecha



ÁRBOL BINARIO DE BÚSQUEDA

- O ABB es un árbol binario en el cual se cumple que para todo elemento los elementos mayores que él se ubican en su rama derecha, mientras que los elementos menores están en su rama izquierda.
- Cada elemento se almacena una sola vez, por lo que no existen elementos repetidos. A menos que se indique lo contrario.





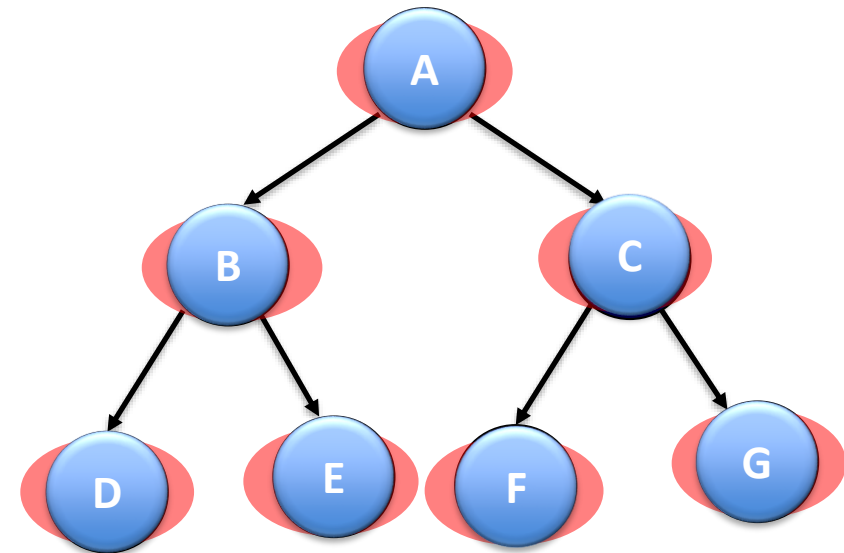
RECORRIDOS DE UN ÁRBOL BINARIO

- Los recorridos se clasifican de acuerdo al momento en que se visita la raíz del árbol y los subárboles izquierdo y derecho.
- Existen tres recorridos:
 - Preorden
 - Orden
 - Postorden
- Recorrido en Preorden: Si visitamos primero el padre, luego el hijo izquierdo y finalmente el hijo derecho.
- Recorrido en orden o inorden: Si visitamos primero el hijo izquierdo, luego el padre y finalmente el hijo derecho.
- Recorrido en Postorden: Si visitamos primero el hijo izquierdo, luego el hijo derecho y finalmente el padre.

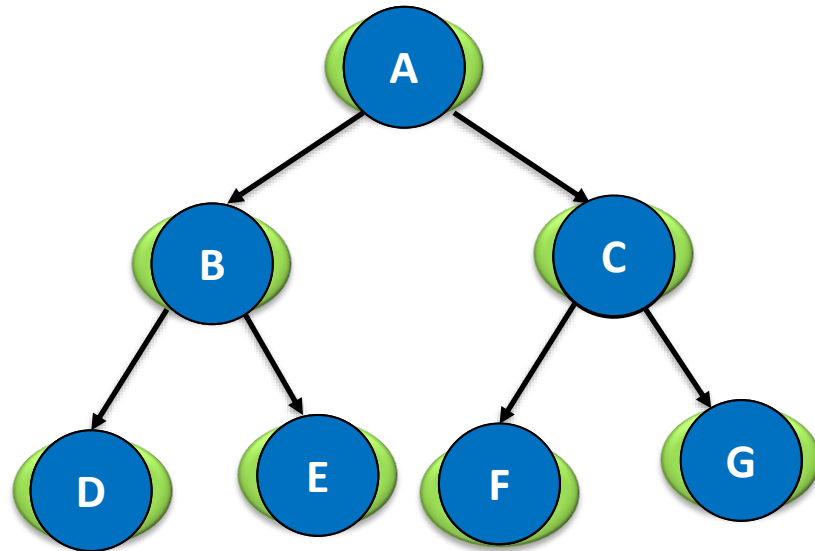
RECORRIDO EN PREORDEN

- Recorrido:
 - Padre.
 - Hijo izquierdo en preorden.
 - Hijo derecho en preorden.

A B D E C F G



RECORRIDO EN ORDEN



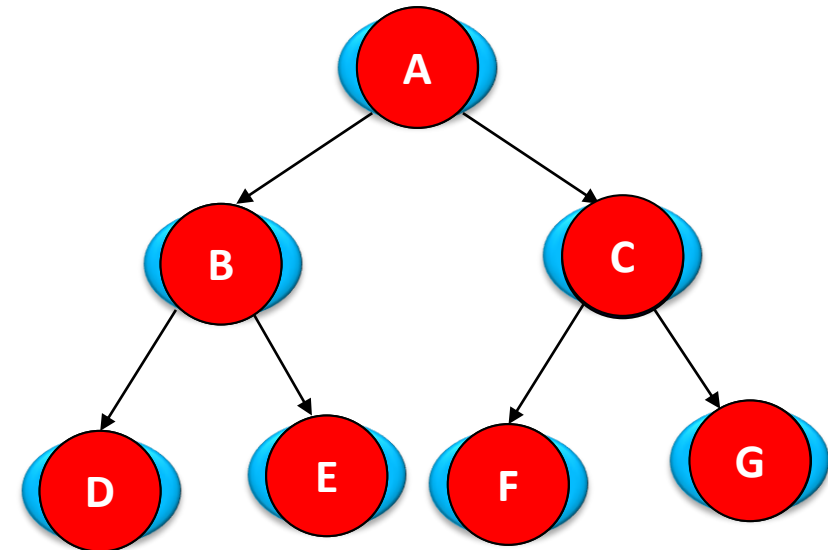
- Recorrido
 - Hijo izquierdo en orden.
 - Padre.
 - Hijo derecho en orden.

D B E A F C G

RECORRIDO EN POSTORDEN

- Recorrido
 - Hijo izquierdo en posorden.
 - Hijo derecho en posorden.
 - Padre.

D E B F G C A



Capítulo 8: Árboles

Bienvenid@s

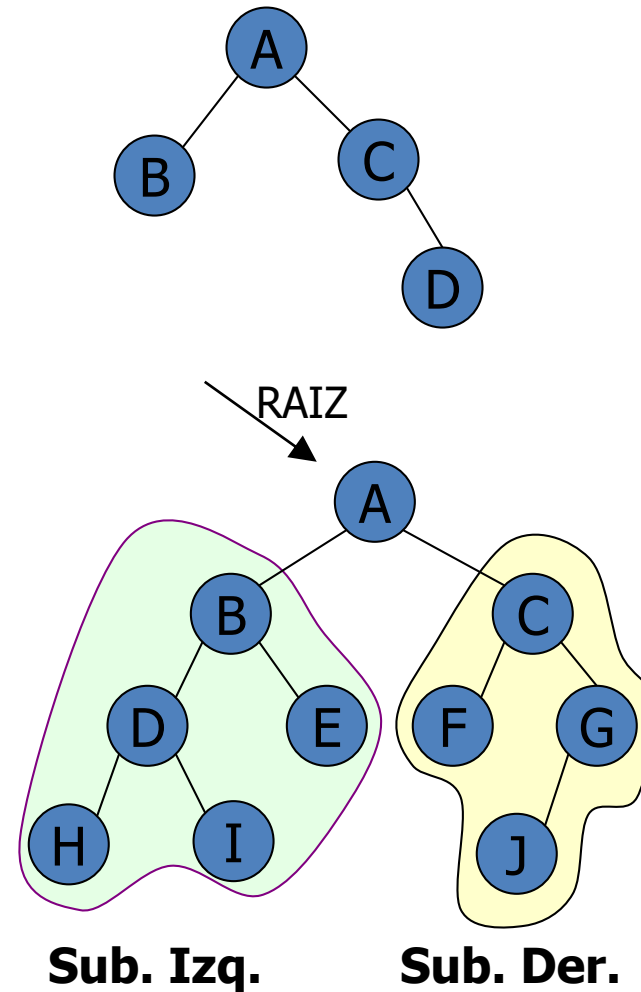
Este simulador esta diseñado para permitir al usuario, insertar buscar y eliminar elementos de un árbol binario

Iniciar

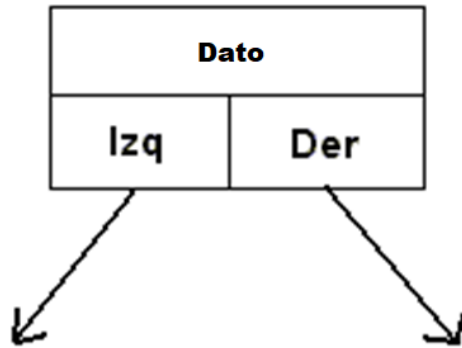
Instrucciones

ARBOLES BINARIOS

- Tipo especial de árbol
 - Cada nodo no puede tener mas de dos hijos
- Un árbol puede ser un conjunto
 - Vacío, no tiene ningún nodo
 - O constar de tres partes:
 - Un nodo raíz y
 - Dos subárboles binarios: izquierdo y derecho
- La definición de un árbol binario es recursiva
 - La definición global depende de si misma



ÁRBOLES BINARIOS – DEFINICIÓN DEL NODO



- Un nodo consta de 3 campos, un puntero al nodo izquierdo, la información (dato) y un puntero al nodo derecho.
- Por otro lado se implementa la clase Arbol con un atributo llamado raíz del tipo NodoABB

NodoABB		
f	dato	int
f	izq	NodoABB
f	der	NodoABB
m	NodoABB(int, NodoABB, NodoABB)	
m	NodoABB(int)	
m	getDato()	int
m	setDato(int)	void
m	getIzq()	NodoABB
m	setIzq(NodoABB)	void
m	getDer()	NodoABB
m	setDer(NodoABB)	void
m	toString()	String

```
public class NodoABB {  
    private int dato;  
    private NodoABB izq;  
    private NodoABB der;  
}
```

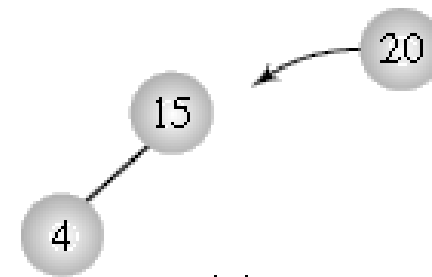
INSERTAR EN UN ÁRBOL



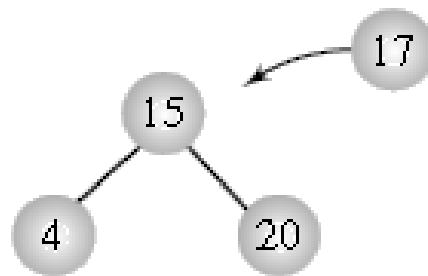
(a)



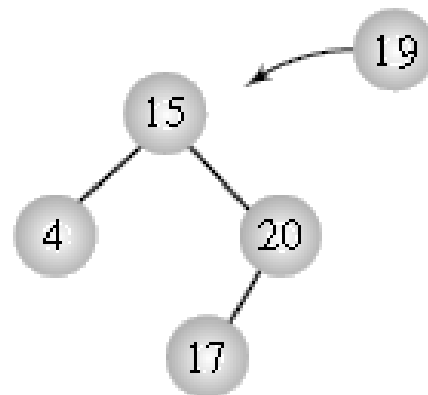
(b)



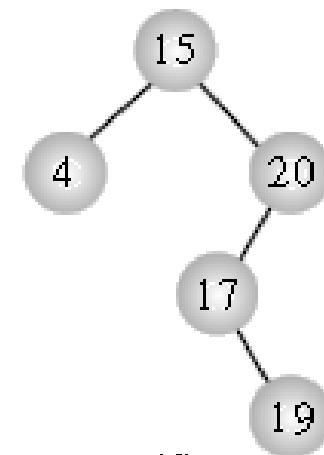
(c)



(d)



(e)



(f)



INSERTAR EN UN ÁRBOL

- Si el árbol no tiene elementos, raíz hace referencia al nodo nuevo
- De lo contrario con la ayuda de una nodo auxiliar y previo se busca donde insertar el nodo;
 - Se inicializa aux en raíz y previo en null
 - Mientras aux sea diferente de null
 - Se actualiza previo con aux y si el dato recibido es mayor que el de aux, se actualiza aux con la referencia derecha, de lo contrario se actualiza con la referencia izquierda.
- Si dato es mayor que el dato de previo en la referencia derecha de previo se guarda el nodo, de lo contrario se guarda en la referencia izquierda.
- Observe que si ingresa los mismos datos, pero en diferente orden, se generarán árboles diferentes.



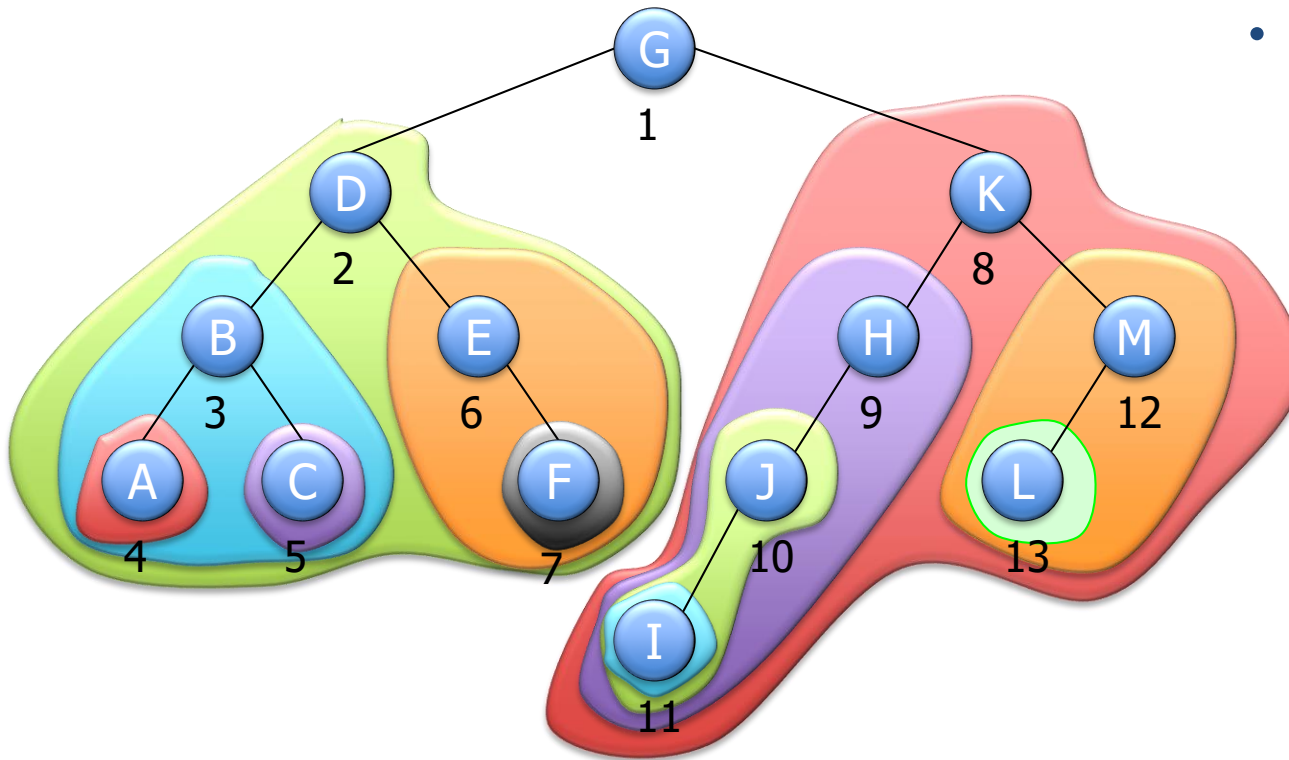
RECORRIDOS DE UN A.B.

- Recorrer es
 - Visitar todos los elementos de una estructura
- Como recorrer un árbol
 - Hay tantos caminos, cual escoger?
 - Existe tres recorridos típicos, nombrados de acuerdo a la posición de la raíz
 - Preorden: raíz - subarbol izq. - subarbol der.
 - Enorden : subarbol izq. - raíz - subarbol der.
 - Postorden : subarbol izq. - subarbol der. -raíz

EJEMPLO PREORDEN

- Para recorrer un árbol en preorden seguimos los siguientes pasos:

- 1. Visitar raíz
- 2. Preorden al Subarbol Izquierdo.
- 3. Preorden al Subarbol Derecho.

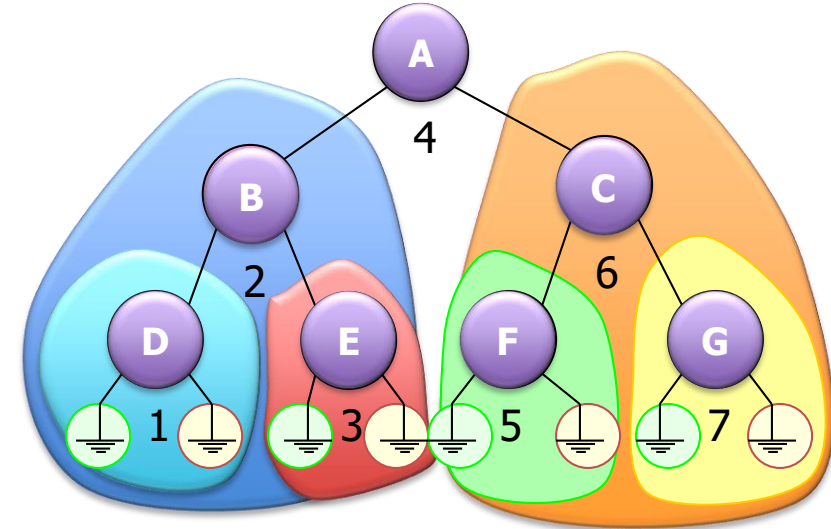


G-D-B-A-C-E-F-K-H-J-I-M-L

EJEMPLO ENORDEN

- 1. Enorden al Subarbol Izquierdo.
- 2. Visitar raíz
- 3. Enorden al Subarbol Derecho.

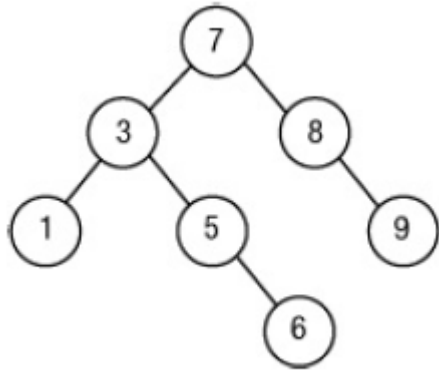
Arbol Vacio!, Terminar



POSORDEN



- 1. Posorden al Subarbol Izquierdo.
- 2. Posorden al Subarbol Derecho.
- 3. Visitar raíz



Postorden: 1, 6, 5, 3, 9, 8, 7

MOSTRAR ÁRBOL



- Por facilidad imprimiremos el árbol girado en 90°
- Se basa en el recorrido en orden, pero, a diferencia de este, primero se visita el nodo derecho
 - `verArbol(arbol->der, nro+1);`
- luego, el padre
 - `cout<<arbol->valor;`
- y finalmente el hijo izquierdo
 - `verArbol(arbol->izq, nro+1);`
- El argumento `nro+1` indica el nivel de anidamiento y es utilizado para imprimir espacios en blanco y darle un aspecto que aparente ser un árbol.