



# **UPN, PASIÓN POR TRANSFORMAR VIDAS**

**UNIDAD 2:  
ÁRBOLES Y GRAFOS**

**SESIÓN 8:  
ÁRBOLES BINARIOS**

**Operaciones: Raíz, hoja, tallo.**

Dr. Eric Gustavo Coronel Castillo  
[eric.coronel@upn.pe](mailto:eric.coronel@upn.pe)



## DESAFIO DEL DIA

"Un buen programador sabe que el código eficiente nace de buena organización."

# MOTIVACIÓN



IDEA FUERZA

Organización y eficiencia van de la mano.

# LOGRO DE LA UNIDAD 1



Al finalizar la unidad, el estudiante implementa algoritmos utilizando árboles y grafos, como resultado del análisis de casos utilizando el lenguaje C# con entorno gráfico, demostrando lógica y habilidad en la implementación de los algoritmos para su proyecto final..

# LOGRO DE LA SESIÓN



Al término de la sesión, el estudiante aprende algoritmos con arboles, arboles binarios y diversas aplicaciones, usándolos con coherencia.

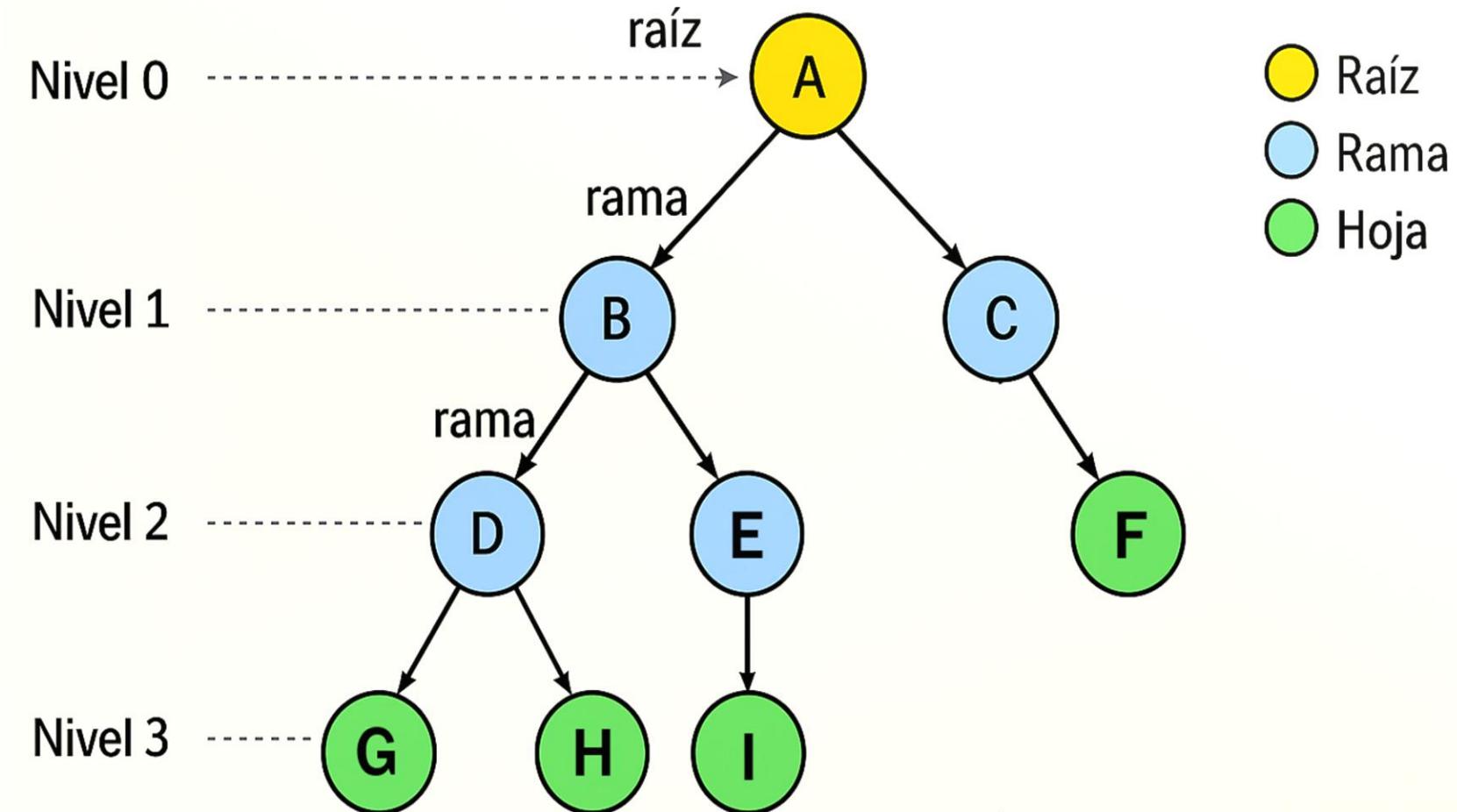
- Árboles: Generalidades.
- Arboles binarios.
- Operaciones: Raíz, hoja, tallo, recorrido inorden, postorden, preorden.

# REFLEXIONA



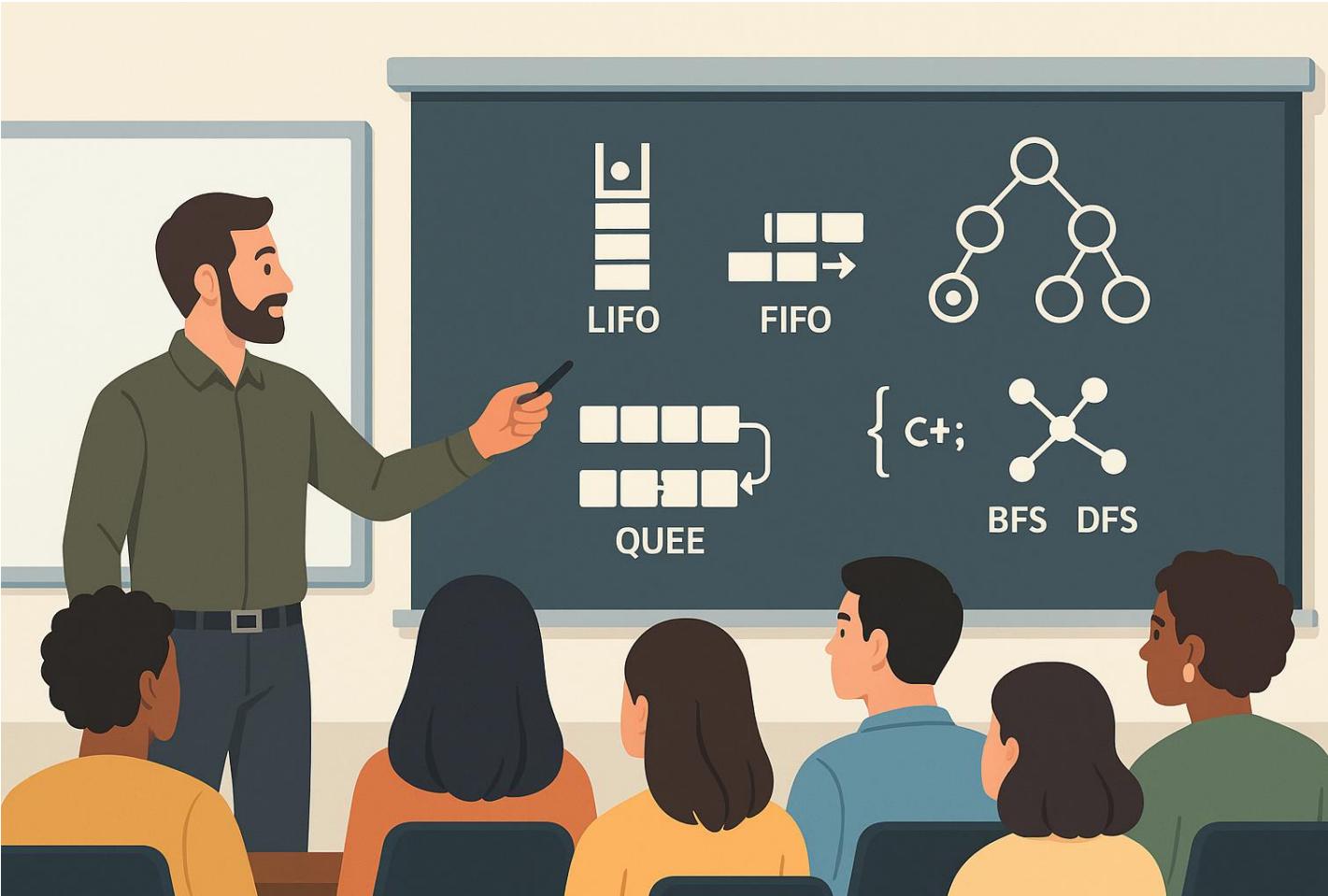
1. ¿Qué es árbol?
2. ¿Qué es un árbol binario?

# REFLEXIONA





# DESARROLLO



# INTRODUCCION



- Las listas enlazadas son estructuras lineales
  - Son flexibles pero son secuenciales, un elemento detrás de otro
- Los árboles
  - Junto con los grafos son estructuras de datos no lineales
  - Superan las desventajas de las listas
  - Sus elementos se pueden recorrer de distintas formas, no necesariamente uno detrás de otro
- Son muy útiles para la búsqueda y recuperación de información

# ÁRBOLES



- ❖ Un **árbol** es una **estructura de datos jerárquica** utilizada en programación para organizar información en forma de niveles. Es como un esquema de familia o de organización, donde cada elemento (llamado **nodo**) puede tener varios "hijos", pero solo un "padre", excepto el nodo principal, que se llama **raíz**.

# ÁRBOLES

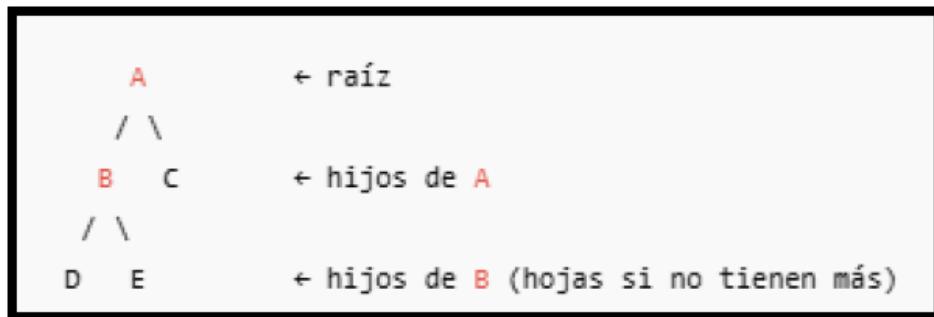


- ❖ Los árboles son considerados las estructuras de datos no lineales y dinámicas de datos muy importantes del área de computación.
- ❖ Los árboles son muy utilizados en informática como un método eficiente para búsquedas grandes y complejas.
- ❖ Casi todos los sistemas operativos almacenan sus archivos en árboles o estructuras similares a árboles

# ÁRBOLES

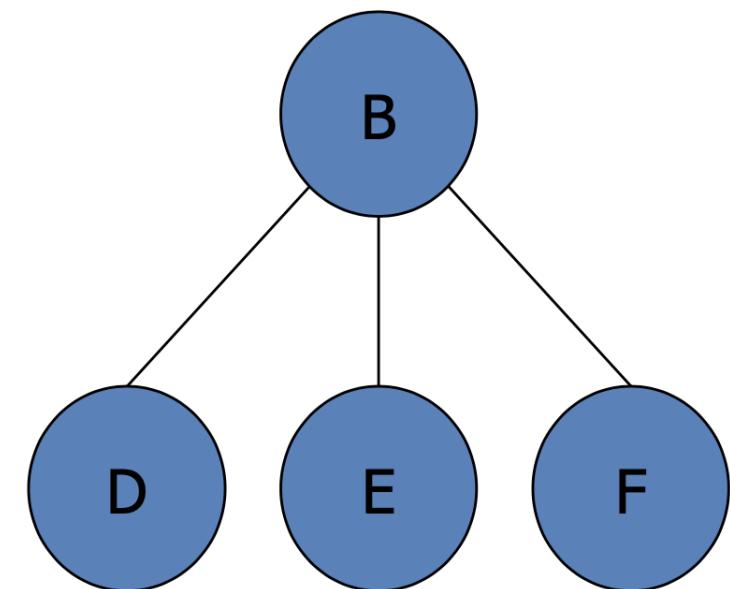


Nombre	Descripción
Raíz	El primer nodo del árbol.
Nodo	Cada elemento del árbol que contiene un valor.
Hijo	Nodo que depende de otro nodo (su "padre").
Padre	Nodo que tiene al menos un hijo.
Hoja	Nodo que no tiene hijos.
Subárbol	Una parte del árbol que también es un árbol por sí sola.
Altura	Longitud del camino más largo desde la raíz hasta una hoja.

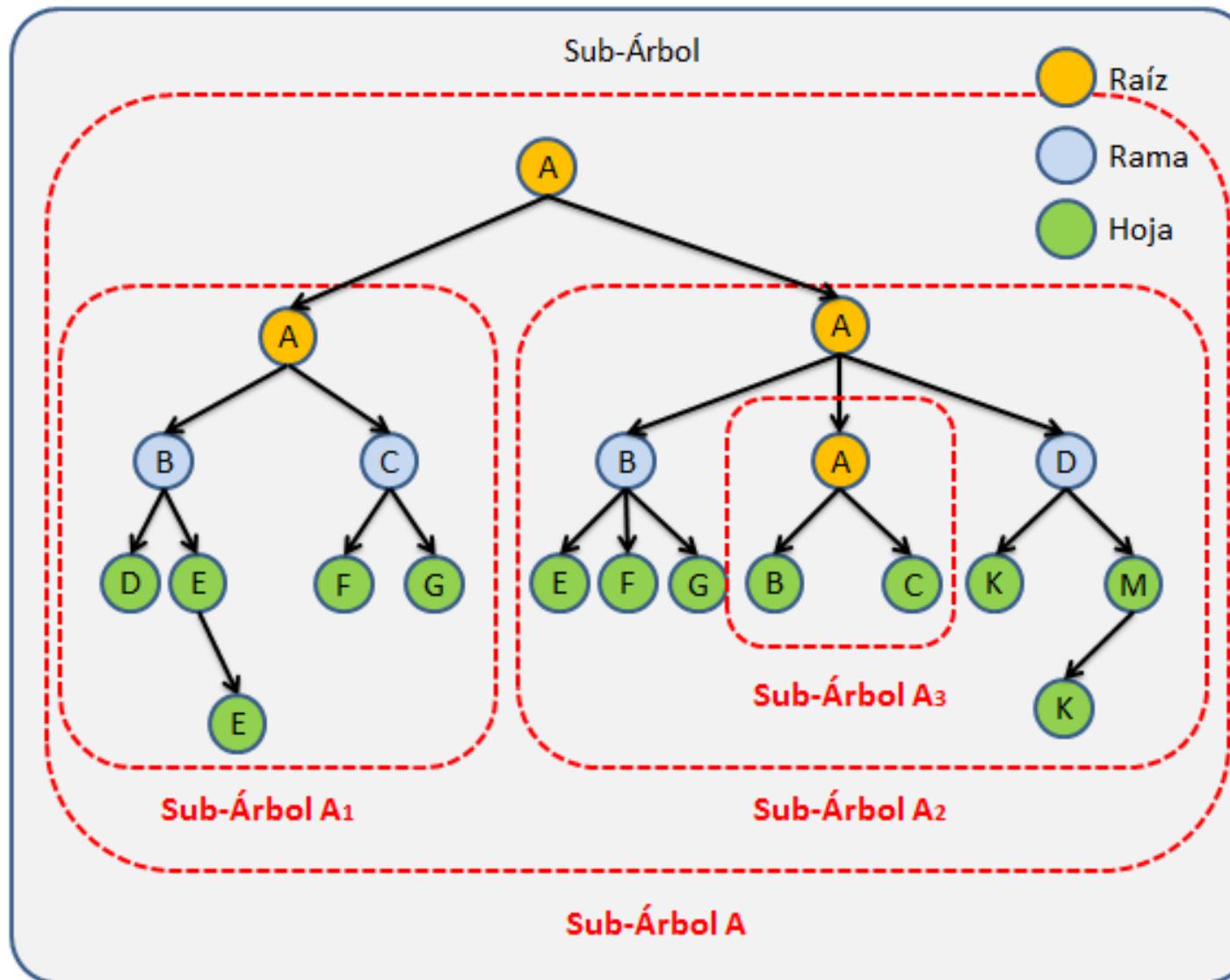


# CONCEPTO

- Estructura que organiza sus elementos formando jerarquías: PADRES E HIJOS
- Los elementos de un árbol se llaman nodos Si un nodo p tiene un enlace con un nodo m,
  - P es el padre y m es el hijo
  - Los hijos de un mismo padre se llaman: hermanos
- Todos los nodos tienen al menos un parente, menos la raíz: A Si no tienen hijos se llaman hoja: D, E, F y C
- Un subárbol de un árbol
  - Es cualquier nodo del árbol junto con todos sus descendientes



# TERMINOLOGIA



# TERMINOLOGIA

- Camino: Secuencia de nodos conectados dentro de un arbol
- Longitud del camino: es el numero de nodos menos 1 en un camino
- Altura del árbol: es el nivel mas alto del árbol
  - Un árbol con un solo nodo tiene altura 1
- Nivel(profundidad) de un nodo: es el numero de nodos entre el nodo y la raíz.

# TERMINOLOGIA

- Nivel de un árbol
  - Es el numero de nodos entre la raíz y el nodo mas profundo del árbol, la altura del un árbol entonces
- Grado(aridad) de un nodo: es numero de hijos del nodo
- Grado(aridad) de un árbol: máxima aridad de sus nodos

# CARACTERÍSTICAS 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 parente en común dentro de la estructura.

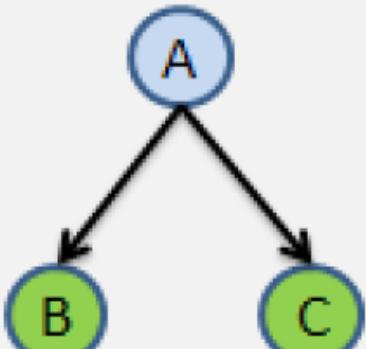
**Nodo Hoja:** Son todos aquellos nodos que no tienen hijos, los cuales siempre se encuentran en los extremos de la estructura.

**Nodo Rama:** Estos son todos aquellos nodos que no son la raíz y que además tiene al menos un hijo.

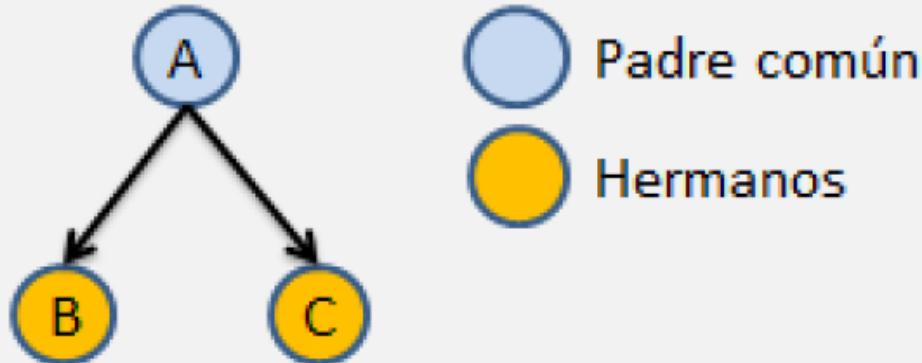
# DATOS IMPORTANTES DE LOS ÁRBOLES



Nodos Padre e Hijos



Nodos Padre e Hijos



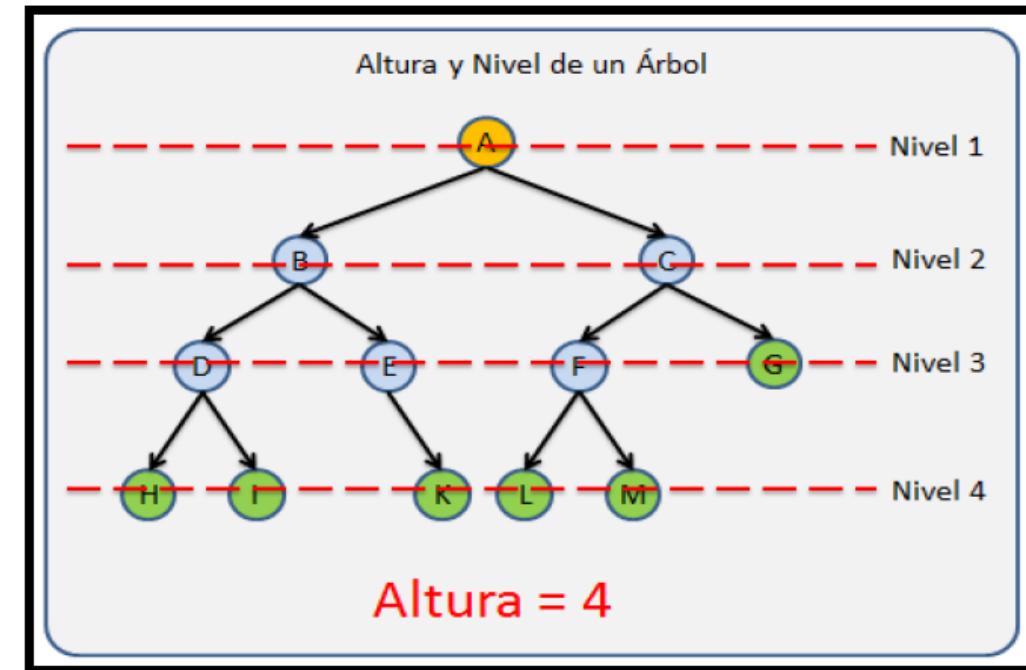
La siguiente imagen muestra de forma gráfica los nodos Padre, Hijo y Hermanos.

# DATOS IMPORTANTES DE LOS ÁRBOLES

11

**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 mas. 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 se 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.



En la imagen se muestran los Niveles y la Altura de un Árbol

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

# DATOS IMPORTANTES DE LOS ÁRBOLES

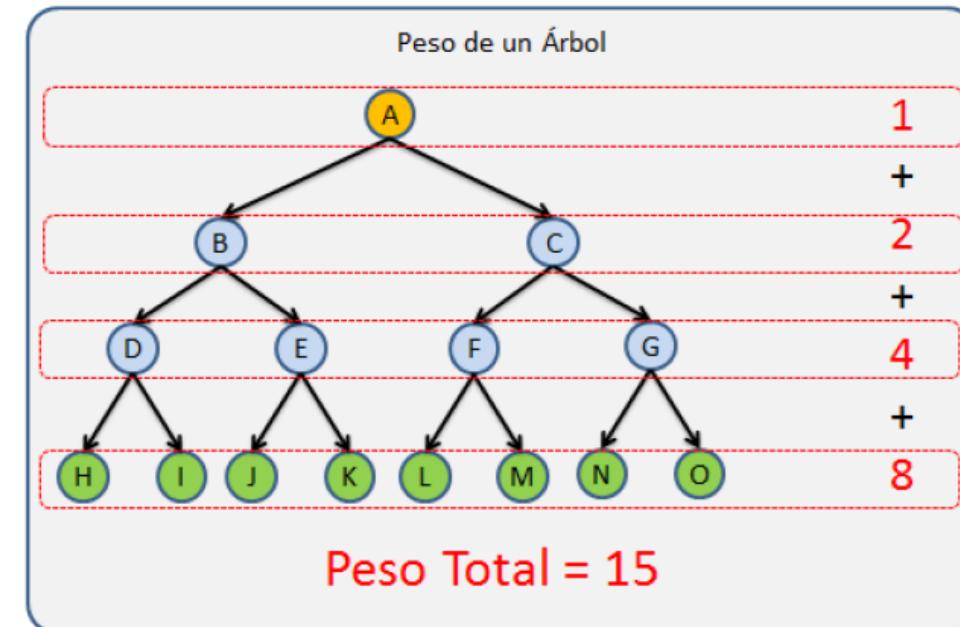


La **altura** es calculado mediante recursividad tomando el nivel mas grande de los dos sub-árboles de forma recursiva de la siguiente manera:

$$\text{altura} = \max(\text{altura}(\text{hijo1}), \text{altura}(\text{hijo2}), \text{altura}(\text{hijoN})) + 1$$

**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.)

Peso. Es el número de nodos que tiene un árbol



La imagen nos muestra como se calcula el peso de un Árbol, el cual es la suma de todos sus nodos, sin importar el orden en que sean contados.

# ARBOLES BINARIOS

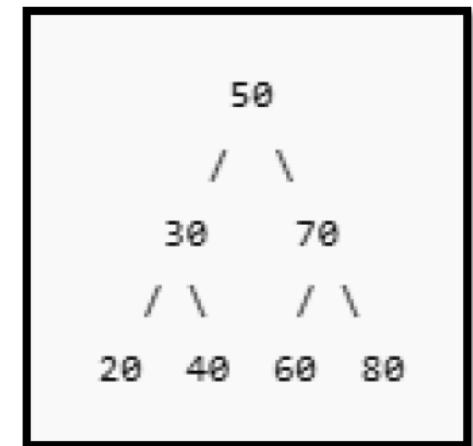


❖ Un **árbol binario** es un tipo especial de estructura de datos en forma de árbol donde **cada nodo tiene como máximo dos hijos**:

- Un **hijo izquierdo**
- Un **hijo derecho**

❖ Cada nodo del árbol binario contiene:

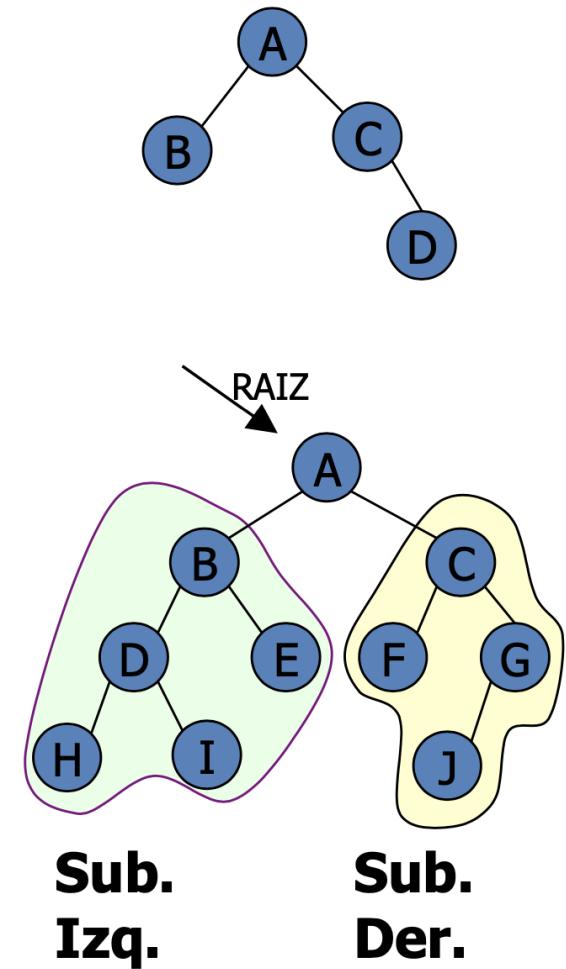
- Un valor o dato** (por ejemplo, un empleado)
- Una referencia al hijo izquierdo** (puede ser nula si no existe)
- Una referencia al hijo derecho** (puede ser nula si no existe)



# ARBOLES BINARIOS



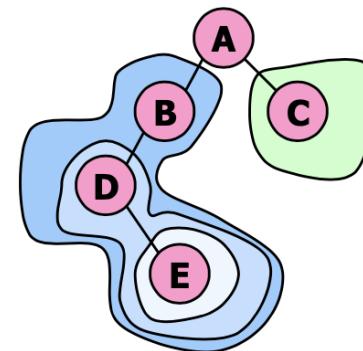
- Tipo especial de árbol
  - Cada nodo no puede tener mas de dos hijos
- Un árbol puede ser un conjunto
  - Vacío, no tiene ningún nodo
  - O constar de tres partes:
    - Un nodo raíz y
    - Dos subárboles binarios: izquierdo y derecho
- La definición de un árbol binario es recursiva
  - La definición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



# DEFINICIONES RECURSIVAS

- La definición del árbol es recursiva
  - Se basa en si misma
- La terminología de los árboles
  - También puede ser definida en forma recursiva
- Ejemplo: NIVEL de un árbol
  - Identificar el caso recursivo y el caso mas básico

SUB. DER.  
Nivel 1



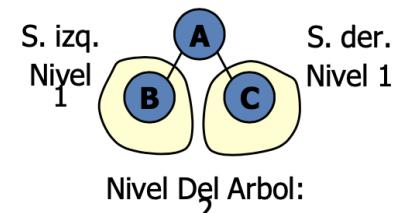
## Caso Básico

Un árbol con un solo nodo tiene nivel

## Caso Recursivo

Si tiene mas de un nodo, el nivel es:  
 $1 + \text{MAX}(\text{Nivel(SubIzq)}, \text{Nivel(SubDer)})$

nivel 1 → A

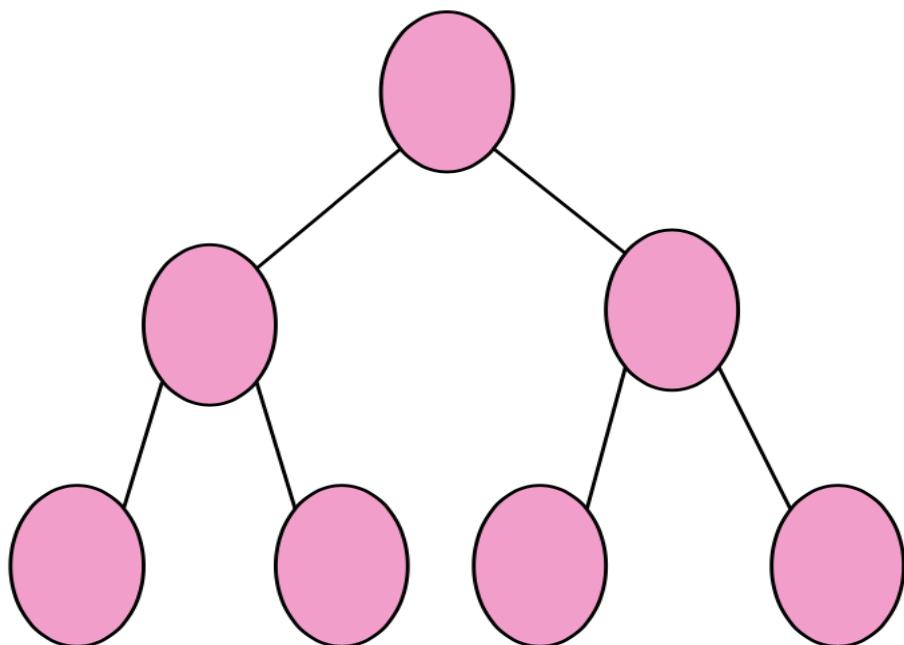


SUB. IZQ.  
Nivel = 3

NIVEL : 4

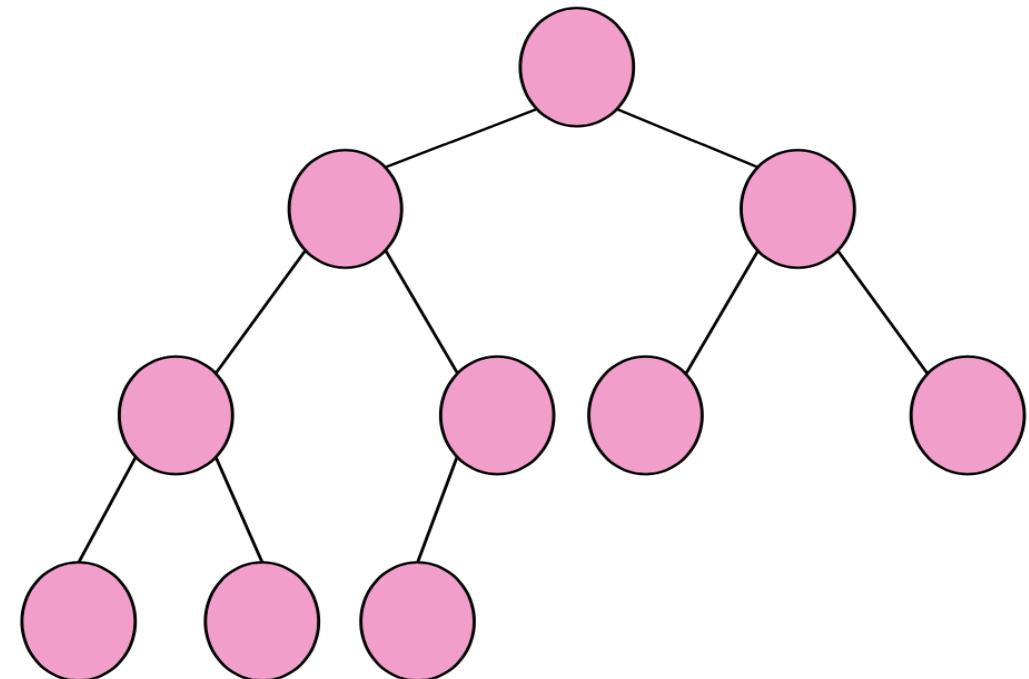
# ARBOLES BINARIOS LLENOS

- Un árbol de altura  $h$ , esta lleno si
  - Todas sus hojas esta en el nivel  $h$
  - Los nodos de altura menor a  $h$  tienen siempre 2 hijos
- Recursiva
  - Si  $T$  esta vacío,
    - Entonces  $T$  es un árbol binario lleno de altura
  - Si no esta vacío, y tiene  $h>0$ 
    - Esta lleno si los subárboles de la raíz, son ambos árboles binarios llenos de altura  $h-1$



# ARBOLES BINARIOS COMPLETOS

- Un arbol de altura  $h$  esta completo si
  - Todos los nodos hasta el nivel  $h-2$  tienen dos hijos cada uno y
  - En el nivel  $h-1$ , si un nodo tiene un hijo derecho, todas las hojas de su subarbol izquierdo estan a nivel  $h$
- Si un arbol esta lleno, tambien esta completo

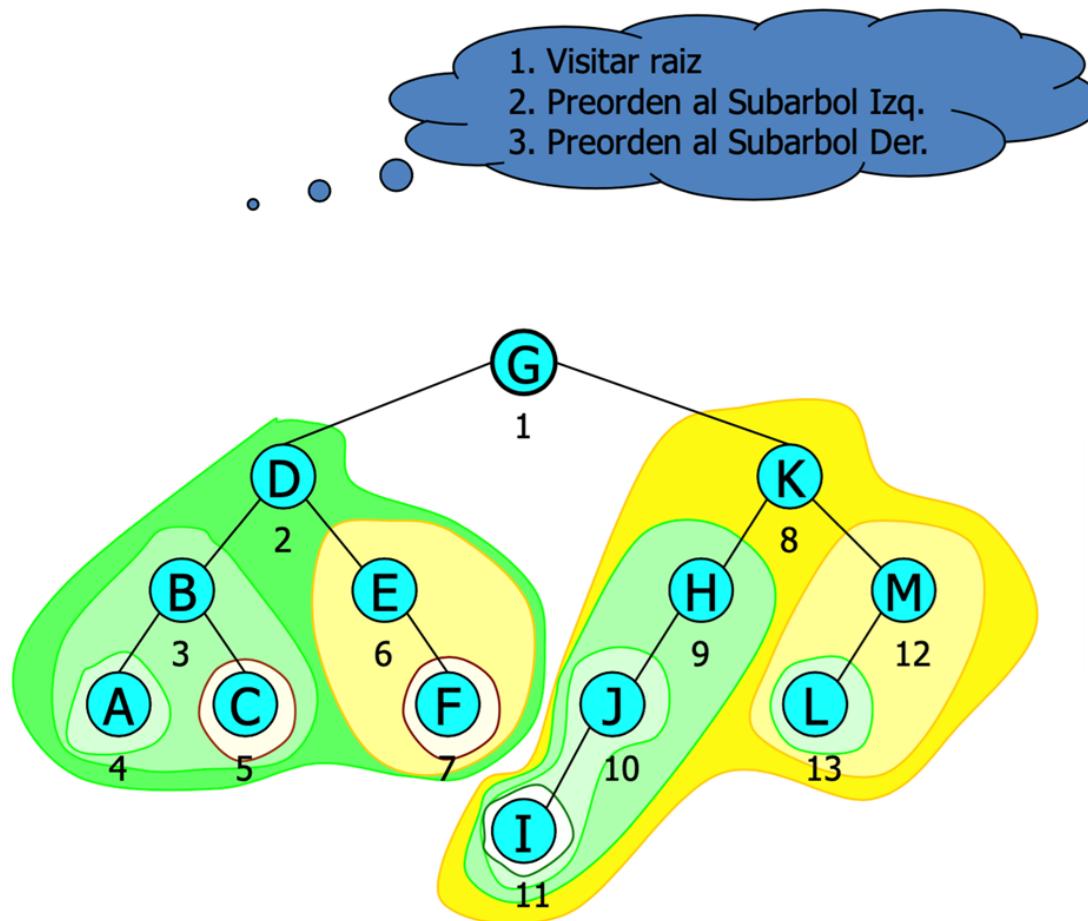


# OTROS



- Un árbol equilibrado es cuando
  - La diferencia de altura entre los subárboles de cualquier nodo es máximo 1
- Un árbol binario equilibrado totalmente
  - Los subárboles izquierdo y derecho de cada nodo tienen la misma altura: es un árbol lleno
- Un árbol completo es equilibrado
- Un árbol lleno es totalmente equilibrado

# EJEMPLO PREORDEN



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

# AB Y NODOAB: DECLARACION

- Un árbol binario: conjunto de nodos
  - Solo se necesita conocer el nodo raíz
- Cada nodo
  - Tiene Contenido y
  - Dos enlaces: árbol hijo izquierdo, árbol hijo derecho
- Un nodo hoja, es aquel cuyos dos enlaces apunta a null
  - Un nodo en un árbol tiene mas punteros a null que un nodo de una lista
- De un árbol solo se necesita conocer su raíz
  - La raíz, que es un nodo, puede definir al árbol o

# AB: OPERACIONES

- Crear y Eliminar
  - AB\_Vaciar(AB \*A);
  - AB\_Eliminar(AB \*A);
- Estado del Arbol
  - bool AB\_EstaVacio(AB A);
- Añadir y remover nodos
  - void AB\_InsertarNodo(AB \*A, NodoAB \*nuevo)
  - NodoAB \*AB\_SacarNodoxContenido(AB \*A, Generico G, Generico\_fnComparar fn);
  - NodoAB \* AB\_SacarNodoxPos(AB \*A, NodoAB \*pos);

# OPERACION EN ORDEN



```
void AB_EnOrden(AB A, Generico_fnImprimir imprimir){  
    if(!AB_EstaVacio(A)){  
        AB_EnOrden(A->izq,imprimir);  
        imprimir(A->G);  
        AB_EnOrden(A->der,imprimir);  
    }  
}
```

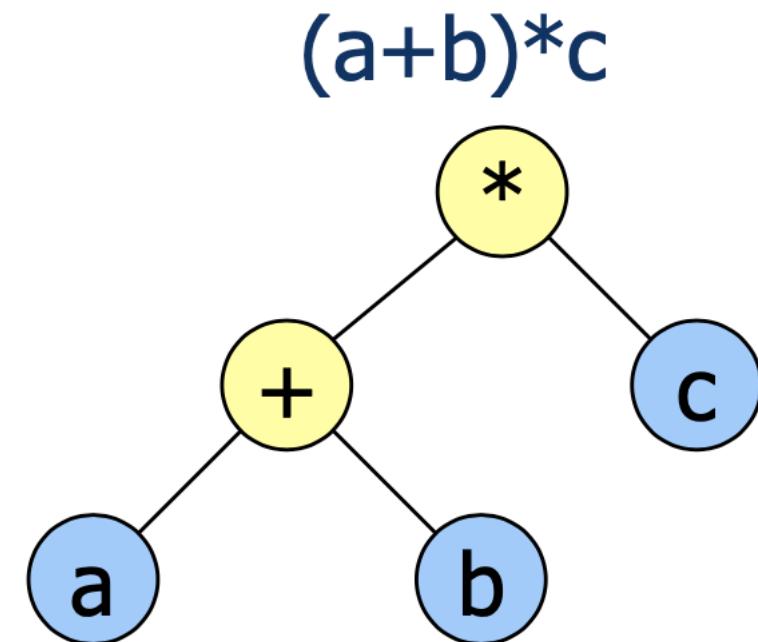
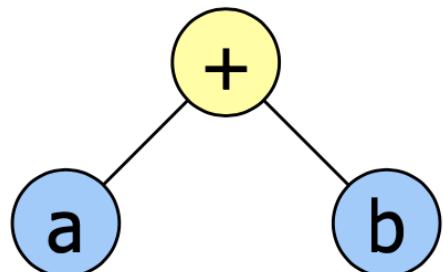
# APLICACIÓN: EVALUACION DE EXPRESIONES



- Ya sabemos lo de las expresiones, cierto?
  - InFija, operador en medio
  - PreFija, operador antes de dos operandos
  - PosFija, operador luego de dos operandos
- Para evaluar una expresion dada, podriamos
  - Pasarla a posfija y usar solo pilas
  - Pasarla a posfija y usar pilas y un arbol de expresion

# ARBOL DE EXPRESION

- Arboles que representan expresiones en memoria
  - Todos los operadores tienen dos operandos
  - La raiz puede contener el operador
  - Hijo izq: operando 1, Hijo derecho: operando 2
  - Ejemplo:  $(a+b)$

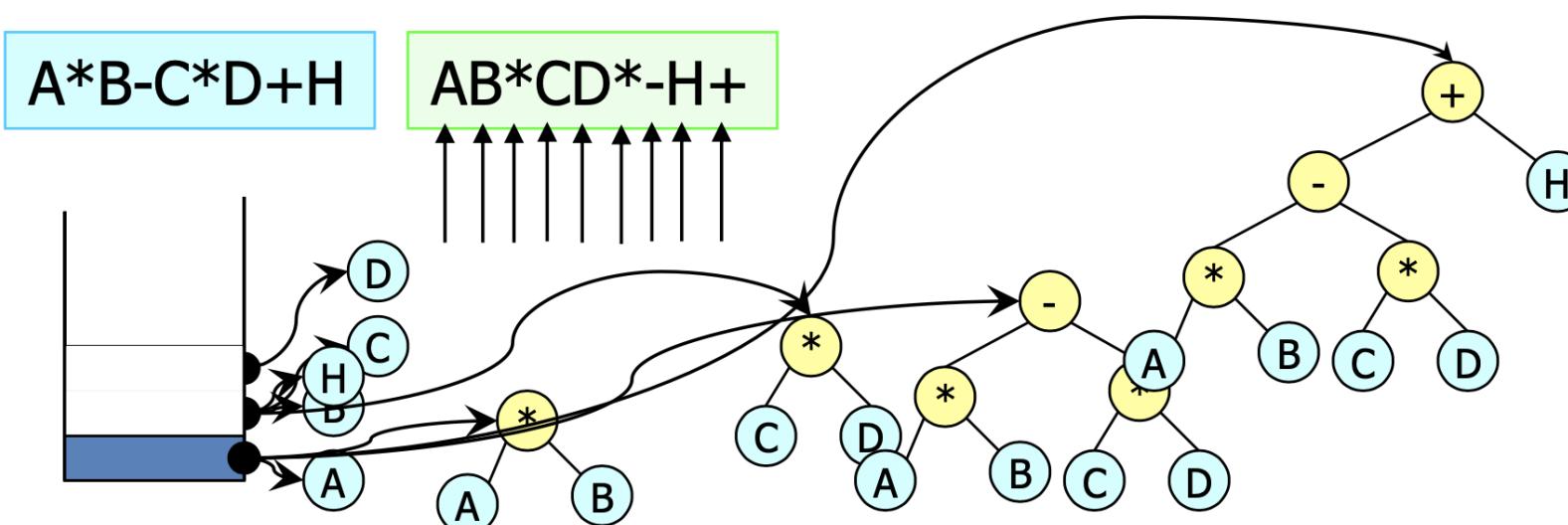


# EVALUAR UNA EXPRESION ARITMETICA EN INFIXA

- La expresion se transforma a la expresion posfija
  - Esto, ya sabemos como hacer
- Crear un arbol de expresion
  - Para esto se va a usar una pila y un arbol de caracteres
- Usando el arbol, evaluar la expresion

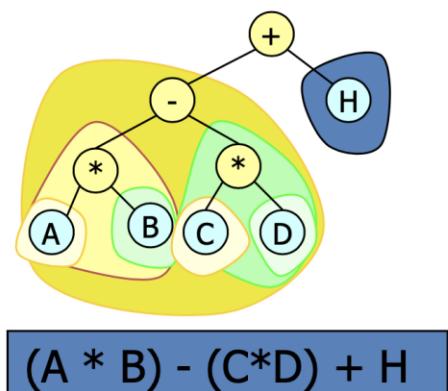
# CREAR UN ARBOL DE EXPRESION

- Los operandos seran siempre nodos hoja del arbol
  - Al revisar un operando, creo una nueva hoja y la recuerdo
- Los operadores seran nodos padre
  - Al revisar un operador, recuerdo las dos ultimas hojas creadas y uno todo
  - No debo olvidar el nuevo arbolito que he creado



# EVALUACION DE LA EXP. POSTFIJA

- Lo ideal es recuperar los dos operandos, el operador, y ejecutar la opcion
  - Que recorrido es el ideal?
  - PostOrden



Para evaluar el arbol:

**Si el arbol tiene un solo nodo** y este almacena un operando

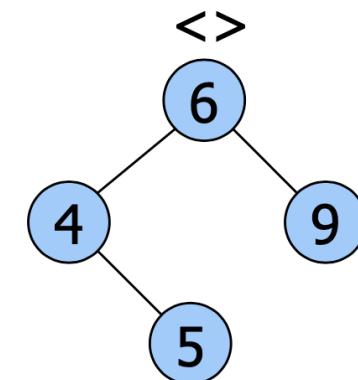
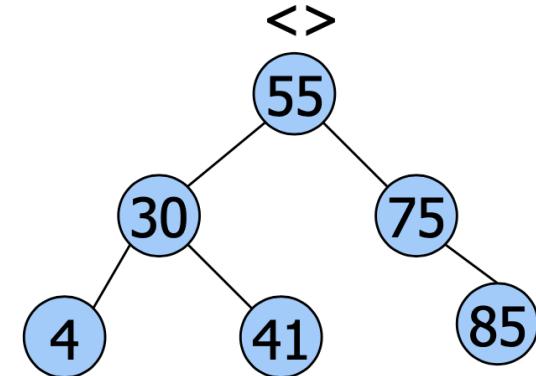
El resultado de la evaluacion es el valor de ese operando

**Si no**

1. Res1 = Evaluo subarbol izquierdo
2. Res2 = Evaluo subarbol derecho
3. Recupero la info de la raiz y efectuo la operación allí indicada, entre Res1 y Res2

# ARBOL BINARIO DE BUSQUEDA

- Los elementos en un arbol
  - Hasta ahora no han guardado un orden
  - No sirven para buscar elementos
- Los arboles de busqueda
  - Permiten ejecutar en ellos busqueda binaria
  - Dado un nodo:
    - Todos los nodos del sub. Izq. Tienen una clave menor que la clave de la raiz
    - Todos los nodos del sub. Der. Tienen una clave mayor que la clave de la raiz

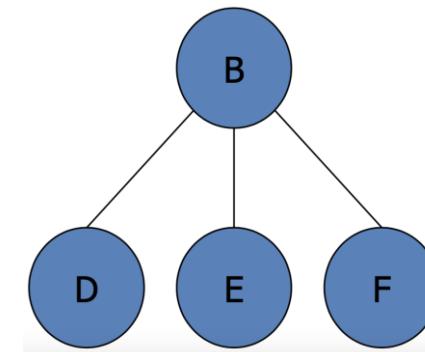
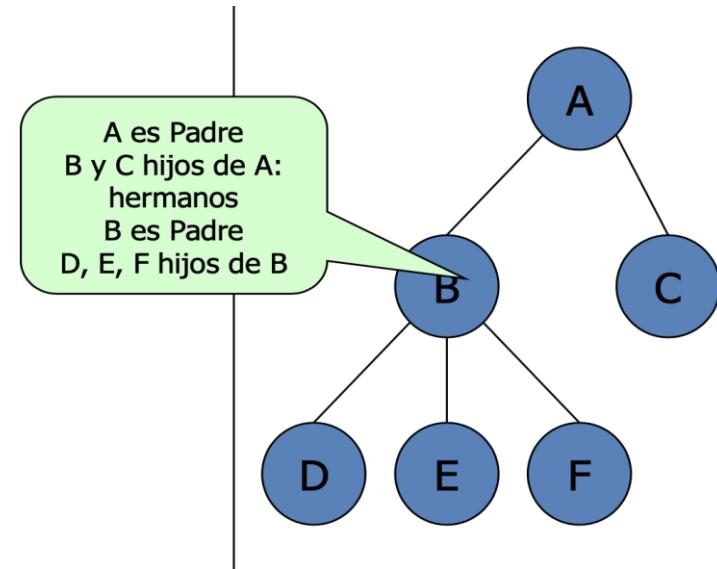




# ARBOLES BINARIOS DE BÚSQUEDA ABB

# CONCEPTO

- Estructura que organiza sus elementos formando jerarquías: PADRES E HIJOS
  - Los elementos de un árbol se llaman nodos
  - Si un nodo p tiene un enlace con un nodo m,
    - p es el padre y m es el hijo
    - Los hijos de un mismo padre se llaman: hermanos
- Todos los nodos tienen al menos un parente, menos la raíz: A
- Si no tienen hijos se llaman hoja: D, E, F y C
- Un subárbol de un árbol
  - Es cualquier nodo del árbol junto con todos sus descendientes



# TERMINOLOGIA

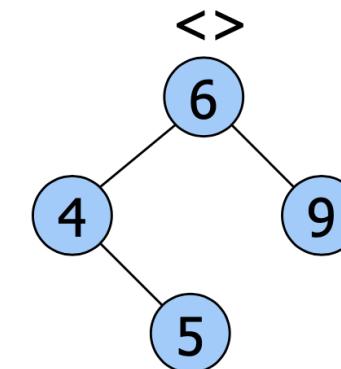
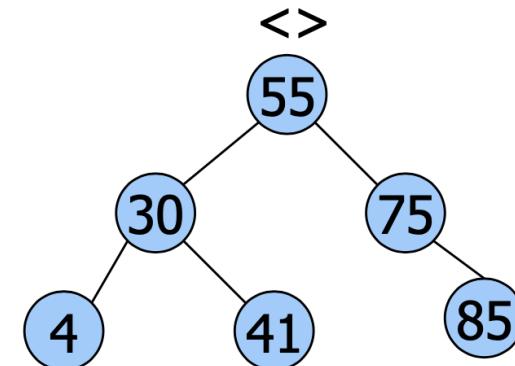
- Camino: Secuencia de nodos conectados dentro de un árbol
- Longitud del camino: es el numero de nodos menos 1 en un camino
- Altura del árbol: es el nivel mas alto del árbol
  - Un árbol con un solo nodo tiene altura 1
- Nivel(profundidad) de un nodo: es el numero de nodos entre el nodo y la raíz.

# TERMINOLOGIA

- Nivel de un árbol
  - Es el numero de nodos entre la raíz y el nodo mas profundo del árbol, la altura del un árbol entonces
- Grado(aridad) de un nodo: es numero de hijos del nodo
- Grado(aridad) de un árbol: máxima aridad de sus nodos

# ARBOL BINARIO DE BUSQUEDA

- Los elementos en un árbol
  - Hasta ahora no han guardado un orden
  - No sirven para buscar elementos
- Los árboles de búsqueda
  - Permiten ejecutar en ellos búsqueda binaria
  - Dado un nodo:
    - Todos los nodos del sub. Izq. Tienen una clave menor que la clave de la raíz
    - Todos los nodos del sub. Der. Tienen una clave mayor que la clave de la raíz



# OPERACIONES CON UN ABB

- Inserción
- Búsqueda
- Recorridos

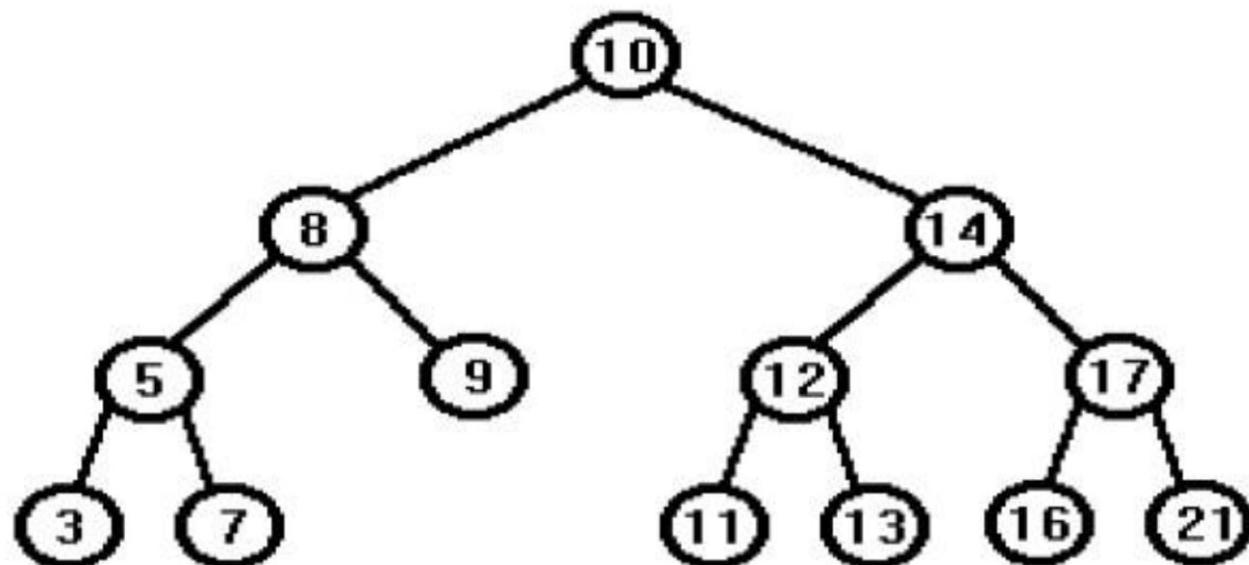
# INSERCIÓN



- Insertar el elemento X
- Si el árbol está vacío, X será la raíz del árbol
- Si no está vacío, se compara el valor de X con el del nodo padre:
  - Si es **menor** se inserta como un *hijo izquierdo*
  - Si es **mayor** se inserta como *hijo derecho*

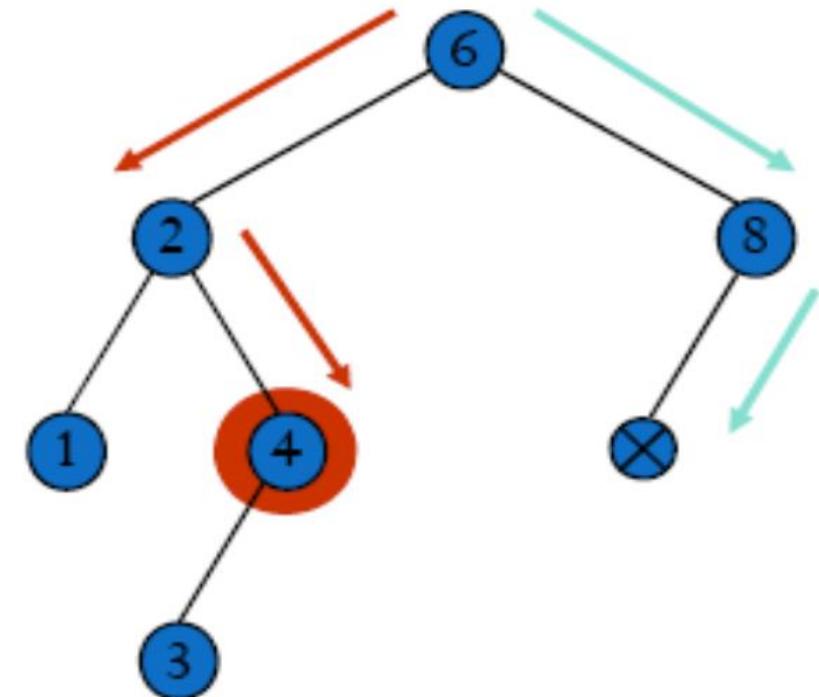
# INSERCIÓN, EJEMPLO

- Crear un ABB insertando los siguientes elementos: 10, 8, 14, 12, 9, 17, 5, 7, 11, 16, 13, 3, 21



# BÚSQUEDA

- Para buscar al elemento X:
  - Si X es la llave de la raíz de un ABB, se ha terminado
  - Si X es **menor** que el padre, *se busca a la izquierda*
  - Si X es **mayor** que el padre, *se busca a la derecha*

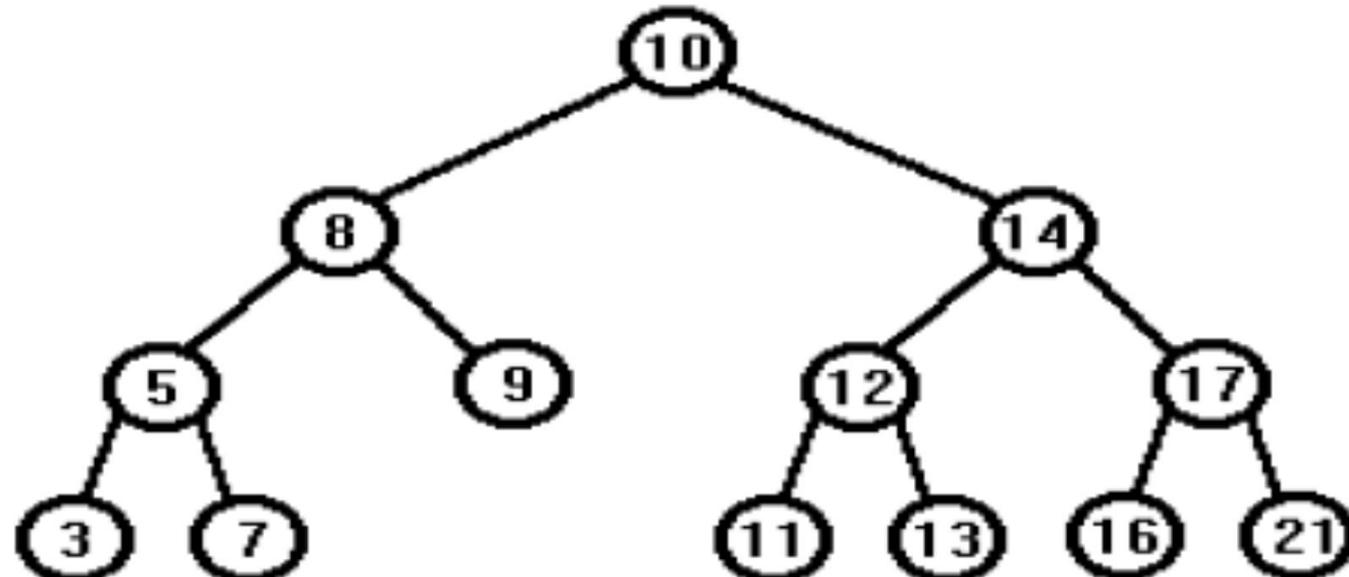


Buscando 4: VERDADERO

Buscando 7: FALSO

# BÚSQUEDA, EJEMPLO

- Si se busca el elemento “7”:
- Se compara 7 con 10 y se desciende por la izquierda
- Se compara 7 con 8 y se desciende por la izquierda
- Se compara 7 con 5 y se desciende por la derecha
- Se encuentra la llave deseada





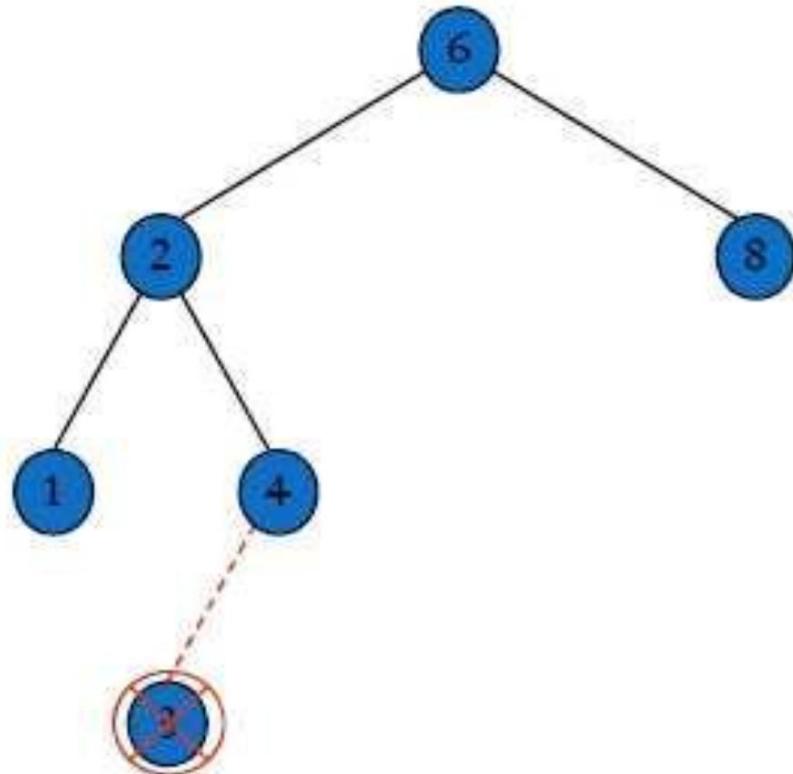
# ELIMINACIÓN

- Existen cuatro distintos escenarios:
  1. Intentar eliminar un nodo que no existe.  
No se hace nada, simplemente se regresa FALSE.
  2. Eliminar un nodo hoja.  
Caso sencillo, se borra el nodo y se actualiza el apuntador del nodo padre a NULL
  3. Eliminar un nodo con un solo hijo  
Caso sencillo, el nodo padre del nodo a borrar se convierte en el padre del único nodo hijo
  4. Eliminar un nodo con dos hijos  
Caso complejo, es necesario mover más de un apuntador

# ELIMINAR (casos sencillos)

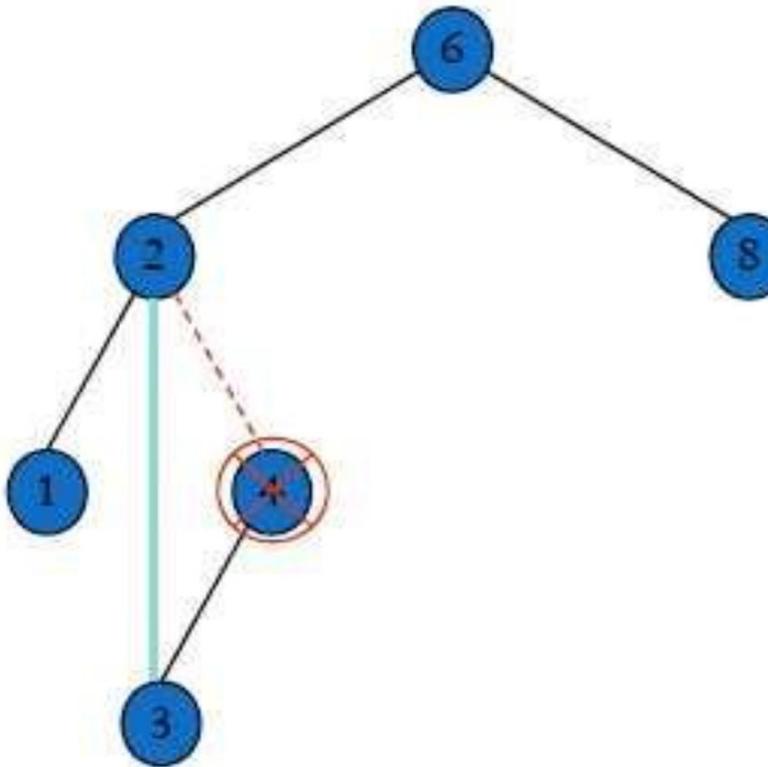
## Eliminar nodo hoja

*Eliminar 3*



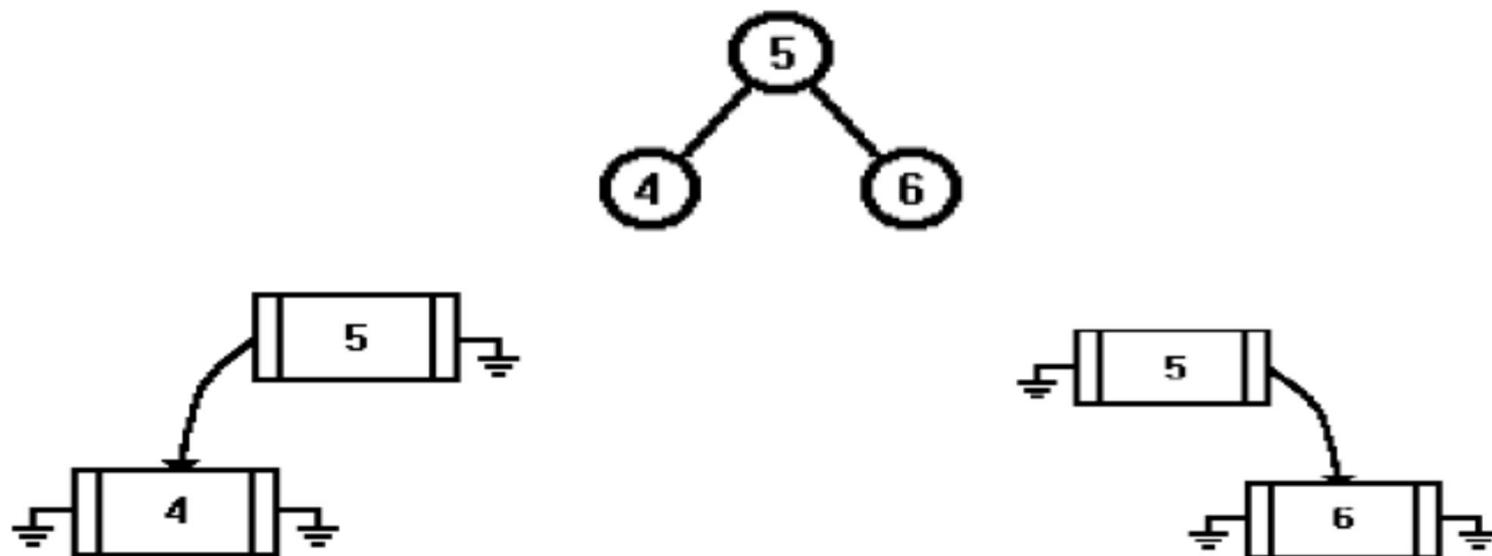
## Eliminar nodo con un hijo

*Eliminar 4*



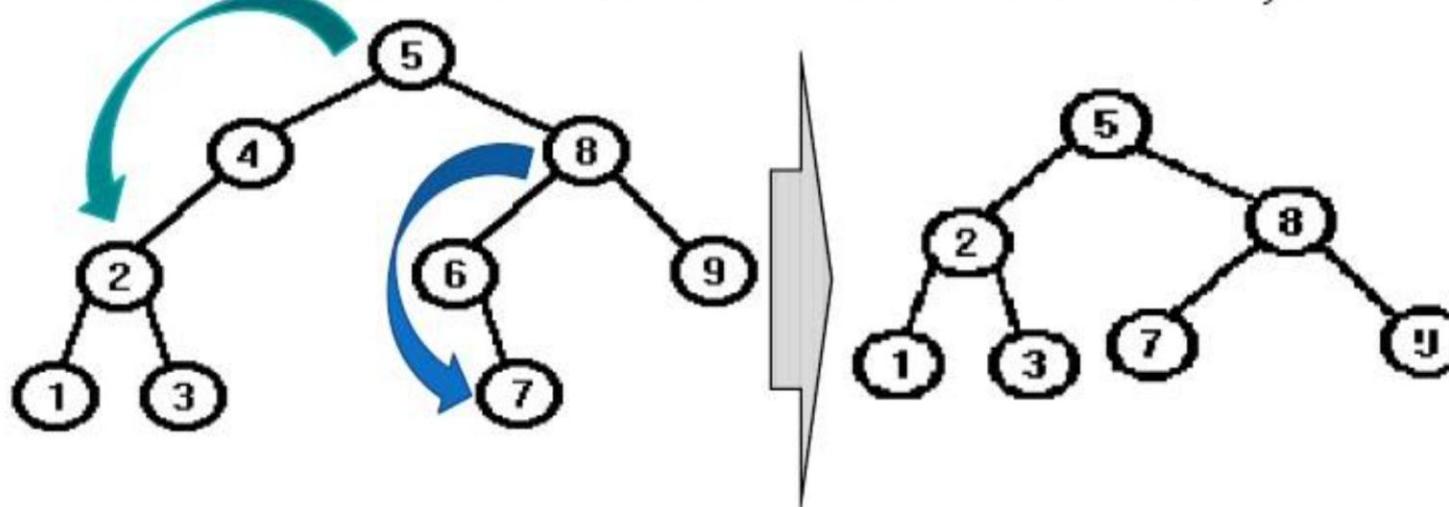
# ELIMINACIÓN DE UNA HOJA

- Caso mas sencillo, en donde lo único que hay que hacer es que la liga que lo referencia debe ser *nil*



# CON UN HIJO ÚNICO

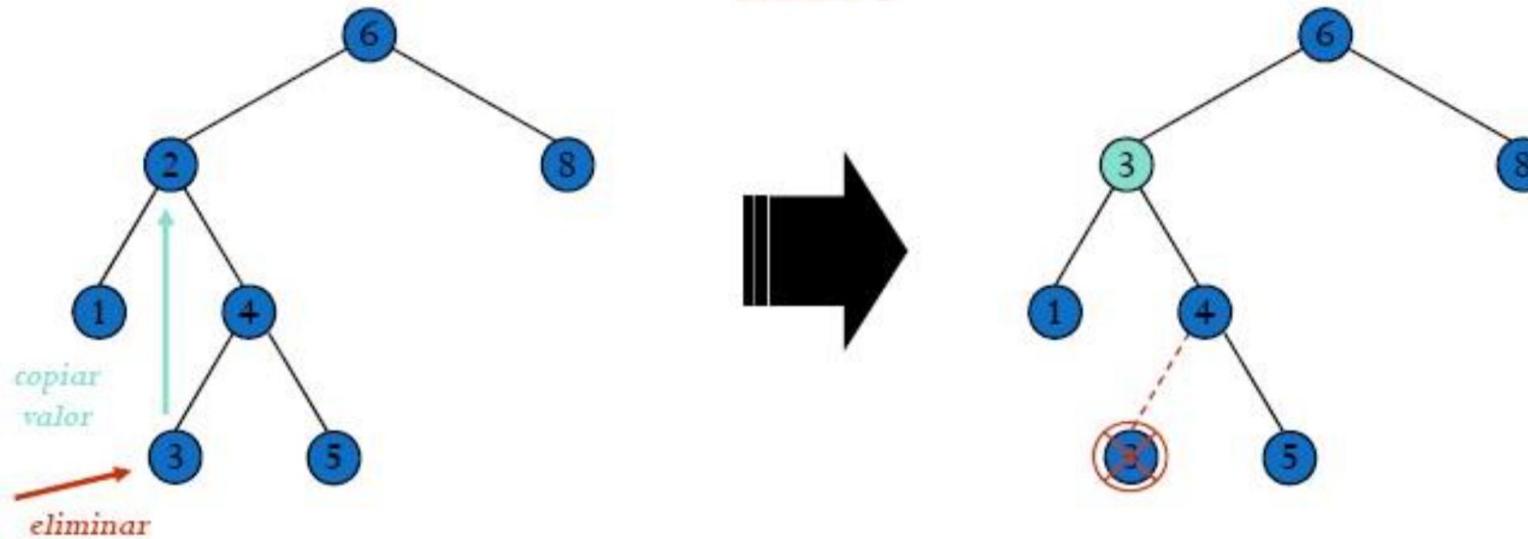
- Para eliminar un nodo, su padre deberá referenciar a su hijo.



- Al eliminar la llave 4, se debe cambiar el subárbol izquierdo de 5 por el que contiene a 2 como raíz.
- Es el mismo procedimiento para eliminar un nodo con un único hijo izquierdo.
- Al eliminar el nodo cuya información es 6, el subárbol izquierdo de 8 referenciará al subárbol derecho del 6 (nodo 7)

# ELIMINAR (CASO COMPLEJO)

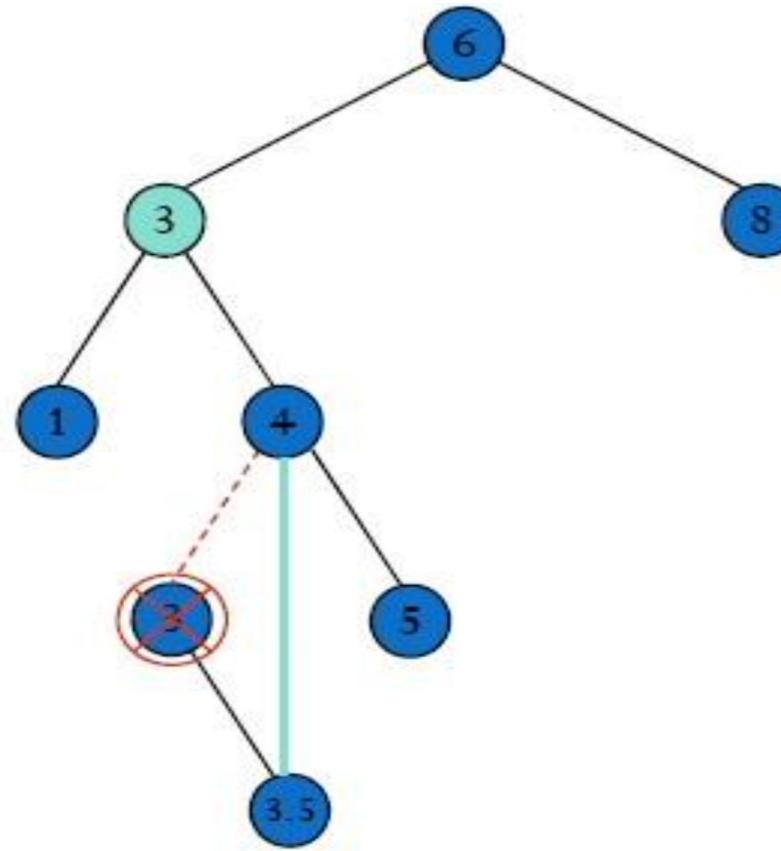
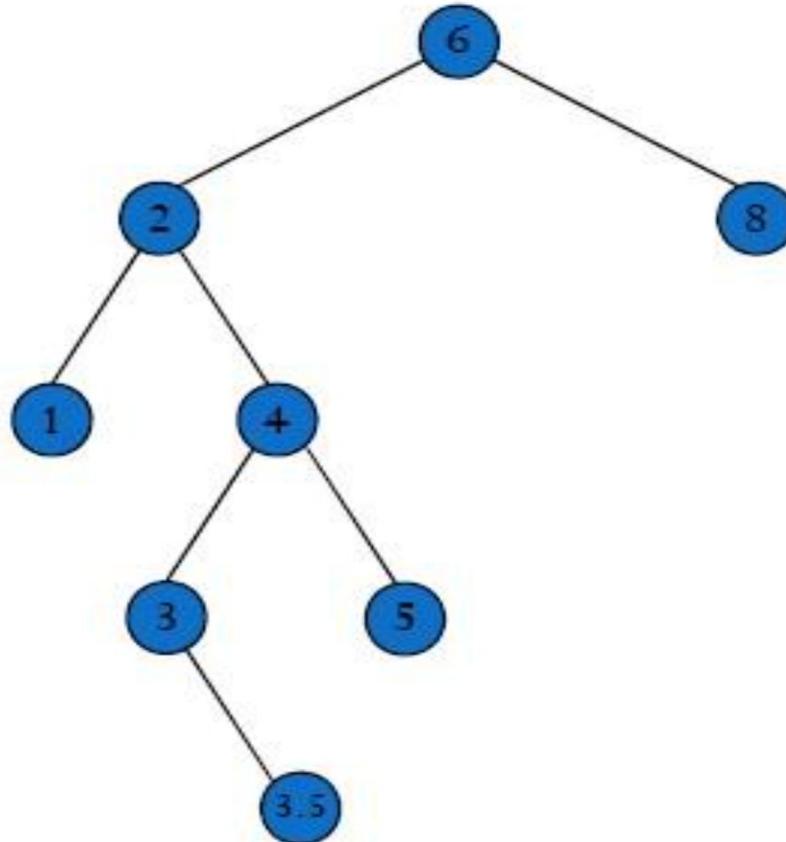
Eliminar nodo con dos hijos



- Remplazar el dato del nodo que se desea eliminar con el dato del nodo más pequeño del subárbol derecho
- Después, eliminar el nodo más pequeño del subárbol derecho (caso fácil)

# ELIMINAR (CASO COMPLEJO)

Eliminar nodo con dos hijos





# CON DOS HIJOS

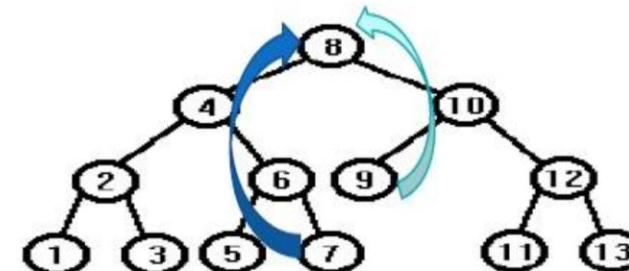
Existen dos opciones:

- Sustituir el valor de la información del nodo a eliminar por el nodo que contiene la menor llave del subárbol derecho (Menor de los mayores).
- Sustuir el valor de la información del nodo a eliminar por el nodo que contiene la mayor llave del subárbol izquierdo (mayor de los menores)

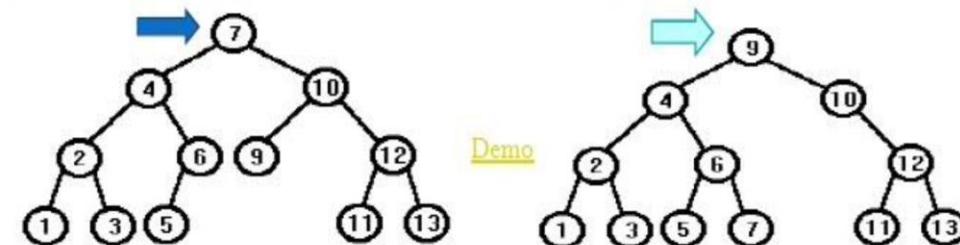
# ELIMINACIÓN CON DOS HIJOS

Si se desea eliminar la raíz:

- Sustituir el valor por la llave con el mayor de los menores y luego eliminar ese valor
- O sustituir el valor de la llave con el menor de los mayores y luego eliminar ese valor



Utilizando el mayor de los menores • Utilizando el menor de los mayores



# ACTIVIDAD EN CLASE



# PROBLEMA 1



Se desea desarrollar una aplicación de consola en C# para gestionar Empleados con atributos código, nombre, DNI utilizando una estructura de árbol binario.

El sistema debe permitir el ingreso, visualización y manipulación de empleados a través de un menú amigable.

No se permite utilizar List.

Desarrollar los siguientes métodos

1. Reporte de Datos de empleado(código, nombre y DNI).
2. Mostrar el peso del árbol .
3. Retornar la altura del árbol.
4. Imprimir el mayor y menor valor de DNI.
5. Retornar la cantidad de nodos del árbol
6. Eliminar el Árbol

# CONCLUSIONES



- Las aplicaciones del uso de estas estructuras de datos son diversas.
- Pueden aplicarse a construcción de aplicaciones donde se gestionen cantidades de datos variables o inciertas.
- Estas aplicaciones pueden ser de gestión de inventarios, gestión de archivos, entre otras.

# ¿PREGUNTAS O COMENTARIOS?





# COMPROBACIÓN DEL LOGRO



# Comprobación del Logro



Cuestionario en la plataforma

Nivel	Rango
Nivel 4	17 – 20
Nivel 3	13 – 16
Nivel 2	9 – 12
Nivel 1	0 - 8

# Comprobación del Logro



Criterio	Nivel 4 (Excelente)	Nivel 3 (Bueno)	Nivel 2 (Regular)	Nivel 1 (Deficiente)
Comprensión conceptual	Demuestra comprensión completa de los conceptos clave.	Comprende la mayoría de los conceptos clave.	Muestra comprensión parcial, con algunas confusiones.	No demuestra comprensión clara de los conceptos.
Aplicación de conocimientos	Aplica correctamente los conceptos en ejemplos o escenarios.	Aplica conceptos con algunos errores menores.	Aplica de forma superficial o con errores evidentes.	No logra aplicar los conceptos o lo hace incorrectamente.
Claridad y precisión en respuestas	Las respuestas son claras, bien estructuradas y precisas.	Las respuestas son mayormente claras, con leves imprecisiones.	Respuestas poco claras o con ideas mal organizadas.	Respuestas confusas, incompletas o incoherentes.
Cobertura del contenido esperado	Responde completamente las 5 preguntas, desarrollando bien cada una.	Responde correctamente 4 de las 5 preguntas.	Responde correctamente 2 o 3 preguntas.	Responde solo 1 pregunta correctamente o no responde.

# BIBLIOGRAFIA REFERENCIAL



- Ceballos Sierra, F. Microsoft C#: Curso de Programación (2a.ed.) 2014  
<https://elibronet.eu1.proxy.openathens.net/es/lc/upnorte/titulos/106417>
- Cesar Liza Avila; Estructura de datos con C/C++

**GRACIAS**

