Estructura de Datos

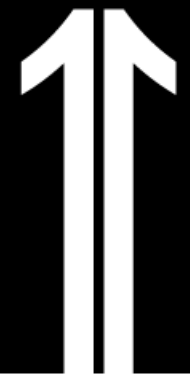
Del 17 al 21 de Junio del 2024

UPN.EDU.PE

Semana 05

ÁRBOLES

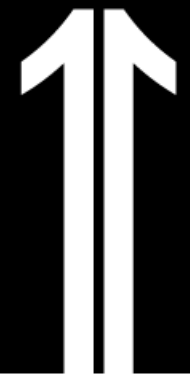
PRESENTACIÓN DE LA SESIÓN
Logro de la Sesión y Temario



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



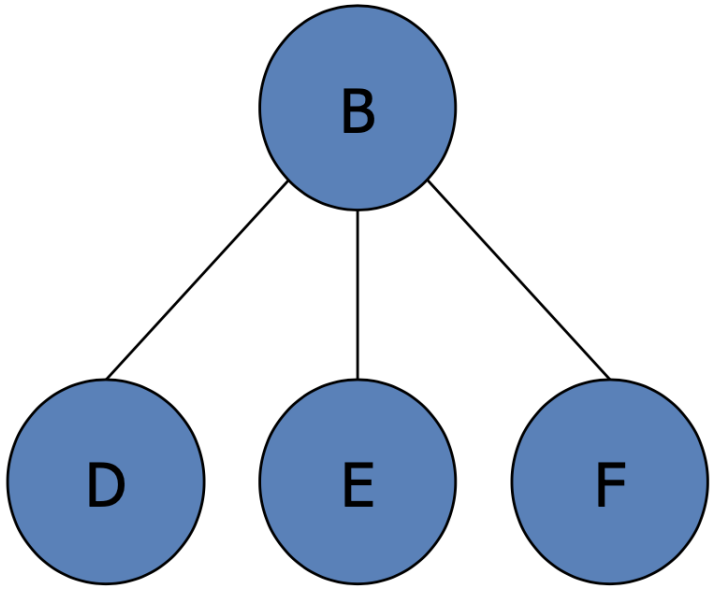
- ¿Qué es árbol?
- ¿Qué es un árbol binario?

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

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 padre, 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 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

TDA ARBOL : DEFINICION INFORMAL

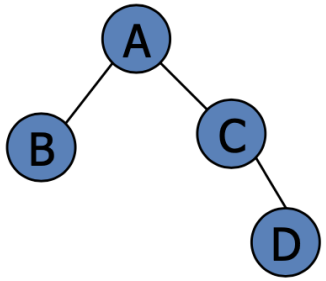
- Valores:
- Conjunto de elementos, donde SOLO se conoce el nodo raíz
- Un nodo puede almacenar contenido y estar enlazado con
    - Sus árboles hijos (pueden ser dos o varios)

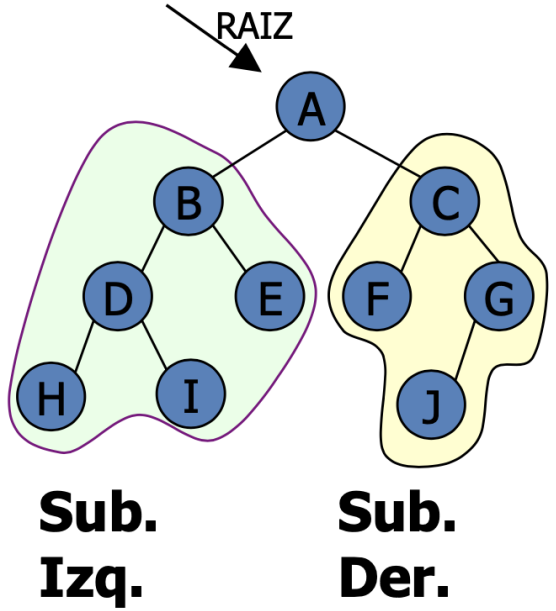
TDA ARBOL : DEFINICION INFORMAL

- Operaciones: Dependen del tipo de árbol, pero en general tenemos
- Vaciar o inicializar el Arbol
  - voidArbol_Vaciar(Arbol*A);
- Eliminar un árbol
  - voidArbol_Eliminar(Arbol*A);
- Saber si un árbol esta vacío
  - boolArbol_EstaVacio(ArbolA);
- Recorrer un árbol
- voidPreOrden(ArbolA)
- voidEnOrden(ArbolA)

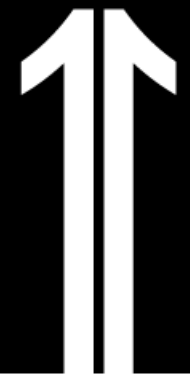
ARBOLES BINARIOS

- 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



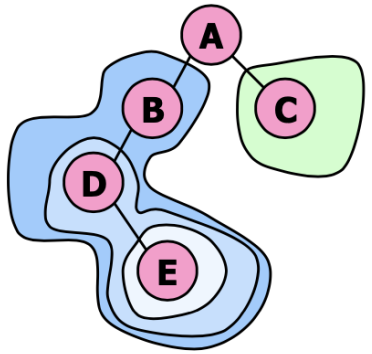


DEFINICIONES RECURSIVAS



SUB. DER.
Nivel 1

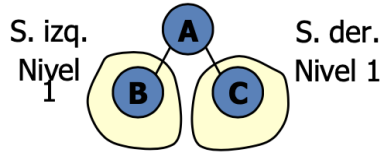
- La definición del árbol es recursiva
  - Se basa en si misma
- La terminología de los árboles
  - También puede ser definida en forma recursiva
- Ejemplo: NIVEL de un árbol
  - Identificar el caso recursivo y el caso mas básico



Caso Básico
Un árbol con un solo nodo tiene nivel 1

Caso Recursivo
Si tiene mas de un nodo, el nivel es:
1 + MAX(Nivel(SubIzq), Nivel(SubDer))

nivel 1 → A

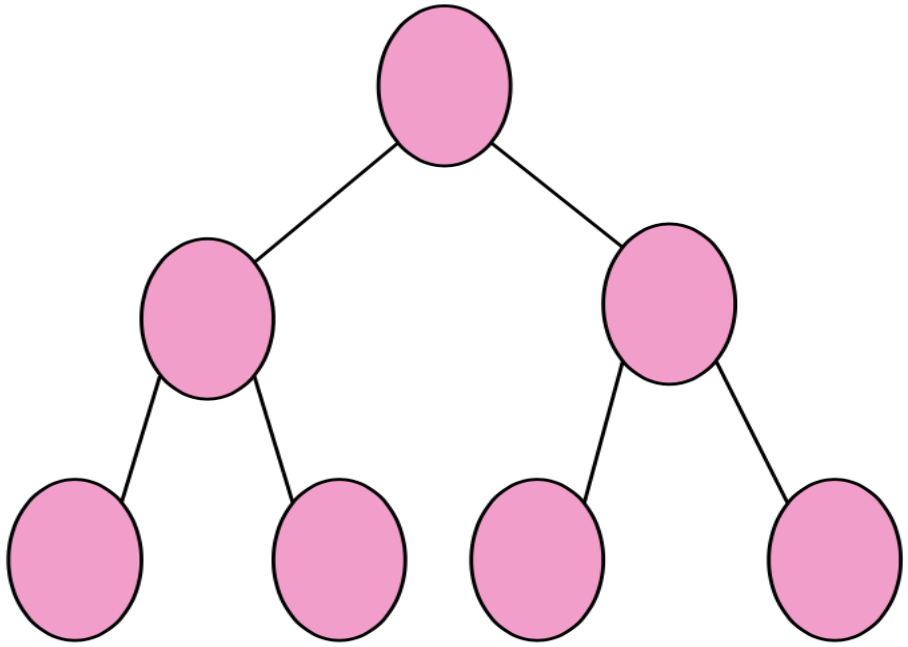


SUB. IZQ.
Nivel = 3

NIVEL : 4

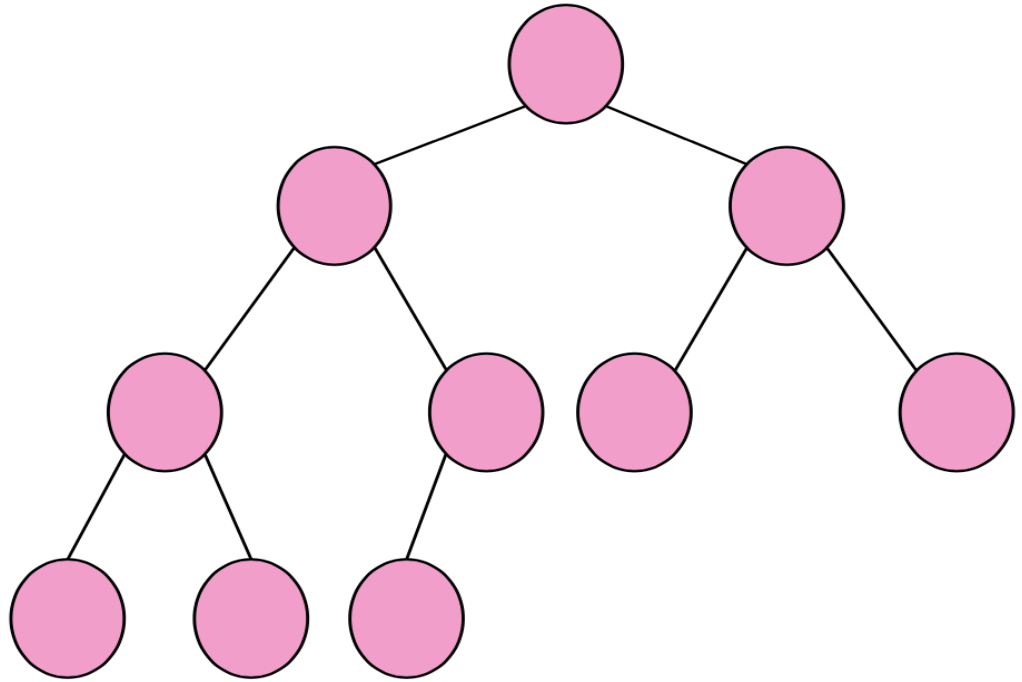
ARBOLES BINARIOS LLENOS

- 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

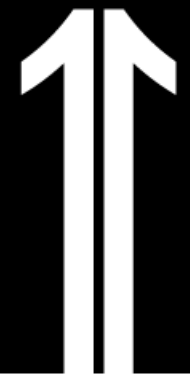


ARBOLES BINARIOS COMPLETOS

- 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 están a nivel h
- Si un arbol esta lleno, tambien esta completo



OTROS

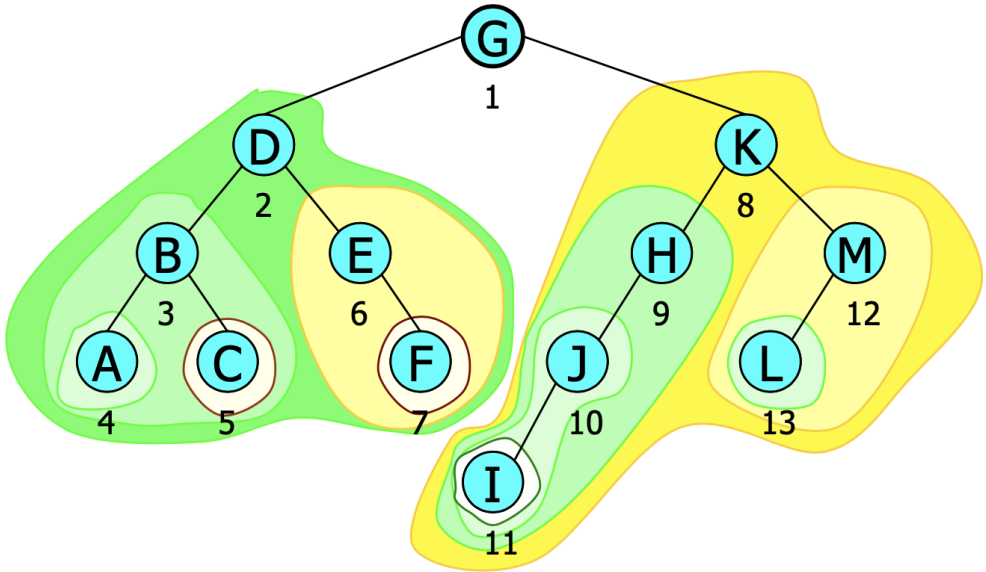


- Un árbol equilibrado es cuando
  - La diferencia de altura entre los subárboles de cualquier nodo es máximo 1
- Un árbol binario equilibrado totalmente
  - Los subárboles izquierdo y derecho de cada nodo tienen la misma altura: es un árbol lleno
- Un árbol completo es equilibrado
- Un árbol lleno es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

EJEMPLO PREORDEN

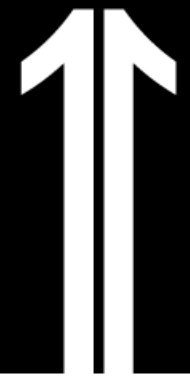
PREORDEN

1. Visitar raiz
2. Preorden al Subarbol Izq.
3. Preorden al Subarbol Der.



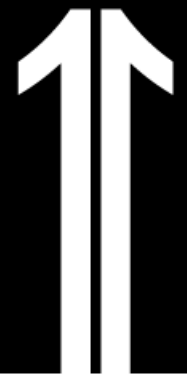
G-D-B-A-C-E-F-K-H-J-I-M-L

AB Y NODOAB: DECLARACION



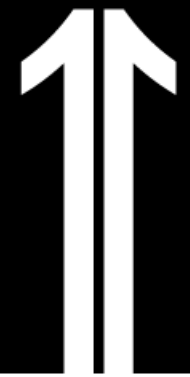
- Un árbol binario: conjunto de nodos
  - Solo se necesita conocer el nodo raíz
- Cada nodo
  - Tiene Contenido y
  - Dos enlaces: árbol hijo izquierdo, árbol hijo derecho
- Un nodo hoja, es aquel cuyos dos enlaces apunta a null
  - Un nodo en un árbol tiene mas punteros a null que un nodo de una lista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

AB: OPERACIONES



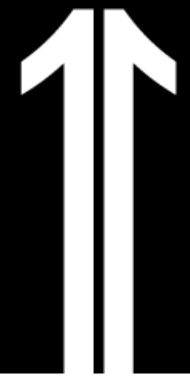
- Crear y Eliminar
  - AB_Vaciar(AB *A);
  - AB_Eliminar(AB *A);
- Estado del Arbol
  - bool AB_EstaVacio(AB A);
- Añadir y remover nodos
  - void AB_InsertarNodo(AB *A, NodoAB *nuevo)
  - NodoAB *AB_SacarNodoxContenido(AB *A, Generico G, Generico_fnComparar fn);
  - NodoAB * AB_SacarNodoxPos(AB *A, 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);
      }
}
```

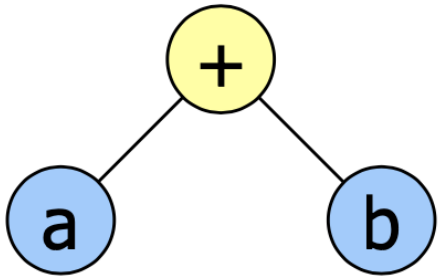
APLICACIÓN: EVALUACION DE EXPRESIONES

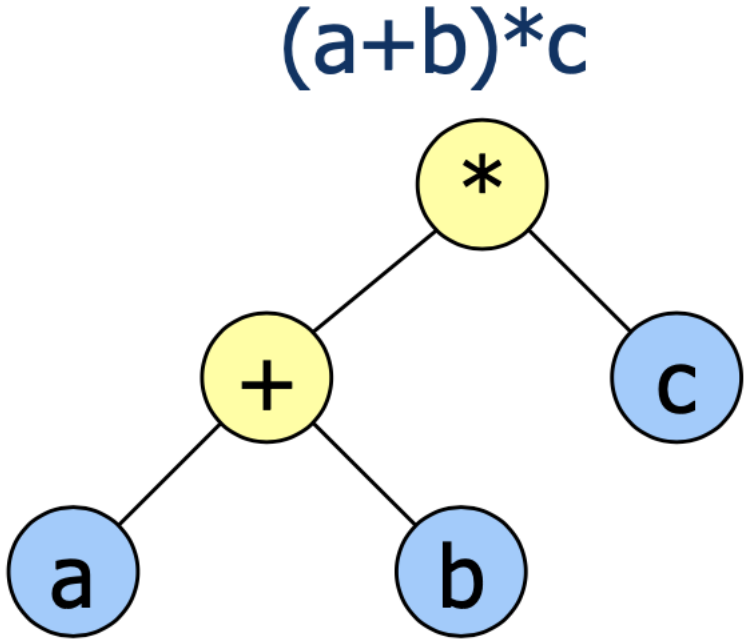


- Ya sabemos lo de las expresiones, cierto?
  - InFija, operador en medio
  - PreFija, operador antes de dos operandos
  - PosFija, operador luego de dos operandos
- Para evaluar una expresion dada, podriamos
  - Pasarla a posfija y usar solo pilas
  - Pasarla a posfija y usar pilas y un arbol de expresion

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)



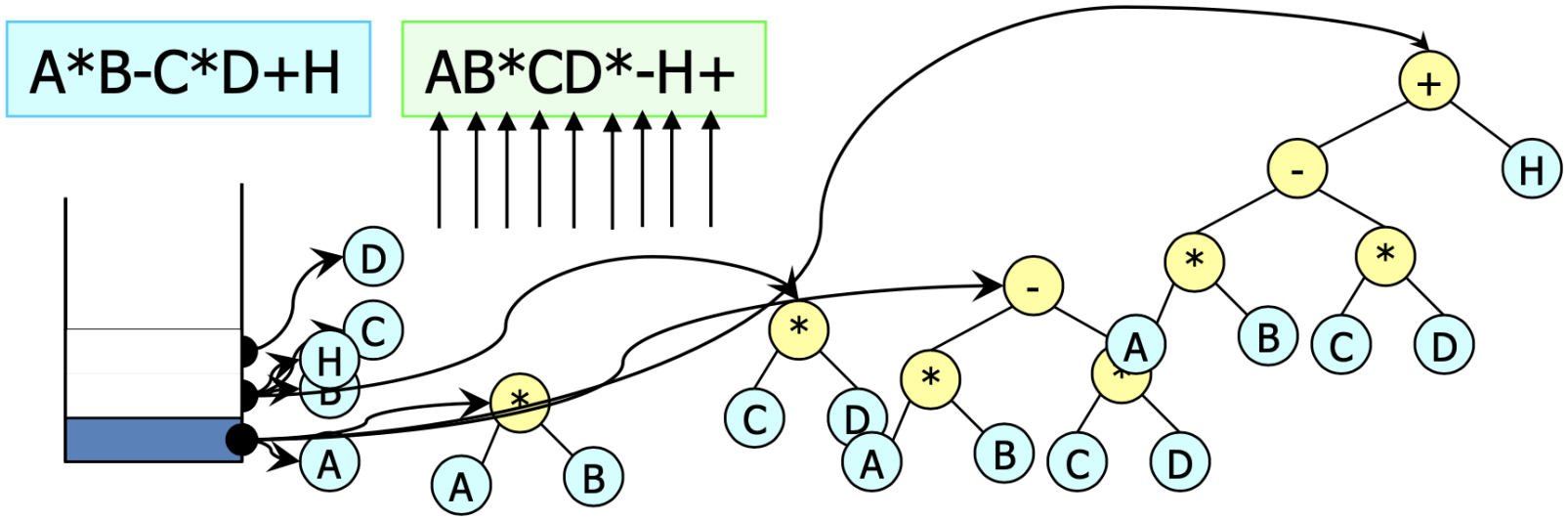


EVALUAR UNA EXPRESION ARTIMETICA EN INFIJA

- 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

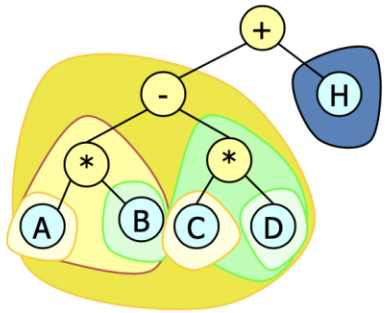
CREAR UN ARBOL DE EXPRESION

- 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



(A * B) - (C*D) + H

Para evaluar el arbol:

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

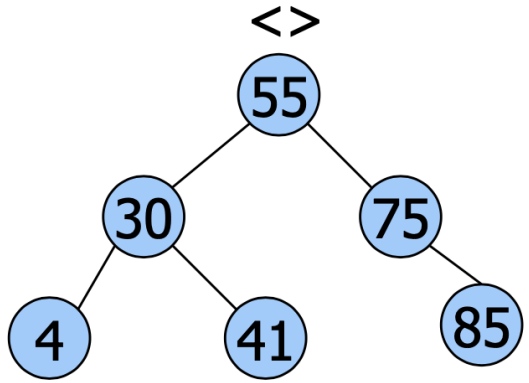
El resultado de la evaluacion es el valor de ese operando

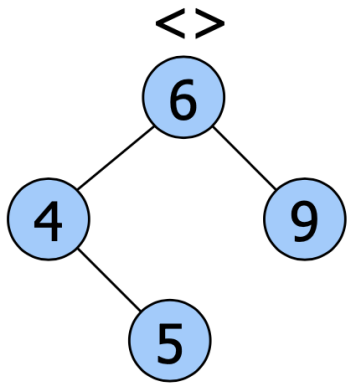
Si no

1. Res1 = Evaluo subarbol izquierdo
2. Res2 = Evaluo subarbol derecho
3. Recupero la info de la raiz y efectuo la operación alli 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

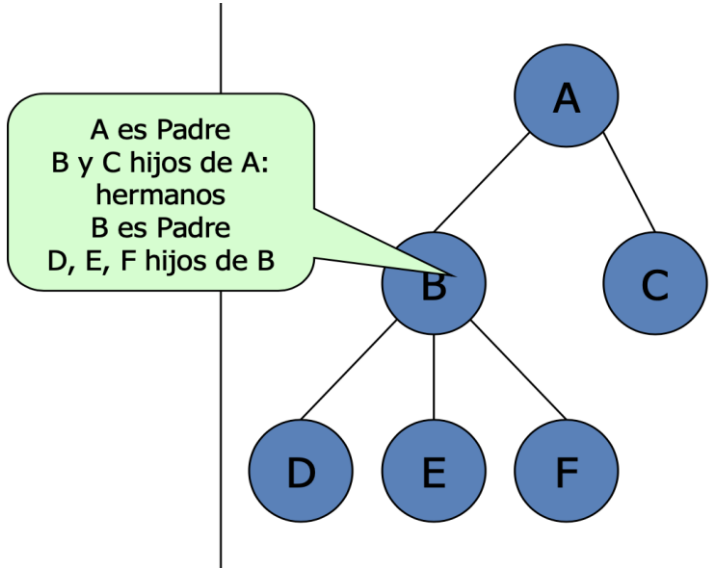


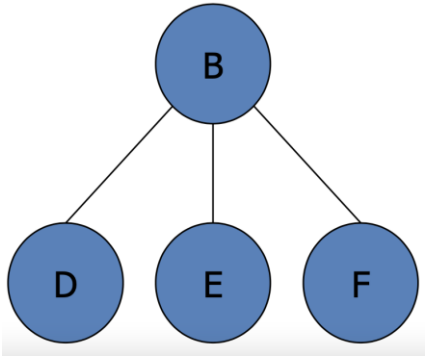


ARBOLES BINARIOS DE BÚSQUEDA ABB

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 padre, 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

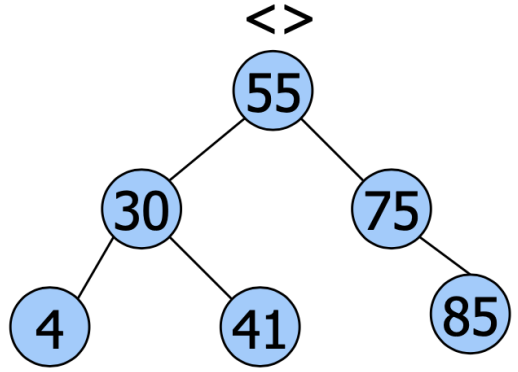
- **Camino:** Secuencia de nodos conectados dentro de un árbol
- **Longitud del camino:** es el numero de nodos menos 1 en un camino
- **Altura del árbol:** es el nivel mas alto del árbol
  - Un árbol con un solo nodo tiene altura 1
- **Nivel(profundidad) de un nodo:** es el numero de nodos entre el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

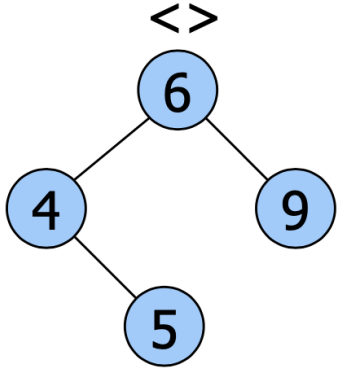
TERMINOLOGIA

- 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

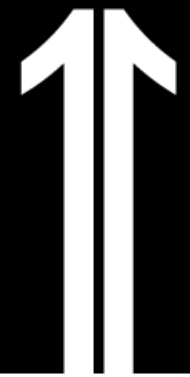
ARBOL BINARIO DE BUSQUEDA

- 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



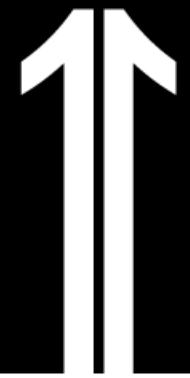


OPERACIONES CON UN ABB



- Inserción
- Búsqueda
- Recorridos

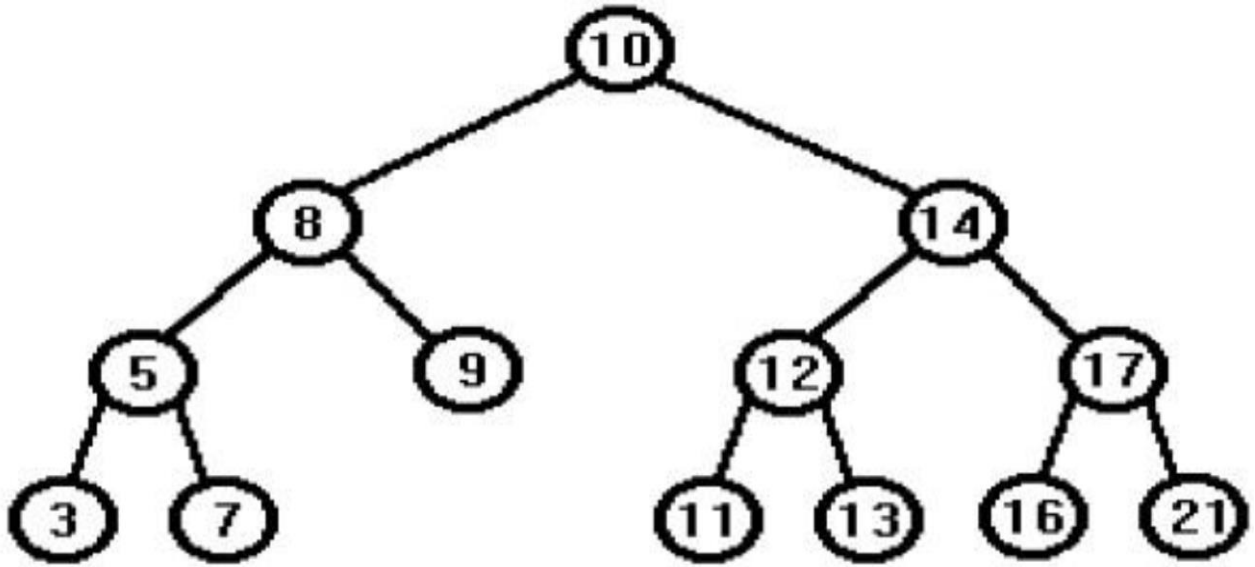
INSERCIÓN



- 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*

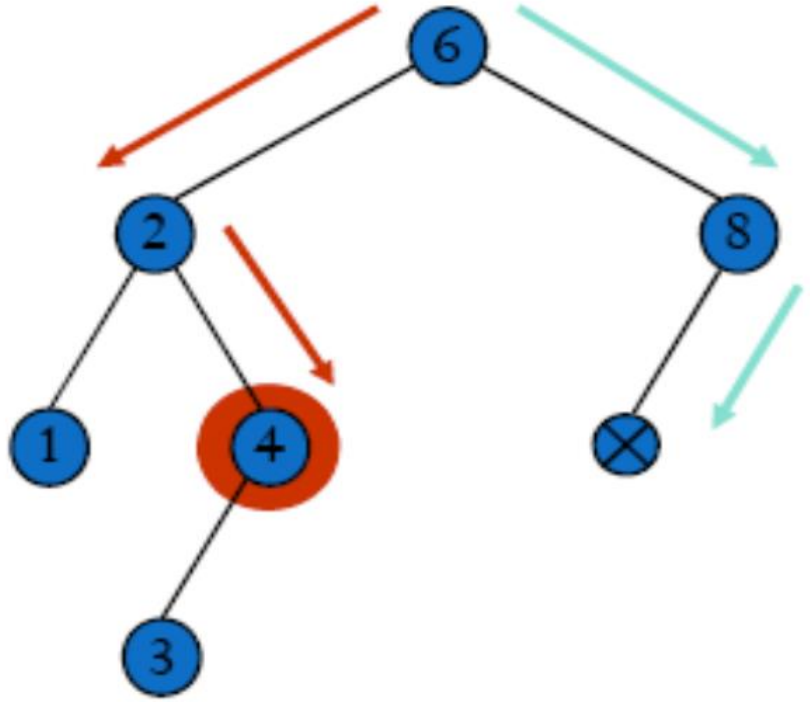
INSERCIÓN, EJEMPLO

- 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*

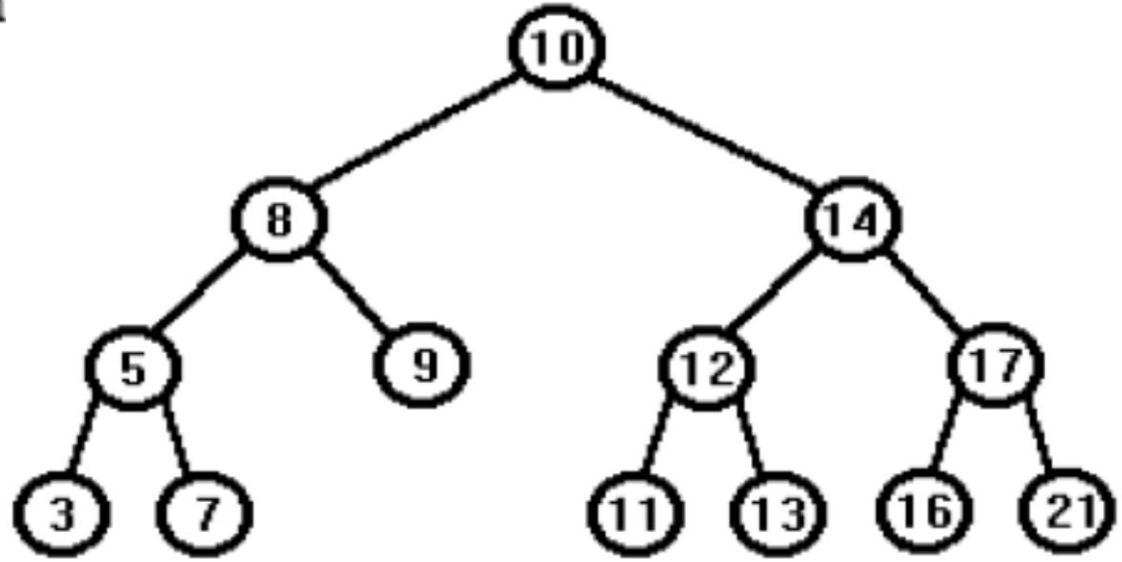


Buscando 4: VERDADERO

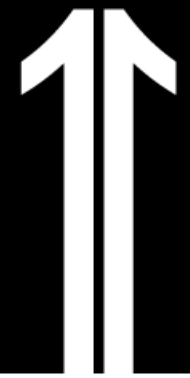
Buscando 7: FALSO

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 derech
- Se encuentra la llave deseada

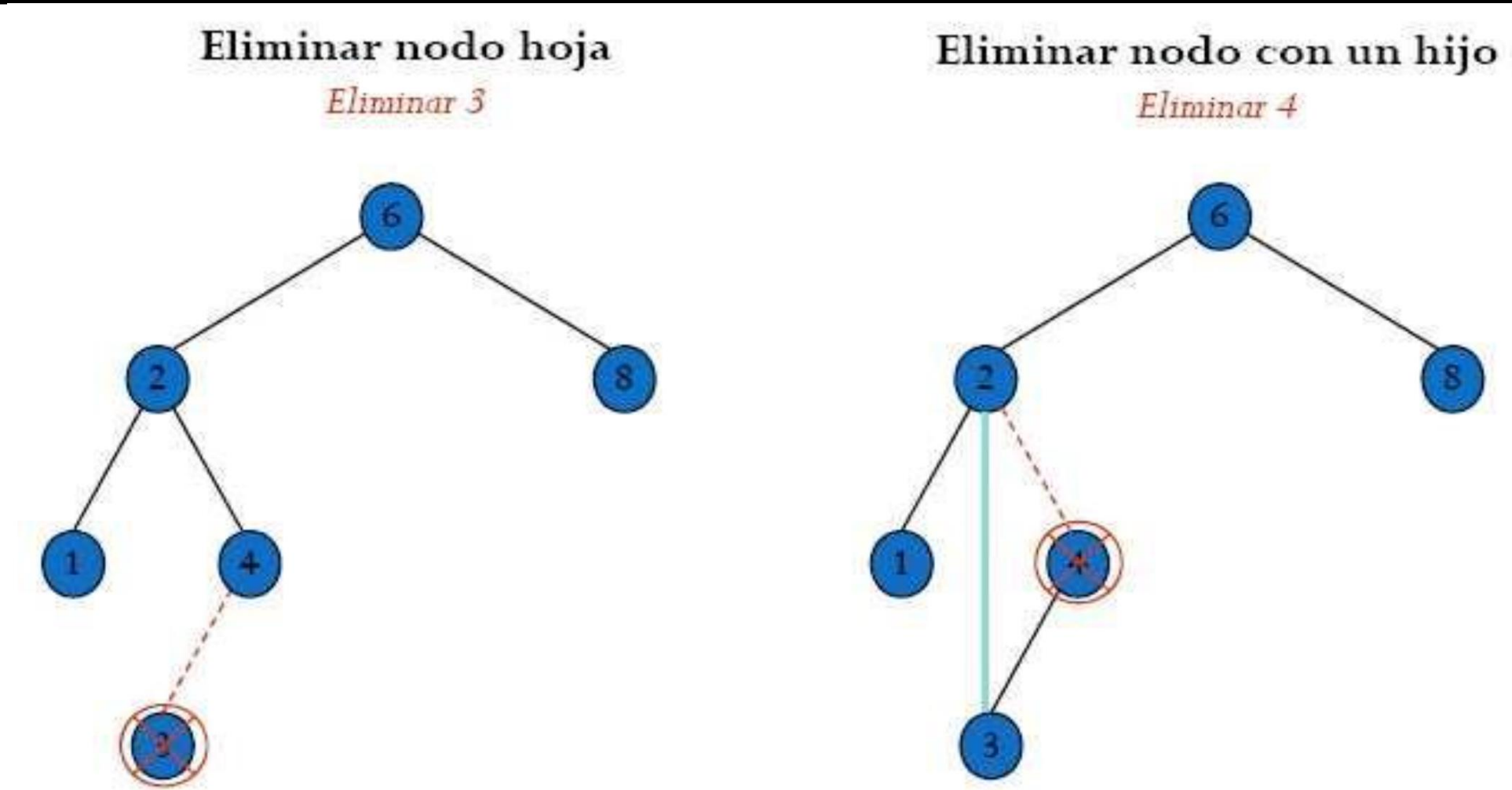


ELIMINACIÓN



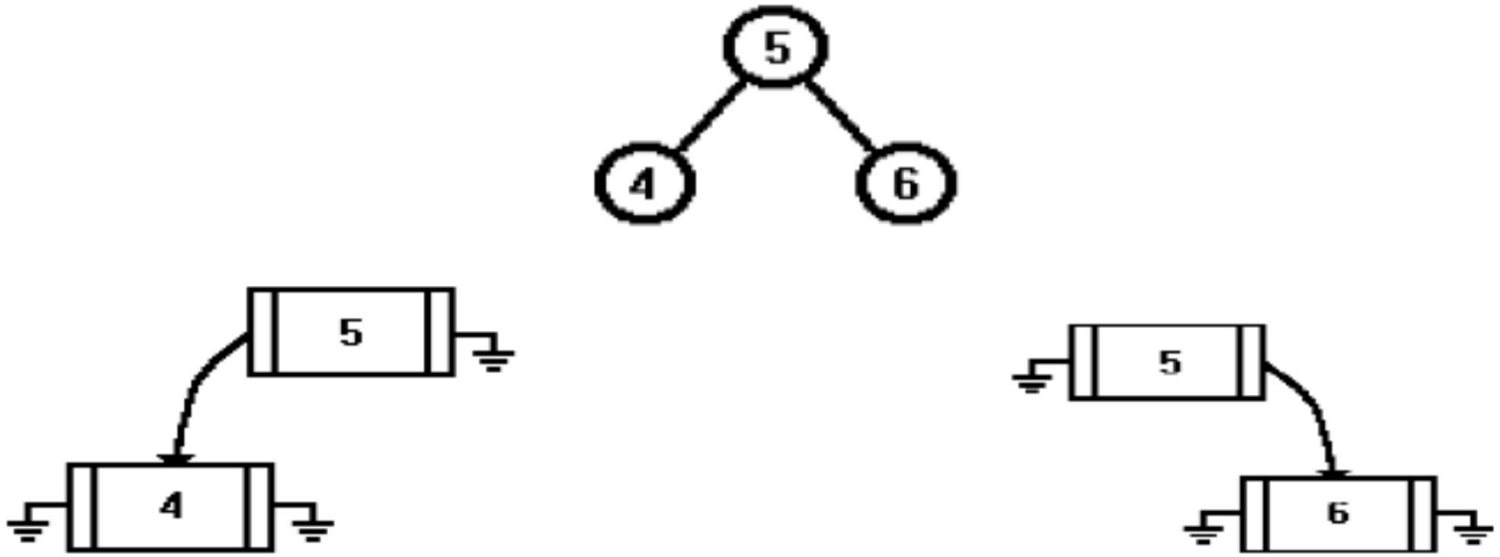
- Existen cuatro distintos escenarios:
    1. Intentar eliminar un nodo que no existe.
       No se hace nada, simplemente se regresa FALSE.
    2. Eliminar un nodo hoja.
       Caso sencillo, se borra el nodo y se actualiza el apuntador del nodo padre a NULL
    3. Eliminar un nodo con un solo hijo
       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)



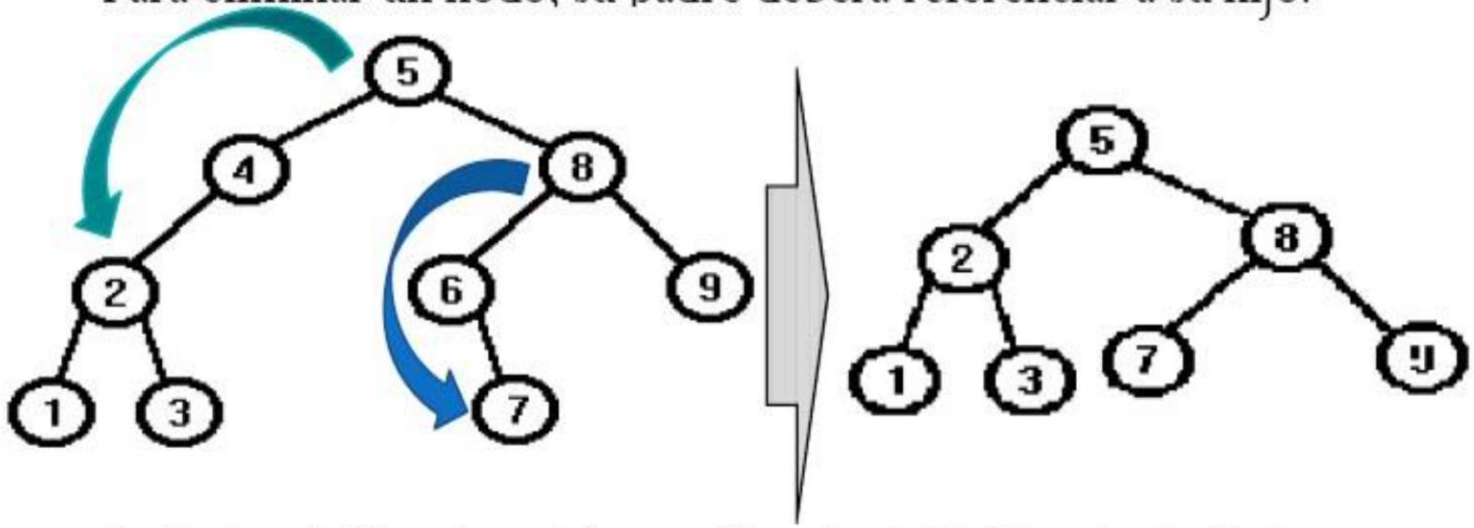
ELIMINACIÓN DE UNA HOJA

- 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

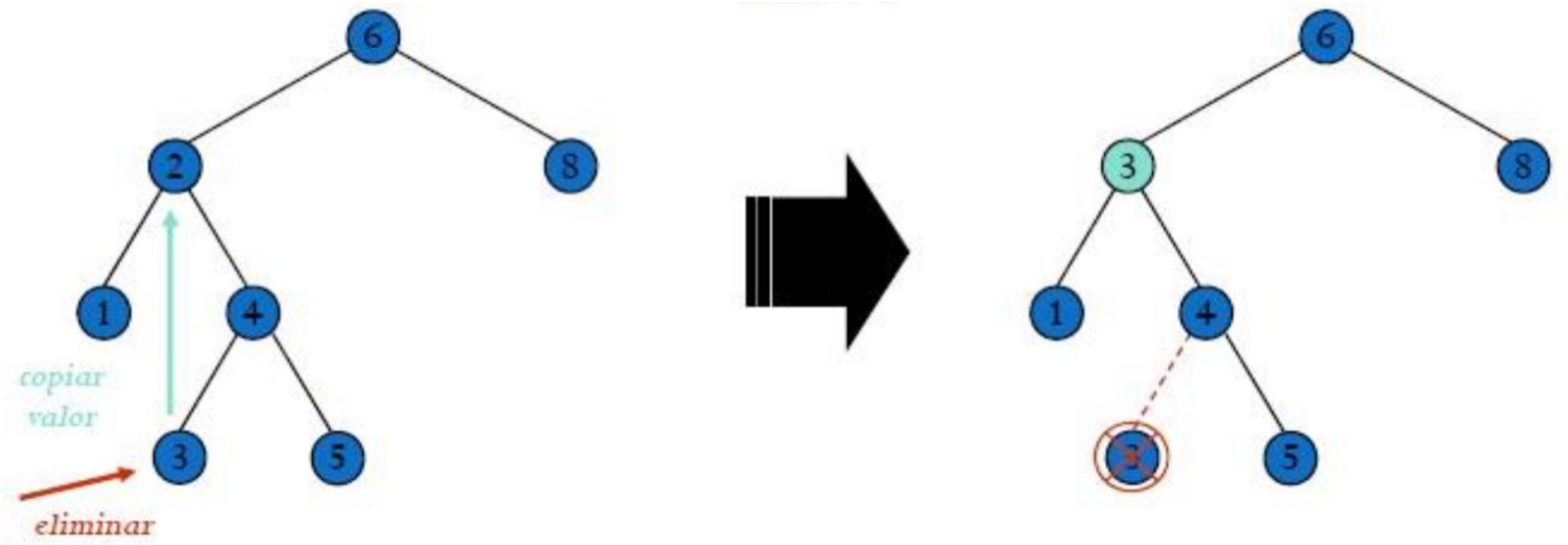
- Para eliminar un nodo, su padre deberá referenciar a su hijo.



- 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)

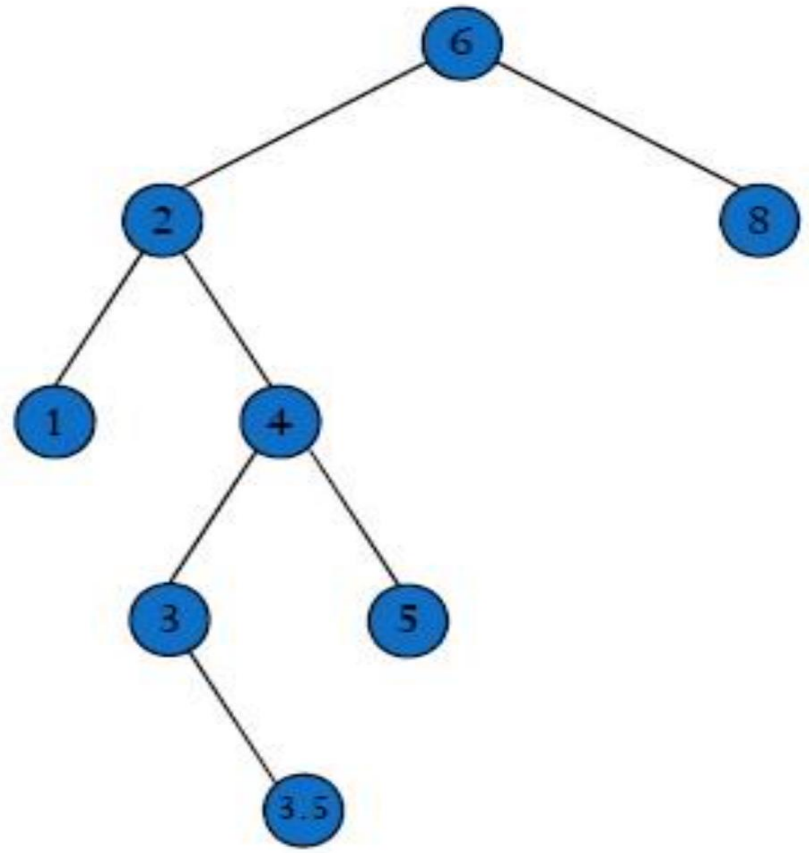
Eliminar nodo con dos hijos

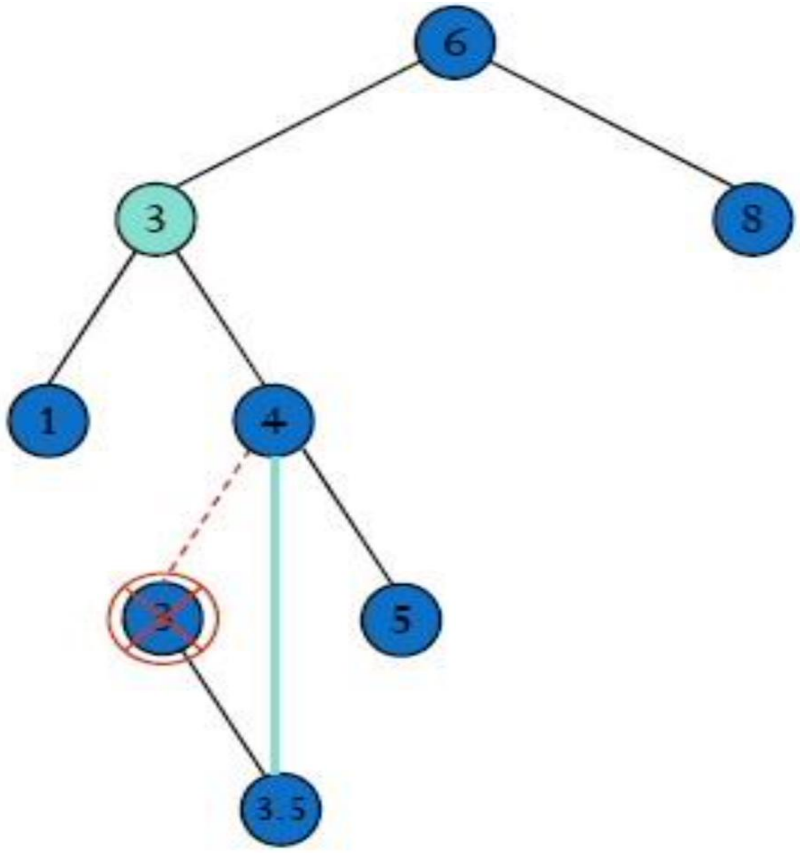


- Remplazar el dato del nodo que se desea eliminar con el dato del nodo más pequeño del subárbol derecho
- Después, eliminar el nodo más pequeño del subárbol derecho (caso fácil)

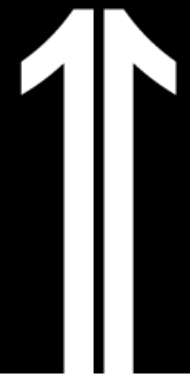
ELIMINAR (CASO COMPLEJO)

Eliminar nodo con dos hijos





CON DOS HIJOS



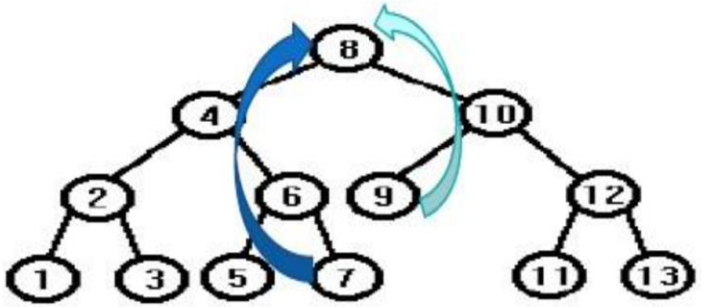
Existen dos opciones:

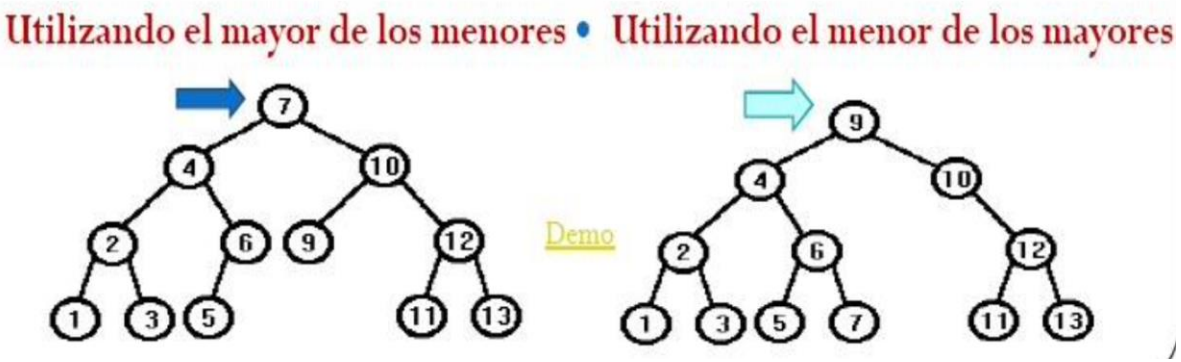
- Sustituir el valor de la información del nodo a eliminar por el nodo que contiene la menor llave del subárbol derecho (Menor de los mayores).
- Sustuir el valor de la información del nodo a eliminar por el nodo que contiene la mayor llave del subárbol izquierdo (mayor de los menores)

ELIMINACIÓN CON DOS HIJOS

Si se desea eliminar la raíz:

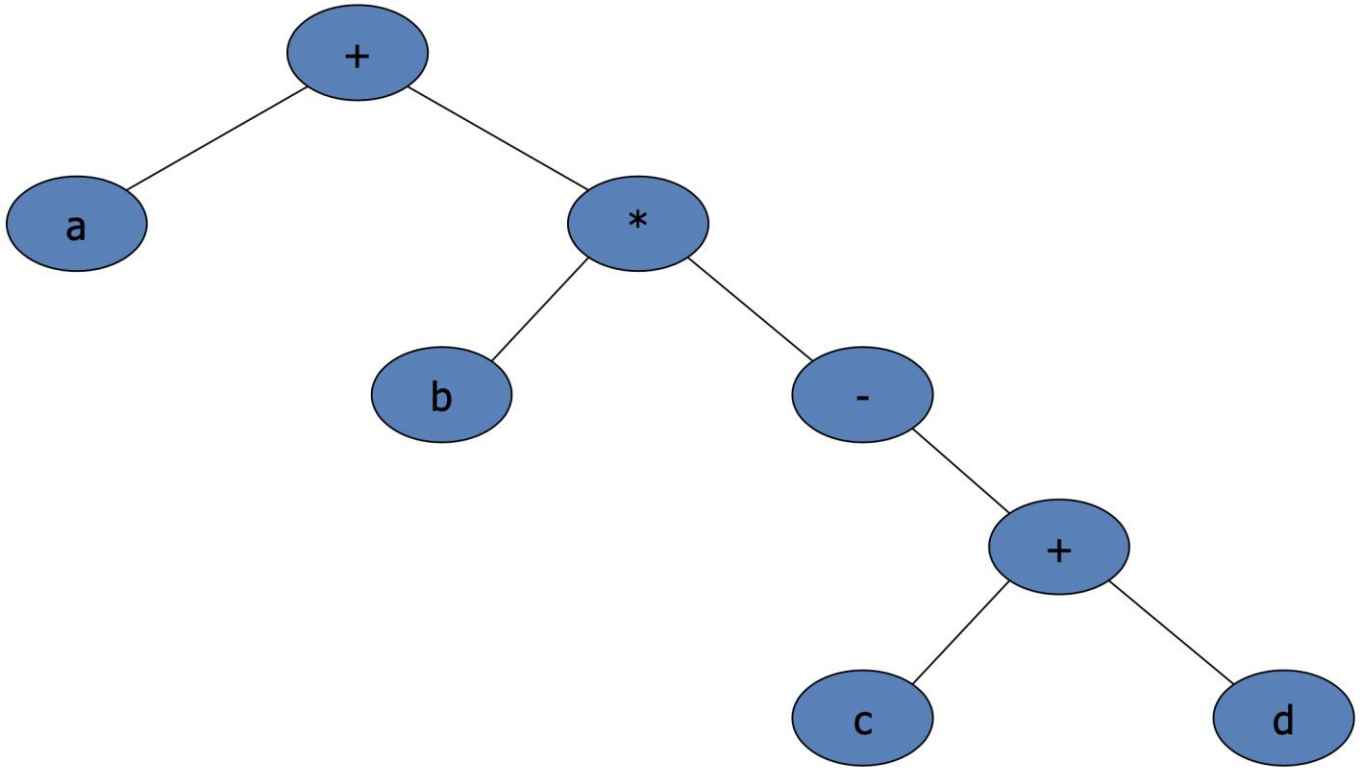
- Sustituir el valor por la llave con el mayor de los menores y luego eliminar ese valor
- O sustutuir el valor de la llave con el menor de los mayores y luego eliminar ese valor



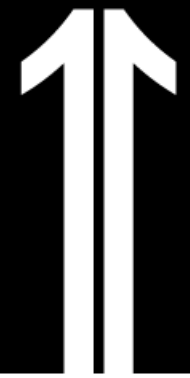


ACTIVIDAD

- Construir arboles de expresion para:
  - [X+(Y*Z)] * (A-B)
- Deducir las expresiones de los siguientes A.B.



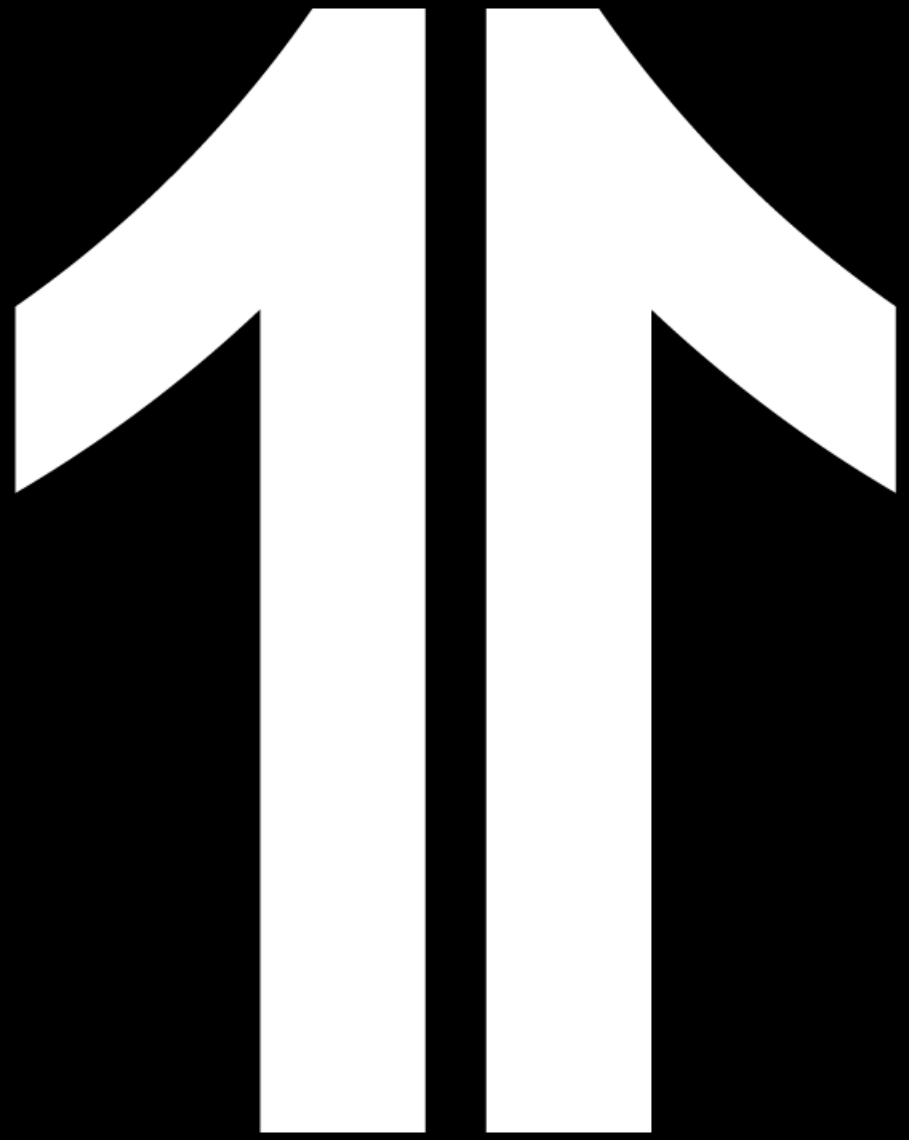
CONCLUSIONES



- Los árboles a diferencia de las listas, son estructuras de datos no lineales.
- Asimismo, la forma de recorrido no necesariamente se da de manera lineal, sino por el contrario también de formas no lineales.
- Los árboles son muy útiles para la búsqueda y recuperación de información

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


UPN
UNIVERSIDAD
PRIVADA
DEL NORTE