Árboles

Un árbol es una estructura jerárquica aplicada sobre una colección de elementos u objetos llamados nodos.

Una de las características importantes en una estructura de árbol es que cada nodo puede apuntar a uno o varios nodos (hijos), pero cada nodo sólo puede ser apuntado por un solo nodo (padre). Esto hace que la estructura de árbol esté fuertemente jerarquizada, y es lo que en realidad les da la apariencia de árbol.

Los árboles crean una relación entre nodos dando lugar a términos tales como: padre, hijo, hermano, antecesor, sucesor, ancestro, raíz, hojas, ramas, etc.

Raíz: Un sólo nodo es por si mismo un árbol, siendo también la raíz de dicho árbol. La raíz del árbol se define como el nodo que no tiene ascendiente.

Hoja: Todo nodo que no tiene ramificaciones es el elemento que no tiene descendientes y se le conoce con el nombre de hoja o nodo terminal.

Altura: La altura de un árbol se define como el nivel del nodo de mayor nivel. Como cada nodo de un árbol puede considerarse a su vez como la raíz de un árbol, también podemos hablar de altura de ramas.

Padre: Es el nodo que contiene un puntero al un (o más) nodo dado.

Hijo: Cualquiera de los nodos apuntados por uno de los nodos del árbol.

Hermano: Los nodos hijos del mismo padre se llaman hermanos. Estos nodos son descendientes directos de un mismo nodo.

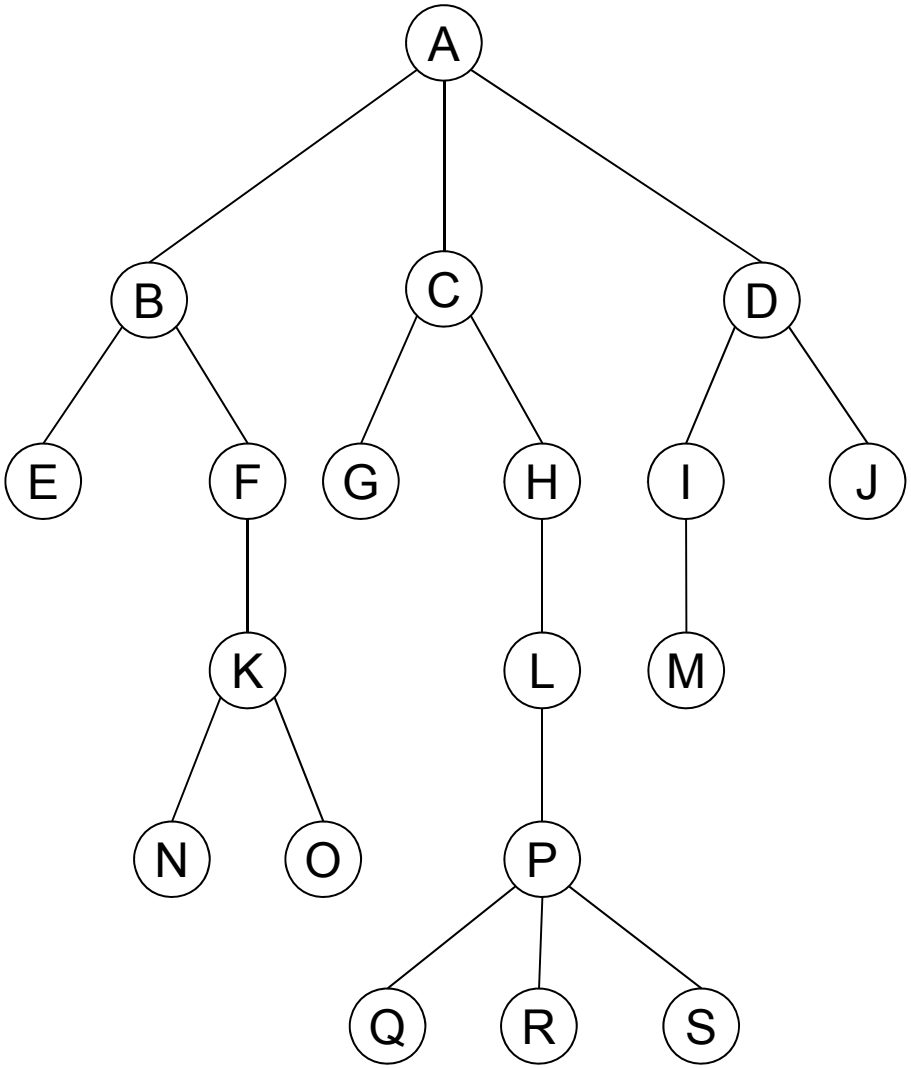
Grado: Esta definido por el número de hijos que puede tener un elemento dentro del árbol.

Peso: Se define como la cantidad de hojas en el árbol.

Ruta: Se define como la longitud de trayectoria interna de un nodo, es el número de aristas o ramas que se recorren desde la raíz al nodo.

Nivel o profundidad: se define para cada elemento del árbol como la distancia a la raíz, medida en nodos. El nivel de la raíz es cero y el de sus hijos uno.

Árbol



Árboles binarios

Una importante simplificación del modelo es la de tomar en cuenta árboles que tengan sólo nodos de grados dos (es el grado mínimo), denominados *árboles binarios*.

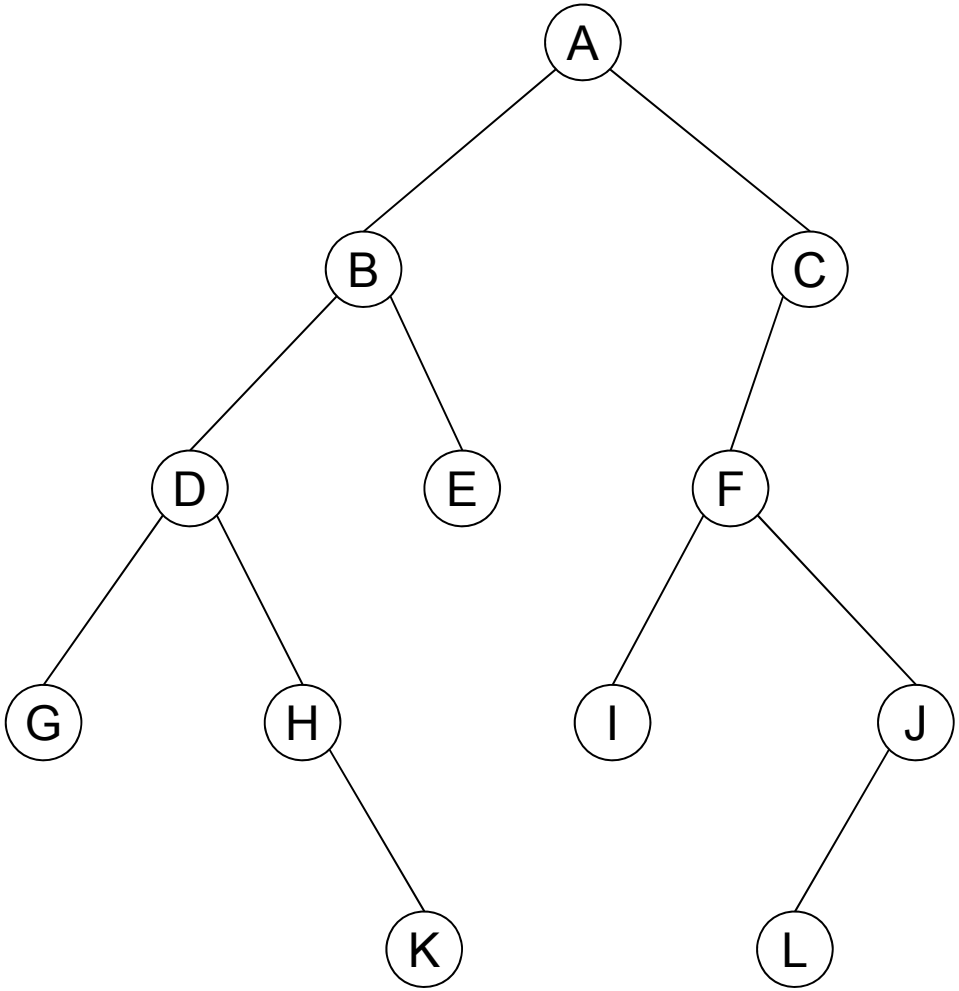
Un árbol binario es aquel en el cual cada nodo tiene no más de dos descendientes, es decir: un nodo puede tener cero, uno, o dos descendientes.

Así cada nodo en un árbol binario puede tener un descendiente izquierdo y/o uno derecho.

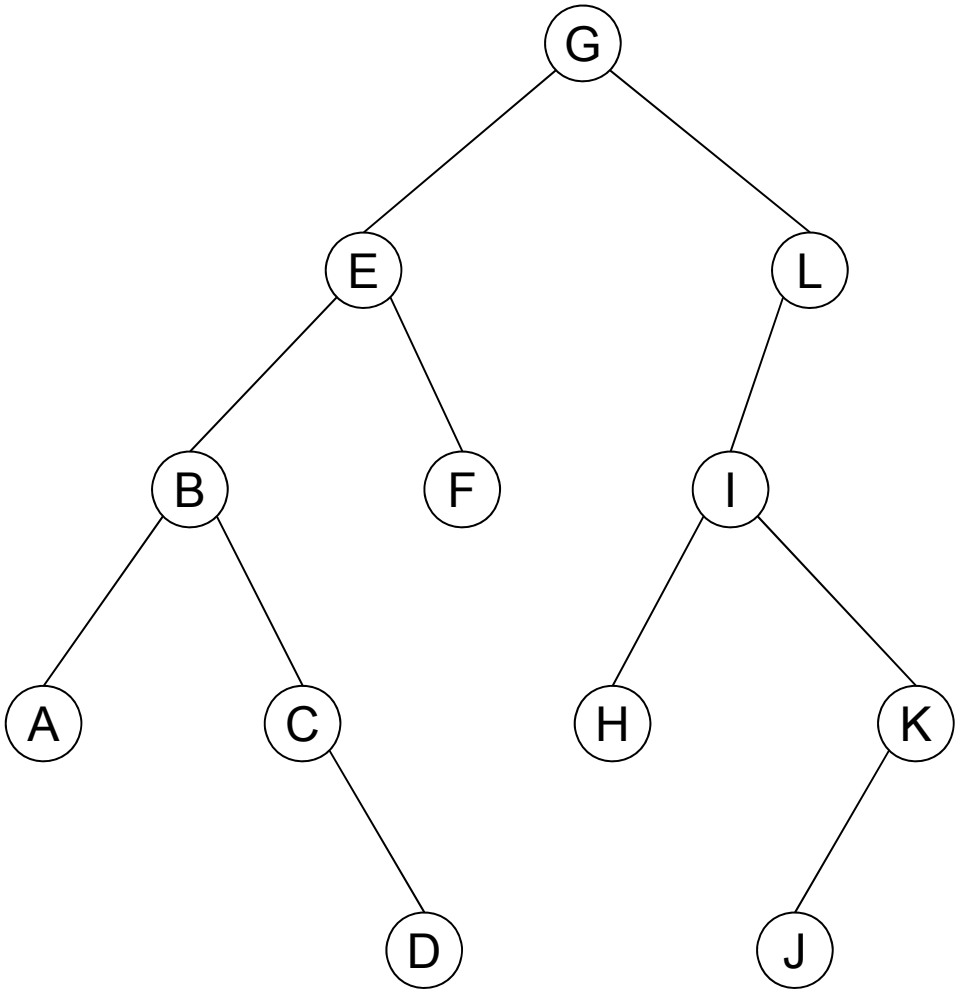
La importancia de un árbol binario radica en que posibilita la creación de una estructura que imita un proceso de decisión si/no (falso/verdadero).

Por ejemplo, si se construye un árbol binario para almacenar valores numéricos, dicho árbol puede tener lo valores menores en los subárboles izquierdos y los valores mayores en los subárboles derechos (o viceversa); lo cual facilita buscar un valor particular en el árbol, con un algoritmo tan simple como el de la búsqueda binaria.

Árbol binario



Árbol binario



Operaciones

inicializa	–	nada
vacío	–	booleano
inserta	–	nada
elimina	–	nada
hijo izquierdo	–	posición
hijo derecho	–	posición
localiza	–	posición
recupera	–	elemento
recorre preorder	–	nada – depende del proceso
recorre inorder	–	nada – depende del proceso
recorre postorder	–	nada – depende del proceso
anula	–	nada

Operaciones

inicializa: Se aplica al crear el árbol, esta operación asegura que el árbol esté vacío en su estado inicial, dejándolo preparado para iniciar operaciones sobre ella. Recibe el árbol que se inicializará. No devuelve nada.

vacío: Revisa el estado actual del árbol e informa si es que se encuentra vacío. Recibe el árbol que se revisará. Si no existe ningún elemento en el árbol regresa *verdadero*, si existe uno o más elementos regresa *falso*.

inserta: Coloca un elemento como hoja del árbol en una posición en la que se mantiene el orden dentro del árbol se genera un error si el árbol se encuentra lleno o si la posición proporcionada es inválida. Recibe el elemento a insertar y la árbol donde se hará la inserción. No devuelve nada.

elimina: Borra un elemento en una posición dada del árbol, cuidando mantener el orden, se genera un error si el árbol encuentra vacío o si la posición proporcionada es inválida. Recibe la posición del elemento a eliminar y árbol donde se hará la eliminación. No devuelve nada.

Operaciones

hijo izquierdo: Obtiene la posición del hijo izquierdo respecto a un subárbol en una posición dada. Recibe un árbol y la posición de un subarbol. Regresa la posición del hijo izquierdo, si el árbol esta vacío, o si la posición dada es inválida, o no existe un hijo izquierdo en ésa posición, regresa una posición inválida.

hijo derecho: Obtiene la posición del hijo deecho respecto a un subárbol en una posición dada. Recibe un árbol y la posición de un subarbol. Regresa la posición del hijo derecho, si el árbol esta vacío, o si la posición dada es inválida, o no existe un hijo derecho en ésa posición, regresa una posición inválida.

localiza: Determina la posición de un elemento dentro de un árbol. Recibe el elemento a buscar y una árbol. Devuelve la posición del elemento dentro del árbol, si le árbol esta vacío, o si el elemento no se encuentra en el árbol, regresa una posición inválida.

recupera: Obtiene el elemento que se encuentra en una determinada posición en el árbol. Se genera un error si el árbol está vacío o si la posición proporcionada es inválida. Recibe una posición y un árbol. Devuelve el elemento en la posición dada dentro del árbol.

Operaciones

recorre preorder: Realiza un proceso (p.e. imprimir) en todos y cada uno de los elementos de un árbol en una secuencia recursiva en el siguiente orden: Procesa elemento actual → preorder en subárbol izquierdo → preorder en subárbol derecho. Recibe el árbol a recorrer. No regresa nada.

recorre inorder: Realiza un proceso (p.e. imprimir) en todos y cada uno de los elementos de un árbol en una secuencia recursiva en el siguiente orden: Inorder en subárbol izquierdo → procesa elemento actual → inorder en subárbol derecho. Recibe el árbol a recorrer. No regresa nada.

recorre postorder: Realiza un proceso (p.e. imprimir) en todos y cada uno de los elementos de un árbol en una secuencia recursiva en el siguiente orden: postorder en subárbol izquierdo → postorder en subárbol derecho → procesa elemento actual. Recibe el árbol a recorrer. No regresa nada.

anula: Elimina todos y cada uno de los elementos dentro de un árbol, resultando en un árbol vacío. Recibe árbol a anular. No devuelve nada.

Procedimiento para la inserción de un elemento en un árbol binario de búsqueda

Si el subárbol actual esta vacío:
    Crear un nuevo nodo asignarlo como raíz del subárbol
    Colocar el elemento en el campo correspondiente
    Inicializar subárboles izquierdo y derecho
De lo contrario:
    Si el elemento a insertar es menor que el elemento en el subárbol actual:
        Insertar el elemento en el subárbol izquierdo
    De lo contrario:
        Insertar el elemento en el subárbol derecho

Inserta siempre como hoja
Mantiene el árbol ordenado
Mantiene la regla de orden de llegada

Ejemplos en pintarrón

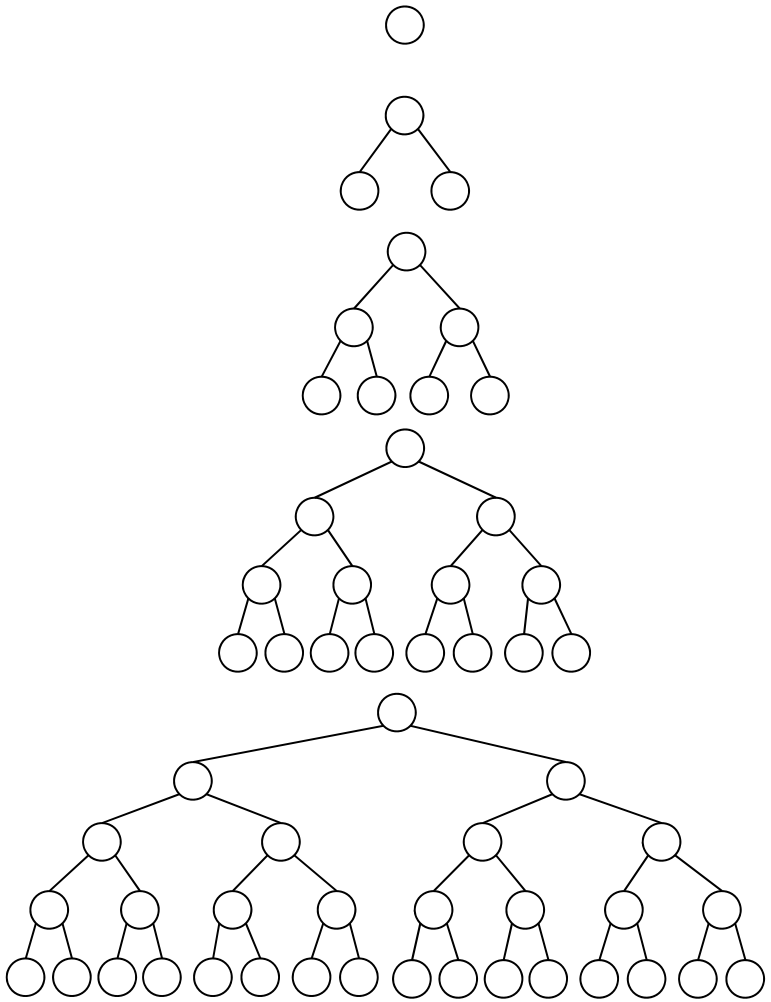
Procedimiento para la eliminación de un elemento en un árbol binario de búsqueda

Si el subárbol a eliminar es una hoja:
    Liberar el espacio de memoria del nodo actual
    Actualizar la referencia a nulo
De lo contrario:
    Sustituir el elemento actual por el elemento mayor del subárbol izquierdo, o por elemento menor del subárbol derecho
    Eliminar el elemento utilizado para sustitución

Elimina siempre una hoja
Mantiene el árbol ordenado
Mantiene la regla de orden de llegada

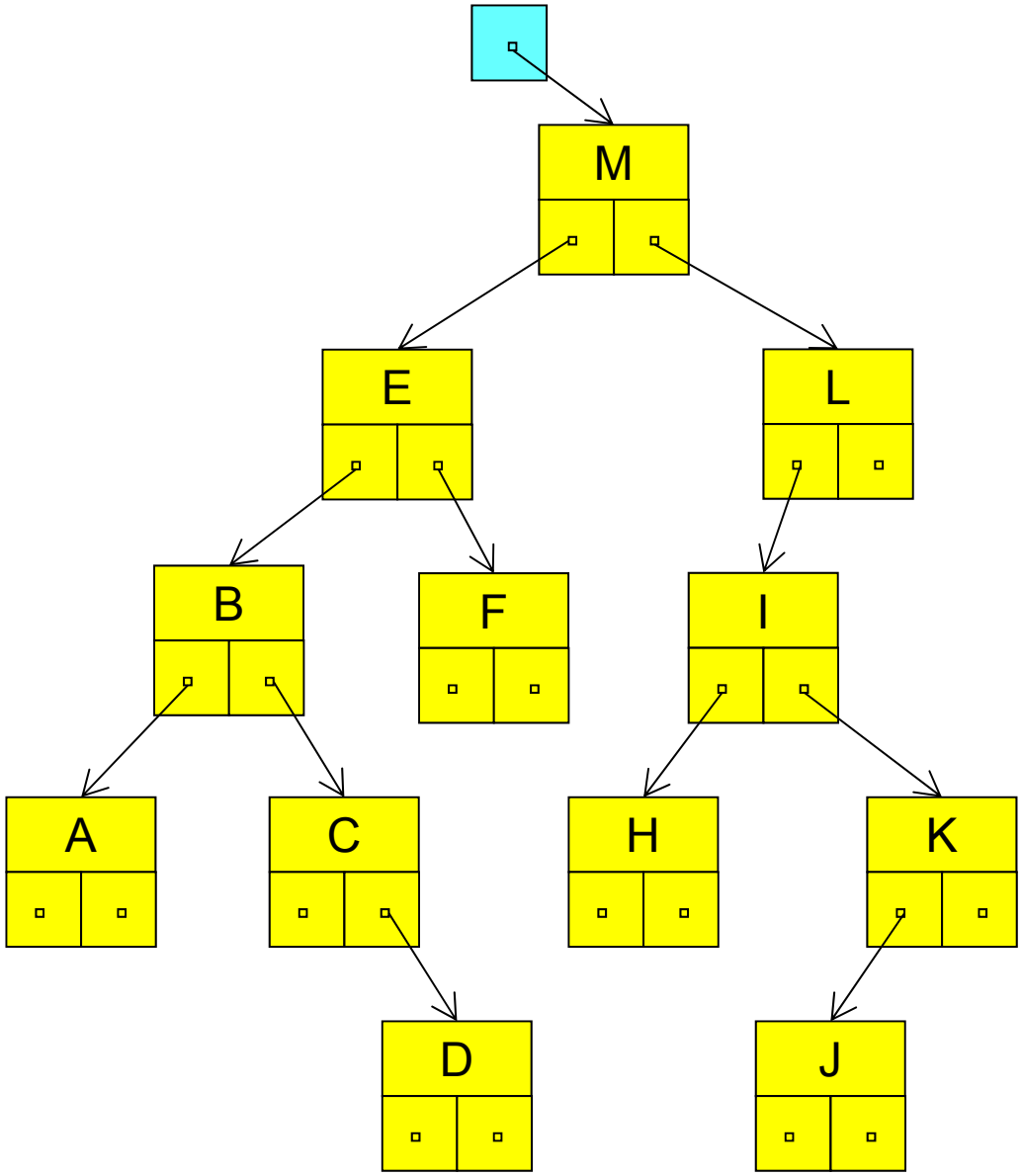
Ejemplos en pintarrón

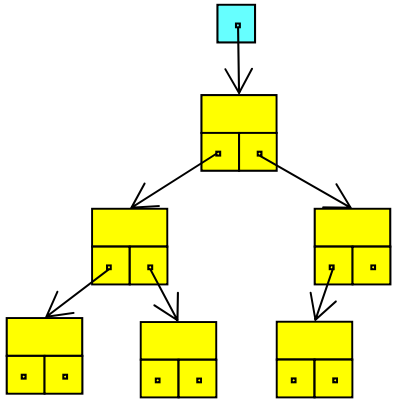
Árbol completo: Todos los nodos tienen todos los hijos que pueden tener

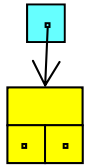


Altura (n)	No. nodos
1	1
2	3
3	7
4	15
5	31

$2^n - 1$

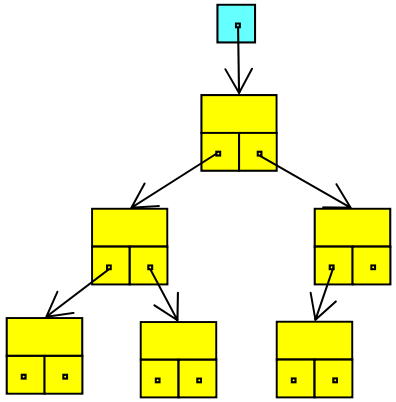

M
E
L
B
F
I
A
C
H
K
D
J

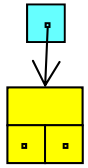




función: **inicializa**
recibe: *árbol*
regresa: nada

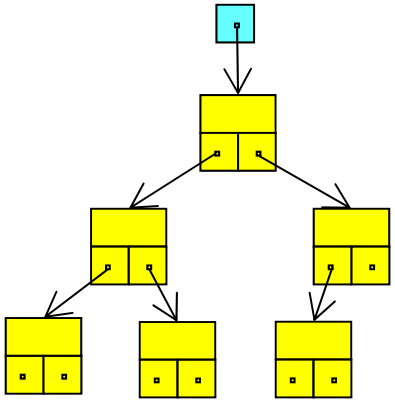
árbol = NULO

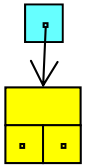




función: **vacío**
recibe: *árbol*
regresa: booleano

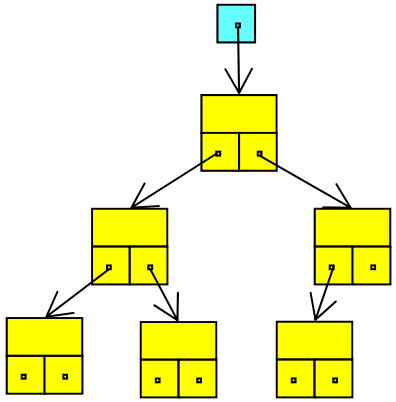
¿*árbol* = NULO?
Sí: devolver: *verdadero*
No: devolver: *falso*

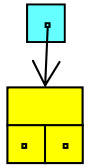




función: **recupera**
recibe: *pos*, *árbol*
regresa: elemento

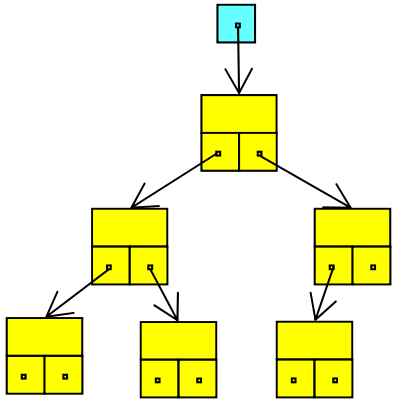
¿*árbol* = NULO ó *pos* = NULO?
Sí: ¡Error de excepción!
Insuficiencia de datos
Terminar
No: devolver: *pos*→elem

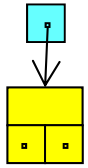




función: **inserta**
recibe: *elem*, *árbol*
regresa: nada

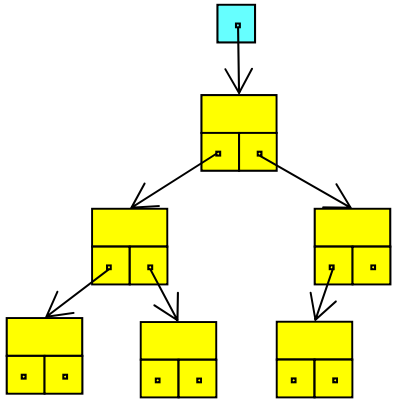
¿*árbol* = NULO?
Sí: aux = nuevo nodo
aux→elem = *elem*
árbol = aux
No: ¿*elem* < árbol→elem?
Sí: **inserta**(*árbol*→izq, *elem*)
No: **inserta**(*árbol*→der, *elem*)

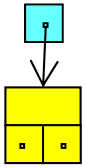




función: **localiza**
recibe: *elem*, *árbol*
regresa: posición

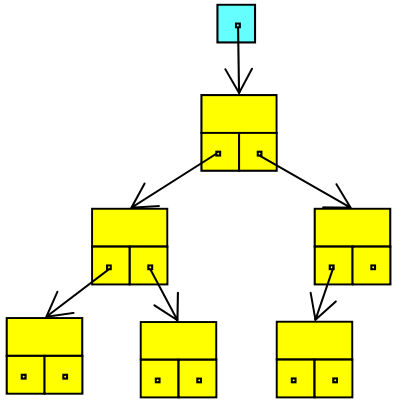
¿*árbol* = NULO?
Sí: devolver NULO
terminar
No: ¿*elem* = *árbol*→elem?
Sí: devolver: *árbol*
No: ¿*elem* < *árbol*→elem
Sí: devolver: **localiza**(*elem*, *árbol*→izq)
No: devolver: **localiza**(*elem*, *árbol*→der)

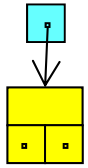




función: **menor**
recibe: *árbol*
regresa: posición

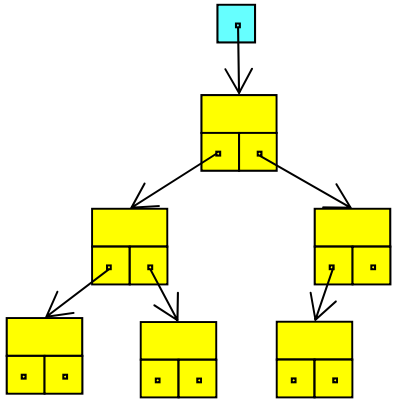
¿*árbol* = NULO?
Sí: devolver: NULO
terminar
No: ¿*árbol*→izq = NULO?
Sí: devolver: *árbol*
No: devolver: **menor**(*árbol*→izq)

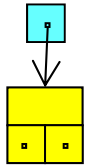




función: **mayor**
recibe: *árbol*
regresa: posición

¿*árbol* = NULO?
Sí: devolver: NULO
terminar
No: ¿*árbol*→der = NULO?
Sí: devolver: *árbol*
No: devolver: **mayor**(*árbol*→der)

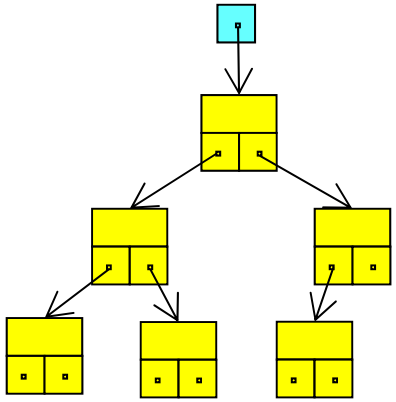


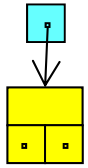


función: **es_hoja**
recibe: *árbol*
regresa: booleano

¿*árbol* = NULO?
Sí: devolver: *falso*

¿*árbol*→izq = NULO y *árbol*→der = NULO?
Sí: devolver: *verdadero*
No: devolver: *falso*





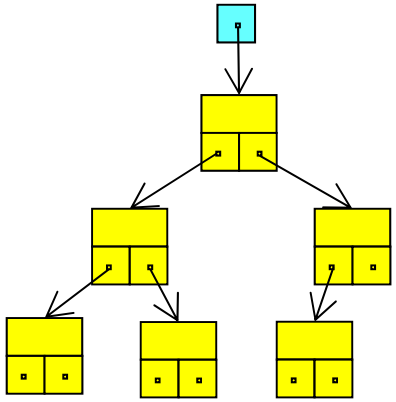
función: **recorrido_preorder**
recibe: *árbol*
regresa: nada

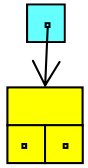
¿*árbol* = NULO?
Sí: terminar

imprimir: *árbol*→elem

recorrido_preorder(*árbol*→izq)

recorrido_preorder(*árbol*→der)





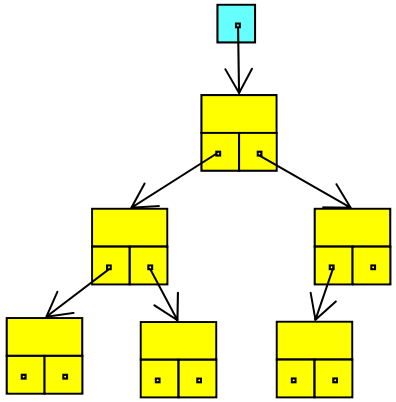
función: **recorrido_inorder**
recibe: *árbol*
regresa: nada

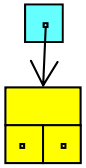
¿*árbol* = NULO?
Sí: terminar

recorrido_inorder(*árbol*→izq)

imprimir: *árbol*→elem

recorrido_inorder(*árbol*→der)





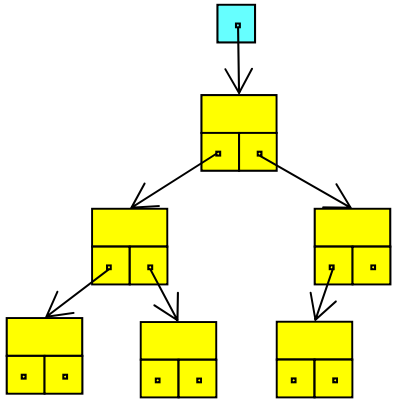
función: **recorrido_postorder**
recibe: *árbol*
regresa: nada

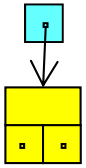
¿*árbol* = NULO?
Sí: terminar

recorrido_postorder(*árbol*→izq)

recorrido_postorder(*árbol*→der)

imprimir: *árbol*→elem

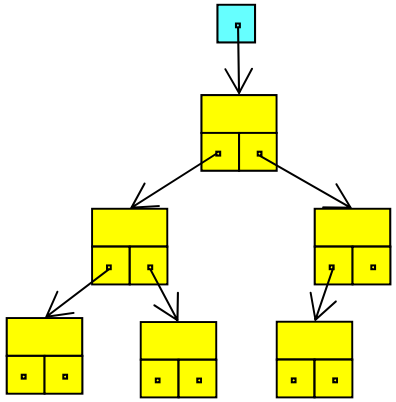


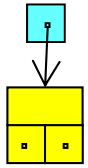


función: **elimina**
recibe: *pos*, *árbol*
regresa: nada

¿*árbol* = NULO ó *pos* = NULO?
Sí:Desplegar mensaje de error
Terminar

¿***eshoja***(*árbol*)?
Sí: liberar *árbol*
árbol = NULO
No:¿*árbol*→izq ≠ NULO?
Sí: pos_reemplazo = **mayor**(*árbol*→izq)
No: pos_reemplazo = **menor**(*árbol*→der)
árbol→elem = pos_reemplazo→elem
elimina(pos_reemplazo, *árbol*)

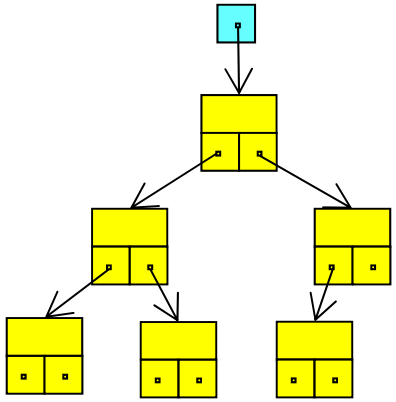


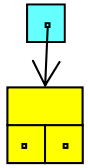


función: **altura**
recibe: *árbol*
regresa: entero

¿*árbol* = NULO?
Sí: devolver: 0
No: alt_izq = **altura**(*árbol*→izq)
alt_der = **altura**(*árbol*→der)

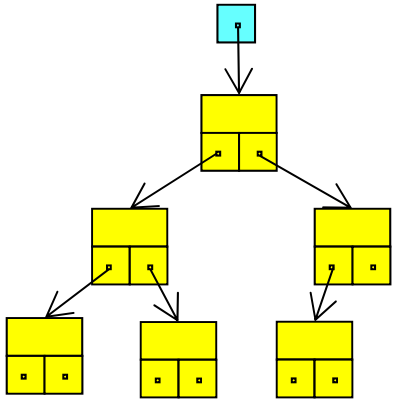
¿alt_izq > alt_der?
Sí: devolver: alt_izq + 1
No: devolver: alt_der + 1

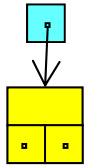




función: **fact_eq**
recibe: *árbol*
regresa: entero

¿*árbol* = NULO?
Sí: devolver: 0
No: devolver: **altura**(*árbol*→der) – **altura**(*árbol*→izq)

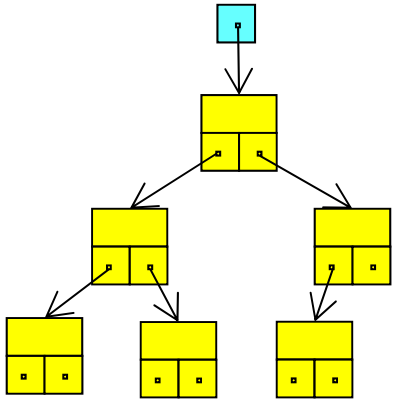


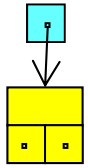


función: **balanceo**
recibe: *árbol*
regresa: nada

¿**fact_eq**(*árbol*) < –1?
    Sí: ¿**fact_eq**(*árbol* →izq) = –1?
        Sí: **rot_sim_der**(*árbol*)
        No: **rot_dob_der**(*árbol*)
    Terminar

¿**fact_eq**(*árbol*) > 1?
    Sí: ¿**fact_eq**(*árbol* →izq) = 1?
        Sí: **rot_sim_izq**(*árbol*)
        No: **rot_dob_izq**(*árbol*)
    Terminar

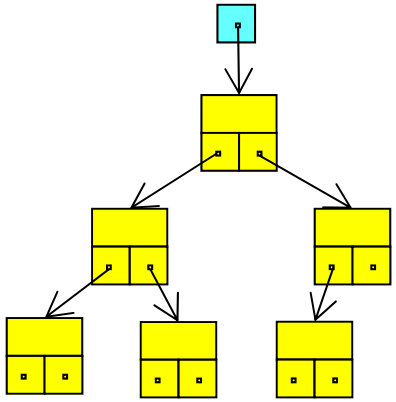


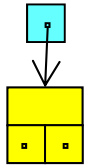


función: **balanceo**
recibe: *árbol*
regresa: nada

¿**fact_eq**(*árbol*) < –1?
    Sí: ¿**fact_eq**(*árbol* →izq) = –1?
        Sí: **rot_sim_der**(*árbol*)
        No: **rot_dob_der**(*árbol*)
    Terminar

¿**fact_eq**(*árbol*) > 1?
    Sí: ¿**fact_eq**(*árbol* →der) = 1?
        Sí: **rot_sim_izq**(*árbol*)
        No: **rot_dob_izq**(*árbol*)
    Terminar

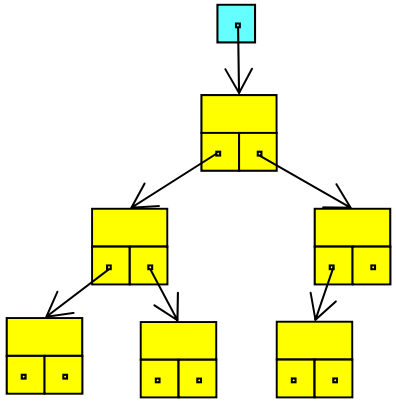


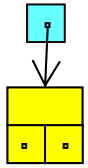


función: **rot_sim_der**
recibe: *árbol*
regresa: nada

aux1 = *árbol*→izq
aux2 = *árbol*→izq→der

árbol→izq = aux2
aux1→der = *árbol*
árbol = aux1

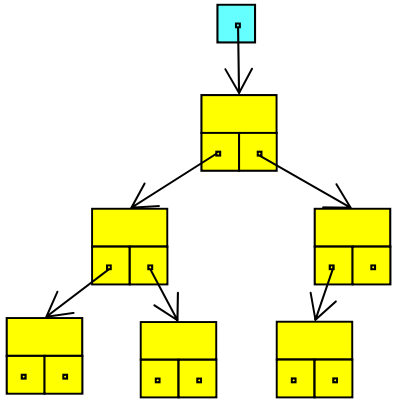


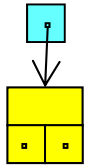


función: **rot_sim_izq**
recibe: *árbol*
regresa: nada

aux1 = *árbol*→der
aux2 = *árbol*→der→izq

árbol→der = aux2
aux1→izq = *árbol*
árbol = aux1

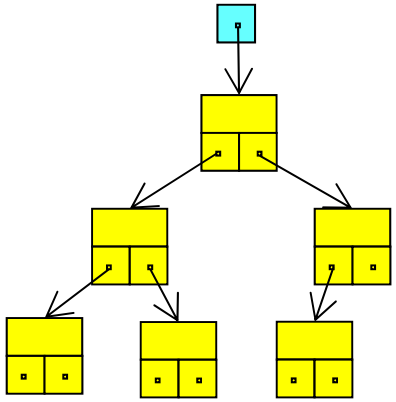


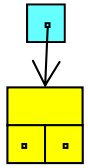


función: **rot_dbl_der**
recibe: *árbol*
regresa: nada

aux1 = *árbol*→izq
aux2 = aux1→der
aux3 = aux2→izq
aux4 = aux2→der

árbol→izq = aux4
aux1→der = aux3
aux2→izq = aux1
árbol = aux2





función: **rot_dbl_izq**
recibe: *árbol*
regresa: nada

aux1 = *árbol*→der
aux2 = aux1→izq
aux3 = aux2→der
aux4 = aux2→izq

árbol→der = aux4
aux1→izq = aux3
aux2→der = aux1
árbol = aux2