

Árboles binarios

Franco Guidi Polanco

Escuela de Ingeniería Industrial
Pontificia Universidad Católica de Valparaíso, Chile
fguidi@ucv.cl

Actualización: 13 de mayo de 2005

Árbol: definición

❖ Árbol (del latín *arbor* –*oris*):

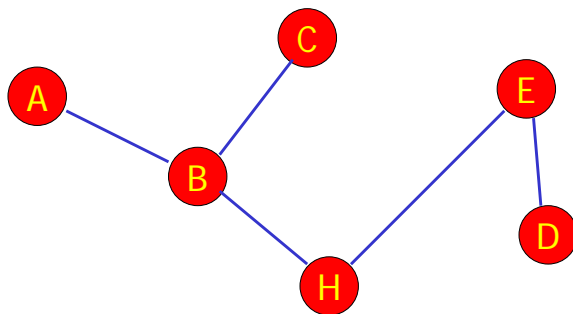
- Planta perenne, de tronco leñoso y elevado, que se ramifica a cierta altura del suelo.
- (otras, ver Real Academia Española...)



Árbol: definición (cont.)

❖ Árbol:

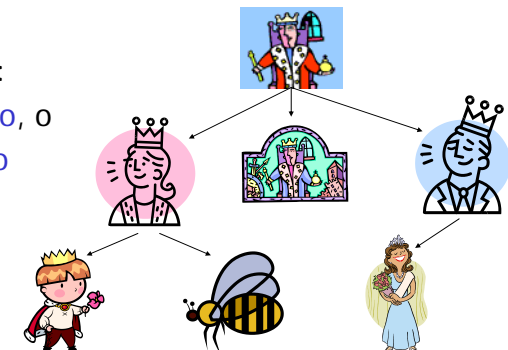
- Grafo conexo, no orientado y acíclico.



Árbol: definición (cont.)

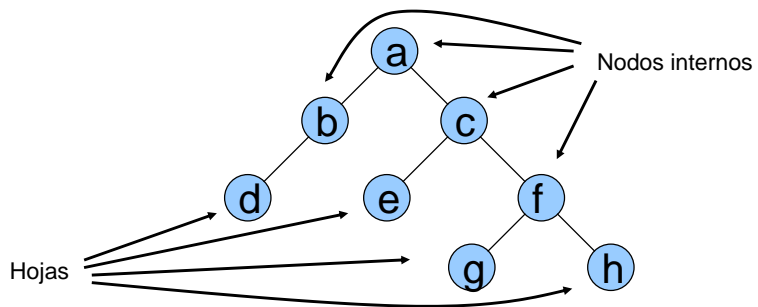
❖ Árbol:

- Una estructura de datos accesada desde un **nodo raíz**. Cada nodo es ya sea una **hoja** o un **nodo interno**. Un **nodo interno** tiene uno o más **nodos hijos**, y se le llama **padre** de sus **nodos hijos**.
- Un **árbol** es, ya sea:
 - un **conjunto vacío**, o
 - una **raíz** con **cero** o **más árboles**

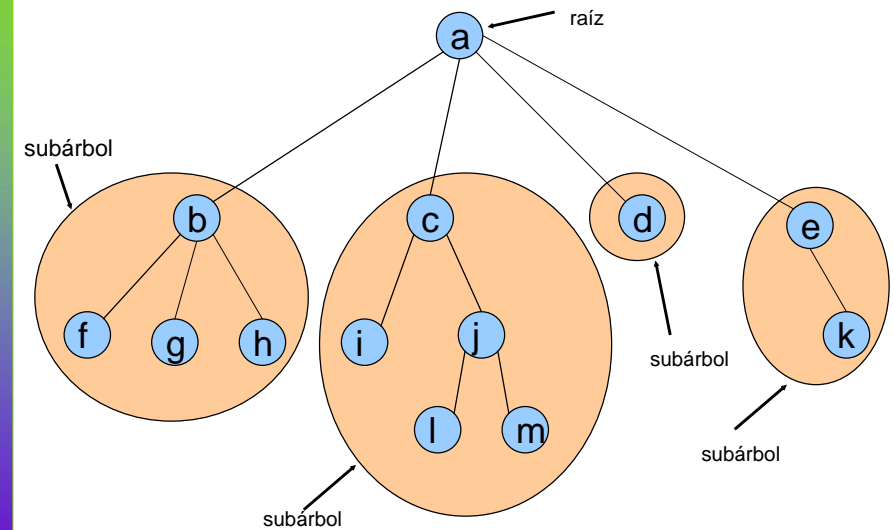


Hojas y nodos internos

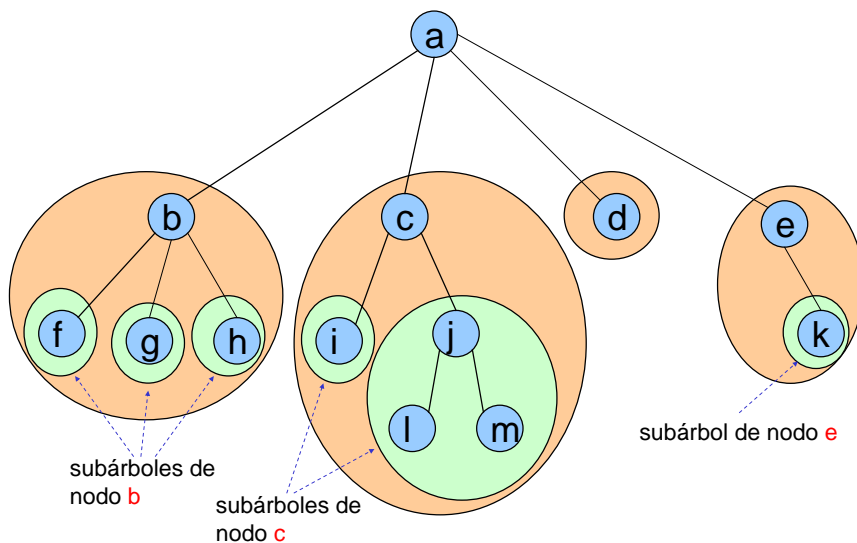
- ❖ Una **hoja** es cualquier nodo que tiene sus hijos vacíos.
- ❖ Un **nodo interno** es cualquier nodo con al menos un hijo no vacío.



Representación de un árbol



Representación de un árbol (cont.)

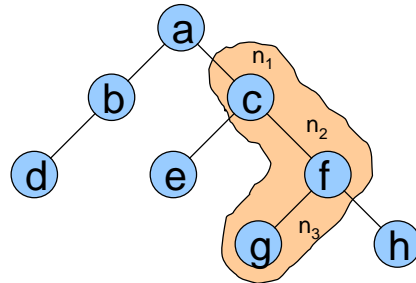


Nodos padres e hijos

- ❖ Las raíces de los subárboles de un árbol son **hijos** de la raíz del árbol.
- ❖ Existe un arco desde cada nodo a cada uno de sus hijos, y se dice que este nodo es padre de sus hijos.

Ruta y largo de una ruta

- ❖ Si n_1, n_2, \dots, n_k es una secuencia de nodos en un árbol, de modo que n_i es padre de n_{i+1} , para $1 \leq i \leq k$, entonces esta secuencia se llama **ruta** desde n_1 a n_k .
- ❖ El **largo** de esta ruta es k .

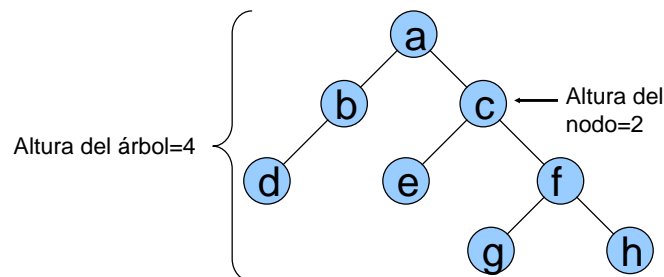


Ancestros y descendientes

- ❖ Si existe una ruta desde un nodo A a un nodo B, entonces A es **ancestro** de B y B es **descendiente** de A.
- ❖ Luego, todos los nodos de un árbol son descendientes de la raíz del árbol, mientras que la raíz es el ancestro de todos los nodos.

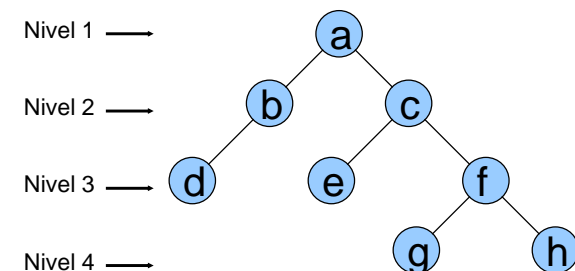
Altura

- ❖ La **altura de un nodo** M de un árbol corresponde al número de nodos en la ruta desde la raíz hasta M .
- ❖ La **altura de un árbol** corresponde a la altura del nodo más profundo.



Niveles

- ❖ Todos los nodos de altura d están en el nivel d en el árbol.
- ❖ La raíz está en el nivel 1, y su altura es 1.

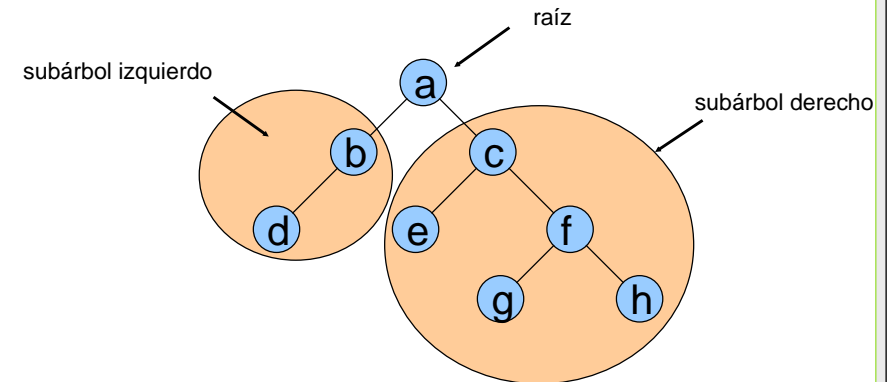


Árboles binarios

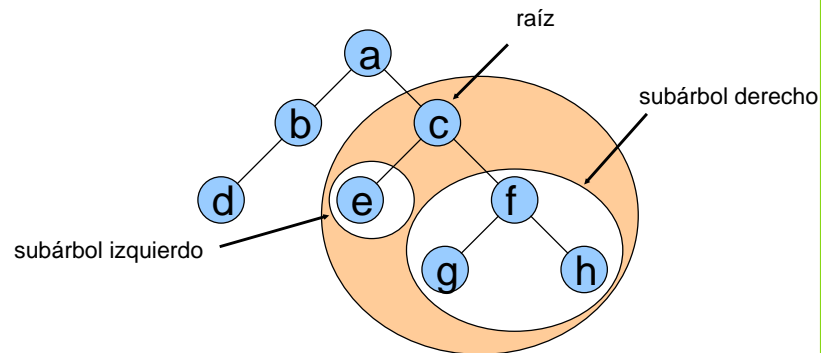
- ❖ Un A.B. está constituido por un conjunto finito de elementos llamados **nodos**.
- ❖ Un árbol binario:
 - no tiene nodos (está vacío); o
 - tiene un nodo llamado raíz, junto con otros **dos** árboles binarios llamados **subárboles derecho** e **izquierdo** de la raíz.

Nota: Una parte importante del material presentado en esta sección fue elaborado por Marcelo Silva F.

Representación de un árbol binario (I)

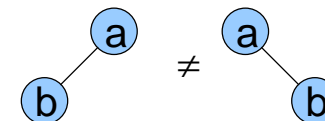


Representación de un árbol binario (II)



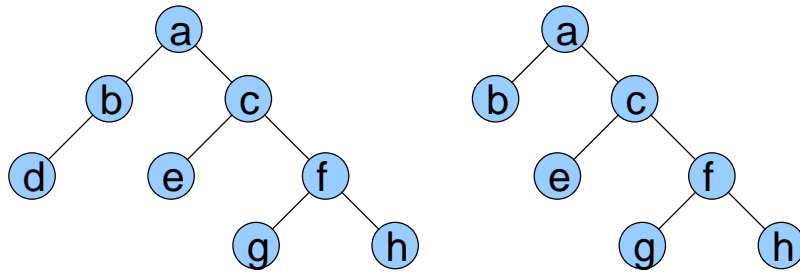
Igualdad de árboles binarios

- ❖ Para ser iguales, dos árboles deben tener tanto la misma estructura, como el mismo contenido.



Árboles binarios llenos

- ❖ Un árbol binario lleno es aquel en que cada nodo es un nodo interno con dos hijos no vacíos, o una hoja.



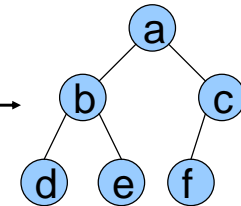
No es A.B. lleno

Es A.B. lleno

Árboles binarios completos

- ❖ Un árbol binario completo tiene una forma restringida, que se obtiene al ser llenado de izquierda a derecha. En un A.B. Completo de altura d , todos los niveles, excepto posiblemente el nivel d están completamente llenos.

Es un A.B. completo
pero no es un A.B. lleno



Número de nodos en un árbol binario

- ❖ El máximo número de nodos en el nivel i de un árbol binario es $2^{(i-1)}$.
- ❖ El máximo número de nodos en un árbol binario de altura K es $2^{(K)}-1$.

Representación de árboles binarios mediante nodos y referencias

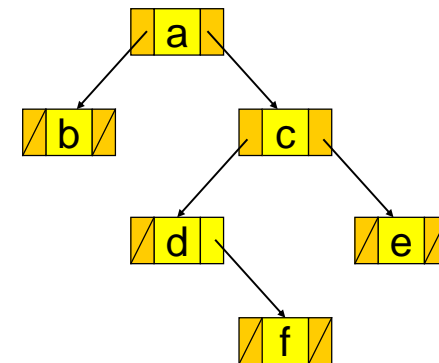


Diagrama de clases de un árbol binario

❖ Diagrama de clases árbol binario de enteros:

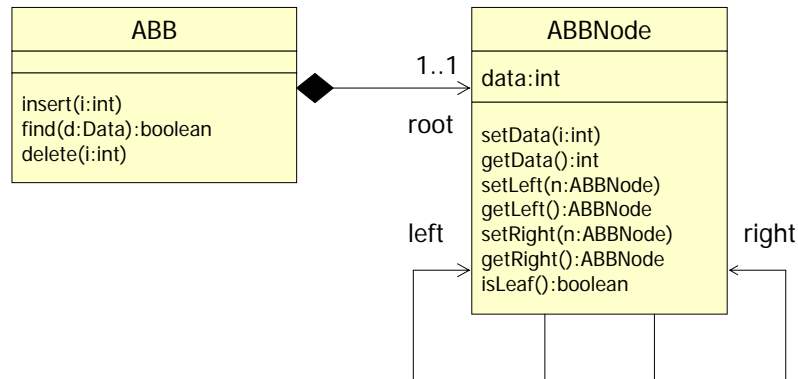
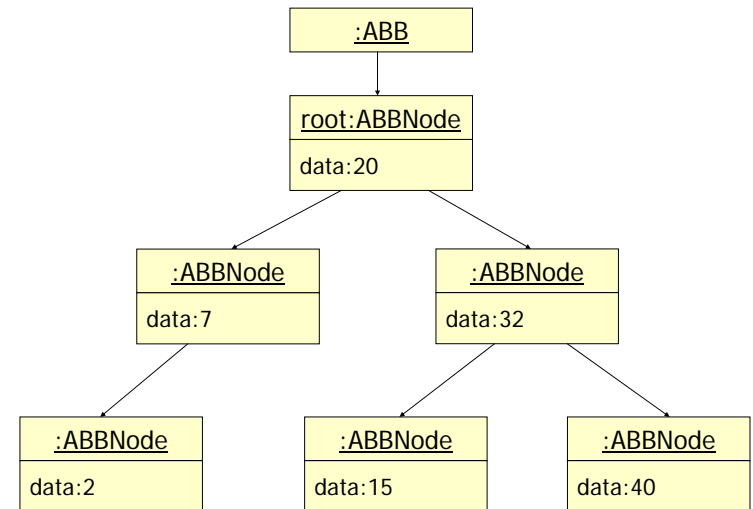
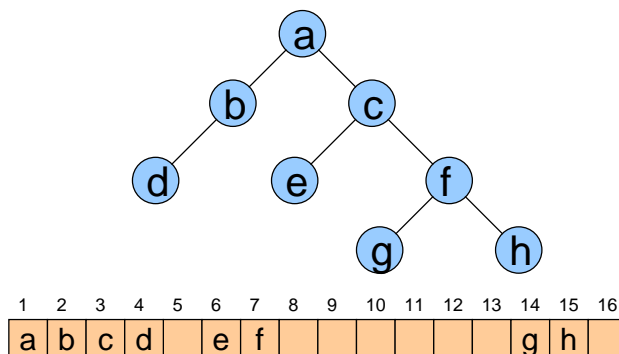


Diagrama de objetos de un árbol binario



Representación de árboles binarios mediante arreglos

- ❖ Si la raíz de un subárbol se almacena en $A[i]$, su hijo izquierdo se almacena en $A[2*i]$, y su hijo derecho en $A[2*i+1]$.



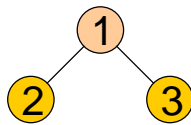
Recorrido de árboles binarios

- ❖ Un **recorrido** es cualquier proceso destinado a visitar los nodos de un árbol binario en un determinado orden.
- ❖ Cualquier recorrido que visite cada nodo exactamente una vez, se denomina una **enumeración** de los nodos del árbol.
- ❖ Recorridos de enumeración a analizar:
- Preorden
 - Inorden
 - Postorden

Recorrido en Preorden

Dado un árbol binario:

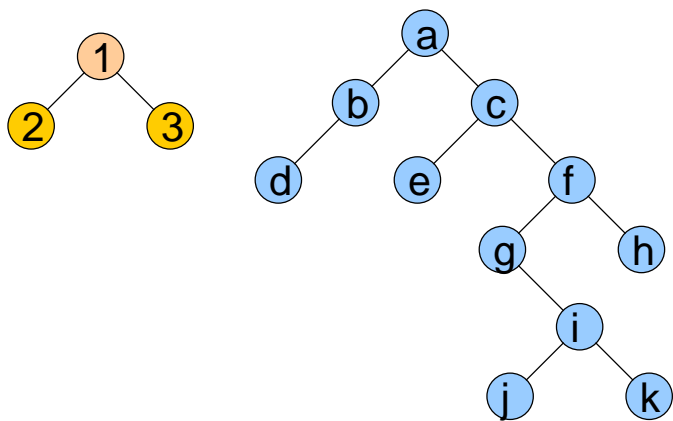
- 1) Visitar su raíz.
- 2) Recorrer en **preorden** su subárbol izquierdo.
- 3) Recorrer en **preorden** su subárbol derecho.



Código para recorrido Preorden

```
void preorder(BinNode rt) // rt es la raíz del subarbol
{
    if (rt==null)
        return; // subarbol vacío
    visit(rt) // hace algo con el nodo
    preorder(rt.left());
    preorder(rt.right());
}
```

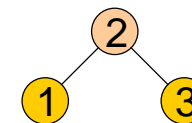
Ejemplo de recorrido en Preorden



Recorrido en Inorden

Dado un árbol binario:

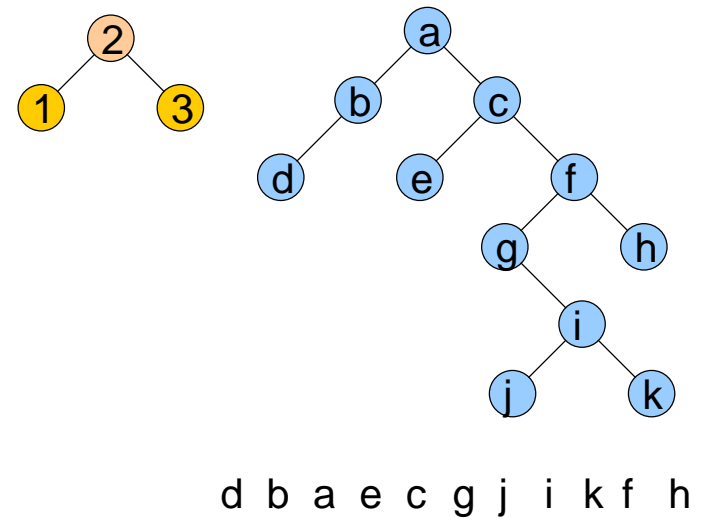
- 1) Recorrer en **inorden** su subárbol izquierdo.
- 2) Visitar su raíz.
- 3) Recorrer en **inorden** su subárbol derecho.



Código para recorrido Inorden

```
void inorder(BinNode rt) // rt es la raíz del subarbol
{
    if (rt==null)
        return;           // subarbol vacío
    inorder(rt.left());
    visit(rt)              // hace algo con el nodo
    inorder(rt.right());
}
```

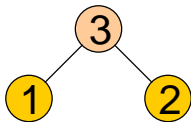
Ejemplo de recorrido en Inorden



Recorrido en Postorden

Dado un árbol binario:

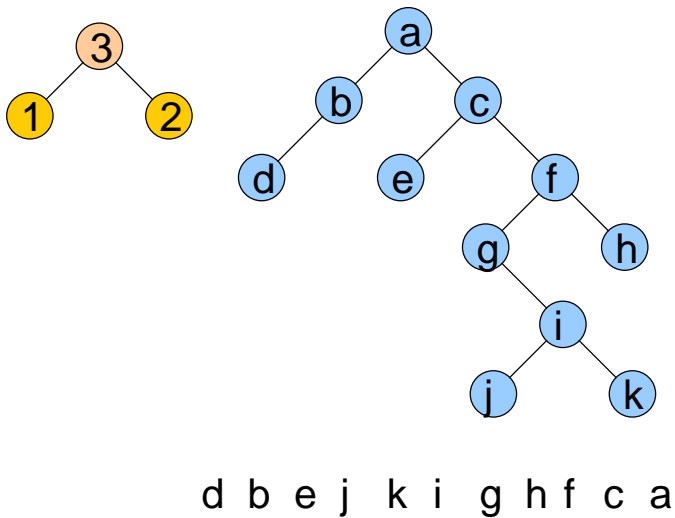
- 1) Recorrer en **postorden** su subárbol izquierdo.
- 2) Recorrer en **postorden** su subárbol derecho.
- 3) Visitar su raíz.



Código para recorrido Postorden

```
void postorder(BinNode rt) // rt es la raíz del subarbol
{
    if (rt==null)
        return; // subarbol vacío
    postorder(rt.left());
    postorder(rt.right());
    visit(rt) // hace algo con el nodo
}
```

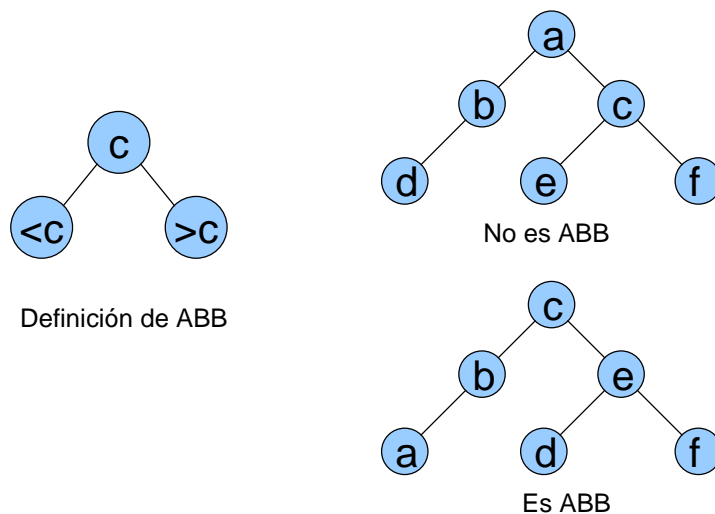

Ejemplo de recorrido en Postorden



Árbol binario de búsqueda

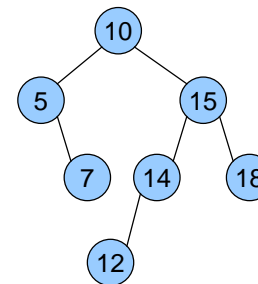
- ❖ Supongamos que tenemos un conjunto de n elementos que pueden ser ordenados por alguna clave.
- ❖ En un **árbol binario de búsqueda (ABB)**, todos los nodos almacenados en el subárbol izquierdo de un nodo cuyo valor clave es C tienen claves menores que C , mientras que todos los nodos ubicados en el subárbol derecho tienen claves mayores que C .

Ejemplos de árboles binarios de búsqueda



Ingreso de elementos a un ABB

{ 10, 5, 7, 15, 14, 12, 18 }



{ 15, 18, 14, 5, 10, 12, 7 }

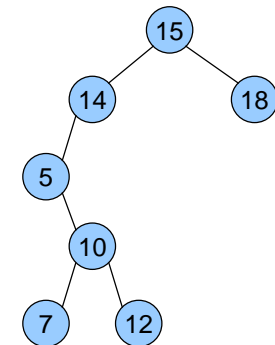
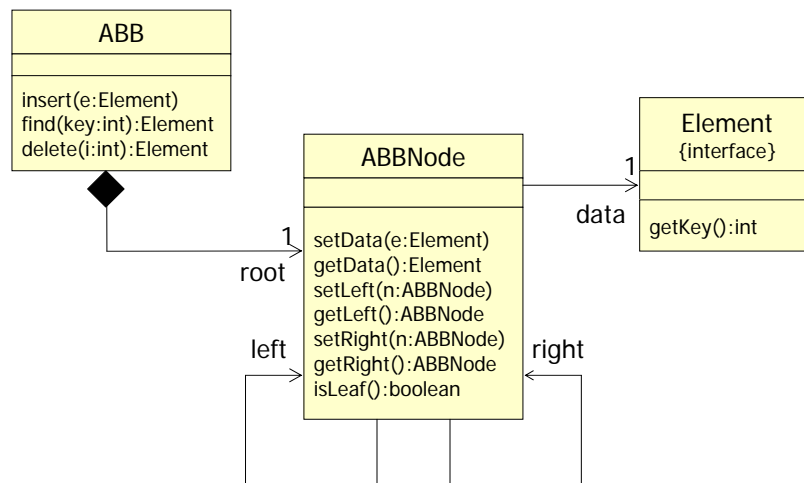


ABB de referencia



Ingreso de elementos a un ABB (cont.)

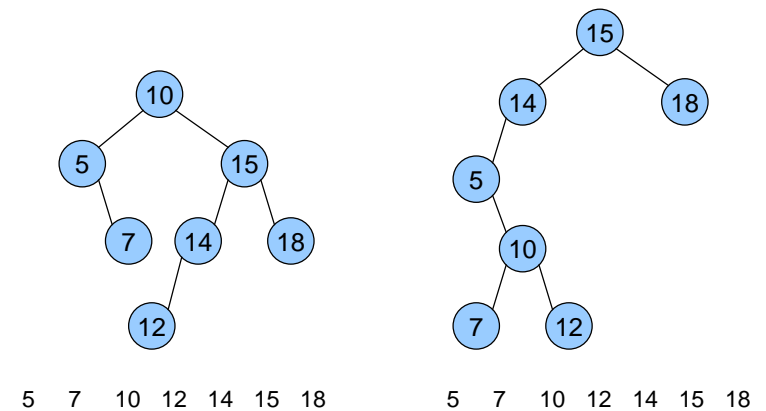
```

private BinNode insert (BinNode rt, Element val)
{
    if (rt == null)
        return new BinNode(val);
    Element it = (Element)rt.element();
    if (it.key() > val.key())
        rt.setLeft(insert(rt.left(), val));
    else
        rt.setRight(insert(rt.right(), val));
    return rt;
}
    
```

Características del ingreso de elementos a un ABB

- ❖ Los elementos agregados a un ABB siempre son incorporados inicialmente como **hojas**.
- ❖ Un conjunto de elementos dado puede generar diversos ABB, dependiendo del orden en que son ingresados.

Recorrido Inorden en ABB



Características del recorrido Inorden de un ABB

- ❖ Si bien existen muchos ABBs posibles para un mismo conjunto de elementos, el recorrido Inorden de todos estos árboles siempre entrega el conjunto ordenado de menor a mayor.

Búsqueda en ABB

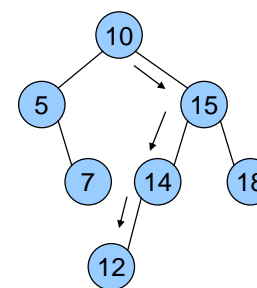
Para hallar un elemento con clave C , en un árbol A :

- ❖ Si la raíz del árbol A almacena C , la búsqueda termina exitosamente.
- ❖ Si C es menor que el valor de la raíz de A , buscar en el subárbol izquierdo. Si C es mayor que el valor de la raíz, buscar en el subárbol derecho.
- ❖ La búsqueda termina al hallar el valor C , o al pretender buscar en un subárbol vacío.

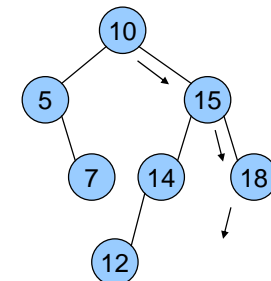
Búsqueda en ABB (cont.)

```
Elem find(BinNode rt, int key) {  
    if (rt == null)  
        return null;  
    Element it = (Element)rt.element();  
    if ((int)it.key() > key)  
        return find(rt.left(), key);  
    else if (it.key() == key)  
        return it;  
    else  
        return find(rt.right(), key);  
}
```

Ejemplo de búsqueda en ABB



Buscar 12
Búsqueda exitosa



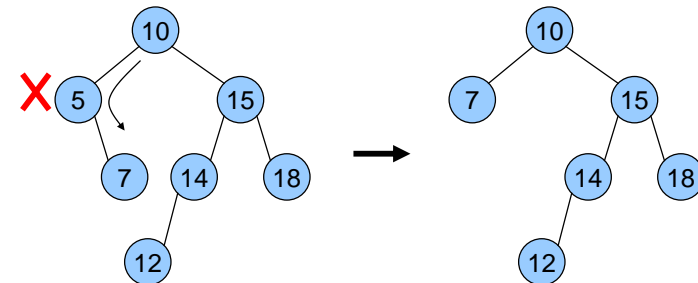
Buscar 16
Búsqueda infructuosa

Eliminación de elementos de un ABB

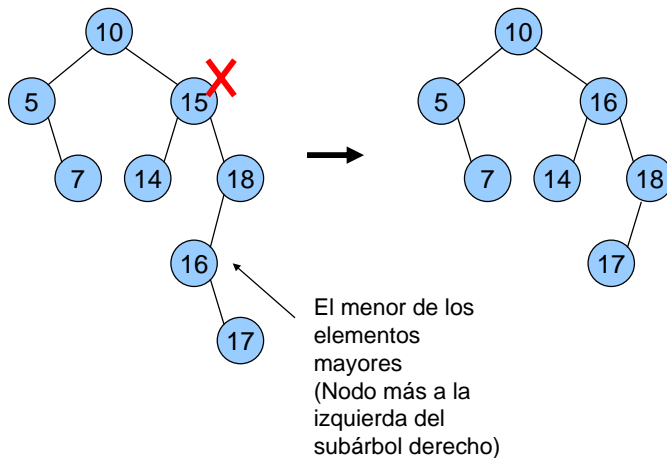
Se pueden presentar tres casos:

- ❖ El elemento no existe.
- ❖ El elemento es una hoja o tiene a lo más un hijo.
- ❖ El elemento tiene dos hijos.

Eliminar nodo que es una hoja o tiene a lo más un hijo



Ejemplo eliminación de nodo con dos hijos



Eliminar nodo con dos hijos

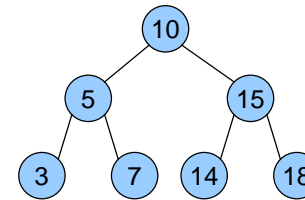
1. Hallar el nodo que contiene el menor de los elementos mayores del nodo a eliminar (el elemento más a la izquierda de su subárbol derecho)
2. Reemplazar los datos del nodo eliminar con los del nodo hallado.
3. Eliminar el nodo hallado, que tiene a lo más un hijo, con el procedimiento descrito previamente.

Utilidad de los árboles binarios de búsqueda

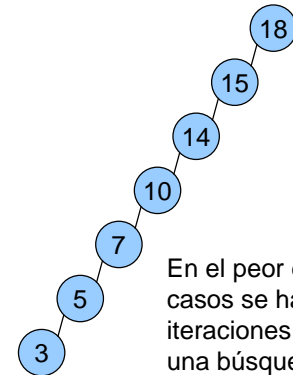
- ❖ Al buscar, el ABB permite descartar a priori un subconjunto de elementos, en forma análoga a la búsqueda binaria en arreglos ordenados.
- ❖ El ABB presenta además la ventaja de poder ser implementado con punteros (estructura dinámica).
- ❖ La incorporación y eliminación de elementos al ABB es mas rápida que en arreglos ordenados.

Importancia de una estructura balanceada en los ABB

- ❖ La estructura de un ABB es importante al momento de realizar búsquedas en él.



En el peor de los casos se hacen 3 iteraciones para una búsqueda.



En el peor de los casos se hacen 7 iteraciones para una búsqueda.