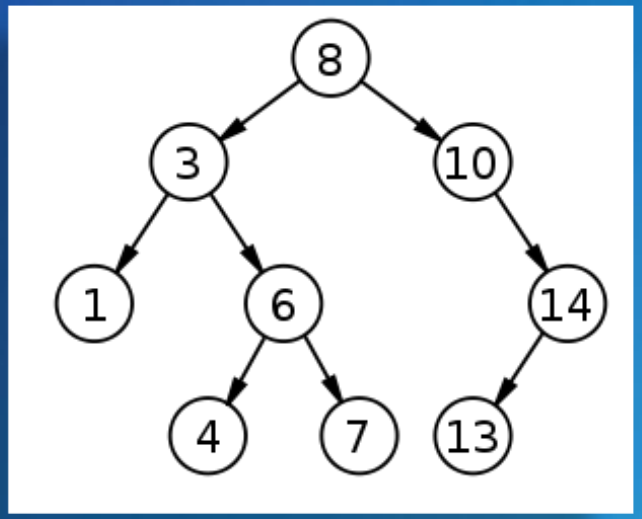
Universidad Rafael Landívar
Tradición Jesuita en Guatemala

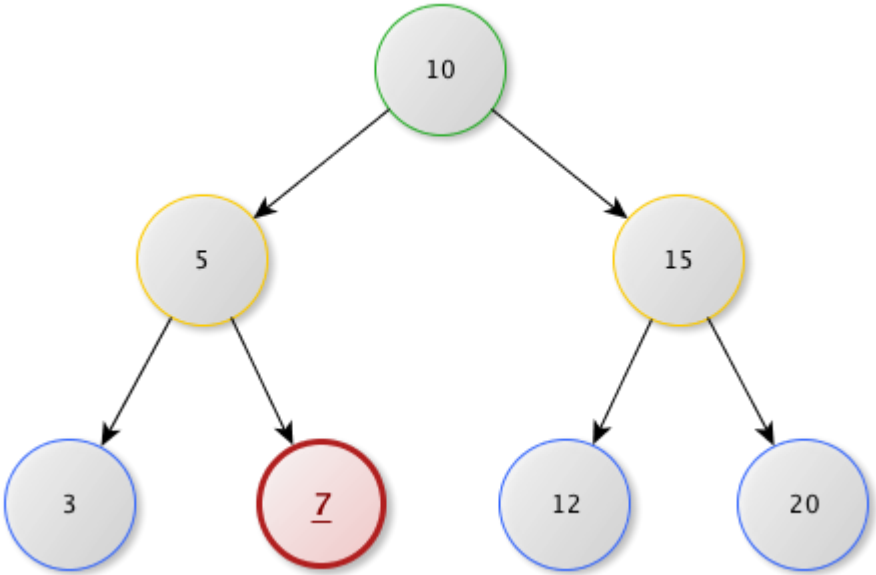
Árboles binarios de búsqueda



Facultad de ingeniería

Árbol binario

- Se definen como árboles de grado 2. Esto es, cada nodo puede tener dos, uno o ningún hijo. Al tratarse como mucho de dos hijos, cada uno de ellos puede identificarse como hijo izquierdo o hijo derecho.



La estructura de un árbol binario

- Se construye con nodos. Cada nodo debe contener el campo dato (datos a almacenar) y dos campos de tipo puntero, uno al subárbol izquierdo y otro al subárbol derecho, que se conocen como puntero izquierdo y puntero derecho respectivamente.

- Un valor NULL indica un árbol o un subárbol vacío

Búsqueda:

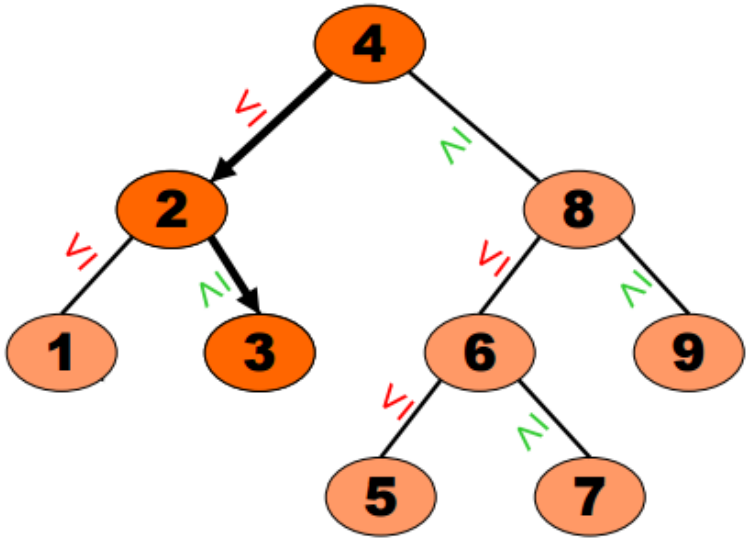
- Un árbol binario de búsqueda es un árbol binario en el que para cada nodo n, todas las claves de los nodos del subárbol izquierdo son **menores** que la clave de n (o iguales) y todas las del subárbol derecho **mayores** (o iguales).

- Buscamos el “3”:

✓ 3<4: Subárbol Izquierdo

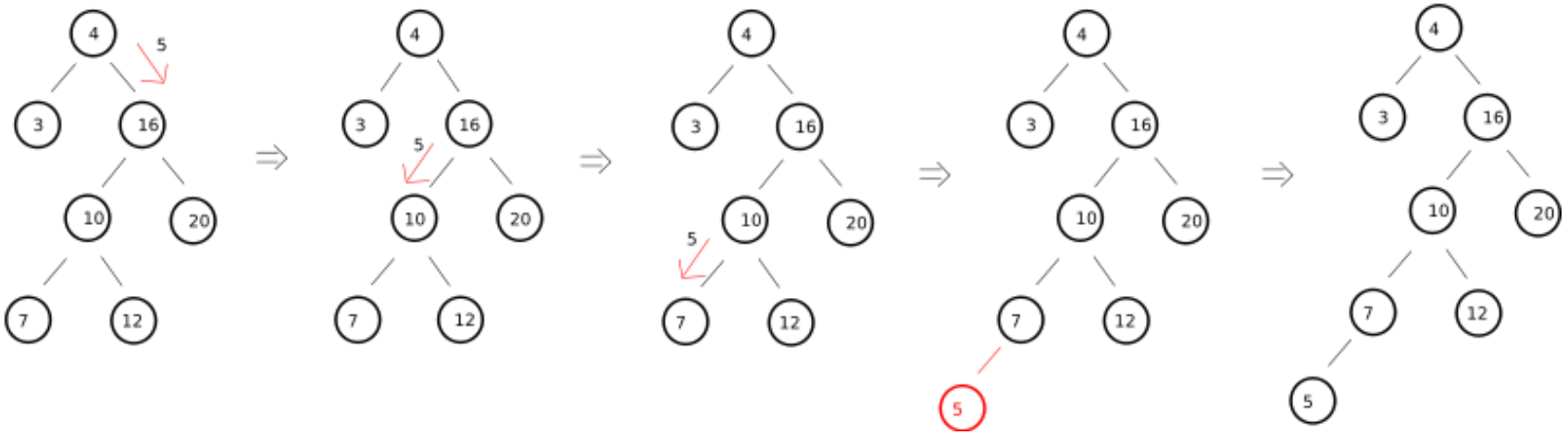
✓ 3>2: Subárbol derecho

✓ 3=3: Elemento encontrado



Inserción:

- La inserción es similar a la búsqueda y se puede dar una solución tanto iterativa como recursiva. Si tenemos inicialmente como parámetro un árbol vacío se crea un nuevo nodo como único contenido el elemento a insertar. Si no lo está, se comprueba si el elemento dado es menor que la raíz del árbol inicial con lo que se inserta en el subárbol izquierdo y si es mayor se inserta en el subárbol derecho.



Búsqueda:

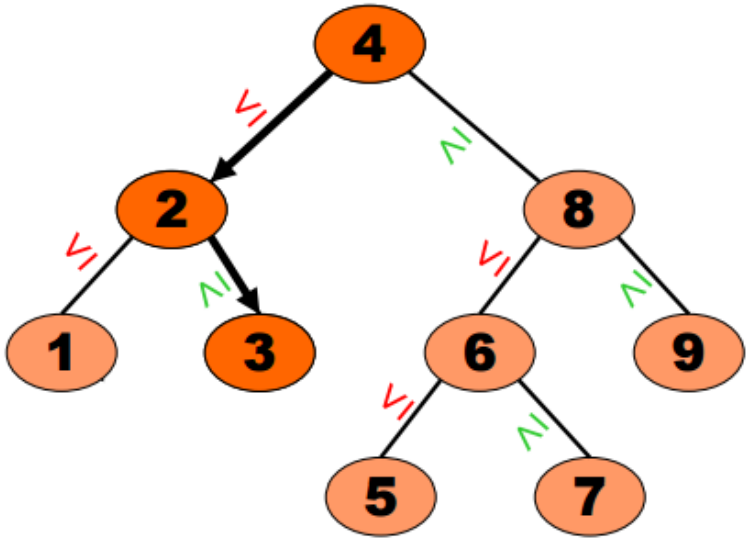
- Un árbol binario de búsqueda es un árbol binario en el que para cada nodo n, todas las claves de los nodos del subárbol izquierdo son **menores** que la clave de n (o iguales) y todas las del subárbol derecho **mayores** (o iguales).

- Buscamos el “3”:

✓ 3<4: Subárbol Izquierdo

✓ 3>2: Subárbol derecho

✓ 3=3: Elemento encontrado



Recorrido en árboles binarios

- Una vez construido un árbol binario, surge la necesidad de recorrerlo, es decir, una manera sistemática de visitar cada nodo del árbol.
- La forma en la cual una lista lineal se recorre es trivial (del primero al último, o viceversa). Sin embargo, para recorrer un árbol, esta forma natural no puede aplicarse.
- Para recorrer un árbol, existen varias formas de lograrlo, las 3 más comunes son *preorden*, *in-orden* y *post-orden*. Estos métodos difieren en el orden en el cual los nodos son visitados. Siguiendo la costumbre de empezar a visitar antes lo que se encuentra a la izquierda que lo de la derecha, se tienen 3 posibilidades de visitar a la raíz: antes, en medio o después.

- Pre-orden o previo:

1. Visitar La Raíz.
2. Recorrer Recursivamente El Subárbol Izquierdo.
3. Recorrer recursivamente el subárbol derecho.

- In-orden o simétrico:

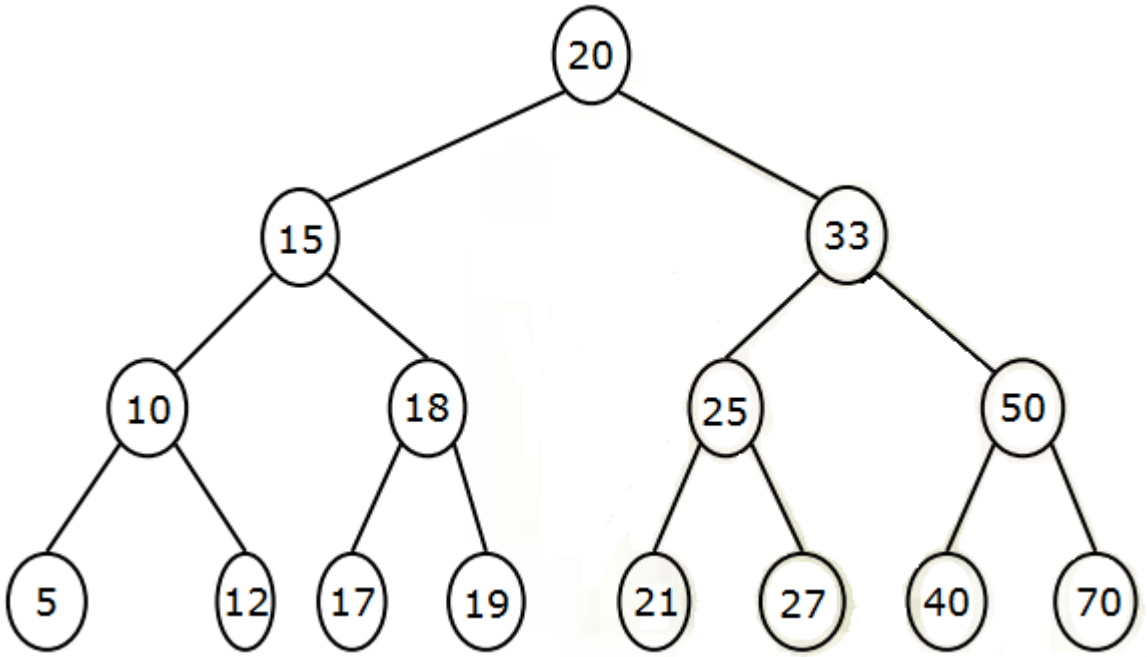
1. Recorrer Recursivamente El Subárbol Izquierdo.
2. Visitar La Raíz.
3. Recorrer recursivamente el subárbol derecho.

- Post-orden o posterior:

1. Recorrer Recursivamente El Subárbol Izquierdo.
2. Recorrer recursivamente el subárbol derecho.
3. Visitar la raíz.

- El resultado que se obtiene al recorrer el árbol:

- Pre-orden: 20, 15 10, 5, 12, 18, 17, 19, 33, 25, 21, 27, 50, 40, 70.
- In-orden: 5, 10, 12, 15, 17, 18, 19, 20, 21, 25, 27, 33, 40, 50, 70.
- Post-orden: 5, 12, 10, 17, 19, 18, 15, 21, 27, 25, 40, 70, 50, 33, 20.



Recorrido Preorden

```
public void preorden() //realiza el recorrido en preorden de un arbol
{
    metodoPreOrden(this.raiz);
}

//meoto recursivo para recorrido en preorden

private void metodoPreOrden(Nodo nodo)
{
    if(nodo != null)
    {
        System.out.print(nodo.valor+ " ");
        metodoPreOrden(nodo.hijoIzquierdo);
        metodoPreOrden(nodo.hijoDerecho);
    }
}
```

Postorden

```
//EMPEZAR RECORRIDO POSTRDEN
public void Posorden()
{
    metodoPosorden(raiz);
}

//meotod recursivo para recorrido posorden
private void metodoPosorden(Nodo nodo)
{
    if( nodo == null )
        return;

    metodoPosorden(nodo.hijoIzquierdo);
    metodoPosorden(nodo.hijoDerecho);
    System.out.print(nodo.valor + " ");
}
```

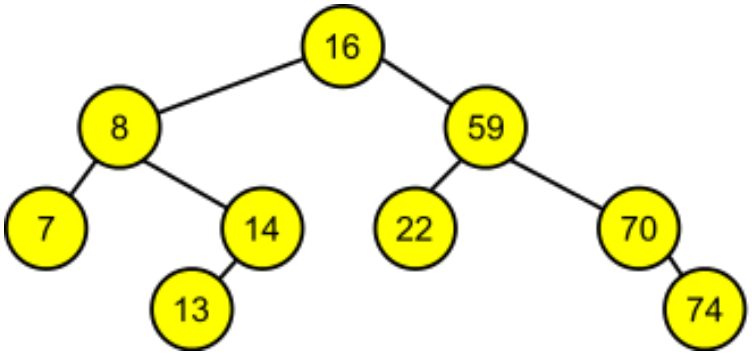
Inorden

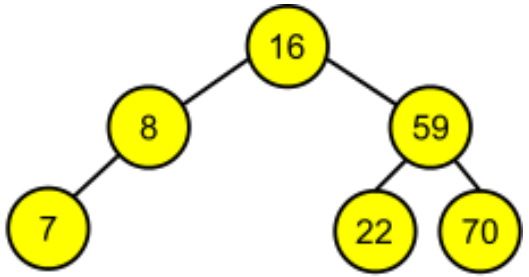
```
//EMPEZAR RECORRIDO INORDEN
public void Inorden()
{
    metodoInorden(this.raiz);
}

//metodo recursivo para recorrido inorden
private void metodoInorden( Nodo nodo)
{
    if(nodo == null)
        return;

    metodoInorden(nodo.hijoIzquierdo);
    System.out.print(nodo.valor + " ");
    metodoInorden(nodo.hijoDerecho);
}
```

Eliminación:

- La operación de borrado no es tan sencilla como las de búsqueda e inserción. Existen varios casos a tener en consideración:
- **Borrar un nodo sin hijos o nodo hoja**: simplemente se borra y se establece a nulo el apuntador de su padre.

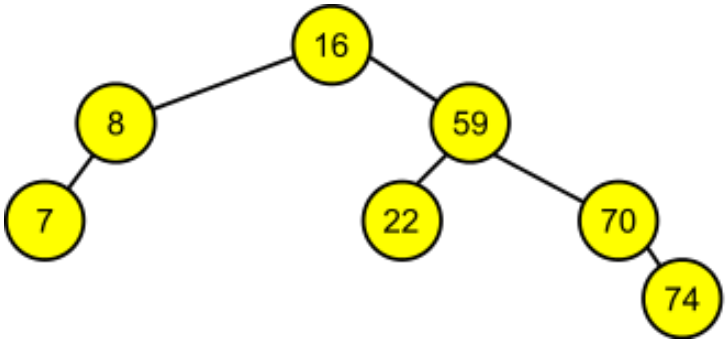
Nodo a eliminar 74

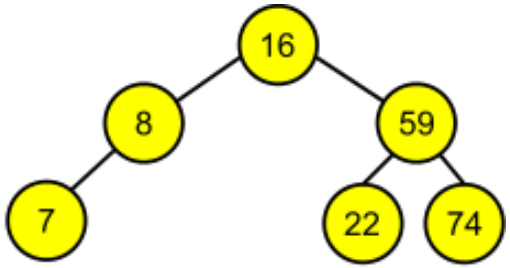




- **Borrar un nodo con un subárbol hijo**: se borra el nodo y se asigna su subárbol hijo como subárbol de su padre.

Nodo a eliminar 70





- **Borrar un nodo con dos subárboles hijo**: la solución está en reemplazar el valor del nodo por el de su predecesor o por el de su sucesor en inorden y posteriormente borrar este nodo. Su predecesor en inorden será el nodo más a la derecha de su subárbol izquierdo (mayor nodo del subárbol izquierdo), y su sucesor el nodo más a la izquierda de su subárbol derecho (menor nodo del subárbol derecho). En la siguiente figura se muestra cómo existe la posibilidad de realizar cualquiera de ambos reemplazos:

- Nodo a eliminar 59