Estructura de Datos y Algoritmos 1

Teórico #10:
Árboles Binarios de Búsqueda

Introducción

La búsqueda secuencial en una lista tiene complejidad O(n)

Si la lista está ordenada, el desempeño de la búsqueda secuencial es mejor, pero sigue siendo en el peor de los casos un O(n)

La búsqueda binaria en un lista basada en *arrays* es un O(log n). Sin embargo, las operaciones de inserción y eliminación siguen siendo un O(n) debido a la necesidad del corrimiento de la información

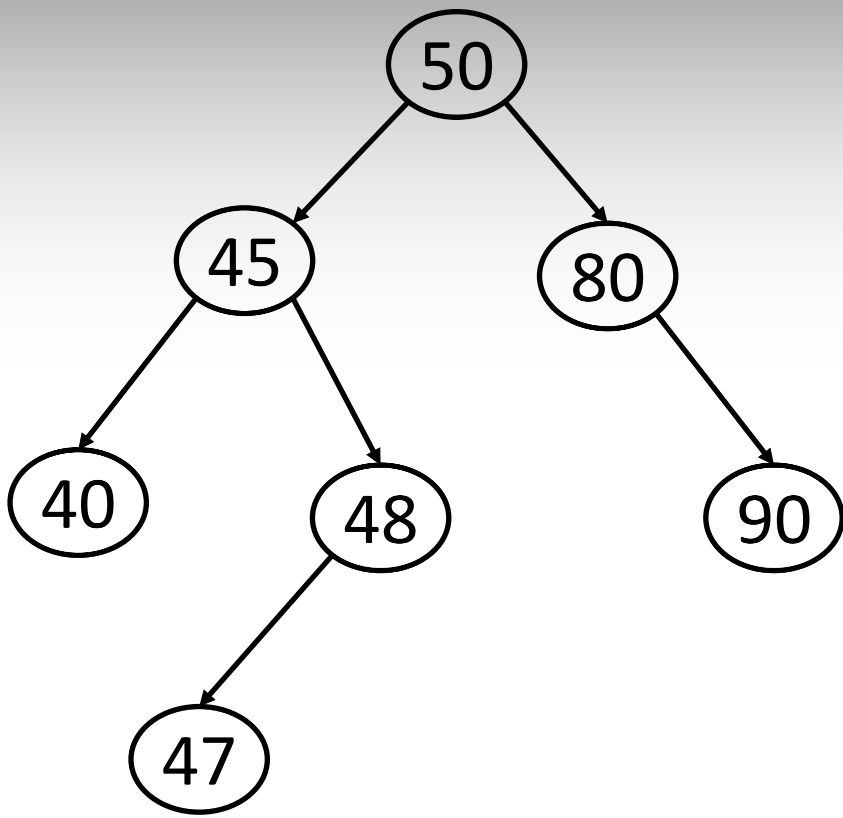
Definición de árbol binario de búsqueda

Un **árbol binario de búsqueda** es un árbol binario que es vacío o tiene las siguientes propiedades:

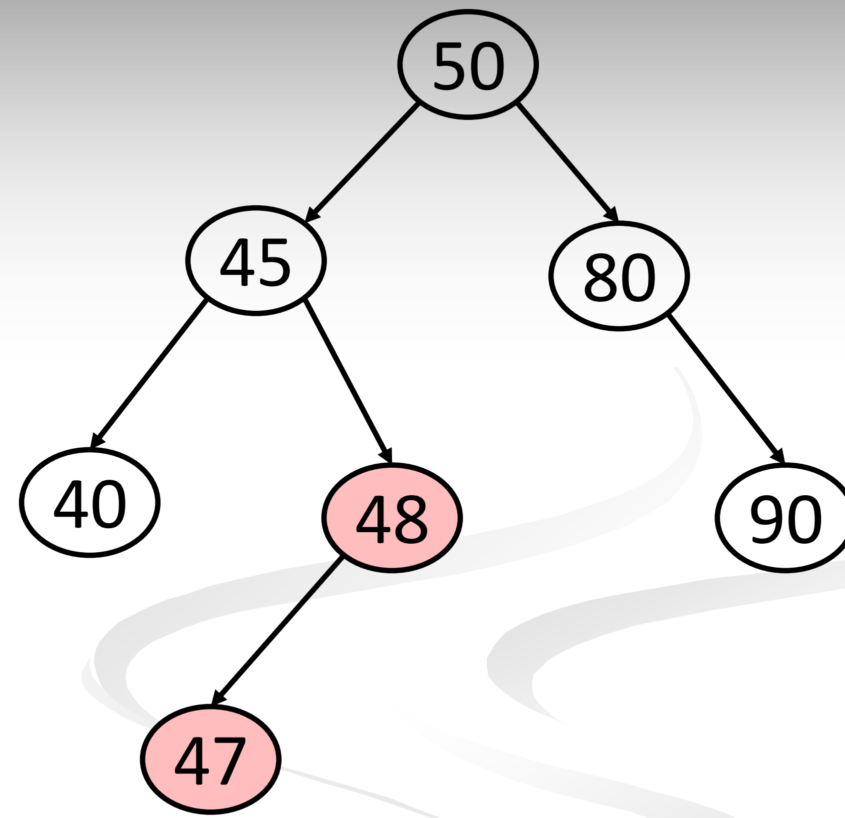
1. Cada elemento del subárbol izquierdo es menor que la raíz
2. Cada elemento del subárbol derecho es mayor que la raíz
3. Tanto el subárbol izquierdo como el subárbol derecho son árboles binarios de búsqueda

IMPORTANTE: Los elementos que almacena deben ser comparables y se puede establecer un orden entre ellos

Ejemplos de árbol binario de búsqueda

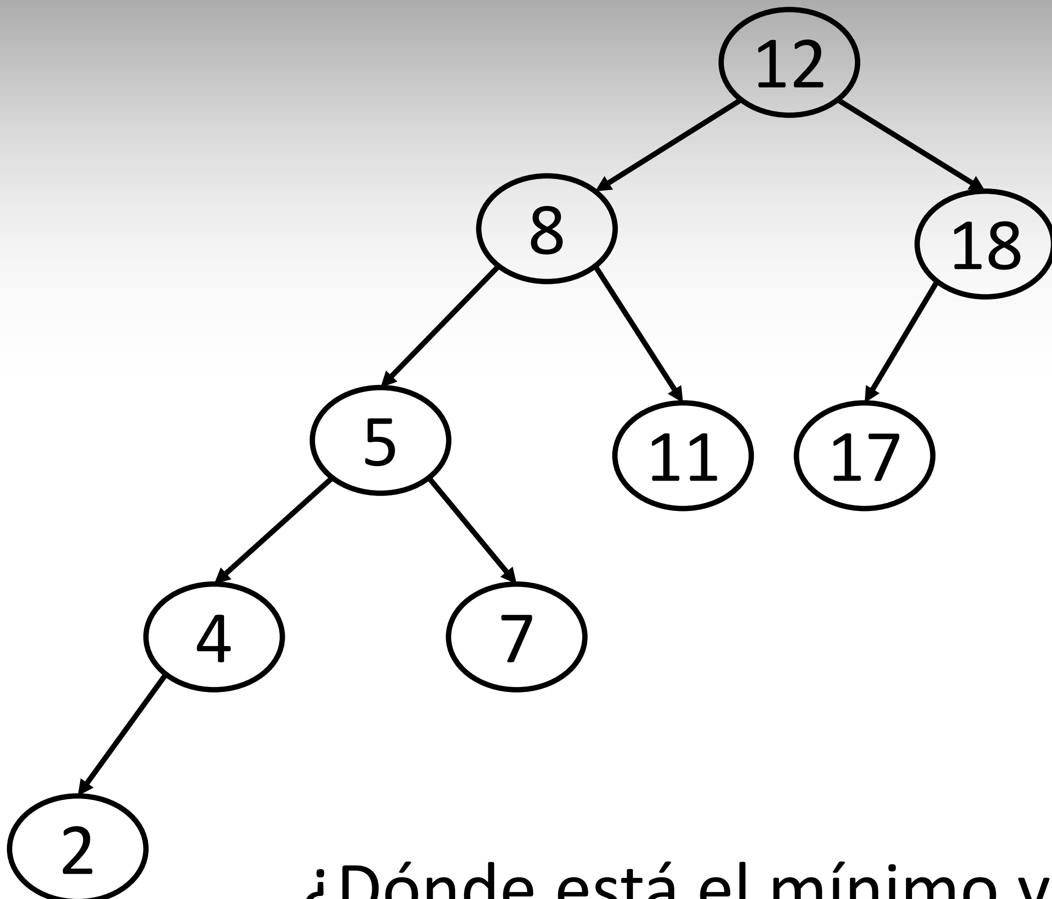


Es un ABB



No es un ABB

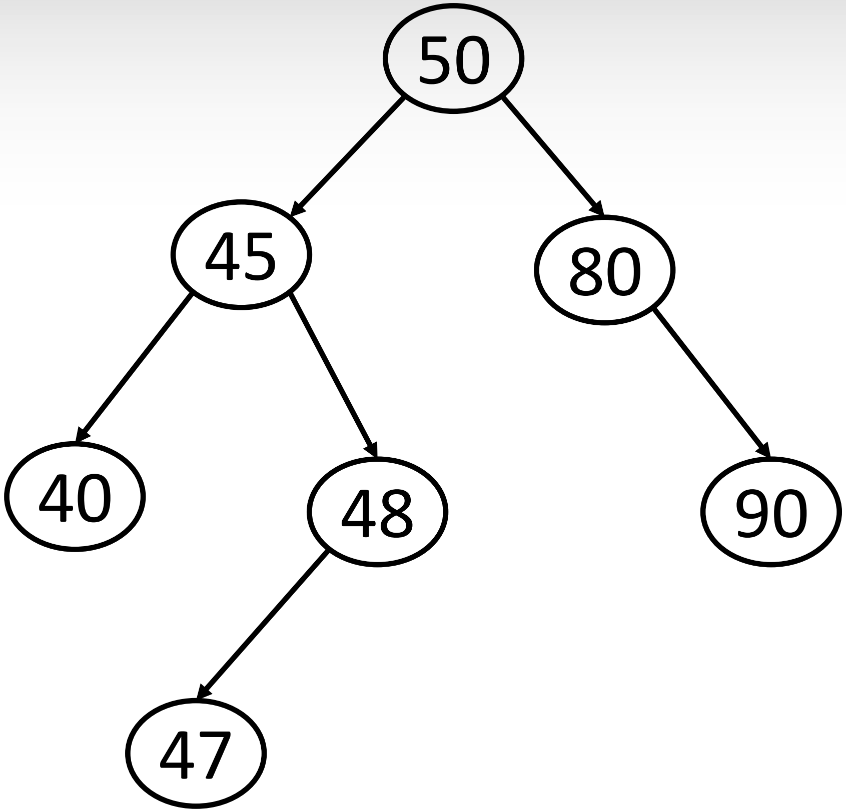
Ejemplos de árbol binario de búsqueda



¿Dónde está el mínimo y el máximo en un ABB?
¿Qué pasa con un recorrido en orden de un ABB?

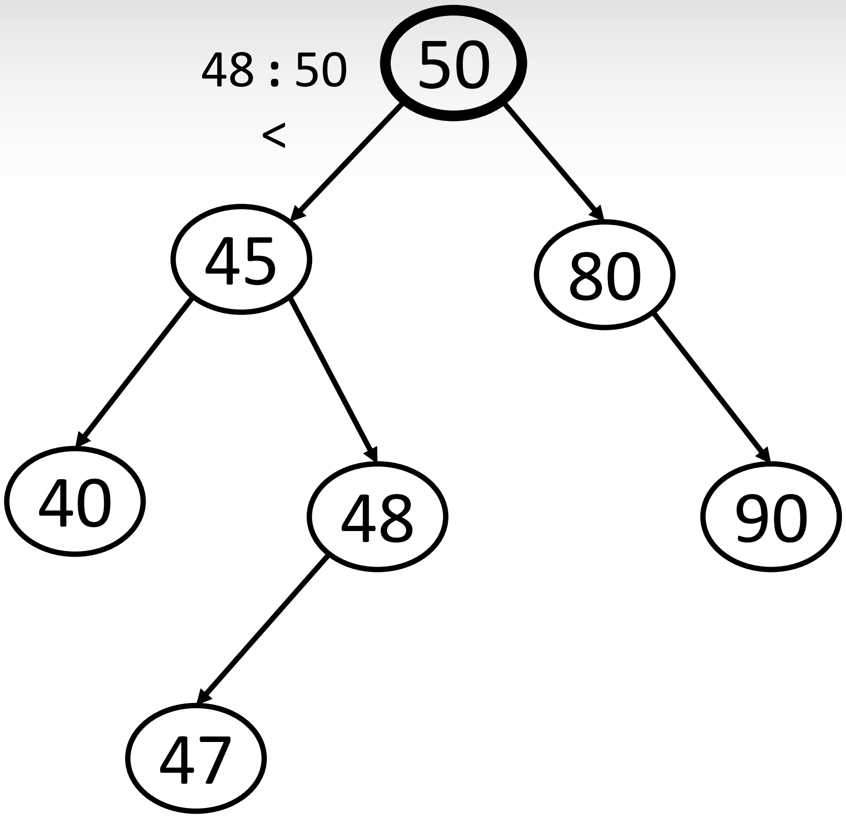
Búsqueda en árbol binario de búsqueda

Búsqueda del 48



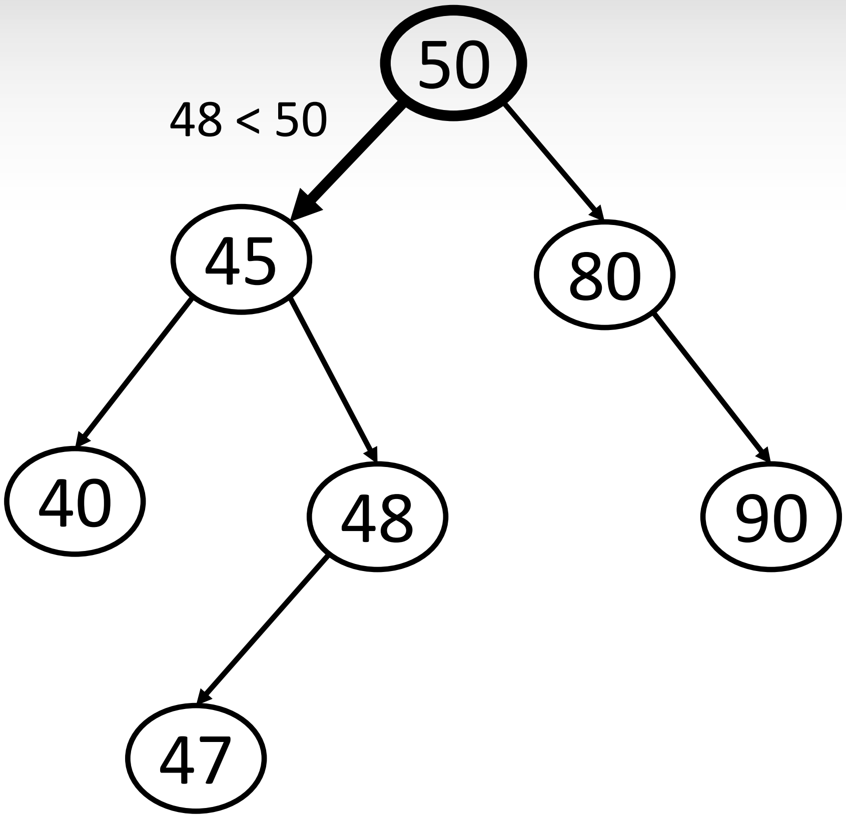
Búsqueda en árbol binario de búsqueda

Búsqueda del 48



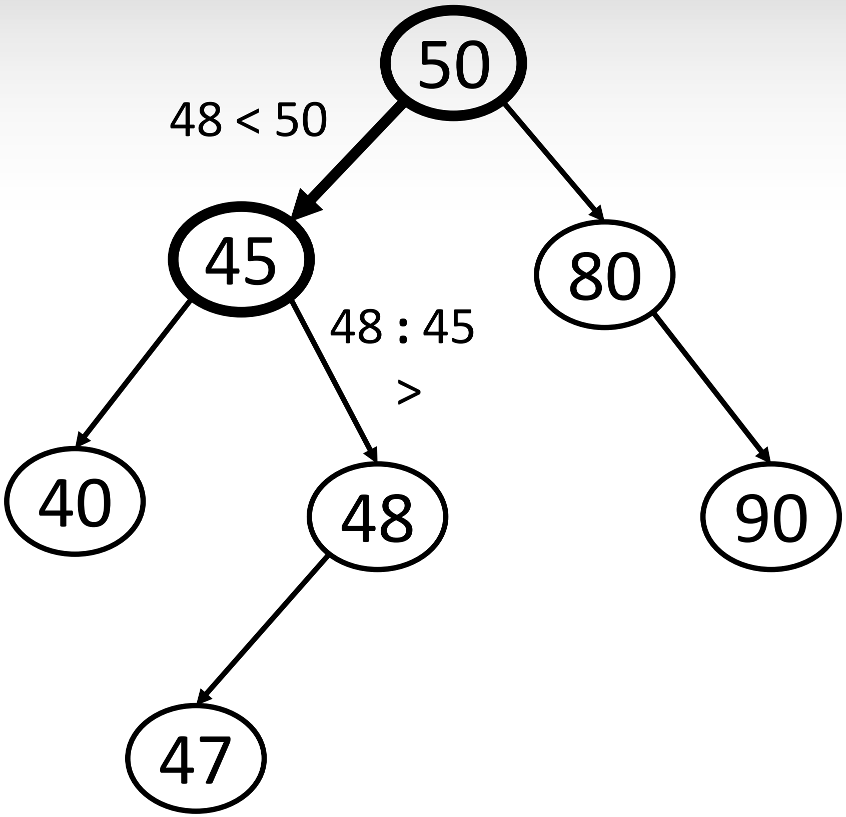
Búsqueda en árbol binario de búsqueda

Búsqueda del 48



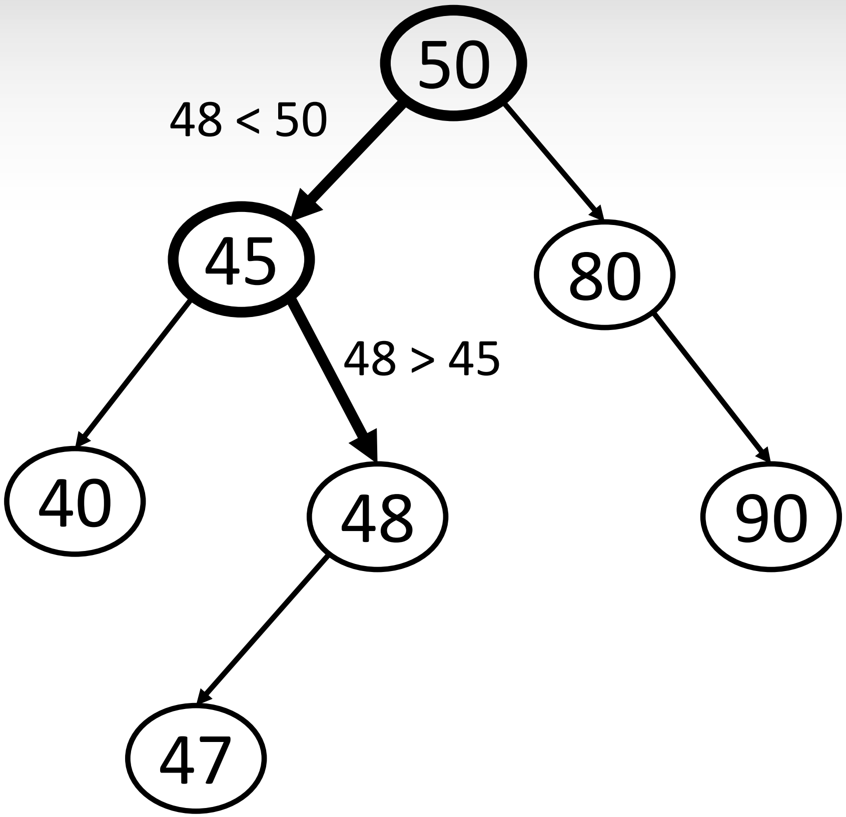
Búsqueda en árbol binario de búsqueda

Búsqueda del 48



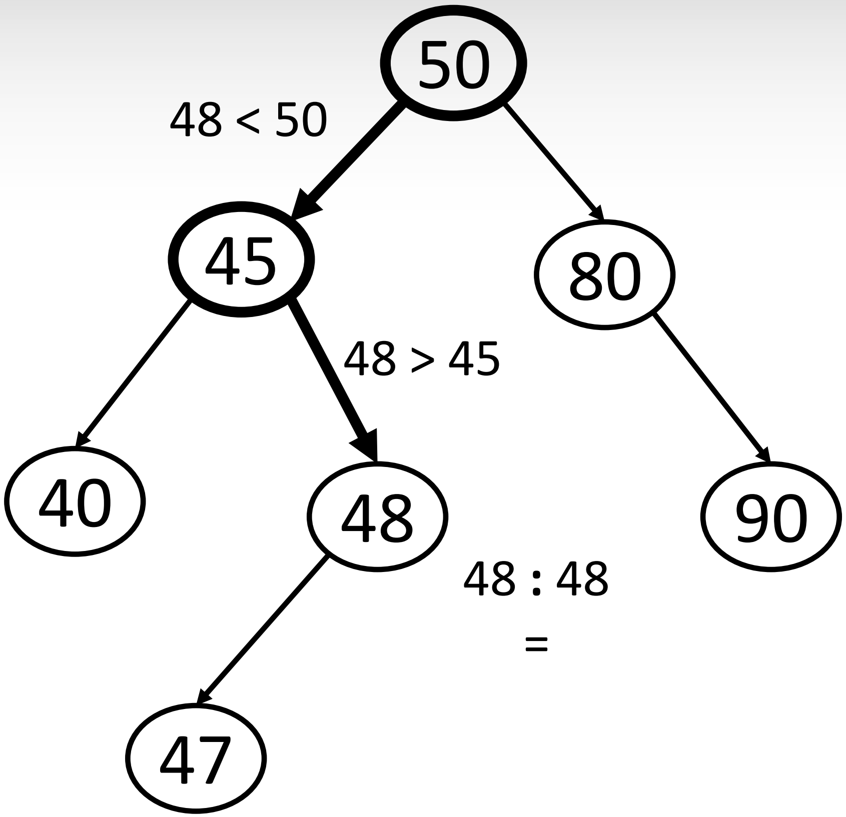
Búsqueda en árbol binario de búsqueda

Búsqueda del 48



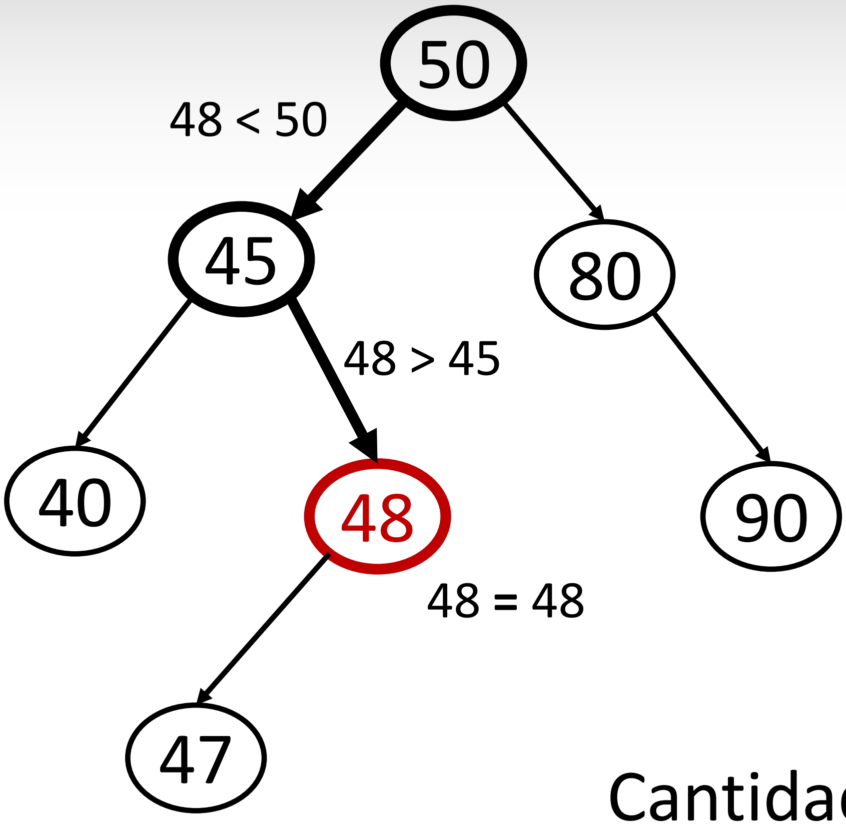
Búsqueda en árbol binario de búsqueda

Búsqueda del 48



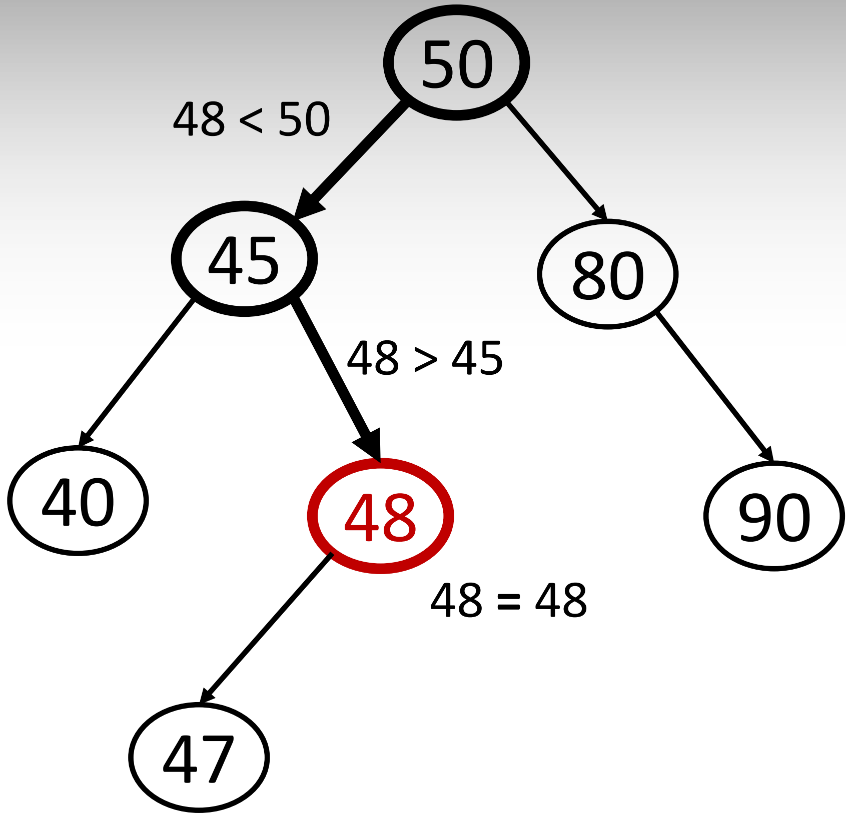
Búsqueda en árbol binario de búsqueda

Búsqueda del 48



Cantidad de comparaciones: 3

Búsqueda en árbol binario de búsqueda



La búsqueda se comienza desde la raíz y recorriendo un camino de búsqueda orientado hacia el subárbol izquierdo o derecho de cada nodo del camino dependiendo solamente del valor del nodo.

Como esta búsqueda sigue un único camino desde la raíz, puede ser fácilmente implementada en forma iterativa

Búsqueda en árbol binario de búsqueda

Versión iterativa

```
NodoAB* buscarIterativo(NodoAB* A, int x){
  NodoAB* cursor = A;
  while ((cursor != NULL)&&(cursor->dato != x)) {
    if (cursor->dato > x)
      cursor = cursor->izq;
    else
      cursor = cursor->der;
  }
  return cursor;
}
```

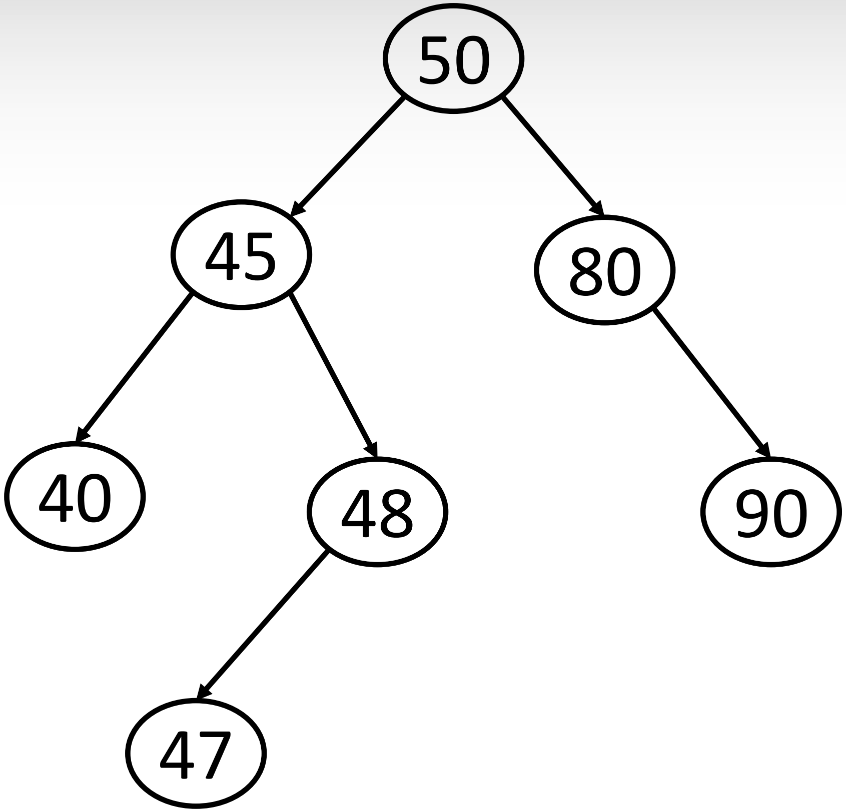
Búsqueda en árbol binario de búsqueda

Versión recursiva

```
NodoAB* buscarRecursivo(NodoAB* A, int x){
  NodoAB* res;
  if (esVacio(A))
    res = NULL;
  else if (A->dato == x)
    res = A;
  else if (A->dato > x)
    res = buscarRecursivo(A->izq, x);
  else res = buscarRecursivo(A->der, x);
  return res
}
```

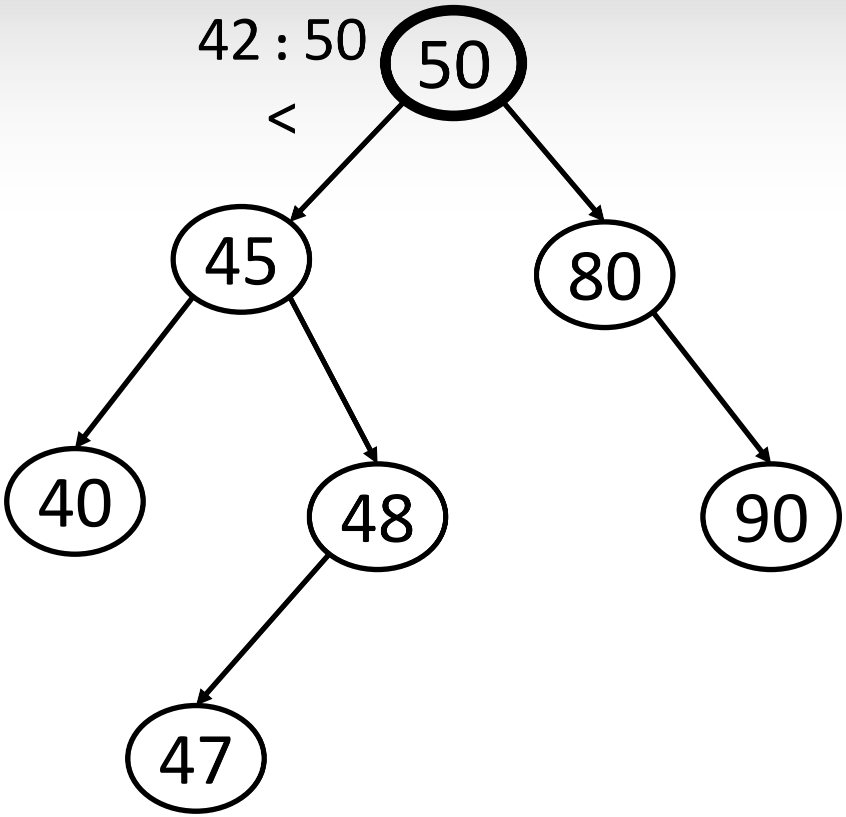
Búsqueda e inserción en ABB

Búsqueda del 42



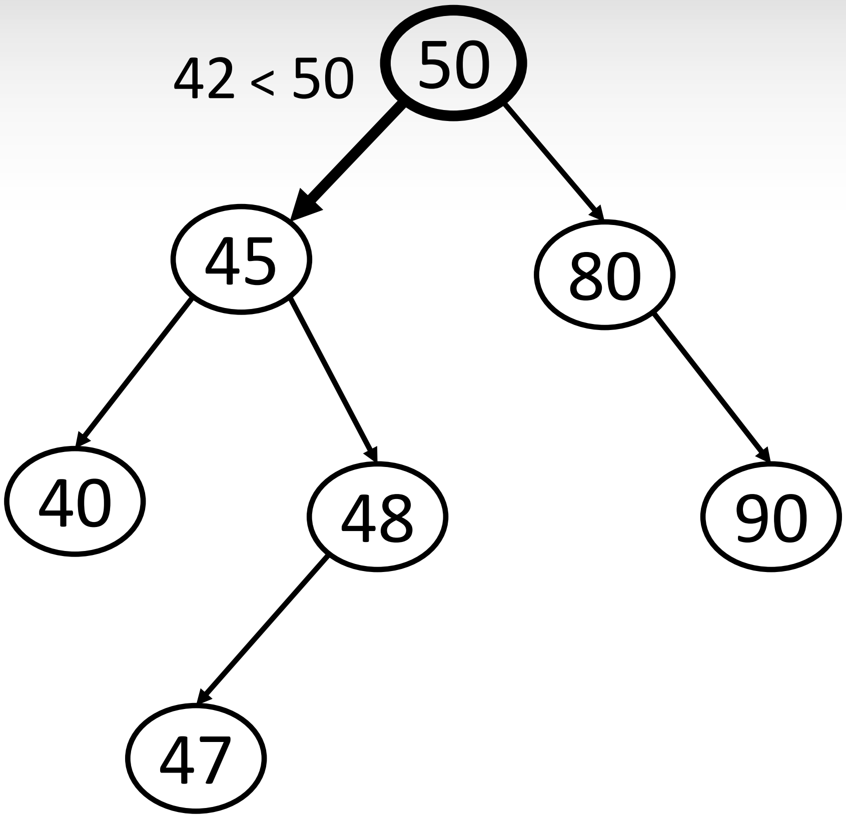
Búsqueda e inserción en ABB

Búsqueda del 42



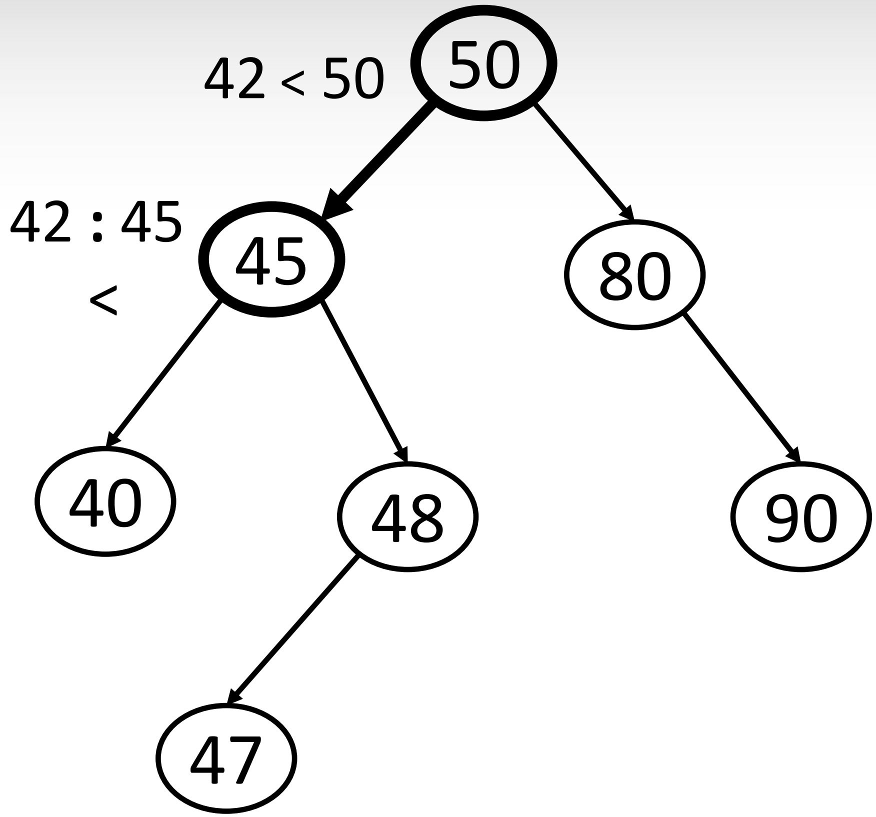
Búsqueda e inserción en ABB

Búsqueda del 42



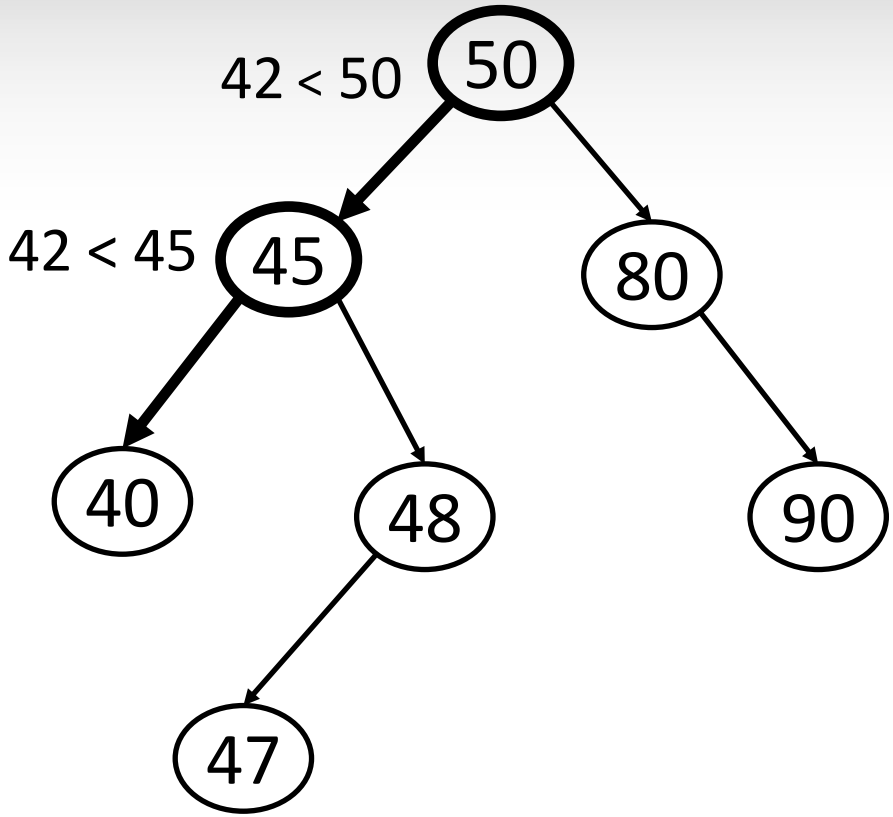
Búsqueda e inserción en ABB

Búsqueda del 42



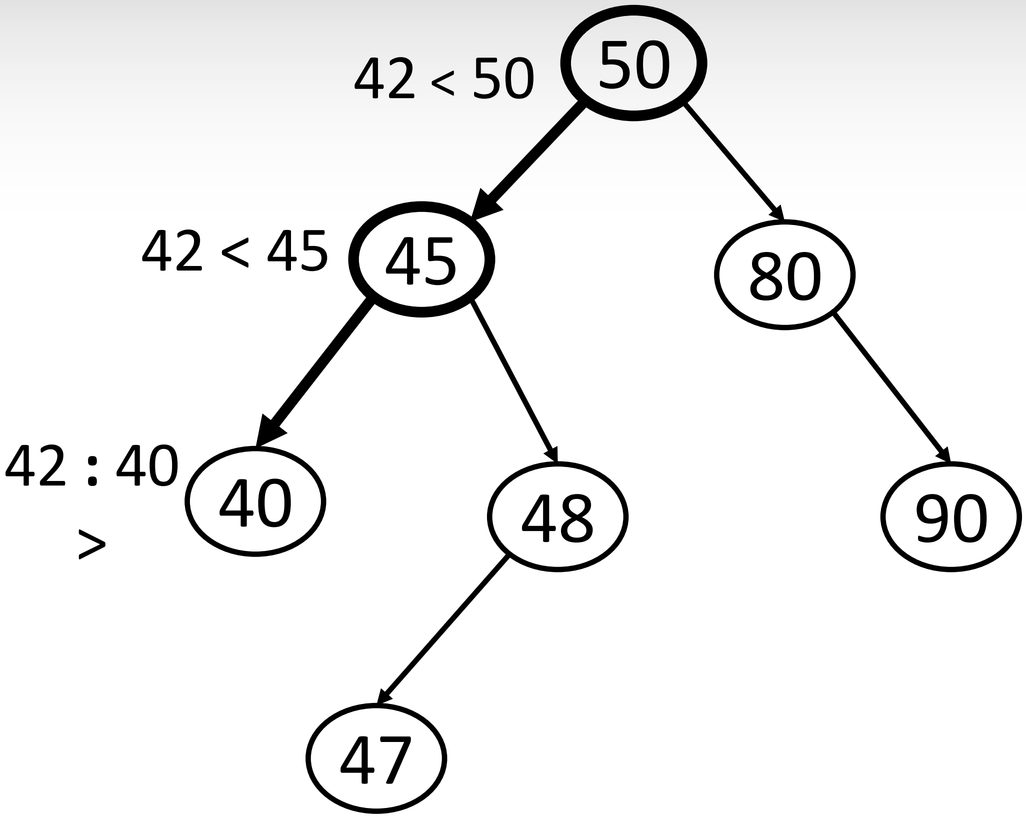
Búsqueda e inserción en ABB

Búsqueda del 42



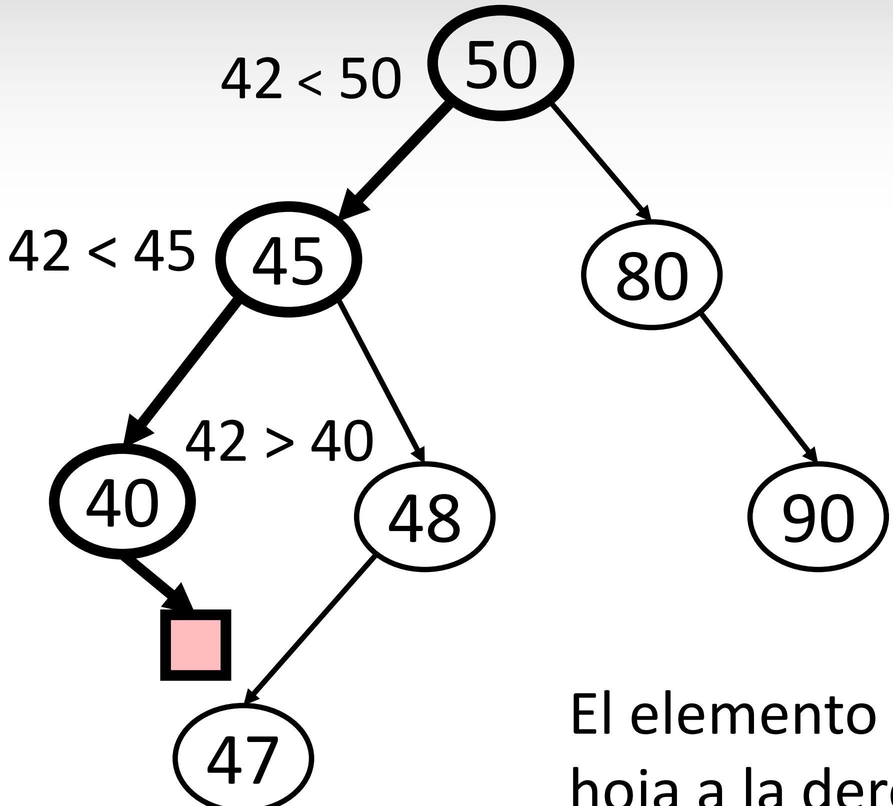
Búsqueda e inserción en ABB

Búsqueda del 42



Búsqueda e inserción en ABB

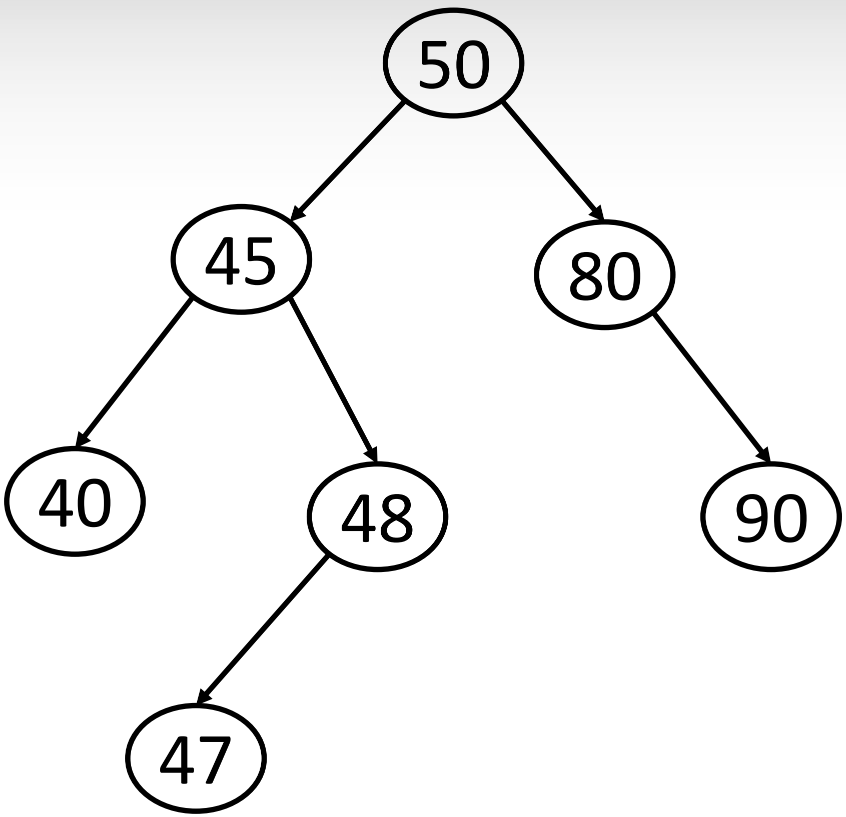
Búsqueda del 42



El elemento no está, debe insertarse una hoja a la derecha del 40 con el valor 42

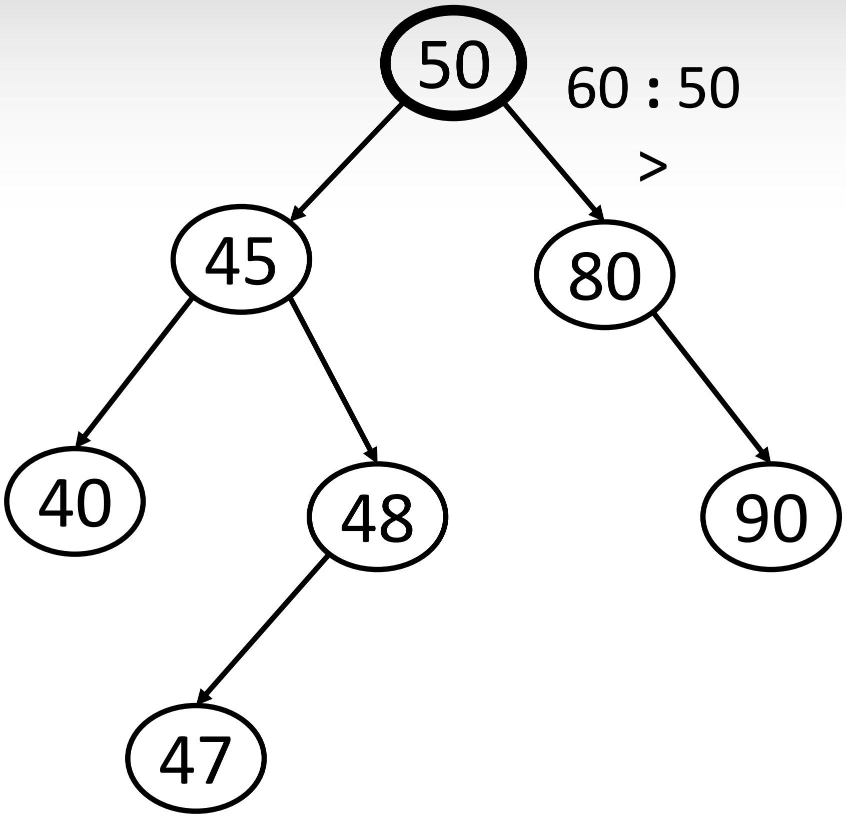
Búsqueda e inserción en ABB

Búsqueda del 60



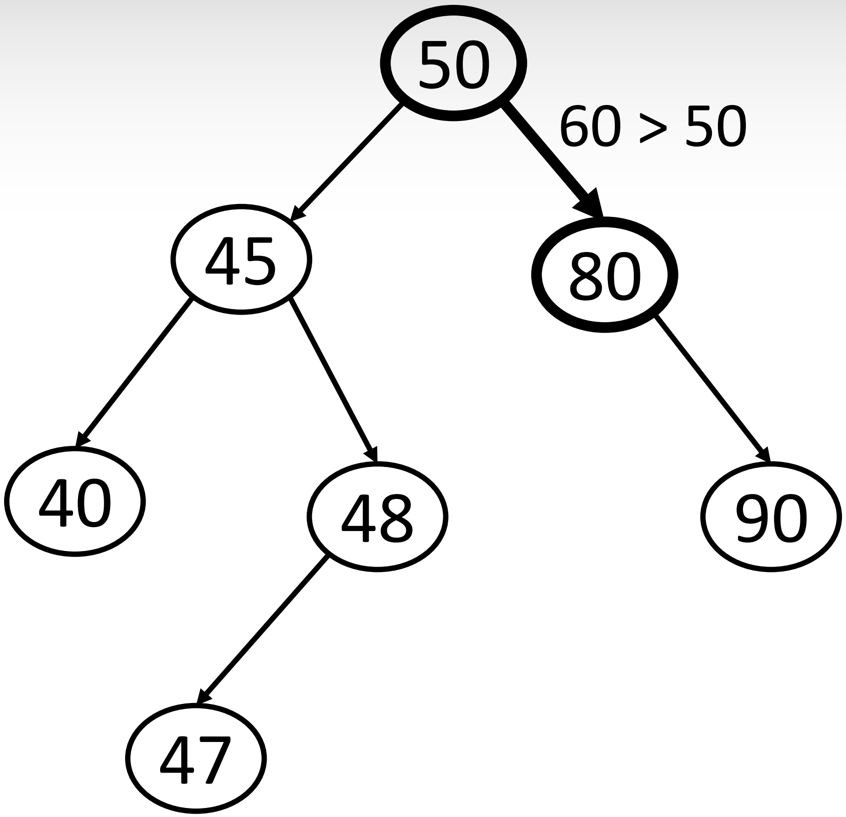
Búsqueda e inserción en ABB

Búsqueda del 60



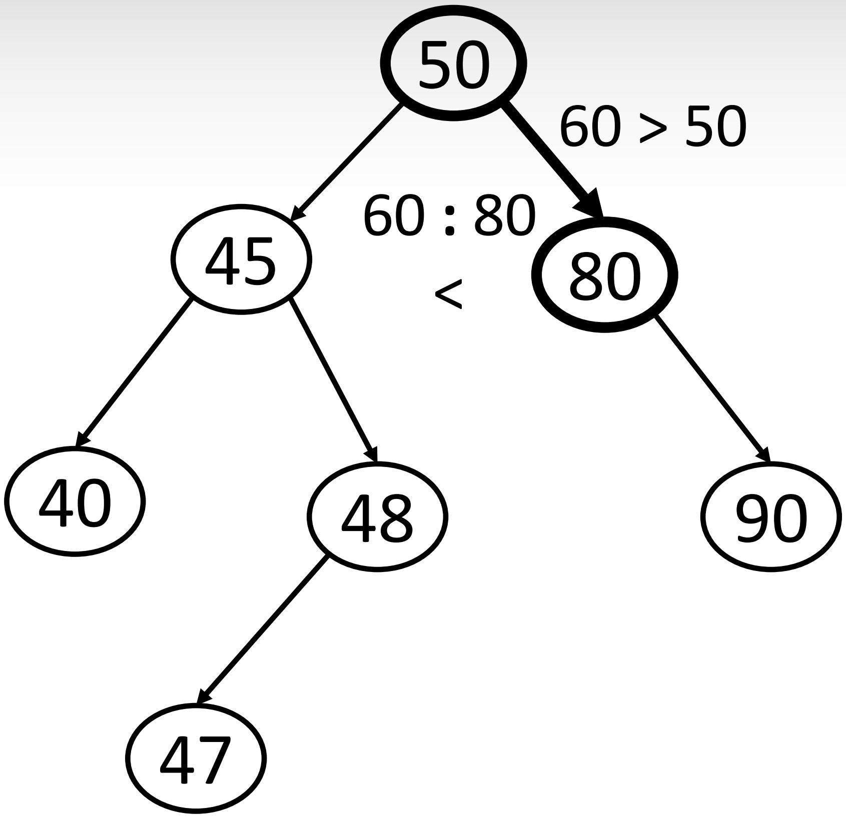
Búsqueda e inserción en ABB

Búsqueda del 60



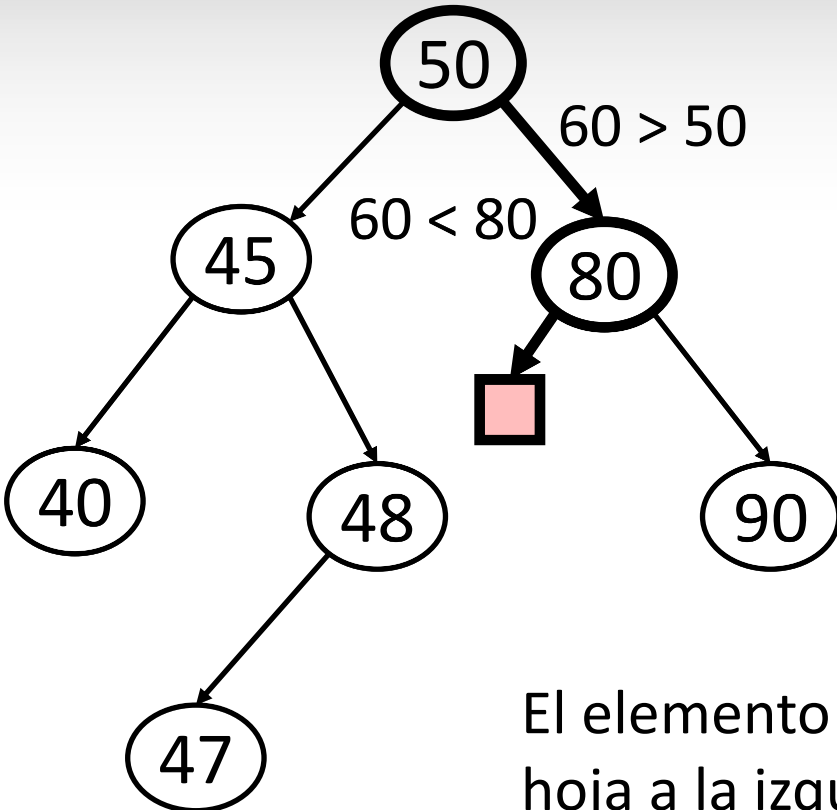
Búsqueda e inserción en ABB

Búsqueda del 60



Búsqueda e inserción en ABB

Búsqueda del 60



El elemento no esta debe insertarse una hoja a la izquierda del 80 con el valor 60

Búsqueda e inserción en ABB

La no existencia de un elemento se concluye cuando en el proceso recursivo se llega a un árbol vacío y es en este momento cuando se realiza la inserción.

En todos los casos siempre se realiza la inserción de una hoja y nunca la de un nodo intermedio

Búsqueda e inserción en ABB

1. Si el árbol es vacío el elemento no se encuentra y se debe crear una hoja que contenga el valor X.
2. Si el árbol no es vacío y X *es igual* a la información guardada en la raíz del árbol, el elemento se encuentra y no es necesario insertarlo en el árbol
3. Si el árbol no es vacío y X *es menor* que la información guardada en la raíz se debe realizar la búsqueda en el subárbol izquierdo
4. Si el árbol no es vacío y X *es mayor* que la información guardada en la raíz se debe realizar la búsqueda en el subárbol derecho

Búsqueda e inserción en ABB

```
void insertar(NodoAB* &A, int x){
  if (esVacio(A))
    A = new NodoAB(x);
  else if (A->dato > x)
    insertar(A->izq, x);
  else if (A->dato < x)
    insertar(A->der, x);
}
```

Búsqueda y eliminación en ABB

Analizaremos dos casos

1. El elemento a eliminar está en un nodo que no posee dos hijos diferentes del árbol vacío

2. El elemento a eliminar está en un nodo que posee dos hijos diferentes del árbol vacío

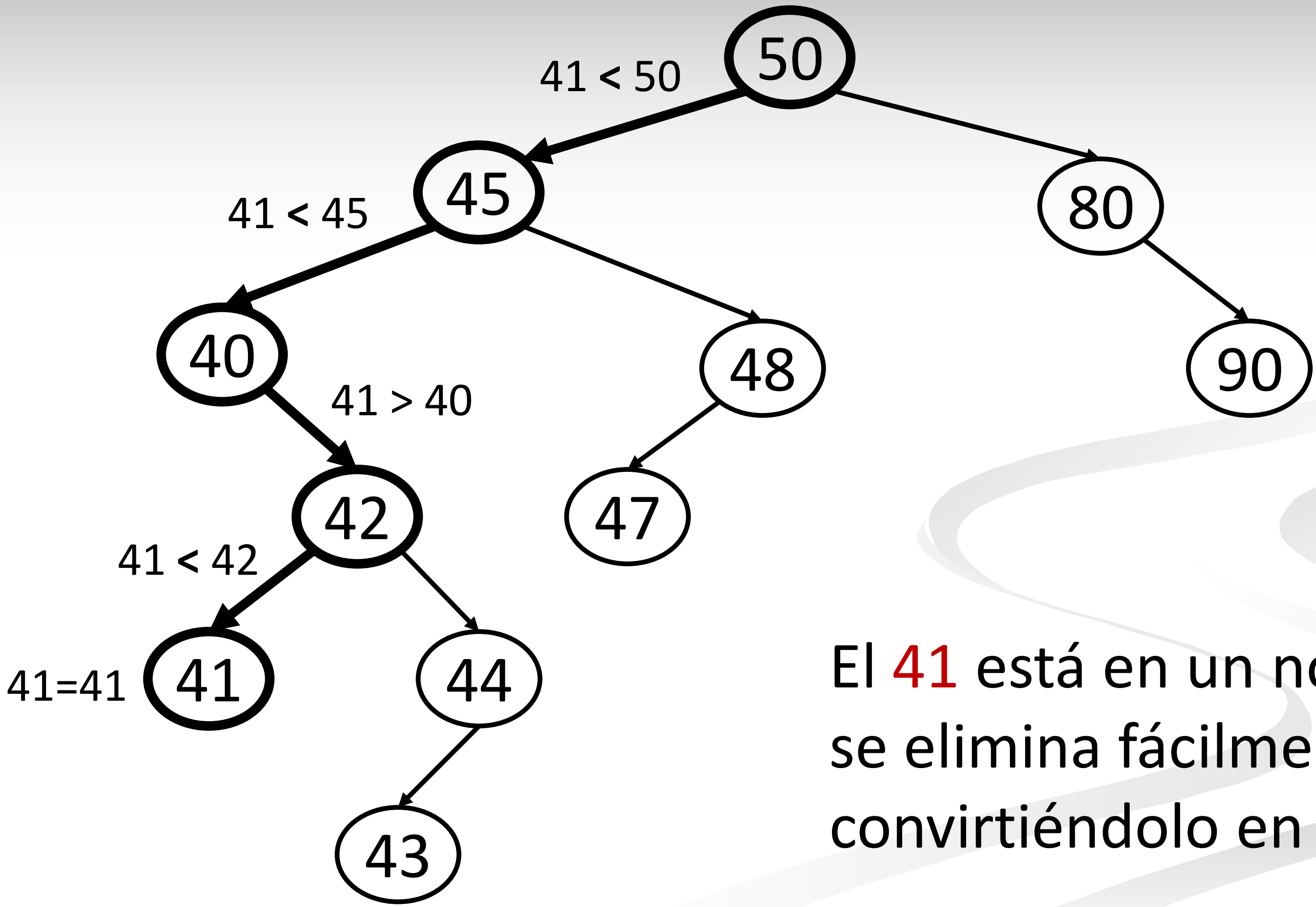
Búsqueda y eliminación en ABB

Caso 1: El elemento a eliminar está en un nodo que no posee dos hijos diferentes del árbol vacío

a) Los dos hijos son vacíos. Se puede eliminar el nodo fácilmente.
b) El hijo derecho es vacío y el izquierdo no. El hijo izquierdo pasa a ocupar su lugar
c) El hijo izquierdo es vacío y el derecho no. El hijo derecho pasa a ocupar su lugar

Búsqueda y eliminación en ABB

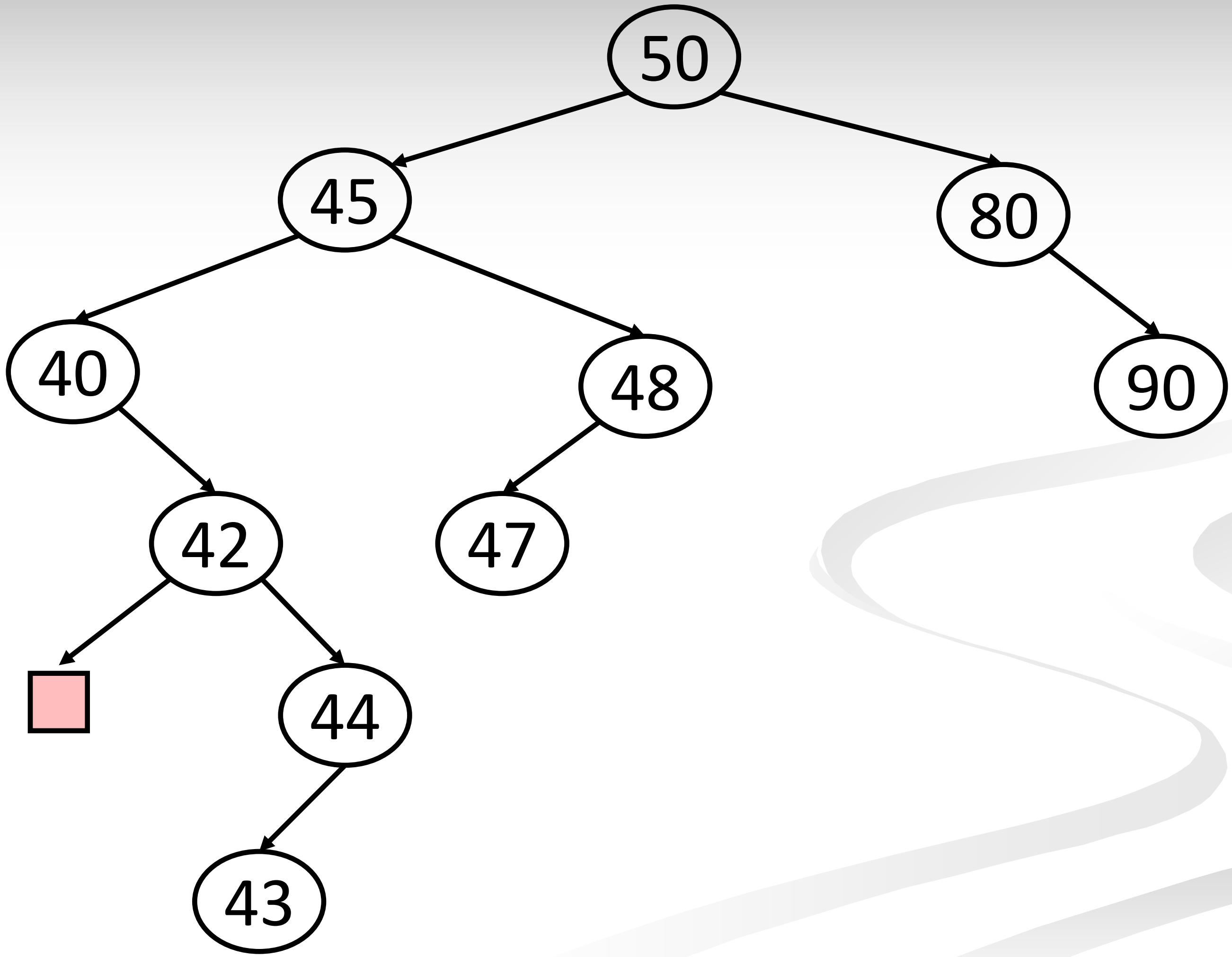
Búsqueda y eliminación del 41 (ambos hijos son vacíos)



El 41 está en un nodo hoja y se elimina fácilmente convirtiéndolo en vacío

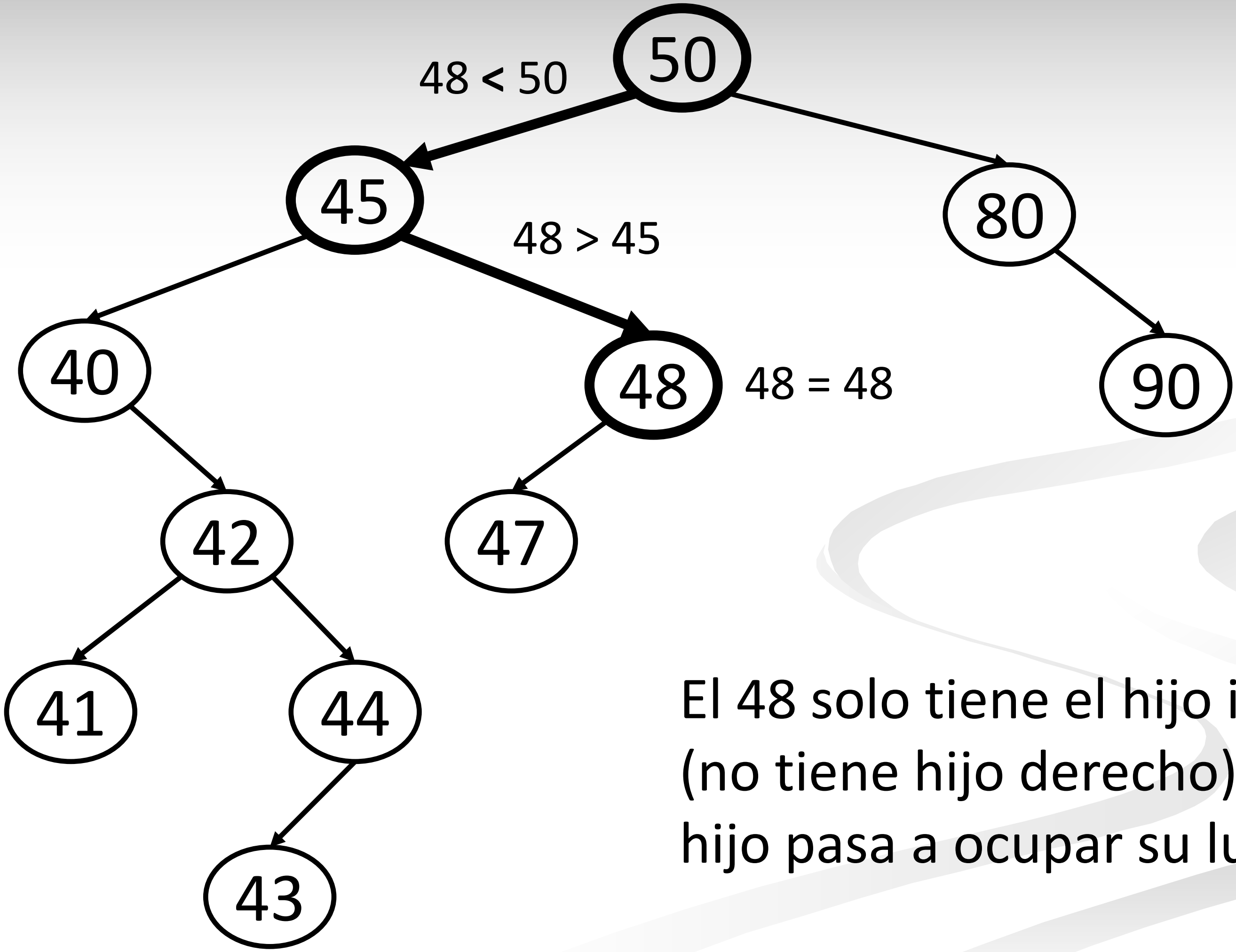
Búsqueda y eliminación en ABB

Búsqueda y eliminación del 41 (ambos hijos son vacíos)



Búsqueda y eliminación en ABB

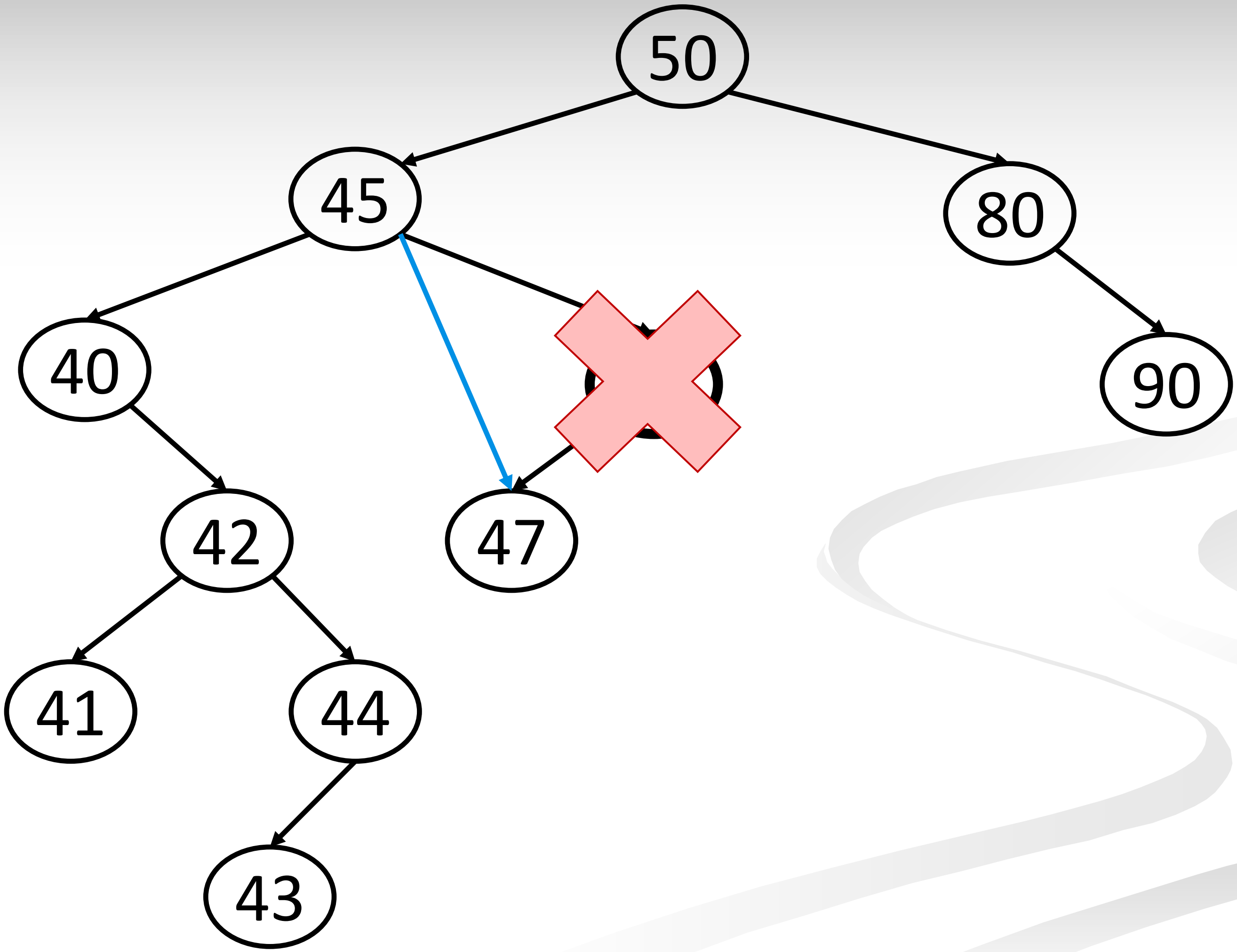
Búsqueda y eliminación del 48 (el hijo derecho es vacío)



El 48 solo tiene el hijo izquierdo (no tiene hijo derecho) y su único hijo pasa a ocupar su lugar

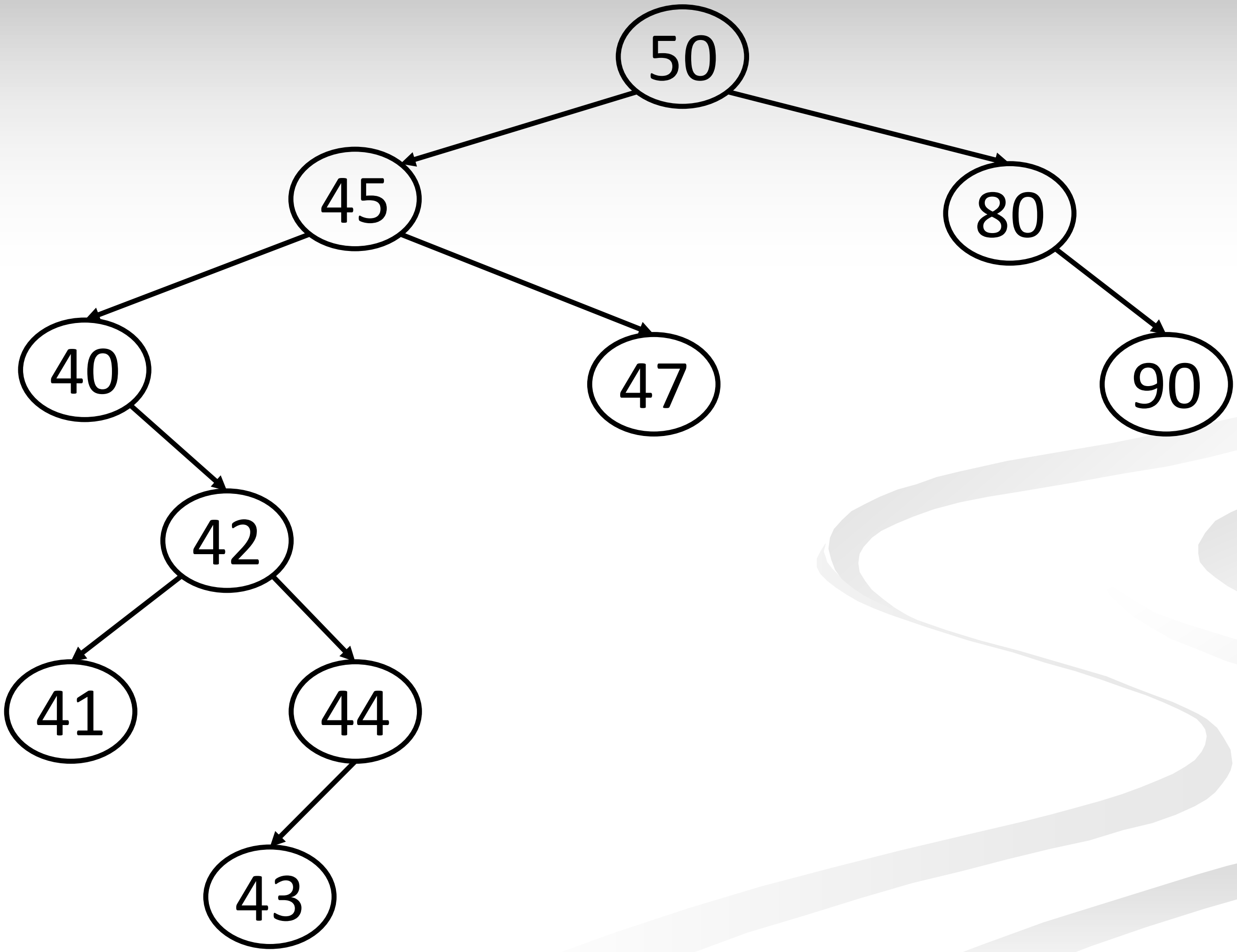
Búsqueda y eliminación en ABB

Búsqueda y eliminación del 48 (el hijo derecho es vacío)



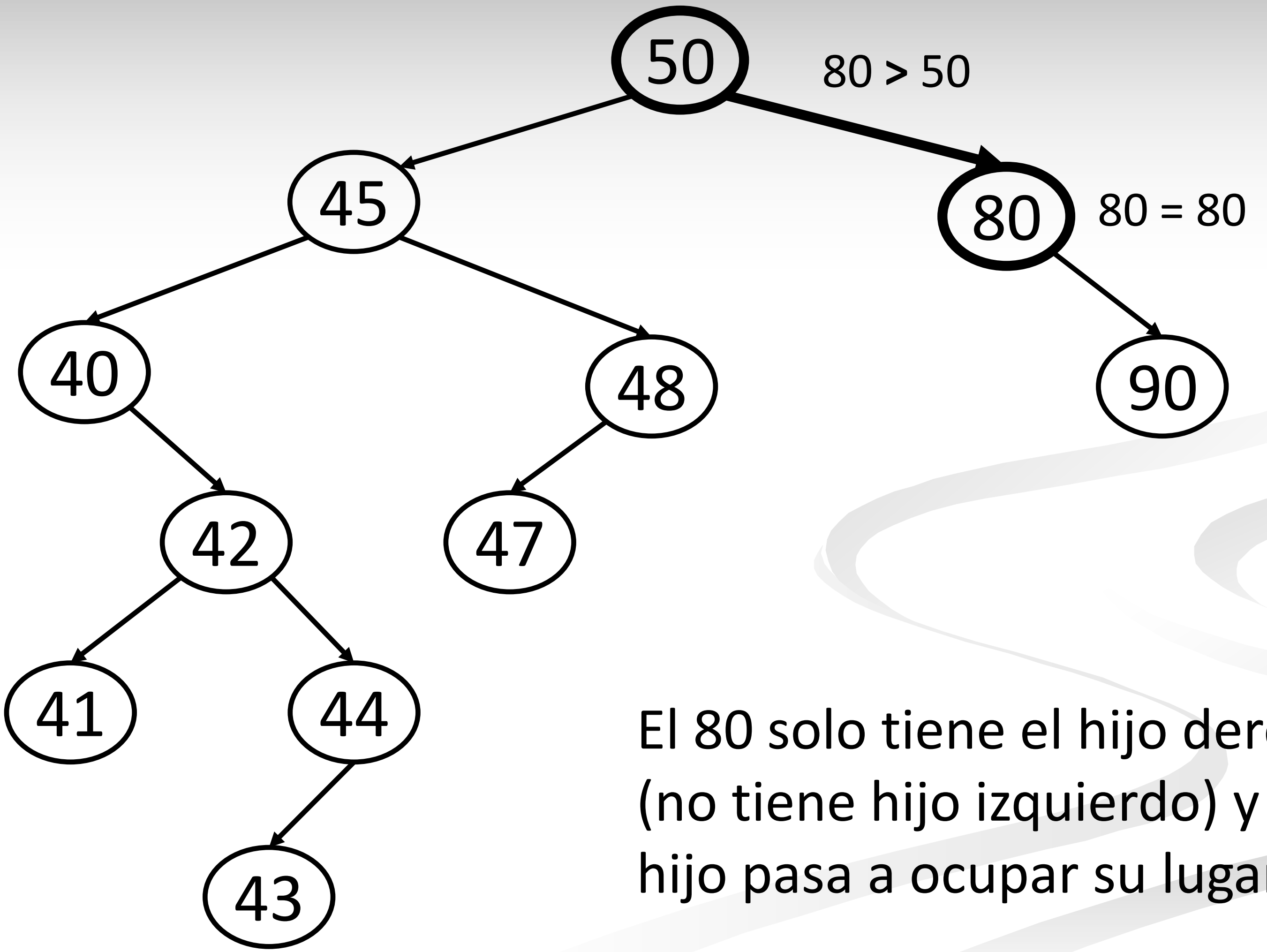
Búsqueda y eliminación en ABB

Búsqueda y eliminación del 48 (el hijo derecho es vacío)



Búsqueda y eliminación en ABB

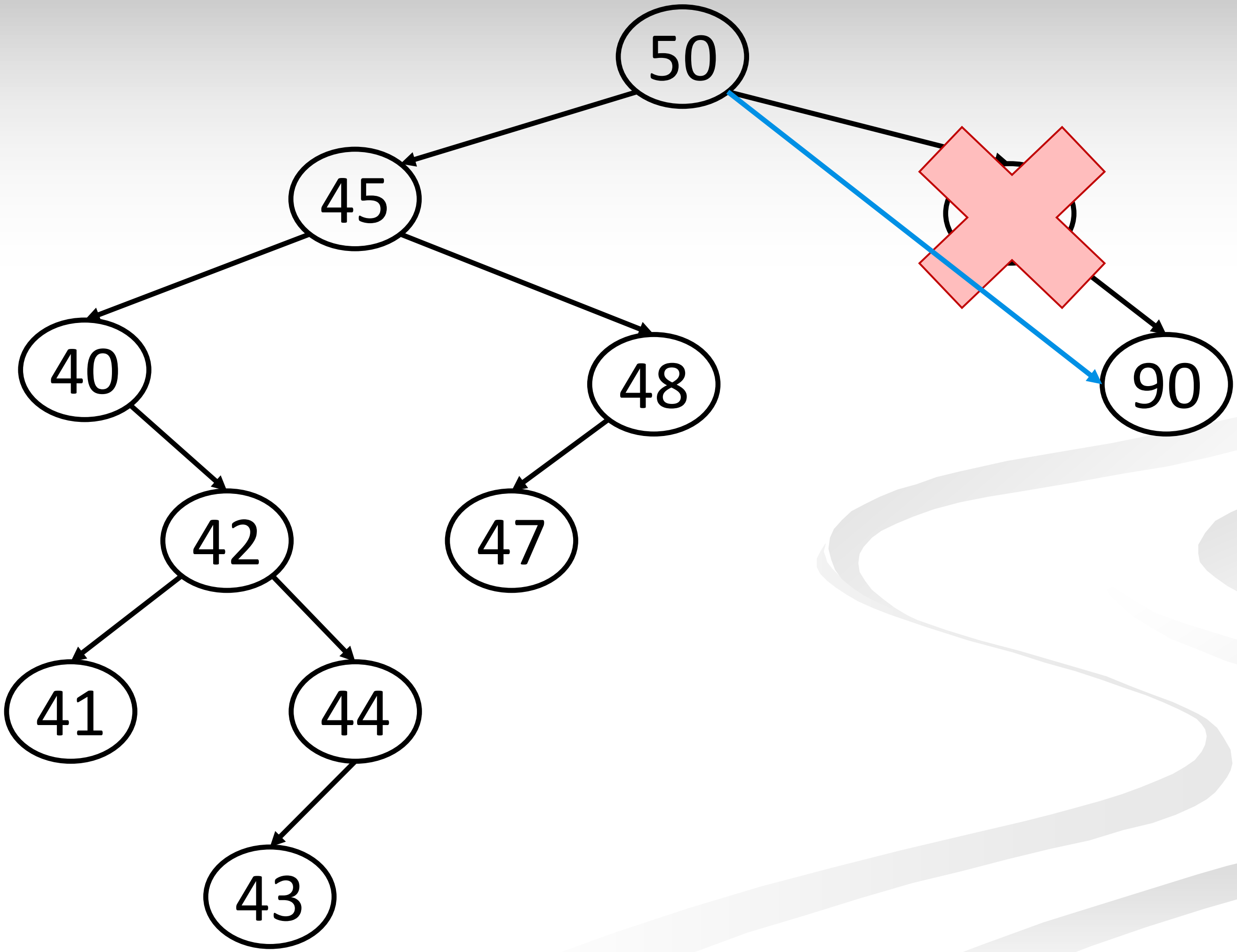
Búsqueda y eliminación del 80 (el hijo izquierdo es vacío)



El 80 solo tiene el hijo derecho (no tiene hijo izquierdo) y su único hijo pasa a ocupar su lugar

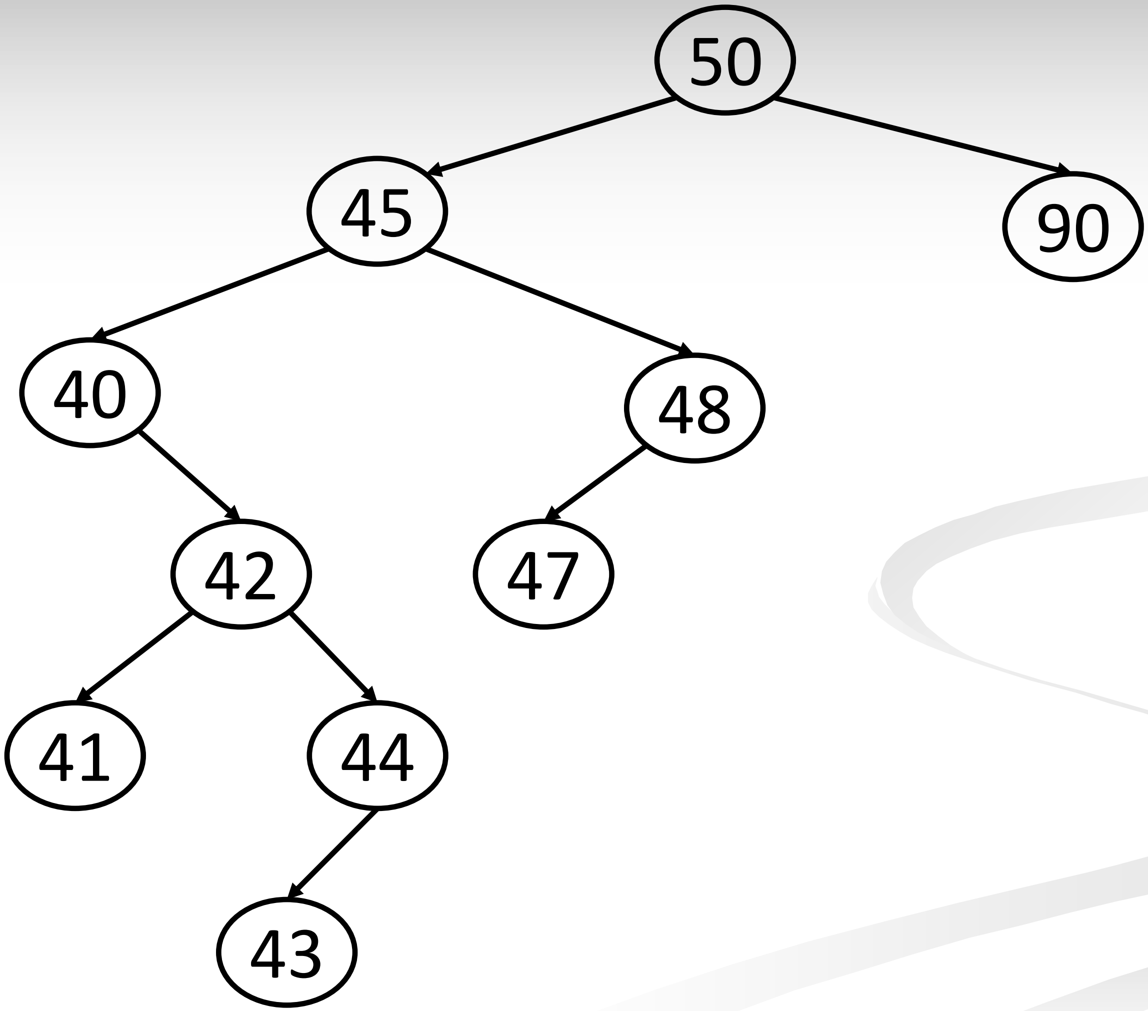
Búsqueda y eliminación en ABB

Búsqueda y eliminación del 80 (el hijo izquierdo es vacío)



Búsqueda y eliminación en ABB

Búsqueda y eliminación del 80 (el hijo izquierdo es vacío)



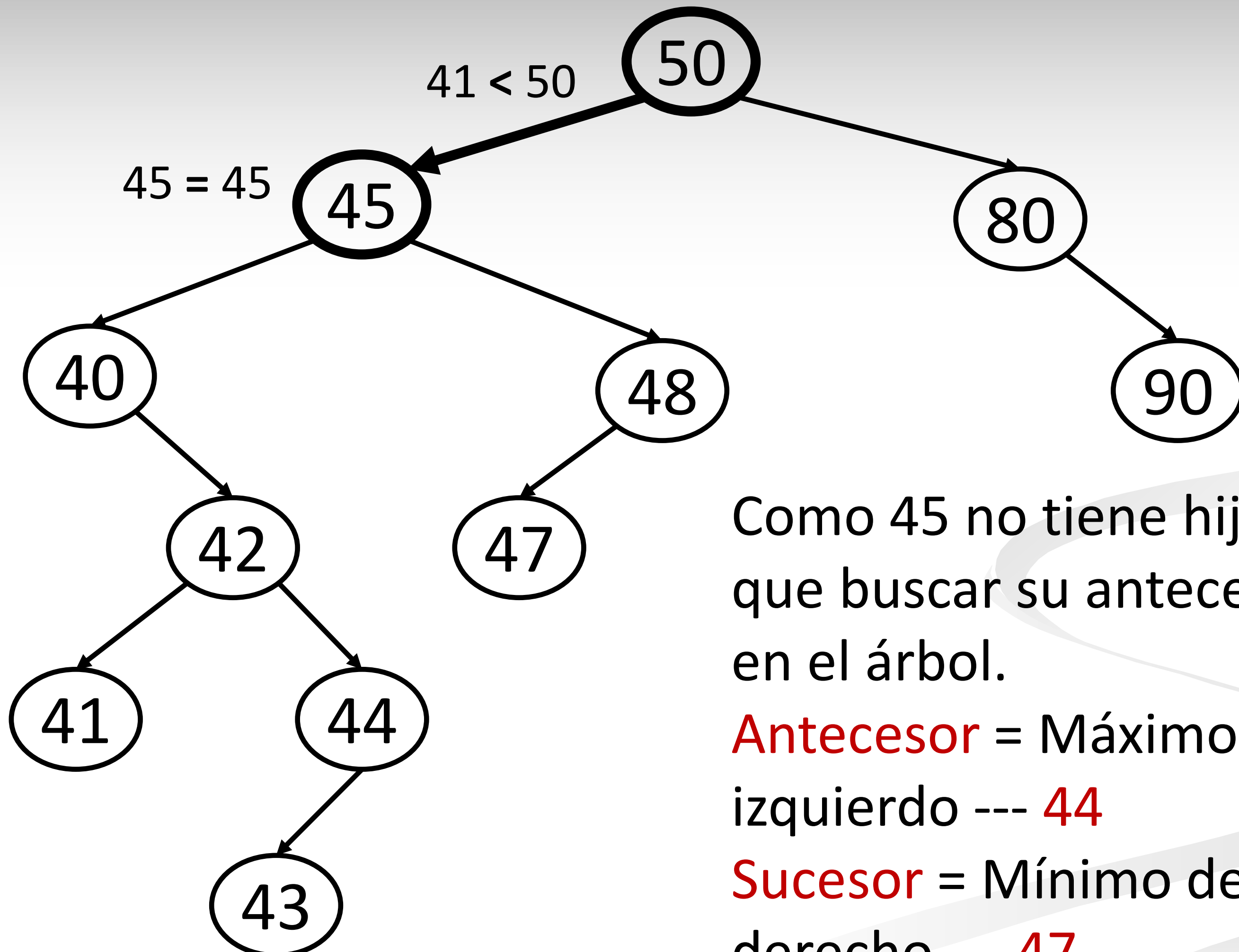
Búsqueda y eliminación en ABB

Caso 2: El elemento a eliminar está en un nodo que posee dos hijos diferentes del árbol vacío

a) Se sustituye el elemento por su predecesor (o sucesor) en entreorden y se elimina el predecesor. Se cae en el caso de la eliminación de un nodo que a lo sumo posee un subárbol diferente del vacío.

Búsqueda y eliminación en ABB

Búsqueda y eliminación del 45 (no tiene hijos vacíos)



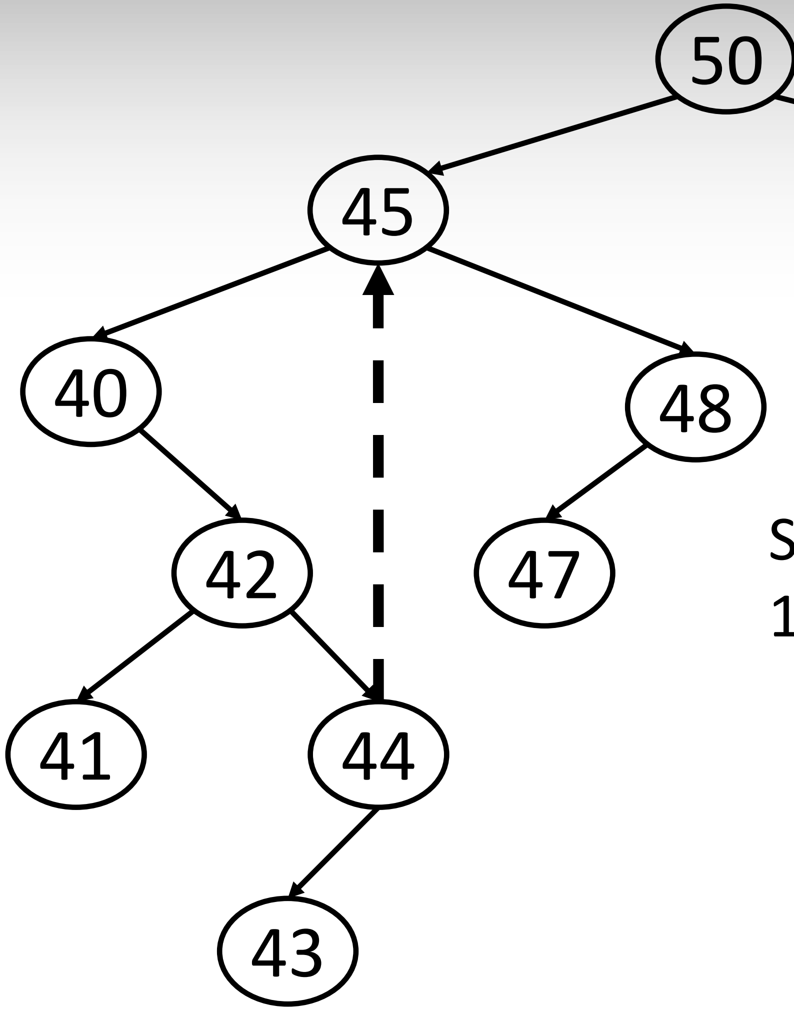
Como 45 no tiene hijos vacíos, hay que buscar su antecesor o su sucesor en el árbol.

Antecesor = Máximo del subárbol izquierdo --- 44

Sucesor = Mínimo del subárbol derecho --- 47

Búsqueda y eliminación en ABB

Búsqueda y eliminación del 45 (no tiene hijos vacíos)

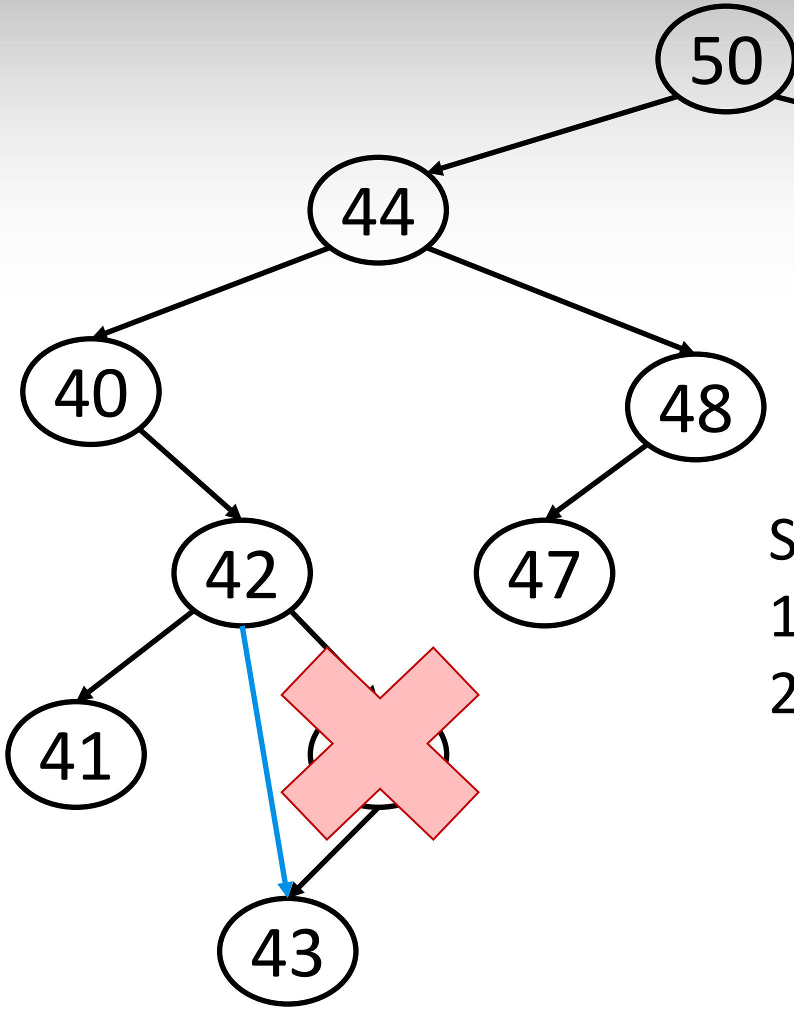


Suponiendo que se toma el antecesor.

1. El dato del nodo 45 pasa a ser 44.

Búsqueda y eliminación en ABB

Búsqueda y eliminación del 45 (no tiene hijos vacíos)

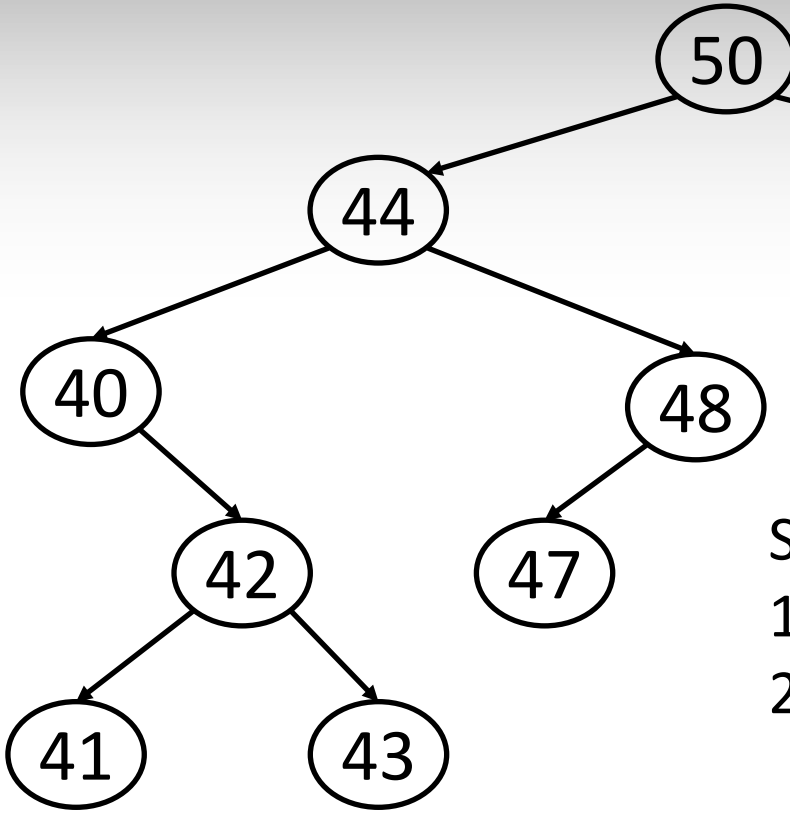


Suponiendo que se toma el antecesor.

1. El dato del nodo 45 pasa a ser 44.
2. Eliminar el 44 en el subárbol izquierdo (cae en caso 1)

Búsqueda y eliminación en ABB

Búsqueda y eliminación del 45 (no tiene hijos vacíos)

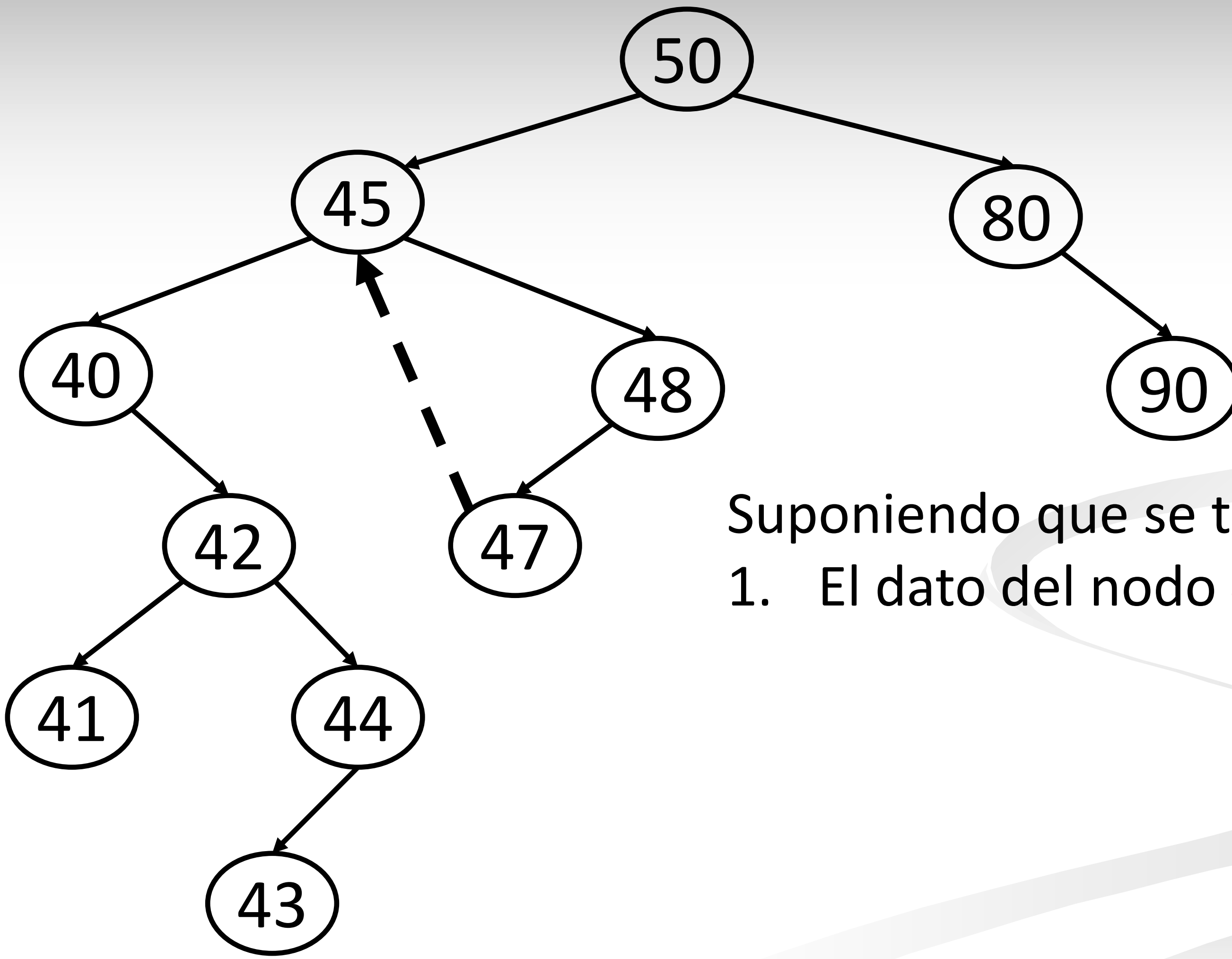


Suponiendo que se toma el antecesor.

1. El dato del nodo 45 pasa a ser 44.
2. Eliminar el 44 en el subárbol izquierdo (cae en caso 1)

Búsqueda y eliminación en ABB

Búsqueda y eliminación del 45 (no tiene hijos vacíos)

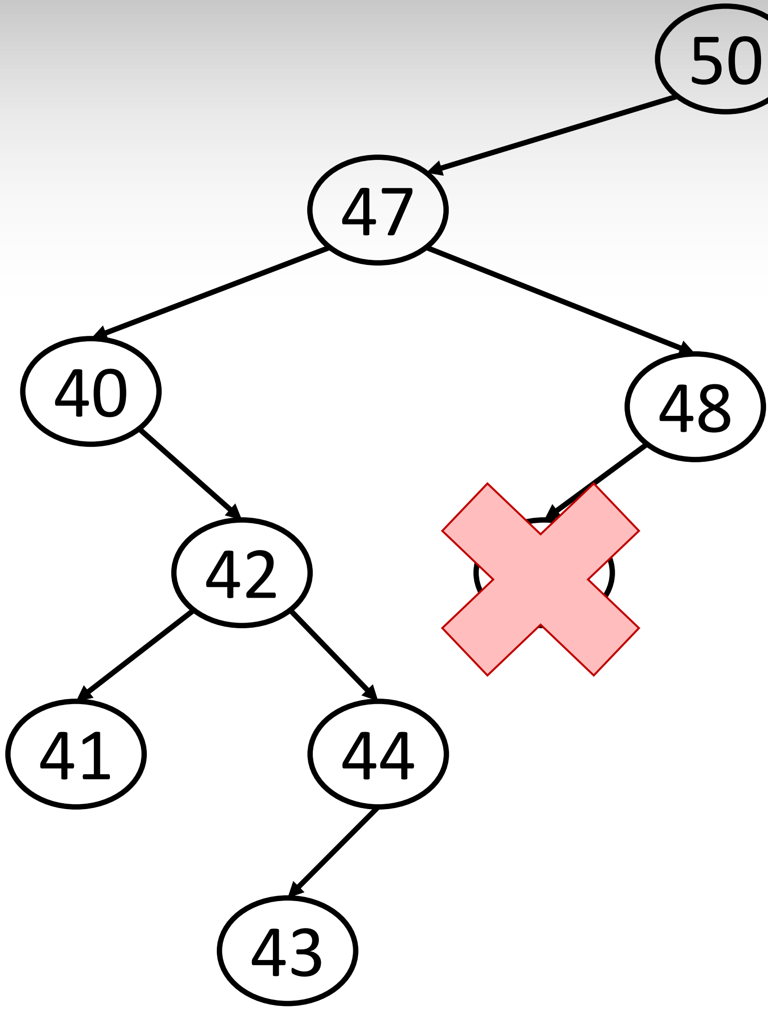


Suponiendo que se toma el sucesor.

1. El dato del nodo 47 pasa a ser 44.

Búsqueda y eliminación en ABB

Búsqueda y eliminación del 45 (no tiene hijos vacíos)

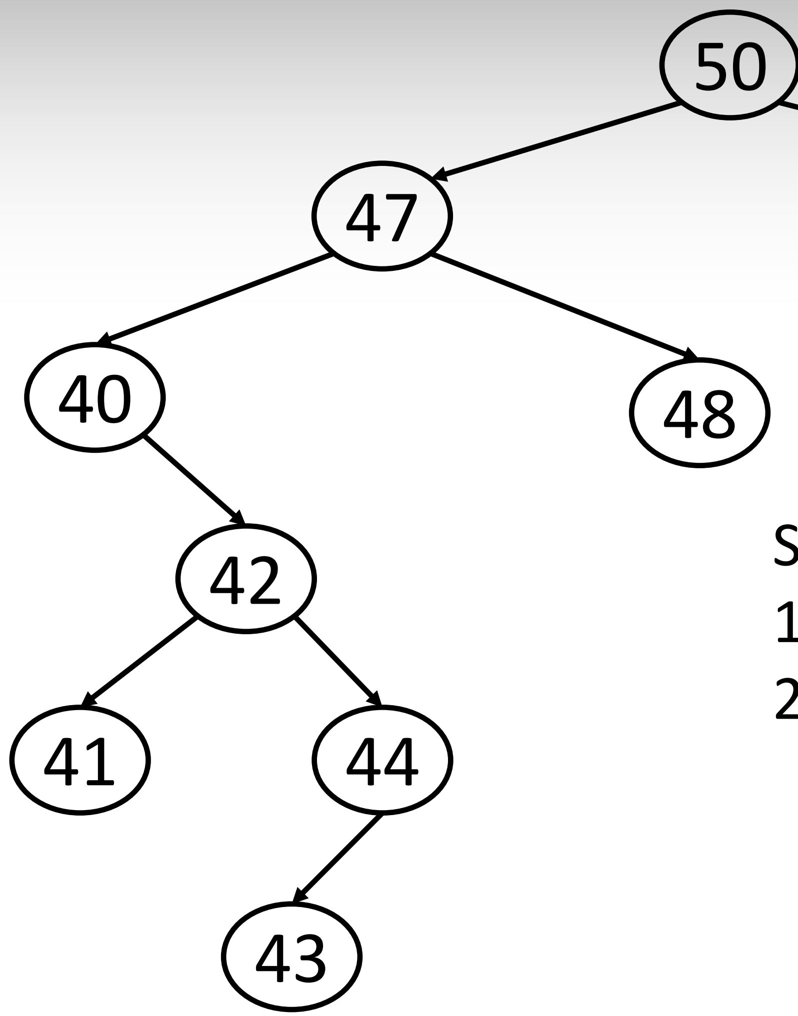


Suponiendo que se toma el sucesor.

1. El dato del nodo 45 pasa a ser 47.
2. Eliminar el 47 en el subárbol derecho (cae en caso 1)

Búsqueda y eliminación en ABB

Búsqueda y eliminación del 45 (no tiene hijos vacíos)



Suponiendo que se toma el sucesor.

1. El dato del nodo 45 pasa a ser 47.
2. Eliminar el 47 en el subárbol derecho (cae en caso 1)

Búsqueda y eliminación en ABB

```
void eliminar(NodoAB* &A, int x){
  if (!esVacio(A)) {
    if (A->dato > x)
      eliminar(A->izq, x);
    else if (A->dato < x)
      eliminar(A->der, x);
    else { //A->dato = x
      if (A->der == NULL) {
        NodoAB* aBorrar = A;
        A = A->izq;
        delete aBorrar;
      }
      else if (A->izq == NULL) {
        //Continúa en siguiente diapositiva
```

Búsqueda y eliminación en ABB

```
        }
        else if (A->izq == NULL) {
          NodoAB* aBorrar = A;
          A = A->der;
          delete aBorrar;
        }
        else {
          int suc = minimo(A->der);
          A->dato = suc;
          eliminar(A->der, suc);
        }
      }
    }
}
```

Búsqueda y eliminación en ABB

```
int minimo(NodoAB* A){
  if (!esVacio(A)) {
    NodoAB* cursor = A;
    while (cursor->izq != NULL)
      cursor = cursor->izq
    return cursor->dato;
  }
}
```

Ejercicio 1

Dado un árbol binario de búsqueda de enteros, y un entero X, eliminar todos los menores que X.