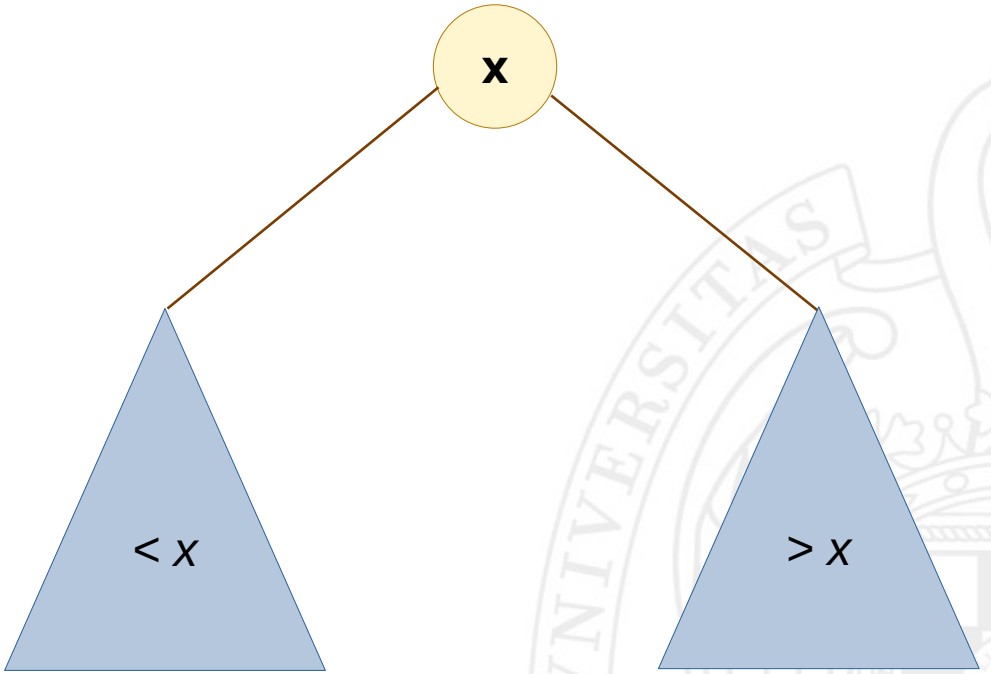
ESTRUCTURAS DE DATOS

TIPOS ABSTRACTOS DE DATOS ARBORESCENTES

Inserción en ABBs

Manuel Montenegro Montes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid

Recordatorio: árboles binarios de búsqueda



A la izquierda están los menores estrictamente de la raíz.

A la izquierda están los mayores estrictamente de la raíz.

Objetivo

- Implementar una función `insert(root, elem)`, que añada un nodo con el valor `elem` al ABB cuya raíz es `root`.

```
void insert(Node *root, const T &elem);
```

- El árbol resultante también ha de ser un ABB.
- Si `elem` ya se encuentra en el ABB, no hace nada.

nodo

elemento.

ABB = ÁRBOL BINARIO DE BÚSQUEDA.

Caso 1: Árbol vacío (`root == nullptr`)

CASO DE ÁRBOL VACÍO.

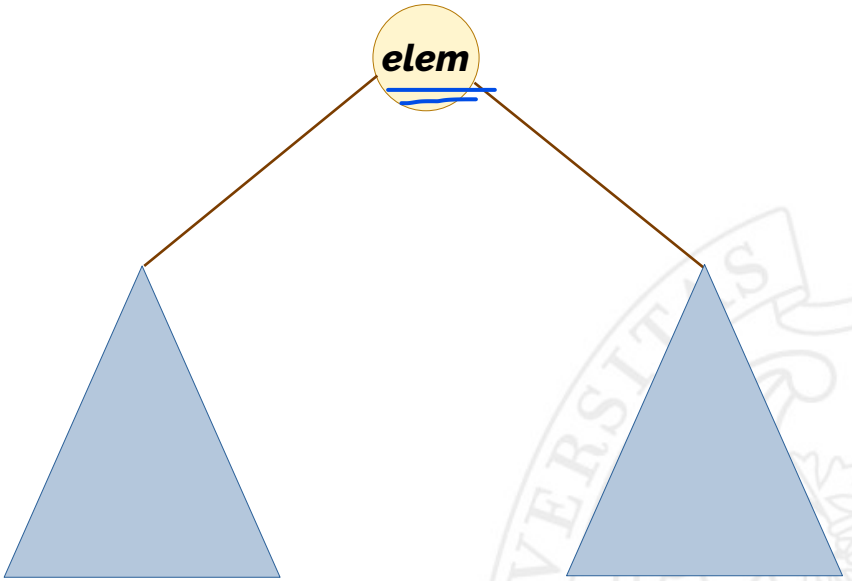
Antes de la inserción

—

Después de la inserción

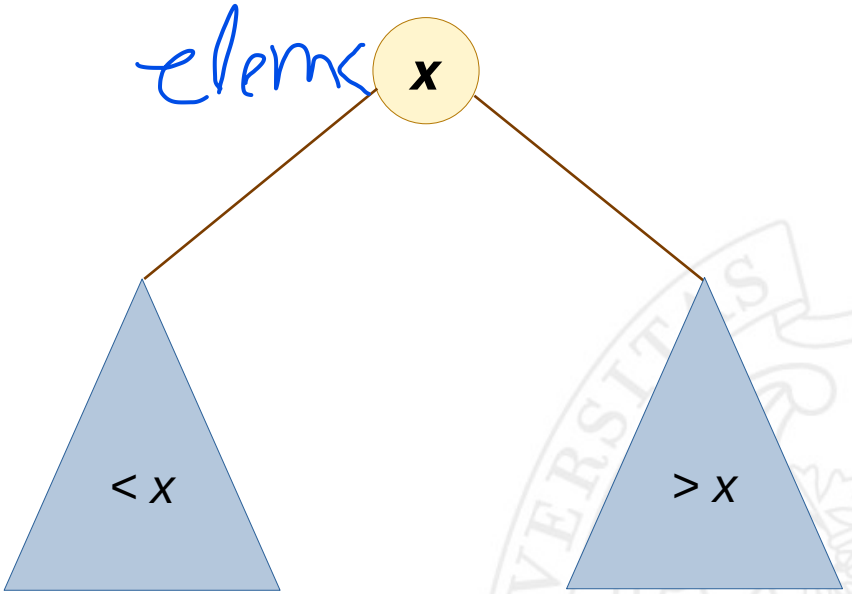
Este nodo va a ser la raíz del árbol.

Caso 2: elem coincide con la raíz



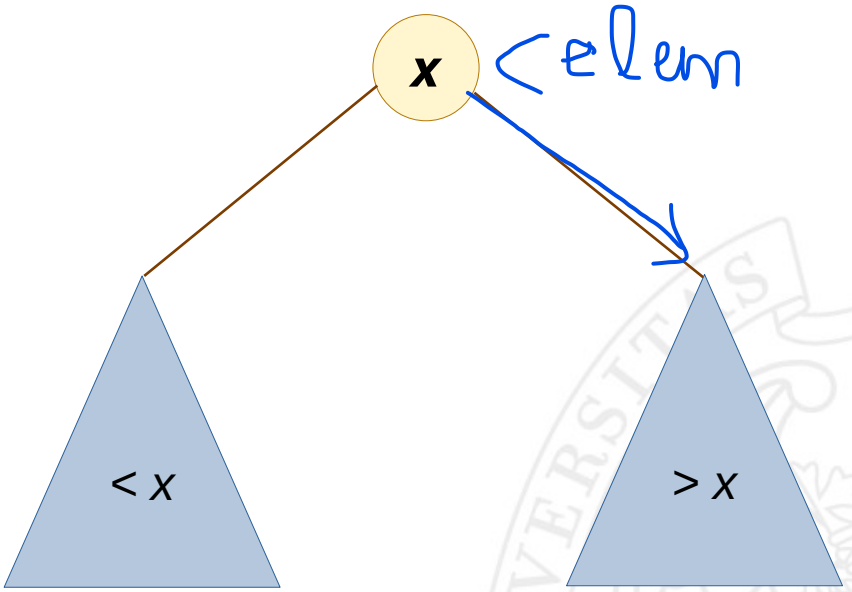
- El elemento que quiero insertar ya está en el árbol. No hacemos nada.

Caso 3: `elem` < raíz



- Insertamos recursivamente `elem` en el hijo izquierdo de la raíz.

Caso 4: `elem` > raíz



- Insertamos recursivamente `elem` en el hijo derecho de la raíz.

Antes de implementar

Al principio root vale nullpointer pero después valdría elem.

- En uno de los casos **la raíz del árbol cambia**.

  Caso 1: si el árbol es vacío, la raíz acaba siendo el nodo recién creado.

- Por tanto, la función `insert` debe devolver también **la nueva raíz del árbol**.
- En lugar de:

```
void insert(Node *root, const T &elem);
```

tendremos:

```
Node * insert(Node *root, const T &elem);
```

Esto lo hacemos porque al añadir un elemento al ABB alguna raíz del árbol puede variar.

Nueva raíz

Ahora devuelve la nueva raíz del árbol tras la inserción.

Implementación

```
Node * insert(Node *root, const T &elem) {
  if (root == nullptr) {
     return new Node(nullptr, elem, nullptr);
  } else if (elem < root→elem) {



  } else if (root→elem < elem) {



  } else {

  }
}
```

elemento izquierda y elemento derecha.

NO tiene ni hijo izquierdo ni hijo derecho.

- **Caso 1:** árbol vacío.

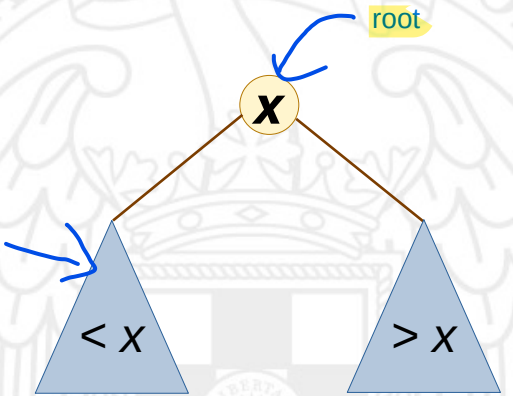
Creamos un nodo con el valor que se quiere insertar, y ese nodo es la nueva raíz del árbol.

Implementación

```
Node * insert(Node *root, const T &elem) {
  if (root == nullptr) {

  } else if (elem < root→elem) {
     Node *new_root_left = insert(root→left, elem);
     root→left = new_root_left;
     return root;
  } else if (root→elem < elem) {



  } else {

  }
}
```

- **Caso 3:** `elem` < raiz

Insertamos en el hijo izquierdo. Conectamos la raíz con la nueva raíz del hijo izquierdo.



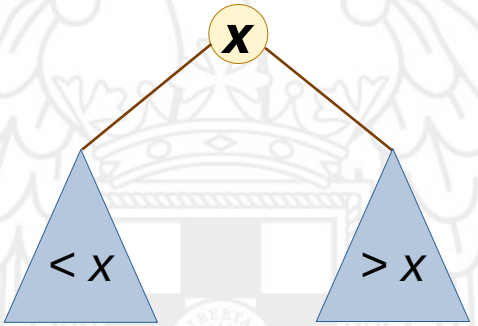
Implementación

```
Node * insert(Node *root, const T &elem) {
  if (root == nullptr) {

  } else if (elem < root→elem) {



  } else if (root→elem < elem) {
     Node *new_root_right = insert(root→right, elem);
     root→right = new_root_right;
     return root;
  } else {

  }
}
```

Análogo a lo que hemos hecho antes.

- **Caso 4:** `elem` > raiz

  Dual al anterior



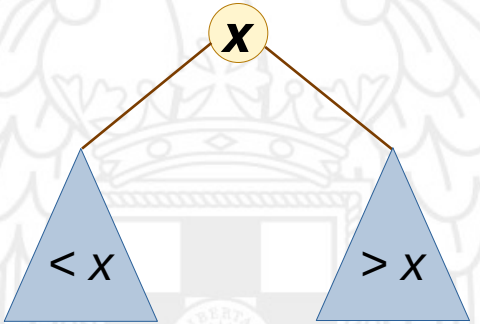
Implementación

```
Node * insert(Node *root, const T &elem) {
  if (root == nullptr) {

  } else if (elem < root→elem) {



  } else if (root→elem < elem) {



  } else {
    return root;
  }
}
```

si el elemento es igual a la raíz devuelve la misma raíz.

- **Caso 2:** `elem` == raiz

  No se hace nada. La raíz no varía.



Implementación

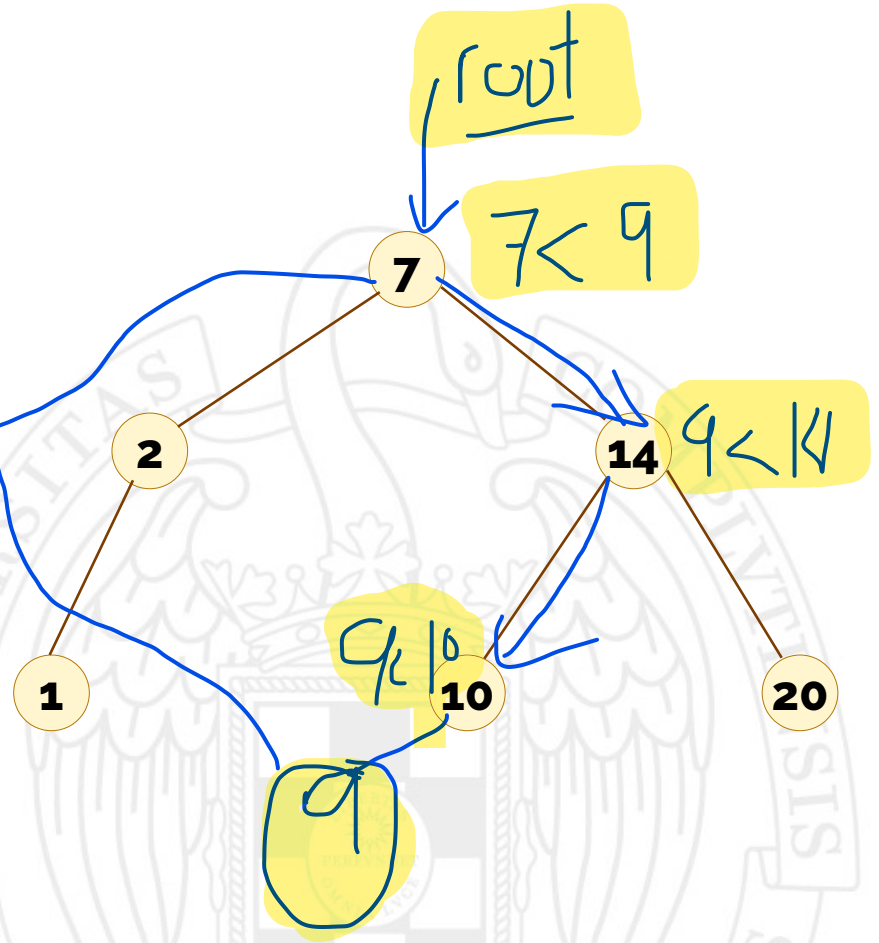
```
Node * insert(Node *root, const T &elem) {
  if (root == nullptr) {
     return new Node(nullptr, elem, nullptr);
  } else if (elem < root→elem) {
     Node *new_root_left = insert(root→left, elem);
     root→left = new_root_left;
     return root;
  } else if (root→elem < elem) {
     Node *new_root_right = insert(root→right, elem);
     root→right = new_root_right;
     return root;
  } else {
     return root;
  }
}
```

IMPORTANTE ESTA IMPLEMENTACIÓN.

Ejemplo

- Insertar el valor **9**

```
Node * insert(Node *root, const T &elem) {
  if (root == nullptr) {
     return new Node(nullptr, elem, nullptr);
  } else if (elem < root→elem) {
     Node *new_root_left = insert(root→left, elem);
     root→left = new_root_left;
     return root;
  } else if (root→elem < elem) {
     Node *new_root_right = insert(root→right, elem);
     root→right = new_root_right;
     return root;
  } else {
     return root;
  }
}
```



Coste en tiempo

- El el caso peor, el nodo se inserta en la rama más larga del árbol.
- Por tanto, si h es la altura del árbol, el coste es $O(h)$.
- Y si n es el número de nodos del árbol:
  - Si el árbol está equilibrado, el coste es $O(\log n)$.
  - Si no, el coste es $O(n)$ en el caso peor.

lineal en la altura del árbol

En el caso de que el árbol no sea equilibrado entonces el coste en el caso peor es lineal respecto al número de nodos.