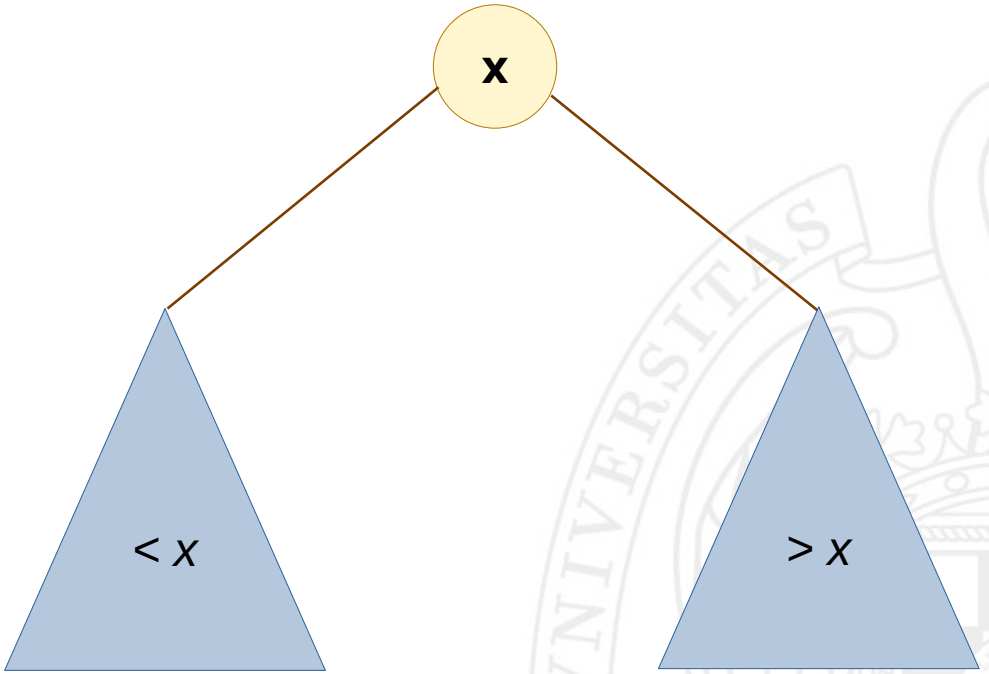
ESTRUCTURAS DE DATOS

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



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.

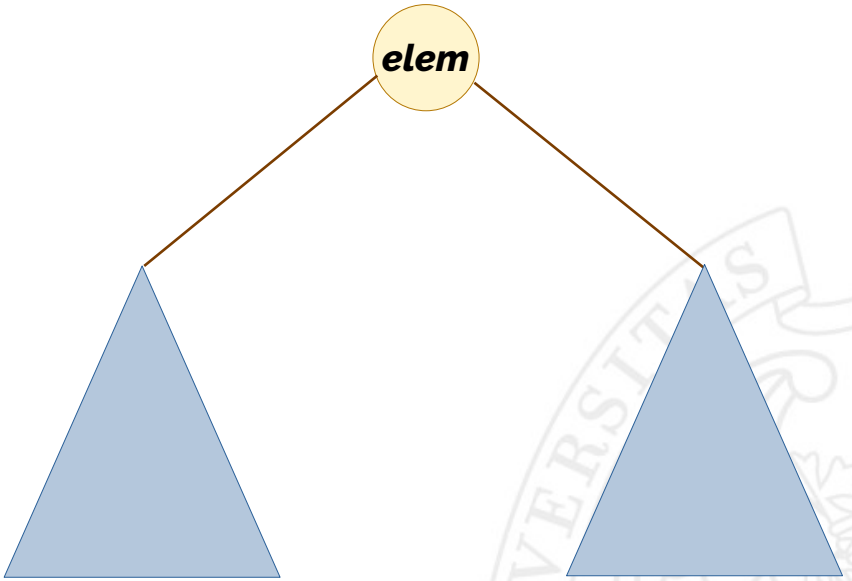
Caso 1: Árbol vacío (`root == nullptr`)

Antes de la inserción

—

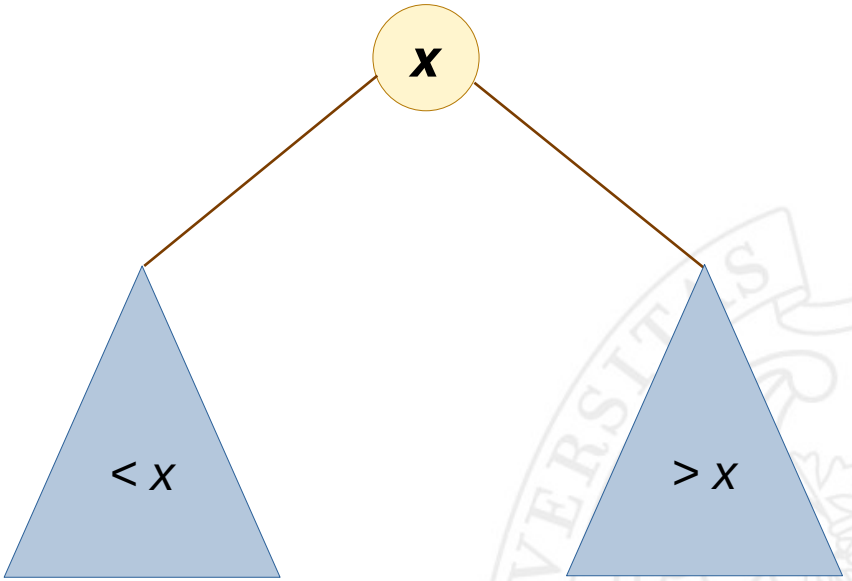
Después de la inserción

Caso 2: `elem` coincide con la raíz



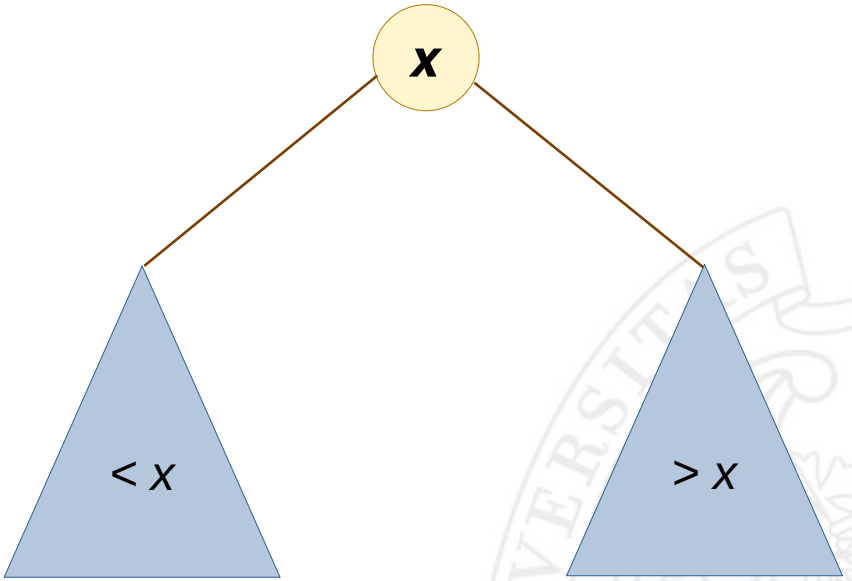
- El elemento que quiero insertar ya está en el árbol. No hacemos nada.

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

- 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);
```

Nueva raíz

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 {

  }
}
```

- **Caso 1:** árbol vacío.

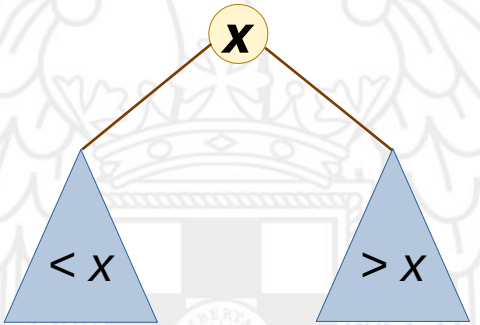
  Creamos un nodo con el valor que se quiere insertar, y ese nodo es la nueva raíz del árbol.

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.

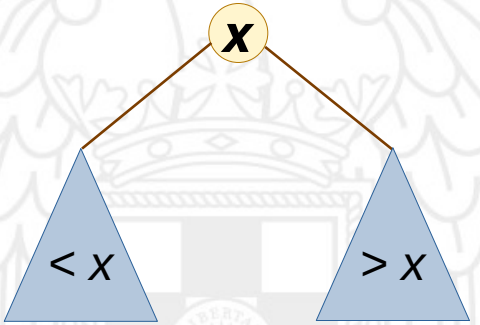


Implementación

```
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 {

  }
}
```

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

  Dual al anterior

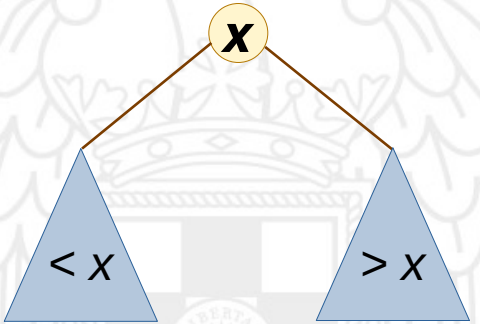


Implementación

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

  } else if (elem < root→elem) {



  } else if (root→elem < elem) {



  } else {
     return root;
  }
}
```

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

  No se hace nada. La raíz no varía.



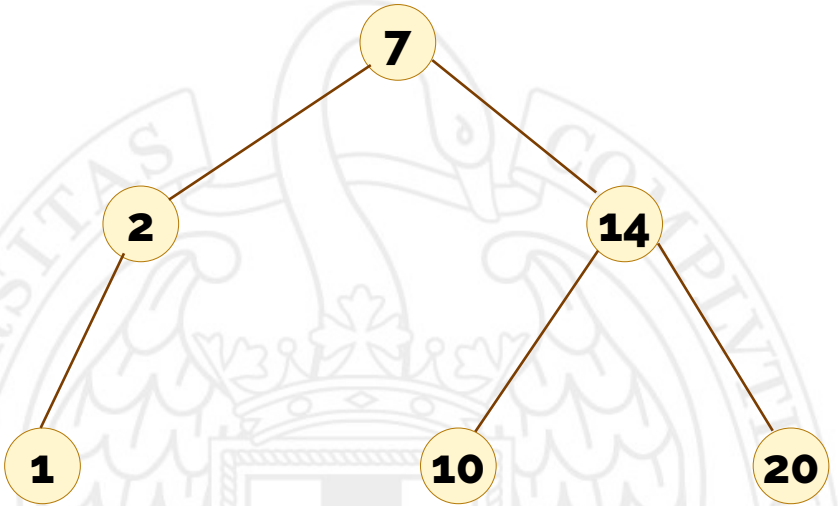
Implementación

```
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;
  }
}
```

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.