

# Sessió 12. Més exercicis d'Arbres

**Estructura de Dades**  
**Curs 2020-2021**

Grau en Enginyeria Informàtica  
Facultat de Matemàtiques i Informàtica,  
Universitat de Barcelona

# Contingut

Problema 1 – Construir AVL

Problema 2 – Construir AVL

Problema 3 – printInterval

Problema 4 – sonIdentics

Problema 5 – getLeafs

# Problema 1

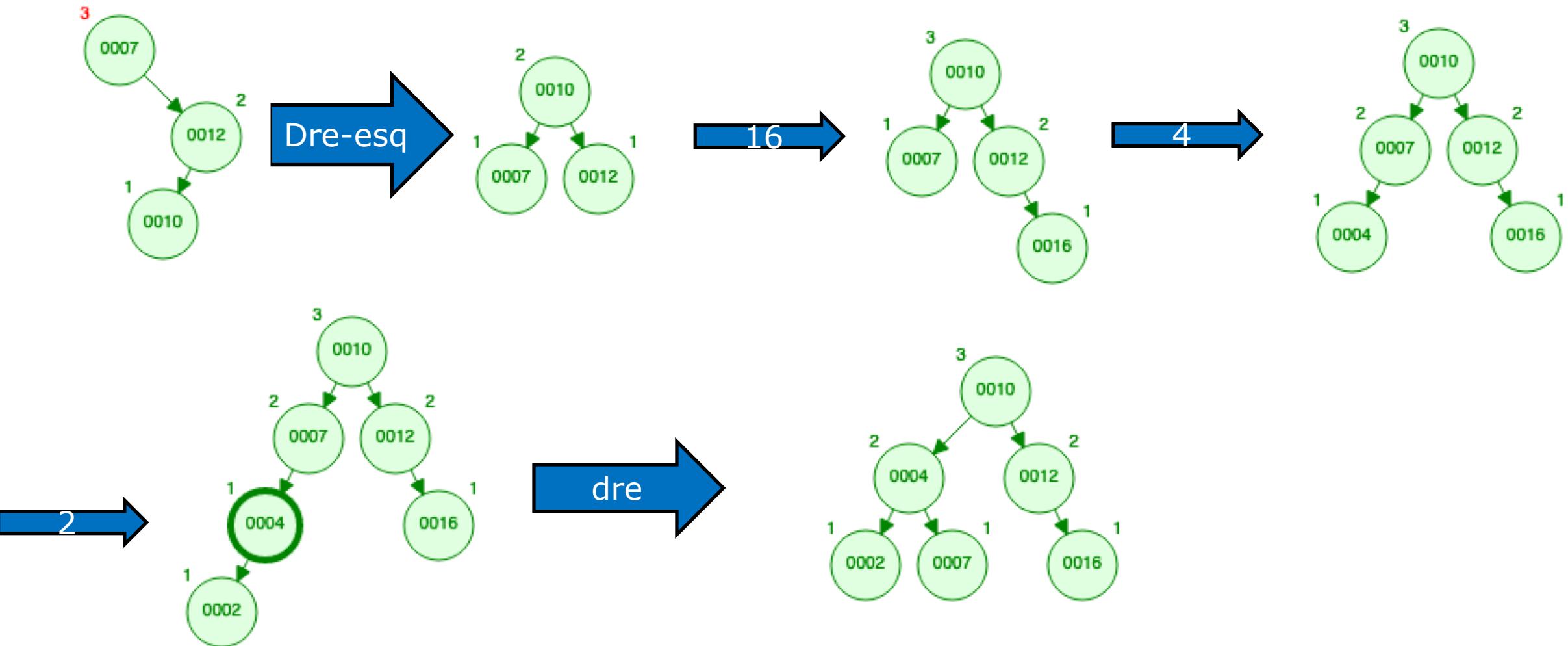
A partir d'un arbre AVL inicialment buit. Inseriu els elements següents en l'ordre especificat:

7, 12, 10, 16, 4, 2

Feu el dibuix de la inserció de tots els elements i indiqueu quina rotació esteu fent en cada moment (si n'hi ha).

# Problema 1 (Solució final)

**7, 12, 10, 16, 4, 2**



# Problema 2

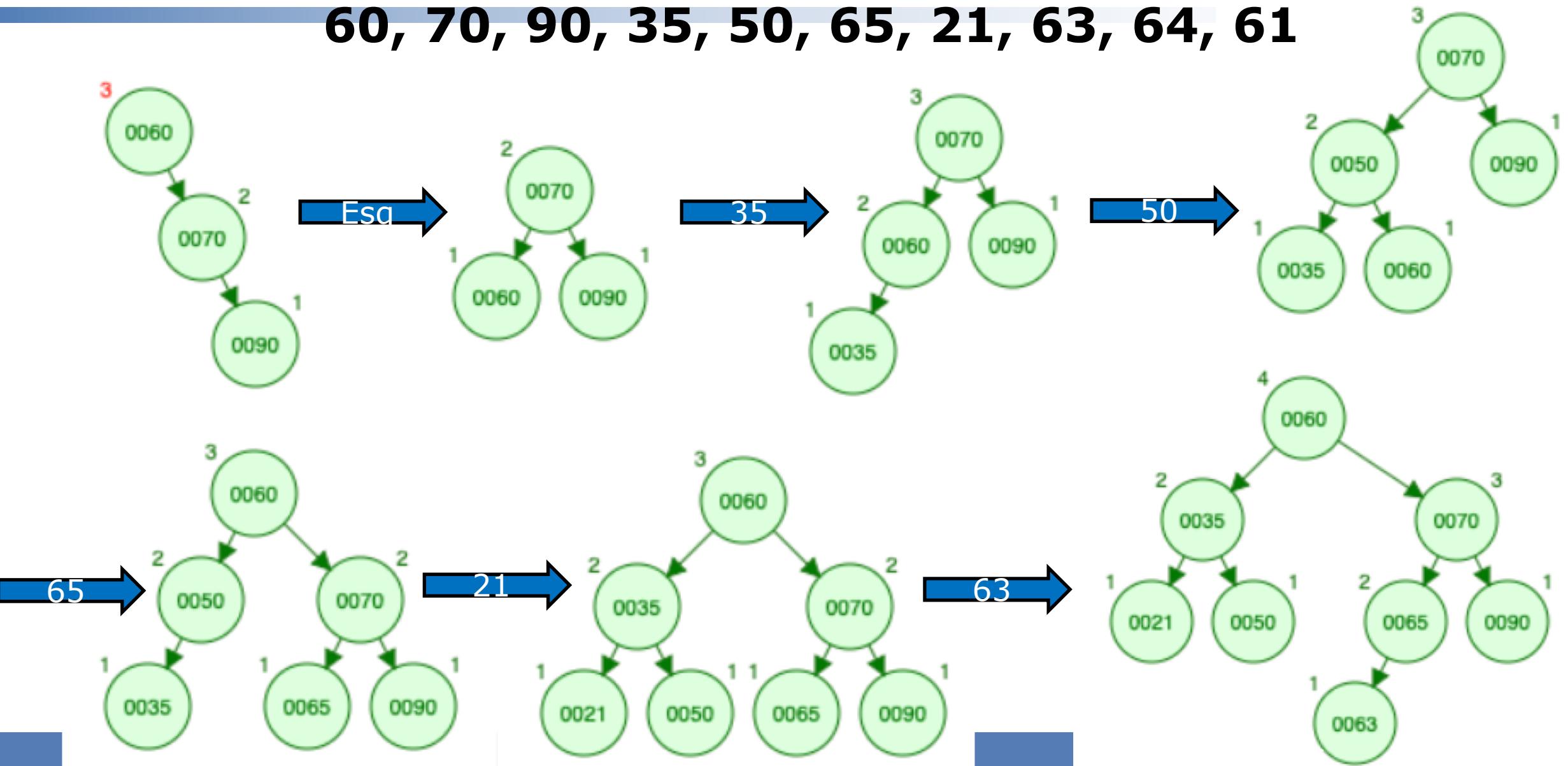
A partir d'un arbre AVL inicialment buit. Inseriu els elements següents en l'ordre especificat:

60, 70, 90, 35, 50, 65, 21, 63, 64, 61

Feu el dibuix de la inserció de tots els elements i indiqueu quina rotació esteu fent en cada moment (si n'hi ha).

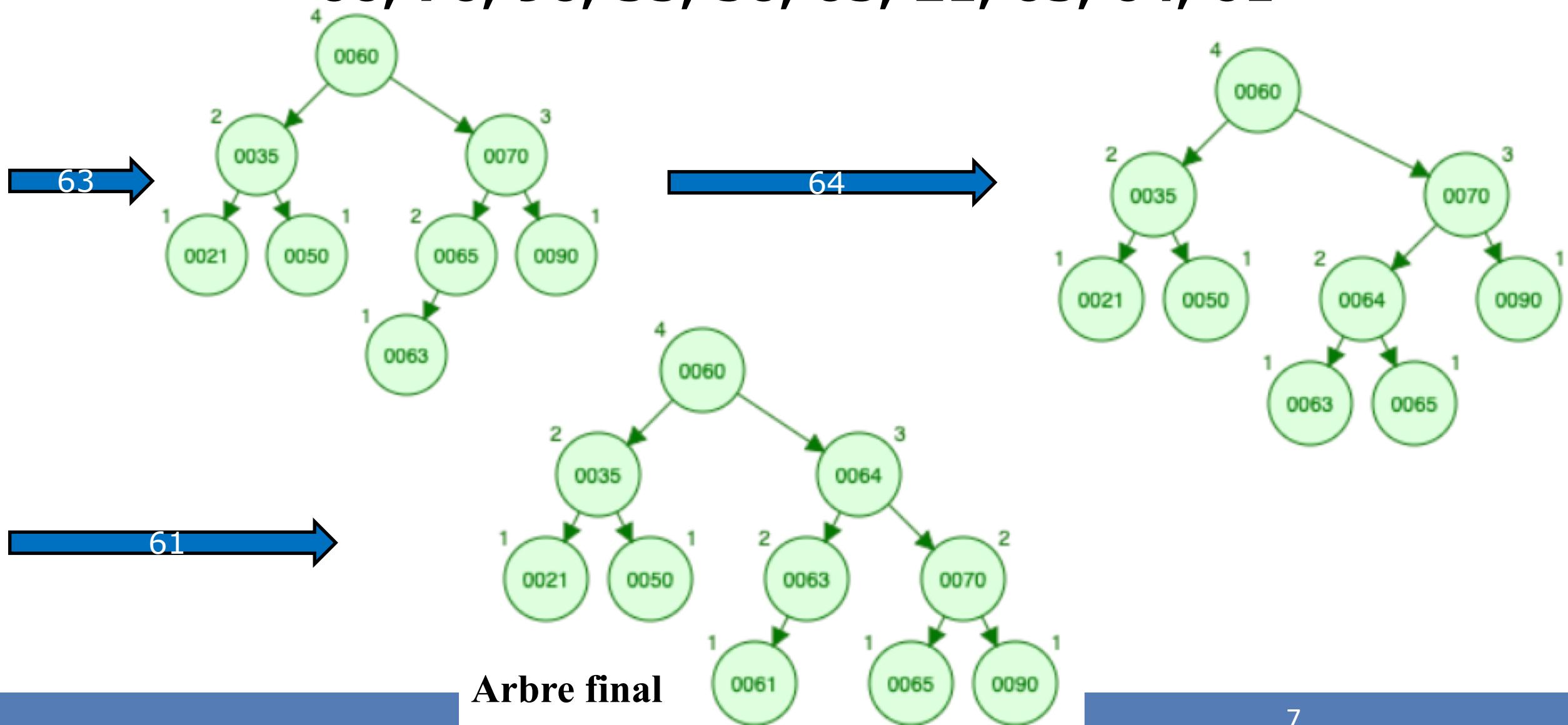
# Problema 2 (Solució final)

**60, 70, 90, 35, 50, 65, 21, 63, 64, 61**



# Problema 2 (Solució final)

60, 70, 90, 35, 50, 65, 21, 63, 64, 61



# Problema 3

Implementeu recursivament un mètode anomenat `printInterval()` de la classe `BSTree` (un arbre de cerca binària). Aquest mètode imprimeix per pantalla tots els elements de l'arbre binari que són més grans o iguals a `from` i més petits o iguals a `to`. Suposeu que l'arbre és d'enters i que el `BSTree` té definit una funció `root()` que retorna el `root_node` que és de tipus `NodeTree *`.

Els `NodeTree` tenen un atribut `_right`, un atribut `_left` i un atribut `_element` privats que s'accedeixen mitjançant les funcions `right()`, `left()`, `element()`, i també teniu una funció `isExternal()` per retornar si un node és fulla o no.

```
template <class Element>

void BSTree<Element>::printInterval(Element from, Element to)
{
    Aquí el vostre codi
}
```

Definiu aquí les funcions auxiliars que necessiteu

# Problema 3 (Solució)

```
template <class Element>
void BSTree<Element>::printInterval(Element from, Element to)
{
    if (empty()) { // Message or throw an exception
        std::cout << "This tree is empty!!! " << "\n";
    } else {
        this->printFromTo(from, to, this->root_node);
    }
}
```

Solució no  
optimitzada !!

# Problema 3 (Solució 1)

```
template <class Element> void BSTree<Element>::printFromTo(Element from,
Element to, NodeTree<Element>* p)
{
    if (p == nullptr) return;
    else {
        if (p->element() < from) this->printFromTo(from, to, p->right());
        else if (p->element() > to) this->printFromTo(from, to, p->left());
        else {
            this->printFromTo(from, to, p->left());
            std::cout << p->element() << " , ";
            this->printFromTo(from, to, p->right());
        }
    }
}
```

Solució no  
optimitzada !!

# Problema 4

Implementeu recursivament un mètode anomenat **sonIdentics()** de la classe **BSTree** (un arbre de cerca binària). Aquest mètode diu si dos arbres són iguals. Suposeu que l'arbre és d'enters i que el BSTree té definit una funció **root()** que retorna el **root\_node** que és de tipus **NodeTree \***.

Els NodeTree tenen un atribut **\_right**, un atribut **\_left** i un atribut **\_element** privats que s'accedeixen mitjançant les funcions **right()**, **left()**, **element()**, i també teniu una funció **isExternal()** per retornar si un node és fulla o no.

```
template <class Element>
void BSTree<Element>::sonIdentics(BSTree& arbre)
{
    Aquí el vostre codi
}
```

Definiu aquí les funcions auxiliars que necessiteu

# Problema 4 (Solució)

```
template <class Element>
bool BSTree<Element>::sonIdentics(BSTree& a)
{
    if (!this->empty() && !a.empty())
    {
        return sonIdentics(this->root(), a.root());
    }
    else return false;
}

template <class Element>
bool BSTree<Element>::sonIdentics(NodeTree<Element>* x, NodeTree<Element>* y)
{
    if (x == nullptr && y == nullptr)
        return true;

    return (x && y) && (x->getElement() == y->getElement()) &&
           sonIdentics(x->left(), y->left()) &&
           sonIdentics(x->right(), y->right());
}
```

# Problema 5

Implementeu recursivament un mètode anomenat `getLeafs()` de la classe `BSTree` (un arbre de cerca binària). Aquest mètode retorna les fulles de l'arbre en un vector. Suposeu que l'arbre és d'enters i que el `BSTree` té definit una funció `root()` que retorna el `root_node` que és de tipus `NodeTree *`.

Els `NodeTree` tenen un atribut `_right`, un atribut `_left` i un atribut `_element` privats que s'accedeixen mitjançant les funcions `right()`, `left()`, `element()`, i també teniu una funció `isExternal()` per retornar si un node és fulla o no.

```
template <class Element>
vector<Element> BSTree<Element>::getLeafs() const
{
    Aquí el vostre codi
}
```

Definiu aquí les funcions auxiliars que necessiteu

# Problema 5 (Solució)

```
template <class Element>
vector<Element> BSTree<Element>::getLeafs() const
{
    vector<Element> leafs;
    _getLeafs(this->root(), leafs);
    return leafs;
}

template <class Element>
void BSTree<Element>::_getLeafs(NodeTree<Element>* node,
                                    vector<Element>& leafs) const
{
    if (node->isExternal()) // It is a leaf
    {
        leafs.push_back(node->element());
    }
    else {
        if (node->left() != nullptr) _getLeafs(node->left(), leafs);
        if (node->right() != nullptr) _getLeafs(node->right(), leafs);
    }
}
```