

Sessió 11. Exercicis d'Arbres

Estructura de Dades
Curs 2020-2021

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

Contingut

Problema 1 – Recorreguts

Problema 2 – Construir AVL

Problema 3 – max recursiu

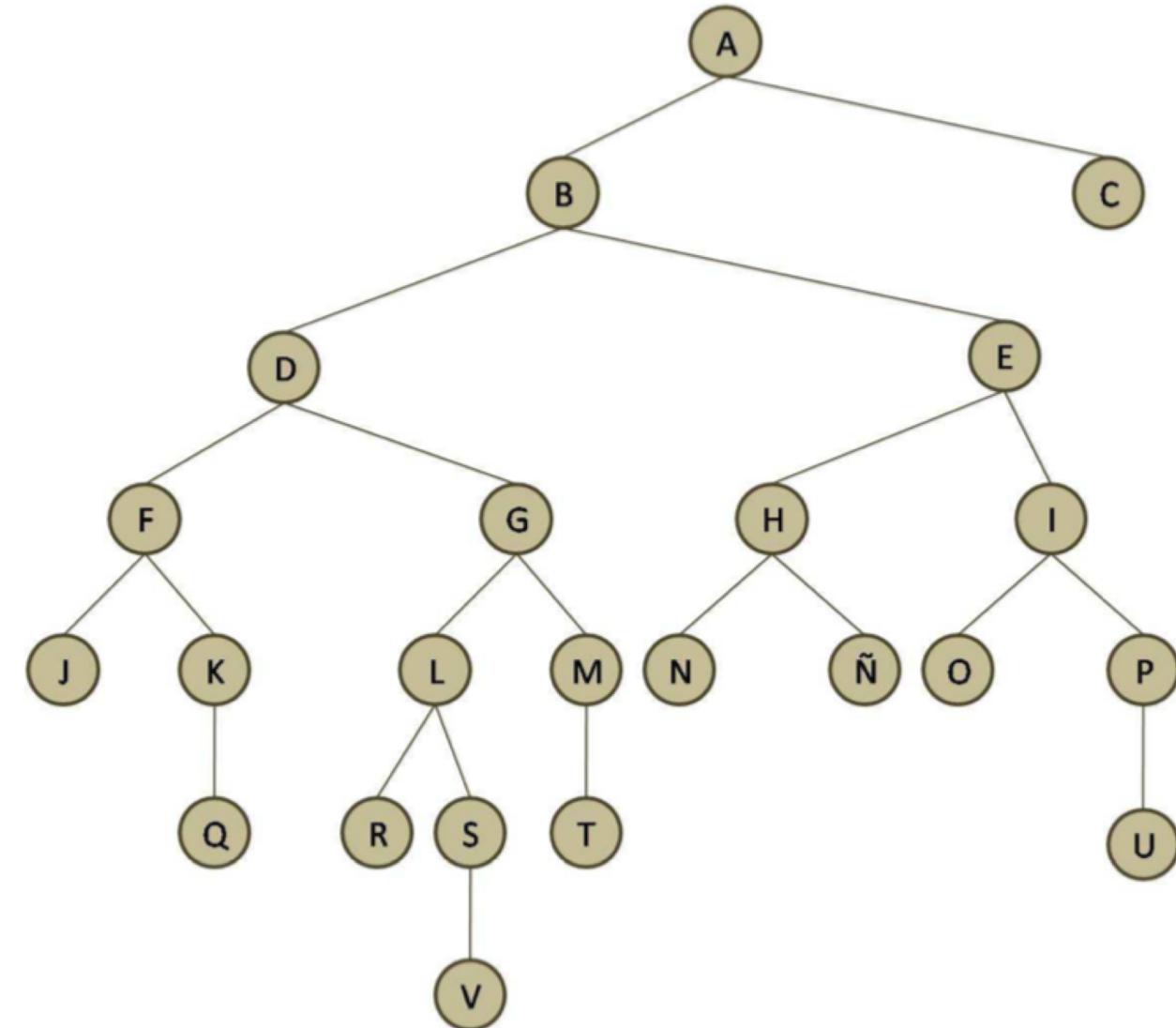
Problema 4 – max iteratiu

Problema 5 – heightNodeValue recursiu

Problema 6 – depthNodeValue recursiu

Problema 1

Indica pel següent arbre quin és el seu recorregut en preordre, inordre i postordre.

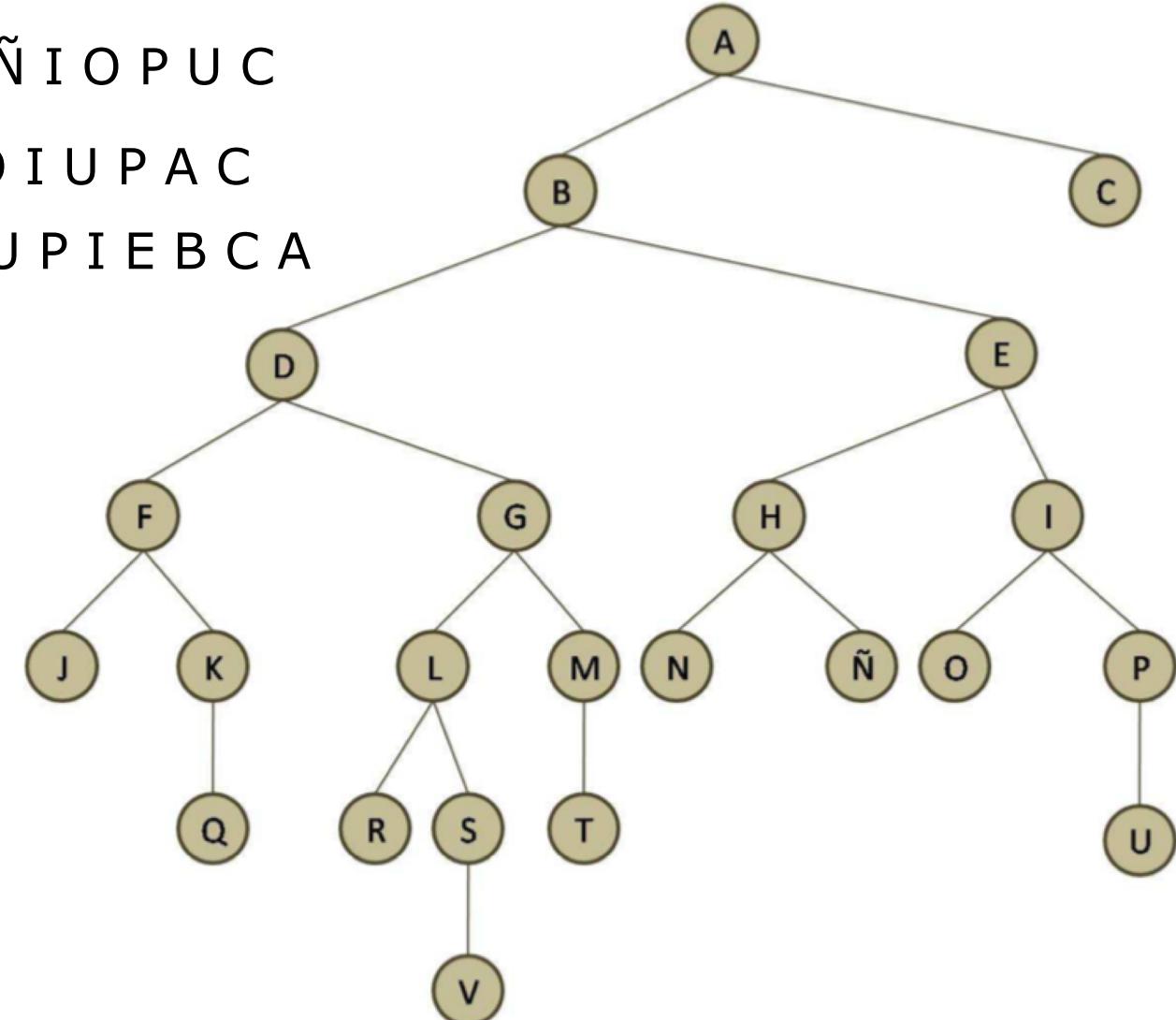


Problema 1 (Solució)

Preordre: A B D F J K Q G L R S V M T E H N Ñ I O P U C

Inordre: J F Q K D R L V S G T M B N H Ñ E O I U P A C

Postordre: J Q K F R V S L T M G D N Ñ H O U P I E B C A



Problema 2

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

14, 6, 24, 35, 59, 17, 21, 32, 4, 7, 15, 22

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)

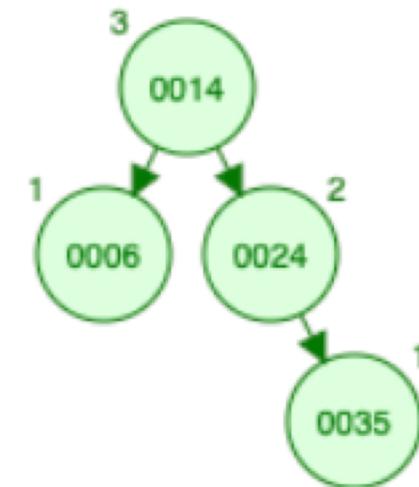
A partir d'un arbre AVL inicialment buit. Inseriu els elements següents en l'ordre especificat:

14, 6, 24, 35, 59, 17, 21, 32, 4, 7, 15, 22

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

Inserit fins al 35

No s'ha fet cap rotació



Problema 2 (Solució)

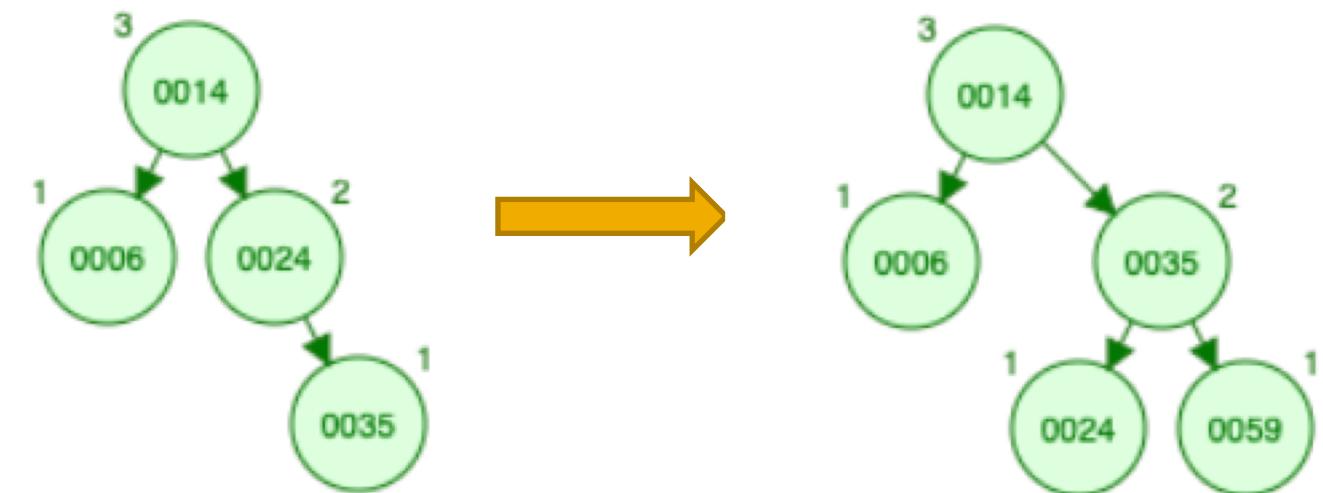
A partir d'un arbre AVL inicialment buit. Inseriu els elements següents en l'ordre especificat:

14, 6, 24, 35, 59, 17, 21, 32, 4, 7, 15, 22

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

Inserit fins al 35

Ara toca inserir el 59. Desequilibra l'arbre
i cal fer una rotació simple a la esquerra



Problema 2 (Solució)

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

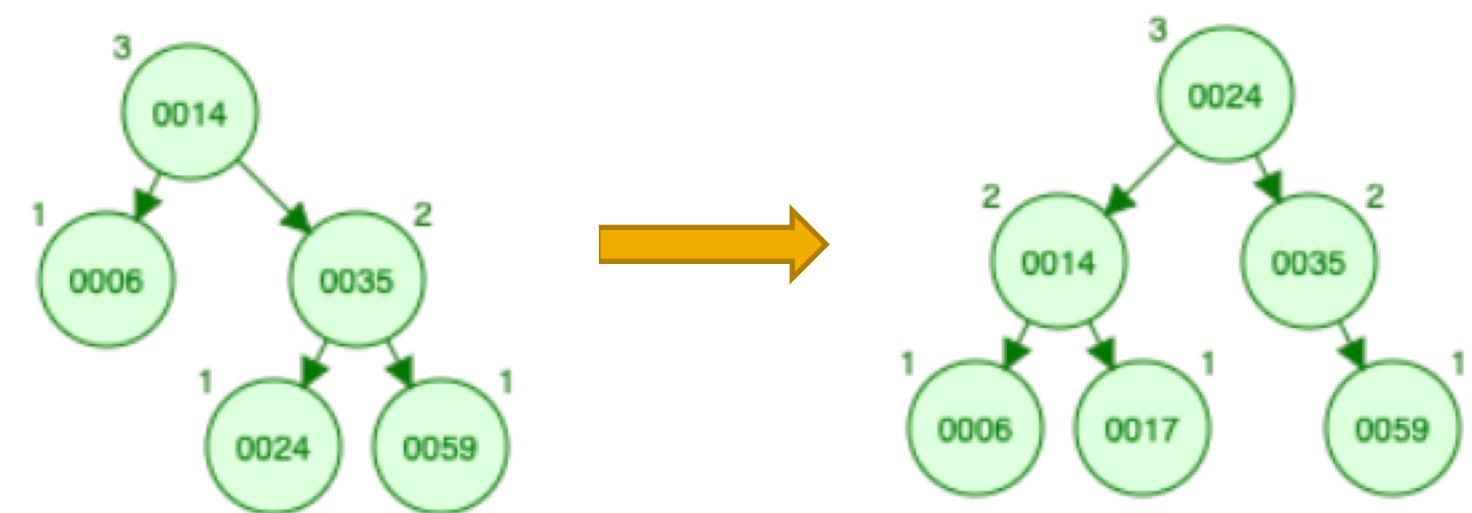
14, 6, 24, 35, 59, 17, 21, 32, 4, 7, 15, 22

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

Inserit fins al 59

Ara toca inserir el 17.

Desequilibra l'arbre
i cal fer una rotació doble a
dreta-esquerra



Problema 2 (Solució)

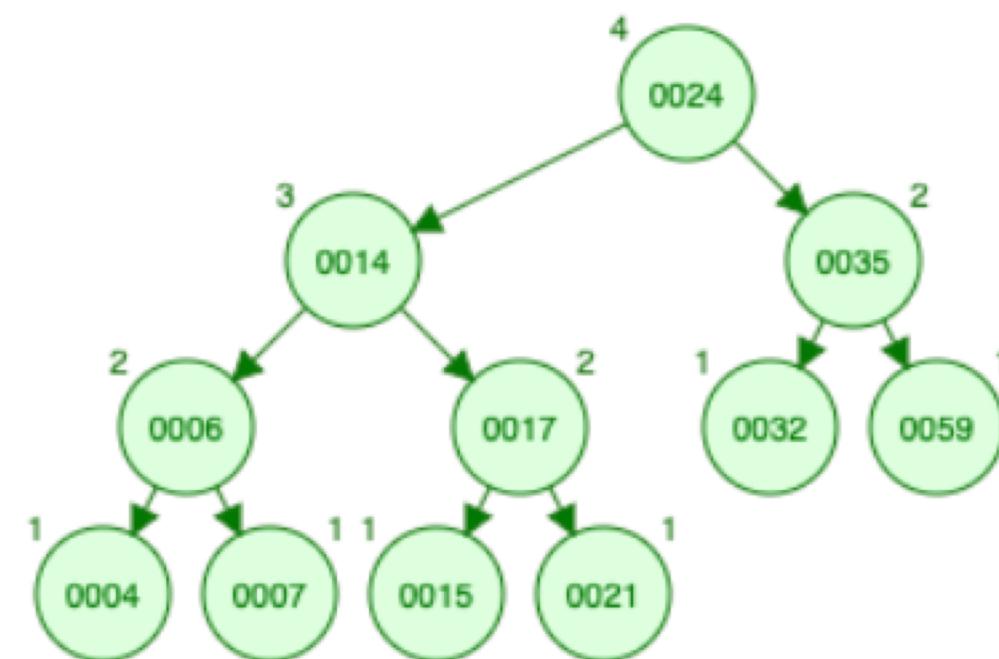
A partir d'un arbre AVL inicialment buit. Inseriu els elements següents en l'ordre especificat:

14, 6, 24, 35, 59, 17, 21, 32, 4, 7, 15, 22

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

Inserit fins al 15

**Ara toca inserir el 22. Desequilibra
l'arbre ...
i cal fer una rotació doble**



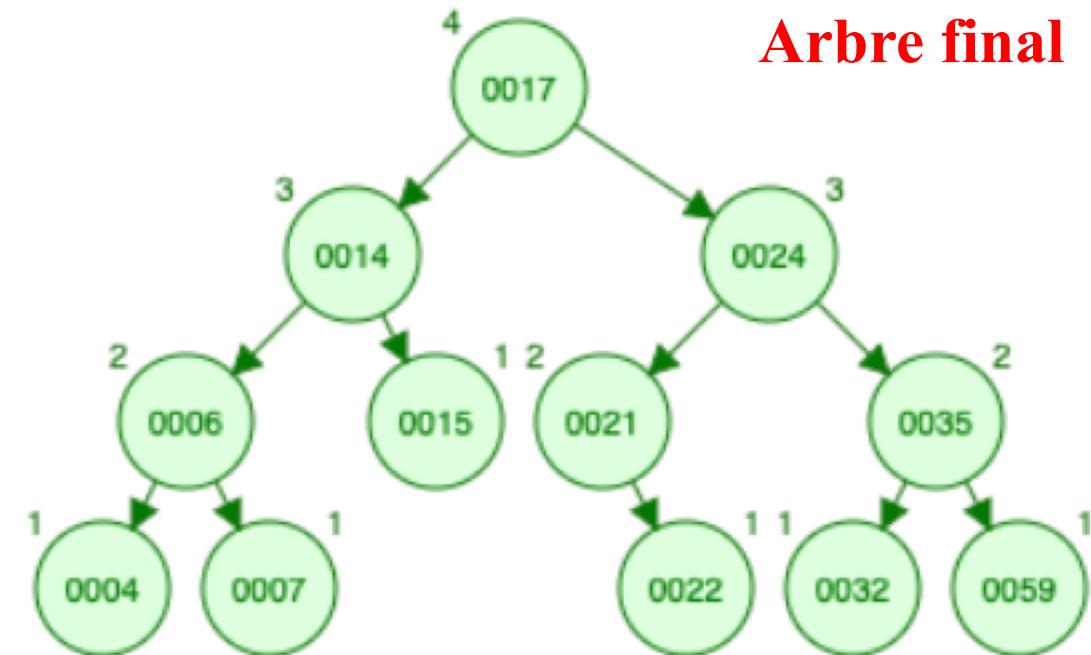
Problema 2 (Solució final)

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

14, 6, 24, 35, 59, 17, 21, 32, 4, 7, 15, 22

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

**Quan s'insereix el 22. Desequilibra
l'arbre
i cal fer una rotació doble
a esquerra-dreta**



Problema 3

Implementeu **recursivament** un mètode anomenat `int max()` de la classe `BSTree` (un arbre de cerca binària). Aquest mètode calcula el valor màxim que hi ha l'arbre binari. Supposeu que l'arbre és d'enters i que el BSTree té definit un `NodeTree * root_node`.

Els NodeTree tenen un atribut `_right`, un atribut `_left` i un atribut `_element` privats que s'accedeixen amb les funcions `right()`, `left()`, `getElement()`

```
template <class Element>
```

```
int BSTree<Element>::max () {
```

Aquí va el vostre codi

```
}
```

Definiu aquí les funcions auxiliars que necessiteu

Problema 3 (Solució)

```
1. template <class Element>
2. int BSTree<Element>::max () {
3.     return this->locateMax (this->root_node) ;
4. }
```



```
5. template <class Element>
6. int BSTree<Element>::locateMax (NodeTree<Element>* p)
7. {
8.     if (p->right () != nullptr) return locateMax (p->right ());
9.     else return p->element ();
10.}
```

La solución està feta amb templates però no feina falta.

Problema 4

Implementeu **ITERATIVAMENT** un mètode anomenat `int max()` de la classe `BSTree` (un arbre de cerca binària). Aquest mètode calcula el valor màxim que hi ha l'arbre binari. Supposeu que l'arbre és d'enters i que el `BSTree` té definit un `NodeTree * root_node`.

Els `NodeTree` tenen un atribut `_right`, un atribut `_left` i un atribut `_element` privats que s'accedeixen amb les funcions `right()`, `left()`, `getElement()`

```
template <class Element>
int BSTree<Element>::max() {
    Aquí va el vostre codi
}
```

Problema 4 (Solució)

```
1. template <class Element>
2. int BSTree<Element>::max () {
3.     NodeTree<Element> *temp = root_node;
4.     while (temp->right() != nullptr)
5.         temp = temp->right();
6.     return temp->getElement();
7. }
```

Problema 5

Implementeu **recursivament** un mètode anomenat `heightNodeValue(int key)` de la classe `BSTree` (un arbre de cerca binària). Aquest mètode retorna l'alçada del node que conté la `key` en l'arbre binari. Supposeu que l'arbre és d'enters i que el `BSTree` té definit un `NodeTree * root_node`.

Els `NodeTree` tenen un atribut `_right`, un atribut `_left` i un atribut `_element` privats que s'accedeixen amb les funcions `right()`, `left()`, `getElement()`

```
template <class Element>
```

```
int BSTree<Element>::heightNodeValue(Element key) {
```

Aquí va el vostre codi

```
}
```

Definiu aquí les funcions auxiliars que necessiteu

Problema 5 (Solució)

```
template<class Element>
int BSTree<Element>::heightNodeValue(Element key)
{ return heightValue(this->root_node, key); } // Aquí falta llençar excepció arbre buit.

template <class Element>
int BSTree<Element>::heightValue(NodeTree<Element>* p, Element ele )
{ int h;
  if (p == nullptr) h = 0;
  else {
    if (p->element() == ele) { h = heightNode(p); } //heightNode calcula height d'una position
    else {
      if (p->element() > ele ) h= heightValue(p->left(), ele);
      else h = heightValue( p->right(), ele);
    }
  }
  return h;
}
```

Solució no
optimitzada !!

Problema 6

Implementeu un mètode **RECURSIU** anomenat `depthNodeValue(int Key)` de la classe `BSTree` (un arbre de cerca binària). Aquest mètode retorna la profunditat o nivell del node que conté la `key` en l'arbre binari. Suposeu que l'arbre és d'enters i que el `BSTree` té definit un `NodeTree` * `root_node`.

Els `NodeTree` tenen un atribut `_right`, un atribut `_left` i un atribut `_element` privats que s'accedeixen mitjançant les funcions `right()`, `left()`, `getElement()`

```
template <class Element>  
  
int BSTree<Element>::depthNodeValue(Element key)
```

{

Aquí va el vostre codi

}

Definiu aquí les funcions auxiliars que necessiteu

Problema 6 (Solució 1)

```
template <class Element>
int BSTree<Element>::depthNodeValue(Element key) {    return depthValue(this->root_node, key, 0); }

template <class Element> int BSTree<Element>::depthValue(NodeTree<Element>* p, Element ele, int h ){
    int d ;
    if (p == nullptr) d = -1;
    else {
        if (p->element() == ele) { d = h; }
        else{   h++;
            if (p->element() > ele ) d = depthValue(p->right(), ele, h);
            else d= depthValue( p->left(), ele, h);
        }
    }
    return d;
}
```

Solució no optimitzada,
Es pot fer de moltes altres
maneres!

Problema 6 (Solució 2)

```
template <class Element>
int BSTree<Element>::depthnodeValue(Element key) {
    return depthValue(this->root_node, key, 0);
}
```

Solució no optimitzada,
Es pot fer de moltes altres
maneres!

```
template <class Element>
int BSTree<Element>::depthValue(NodeTree<Element>* p, Element ele, int h ){
    if (p == nullptr) return -1;
    else if (p->element() == ele) {return h; }
    else if (p->element() > ele ) return depthValue(p->right(), ele, h+1);
    else return depthValue( p->left(), ele, h+1);
}
```