UT 4 - PD 3

Ejercicio # 1

```
Clase Nodo

insertar ( nodo ) → boolean
    If nodo.etiqueta == Raiz.etiqueta then
        Retunn false ;  // Elemento duplicado
    end if
    If nodo.etiqueta < Raiz.etiqueta then
        If Raiz.hijoIzq == null then
            Raiz.hijoIzq ← nodo
            Retunn true ;  // inserción con éxito
        else
            Raiz.hijoIzq.insertar(nodo)  // Recursivo
        end if
    else
        If nodo.etiqueta > Raiz.etiqueta then
            If Raiz.hijoDer == null then
            {Raiz.hijoDer ← nodo
            Retunn true ;}
            else
                Raiz.hijoDer.insertar(nodo)
            end if
    end if
```

Complejidad: Si está balanceado $O(\log(n))$
sino $O(n)$

Clase Arbol

```
insertar ( Comparable etiqueta , T Dato ) → Bool
    If (Arbol Vacio then
        Raiz ← new Nodo ( etiqueta , dato )
        Return true ;
    end if

    Return Raiz.insertar ( etiqueta , dato )
```

(2)

Metodo para contar todas las hojas

```
hojas ( )                                   Clase nodo
    int hojas = 0 ;
    If raiz.esHoja() then
        hojas ++ ;
    end if
    If Raiz.GetHijoHizq != null then
        Raiz.hijoIzq.hojas ()
    end if
    If Raiz.GetHijoDer != null then
        Raiz.hijoDer.hojas ()
    end if

    Return hojas
```

Pre: . Raiz no null

Post: . Se devuelve un contador con el número exacto de hojas en el Árbol
. El Árbol no se modifica

Big O ? O(n) ; pues se recorre todo el Arbol.

Clase Anbol

hojas ( )
    If Anbol Vacío then
        Ennor : "El Anbol está Vacío"
    end if

    Retunn Raíz . hojas ( ) ;
end

---

(3) Suma de las claves de todos los elementos — Clase NODO

CalcularSuma ( ) → int
    int suma = ConvertToInt (Raíz . Clave) ;
    if (Raíz . HijoIzq != null) then
        suma = suma + Raíz . HijoIzq . CalcularSuma ( )
    end if
    If (Raíz . HijoDer != null) then
        suma += Raíz . HijoDer . CalcularSuma ( )
    end if

    Return suma

Pre: Nodo o Raíz no nulo

Post: Devuelve un contador que va sumando las claves de los nodos.

BiG O : O(n) ; Recorres todo el árbol

Clase
Arbol

```
SumaClaves ( ) → int
    If (Raiz == null) then
        Return 0;

    Return Raiz.CalcularSuma ()
```

(A) Metodo nodos en un nivel DADO

Clase Nodo

```
nodosPorNivel ( nodo, int nivelBusc, int nivelActual ) → int
    If (nodo == null) then
        Return 0;
    end if
    If (nivelActual == nivelBuscado) then
        Return 1;
    end if

    int izq = nodosPorNivel ( nodo.hijoIzq, nivelBuscado, NvActual+1)
    int der = nodoPorNivel (nodo.hijoDer, nivel buscado, NivelActual+1)
    Return (izq + der)
```

Pre: Nodo raiz no es hoja *

Post: Devuelve la suma de los niveles de los nodos de CADA subárbol

Big O. O(n), Recorres todo el Arbol

* nivel ∧ buscar > 0.

Clase Arbol

```
NodoNivel ( int nivelBuscado )
    If (nivelBuscado < 0) then
        Eanon : "No puede ser"
    End if

    Retuan Raiz.NodosPorNivel ( Raiz , nivelBuscado, 0)
end
```