UT4 - PD3

Ejercicio 1:

Lenguaje Natural - El metodo debe de comenzar por el Nodo que se quiere agregar, evaluado de que no este agregado ya en el arbol.

Luego, valida el arbol y compara si el valor del nodo que se quiere agregar, es mayor o menor que la raiz. Si es menor compara con el hijo izquierdo y vuelve a preguntar lo mismo de antes, hasta que llegue a ser null el hijo izquierdo o derecho, dependiendo de donde se quiera agregar. Hace lo mismo en el caso de que sea mayor.

Precondiciones: - validar que el nodo no exista antes ya.
- verificar que el arbol sea distinto de nulo

Postcondiciones: El arbol debe mantenerse en equilibrio tras la inserción del nodo.

El arbol debe incrementar su tamaño (cant de nodos)

Pseudocodigo: Inicio NodoArbol
entero dato
Cadena nombre
NodoArbol hijoIzquierdo, hijoDerecho

Constructor (d, nom)
-
dato = d
nombre = nom
hijoIzquierdo = nulo
hijoDerecho = nulo

```
Inicio AgregarNodoArbolBinario (d, nom)
    raíz ← nulo
    entero d
    string nom
    Nuevo = Crear NodoArbol (d, nom)
    si (raíz == nulo) hacer
        raíz = nuevo
    Sino
        NodoArbol auxiliar = raíz
        NodoArbol Padre
        Mientras (verdadero)
            Padre = auxiliar
            Si (d < auxiliar.dato) hacer
                auxiliar = auxiliar.hijoIzquierdo
                Si (auxiliar == nulo)
                    Padre.hijoIzquierdo = nuevo
                    devolver
                Fin Si
            Sino
                auxiliar = auxiliar.hijoderecho
                Si (auxiliar == null)
                    padre.hijoDerecho = nuevo
                    devolver
                Fin Si
            Fin Si
        Fin Si
    Fin Mientras
Fin Si
Imprimir Arbol
Fin Algoritmo
```

Orden del Tiempo de Ejecución del Algoritmo

Al ser un arbol binario, el orden del tiempo de ejecución del Algoritmo, es de O(log n) si esta balanceado, si no esta balanceado esta es O(n).

Ejercicio 2

Contar todas la hojas que tiene el Árbol.

Lenguaje Natural: Lo que deberia hacer es un programa que vaya verificando si cada nodo tiene hijos, en la izquierda, en la derecha o en ambas. Si tiene hijos no se la considera hoja, si no tiene hijos, se la considera hoja por lo que aumenta el contador en 1, partiendo desde 0.

Precondiciones: El arbol no este vacío.
Tener un contador para contar las hojas

Postcondiciones: Devolver un contador.

Pseudocodigo: Inicio Contar Hojas (contador i)

```
i ← 0
Nodoactual
si Nodoactual es nulo
        devolver 0

Imprimir "verificando nodo con valor:", nodo.valor
tiene_Izquierda = "si" si nodo.Izquierdo no es nulo sino "no"
tiene_derecho = "si" si nodo.derecho no es nulo sino "no"
Imprimir "Hijo Izquierdo", tiene_Izquierdo
         "Hijo derecho", tiene_derecho
```

hojas_Izquierdo = contar_hojas y verificar_hijos (nodo.Izquierdo)
|| derecho = contar_hojas y verificar_hijos (nodo.derecho)

es_hoja = 1 si nodo.Izquierdo es nulo y nodo.derecho es nulo sino 0

devolver cont_hojas = hojas_Izquierdo + hojas_derecho + es_hoja

Imprimir 'cont_hojas

Orden de Ejecución de Algoritmo

El orden de Ejecución de este Algoritmo, es en $O(n)$, en donde n es la cantidad de nodos del arbol.