S2-Árboles

Número de elementos de un árbol - Básico

Número de elementos de un árbol - Básico

Ficheros requeridos: TamañoArbol.py (Descargar)
Tipo de trabajo: Individual

Dado un árbol como parámetro de una función, retornar la cantidad de nodos que este tiene:
Entrada:
Un árbol como parametro
Salida:
Un entero que indica la cantidad de nodos en el árbol

Ficheros requeridos

TamañoArbol.py

```
1  class Node:
2      def __init__(self, val=0, left=None, right=None):
3          self.val = val
4          self.left = left
5          self.right = right
6
7  def TamañoArbol(root):
8      #codigo aqui
```

***S2* - Buscar en un BST - Sencillo**

S2 - Buscar en un BST - Sencillo

Ficheros requeridos: BuscarBST.py (Descargar)
Tipo de trabajo: Individual

A diferencia de los árboles binarios , los árboles binarios balanceados (en inglés, BST) son muy fáciles de programar: Cuando buscas un elemento tienes que ir bajando por el árbol decidiendo si ir a la derecha o a la izquierda según si el elemento que buscas es mayor o menor al elemento encontrado en el momento.

Entrada

Un árbol binario de búsqueda como parámetro de una función

Salida

retornar un YES si el valor se encuentra o un NO en caso contrario

Ficheros requeridos

BuscarBST.py

```
class Node:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def BuscarBST(root,val):
    #codigo aqui
```

***S2* - Insertar en un BST - Sencillo**

S2 - Insertar en un BST - Sencillo

Ficheros requeridos: InsertarBST.py (Descargar)
Tipo de trabajo: Individual

A diferencia de los árboles binarios balanceados, los árboles binarios no balanceados son muy fáciles de programar: cuando insertas un nuevo elemento, vas bajando por el árbol (hacia el subárbol izquierdo si el elemento a insertar es menor que el del nodo actual y hacia el subárbol derecho en caso contrario) hasta colocar el elemento en el primer lugar que esté libre.

Entrada

Un arbol binario de busqueda y un entero, todo como parametros

Salida

Retornar la raiz del arbol luego de ser modificado

Ficheros requeridos

InsertarBST.py

```
class Node:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def InsertarBST(root,val):
    #codigo aqui
```

S2 - Decir si dos árboles son iguales - Sencillo

S2 - Decir si dos árboles son iguales - Sencillo

Ficheros requeridos: ArbolesIguales.py (Descargar)
Tipo de trabajo: Individual

Queremos saber si dos árboles son iguales dadas sus raíces.

Entrada

Dos arboles como parámetros de una función.

Salida

Retornar True o False dependiendo si son o no iguales.

Ficheros requeridos

ArbolesIguales.py

```
class Node:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def ArbolesIguales(root1,root2):
    #codigo aqui
```

S2 - Vista superior de un árbol - Intermedio

S2 - Vista superior de un árbol - Intermedio

Ficheros requeridos: VistaSup.py (Descargar)
Tipo de trabajo: Individual

Dado un puntero a la raíz de un árbol binario, retornar la vista superior del árbol binario.

El árbol, visto desde la parte superior de los nodos, se denomina vista superior del árbol.

Vista superior : $1 -> 2 -> 5 -> 6$

Completa la función vistaSup y retorne los valores resultantes en una sola línea separada por espacio.

Entrada:

Se da la función vistaSup(root) donde root es la raíz del árbol.

Restricciones:

1<=nodos en el árbol <=500

Salida:

Retorna los valores en una cadena que tenga una sola línea con los valores separados por espacios.

Ejemplo:

Entrada

Salida

1 2 5 6

Nota:

- Recuerde que el ejercicio es totalmente INDIVIDUAL.
- El ejercicio debe solucionarse con ÁRBOLES.
- La raíz root se pasa como parámetro a la función que debes completar.

Ficheros requeridos

VistaSup.py

```
class Node:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def vistaSup(root):
    # codigo aqui
```

S2 - De árbol binario a lista enlazada - Intermedio

S2 - De árbol binario a lista enlazada - Intermedio

Ficheros requeridos: ArbolAListaa.py (Descargar)
Tipo de trabajo: Individual

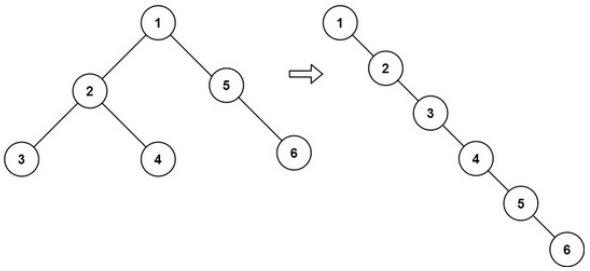
Dada la raíz de un árbol binario convertirlo en una lista enlazada teniendo en cuenta que:

- La lista enlazada debe utilizar la misma clase TreeNode donde el puntero hijo derecho apunta al siguiente nodo de la lista y el puntero hijo izquierdo es siempre nulo.
- La lista enlazada debe estar en el mismo orden que un recorrido de preorden del árbol binario.

Nota1: NO tienes que retornar nada, solo modificar

Nota2: La raíz se te da como parámetro

Ejemplo 1:



Input

root = [1,2,5,3,4,null,6]

Output

[1,null,2,null,3,null,4,null,5,null,6]

Ficheros requeridos

ArbolAListaa.py

```
class Node:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def ArbolAListaa(root: Node) -> None:

```