**Recupera el árbol**

**22/04/2024**

**Presentado por**

**Santiago Botero**

**Santiago Hurtado**

**Oscar Merino**

**Problema**

Se debe construir un árbol binario bst-avl dada su configuración en pre-orden e in-orden

**Entrada**

La entrada consta de 2 líneas.

La primera es la configuración pre-orden del árbol.

La segunda es la configuración in-orden

**Salidas**

La salida es una línea que corresponde a la configuración pos-orden del árbol binario

**Estrategia**

La estrategia a utilizar fue utilizar la clase Nodo para referencias a su hijo izquierdo y derecho.

Además de esto, se dividió el problema en subpoblemas, construyendo tanto el árbol izquierdo como el derecho recursivamente.  
Finalmente, se concatenan los resultados para obtener el árbol resultante.

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| Pre-orden ““  In-orden “12345” | Pre-orden vacío | “” |
| Pre-orden “12345 In-orden “” | In-orden vacío | “” |
| Pre-orden “1” In-orden “1” | Pre-orden e in-orden con longitud 1 | “1” |

**from** **sys** **import** stdin

**class** **Nodo**:

**def** **\_\_init\_\_**(self, valor):

self.valor = valor

self.izquierda = None

self.derecha = None

**class** **ArbolBinario**:

**def** **\_\_init\_\_**(self):

self.raiz = None

**def** **construir\_arbol**(self, preorden, inorden):

**if** **not** preorden **or** **not** inorden:

**return** None

valor\_raiz = preorden[**0**]

nodo\_raiz = Nodo(valor\_raiz)

indice\_raiz\_inorden = inorden.index(valor\_raiz)

preorden\_izquierda = preorden[**1**:indice\_raiz\_inorden + **1**]

preorden\_derecha = preorden[indice\_raiz\_inorden + **1**:]

inorden\_izquierda = inorden[:indice\_raiz\_inorden]

inorden\_derecha = inorden[indice\_raiz\_inorden + **1**:]

nodo\_raiz.izquierda = self.construir\_arbol(preorden\_izquierda, inorden\_izquierda)

nodo\_raiz.derecha = self.construir\_arbol(preorden\_derecha, inorden\_derecha)

**return** nodo\_raiz

**def** **posorden**(self, nodo):

**if** nodo **is** None:

**return** ''

**return** self.posorden(nodo.izquierda) + self.posorden(nodo.derecha) + nodo.valor

**def** **main**():

entrada = stdin.readline().strip().split()

preorden = entrada[**0**]

inorden = entrada[**1**]

arbol = ArbolBinario()

raiz = arbol.construir\_arbol(preorden, inorden)

posorden\_resultante = arbol.posorden(raiz)

**print**(posorden\_resultante)

**if** \_\_name\_\_ == "\_\_main\_\_":

main()