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()
```