0. Práctica árboles

TUIA - Programación 2

Recursión

Practica 3

Árboles Binarios

A lo largo de esta práctica trabajaremos con la siguiente estructura, observe lo similar que es a `Node` de Listas Enlazadas, pero en lugar de `next`, posee `left` y `right`.

```
class Tree:
  def __init__(self, cargo, left = None, right = None):
    self.cargo = cargo
    self.left = left
    self.right = right
```

Ejercicio 1

Dibuje ejemplos de árboles en su hoja con las siguientes características, luego, construya sus ejemplos en Python

1. Un árbol con únicamente su raíz.
2. Un árbol parecido a una lista de largo 3.
3. Un árbol completo de altura 1.
4. Un árbol vacío ¿Puede hacerlo?

Ejercicio 2

Implemente en la clase Tree los siguiente métodos:

Ayuda: pensar que cada árbol tiene a su izquierda y derecha objetos árboles como sus hijos.

- **nodos**: devuelve la cantidad de nodos del árbol
- **menor_mayor**: devuelve el menor y el mayor elemento del árbol en una tupla
- **buscar**: busca si un elemento está o no en el árbol
- **altura**: calcula la altura del árbol, la distancia desde la raíz hasta la hoja más lejana

Ejercicio 3

a. Pensar y dibujar un ejemplo de árbol en papel, escribir los resultados de PreOrder, InOrder y PostOrder

b. Implementar los recorridos PreOrder, InOrder y PostOrder como funciónes recursivas, verificar sus resultados

c. Implementar los recorridos PreOrder, InOrder y PostOrder como funciónes iterativas, verificar sus resultados

Ayuda: Para las versiones iterativas, necesitará utilizar una Pila como estructura de datos adicional. Puede importar una implementacion cualquiera de Pila que haya realizado en la Practica anterior.

Ejercicio 4

Escriba una función `copiar` que reciba un árbol y devuelva un *nuevo* árbol idéntico al original.

Ejercicio 5

Escriba una función `invertir` que reciba un árbol binario e intercambie los hijos derechos por los izquierdos de todos los nodos.

Ejercicio 6

Escriba una función `sumatoria` que reciba un árbol binario que contiene números en sus nodos y devuelva la suma de todos los nodos.

Ejercicio 7

Escriba una función que reviba un árbol binario A cuyos nodos contienen números y que dado un entero M, una clave inicial `inicio` y una clave final `final`, calcula la suma de todos los números del árboles que se encuentren entre `inicio` y `final` y a lo sumo en el nivel M.

Árboles Binarios de Búsqueda

Ejercicio 8

Dibuje un árbol binario de búsqueda de palabras, con al menos 5 palabras, utilizando orden de diccionario (lexicográfico). Acomódelo como más le guste, mientras sea correcto. Luego indique en qué lugar del árbol se insertaría la palabra `python`.

Ejercicio 9

Utilizando la misma clase Tree de la sección anterior, implemente otra clase llamada BSTree que herede de esta, reimplemente los métodos **menor_mayor**, **buscar** e implemente un nuevo método llamado **insertar** que inserte un elemento.

Ayuda: puede optar por definir métodos menor y mayor **internamente** por separado para hacer la implementación más sencilla, pero no es estrictamente necesario.

Ejercicio 10

Escriba una función `combinar` que combine dos árboles binarios de búsqueda en uno solo. El resultado tambien debe ser un árbol binario de búsqueda.

Ayuda: quizás resulte conveniente implementar una función de copia pero para BSTree.

Ejercicio 11

Escriba una función `borrar_raiz`. Dado un árbol binario de búsqueda, esta función deberia devolver un nuevo árbol binario de búsqueda que contenga los mismos datos, a excepcion de la raiz.

Ejercicio 12

Escriba una función `borrar_valor` que dado un árbol binario de búsqueda y un valor, devuelva un árbol binario de búsqueda sin ese valor.

Ejercicios adicionales

Ejercicio 13

En la unidad anterior mencionamos cómo utilizamos notación postfija de expresiones para evaluar en un Stack y sin usar paréntesis. Con árboles podemos representar expresiones infijas sin paréntesis. Cada nodo interno del árbol representa un operador, izquierda y derecha son subexpresiones, y las hojas son números. Implementar una clase Expression que herede de Tree, un árbol de expresiones infijas, con dos métodos.

- **imprimir**: que imprime la expresión de forma infija con paréntesis.
- **evaluar**: evalúa todo el árbol y lo reduce a un número.

Ejemplo

```
     *
   /   \
  +     -     ==>   ((1 + 2) * (3 - 4))   ==>   -2
 / \   / \
1   2 3   4
```

Ejercicio 14

¿Se podrían representar los árboles (pensados como TAD) utilizando una estructura de datos contigua - por ejemplo un arreglo? ¿Porque?