Estructuras Discretas 2021-2
Práctica 5: Árboles binarios

Facultad de Ciencias, UNAM

Fecha de entrega: 10 de diciembre del 2021

Resuelva los siguientes ejercicios, la práctica es **parejas**. En el archivo `arboles.py` se encuentra la firma correspondiente para cada función que debe completarse con código. Después de haber resuelto todos los ejercicios, puede ejecutar el comando `python3 -m unittest -v test_arboles.py` dentro de la carpeta para asegurarse que la salida de cada función es correcta; es importante mencionar que el pasar todas las pruebas no asegura que la implementación de todas las funciones sea correcta (por ejemplo, el haber usado funciones ya definidas en `Python` que resuelvan directamente el problema es un error). Sin embargo, las pruebas serán útiles para darse cuenta si alguna función es incorrecta.

Para esta práctica únicamente `es_vacio` y `es_hoja` son no recursivas, **todas las demás funciones deben ser recursivas**, de otro modo no se considerarán correctas.

Para la entrega, siga las especificaciones publicadas en `Classroom`.

1. Defina la función recursiva `__repr__` que devuelve una representación en cadena de un árbol binario.

   Ejemplo:

```
>>> t = Arbol()
>>> t
ø
>>> t1 = Arbol(1)
>>> t1
(1
  ø
  ø)
>>> t2 = Arbol(2)
>>> t3 = Arbol(3, t1, t2)
>>> t3
(3
  (1
    ø
    ø)
  (2
    ø
    ø))
```

2. Defina la función `es_vacio` que devuelve `True` si el árbol es vacío, y `False` en otro caso.

   Ejemplo:

```
>>> t = Arbol()
>>> t.es_vacio()
True
>>> t1 = Arbol(1)
>>> t1.es_vacio()
False
>>> t1.izquierdo.es_vacio()
True
```

3. Defina la función `es_hoja` que devuelve `True` si el árbol es no vacío y ambos hijos son vacíos, y `False` en otro caso.

   Ejemplo:

```
>>> t = Arbol()
>>> t.es_hoja()
False
>>> t1 = Arbol(1)
>>> t1.es_hoja()
```

```
True
>>> t2 = Arbol(2, t1, t1)
>>> t2.es_hoja()
False
>>> t2.izquierdo.es_hoja()
True
```

4. Defina la función recursiva `copia` que devuelve un árbol idéntico a este.

   Ejemplo:

```
>>> t = Arbol()
>>> o = t.copia()
>>> o.es_vacio()
True
>>> t1 = Arbol(1)
>>> t2 = Arbol(2)
>>> t3 = Arbol(3, t1, t2)
>>> o3 = t3.copia()
>>> o3.raiz
3
>>> o3.izquierdo
(1
  ø
  ø)
>>> o3.derecho
(2
  ø
  ø)
>>> t3 is o3
False
>>> t3.izquierdo is o3.izquierdo
False
>>> t3.derecho is o3.derecho
False
```

5. Defina la función recursiva `num_nodos` que devuelve el número de nodos de un vértice.

   Ejemplo:

```
>>> t = Arbol()
>>> t.num_nodos()
0
>>> t1 = Arbol(1)
>>> t1.num_nodos()
1
>>> t2 = Arbol(2, t1, t1)
>>> t2.num_nodos()
3
>>> t3 = Arbol(3, t1, t2)
>>> t3.num_nodos()
5
```

6. Defina la función recursiva `direccion`, que recibe un elemento y devuelve la dirección (en forma de cadena binaria) del primer nodo en un recorrido `in-order` que contiene al elemento, o `False` en caso de que el elemento no ocurra en el árbol.

   Ejemplo:

```
>>> t1 = Arbol(1)
>>> t2 = Arbol(2)
>>> t3 = Arbol(1, t1, t2)
>>> t4 = Arbol(1, t1, t1)
>>> t5 = Arbol(1, t4, t3)
>>> t6 = Arbol(1, t4, t4)
>>> t7 = Arbol(1, t5, t6)
>>> t8 = Arbol(1, t6, t5)
>>> t4.direccion(2)
False
>>> t7.direccion(2)
'011'
```

```
>>> t8.direccion(2)
'111'
```

7. Defina la función recursiva `gira`, que recibe una dirección (en forma de cadena binaria) y devuelve una copia del árbol en la que el subárbol que tiene como raíz al nodo dado por la dirección recibida fue girado (su subárbol izquierdo y derecho fueron intercambiados). Si la dirección no corresponde a un nodo del árbol, simplemente devuelve una copia del árbol.

   Ejemplo:

```
>>> t1 = Arbol(1)
>>> t2 = Arbol(2)
>>> t3 = Arbol(3, t1, t2)
>>> t4 = Arbol(4, t2, t1)
>>> t5 = Arbol(5, t3, t4)
>>> t6 = Arbol(6, t4, t5)
>>> t6.gira("10")
(6
  (4
    (2
      ø
      ø)
    (1
      ø
      ø))
  (5
    (3
      (2
        ø
        ø)
      (1
        ø
        ø))
    (4
      (2
        ø
        ø)
      (1
        ø
        ø))))
>>> t6.gira("0000")
(6
  (4
    (2
      ø
      ø)
    (1
      ø
      ø))
  (5
    (3
      (1
        ø
        ø)
      (2
        ø
        ø))
    (4
      (2
        ø
        ø)
      (1
        ø
        ø))))
```

8. Defina recursivamente la función `es_isomorfo`, que recibe a un árbol como entrada, y devuelve `True` si el árbol es isomorfo a la entrada (uno puede obtenerse a partir del otro aplicando una sucesión de giros), y `False` en otro caso.

   Ejemplo:

```
>>> t1 = Arbol(1)
```

```
>>> t2 = Arbol(2)
>>> t3 = Arbol(3)
>>> t4 = Arbol(1, t1, t2)
>>> t5 = Arbol(1, t2, t1)
>>> t6 = Arbol(1, t3)
>>> t7 = Arbol(1, None, t3)
>>> t8 = Arbol(1, t4, t7)
>>> t9 = Arbol(1, t5, t6)
>>> t8.es_isomorfo(t9)
True
```

9. Defina recursivamente la función `__eq__`, que recibe a un árbol como entrada, devuelve `True` si el árbol es isomorfo a la entrada, y `False` en otro caso.

   Ejemplo:

```
>>> t = Arbol()
>>> o = Arbol()
>>> t == o
True
>>> t1 = Arbol(1)
>>> o1 = Arbol(1)
>>> t2 = Arbol(2)
>>> o2 = Arbol(2)
>>> t3 = Arbol(3)
>>> o3 = Arbol(3)
>>> t3 == o3
True
>>> t3 == o2
False
```