**Control de lectura 05-AYED**

Jefer Alexis González Romero. AYED-2. 26/10/2021

**1.** ¿Cuál de los siguientes árboles son red-black BSTs?

Imagen que contiene reloj, objeto

Descripción generada automáticamente

Son árboles red-black BSTs el *iii* y *iv*

El *i* y *ii* no son, ya que hay rutas desde la raíz hasta las hojas vacías que no pasan por el mismo número de nodos negros.

**2.** Dibuje el paso a paso de insertar las letras desde la A hasta la K, de manera ascendente y descendente, a un red-black BST que está vacío inicialmente.

**Ascendente**

Insertar A

**A**

Insertar B

**A**

**B**

Insertar C

**B**

**A**

**C**

Insertar D

**B**

**C**

**A**

**D**

Insertar E

**B**

**A**

**D**

**C**

**E**

Insertar F

**D**

**E**

**B**

**F**

**C**

**A**

Insertar G

**D**

**C**

**A**

**G**

**E**

**B**

**F**

Insertar H

**D**

**C**

**A**

**G**

**E**

**HH**

**B**

**F**

Insertar I

**D**

**C**

**A**

**E**

**H**

**B**

**F**

**I**

**Gg**

Insertar J

**D**

**C**

**A**

**.I**

**F**

**B**

**H**

**G**

**E**

**J**

Insertar K

**D**

**C**

**A**

**B**

**H**

**G**

**E**

**K**

**I**

**F**

**J**

**Descendente**

Insertar K

**K**

Insertar J

**K**

**J**

Insertar I

**J**

**I**

**K**

Insertar H

**J**

**K**

**.I**

**H**

Insertar G

**J**

**K**

**H**

**I**

**G**

Insertar F

**H**

**G**

**J**

**F**

**I**

**K**

Insertar E

**H**

**G**

**E**

**K**

**I**

**F**

**J**

Insertar D

**H**

**I**

**K**

**E**

**G**

**D**

**J**

**F**

Insertar C

**H**

**I**

**K**

**G**

**D**

**J**

**F**

**C**

**Eg**

Insertar B

**H**

**I**

**K**

**C**

**F**

**J**

**D**

**E**

**G**

**B**

Insertar A

**H**

**I**

**K**

**J**

**D**

**E**

**G**

**A**

**C**

**F**

**B**

**3.** Dibuje el árbol AVL resultante al aplicar las siguientes operaciones:

Un dibujo en blanco y negro

Descripción generada automáticamente con confianza media

**5**

**2**

**4**

**20**

**15**

**12**

**10**

**{**Dibujo en blanco y negro

Descripción generada automáticamente con confianza media

**5**

**21**

**20**

**18**

**15**

**25**

**10**

**11**

**12**

**2**

**12.2.1** Supongamos que tenemos números entre 1 y 100 en un árbol de búsqueda binario, y queremos hacer una búsqueda para el número 363. ¿Cuáles de las siguientes secuencias no puede ser la secuencia de nodos examinada?

a. 2, 252, 401, 398, 330, 344, 397, 363.

b. 924, 220, 911, 244, 898, 258, 362, 363.

c. 925, 202, 911, 240, 912, 245, 363.

d. 2, 399, 387, 219, 266, 382, 381, 278, 363.

e. 935, 278, 347, 621, 299, 392, 358, 363.

Solo el c no sirve como secuencia de nodos, ya que, a la izquierda del 911 estaría un 912 y en un árbol de búsqueda binaría eso no es posible.

**12.2-2** Escribe versiones recursivas de Tree-Minimum y Tree-Maximum

def Tree\_Minimum(self):  
 if self.left\_tree is not None:  
 return self.left\_tree.Tree\_Minimum()  
 return self

def Tree\_Maximum(self):  
 if self.right\_tree is not None:  
 return self.right\_tree.Tree\_Maximum()  
 return self

**12.2-3** Escribe el Tree-Predecessor

Si el elemento al cual queremos encontrar el predecesor tiene subárbol izquierdo entonces tomamos el nodo con mayor valor de ese árbol. Si no tiene, entonces comenzamos a subir por los padres de los nodos hasta que el subárbol izquierdo del padre sea diferente al elemento, que en este caso si se encontraría un predecesor, pero si llegamos hasta la raíz del árbol, entonces el elemento no tiene predecesor.

def predecesor(self):  
 if self.get\_left()is not None:  
 return self.get\_left().maximum()  
 else:  
 higher = self.higher  
 element = self  
 while higher is not None and element == higher.get\_left():  
 element = higher  
 higher = element.higher  
 if higher is None:  
 return None  
 else:  
 return higher