Hoja de trabajo

Parte 1: Análisis de las búsquedas en un árbol binario de búsqueda

Material de referencia:

- Lectura “El método de búsqueda en un árbol (binario de búsqueda)”.
  Libro: "*Data Structures and Program Design in C++*" de Kruse y Ryba.
  Páginas: 448-450.

1. ¿Cuál es la forma del árbol donde, en promedio, son más rápidas las búsquedas (menor cantidad de comparaciones)? En un árbol con altura logarítmica. Su altura es $\log_2(n)$ donde n es la cantidad de nodos en el árbol. La forma es un árbol bien "bushy", es ancho.

2. ¿Cuál es la forma del árbol donde, en promedio, son más lentas las búsquedas (mayor cantidad de comparaciones)? Un árbol muy alto, dónde la altura es lineal al número de nodos en el árbol.

3. ¿Qué concluye sobre el análisis de cantidad de comparaciones en árboles binarios de “diferente forma”? Si queremos aprovechar la estructura binaria de un árbol binario de búsqueda, debemos considerar usar un árbol balanceado para que las operaciones de búsqueda sean lo más rápidas posible.

4. Escriba un algoritmo para calcular la altura de un árbol binario. Considere que un árbol vacío tiene una altura 0 y un árbol que tiene un solo nodo tiene una altura 1.

   ```
   ALGORITMO altura (se llama recursivamente, comenzando por la raíz)
   INICIO
     altura = 0
     Para cada nodo REPETIR
       SI nodo es hoja ENTONCES
       altura = Max(altura, Profundidad(nodo))
   FIN
   ```

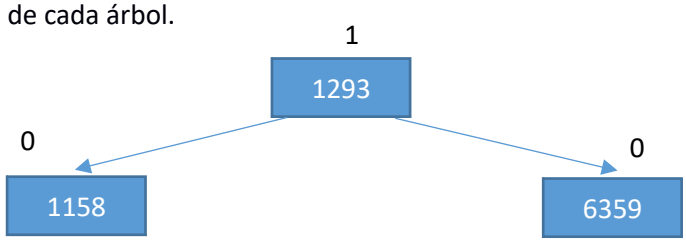
Parte 2: Balance de árboles

Material de referencia:

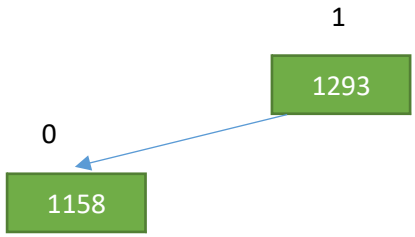
- Lectura “Balance de altura: árboles AVL”.
  Libro: "*Data Structures and Program Design in C++*" de Kruse y Ryba.
  Páginas: 473-474.

5. ¿Cuál es la diferencia máxima de altura entre el subárbol izquierdo y el subárbol derecho, en un árbol AVL?
    - 1

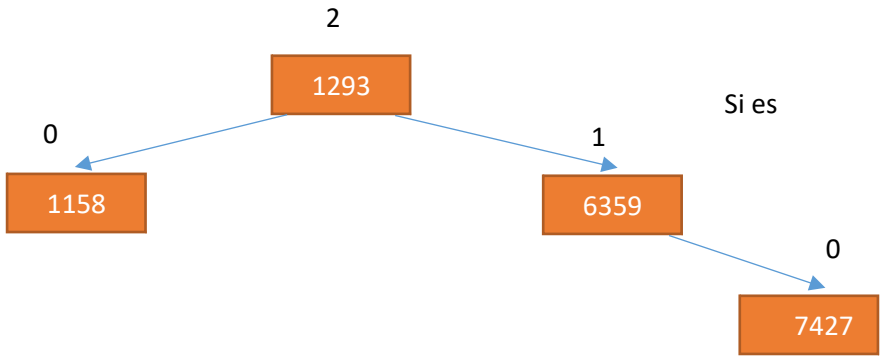
6. ¿Cuáles de los siguientes son árboles AVL? ¿Por qué? Señalar el factor de balance en cada nodo, de cada árbol.



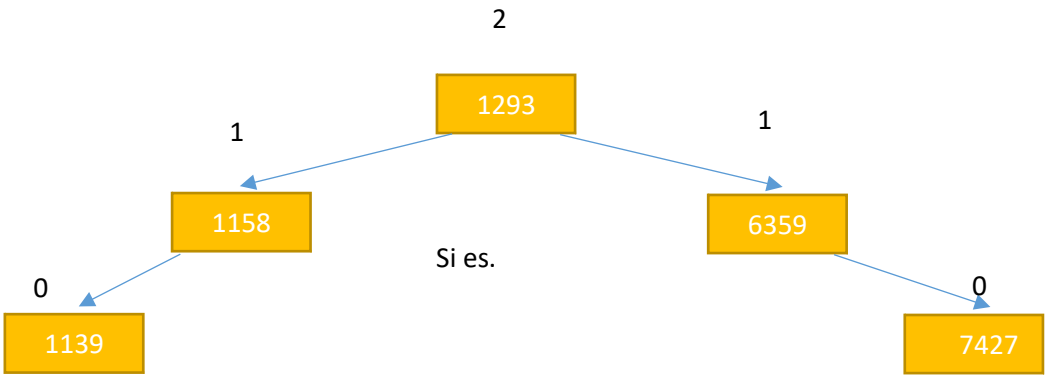
Si es



Si es

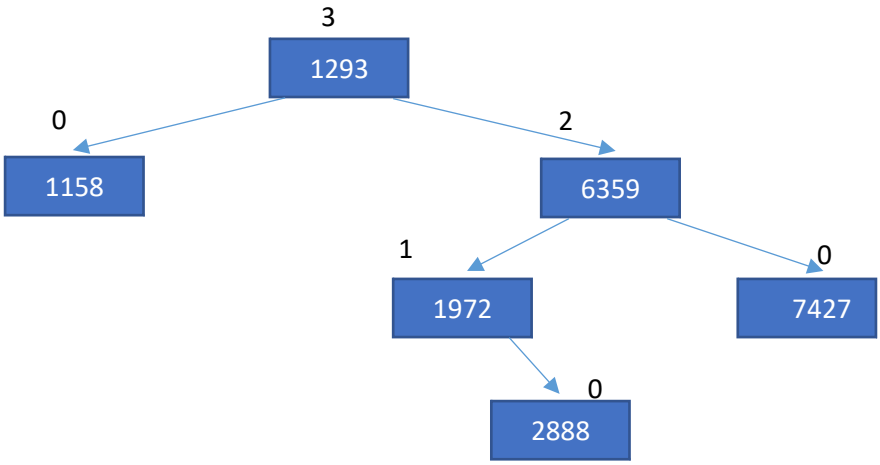


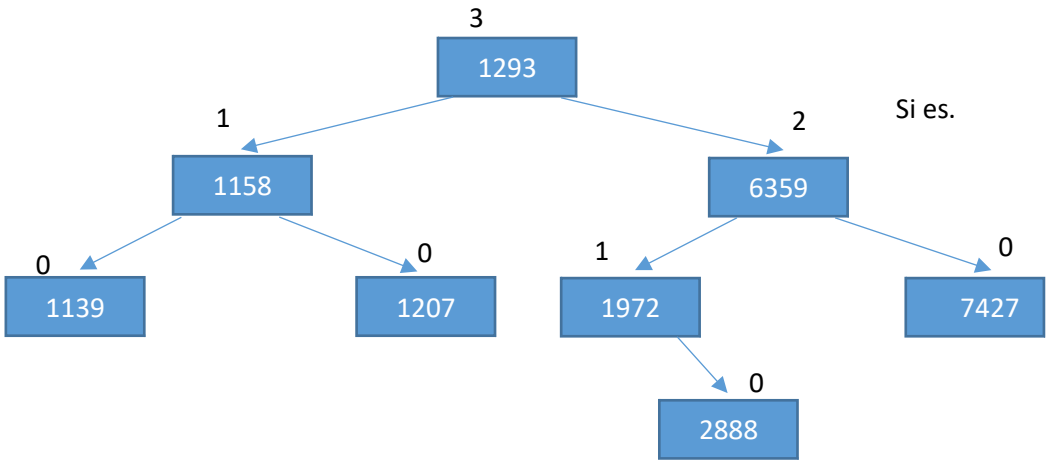
Si es



Si es.

No es.





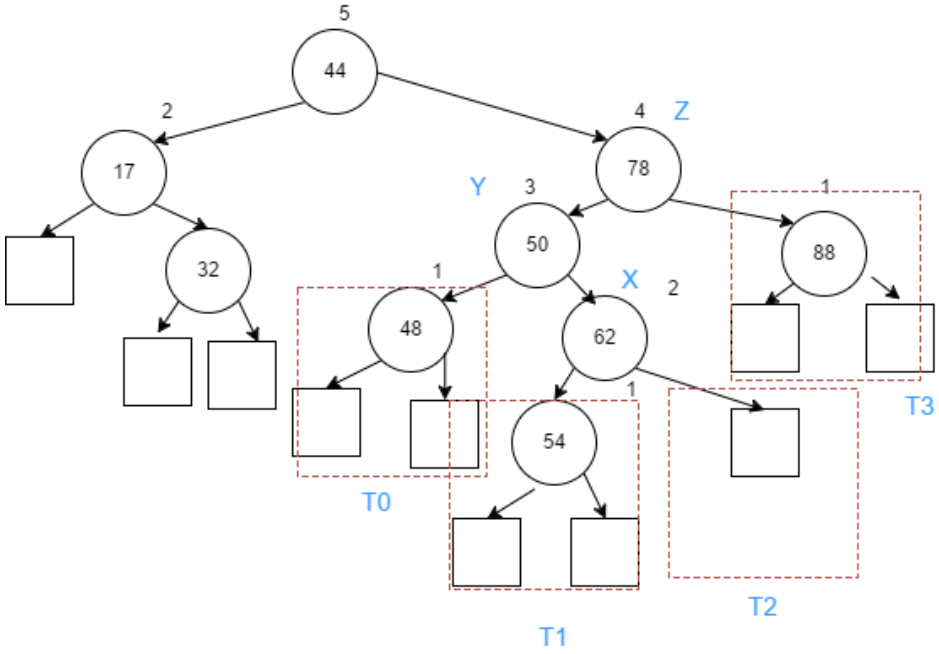
Si es.

7. Investigue el algoritmo para insertar en un árbol AVL. Explique en sus palabras este algoritmo. Puede utilizar diagramas o esquemas si es necesario. Coloque las referencias bibliográficas revisadas durante la investigación (libros, direcciones de páginas Web, videos, etc.).
    - Libro consultado: Data Structures and Algorithms in Java.
- Es importante notar que este algoritmo funciona para una implementación de árbol binario de búsqueda propio.

    - Partimos de la idea que al insertar, es posible que desbalanciemos el árbol, así que debemos revisar el árbol para ver si esta balanceado.
    - Partimos desde el nodo que acabamos de insertar. Nos movemos hacia arriba por medio de la referencia al padre del nodo. Nos seguimos moviendo hasta que encontremos un nodo desbalanceado o lleguemos a la raíz y la raíz esté balanceada.

- Si encontramos un nodo desbalanceado se procede de la siguiente manera:
  - Z será el nodo que encontramos desbalanceado. Tomamos a Y como el hijo más alto de Z. Tomamos a X como el hijo más alto de Y. Tomamos los subarboles T0, T1, T2 y T3 como los subarboles en orden de X, Y, Z, que no tengan como raíz a X, Y, Z.
  - Temporalmente nombramos a los nodos X,Y,Z como A,B,C tal que A preceda a B y B preceda a C en un recorrido en orden.
  - Reemplazamos a Z con B.
  - Hacemos que A sea hijo izquierdo de B.
  - Hacemos que C sea hijo derecho de B.
  - Hacemos que T0 y T1 sean hijos de A.
  - Hacemos que T2 y T3 sean hijos de C.
  - El árbol está balanceado ahora.

El árbol se desbalancea luego de insertar el nodo con el número 54. Notamos que los nodos 44 y 78 están desbalanceados.



El árbol se balancea.