Práctica 2

Christian Néstor Barriga Marcapura
Weimar Ccapatinta Huamani

Agosto 27, 2022

Contents

1 Introducción

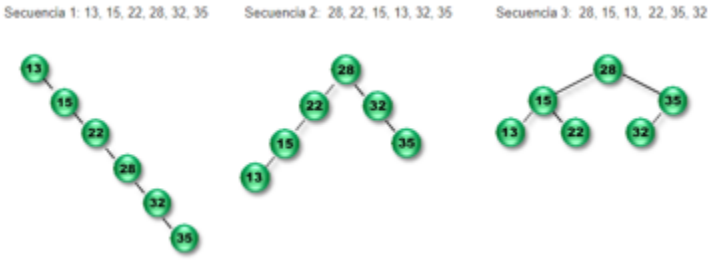
En ciencias de la computación, un árbol binario es una estructura de datos en la cual cada nodo puede tener un hijo izquierdo y un hijo derecho. No pueden tener más de dos hijos (de ahí el nombre "binario"). Si algún hijo tiene como referencia a null, es decir que no almacena ningún dato, entonces este es llamado un nodo externo. En el caso contrario el hijo es llamado un nodo interno. Usos comunes de los árboles binarios son los árboles binarios de búsqueda, los montículos binarios y Codificación de Huffman.

2 Estructucturas de datos de árbol

Se han evaluado los siguientes algoritmos:

2.1 AVL Tree

Fue la primera estructura de datos de este tipo que se inventó. En un árbol AVL, las alturas de los dos subárboles secundarios de cualquier nodo difieren como máximo en uno; si en algún momento difieren en más de uno, se realiza un reequilibrio para restaurar esta propiedad. La búsqueda, la inserción y la eliminación toman tiempo O (log n) tanto en el promedio como en el peor de los casos, donde estilo de visualización nnortees el número de nodos en el árbol antes de la operación. Las inserciones y eliminaciones pueden requerir que el árbol sea reequilibrado por una o más rotaciones de árbol .



El mismo conjunto de datos, por ejemplo: 13, 15, 28, 32, 35, 22, puede generar con distintas alturas.

```
#####    RESULTADOS   ##############
#         10
#        /    \
#       5     15
#     /  \       \
#    1     9        30


# deleting --> 5

#         10
#        /    \
#       9     15
#     /          \
#    1            30


# Maximum element is 30
# Minimum element is 1
# Search element is 9
```

Figura 1 Resultados algoritmo AVL Tree - Elaboración propia

2.2 B tree

son árboles balanceados de búsqueda, pero cada nodo puede poseer más de dos hijos. Los árboles B mantienen los datos ordenados y las inserciones y eliminaciones se realizan en tiempo logarítmico amortizado.

Dado que se permite un rango variable de nodos hijo, los árboles-B no necesitan rebalancearse tan frecuentemente como los árboles binarios de búsqueda auto-balanceables. Pero, por otro lado, pueden desperdiciar memoria, porque los nodos no permanecen totalmente ocupados. Los límites (uno superior y otro inferior) en el número de nodos hijo son definidos para cada implementación en particular.

```
#####    RESULTADOS   ##############
#
# Level  0   4:(2, 4) (5, 10) (8, 16) (11, 22)
# Level  1   2:(0, 0) (1, 2)
# Level  1   2:(3, 6) (4, 8)
# Level  1   2:(6, 12) (7, 14)
# Level  1   2:(9, 18) (10, 20)
# Level  1   3:(12, 24) (13, 26) (14, 28)

#  Deleting --> 10

# Level  0   3:(2, 4) (8, 16) (11, 22)
# Level  1   2:(0, 0) (1, 2)
# Level  1   4:(3, 6) (4, 8) (6, 12) (7, 14)
# Level  1   2:(9, 18) (10, 20)
# Level  1   3:(12, 24) (13, 26) (14, 28)

#  Search element is 6
```

Figura 2 Resultados algoritmo B Tree - Elaboración propia

3 Implementación

Se envía el siguiente enlace en Github, se ha colocado dos carpetas donde una se encuentra los algoritmos de balance realizados en python y en otra la animación realizada en p5.js.

https://github.com/weicap/MCC_Practica_2

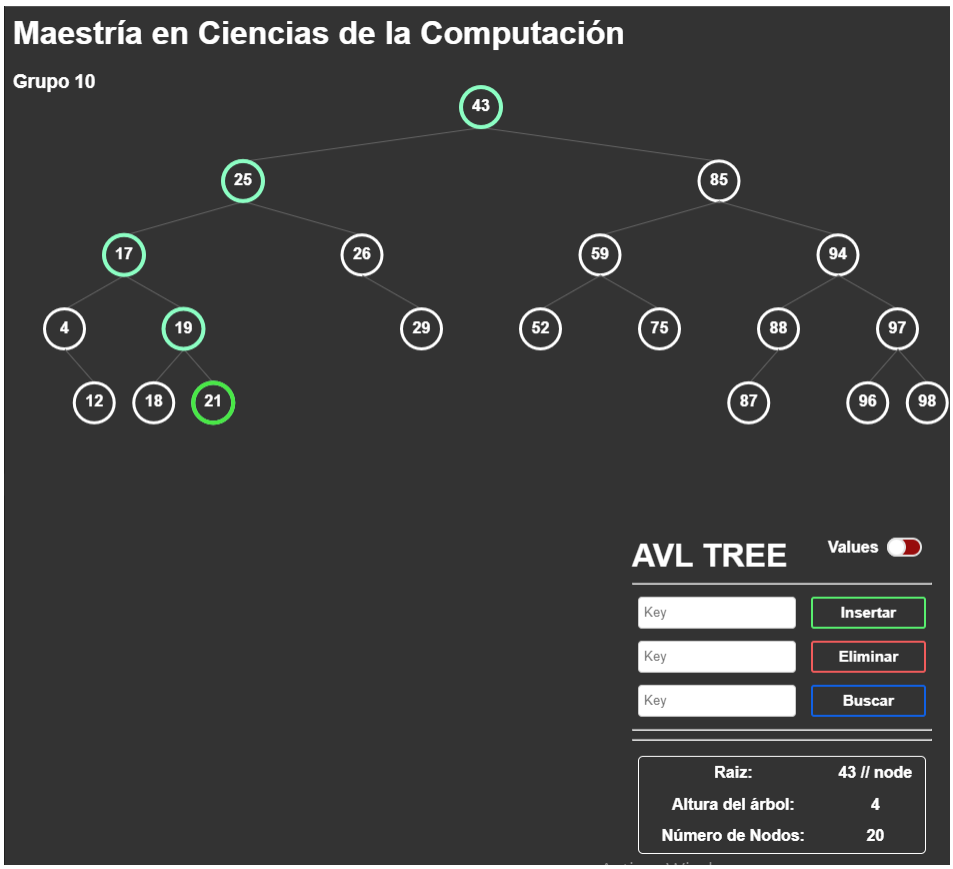


Figura 3 Animacion algoritmo AVL Tree - Elaboración propia

4 Conclusiones

- Los árboles AVL son árboles binarios equilibrados que se utilizan principalmente en la indexación de bases de datos.
- Todas las operaciones realizadas en los árboles AVL son similares a las de los árboles de búsqueda binarios, pero la única diferencia en el caso de los árboles AVL es que necesitamos mantener el factor de equilibrio, es decir, la estructura de datos debe permanecer como un árbol equilibrado como resultado de varias operaciones. Esto se logra mediante el uso de la operación Rotación de árbol AVL.
- Los árboles B tienen ventajas sustanciales sobre otras implementaciones cuando el tiempo de acceso a los nodos excede al tiempo de acceso entre nodos.