SOFT-10 Estructuras de datos

Segundo estudio de caso

Factor de balance y corrección de desbalances de un árbol AVL

Sección

SCV2

Estudiante

Andrés Vargas Guevara

Profesor

Romario Salas Cerdas

Fecha

30 de Noviembre, 2025

Tabla de contenidos

Introducción

Este documento presenta el segundo caso de estudio, el cual consiste en el desarrollo de una investigación en torno a la implementación de los algoritmos de evaluación del factor de balance en los árboles autobalanceados AVL, así como de los algoritmos de rotación que permiten la corrección de los desbalances que aparecen en dichas estructuras.

Desarrollo

Un árbol AVL es un árbol binario de búsqueda autobalanceado, donde el factor de equilibrio (también conocido como factor de balance) siempre está entre -1 y +1.

Factor de equilibrio

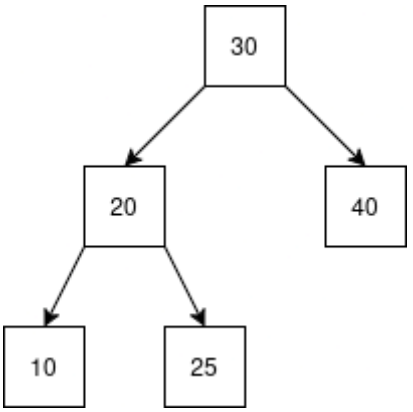
El factor de equilibrio (BF) de un nodo se define como la diferencia de alturas entre el subárbol izquierdo y el derecho.

Un algoritmo que permite calcular los factores de equilibrio recibe de entrada la raíz del árbol, evalúa si el árbol está vacío, y dependiendo de la respuesta, realiza una diferencia de alturas recursiva de los subárboles izquierdo y derecho.

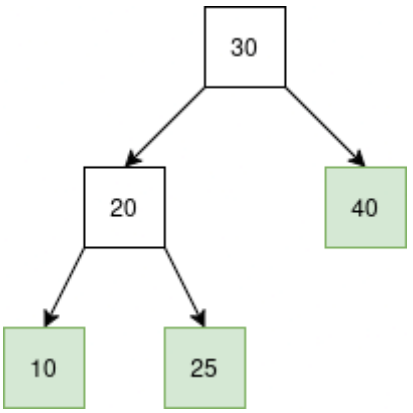
Algoritmo en pseudocódigo

```
Algoritmo EvaluarBF(T):

  Entrada: raíz del árbol T

  Salida: altura y BF de cada nodo


  función calcular(v):

    si v es nulo:

      retornar -1


    hl = calcular(v.izq)

    hr = calcular(v.der)


    altura(v) = 1 + max(hl, hr)

    BF(v) = hl - hr


    retornar altura(v)


  calcular(T.raíz)
```

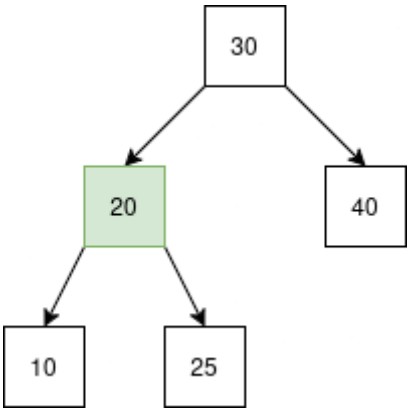
Representación gráfica



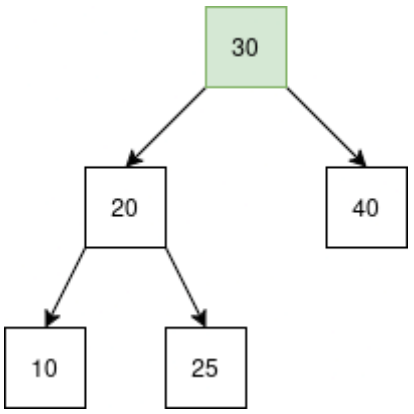
El árbol no es nulo, por lo que se procede a calcular, recursivamente, los factores de equilibrio de los distintos nodos.



Los nodos hoja “10”, “25” y “40” tienen altura y factor de equilibrio de 0.



El nodo “20” tiene una altura de 1 y un factor de equilibrio de 0 - 0 = 0.



El nodo “30” tiene una altura de 2 y un factor de equilibrio de 1 - 0 = 1.

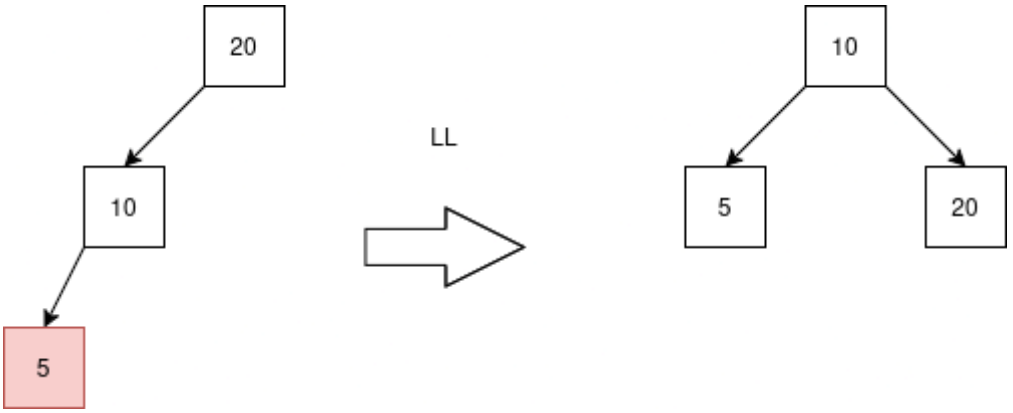
Rebalanceo en inserciones y eliminaciones

Existen cuatro casos de rotación:

1. LL (rotación derecha): Inserción en subárbol izquierdo del hijo izquierdo.
2. RR (rotación izquierda): Inserción en subárbol derecho del hijo derecho.
3. LR (rotación doble izquierda-derecha): Inserción en subárbol derecho del hijo izquierdo.
4. RL (rotación doble derecha-izquierda): Inserción en subárbol izquierdo del hijo derecho.

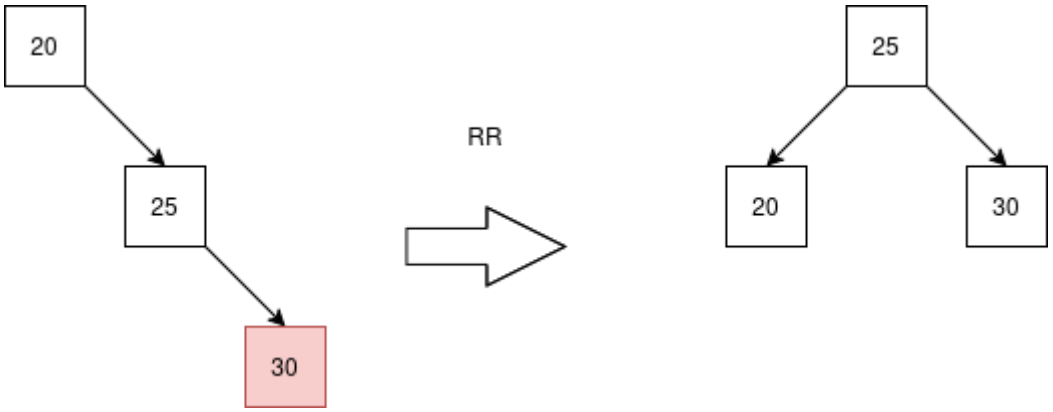
Representación gráfica - LL

La inserción del elemento “5” causa un desbalance que se resuelve mediante una rotación derecha.



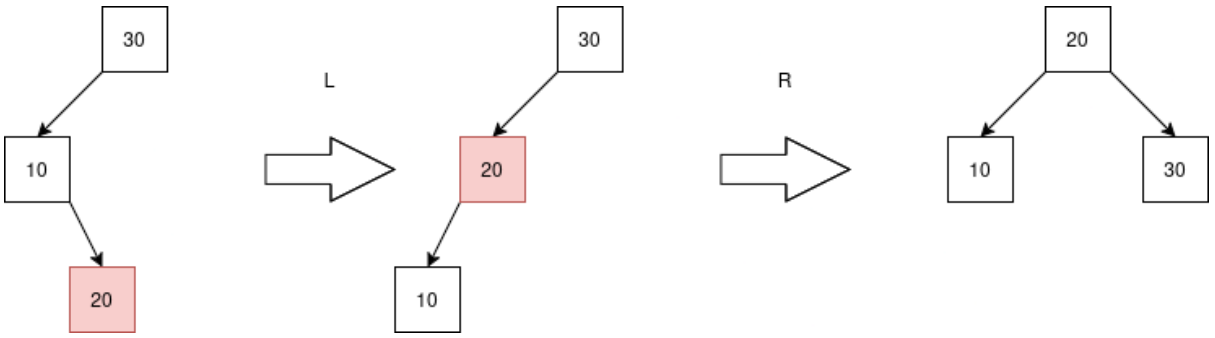
Representación gráfica - RR

La inserción del elemento “30” causa un desbalance que se resuelve mediante una rotación izquierda.



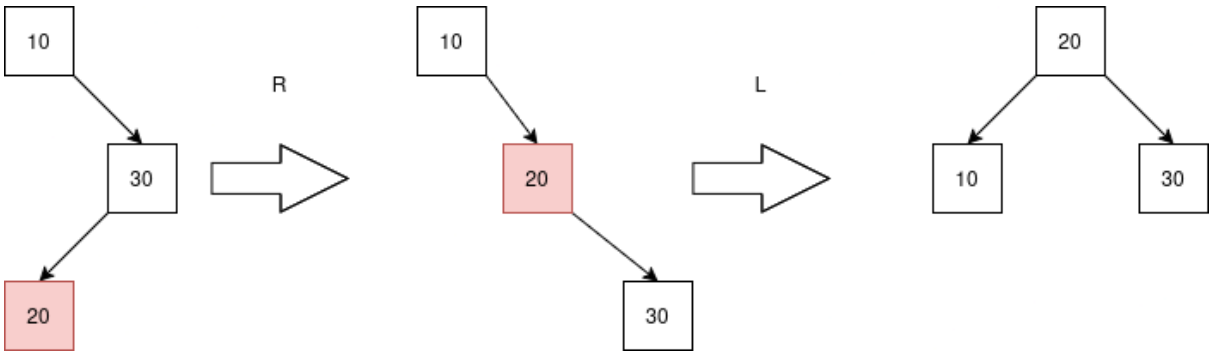
Representación gráfica - LR

La inserción del elemento “20” causa un desbalance que se resuelve mediante una rotación izquierda seguida de una rotación derecha.



Representación gráfica - RL

La inserción del elemento “20” causa un desbalance que se resuelve mediante una rotación derecha seguida de una rotación izquierda.



Aplicaciones principales de los árboles AVL

- Bases de datos e indexación
    - Particularmente en la implementación de índices de bases de datos para búsquedas rápidas. Ya que se garantizan tiempos de búsqueda logarítmicos incluso en el peor caso. El balance estricto asegura un rendimiento predecible, lo que es crucial para consultas y transacciones.
- Gestión de memoria
    - Específicamente en compiladores y sistemas operativos. Ya que permite administrar eficientemente los bloques de memoria libres, incluso luego de inserciones y eliminaciones frecuentes. Además de que se mantienen las referencias a memoria organizadas, lo que reduce la fragmentación.
- Tablas de enrutamiento
    - Proporciona velocidad de búsqueda consistente. Es escalable, ya que las búsquedas mantienen su complejidad incluso cuando aumenta el volumen de datos.

Implementación

Ubicación del repositorio https://github.com/avargasgu-ucenfotec/avl

Referencias bibliográficas

AVL Tree program in Java. (2025, Abril 4). GeeksforGeeks. *https://www.geeksforgeeks.org/java/avl-tree-program-in-java/*

DSA AVL Trees. W3Schools. *https://www.w3schools.com/dsa/dsa_data_avltrees.php*