OBSERVACIONES DEL LA PRACTICA

1. Maria Paula Nizo Vega, m.nizo@uniandes.edu.co, 202213902

2. Paul Paffen Suarez, p.paffen@uniandes.edu.co, 202222496

3. Andres Camilo Caballero Ayala, Ac.caballero@uniandes.edu.co, 202216295

Preguntas de análisis

1) ¿Qué diferencia existe entre las alturas de los dos árboles (BST y RBT)?

En un árbol BST, la altura del árbol puede variar significativamente dependiendo del orden en que se insertan los nodos. Si los nodos se insertan en un orden que produce un árbol desequilibrado (por ejemplo, si se insertan en orden ascendente o descendente), la altura del árbol puede ser lineal con respecto al número de nodos, lo que hace que las operaciones de búsqueda, inserción y eliminación sean menos eficientes. Por otro lado, si los nodos se insertan en un orden aleatorio, es más probable que el árbol esté más equilibrado y tenga una altura más baja.

En un árbol RBT, sin embargo, se toman medidas para garantizar que el árbol siempre esté relativamente equilibrado, independientemente del orden en que se inserten los nodos. Para hacer esto, se utilizan reglas específicas de coloración de nodos y rotaciones de árboles para garantizar que la altura negra de cualquier camino desde la raíz a cualquier hoja sea aproximadamente la misma. Esto significa que, en promedio, la altura de un árbol RBT es menor que la de un árbol BST no equilibrado.

2) ¿Percibe alguna diferencia entre la ejecución de los dos árboles (RBT y BST)? ¿Por qué pasa esto?

Sí, hay una diferencia en el tiempo de ejecución entre los árboles RBT y BST, esto puede depender del tipo de operaciones que se realicen en los árboles y del número de nodos que contengan.

En un árbol BST, si los nodos están ordenados en una lista lineal, es decir, si el árbol está desequilibrado, la búsqueda, inserción y eliminación de nodos pueden requerir un tiempo proporcional al número de nodos en el árbol, lo que significa que su tiempo de ejecución podría ser O(n). En un árbol equilibrado, en cambio, el tiempo de búsqueda, inserción y eliminación de nodos puede ser O(log n), donde n es el número de nodos en el árbol.

En un árbol RBT, por otro lado, debido a su estructura equilibrada, las operaciones de búsqueda, inserción y eliminación de nodos tienen un tiempo de ejecución garantizado O(log n) en el peor de los casos. Además, el hecho de que los árboles RBT estén equilibrados también significa que su tiempo de ejecución promedio para estas operaciones es menor que el de un árbol BST no equilibrado.

3) ¿Existe alguna diferencia de complejidad entre los dos árboles (RBT y BST)? Justifique su respuesta.

Como ya se explicó anteriormente con detalle, aunque ambos árboles pueden almacenar y operar sobre los mismos datos, el árbol RBT tiene una complejidad garantizada O(log n) para la búsqueda, inserción y eliminación de nodos en el peor de los casos, mientras que la complejidad de un árbol BST depende de la estructura del árbol y puede ser O(n) en el peor caso.

4) ¿Existe alguna manera de cargar los datos en un árbol RBT de tal forma que su funcionamiento mejore? Si es así, mencione cuál.

Sí, la forma en que se cargan los datos en un árbol RBT puede afectar su funcionamiento y hay ciertas técnicas que se pueden utilizar para mejorar su rendimiento.

Una técnica común para cargar datos en un árbol RBT de manera eficiente es utilizar una técnica llamada "inserción masiva" o "bulk-loading" en inglés. En lugar de insertar cada elemento de datos individualmente en el árbol, lo que podría generar un árbol desequilibrado, la técnica de inserción masiva implica la creación del árbol completo de una sola vez.

Existen diferentes algoritmos para la inserción masiva en un árbol RBT, algunos de los cuales se basan en la creación de un árbol equilibrado en lugar de insertar nodos uno por uno. Un ejemplo de tal algoritmo es el algoritmo de inserción de ordenación previa o "pre-order insertion", que se basa en ordenar previamente los datos antes de crear el árbol RBT. Este algoritmo puede garantizar que el árbol RBT resultante tenga una altura balanceada y, por lo tanto, un mejor rendimiento.

Otro enfoque para la inserción masiva es utilizar el algoritmo "bulk insert", que funciona insertando elementos en el árbol en grupos en lugar de uno por uno. Este enfoque puede reducir la cantidad de rotaciones necesarias para equilibrar el árbol y, por lo tanto, mejorar el rendimiento.

5) https://github.com/EDA-202310-SEC11-G03/LabBalanceTree-L-09--G-03-.git

Diagrama de la estructura