OBSERVACIONES DE LA PRÁCTICA

Fabio Salazar - 202511238
Samuel Tovar - 202211625
Juan Esteban Espitia - 202516958

Preguntas de análisis

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

La altura de un RBT es menor o igual que la de un BST para la misma cantidad de datos.

Esto se debe a que el RBT mantiene balanceadas sus ramas automáticamente mediante rotaciones y cambios de color cada vez que se inserta un nodo, garantizando que ningún camino de la raíz a una hoja sea más del doble de largo que otro.

En cambio, el BST no tiene ningún mecanismo de balanceo: si los datos se insertan en orden ascendente o descendente, el árbol se "deforma" y se comporta como una lista enlazada (la altura sería "similar" al número de nodos). Lo que haría que, en el peor caso (osea, que todos los elementos sean mayores o menores al root), el árbol se convierta en una sll.

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

Sí, el RBT ejecuta las operaciones más rápido que el BST.

En el RBT las operaciones de búsqueda, inserción y eliminación se realizan sobre un árbol casi balanceado, por lo que se recorren menos niveles.

En el BST, si los datos no están distribuidos aleatoriamente, el árbol puede desbalancearse, haciendo que las operaciones tarden más (en el peor caso, O(n)).

En otras palabras, la ejecución del RBT es más eficiente debido al balanceo automático de este, mientras que la del BST depende del orden de inserción de los datos.

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

Sí, existe una diferencia importante en la complejidad entre un árbol binario de búsqueda (BST) y un árbol rojo-negro (RBT). Aunque en promedio ambos presentan un tiempo de ejecución de O(log n) para las operaciones de búsqueda, inserción y eliminación, su comportamiento en el peor caso es distinto.

En el BST, si los datos se insertan en un orden "desbalanceado" (por ejemplo, ascendente o descendente), el árbol puede desbalancearse completamente y adoptar una forma lineal, igual a una sll. En ese caso, la altura del árbol equivaldría al número total de nodos, y las operaciones básicas pasan a tener una complejidad de O(n).

Por el contrario, el RBT mantiene un balance automático mediante rotaciones en cada inserción o eliminación (además de las reglas de los colores de los lazos). Esto garantiza que la altura del árbol siempre permanezca acotada por un valor proporcional a log n, por lo que las operaciones de búsqueda, inserción y eliminación conservan una complejidad O(log n) en el peor de los casos.

En resumen, el RBT es más eficiente y estable que el BST porque asegura un rendimiento logarítmico constante, independientemente del orden en que se carguen los datos. Esto dado a su mecanismo de auto-balanceo.

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. Aunque el RBT ya se balancea solo, su rendimiento puede mejorar si se reduce la cantidad de rotaciones y cambios de color durante la carga de datos.
Algunas formas de lograrlo son:

-Cargar los datos previamente ordenados por la llave

-Insertar los datos en un orden aleatorio, evitando secuencias completamente ascendentes o descendentes que provoquen más rotaciones/cambios de color.

Aunque el RBT ya mantiene un buen rendimiento sin importar el orden, estas estrategias podrían optimizar el tiempo de carga aún más.