**OBSERVACIONES DEL LA PRACTICA**

Estudiante 1 Cod XXXX

Estudiante 2 Cod XXXX

Estudiante 3 Cod XXXX

# **Preguntas de análisis**

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

* **BST (árbol de búsqueda binaria):**
* Propiedades: en un BST, la altura puede variar dependiendo del orden en el que se inserten los elementos. Si los elementos se insertan en un orden que resulta en un árbol desequilibrado, la altura puede ser grande, como en este caso (altura de 29). Los peores casos de inserción pueden llevar a un árbol que se comporta mas como una lista enlazada.
* Equilibrio: Los BST no garantizan un equilibrio automático después de cada inserción o eliminación, lo que puede llevar a arboles desequilibrados y alturas grandes en ciertos casos
* **RBT (Árbol Rojo- Negro):**
* Altura:13
* Propiedades: Los RBT son arboles binarios de búsqueda con reglas adicionales de color que garantizan un equilibrio relativo y una altura limitada. Las reglas de color mantienen el árbol relativamente equilibrado incluso después de inserciones y eliminaciones, lo que limita la altura máxima de árbol
* Equilibrio: Los RBT están diseñados para mantener un equilibrio relativo que limita la altura del árbol a O (log n), donde n es el numero de elementos en el árbol. Esto significa que incluso en casos de inserciones o eliminaciones, el árbol tiende a mantenerse equilibrado y su altura se mantiene relativamente baja en comparación con BST desequilibrado.

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

Si hay diferencias perceptibles en la ejecución de los arboles RBT y BST debido a sus propiedades y estructuras:

1. Tiempo de inserción y eliminación:

* RBT: en general, los RBT tienden a tener tiempos de inserción y eliminación más rápidos en comparación con los BST desequilibrados. Esto se debe a que los RBT mantienen un equilibrio relativo a través de sus reglas de color, lo que limita la altura del árbol y mejora el rendimiento de las operaciones.
* BST: En un BST desequilibrado, especialmente en casos extremos como un árbol que se comporte como una lista, las operaciones de inserción y eliminación pueden ser lentas debido a la alta altura del árbol y la estructura desequilibrada que puede requerir recorridos largos para encontrar elementos

1. Búsqueda y recorrido:

* RBT: La búsqueda en un RBT suele ser eficiente, ya que mantiene una estructura relativamente equilibrada que limita la altura del árbol y por lo tanto, reduce la cantidad de comparaciones necesarias para encontrar un elemento.
* BST: En un BST desequilibrado, la búsqueda puede volverse ineficiente, en especial en el peor caso de un árbol desequilibrado.

1. Espacio en memoria

* RBT: los RBT pueden requerir un poco mas de espacio en memoria debido a la necesidad de almacenar información adicional de color para cada nodo
* BST: un BST desequilibrado puede requerir menos espacio que un RBT debido a su estructura más simple.

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

Si, existe una diferencia de complejidad entre los árboles RBT y BST:

Complejidad de inserción y eliminación:

* RBT: es O (log n) , donde n es el numero de elementos en el árbol. Esto se debe a que los RBT mantienen un equilibrio relativo a través de sus reglas, lo que limita la altura del árbol y garantiza un rendimiento eficiente para estas operaciones.
* BST: en un BST desequilibrado, la complejidad de búsqueda puede ser de O(n) en el peor caso, esto ocurre cuando el árbol está desequilibrado y requiere un recorrido lineal.

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

* Si, una combinación de las siguientes estrategias suele ser efectiva.
* Inserción balanceada: es fundamental insertar los elementos de forma balanceada en el RBT para evitar desequilibrios significativos
* Subconjuntos de datos: si se trabaja con grandes conjuntos de datos, dividirlos en subconjuntos mas pequeños y cargarlos así, puede ayudar a mantener un equilibrio mas uniforme en el árbol y evitar desequilibrios
* Rebalanceo: implementa mecanismos de rebalanceo para ajustar la estructura del árbol durante las operaciones de inserción y eliminación, manteniendo el equilibrio del árbol en todo momento.