OBSERVACIONES DEL LA PRACTICA

Ana Sofía Castellanos 202114167
Martín Santiago Galván Castro 201911013

Preguntas de análisis

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

El árbol BST cuenta con una altura de 29 la cual es significativamente mayor a la altura del RBT que es de 13. Para el caso del RBT, la altura es muy cercana a la predicha por la fórmula de un árbol binario balanceado:

$$floor(log_2(N)) = h$$

$$floor(log_2(1117)) = 10$$

Aunque si bien en un árbol perfectamente balanceado la altura sería 10, esto no se cumple al RBT ya que este no garantiza un balanceo perfecto, pero si uno parcial el cual es mucho mejor que el árbol que resulta de un BST.

b) ¿Por qué pasa esto?

Esta diferencia en las alturas del BST y el RBT ocurre debido a que el BST si bien configura los datos en un árbol binario, no garantiza que dicho árbol esté balanceado, caso contrario con el RBT el cual a partir de diversas operaciones permite generar un árbol binario parcialmente balanceado, que contará con una altura cercana al predicho por la fórmula de un árbol perfectamente balanceado.

Lo anterior sucede debido a las propiedades del árbol rojo-negro. En donde los nodos pueden ser ya sea rojos o negros. De ahí el nombre, adicionalmente:

- Solo los hijos izquierdos pueden ser rojos.
- Ningún nodo puede tener dos hijos rojos al mismo tiempo
- Un nodo no puede tener un hijo izquierdo y un nieto rojo izquierdo al mismo tiempo
- Cuando se inserta un nuevo nodo, este es un hijo rojo.

Para lograr las propiedades anteriores, al insertar hijos se revisa que se cumplan las propiedades. De no cumplirlas, se llevan a cabo instrucciones con el propósito de cumplirlas. Estas son las siguientes:

- Si un hijo rojo queda a la derecha, se realiza una rotación izquierda sobre el padre
- Si un hijo rojo quedo en la izquierda, y su padre también es un hijo rojo, se hace una rotación derecha
- Si los dos hijos de un nodo son rojos, se cambia el color de los hijos a negro y el padre cambia de color.
- Se valida que las anteriores condiciones se cumplan, de no serlo, se repiten las instrucciones anteriores hasta que se cumplan las propiedades.

En términos de código, en la librería de funciones RBT, que se encuentra en el directorio DataStructures, la función put() mete las nuevas parejas llave-valor dentro de un RBT. La

función anterior hace uso de la función recursiva isnertNode(…) que tiene como parámetros la raíz de un árbol, la llave de la pareja, el valor de la pareja, y una función de comparación. La función insertNode(…) primero revisa si la raíz insertada es vacía, es decir, que el árbol o el nodo raíz estén vacíos. De serlo, crea un nuevo nodo y lo devuelve. Después, compara la llave de la pareja con la llave de la raíz con el propósito de verificar si la llave es menor o menor, o igual. De ser ya sea menor o mayor, aplica la función recursiva sobre ya sea el hijo derecho o izquierdo de la raíz. Si la llave es igual a la llave de la raíz, remplaza el valor de la pareja.

Después, se ajusta el balanceo del árbol, esto se hace verificando si las propiedades del árbol una por una y aplicando las operaciones necesarias. Como la función es recursiva, dicho balanceo también se realiza de manera recursiva. Por lo que se asegura que el árbol estará balanceado.

Por último, la función devuelve la raíz del árbol después de añadir el nuevo nodo y de haber realizado el balanceo