LABORATORIO 09 EDA SEC-09

G-06

Julián Restrepo

j.restrepo112@uniandes.edu.co
202320177

Alejandro Salcedo

a.salcedo11@uniandes.edu.co
202321921

Samuel Osorio

sd.osorio2@uniandes.edu.co
202324806

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

Al abordar los archivos de “Boston Crimes” y usando un árbol de tipo “RBT” vemos que la altura para el árbol de la llave Analyzer[“dateIndex”] es de 13 mientras que para la llave de Analyzer[“areaIndex”] es de 12, hay mayor densidad de datos o llaves de mapas en el árbol de fechas que en la de áreas.

Para el tipo de árbol “BST” la altura en el Analyzer[“dateIndex”] es de 29 mientras que para la llave Analyzer[“areaIndex”] la altura es de 19, por lo que podemos concluir que los tipos de árbol “BST” al enfocarse en el orden de sus llaves olvidan la parte del balance del árbol a diferencia de los árboles “RBT”

```
Cargando información de crimenes ....
Crimenes cargados: 319073
Altura del arbol: 29
Elementos en el arbol: 1177
Menor Llave: 2015-06-15
Mayor Llave: 2018-09-03
Altura indice por areas: 19
Elementos indice por areas: 879
```

BST

```
Cargando información de crimenes ....
Crimenes cargados: 319073
Altura del arbol: 13
Elementos en el arbol: 1177
Menor Llave: 2015-06-15
Mayor Llave: 2018-09-03
Altura indice por areas: 12
Elementos indice por areas: 879
```

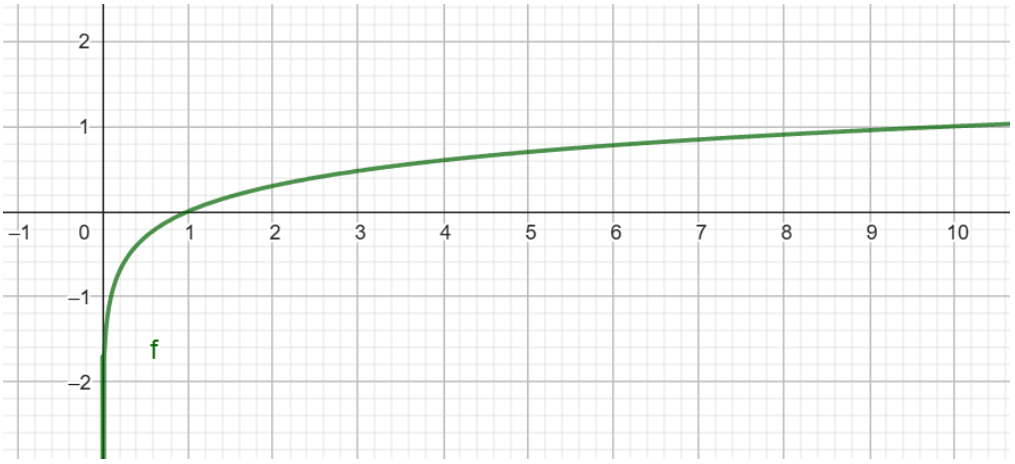
RBT

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

En las pruebas realizadas se puede notar como por lo general la creación de los árboles “BST” son más eficientes en términos de tiempo para su creación pues estos solo se encargan de mandar las llaves a la izquierda o derecha, mientras que los árboles “RBT” deben establecer el balanceo del árbol lo cual la hace una función costosa y por tanto el tiempo de ejecución se extiende. al crear el árbol se deben hacer las tareas correspondientes:

type= “BST”

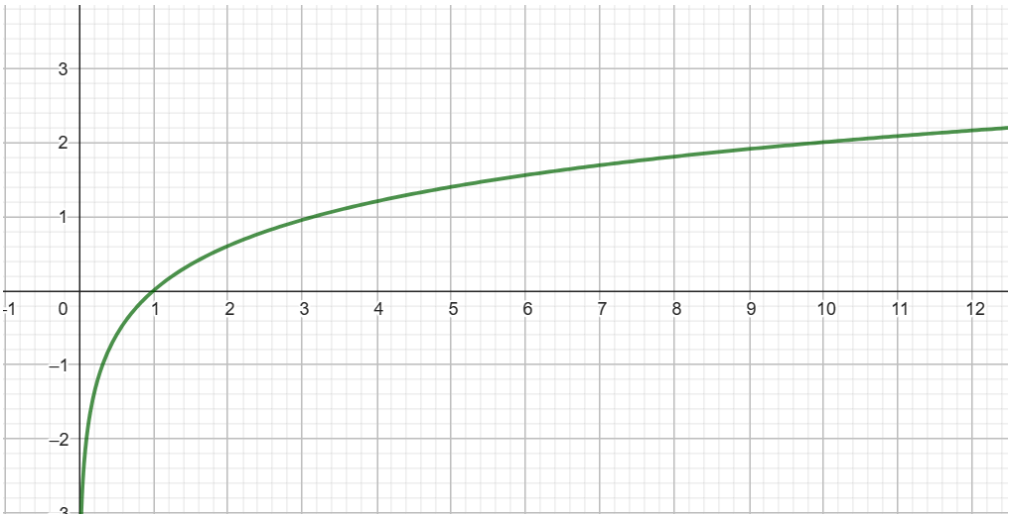
Tarea → ordenar las llaves → O(logn)



type= “RBT”

Tarea → ordenar las llaves → O(logn)

Tarea → balancear el árbol → O(logn)



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

En términos de la creación del árbol vemos que la diferencia radica en que los de tipo RBT tienden a tardar un poco más debido al trabajo de balancear, para cuestiones de obtener un dato, inserción o eliminación de un dato en el árbol por lo general ambos tipos presentan una complejidad de O(logn) más sin embargo al eliminar datos recordemos que el árbol RBT nuevamente debe verificar si está balanceado lo que lo hace un poco más complejo en ese aspecto.

TYPE	Inserción	Eliminación	Búsqueda
RBT	O(logn) tiene ventaja si la altura es menor	O(logn) suele tardar un poco más debido al nuevo balanceo	O(logn) suele ser más eficiente debido a su menor altura
BST	O(logn) puede tener casos hasta de O(n)	O(logn) suele ser más eficiente ya que solo vuelve a juntar nodos	O(logn) suele tardar un poco más debido a su mayor altura

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

Una manera de lograr una mayor rapidez en la creación de un árbol RBT puede ser ingresando las llaves de modo que estén organizadas por mitades, es decir, la raíz debe ser la mitad de todos los datos, los hijos de la raíz deben ser la mitad de la mitad en cada lado y así consecutivamente. el orden en el que llegan las llaves se puede expresar como:

Nodos en nivel l = n/2**l

así en el primer nivel (raíz) nos dará el valor del medio y en el segundo nos dará la cuarta parte pero se debe evaluar antes de la mitad y después de la mitad de datos, en resumen es como si se tratase de un tipo de sorting (Quick sort).