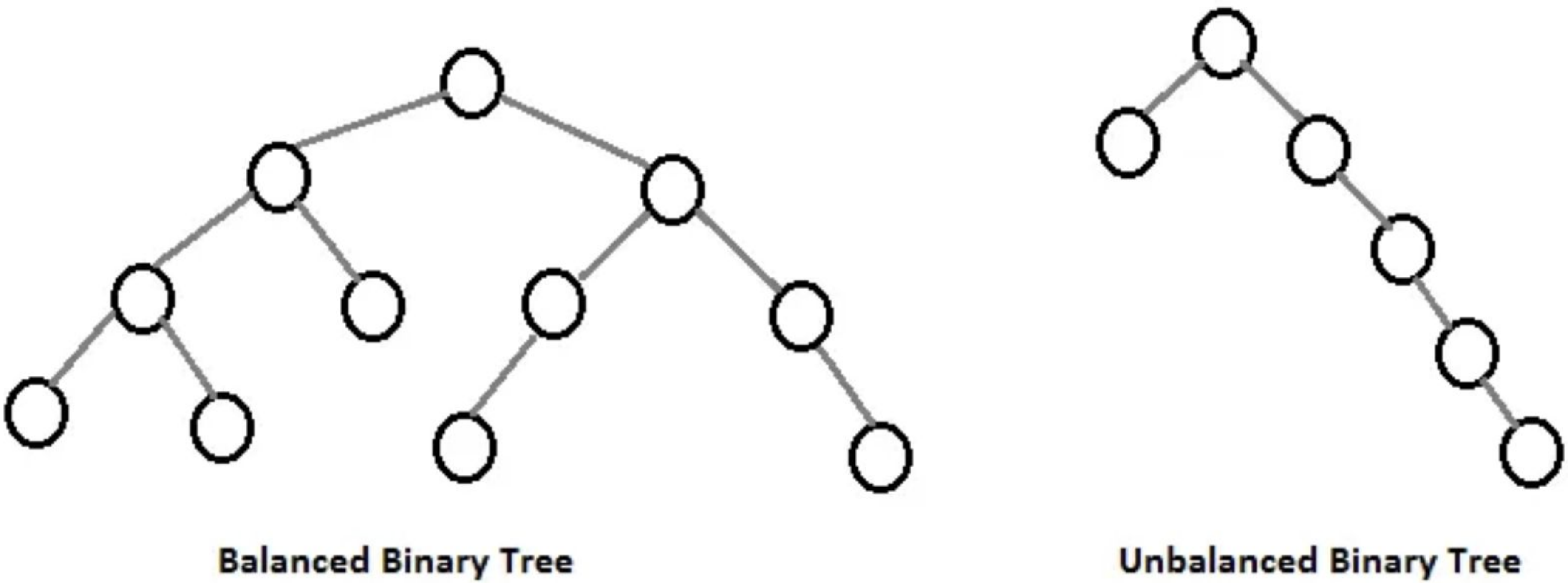
CLC - Red Black Tree

Duc Vo & Spencer Meren

Definition

_ A recursive tree is one of the most common data structures with high efficiency for sorting and searching of around O(log n)

_ A normal binary tree, however, has a slight problem with self-balancing, since the first node would be chosen as the root, and there is a high chance that the tree would be more populated on either the left or right branch, making it unbalanced and reducing its efficiency to O(n)

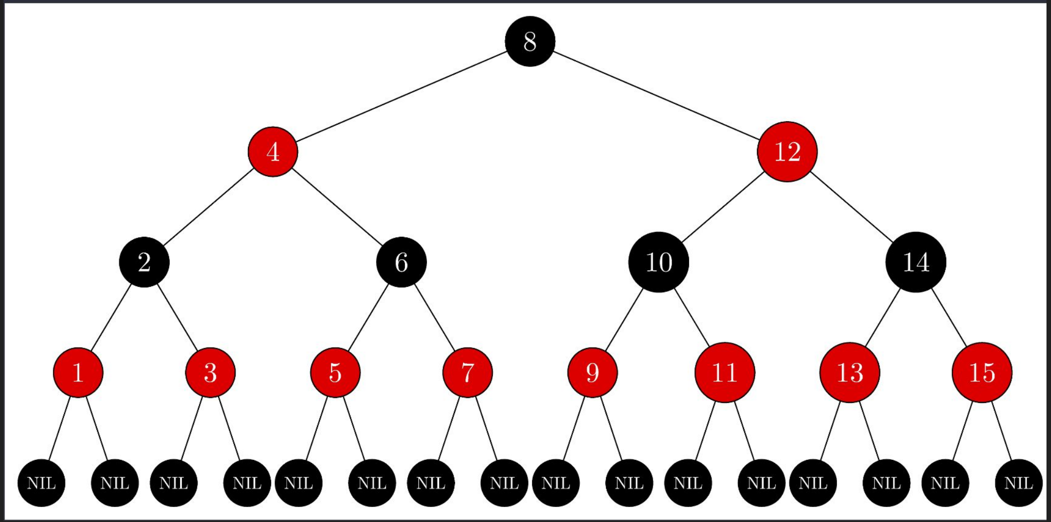


https://estherkang14.medium.com/javascript-data-structures-trees-pt-2-a5713b5a6cf5

Definition (cont.)

_ A Red-Black tree is a self-balancing tree, following a certain rule set in order to keep the efficiency of O(log n)

_ The rules are mostly centered around keeping the black and red nodes in check



https://walkccc.me/CLRS/Chap13/13.1/

Rules

1. The root is always black
2. The children of the red nodes are black, and the red node cannot have a red parent or a red child
3. All the leaves have the same black depth (“black ancestors”)
4. Every path from the root to the leaves must have the same number of black nodes

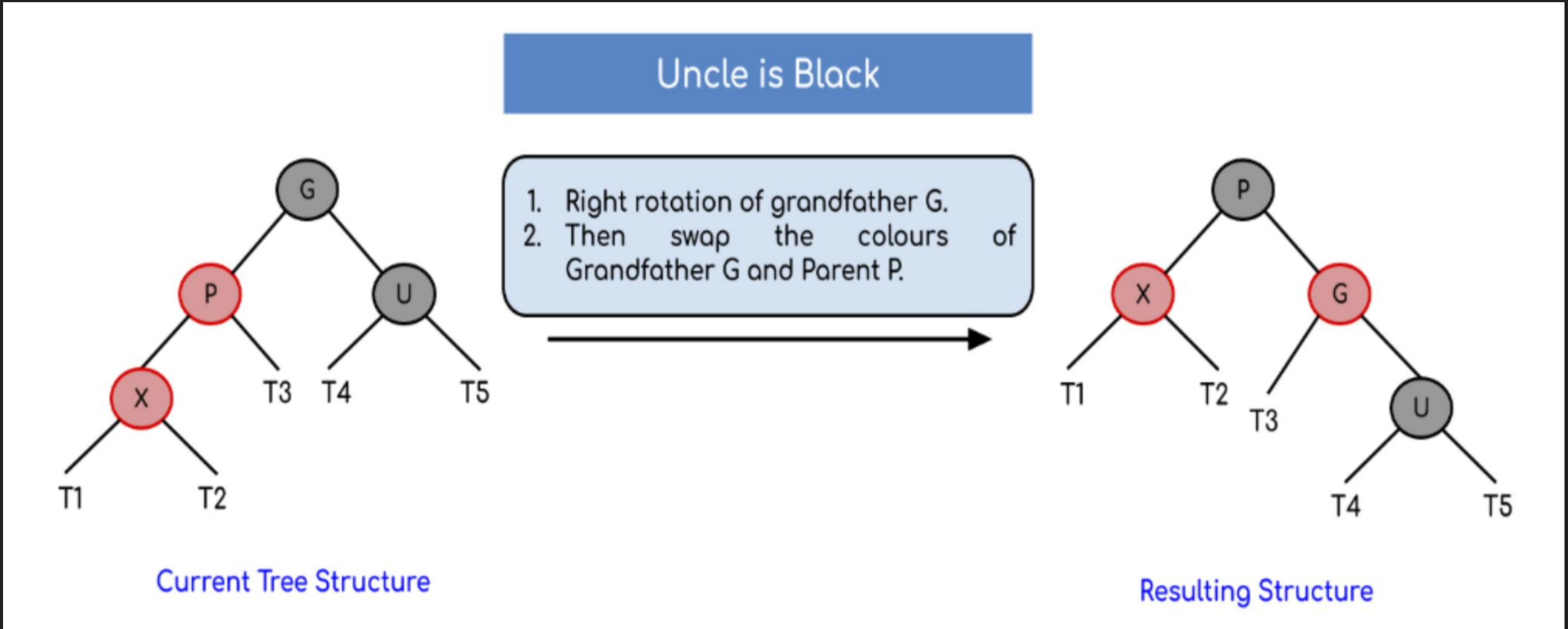
Rules (cont.)

_ During insertion and deletion, whenever one of these rules is compromised, the tree will be balanced around it

_ There are two solutions:

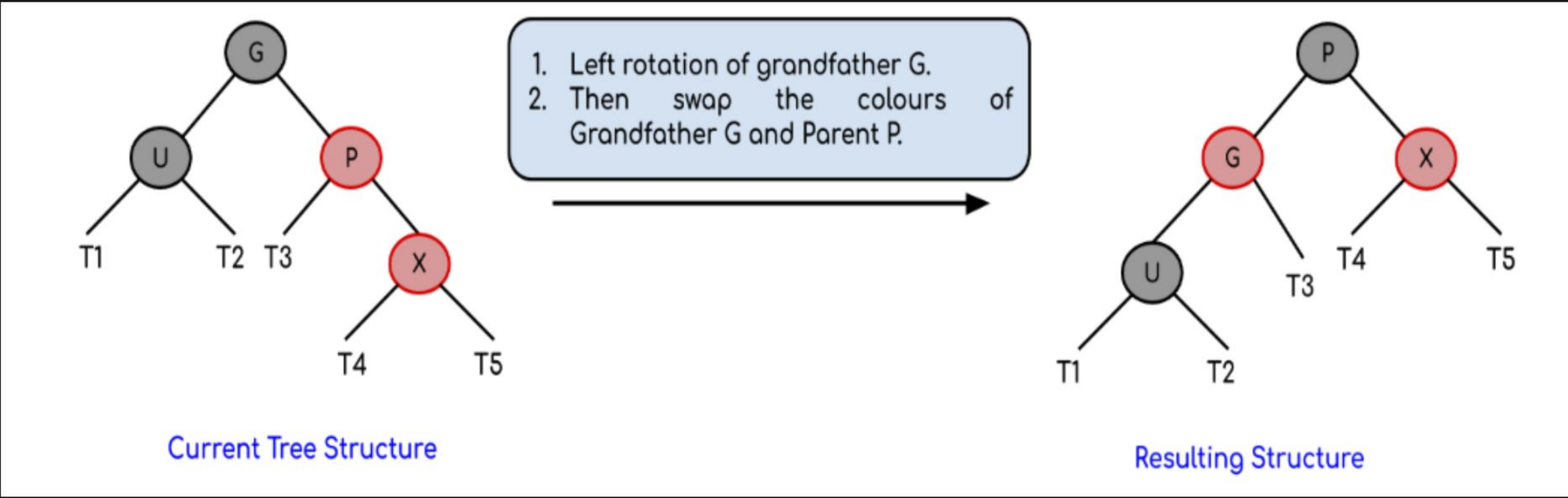
+ Recolor
+ Rotate

Left - Left rotation



https://www.geeksforgeeks.org/insertion-in-red-black-tree/

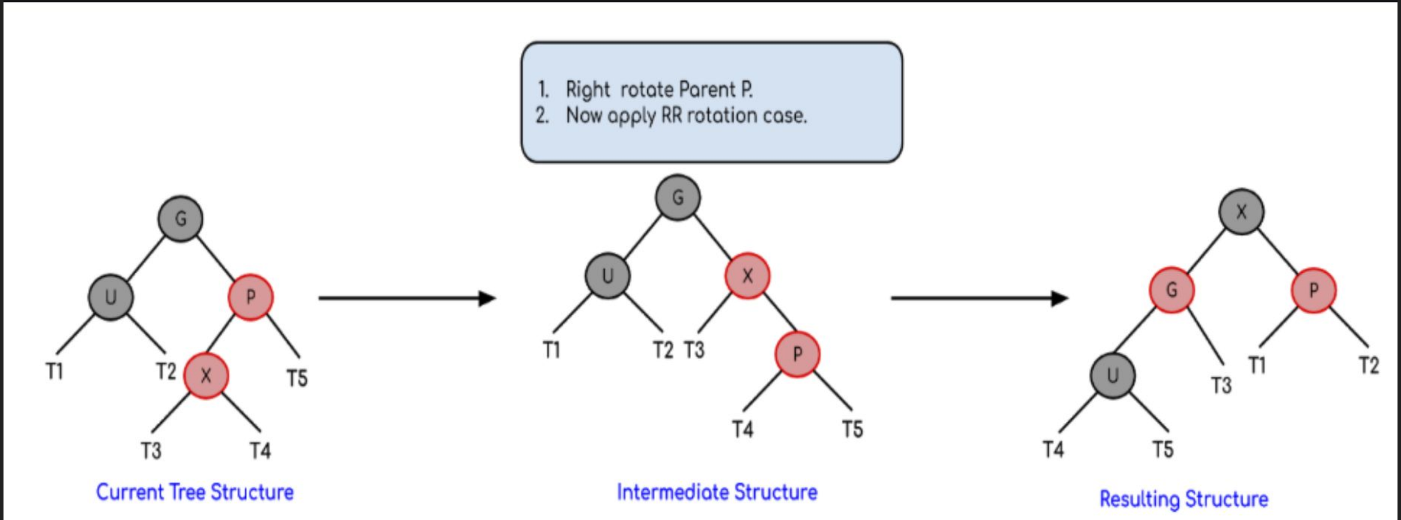
Right - Right rotation



https://www.geeksforgeeks.org/insertion-in-red-black-tree/

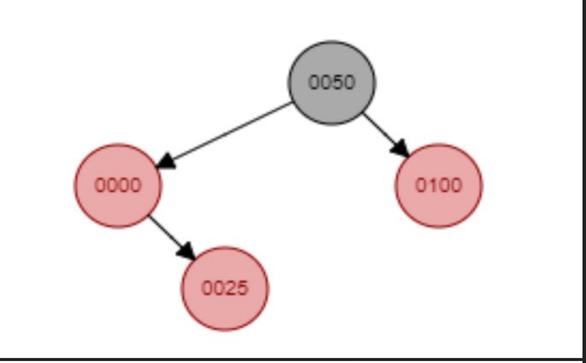
Right - Left and Left - Right Rotation

_ For this case, we would simply move the parent so that the tree is set up for a right-right or left-left rotation, for example

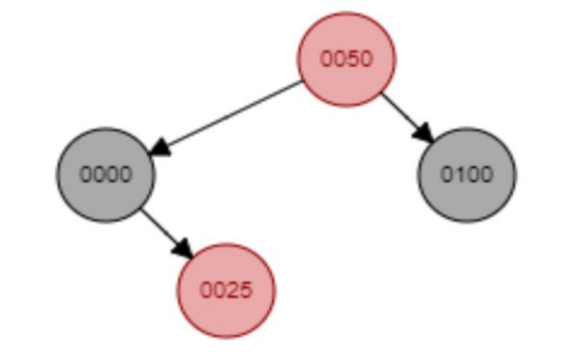


https://www.geeksforgeeks.org/insertion-in-red-black-tree/

Starting with a simple Red-Black tree, the root is black node, and the leaves are red

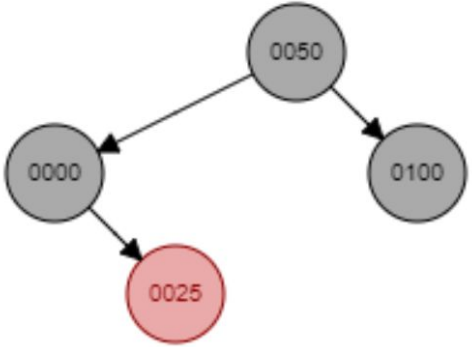


Inserting 25, this would end up like this

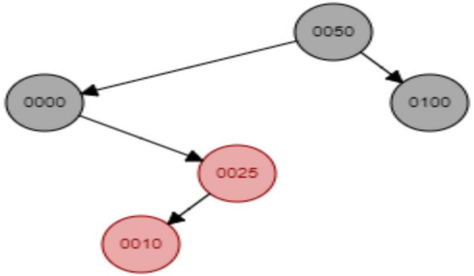


But since both 0 and 25 is red, the parent (0 and 100) will be black and the grandparent being red

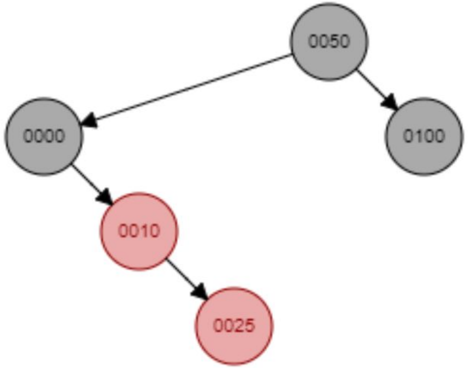
https://www.cs.usfca.edu/~galles/visualization/RedBlack.html



The root must be black, so we change 50 back to black

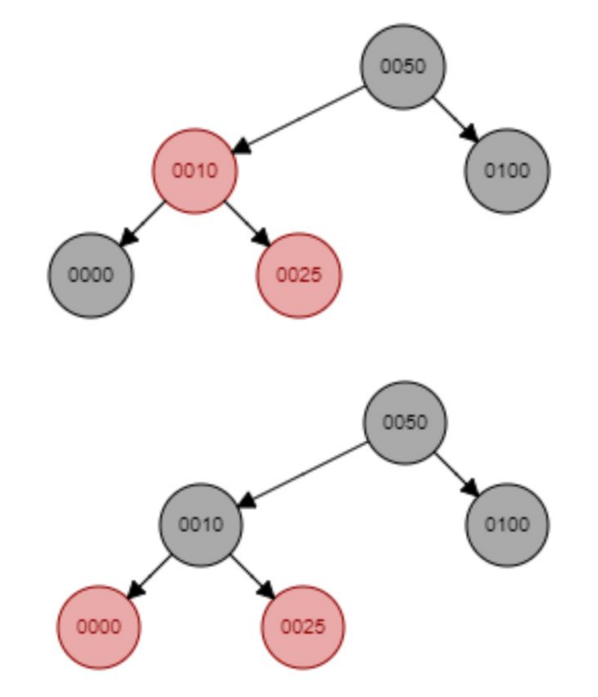


Adding 10 would be like this, making the tree unbalanced, this is a right-left rotation



First, a right rotate with the parent

https://www.cs.usfca.edu/~galles/visualization/RedBlack.html



Follow up with a left rotate

Finally, recolor the node accordingly

https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Advantage

_ Red-Black tree is very efficient for insertion and deletion

_ But for searching, the AVL tree is much more efficient

Data Structure	Time Complexity								Space Complexit
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Stack	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Queue	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Singly-Linked List	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Doubly-Linked List	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Skip List	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n \log(n))$
Hash Table	N/A	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Binary Search Tree	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Cartesian Tree	N/A	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$
B-Tree	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
Red-Black Tree	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$

https://www.bigocheatsheet.com/

Work Cited

GeeksforGeeks. (2023, January 31). *Insertion in red-black tree*. GeeksforGeeks. Retrieved March 25, 2023, from https://www.geeksforgeeks.org/insertion-in-red-black-tree/

Kang, E. 승연 (2021, January 17). *JavaScript data structures: Trees (pt. 2)*. Medium. Retrieved March 25, 2023, from https://estherkang14.medium.com/javascript-data-structures-trees-pt-2-a5713b5a6cf5

Know thy complexities! Big O Notation. (n.d.). Retrieved March 25, 2023, from https://www.bigocheatsheet.com/

Red/Black Tree. Red/Black Tree Visualization. (n.d.). Retrieved March 25, 2023, from https://www.cs.usfca.edu/~galles/visualization/RedBlack.html