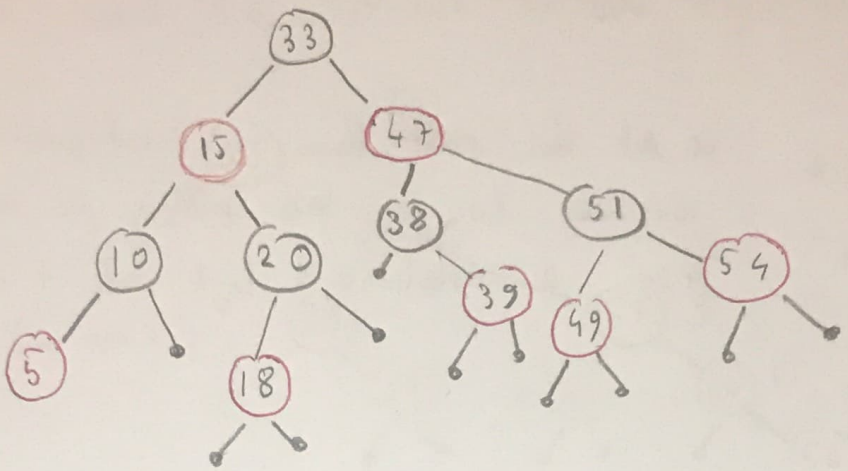
CS301 Homework 3

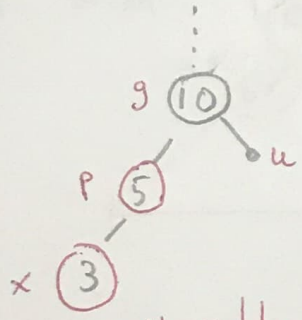
Ulaş Eraslan
25058

1-



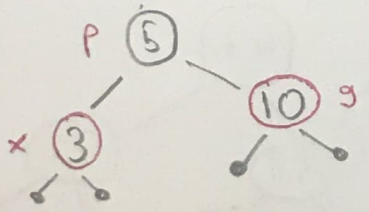
a) Insert 3

If I insert 3, it will be inserted as left child of 5.



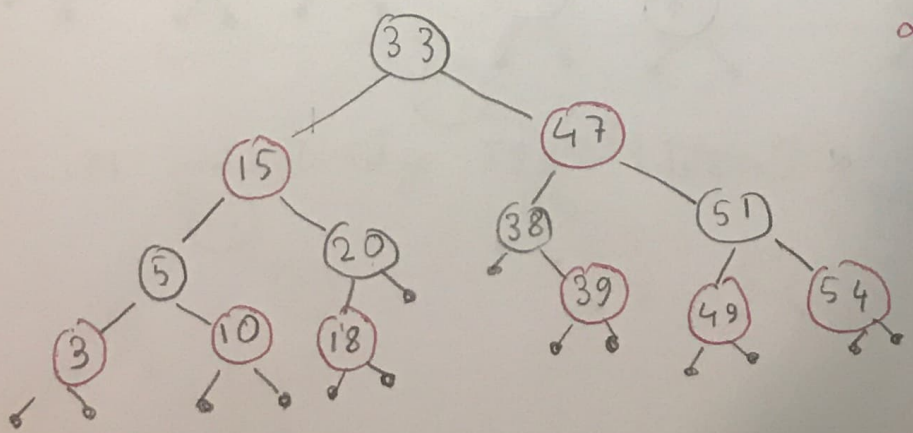
∝ At this point there is "red-parent-red-child" problem. To solve that we will apply right-rotate on 5 (p).

right-rotate



∝ Fix color of p black.

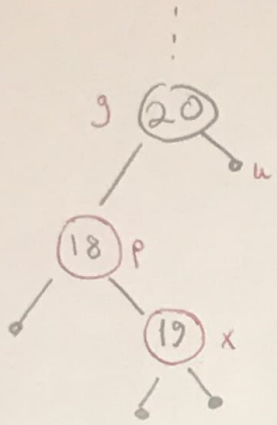
∝ Fix color of g red.



∝ Completed RBT after inserting 3.

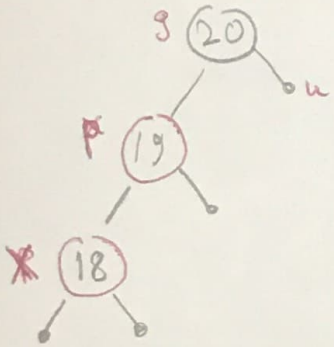
b) Insert 19

If I insert 19, it will be right child of 18.

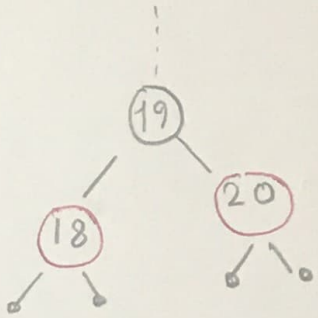


∝ At this point, there is "red-parent-red-child" issue so to fix this problem, we may use 2 rotation. 1 left and 1 right. (Case 2)

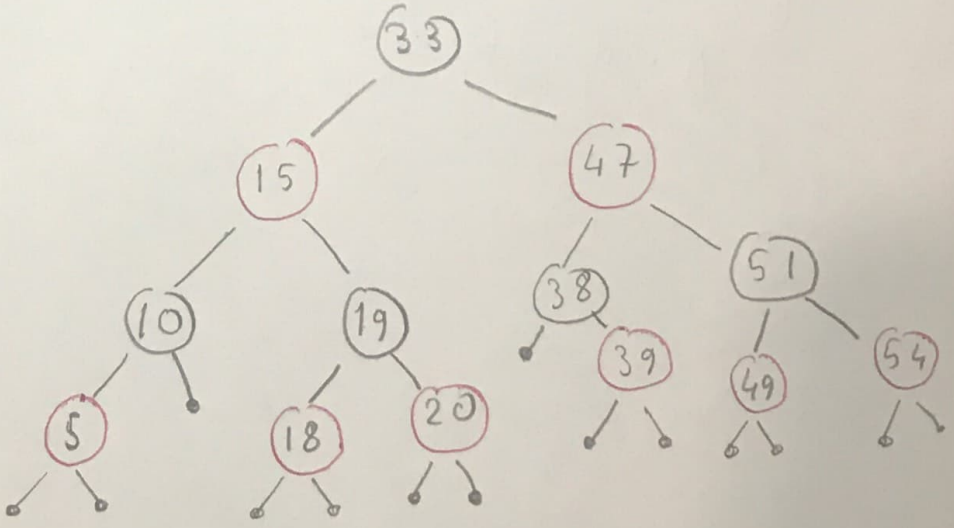
⇓ left-rotation



right-rotation ⟹ to fix "red-parent red-child" issue



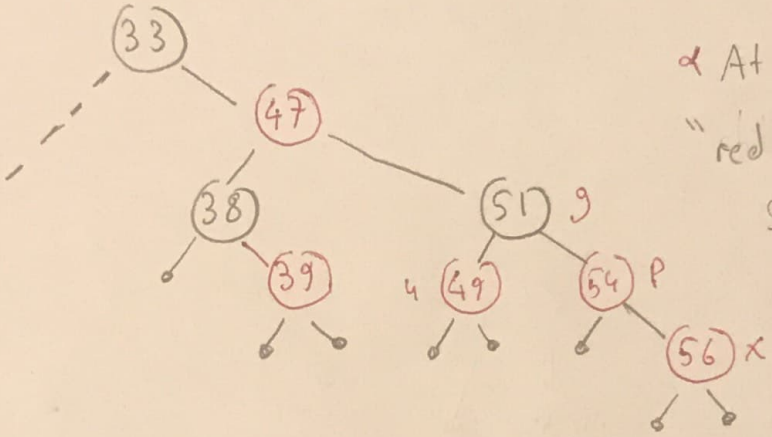
∝ Make g red
Make p black



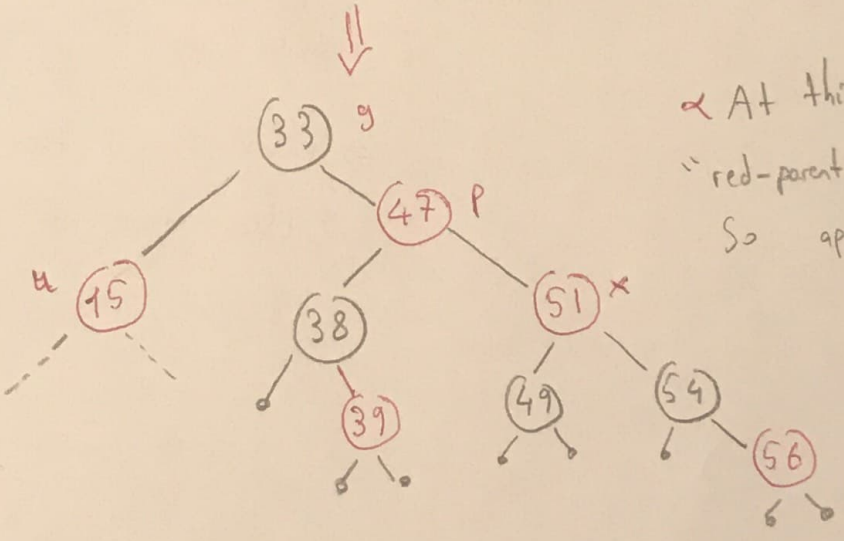
∝ Completed RBT after inserting 19.

c) Insert 56

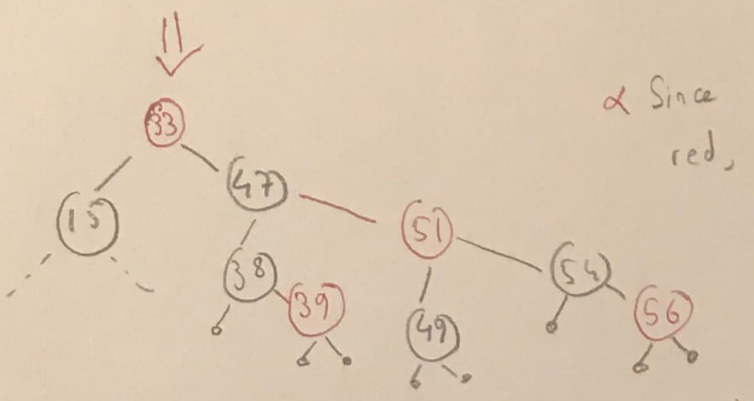
If I insert 56, it will be the right child of 54.



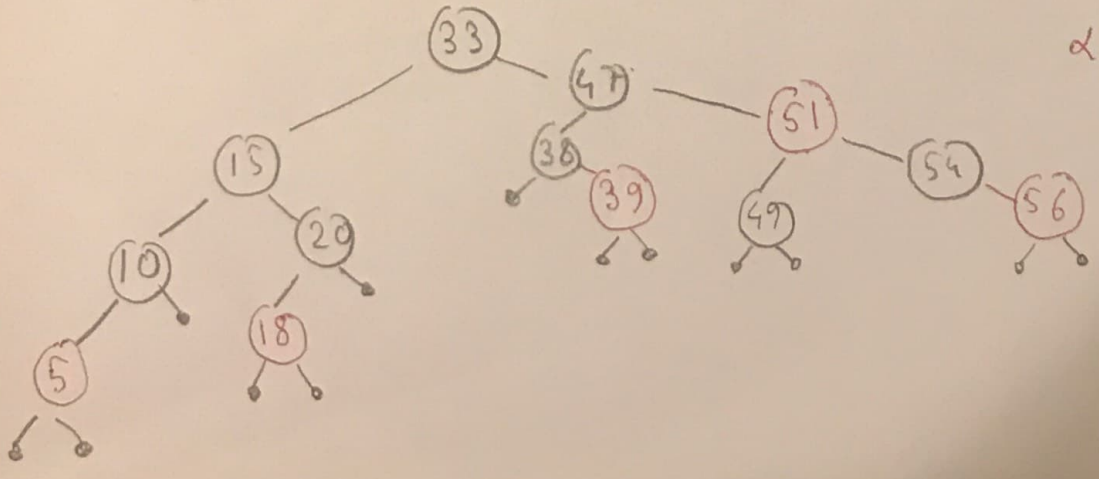
∝ At this point we have "red-parent-red-child" problem. So to fix this, push color of parent ad uncle to grandparent. (Case 1)



∝ At this point, there is "red-parent-red-chid" issue again. So apply the previous solution.



∝ Since root can not be red, convert it to black.



∝ Completed RBT after inserting 56.

2- a) To avoid from calculating same sub-problems, we can use something similar to recursive. This method traverse over the items and take the maximum value given items that are subject to weight constraint. We may utilize a 2-D matrix to store values..

So the algorithm is the following:

findOptimalValue(W, n, values_of_items, weight_of_items){

- W: weight constraint
- n: total number of items
- values_of_items: An array that contains values of items
- weight_of_items: An array that contains weights of items

for(i, i<=n, i++) => iterate over all items

for(j, j<=W, j++) => iterate over each weight $\leq W$

if(i or j is 0)

put 0 to that cell in 2-D matrix

else if weight of the item in cell > Weight Constraint

put the value of item in the previous row and same column (T[i-1][j])

else

find max value of following and put it corresponding cell

$$\max\begin{cases} 1 \cdot val[i] + T[i-1, j-\text{weight of } i] \\ 2 \cdot T[i-1][j] \end{cases}$$

At the end, return the 2-D value matrix (T)

}

b) Since we have 2 for loops we should consider them.

Inner loop iterates W times.

Outer loop iterates n times.

$$O(W * n)$$

c) To find best solution, we can construct a matrix. In that matrix columns = W, rows = items

	value	weight
1)	12	2
2)	10	1
3)	20	3
4)	15	2

	0	1	2	3	4	5.
	0	0	12	12	12	12
	0	10	12	22	22	22
	0	10	12	22	30	32
	0	10	15	25	30	37

- The max value that we can carry with 5 tons is 37 * 0.001

- Items that we use are:
Item 1 - Item 2 - Item 4