Final Assignment

Date : 15/04/22
Day : friday

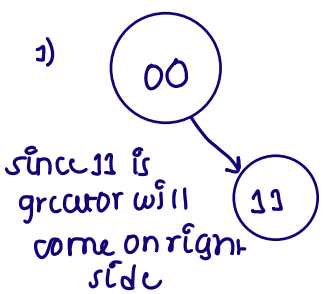
↳ Binary Search Trees Rules

1) Nodes in leftsubtree of a node should be less then that node
2) The nodes present in the right subtree of a node should be greator than that node
3) Above two conditions should satisfy for all nodes

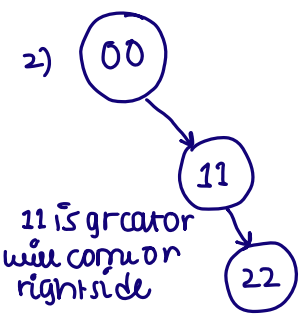
VALUES :

00, 11, 22, 66, 111, 77, 88, 99, 100, 11, 33, 44, 55

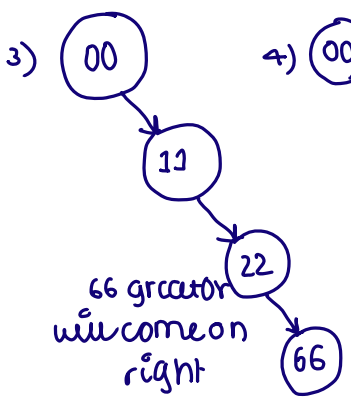
1)



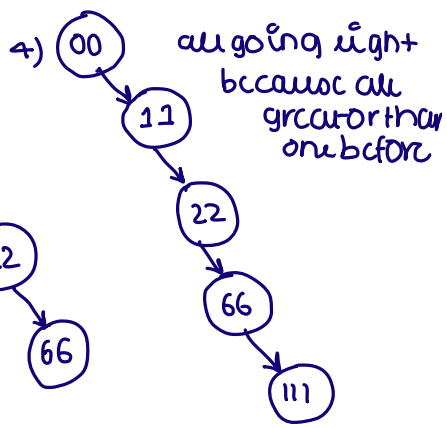
2)



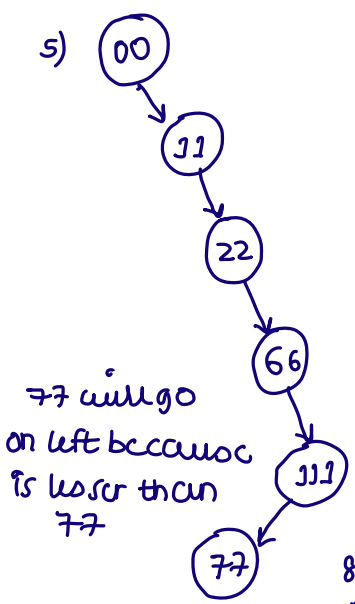
3)



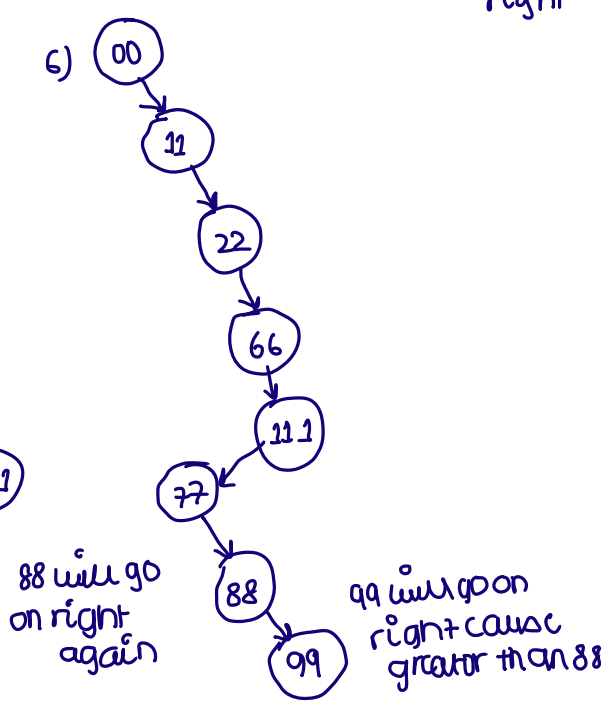
4)



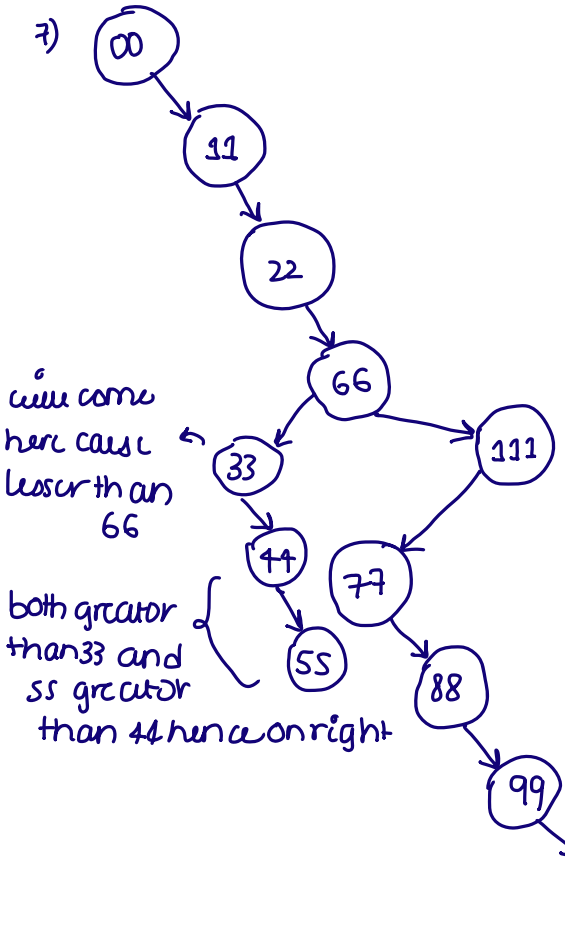
5)



6)



7)



IGNORING 11 DUPLICATES CANNOT BEADDED

Red Black Tree Rules

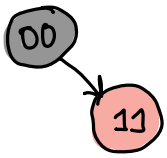
1) The root of the tree is always black
2) No two adjacent red nodes (red node cannot have red parent or child)
3) Every path from a node (including root) to any of its descendants NULL nodes has same number of black nodes
4) All leaf nods are black node

00, 11, 22, 66, 111, 77, 88, 99, 100, 11, 33, 44, 55

1)

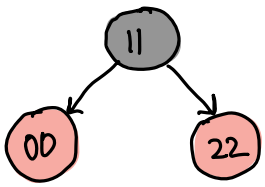
root is always black

2)



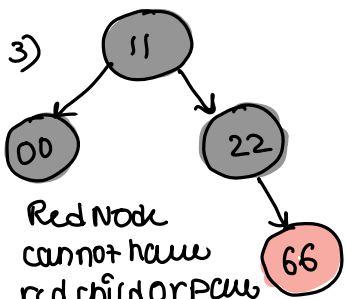
child of black node is red since greater hence on right

3)



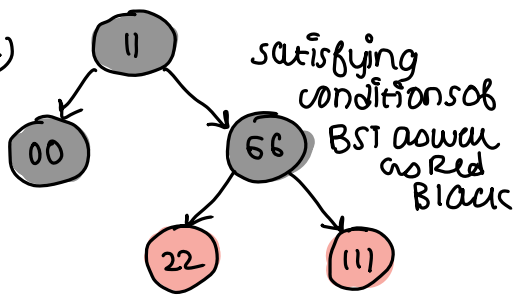
Rooting bc cause if not rotated conditions will not satisfy

3)



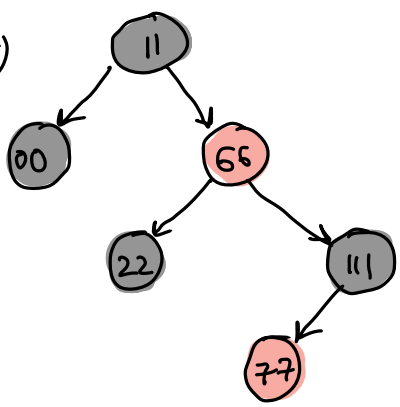
Red Node cannot have red child or pair not satisfying condition

4)

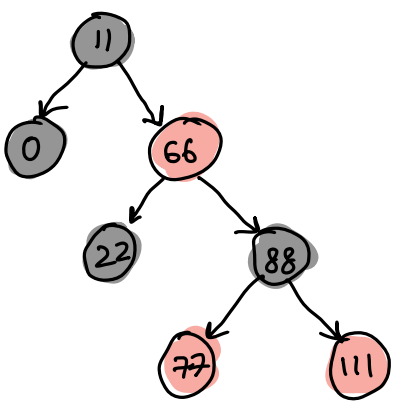


satisfying conditions of BST as well as Red Black

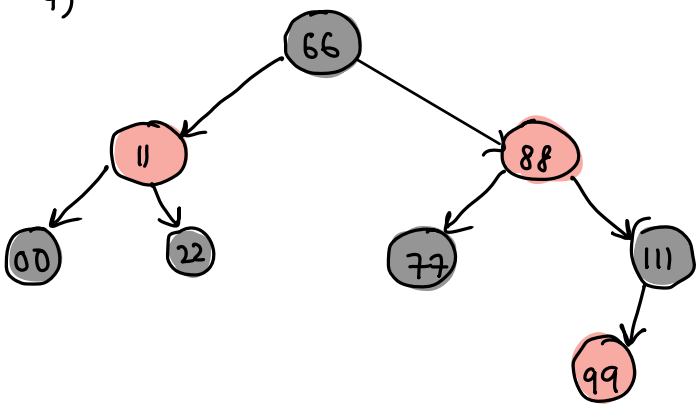
5)



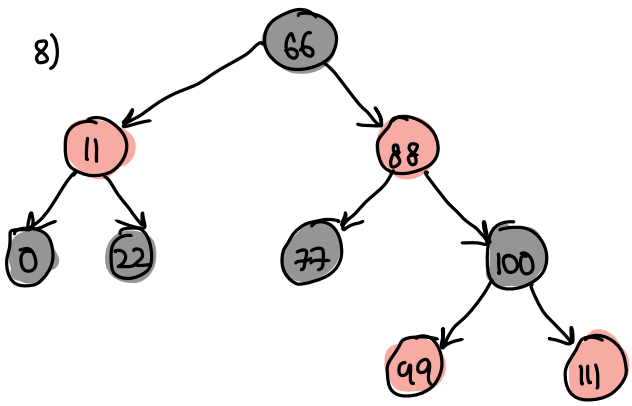
6)



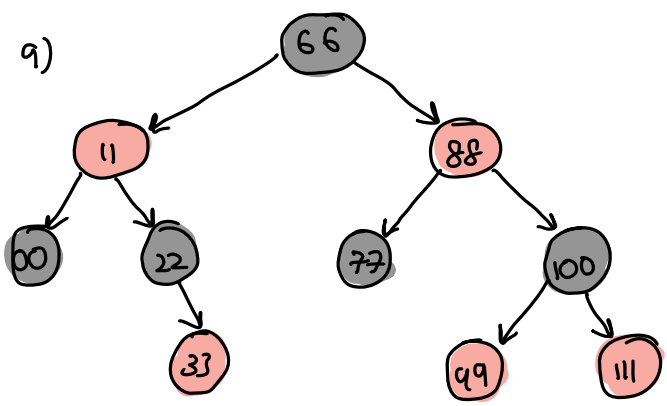
7)


66
11
88
00
22
77
111
99

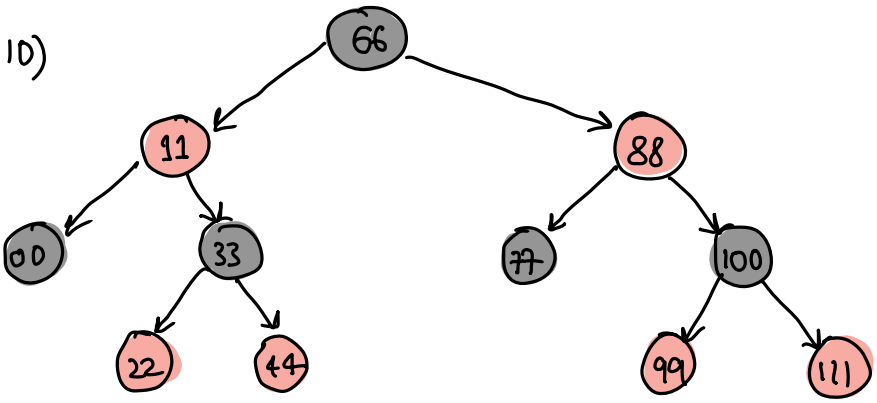
8)


66
11
88
0
22
77
100
99
111

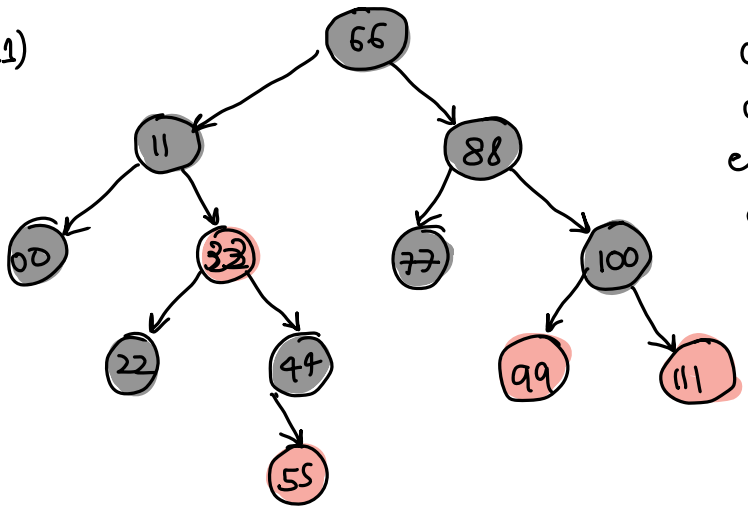
9)


66
11
88
00
22
77
100
33
99
111

10)



11)



check all BST and rcd black conditions satisfied equal level of black node etc and that all values on Right are greator and all on left are lesser
All levels of black equal for each node that is true as well.

2-3-4 Tree Rules

1) 2-node has one key and two child nodes (just like BST)
2) 3-node has two keys and three child nodes
3) 4-node has three keys and four child nodes

} all should be satisfied

00, 11, 22, 66, 111, 77, 88, 99, 100, 11, 33, 44, 55

1) 0
↓
1 Node

2) 0 | 11
continued
cannot be split

3) 0 | 11 | 22
continued
cannot be split

4) 11
0 | 22 | 66
can be split as
0 less than 11 and 22
and 66
a two node can greater
have one key

5) 11
0 | 22 | 66 | 111
continuation another
greator value added
to right

6) 11 | 66
0 | 22 | 77 | 111
a three node situation is possible
none made

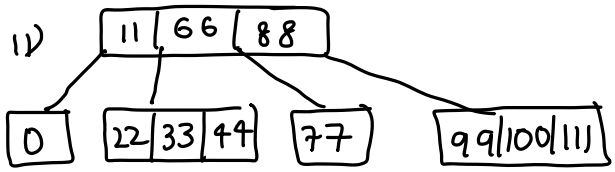
7) 11 | 66
0 | 22 | 77 | 88 | 111
3 node has 2 keys
continued

8) 11 | 66 | 88
0 | 22 | 77 | 99 | 111
4 node has
3 keys

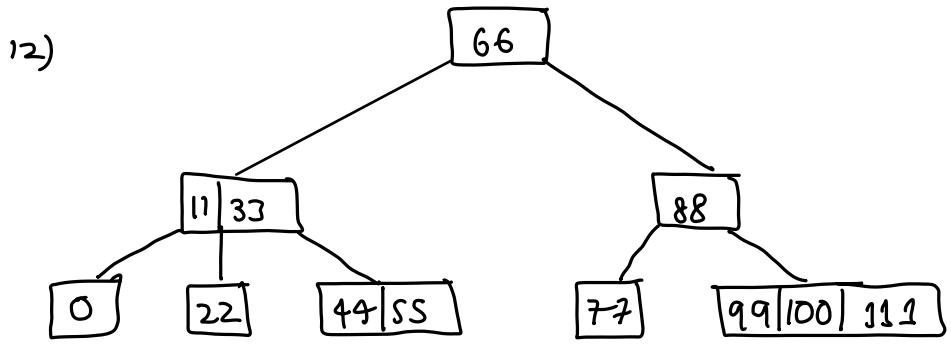
9) 11 | 66 | 88
0 | 22 | 77 | 99 | 100 | 111

10) 11 | 66 | 88
0 | 22 | 33 | 77 | 99 | 100 | 111
keep doing this
satisfying cond-
ition of Nodes

11)

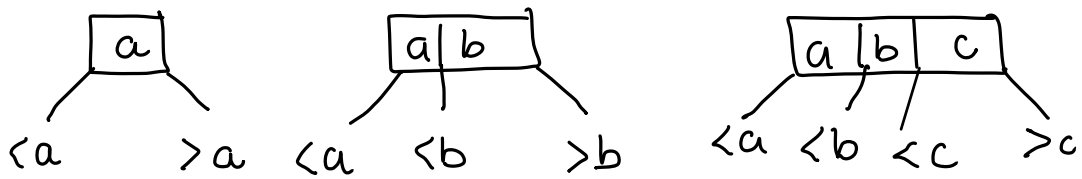


12)



The logic is to make the tree completely balanced

so it went like



(a) 2 Node (b) 3 Node (c) 4 Node

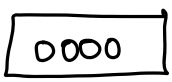
<

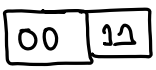
Rules for BT Datastructure:

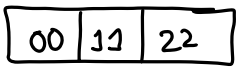
1) Every node in a B-tree contains at most m children
2) Every node in a B-tree except root node and the leaf node contain m/2 children
3) The root node must have atleast 2 nodes
4) All leaf nodes must be at same level

6 degree as 6 keys per node

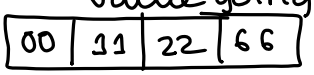
00, 11, 22, 66, 111, 77, 88, 99, 100, 11, 33, 44, 55

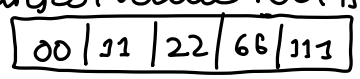
1) 

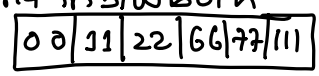
2) 

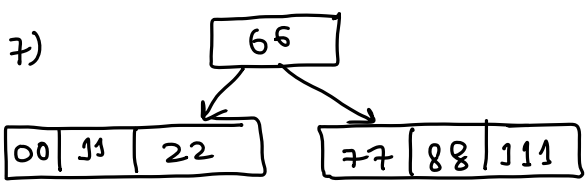
3) 

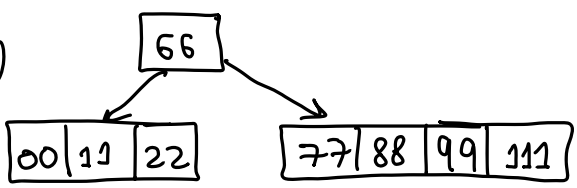
its a seven degree first of all make sure all the rules satisfy you can add 6 then it cannot go beyond it'll break make sure each left value is smaller then the other going left smallest value going right largest value root is something in b/w both

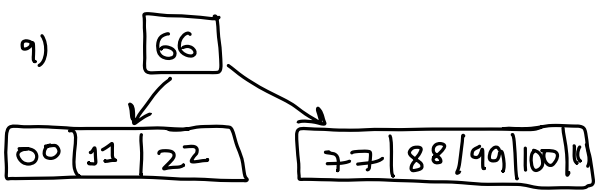
4) 

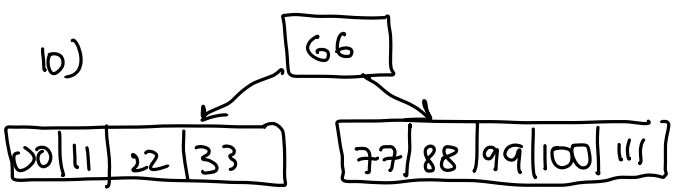
5) 

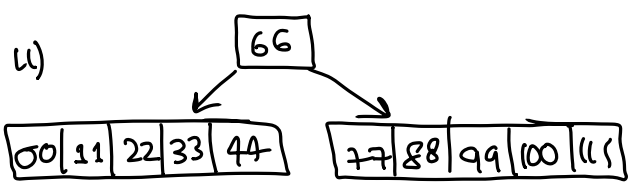
6) 

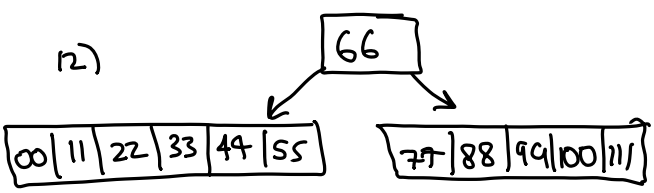
7) 

8) 

9) 

10) 

11) 

12) 

Reflection:

BST = has the longest height

2-3-4 = lesser height compared to BST due to the rules and balancing properties, store multiples values in one or a node has more height than B

B-trees = have the shortest height compared to all as they're of seven degrees store multiple values in node

Red-Black Trees = These are not as short as 2-3-4 or B-trees but quite shorter compared to Binary search Tree which is very long and extensive

Understandability:

Understanding each tree is not a very tough task although all of them are hard to implement in comparison to Binary Search Tree since they all have certain set of rules that always need to satisfy

2-3-4 and Red Black trees are almost the same level of difficulty in understanding in terms of implementation however B-trees are much harder to implement because of the complicated splitting mechanism hence 2-3-4 and Red Black trees would be easier to implement and Binary Search Tree would be easiest to understand and implement.

FOR ALL CASES SATISFY ALL RULES TO ASSURE
TREE IS PERFECT
ALL DUPLICATES HAVE BEEN AVOIDED

Easiest to understand and implement = BST fairly straightforward with easy Rules

2nd Easiest Red Black easier to implement rules not as easy as BST but fairly simple

3rd Easiest 2-3-4 hard to implement and understand due to self balancing rules still easier than B-trees

4th B trees hard to implement and understand because of splits and degrees