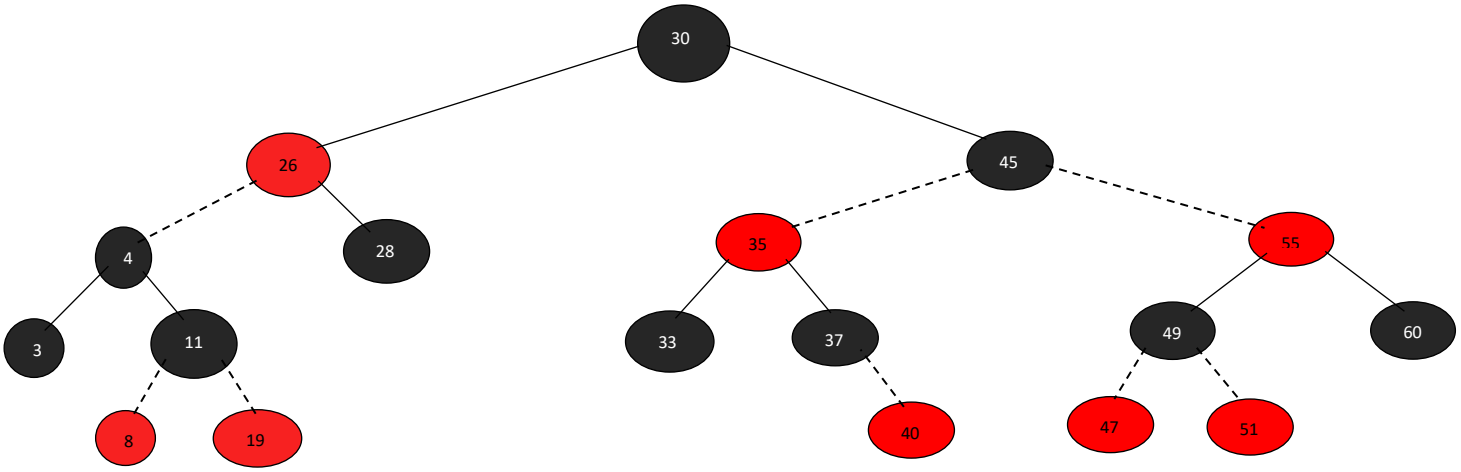
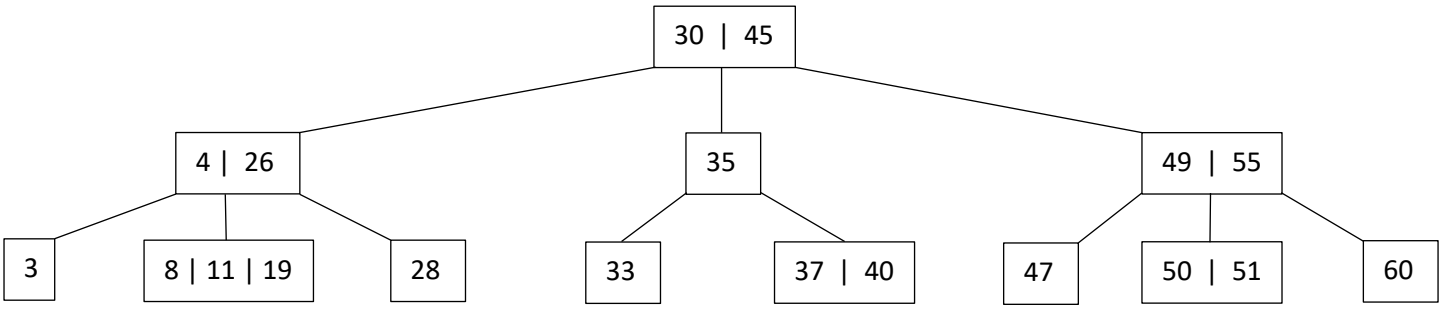

30
26
45
4
28
35
55
3
11
33
37
49
60
8
19
40
47
51


30 | 45
4 | 26
35
49 | 55
3
8 | 11 | 19
28
33
37 | 40
47
50 | 51
60

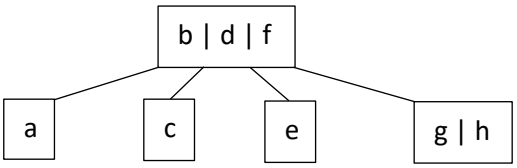
2-)

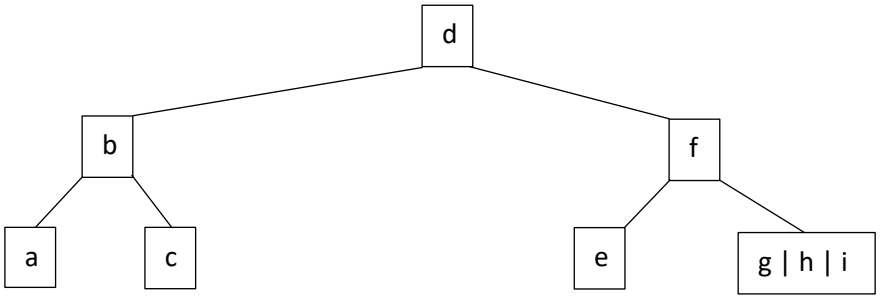
a-)

max number of nodes can be: $(h^3 - 1)$

b-)

when “i” inserted tree will be expand.





c-)

It is same running time with binary search tree sorting which is O(n logn) because they have same order. Smaller value in the left child, grater value in the right child.

d-)

Since root have to be black node and if root has a red child then we can easily say, not each subtree is a red-black tree. Eg. 30 and 26 are in same node(first tree in that document).

3-)

a- First create a hash table with size n. Then loop over the given array. While looping check the hash table whether “target – arr[i]” (arr[i] is current item of the loop) is already in the hash table. If hash table contains “target – arr[i]” (target – arr[i] + arr[i] = target) then array contains pair which sum is target. (Hash Table insertion O(n) time, for n element O(n) time and search in hash table is O(1) time, for n element O(n) time. Total is O(2n) which is equal to O(n))

b-

Linear probing

Slot	0	1	2	3	4	5	6
Content	30	15	22	11	14	18	

Quadratic probing

Slot	0	1	2	3	4	5	6
Content	30	15	22	11	14	18	