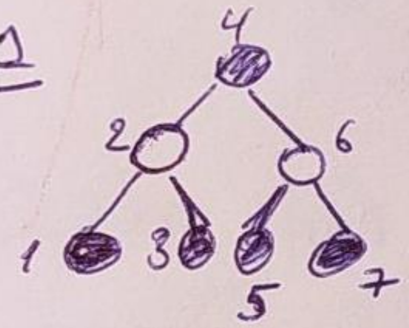
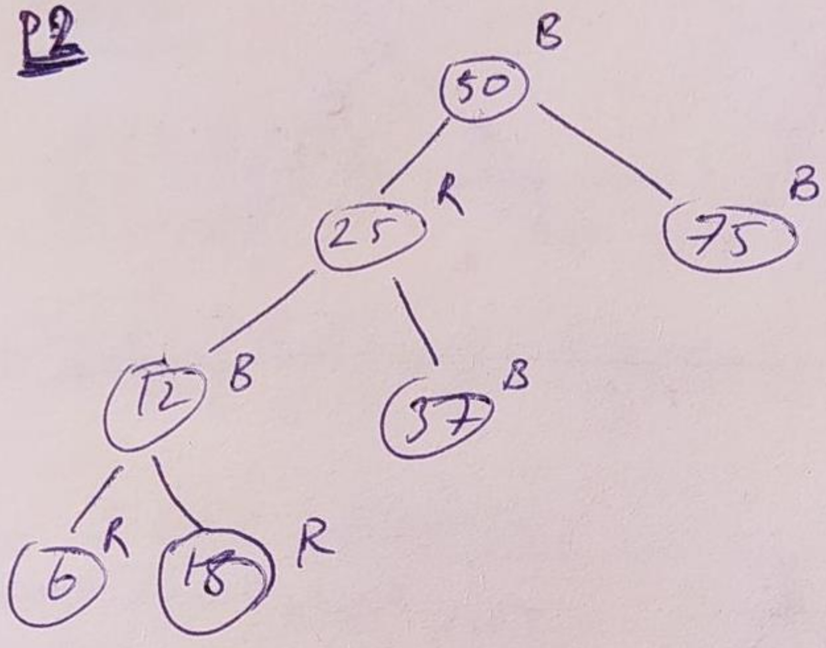
P.1



This Red-Black tree is a valid tree but we can not obtain this tree with any insertion sequence. ~~Bh~~ The last inserted node must be a red & leaf node. But in the tree above there is not a single red leaf node.

P2

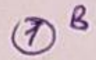


This is a Valid Red-Black tree from the lecture (lecture-6, P.25) But it does not suffice the AVL tree requirements.

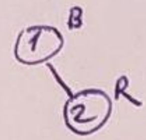
Prob.3.

(a) 1, 2, 3, 4, 5, 6, 7, 8

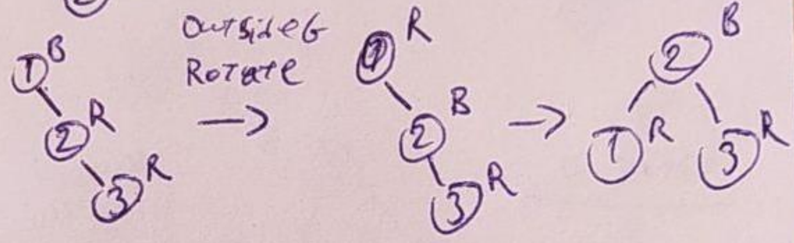
insert-1:



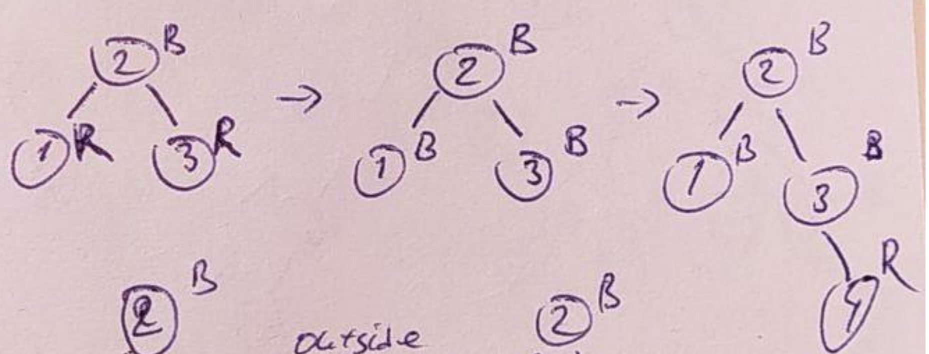
insert-2:



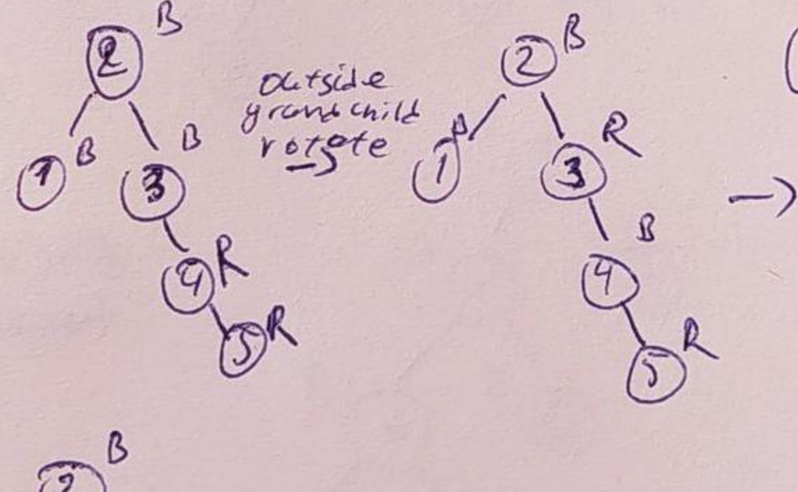
insert-3:

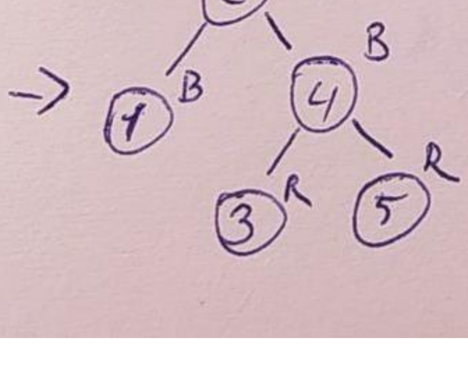


insert-4:

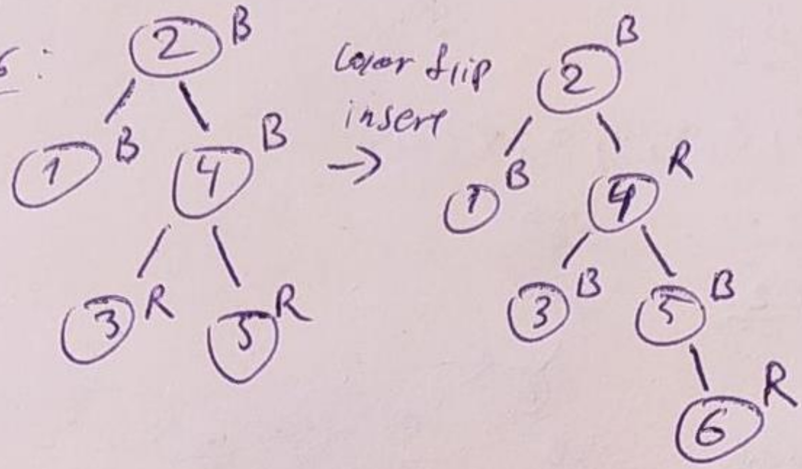


insert 5:

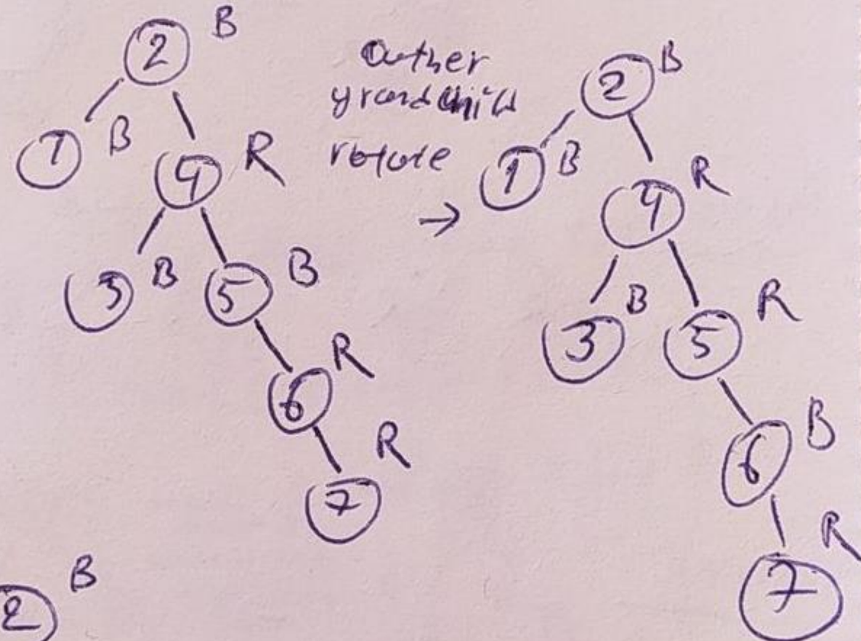


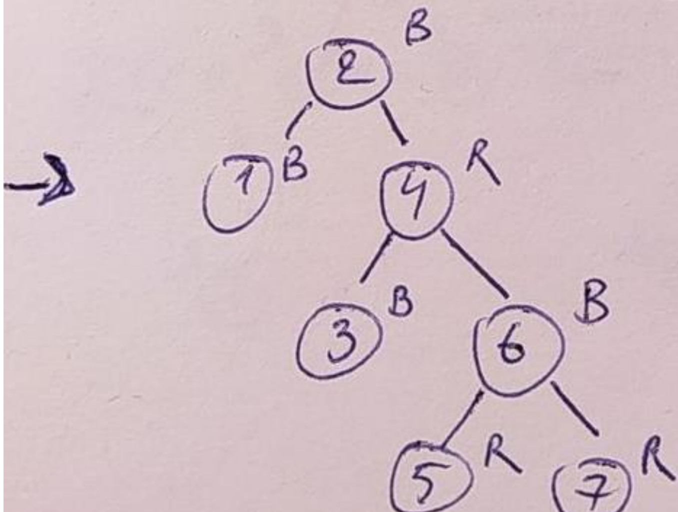


insert-6:



insert-7:

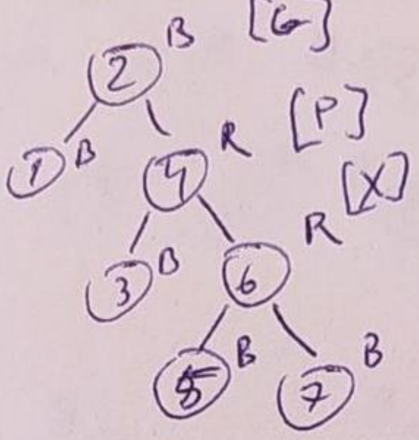




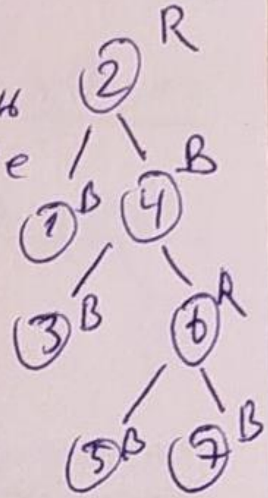
$Bh(n) = 2$

~~for every~~

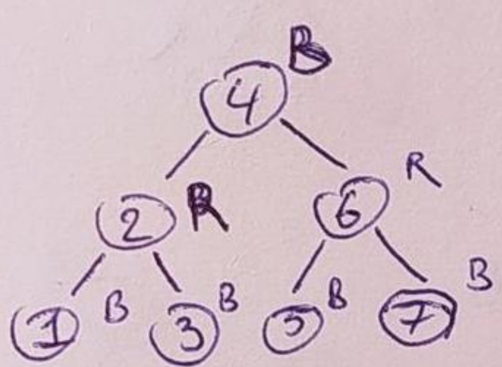
insert-8:



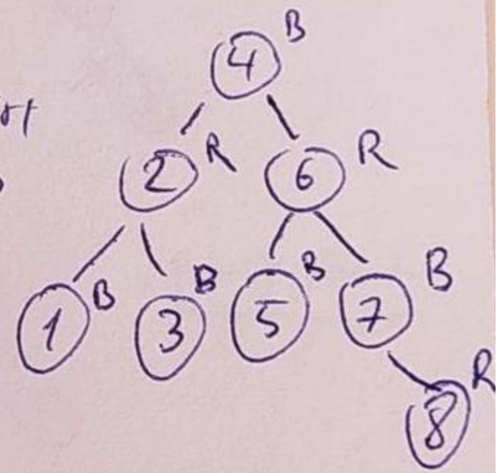
outher grandchild rotate →



→

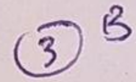


insert →

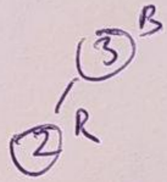


(b) 3, 2, 1, 4, 5, 6

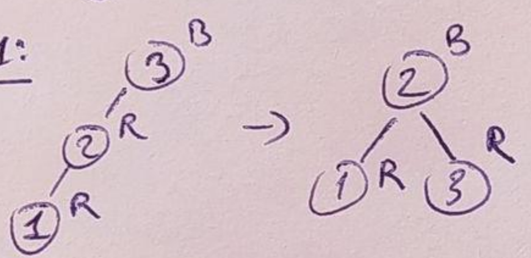
insert-3:



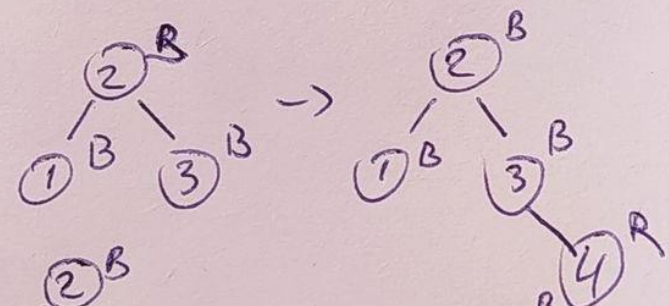
insert-2:



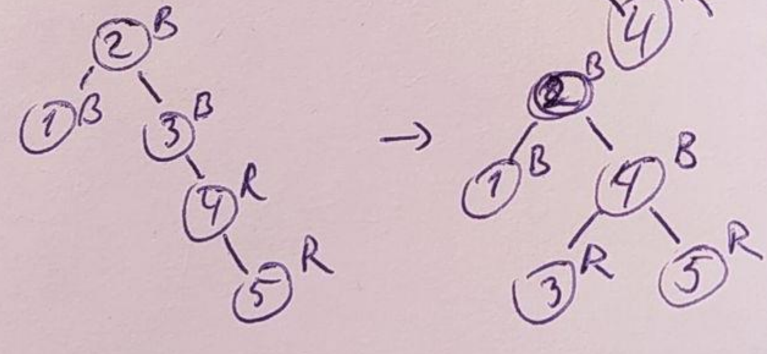
insert-1:



insert-4:

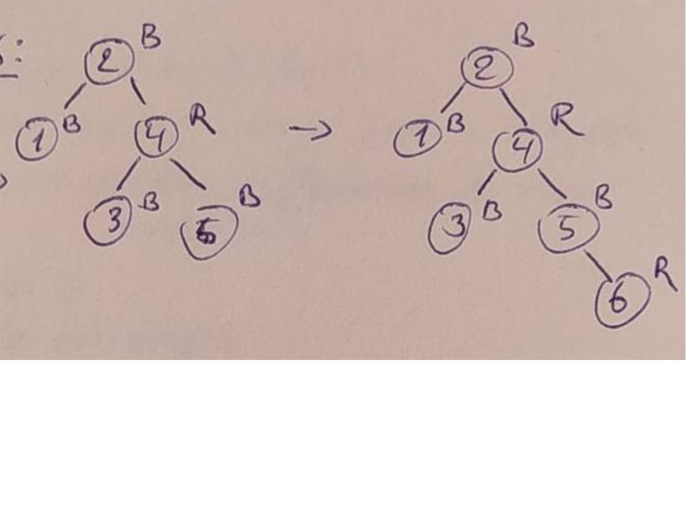


insert-5:



insert-6:

(color flip) ->



4. The algorithm that I’m proposing uses only a Binary search. Hence the running time of the algorithm is O(log N) which is o(N)

```
Algorithm Problem4 (arr):
    input: sorted array of distinct integers
    output: is there any n that
                arr[n] = n

    L ← 0
    R ← arr.length

    While L < R:
        mid ← (L+R) / 2
        if arr[mid] == mid then
                return true

        if arr[mid] < mid then
                L ← mid + 1
        else
                R ← mid - 1

    return false
```