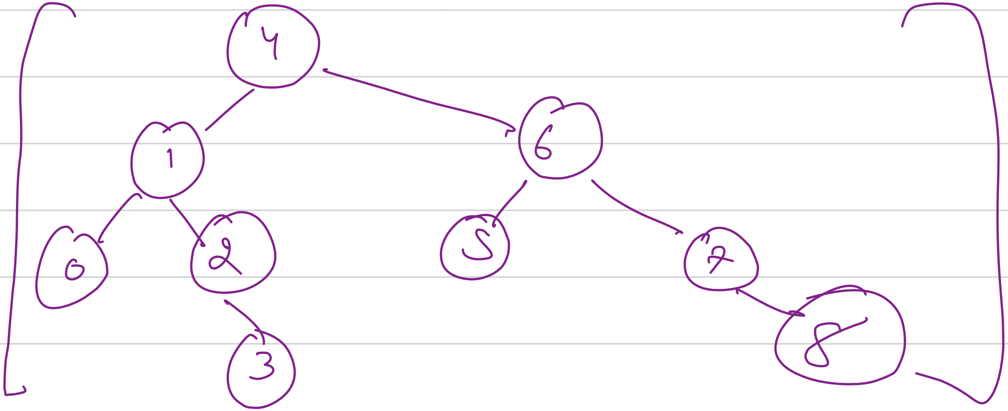
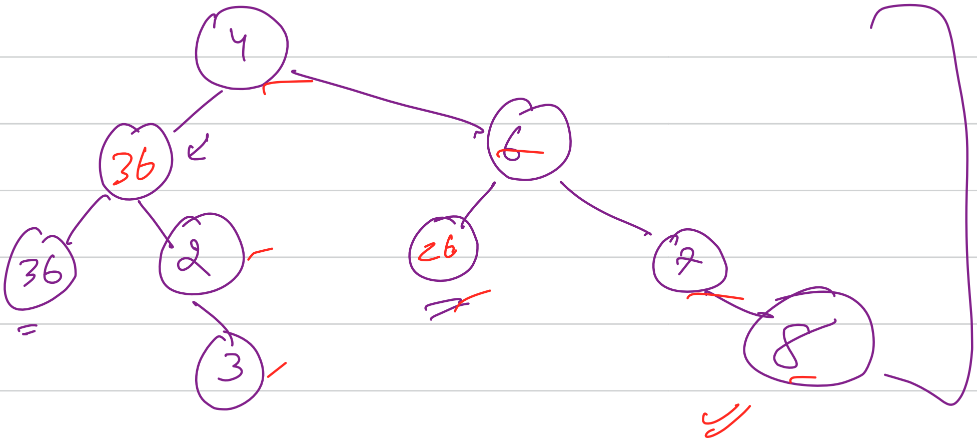
Binary Search Tree

Q:) You're given a BST, convert that BST into a new tree, where every node's value of original BST is replaced by the sum of original value & all the nodes having values greater than the original node.

Constraints →

1 ≤ no. of nodes ≤ 10^6 ✓



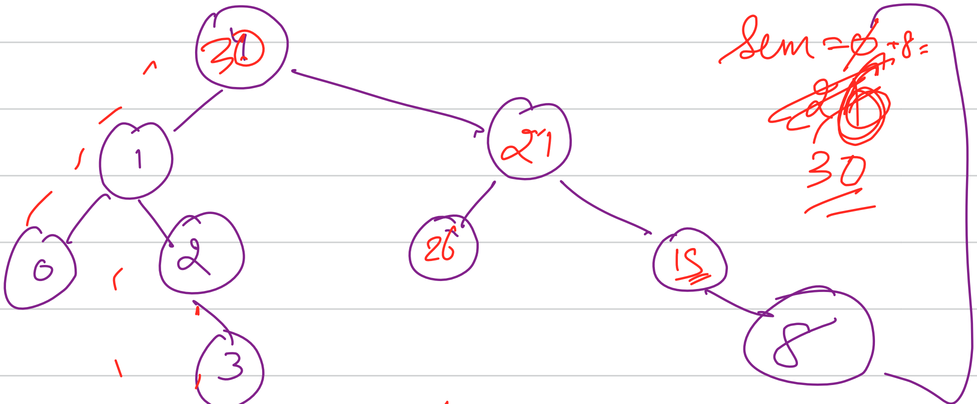


① → From each node traverse every thu on the right subtree again & again & Sum the values

→ efficient → You can precompute Sum of all greater nodes

For any node, if we want to precompute, the sum of larger nodes, we need to traverse them first

BST → for every node → larger nodes → RST



⇒ TC → O(n)

we're touching each node once

SC → O(n) → loose bound

→ O(h) ✓

what kind of traversal is this??

call stack

BBST

Pre → node
(left)
(right)

In → (left)
node
(right)

(Right())
node
(left)
→ Reverse Inorder

Post → (left)
(right)
node

Q=> You're given 2 integer arrays of odd length. You're supposed make pair (p_1, p_2) such that $p_1 \in A_1$ $p_2 \in A_2$. with a constraint that the xor of each pair should be equal.

$\rightarrow [a, b, c] \rightarrow n$

$\rightarrow [d, e, f] \rightarrow n$

$\boxed{len \le 10^6}$

$a \,\hat{}\, e = v$

$b \,\hat{}\, f = v$

$c \,\hat{}\, d = v$

$\hookrightarrow$ Yes

$a[i] \le 10^9$

$[a, b, c]$

$[d, e, f]$

$a \wedge e = v$ —— ①

$b \wedge f = v$ —— ②

$c \wedge d = v$ —— ③

We xor ① ② ③

$$(a \wedge e) \wedge (b \wedge f) \wedge (c \wedge d) = \underbrace{v \wedge v}_{0} \wedge v \rightarrow v$$

$$\boxed{\text{xor of elements from both array} = v}$$

for any x

$x \wedge x = 0$

$0 \wedge x = x$

original arr

$a \wedge b \wedge c \wedge d \wedge e \wedge f = v$

[e] freq

[a, b, c]

[d, e, f] → hash map

$a \wedge c = v$ ⇒ xoring c on both sides

hashmap

$\underbrace{a \wedge \underbrace{c \wedge c}_{0}} = v \wedge c$ $O(n^2)$

eve

$\frac{n-1}{2}$ pair

0

last pair → v

$\boxed{a = v \wedge c}$

$\boxed{e = v \wedge a}$

searchy problem ←

→ binary search $a \wedge v$ $(n \log n$

$a \wedge v$ → false

Q: You have an unsorted array of first n natural no's, without repetition. You are given a positive integer k. You can do a swap operation on the array at max k number of times. Determine the largest lexicographical value array that can be created from the given array with no more than k swaps.

$[4, 2, 3, 5, 1] \longrightarrow \boxed{[5, 2, 3, 4, 1]}$

$k = 1$

$\boxed{a[i] \leq 10^8}$

$\boxed{\begin{array}{l} n \leq 10^7 \\ k \leq 10^9 \end{array}}$

for a n-size array → range of elements [1-n]

↳ get the largest lexicographical array → our agenda is to move the array towards a reverse sorted array

for an array to be moving towards reverse sorted state, the largest element should be at 0th index then the 2nd largest is at 1st index and so on find n natural no.

no repetition

greedy

n → 0th index

(n-1) → 1st index

(n-2) → 2nd index

mp.(5)

0th

4

[4, 2, 3, 5, 1]

K = 1

→ unordered_map <value, index>

4 — 0
2 — 1
3 — 2
5 — 3
1 → 4

n → 1

→ 5

for (i → n)

< n

any n natural no.

n → 1

i → check if i is a key and a no.?

O (n × check)

Q→ Given a value n, give the count of binary strings with no consecutive ones. of size n.

$\boxed{n \le 10^8}$

$\boxed{n = 2} \longrightarrow$

00
01
10

ans → $\boxed{3}$

$n = 3 \longrightarrow$

000
001
010
100
101

ans → 5

0, 1, 1, 2, 3, 5, 8, 13

n = 1 → 0
1

↳ 2

n = 2

↳ 3

n = 3

↳ 5

n = 4 → 0000
0001
0010
0100
1000
0101
1010
1001

↳ 8

2, 3, 5, 8, 13

n = 5

↳ 13

fibonacci

n^{th} fib

1st → 2

2nd → 3

$f(n) = f(n-1) + f(n-2)$ → Recursion

↳ $O(2^n)$

$\boxed{DP \rightarrow O(n)}$

→ Binet's Formula

→ Matrix Exponentiation → $O(\log n)$