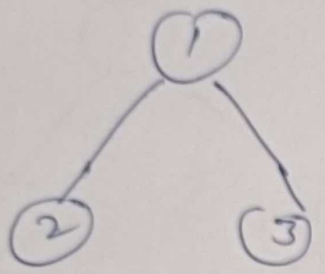
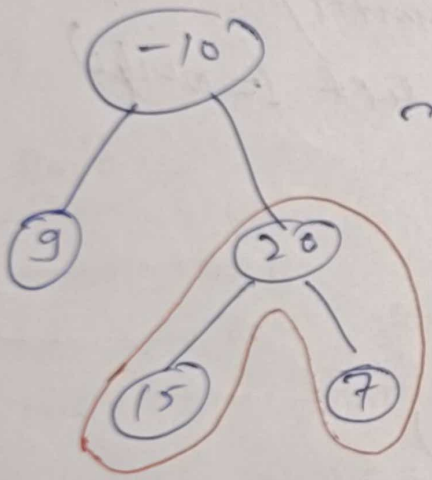
124. Binary Tree Maximum Path Sum



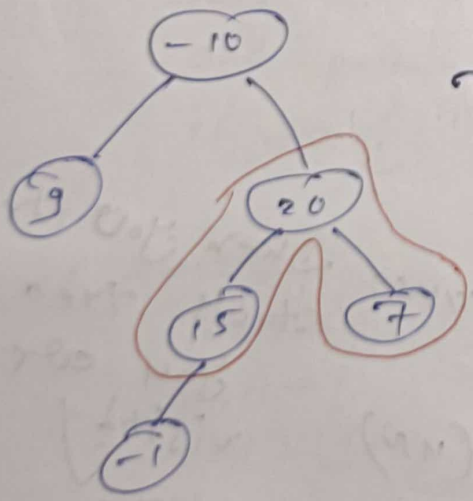
maximum path sum = 2 → 1 → 3

2 + 1 + 3 = 6



maximum path sum ⇒ 15 → 20 → 7

= 42



maximum path sum ⇒ 15 → 20 → 7

= 42

Approach - 1 Brute force approach.

Go for every possible path & whichever path is giving me maximum path sum the pick up that path.

Now if we have N node

nodes $\longrightarrow$ N

total paths possible for every node $\longrightarrow N^2$

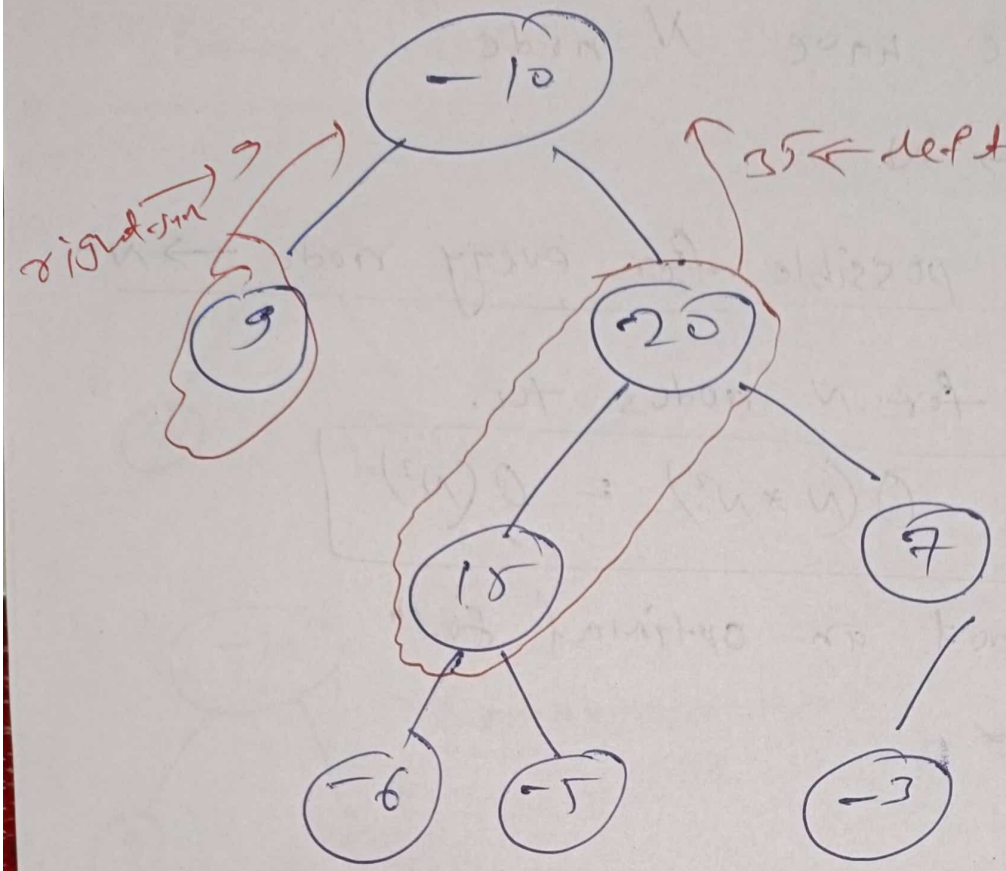
that means for N nodes tc.

$$T.C. = O(N * N^2) = O(N^3)$$

This is not an optimal soln

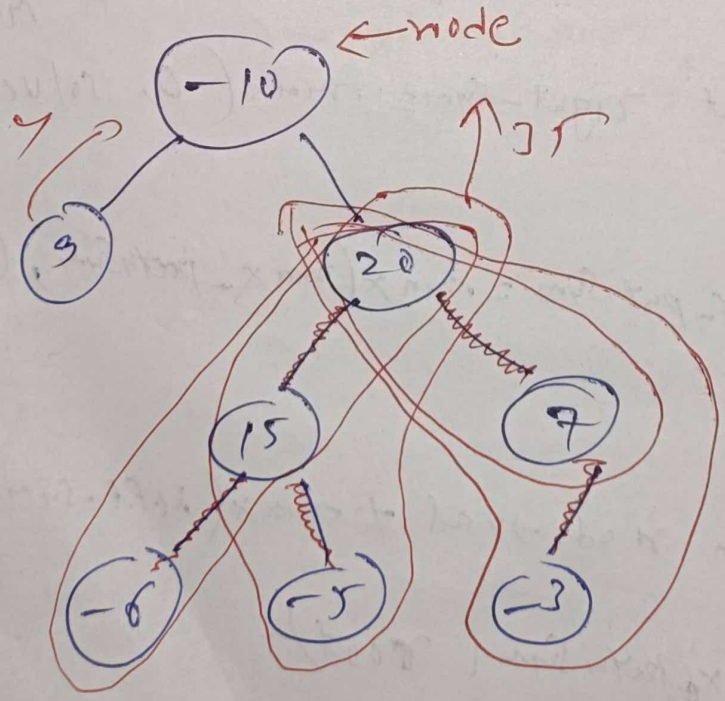
Approch - 2:

```
int solve (node, max_pathSum)
{   if (node == NULL)
        return 0;

    int left_sum = max(0, solve(node->left,
                   max_pathSum));
    int right_sum = max(0, solve(node->right,
                   max_pathSum));

    max_pathSum = max(max_pathSum, (node->val
                  + left_sum +
                  right_sum));

    return node->val + max(left_sum, right_sum);
}
int maxpathSum (root)
{
    ——
    ——
    ——
}
```

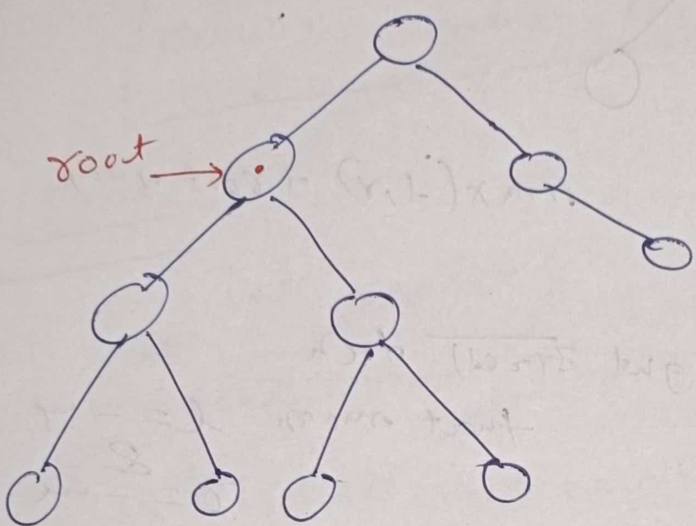


left-sum = node के left sub tree में possible paths में से max path sum store कर देते है।

right-sum = same as left sum



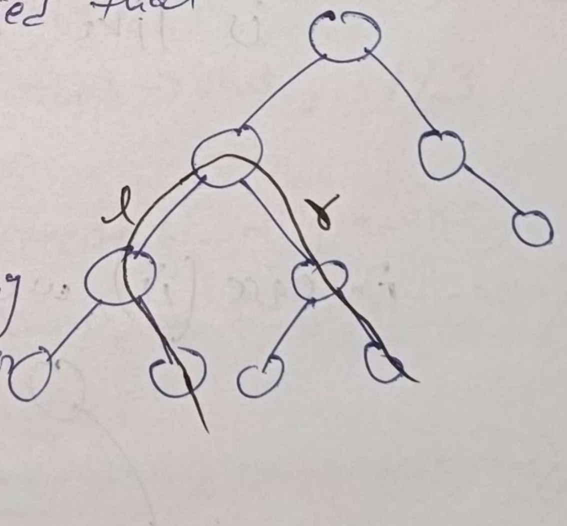
⇒



write down all possibilites that which path gives yous best answer.

(i) left side, right side से अच्छा sum आया है।

That means you are satisfied that this is your pat sum.

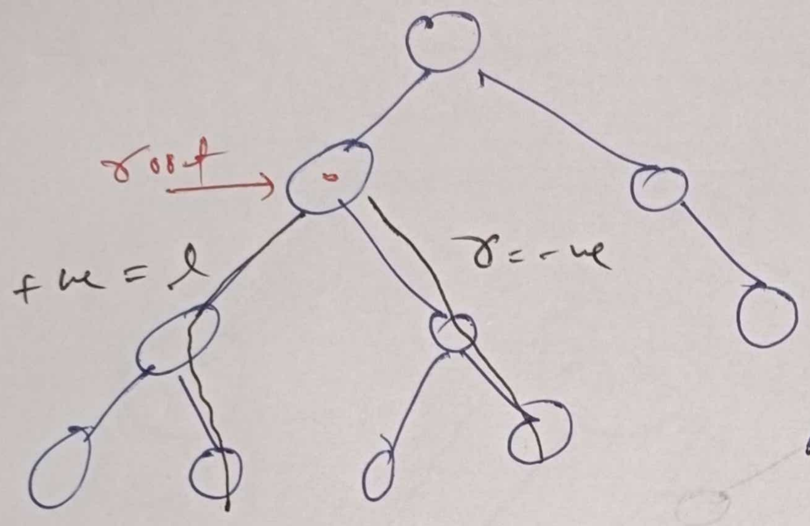


l + r + root → val

↳ By this we are deciding that नीचे से ही actual answer है

(ii) left या right में जो अच्छा sum हो उसे लेंगे तब means if get -ve in left or either right and we add this in node → val this will decrease our sum

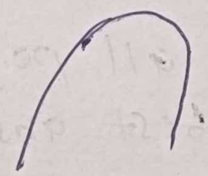
max(l, r) + root → val



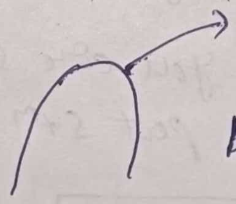
max(l, r) + root → val

(iii) ना left, ना right अच्छा then
that mean l = -ve & r = -ve
root → val

(iv) in (i) case path is like .

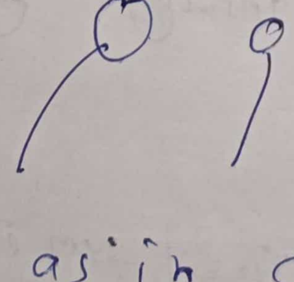


and we are not returning this because if we return this that mean the path is like that



but this is not be true

in case (ii) we can return max(l, r) + root → val
because we are hopping that upper जाके कोई अच्छा ans मिल सकता है।



same as in case (iii).

maxSum = max(maxSum, 1, 2, 3)

return max(2, 3);

Code

```
int maxSum;
int solve(TreeNode* root)
{
    if(root == NULL)
        return 0;

    int l = solve(root->left);
    int r = solve(root->right);

    int neeche_hi_milgya_ans = l+r+root->val   //1
    int koi_ek_acha = max(l,r) + root->val;    //2
    int only_root_acha = root->val;            //3

    maxSum = max({maxSum, neeche_hi_milgya_ans,
                  koi_ek_acha, only_root_acha});

    return max(koi_ek_acha, only_root_ans);
}
int maxPathSum(TreeNode* root);
{
    maxSum = INT_MIN;
    solve(root);
    return maxSum;
}
```