LAB -04

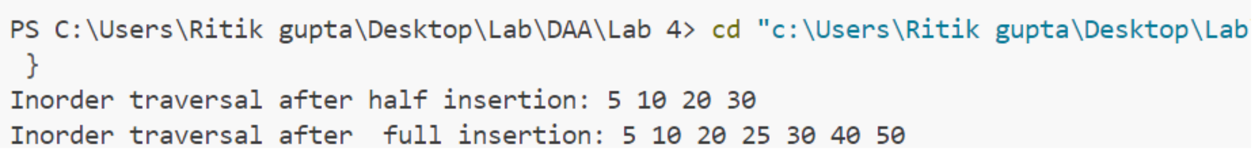
Task -01:

Q1-Implement AVL tree-

```
#include <iostream>
using namespace std;
class Node{
    public:
    int data;
    int height;
    Node* left;
    Node* right;
    Node(int d){
        data = d;
        left = NULL;
        right = NULL;
        height = 1;
    }
};
int getheight(Node* root){
    if(root==NULL)return 0;
    return root->height;
}
int balancefactor(Node* root){
    if(root==NULL)return 0;
    getheight(root->right) -getheight(root->left);
}
void updateHeight(Node* node){
    node->height = 1 + max(getheight(node->left) , getheight(node->right));
}
Node* rightRotate(Node* y) {
    Node* x = y->left;
    Node* T2 = x->right;
    x->right = y;
    y->left = T2;
    updateHeight(y);
    updateHeight(x);
    return x;
}
Node* leftRotate(Node* x) {
    Node* y = x->right;
    Node* T2 = y->left;
    y->left = x;
    x->right = T2;
    updateHeight(x);
    updateHeight(y);
    return y;
}
Node* insert(Node* root, int data) {
    if (!root) return new Node(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else if (data > root->data)
        root->right = insert(root->right, data);
    else
        return root;
    updateHeight(root);
    int balance = balancefactor(root);
    if (balance > 1 && data < root->left->data)
```

```
        return rightRotate(root);
    if (balance < -1 && data > root->right->data)
        return leftRotate(root);
    if (balance > 1 && data > root->left->data) {
        root->left = leftRotate(root->left);
        return rightRotate(root);
    }
    if (balance < -1 && data < root->right->data) {
        root->right = rightRotate(root->right);
        return leftRotate(root);
    }
    return root;
}
void inorder(Node* root) {
    if (root) {
        inorder(root->left);
        cout << root->data<< " ";
        inorder(root->right);
    }
}
int main(){
    Node* root = new Node(5);
    root = insert(root, 10);
    root = insert(root, 20);
    root = insert(root, 30);
    cout << "Inorder traversal after half insertion: ";
    inorder(root);
    cout << endl;
    root = insert(root, 40);
    root = insert(root, 50);
    root = insert(root, 25);
    cout << "Inorder traversal after  full insertion: ";
    inorder(root);
    cout << endl;
}
```

Output:



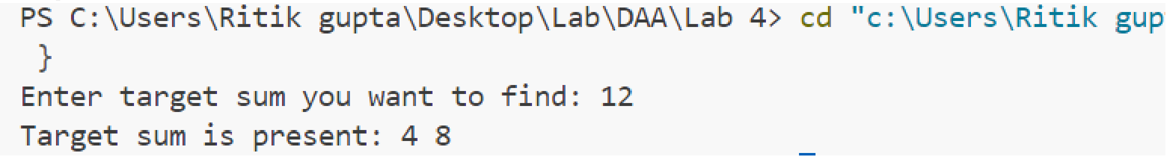
```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 4> cd "c:\Users\Ritik gupta\Desktop\Lab
 }
Inorder traversal after half insertion: 5 10 20 30
Inorder traversal after  full insertion: 5 10 20 25 30 40 50
```

Task -02:

Q1-Target Pair Sum in array-

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int size = 6;
    vector<int>arr = {2,6,4,8,9,4};
    int target;
    cout<<"Enter target sum you want to find: ";
    cin>>target;
    sort(arr.begin() , arr.end());
    int s=0;
    int e=size-1;
    bool found = false;
    while(s<=e){
        if(arr[s] + arr[e] == target){
            found = true;
            break;
        }
        else if(arr[s] + arr[e] > target)e--;
        else s++;
    }
    if(found){
        cout<<"Target sum is present: ";
        cout<<arr[s]<<" "<<arr[e];
    }
    else{
        cout<<"Target sum is not present";
    }
    return 0;
}
```

Output:



```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 4> cd "c:\Users\Ritik gup
 }
Enter target sum you want to find: 12
Target sum is present: 4 8
```

Q2-Digital Root

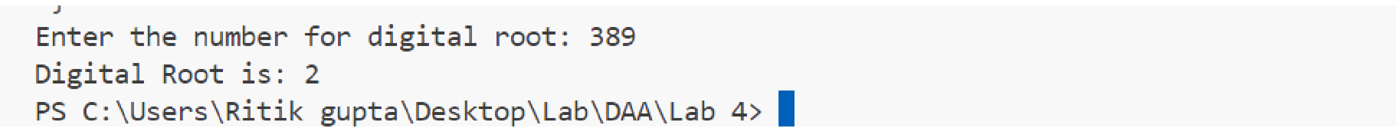
```
#include <bits/stdc++.h>
using namespace std;

    int sumDigit(int n){
        if(n==0)return 0;

        return n%10 + sumDigit(n/10);
    }
    int findRoot(int n){
        if(n<10)return n;
        return findRoot(sumDigit(n));
    }

int main(){
    int n;
    cout<<"Enter the number for digital root: ";
    cin>>n;
    int ans = findRoot(n);
    cout<<"Digital Root is: "<<ans;
}
```

Output:

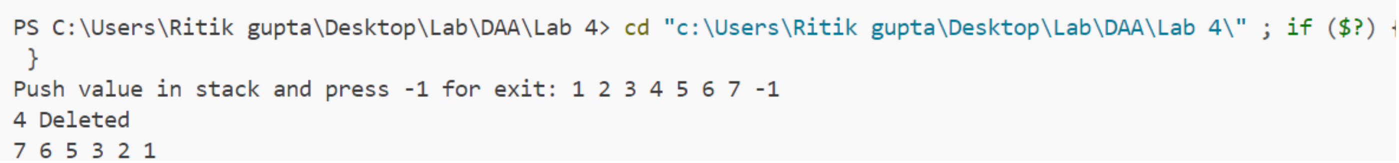


```
Enter the number for digital root: 389
Digital Root is: 2
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 4>
```

Q3. Delete Middle Element of Stack:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    stack<int>st;
    stack<int> temp;
    int x;
    cout<<"Push value in stack and press -1 for exit: ";
    while(true){
        cin>>x;
        if(x==-1)break;
        st.push(x);
    }
    int size = st.size();
    for(int i =0 ;i<(size)/2;i++){
        temp.push(st.top());
        st.pop();
    }
    cout<<st.top()<<" Deleted"<<endl;
    st.pop();
    while(!temp.empty()){
        st.push(temp.top());
        temp.pop();
    }
    while(!st.empty()){
        cout<<st.top()<<" ";
        st.pop();
    }}
```

Output:

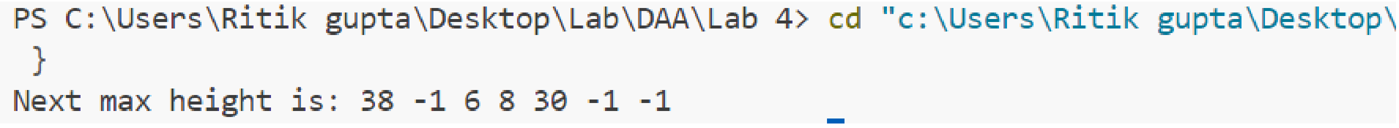


```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 4> cd "c:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 4\" ; if ($?)
 }
Push value in stack and press -1 for exit: 1 2 3 4 5 6 7 -1
4 Deleted
7 6 5 3 2 1
```

Q4.Next Max Height:

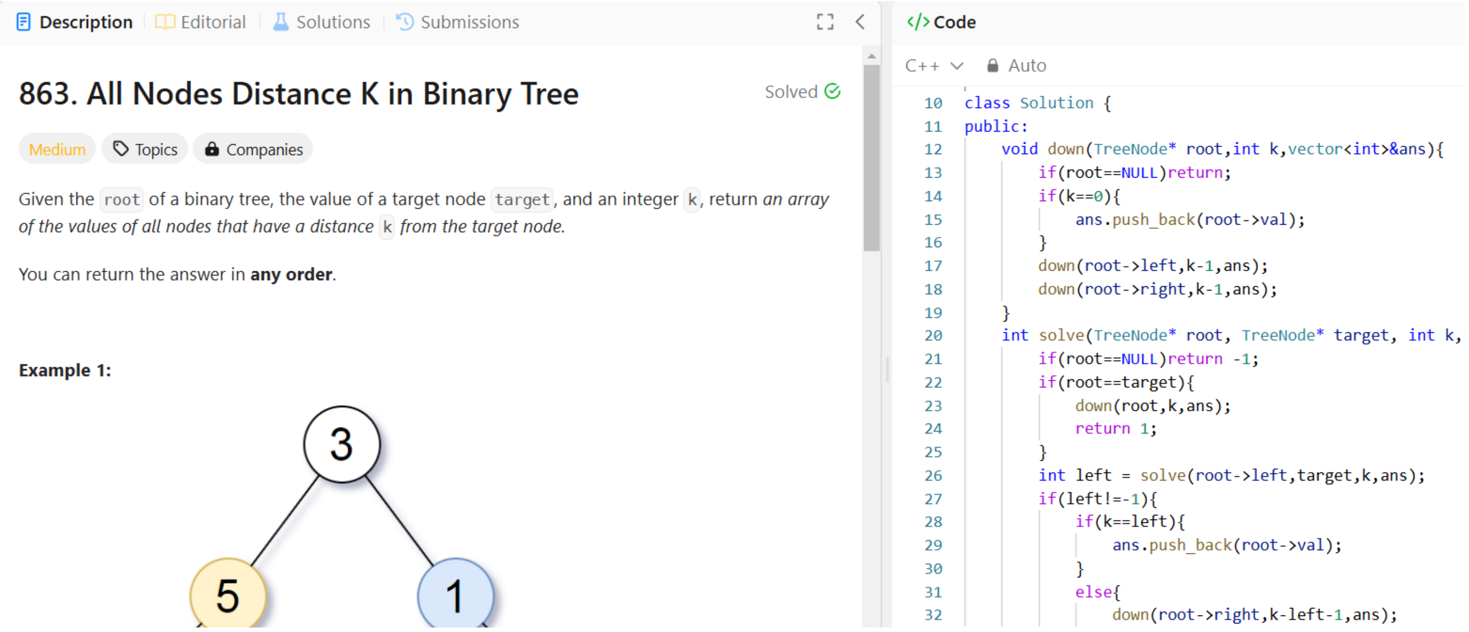
```
#include <bits/stdc++.h>
using namespace std;
    void next_height(vector<int>arr,vector<int>&ans){
        stack<int>st;
        for(int i =arr.size()-1; i>=0; i--){
            while(!st.empty() && st.top() <= arr[i]){
                st.pop();
            }
            if(st.empty())ans[i] = -1;
            else ans[i]=st.top();
            st.push(arr[i]);
        }
    }
int main(){
    vector<int> arr = {3,38,2,6,8,30,12};
    vector<int> ans(arr.size());
    next_height(arr,ans);
    cout<<"Next max height is: ";
    for(int i :ans){
        cout<<i<<" ";
    }
}
```

Output-



```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 4> cd "c:\Users\Ritik gupta\Desktop\
 }
Next max height is: 38 -1 6 8 30 -1 -1
```

Q1. Print kth Distant Node



Description | Editorial | Solutions | Submissions

863. All Nodes Distance K in Binary Tree

Solved ✓

Medium | Topics | Companies

Given the `root` of a binary tree, the value of a target node `target`, and an integer `k`, return *an array of the values of all nodes that have a distance* `k` *from the target node.*

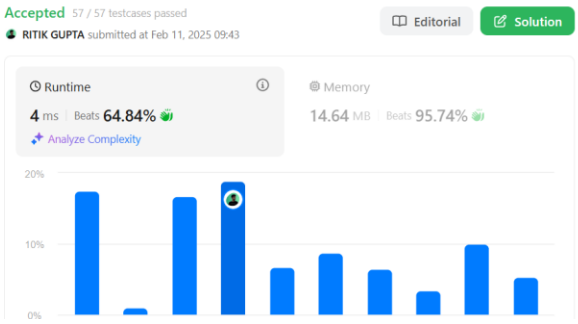
You can return the answer in **any order**.

Example 1:

Code

C++ | Auto

```
class Solution {
public:
    void down(TreeNode* root,int k,vector<int>&ans){
        if(root==NULL)return;
        if(k==0){
            ans.push_back(root->val);
        }
        down(root->left,k-1,ans);
        down(root->right,k-1,ans);
    }
    int solve(TreeNode* root, TreeNode* target, int k,
        if(root==NULL)return -1;
        if(root==target){
            down(root,k,ans);
            return 1;
        }
        int left = solve(root->left,target,k,ans);
        if(left!=-1){
            if(k==left){
                ans.push_back(root->val);
            }
            else{
                down(root->right,k-left-1,ans);
```



Accepted 57 / 57 testcases passed

RITIK GUPTA submitted at Feb 11, 2025 09:43

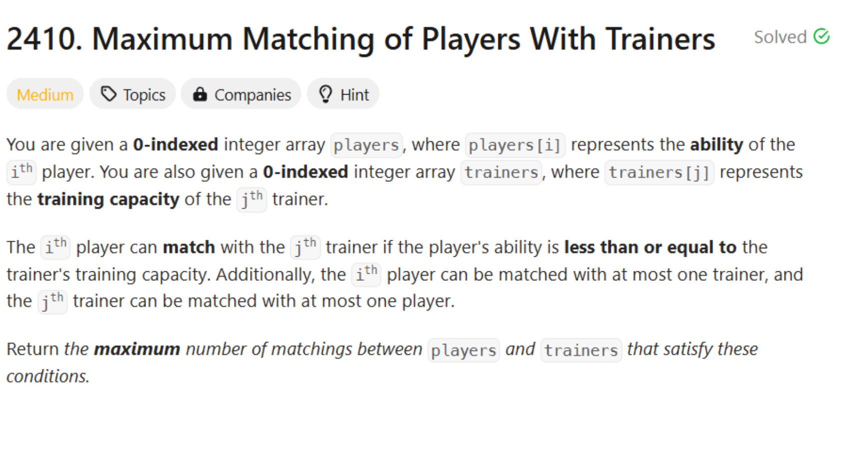
Editorial | Solution

Runtime: 4 ms | Beats 64.84%

Analyze Complexity

Memory: 14.64 MB | Beats 95.74%

Q2.Maximum Matching of players



2410. Maximum Matching of Players With Trainers

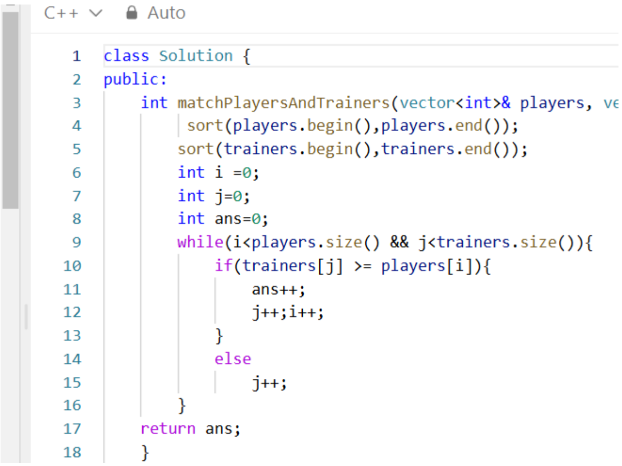
Solved ✓

Medium | Topics | Companies | Hint

You are given a **0-indexed** integer array `players`, where `players[i]` represents the **ability** of the i^{th} player. You are also given a **0-indexed** integer array `trainers`, where `trainers[j]` represents the **training capacity** of the j^{th} trainer.

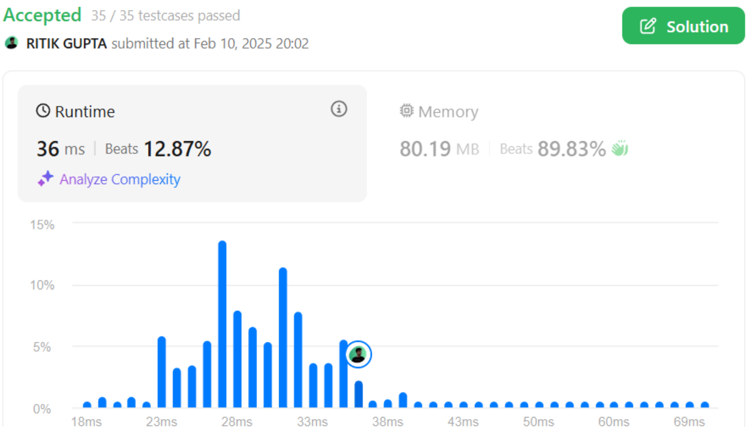
The i^{th} player can **match** with the j^{th} trainer if the player's ability is **less than or equal to** the trainer's training capacity. Additionally, the i^{th} player can be matched with at most one trainer, and the j^{th} trainer can be matched with at most one player.

Return *the* ***maximum*** *number of matchings between* `players` *and* `trainers` *that satisfy these conditions.*



C++ | Auto

```
class Solution {
public:
    int matchPlayersAndTrainers(vector<int>& players, ve
        sort(players.begin(),players.end());
        sort(trainers.begin(),trainers.end());
        int i =0;
        int j=0;
        int ans=0;
        while(i<players.size() && j<trainers.size()){
            if(trainers[j] >= players[i]){
                ans++;
                j++;i++;
            }
            else
                j++;
        }
    return ans;
    }
```



Accepted 35 / 35 testcases passed

RITIK GUPTA submitted at Feb 10, 2025 20:02

Solution

Runtime: 36 ms | Beats 12.87%

Analyze Complexity

Memory: 80.19 MB | Beats 89.83%