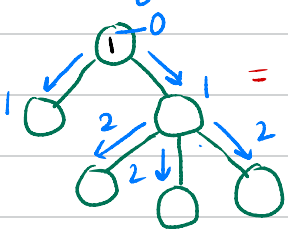
September 27, 2025

TREE DATA STRUCTURE IDEAS

Ques 1. For a given tree, Find $\sum_{i=1}^{N} Dist(1, i)$

From node 1, have to find sum of distance to every node.

Ex:



$= 2+2+2+1+1+0$

$= 8$ (Ans)

```
int n;
vector<vector<int>> g;
// ancestor
vector<int> depth,subsz;


void dfs(int nn,int pp,int dd){
    depth[nn] = dd;
    subsz[nn] = 1;
    for(auto v:g[nn]){
        if(v!=pp){
            dfs(v,nn,dd+1);
            subsz[nn] += subsz[v];
        }
    }
}
```

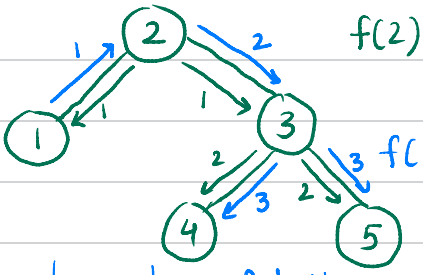
```
void solve(){
    cin>>n;
    g.resize(n+1);
    depth.resize(n+1);
    subsz.resize(n+1);

    for(int i=0;i<n-1;i++){
        int a,b;
        cin>>a>>b;
        g[a].push_back(b);
        g[b].push_back(a);
    }

    dfs(1,0,0);
    ans = 0;
    for(int i=1;i<=n;i++){
        ans += depth[i];
    }
    cout<<ans<<endl;

}
```

Ques 2. Calc value of $f(x)$

$F(x) = \sum_{i=1}^{n} Dist(x, i)$ for all $1 \le x \le N$. $(1 \le N \le 10^5)$

Need O(N) soln

Ex:



$f(2) = 1+1+2+2 = 6$

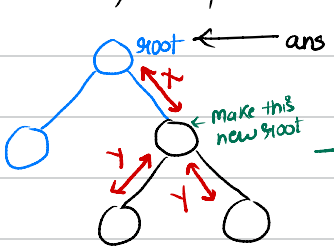
$f(1) = 1+2+3+3 = 9$

You have to print N numbers, not add.

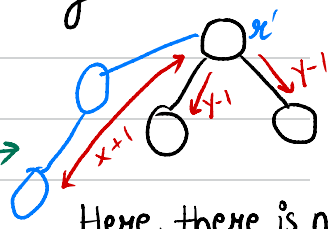
Can we calculate $f(2)$ in O(N)?
YES (done in prev ques w $f(1)$)

- Use of Pre-rerooting

Given a root and the answer, we find the answer of a new root, r' using it.



Here, there is a cost to reach r' which trickles down the subtree



Here, there is no cost to go to r'. Instead, there is additional edge contribution

Outside r' subtree nodes distance $= x+1$. Inside nodes $= y-1$.

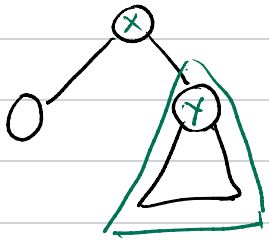
AlgoZenith

All the nodes outside has to increase by 1. All the nodes that are inside have to decrease by 1.

So, if we had root → Ans

Then, y' → (Ans - DecreementSubtree + IncrementSubtree)

→ How to calcuate DecreementSubtree

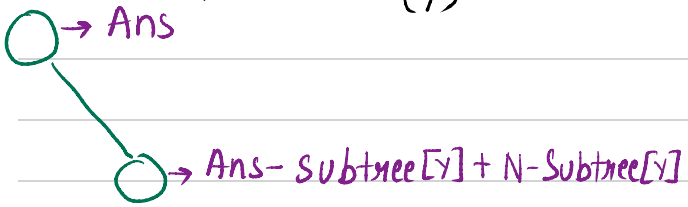


when we move from x → y, only the nodes inside the y subtree will decrease. So, we already know the size of the subtree (no. of nodes in subtree), subsize.

→ IncrementSubtree

Everything except subtree.

⇒ N - Subtree (y)




```cpp
int n;
vector<vector<int>> g;
// ancestor
vector<int> depth,subsz;


void dfs(int nn,int pp,int dd){
    depth[nn] = dd;
    subsz[nn] = 1;
    for(auto v:g[nn]){
        if(v!=pp){
            dfs(v,nn,dd+1);
            subsz[nn] += subsz[v];
        }
    }
}

vector<int> finalans;
void reroot(int nn,int pp,int ans){
    finalans[nn] = ans;
    for(auto v:g[nn]){
        if(v!=pp){
            reroot(v,nn,ans - subsz[v] + n-subsz[v]);
        }
    }
}

void solve(){
    cin>>n;
    g.resize(n+1);
    depth.resize(n+1);
    subsz.resize(n+1);

    for(int i=0;i<n-1;i++){
        int a,b;
        cin>>a>>b;
        g[a].push_back(b);
        g[b].push_back(a);
    }

    dfs(1,0,0);
    ans = 0;
    for(int i=1;i<=n;i++){
        ans += depth[i];
    }
    cout<<ans<<endl;

}
```

Homework : https://cses.fi/alon/task/1133

You are given a tree consisting of n nodes.

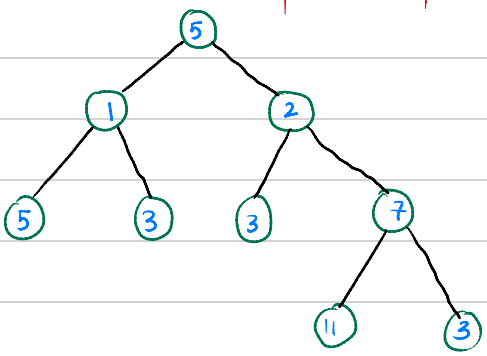
Your task is to determine for each node the sum of the distances from the node to all other nodes.

AlgoZenith

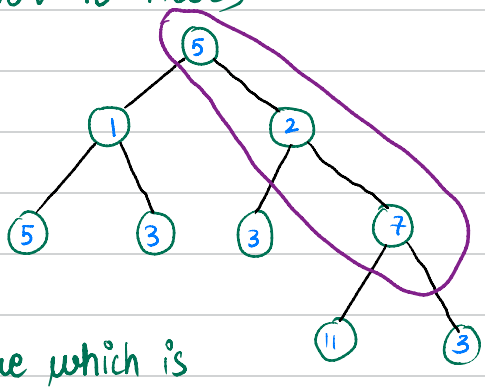
Ques 3 Given a tree, Find for each node:

a. farthest value on path to root.

b. closest value on path to root

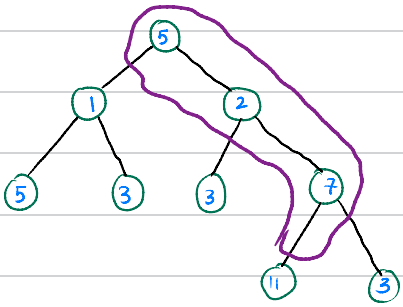


Example, for rightmost node with value 3 (on the path from root to node)



Value which is farthest from the current value (3) is 7 (from 5, 2, 7)

- For node with value = 11



{5, 2, 7}

Ans = 2

↑ farthest val

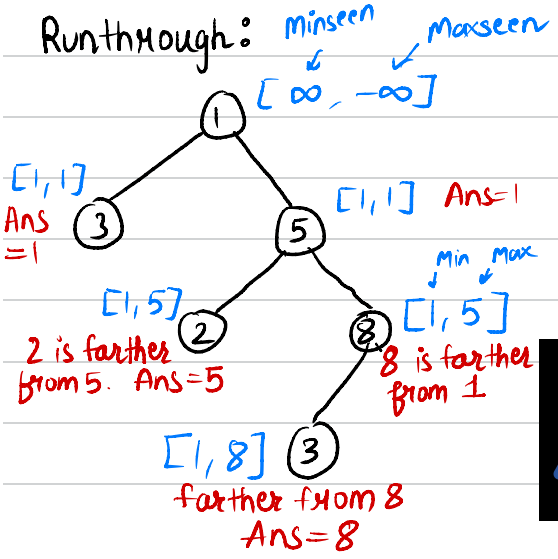
→ For closest (still on the path → root)

- Val = 11, Ans = 7
- Val = 3 (rightmost), Ans = 2.

// Solution a. (farthest)

```
int n;
int arr[100100];

vector<vector<int>> g;

vector<int> farthest;



void dfs(int nn,int pp,int maxseen,int minseen){

    if(abs(maxseen-arr[nn]) < abs(minseen-arr[i]))
        farthest[nn] = minseen;
    else farthest[nn] = maxseen;

    minseen = min(minseen,arr[nn]);
    maxseen = max(maxseen,arr[nn]);
    for(auto v:g[nn]){
        if(v!=pp){
            dfs(v, nn, maxseen, minseen);
        }
    }
}
void solve(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>arr[i];
    }
    g.resize(n+1);
    farthest.resize(n+1);

    for(int i=0;i<n-1;i++){
        int a,b;
        cin>>a>>b;
        g[a].push_back(b);
        g[b].push_back(a);
    }

    dfs(1,0,0,-1e9,1e9);

}
```

Run through:



// Solution b (Closest)

DFS (Let us maintain a DS such that)
- enters → insert()
- child
- exit → remove()

```
vector<int> farthest;
vector<int> closest;


multiset<int> mt;

void dfs(int nn,int pp,int maxseen,int minseen){

    if(abs(maxseen-arr[nn]) < abs(minseen-arr[i]))
        farthest[nn] = minseen;
    else farthest[nn] = maxseen;

    // closest
    int x = INF, y=INF;
    auto it = mt.lower_bound(arr[i]);
    if(it!=mt.end()) y = *it;
    if(it!=mt.begin()){
        it--;
        x = *it;
    }
    if(abs(x-arr[nn]) < abs(y-arr[i]))closest[nn] = x;
    else closest[nn] = y;

    minseen = min(minseen,arr[nn]);
    maxseen = max(maxseen,arr[nn]);

  mt.insert(arr[nn]);
  for(auto v:g[nn]){
      if(v!=pp){
          dfs(v, nn, maxseen, minseen);
      }
  }
  mt.erase(mt.find(arr[i]));
```

* DSU on Tree Technique

For every node, find out the no. of distinct values seen in a subtree.

Small to large merging: If you want to merge bunch of childs. Figure out the biggest child which has the highest size and merge the other maps into that map & use that map directly.

↑ map

⇒ NlogN

```
int n;
int arr[100100];
vector<vector<int>> g;
int sz[100100];

// Small to large merging.
map<int,int> freq[100100];

void dfs(int nn,int pp){
    sz[nn] = 1;
    int big_ch = -1;
    for(auto v:g[nn]){
        if(v!=pp){
            dfs(v,nn);
            sz[nn] += sz[v];
            if(big_ch==-1 || sz[v]>sz[big_ch]) big_ch=v;
        }
    }
    if(big_ch==-1){
        freq[nn][arr[nn]]=1;
    }else{
        swap(freq[nn],freq[big_ch]);
        freq[nn][arr[nn]]++;

        for(auto v:g[nn]){
            if(v!=pp && v!=big_ch){
                for(auto x:freq[v]){
                    freq[nn][x.first]+=x.second;
                }
                freq[v].clear();
            }
        }
    }
    ans[nn] = freq[nn].size();
}
void solve(){
    cin>>n;
    for(int i=1;i<=n;i++)cin>>arr[i];

    g.resize(n+1);
    ways.resize(n+1);
    for(int i=0;i<n-1;i++){
        int a,b;
        cin>>a>>b;
        g[a].push_back(b);
        g[b].push_back(a);
    }
    dfs(1,0);
    for(int i=1;i<=n;i++){
        cout<<ans[i]<<endl;
    }
}
```