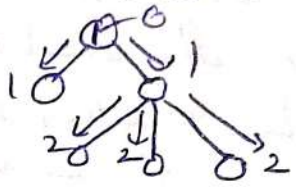
Tree structure Idea

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

Approach : Apply DFS from the root (node 1) to calculate the depth of each node.

T.C : O(N)



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

$= 8$

```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];
        }
    }
```

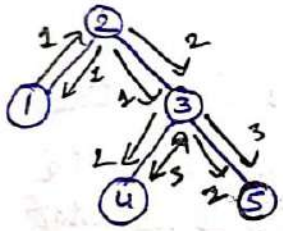
```
void solve(){
    cin>>n;
    g.resize(n+1);
    depth.resize(n+1);
    subsz.resize(n+1);

    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;

}
```

2. Calculate value of $f(x) \cdot F(x) = \sum_{i=1}^{n} Dist(x, i)$ for all $1 \le x \le N$



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

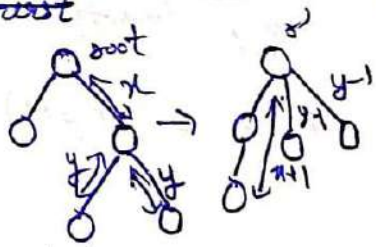
$f(1) = 1+2+3+3 = 9$

Approach : This problem can be solved using re-rooting ~~Fast~~

When moving from a parent p to its child c :

→ All nodes in the subtree of c get ~~farther from~~ closer to increased ~~the new root~~ by 1.

→ All nodes outside the subtree of c get ~~farther from~~ decreased ~~new root by 1.~~ by 1.

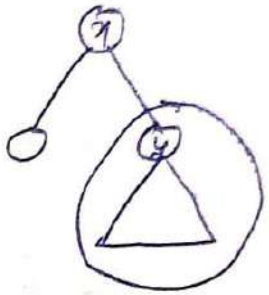


So, if we had root → Ans

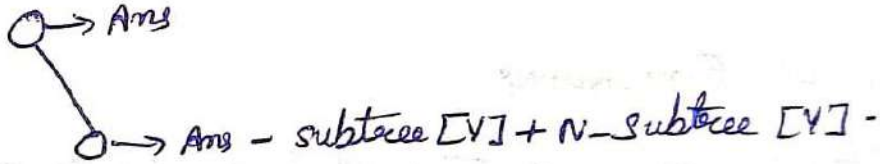
Then $x' \rightarrow$ (Ans − Decrement subtree + Increment subtree)

How to calculate decrement subtree ?

When we move from $x \rightarrow y$ only nodes inside the y subtree will decrease.



Increment subtree ⇒ N − subtree (Y)
(Everything except subtree)

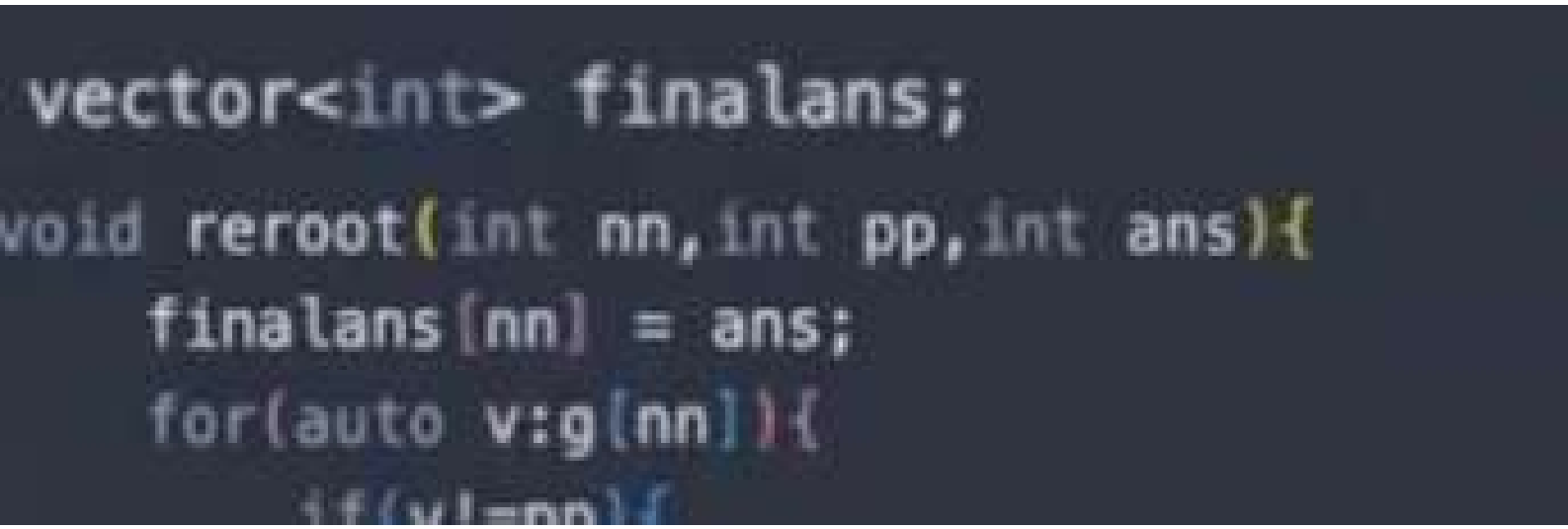


H.W CSES.fi/alon/task/1133

```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]){

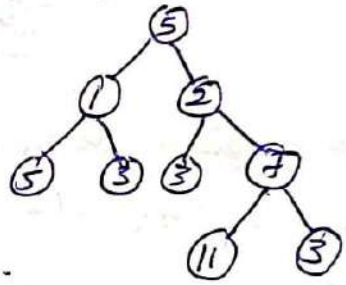
```
    finalans[nn] = ans;
    for(auto v:g[nn]){
        if(v!=pp){
            reroot(v,nn,ans - subsz[v] + n-subsz[v]);
        }
    }


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];
    }
```

3: Given a tree, find for each node:
a) farthest value on path to root (b) closest value on path to root.



For farthest value:

Approach: Apply DFS from the root. During traversal, we maintain the min & max values seen on the path from the root to current node.
∴ farthest value will be either min or max value depending on larger absolute difference with current node's value.

For closest value:-

Approach: DSU on Tree Technique

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

→ small to large Merging: If we want to merge bunch of child find biggest child (map) and merge the other maps into that map & use that map directly.

T.C: O(N log N)

```cpp
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;
    }
```