**//**[**Heavy-Light Decomposition**](http://lightoj.com/volume_problemcategory.php?category=Heavy-Light%20Decomposition)

(code by aninda\_codechef)

int arr[MX], tChain, child[MX], parent[MX], inChain[MX], chainHead[MX], level[MX], chainLen[MX], chainPos[MX];

vector<int>adj[MX], chain[MX];

void update(int child,int idx,int st,int ed,int pos,int val)

{

if(st==ed)

{

chain[child][idx]=val;

return;

}

int lft=2\*idx, rgt=lft+1, mid=(st+ed)/2;

if(pos<=mid) update(child, lft, st, mid, pos, val);

else update(child, rgt, mid+1, ed, pos, val);

chain[child][idx]=chain[child][lft]+chain[child][rgt];

return;

}

int query(int child, int idx, int st, int ed, int s, int e)

{

if(st==s && ed==e) return chain[child][idx];

int lft=2\*idx, rgt=lft+1, mid=(st+ed)/2;

if(e<=mid) return query(child, lft, st, mid, s, e);

else if(s>mid) return query(child, rgt, mid+1, ed, s, e);

else return query(child,lft,st,mid,s,mid)+query(child, rgt, mid+1, ed, mid+1, e);

}

void build\_chain(int uu)

{

int u=uu;

vector<int>v;

int pos=0;

while(u!=-1)

{

v.push\_back(u);

inChain[u]=tChain;

chainHead[u]=uu;

chainPos[u]=pos++;

u=child[u];

}

chainLen[tChain]=v.size();

chain[tChain].resize(4\*chainLen[tChain], 0);

for(int i=0;i<chainLen[tChain];i++)

update(tChain, 1, 0, chainLen[tChain]-1, i, arr[v[i]]);

tChain++;

return;

}

int dfs(int u, int par, int lvl)

{

parent[u]=par;

level[u]=lvl;

child[u]=-1;

int tmp, ret=1, mx=0;

for(int i=0;i<adj[u].size();i++)

{

int v=adj[u][i];

if(v!=par)

{

tmp=dfs(v, u, lvl+1);

ret+=tmp;

if(tmp>mx)

{

mx=tmp;

child[u]=v;

}

}

}

for(int i=0;i<adj[u].size();i++)

if(adj[u][i]!=par && adj[u][i]!=child[u])

build\_chain(adj[u][i]);

return ret;

}

int getLCA(int x,int y)

{

while(inChain[x]!=inChain[y])

{

if(level[chainHead[x]]>level[chainHead[y]]) x=parent[chainHead[x]];

else y=parent[chainHead[y]];

}

return (level[x]<level[y]? x:y);

}

int lftRgt(int x,int lca)

{

int ret=0;

while(level[x]>level[lca])

{

ret+=query(inChain[x], 1, 0, chainLen[inChain[x]]-1,(level[chainHead[x]]>level[lca]? chainPos[chainHead[x]]:chainPos[lca]+1), chainPos[x]);

x=parent[chainHead[x]];

}

return ret;

}

int main()

{

int tc,kk=1, n, u, v, q, t, x, y;

cin>>tc;

while(tc--)

{

tChain=0;

cin>>n;

for(int i=0;i<n;i++) scanf("%d", &arr[i]), adj[i].clear();

n--;

while(n--)

{

scanf("%d%d", &u, &v);

adj[u].push\_back(v);

adj[v].push\_back(u);

}

dfs(0, -1, 0);

build\_chain(0);

cin>>q;

cout<<"Case "<<kk++<<":"<<endl;

while(q--)

{

scanf("%d%d%d", &t, &x, &y);

if(t) update(inChain[x], 1, 0, chainLen[inChain[x]]-1, chainPos[x], y);

else

{

int ans=0, lca = getLCA(x, y);

ans+=lftRgt(x, lca); //left brance

ans+=lftRgt(y, lca); //right brance

ans+=query(inChain[lca], 1, 0, chainLen[inChain[lca]]-1, chainPos[lca], chainPos[lca]); //lca point

printf("%d\n", ans);

}

}

}

return 0;

}