```cpp
// Problem: Can you answer these queries VII
// Contest: SPOJ - Classical
// URL: https://www.spoj.com/problems/GSS7/en/
// Memory Limit: 1536 MB
// Time Limit: 1000 ms
//
// Powered by CP Editor (https://cpeditor.org)


// By AmmarDab3an - Aleppo University

#include "bits/stdc++.h"

using namespace std;

#define int int64_t
#define ll  int64_t

// typedef unsigned int        uint;
// typedef long long int       ll;
// typedef unsigned long long  ull;
typedef pair<int, int>    pii;
typedef pair<ll, ll>      pll;
typedef pair<int, pii>    iii;
typedef pair<ll, pll>     lll;
typedef vector<int>       vi;
typedef vector<ll>        vl;
typedef vector<pii>       vpii;
typedef vector<pll>       vpll;

#define endl '\n'
#define fastIO ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define freopenI freopen("input.txt", "r", stdin);
#define freopenO freopen("output.txt", "w", stdout);

const int INF = 0x3f3f3f3f;
const ll INFLL = 0x3f3f3f3f3f3f3f3f;
const int MOD = 1e9 + 7;
const double EPS = 1e-9;
const double  PI = acos(-1);

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

int rand(int x, int y) {
    return uniform_int_distribution<int>(x, y)(rng);
}

int mul(int a, int b){
    return (1ll * (a%MOD) * (b%MOD)) % MOD;
}

int add(int a, int b){
    return (1ll * (a%MOD) + (b%MOD) + MOD + MOD) % MOD;
}

int pow_exp(int n, int p){
    if(!p) return 1;
    if(p&1) return mul(n, pow_exp(n, p-1));
    int tmp = pow_exp(n, p/2);
    return mul(tmp, tmp);
}

const int NMAX = 1e5 + 10;
const int LOG_MAX = ceil(log2(double(NMAX)));

int n, log_n;
int arr[NMAX];
vi adj[NMAX];
int sz[NMAX], depth[NMAX], par[NMAX][LOG_MAX];
```

```
int in[NMAX], rin[NMAX], out[NMAX], nxt[NMAX], t;

struct node{
    int sm, pre, suf, ans;
};

node tree[NMAX<<2];
int lazy[NMAX<<2];
bool lazy_vis[NMAX<<2];

node merge(node a, node b){
    node ret;
    ret.sm = a.sm + b.sm;
    ret.pre = max(a.pre, a.sm+b.pre);
    ret.suf = max(b.suf, a.suf+b.sm);
    ret.ans = max({a.ans, b.ans, a.suf+b.pre});
    return ret;
}

void push(int nd, int l, int r){

    if(!lazy_vis[nd]) return;

    node &t = tree[nd];
    t.sm = lazy[nd] * (r-l+1);
    t.pre = t.suf = t.ans = max(t.sm, int(0));

    if(l != r){
        lazy[nd*2] = lazy[nd*2+1] = lazy[nd];
        lazy_vis[nd*2] = lazy_vis[nd*2+1] = true;
    }

    lazy_vis[nd] = false;
}

void build(int nd, int l, int r){

    if(l == r){
        node &t = tree[nd];
        t.sm = arr[rin[l]];
        t.pre = t.suf = t.ans = max(t.sm, int(0));
        return;
    }

    int mid = (l+r)/2;
    build(nd*2, l, mid);
    build(nd*2+1, mid+1, r);

    tree[nd] = merge(tree[nd*2], tree[nd*2+1]);
}

void update(int nd, int l, int r, int q_l, int q_r, int val){

    push(nd, l, r);

    if(r < q_l || q_r < l){
        return;
    }

    if(q_l <= l && r <= q_r){
        lazy[nd] = val;
        lazy_vis[nd] = true;
        push(nd, l, r);
        return;
    }

    int mid = (l+r)/2;
    update(nd*2, l, mid, q_l, q_r, val);
    update(nd*2+1, mid+1, r, q_l, q_r, val);
```

```
    tree[nd] = merge(tree[nd*2], tree[nd*2+1]);
}

node query(int nd, int l, int r, int q_l, int q_r){

    push(nd, l, r);

    if(r < q_l || q_r < l){
        return (node){0, 0, 0, 0};
    }

    if(q_l <= l && r <= q_r){
        return tree[nd];
    }

    int mid = (l+r)/2;
    node stPath = query(nd*2, l, mid, q_l, q_r);
    node ndPath = query(nd*2+1, mid+1, r, q_l, q_r);

    return merge(stPath, ndPath);
}

void dfs(int u, int p){

    sz[u] = 1;

    for(auto &v : adj[u]) if(v != p){

        depth[v] = depth[u] + 1;

        par[v][0] = u;
        for(int i = 1; i < log_n; i++){
            par[v][i] = par[par[v][i-1]][i-1];
        }

        dfs(v, u);
        sz[u] += sz[v];

        if((sz[v] > sz[adj[u][0]]) || (adj[u][0] == p)){
            swap(adj[u][0], v);
        }
    }
}

void hld(int u, int p){

    in[u] = t;
    rin[t] = u;
    t++;

    for(auto v : adj[u]) if(v != p){
        nxt[v] = (v == adj[u][0]) ? nxt[u] : v;
        hld(v, u);
    }

    out[u] = t;
}

int lca(int u, int v){
    if(depth[u] < depth[v]) swap(u, v);
    int dif = depth[u] - depth[v];
    for(int i = 0; i < log_n; i++) if(dif&(1<<i)) u = par[u][i];
    if(v==u) return u;
    for(int i = log_n-1; i >= 0; i--) if(par[u][i] != par[v][i]) u = par[u][i], v =
    par[v][i];
    return par[u][0];
}

```

```
node query_up(int u, int p){

    node ans = (node){0, 0, 0, 0};

    while(true){

        if(nxt[u] == nxt[p]){
            if(u==p) break;
            node que = query(1, 0, n-1, in[p]+1, in[u]);
            ans = merge(que, ans);
            break;
        }

        node que = query(1, 0, n-1, in[nxt[u]], in[u]);
        ans = merge(que, ans);
        u = par[nxt[u]][0];
    }

    return ans;
}

void update_up(int u, int p, int val){

    while(true){

        if(nxt[u] == nxt[p]){
            update(1, 0, n-1, in[p], in[u], val);
            break;
        }

        update(1, 0, n-1, in[nxt[u]], in[u], val);
        u = par[nxt[u]][0];
    }
}

int32_t main(){

    fastIO;

#ifdef LOCAL
    freopenI;
    freopenO;
#endif

    // freopen("name.in", "r", stdin);

    cin >> n;
    log_n = ceil(log2(double(n)));

    for(int i = 0; i < n; i++) cin >> arr[i];

    for(int i = 0; i < n-1; i++){

        int u, v;
        cin >> u >> v;
        u--, v--;

        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    dfs(0, -1);
    hld(0, -1);
    build(1, 0, n-1);

    int m; cin >> m; while(m--){

        int q;
        cin >> q;
```

```
276
277         if(q == 1){
278
279             int u, v;
280             cin >> u >> v;
281             u--, v--;
282
283             int p = lca(u, v);
284
285             node stPath = query_up(u, p);
286             node ndPath = query_up(v, p);
287
288             swap(ndPath.pre, ndPath.suf);
289
290             node rdPath;
291             rdPath.sm = query(1, 0, n-1, in[p], in[p]).sm;
292             rdPath.pre = rdPath.suf = rdPath.ans = max(rdPath.sm, int(0));
293
294             cout << merge(merge(ndPath, rdPath), stPath).ans << endl;
295         }
296         else{
297
298             int u, v, c;
299             cin >> u >> v >> c;
300             u--, v--;
301
302             int p = lca(u, v);
303
304             update_up(u, p, c);
305             update_up(v, p, c);
306         }
307     }
308 }
309
```