

```

1 // Problem: Can you answer these queries VII
2 // Contest: SPOJ - Classical
3 // URL: https://www.spoj.com/problems/GSS7/en/
4 // Memory Limit: 1536 MB
5 // Time Limit: 1000 ms
6 //
7 // Powered by CP Editor (https://cpeditor.org)
8
9
10 // By AmmarDab3an - Aleppo University
11
12 #include "bits/stdc++.h"
13
14 using namespace std;
15
16 #define int int64_t
17 #define ll int64_t
18
19 // typedef unsigned int      uint;
20 // typedef long long int      ll;
21 // typedef unsigned long long ull;
22 typedef pair<int, int>      pii;
23 typedef pair<ll, ll>      pll;
24 typedef pair<int, pii>      iii;
25 typedef pair<ll, pll>      lll;
26 typedef vector<int>      vi;
27 typedef vector<ll>      vl;
28 typedef vector<pii>      vpii;
29 typedef vector<pll>      vpll;
30
31 #define endl '\n'
32 #define fastIO ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
33 #define freopenI freopen("input.txt", "r", stdin);
34 #define freopenO freopen("output.txt", "w", stdout);
35
36 const int INF = 0x3f3f3f3f;
37 const ll INFL = 0x3f3f3f3f3f3f3f3f;
38 const int MOD = 1e9 + 7;
39 const double EPS = 1e-9;
40 const double PI = acos(-1);
41
42 mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
43
44 int rand(int x, int y) {
45     return uniform_int_distribution<int>(x, y)(rng);
46 }
47
48 int mul(int a, int b){
49     return (1ll * (a%MOD) * (b%MOD)) % MOD;
50 }
51
52 int add(int a, int b){
53     return (1ll * (a%MOD) + (b%MOD) + MOD + MOD) % MOD;
54 }
55
56 int pow_exp(int n, int p){
57     if(!p) return 1;
58     if(p&1) return mul(n, pow_exp(n, p-1));
59     int tmp = pow_exp(n, p/2);
60     return mul(tmp, tmp);
61 }
62
63 const int NMAX = 1e5 + 10;
64 const int LOG_MAX = ceil(log2(double(NMAX)));
65
66 int n, log_n;
67 int arr[NMAX];
68 vi adj[NMAX];
69 int sz[NMAX], depth[NMAX], par[NMAX][LOG_MAX];

```

```

70     int in[NMAX], rin[NMAX], out[NMAX], nxt[NMAX], t;
71
72     struct node{
73         int sm, pre, suf, ans;
74     };
75
76     node tree[NMAX<<2];
77     int lazy[NMAX<<2];
78     bool lazy_vis[NMAX<<2];
79
80     node merge(node a, node b){
81         node ret;
82         ret.sm = a.sm + b.sm;
83         ret.pre = max(a.pre, a.sm+b.pre);
84         ret.suf = max(b.suf, a.suf+b.sm);
85         ret.ans = max({a.ans, b.ans, a.suf+b.pre});
86         return ret;
87     }
88
89     void push(int nd, int l, int r){
90
91         if(!lazy_vis[nd]) return;
92
93         node &t = tree[nd];
94         t.sm = lazy[nd] * (r-l+1);
95         t.pre = t.suf = t.ans = max(t.sm, int(0));
96
97         if(l != r){
98             lazy[nd*2] = lazy[nd*2+1] = lazy[nd];
99             lazy_vis[nd*2] = lazy_vis[nd*2+1] = true;
100         }
101
102         lazy_vis[nd] = false;
103     }
104
105     void build(int nd, int l, int r){
106
107         if(l == r){
108             node &t = tree[nd];
109             t.sm = arr[rin[l]];
110             t.pre = t.suf = t.ans = max(t.sm, int(0));
111             return;
112         }
113
114         int mid = (l+r)/2;
115         build(nd*2, l, mid);
116         build(nd*2+1, mid+1, r);
117
118         tree[nd] = merge(tree[nd*2], tree[nd*2+1]);
119     }
120
121     void update(int nd, int l, int r, int q_l, int q_r, int val){
122
123         push(nd, l, r);
124
125         if(r < q_l || q_r < l){
126             return;
127         }
128
129         if(q_l <= l && r <= q_r){
130             lazy[nd] = val;
131             lazy_vis[nd] = true;
132             push(nd, l, r);
133             return;
134         }
135
136         int mid = (l+r)/2;
137         update(nd*2, l, mid, q_l, q_r, val);
138         update(nd*2+1, mid+1, r, q_l, q_r, val);

```

```

139
140     tree[nd] = merge(tree[nd*2], tree[nd*2+1]);
141 }
142
143 node query(int nd, int l, int r, int q_l, int q_r){
144     push(nd, l, r);
145
146     if(r < q_l || q_r < l){
147         return (node){0, 0, 0, 0};
148     }
149
150     if(q_l <= l && r <= q_r){
151         return tree[nd];
152     }
153
154     int mid = (l+r)/2;
155     node stPath = query(nd*2, l, mid, q_l, q_r);
156     node ndPath = query(nd*2+1, mid+1, r, q_l, q_r);
157
158     return merge(stPath, ndPath);
159 }
160
161 void dfs(int u, int p){
162     sz[u] = 1;
163
164     for(auto &v : adj[u]) if(v != p){
165
166         depth[v] = depth[u] + 1;
167
168         par[v][0] = u;
169         for(int i = 1; i < log_n; i++){
170             par[v][i] = par[par[v][i-1]][i-1];
171         }
172
173         dfs(v, u);
174         sz[u] += sz[v];
175
176         if((sz[v] > sz[adj[u][0]]) || (adj[u][0] == p)){
177             swap(adj[u][0], v);
178         }
179     }
180 }
181
182 void hld(int u, int p){
183     in[u] = t;
184     rin[t] = u;
185     t++;
186
187     for(auto v : adj[u]) if(v != p){
188         nxt[v] = (v == adj[u][0]) ? nxt[u] : v;
189         hld(v, u);
190     }
191
192     out[u] = t;
193 }
194
195 int lca(int u, int v){
196     if(depth[u] < depth[v]) swap(u, v);
197     int dif = depth[u] - depth[v];
198     for(int i = 0; i < log_n; i++) if(dif & (1<<i)) u = par[u][i];
199     if(v==u) return u;
200     for(int i = log_n-1; i >= 0; i--) if(par[u][i] != par[v][i]) u = par[u][i], v =
201     par[v][i];
202     return par[u][0];
203 }
204
205
206

```

```

207 node query_up(int u, int p){
208
209     node ans = (node){0, 0, 0, 0};
210
211     while(true){
212
213         if(nxt[u] == nxt[p]){
214             if(u==p) break;
215             node que = query(1, 0, n-1, in[p]+1, in[u]);
216             ans = merge(que, ans);
217             break;
218         }
219
220         node que = query(1, 0, n-1, in[nxt[u]], in[u]);
221         ans = merge(que, ans);
222         u = par[nxt[u]][0];
223     }
224
225     return ans;
226 }
227
228 void update_up(int u, int p, int val){
229
230     while(true){
231
232         if(nxt[u] == nxt[p]){
233             update(1, 0, n-1, in[p], in[u], val);
234             break;
235         }
236
237         update(1, 0, n-1, in[nxt[u]], in[u], val);
238         u = par[nxt[u]][0];
239     }
240 }
241
242 int32_t main(){
243
244     fastIO;
245
246     #ifdef LOCAL
247         freopenI;
248         freopenO;
249     #endif
250
251     // freopen("name.in", "r", stdin);
252
253     cin >> n;
254     log_n = ceil(log2(double(n)));
255
256     for(int i = 0; i < n; i++) cin >> arr[i];
257
258     for(int i = 0; i < n-1; i++){
259
260         int u, v;
261         cin >> u >> v;
262         u--, v--;
263
264         adj[u].push_back(v);
265         adj[v].push_back(u);
266     }
267
268     dfs(0, -1);
269     hld(0, -1);
270     build(1, 0, n-1);
271
272     int m; cin >> m; while(m--){
273
274         int q;
275         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

```