

```

1  /***12998- Tree Weights:
2  Given a rooted tree with N nodes. Nodes are labeled 1 to N, 1 being the root
3  of the tree. Each of the leaves of this tree has a value assigned to it, which
4  is zero at the beginning. The value for each internal node U is calculated as
5  the sum of the values of all the nodes in the sub-tree rooted at U. An internal
6  node is a node, which has at least one child node.
7  You will be given two kinds of operations:
8    Type 1: given U, find the value of node U.
9    Type 2: given U and X, increase the value of the Leaf U with X.
10 Given N (0<N<=10^5), and N-1 edges. two integers U and V, indicating an edge
11 between U and V. Next there will be Q (0 < Q <=10^5), number of operations.
12 Next Q line will contain firstly TP ('1' or '2'), the type of the operation.
13 Then based on the operation type, there will be one or two integers, U or U
14 and X (1<=U<=N, |X|<=10^9). In case of TP = 2, U will always be a Leaf node.
15 For each operation of type 1, print the answer modulo 1,000,000,007.
16 */
17 #include<bits/stdc++.h>
18 using namespace std;
19 #define ll long long
20 #define mod 1000000007
21 #define mx 100005
22 ll lev[mx+5],dis[mx+5],fin[mx+5],a[mx+5];
23 ll PW[mx+5],IN[mx+5],tree[8*mx+5],mm;
24 bool vis[mx+5];
25 vector<int>ed[mx+5];
26 void dfs(int u,int lv){
27     vis[u] = true; lev[u] = lv; dis[u] = ++mm;
28     for(int i=0; i<ed[u].size(); i++){
29         int v = ed[u][i];
30         if(!vis[v]) dfs(v,lv+1);
31     }
32     fin[u] = ++mm;
33 }
34 void update(int nd,int b,int e,int u,ll x){
35     if(u<b||u>e) return;
36     if(b==u&&e==u){tree[nd]=x; return;}
37     int lf=2*nd, rg=2*nd+1, md=(b+e)/2;
38     update(lf,b,md,u,x);
39     update(rg,md+1,e,u,x);
40     tree[nd] = (tree[lf]+tree[rg])%mod;
41 }
42 ll query(int nd,int b,int e,int u,int v){
43     if(v<b||u>e) return 0;
44     if(b>=u&&e<=v){return tree[nd];}
45
46     int lf=2*nd, rg=2*nd+1, md=(b+e)/2;
47     ll lp = query(lf,b,md,u,v);
48     ll rp = query(rg,md+1,e,u,v);
49     return (lp+rp)%mod;
50 }
51 ll BIGMOD(ll b,ll p){
52     ll ans = 1, k = b;
53     while(p){
54         if(p&1){
55             ans *= k; ans%=mod;
56         }
57         k*=k; k%=mod;
58         p = p>>1LL;
59     }
60     return ans;
61 }
62 ll MODINVERSE(ll b){
63     return BIGMOD(b,mod-2);
64 }
65 void PRECALCULATION(){
66     PW[0] = 1;
67     IN[0] = MODINVERSE(PW[0]);
68     for(int i=1; i<mx; i++){
69         PW[i] = (PW[i-1] + PW[i-1])%mod;
70         IN[i] = MODINVERSE(PW[i]);
71     }
72 }
```

```
73 int main(){
74     PRECALCULATION();
75
76     int tt; scanf("%d",&tt);
77     for(int ks=1; ks<=tt; ks++){
78         printf("Case %d:\n",ks);
79         int n; scanf("%d",&n);
80         for(int i=1; i<n; i++){
81             int u,v; scanf("%d%d",&u,&v);
82             ed[u].push_back(v);
83             ed[v].push_back(u);
84         }
85
86         memset(tree,0,sizeof(tree));
87         memset(vis,false,sizeof(vis));
88         memset(a,0,sizeof(a));
89         mm = 0;
90
91         dfs(1,0);
92
93         int q; scanf("%d",&q);
94         for(int i=1; i<=q; i++){
95             int tp; scanf("%d",&tp);
96             if(tp==1) {
97                 int u; scanf("%d",&u);
98                 ll sum;
99                 if(fin[u]-dis[u]>1){ // internal node
100                     sum = query(1,1,mm,dis[u],fin[u]);
101                     ll lx = IN[lev[u]];
102                     sum = (sum*lx)%mod;
103                 }else{
104                     sum = a[u]%mod; // Leaf node
105                 }
106                 printf("%lld\n",sum);
107             }else{
108                 ll u,x; scanf("%lld%lld",&u,&x);
109
110                 a[u] = a[u] + x; a[u] = (a[u]+mod)%mod;
111
112                 ll lx;
113                 if(lev[u]==0) lx = 1;
114                 else lx = PW[lev[u]-1];
115
116                 ll sum = (a[u]*lx)%mod;
117                 update(1,1,mm,dis[u],sum);
118             }
119         }
120
121         for(int i=0; i<=n; i++)ed[i].clear();
122     }
123     return 0;
124 }
```