```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int, int> P;
const int inf = numeric_limits<int>::max()/2;
const int MAX_V = 100010;
const int MAX_N = (1 << 21) - 1;

int v;
vector< vector<int> > G;
int root;
int vs[MAX_V * 2];
int depth[MAX_V * 2];
int id[MAX_V];

P segtree[MAX_N];

void init_segtree(int k,int l,int r){
  if(r - l == 1){
    segtree[k] = P(depth[l],vs[l]);
  }else{
    int chl = k*2+1;
    int chr = k*2+2;
    init_segtree(chl, l, (l+r)/2);
    init_segtree(chr, (l+r)/2, r);
    segtree[k] = min(segtree[chl],segtree[chr]);
  }
}

P find(int a,int b,int k,int l,int r){
  if(r <= a || b <= l)return P(inf,-1);
  else if(a <= l && r <= b)return segtree[k];
  else{
    int chl = k*2+1;
    int chr = k*2+2;
    P vl = find(a,b,chl,l,(l+r)/2);
    P vr = find(a,b,chr,(l+r)/2,r);
    return min(vl,vr);
  }
}

void dfs(int n,int p,int d,int &k){
  id[n] = k;
  vs[k] = n;
  depth[k] = d;
  ++k;
  for(int i : G[n]){
    if(i != p){
      dfs(i,n,d+1,k);
      vs[k] = n;
      depth[k] = d;
      ++k;
    }
  }
}

void init(){
  fill(vs,vs + MAX_V*2,-1);
  fill(depth,depth + MAX_V*2,inf);
  fill(id,id + MAX_V,-1);
  int k = 0;
  dfs(root,-1,0,k);
  init_segtree(0,0,2*v);
}

int lca(int a,int b){
  return find(min(id[a],id[b]),max(id[a],id[b])+1,0,0,2*v).second;
}

int main(void){
  root = 0;
  cin >> v;
  G.clear();G.resize(v);
  for(int i = 0;i < v;++i){
    int k;
    cin >> k;
    for(int j = 0;j < k;++j){
      int c;
      cin >> c;
      G[i].push_back(c);
    }
  }
  init();
  int q;
  cin >> q;
  for(int i = 0;i < q;++i){
    int a,b;
    cin >> a >> b;
    cout << lca(a,b) << endl;
  }
  return 0;
}
```