Week 17: Graph & Tree Advanced Topics

Topics: - Heavy-Light Decomposition (HLD) - Euler Tour Technique for trees - Lowest Common Ancestor (LCA) algorithms (Binary Lifting, RMQ) - Tree DP (subtree sums, paths, diameters) - Path queries and updates on trees

Weekly Tips: - HLD splits tree into chains for efficient path queries and updates. - Euler Tour linearizes tree to allow segment tree operations. - LCA is essential for answering ancestor queries in O(log n). - Tree DP helps solve problems related to subtrees, diameters, and paths. - Practice combining these techniques to answer complex tree queries efficiently.

Problem 1: Lowest Common Ancestor (Binary Lifting) Link: CSES LCA **Difficulty:** Intermediate

C++ Solution with Explanation Comments:

```cpp
#include <bits/stdc++.h>
using namespace std;
const int MAXN=100005, LOG=17;
vector<int> adj[MAXN];
int up[MAXN][LOG];
int depth[MAXN];
void dfs(int v,int p){
    up[v][0]=p;
    for(int i=1;i<LOG;i++) up[v][i]=up[up[v][i-1]][i-1];
    for(int u:adj[v]) if(u!=p){ depth[u]=depth[v]+1; dfs(u,v); }
}
int lca(int u,int v){
    if(depth[u]<depth[v]) swap(u,v);
    for(int i=LOG-1;i>=0;i--) if(depth[u]-(1<<i)>=depth[v]) u=up[u][i];
    if(u==v) return u;
    for(int i=LOG-1;i>=0;i--) if(up[u][i]!=up[v][i]){ u=up[u][i]; v=up[v][i]; }
    return up[u][0];
}
int main(){
    int n,q; cin>>n>>q;
    for(int i=0;i<n-1;i++){
        int u,v; cin>>u>>v;
        adj[u].push_back(v); adj[v].push_back(u);
    }
    dfs(1,1);
    while(q--){
        int u,v; cin>>u>>v;
        cout<<lca(u,v)<<endl;
    }
}
```

Explanation Comments: - Binary lifting precomputes ancestors at powers of two. - Depth alignment ensures both nodes are at the same level. - Jump up by powers of two to find LCA in O(log n).

Problem 2: Subtree Sum Queries (Tree DP + Euler Tour) Link: [CSES Subtree Queries] **Difficulty:** Intermediate

C++ Solution with Explanation Comments:

```cpp
#include <bits/stdc++.h>
using namespace std;
const int MAXN=100005;
vector<int> adj[MAXN];
int in[MAXN], out[MAXN], timer=0;
long long a[MAXN], bit[2*MAXN];
void update(int idx,long long val,int n){ for(;idx<=n;idx+=idx&-idx) bit[idx]
+=val; }
long long query(int idx){ long long res=0; for(;idx>0;idx-=idx&-idx)
res+=bit[idx]; return res; }
void dfs(int v,int p){
    in[v]=++timer; update(in[v],a[v],MAXN);
    for(int u:adj[v]) if(u!=p) dfs(u,v);
    out[v]=timer;
}
int main(){
    int n,q; cin>>n>>q;
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=0;i<n-1;i++){
        int u,v; cin>>u>>v;
        adj[u].push_back(v); adj[v].push_back(u);
    }
    dfs(1,0);
    while(q--){
        int v; cin>>v;
        cout<<query(out[v])-query(in[v]-1)<<endl;
    }
}
```

Explanation Comments: - Euler tour flattens tree to allow subtree range queries. - BIT (Fenwick Tree) used for efficient prefix sum updates. - Subtree sum = sum of values in the range [in[v], out[v]]. - Time complexity: O(n log n) for preprocessing, O(log n) per query.

End of Week 17 - Master advanced tree techniques: HLD, Euler Tour, LCA, Tree DP. - Practice combining queries and updates efficiently. - These skills are critical for complex tree problems in ACM-ICPC.