### Week 35: Advanced Dynamic Programming – DP on Trees & Tree Decomposition

**Topics:** - Tree DP Basics: Rooted Tree, Subtree Sizes, and DP States - DP on Trees for Paths, Subtree Sums, and Diameter Problems - Rerooting Technique for Tree DP - Heavy-Light Decomposition (HLD) for Path Queries - Centroid Decomposition for Divide-and-Conquer on Trees - Applications: Tree Queries, Path Counting, Optimization Problems

**Weekly Tips:** - Identify the DP state: what each node needs from its children and parent. - Rerooting allows computing DP values for all nodes efficiently. - HLD splits tree into paths to allow segment tree queries on paths. - Centroid decomposition splits tree recursively on centroids for efficient divide-and-conquer. - Visualize DP transitions to understand parent-child dependencies.

**Problem 1: Tree DP – Subtree Sums** **Link:** [CSES Tree Distances I](https://cses.fi/problemset/task/1132/) **Difficulty:** Advanced

**C++ Solution with Explanation Comments:**

#include <bits/stdc++.h>  
using namespace std;  
vector<vector<int>> adj;  
vector<int> sz, dp;  
void dfs(int u,int p){  
 sz[u]=1; dp[u]=0;  
 for(int v:adj[u]) if(v!=p){  
 dfs(v,u);  
 sz[u]+=sz[v];  
 dp[u]+=dp[v]+sz[v];  
 }  
}  
void dfs2(int u,int p,int n){  
 for(int v:adj[u]) if(v!=p){  
 dp[v]=dp[u]-sz[v]+(n-sz[v]);  
 dfs2(v,u,n);  
 }  
}  
int main(){  
 int n; cin>>n; adj.assign(n,{}); sz.assign(n,0); dp.assign(n,0);  
 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); dfs2(0,-1,n);  
 for(int x:dp) cout<<x<<' '; cout<<endl;  
}

**Explanation Comments:** - First DFS computes subtree sizes and DP values. - Second DFS reroots DP values to compute result for all nodes. - Efficient O(n) solution for tree-wide DP queries.

**Problem 2: Heavy-Light Decomposition (HLD) for Path Queries** **Link:** [CP-Algorithms HLD](https://cp-algorithms.com/graph/hld.html) **Difficulty:** Advanced

**C++ Solution with Explanation Comments:**

#include <bits/stdc++.h>  
using namespace std;  
vector<vector<int>> adj;  
vector<int> parent, depth, heavy, head, pos;  
int cur\_pos;  
int dfs(int v){  
 int size=1,max\_c\_size=0;  
 for(int c:adj[v]) if(c!=parent[v]){  
 parent[c]=v; depth[c]=depth[v]+1;  
 int c\_size=dfs(c);  
 size+=c\_size;  
 if(c\_size>max\_c\_size){ max\_c\_size=c\_size; heavy[v]=c; }  
 }  
 return size;  
}  
void decompose(int v,int h){  
 head[v]=h; pos[v]=cur\_pos++;  
 if(heavy[v]!=-1) decompose(heavy[v],h);  
 for(int c:adj[v]) if(c!=parent[v] && c!=heavy[v]) decompose(c,c);  
}  
int main(){  
 int n; cin>>n; adj.assign(n,{});  
 parent.assign(n,-1); depth.assign(n,0); heavy.assign(n,-1);  
 head.assign(n,0); pos.assign(n,0); cur\_pos=0;  
 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); decompose(0,0);  
 for(int i=0;i<n;i++) cout<<head[i]<<' '<<pos[i]<<endl;  
}

**Explanation Comments:** - DFS finds heavy child for each node. - Decompose assigns chains for HLD, enabling segment tree queries on paths. - HLD allows efficient query/update on tree paths, O(log n) per operation.

**End of Week 35** - Advanced DP on trees with rerooting, HLD, and centroid decomposition is key for tree-based ACM-ICPC problems. - Practice subtree DP, path queries, and rerooting techniques extensively.