Trees

Definition & Terms

- An acyclic connected and undirected graph
- It contains N nodes and $N-1$ edges
- Every pair of nodes has exactly one simple path between them
- **Forest** : A graph such that each connected component is a tree
- A star graph has two common definitions:
    1. Only one node has degree greater than 1
    2. Only one node has degree greater than 2
- The subtree of a node n are the set of nodes that have n as an ancestor. A node is typically considered to be in its own subtree

Tree traversal & DP

General Approach for Tree+DP

The typical way to traverse a tree is to start a depth-first search at an arbitrary node. eg:

```
void dfs(int s, int e)
{
// process node s
for (auto u : adj[s])
    if (u != e)
        dfs(u, s);

}
```

The function is given two parameters: the current node s and the previous node e. Unlike DFS on graphs, here you don't require a visited array bcoz it is not possible to reach a node from multiple directions.

- Decide what info you wish to store for each node (DP state)
- Root the tree at node 1 (usually)
- Identify the base case -> usually of form dp[leaf] = something
- Identify the recurrence (transition of states)
- Implementation is generally via DFS -> compute for children and combine to get state of current node

General template

```
bool isLeaf(ll u, vl adj[])
{
    if (u != 1 && adj[u].size() == 1)
        return true;
    return false;
}
```

```
void dfs(ll s, ll e, vl adj[], vl &dp)
{
    if (isLeaf(s, adj))
    { // leaf base case
      // dp[s] = something;
        return;
    }
    for (auto u : adj[s])
    { // computing for children
        if (u == e)
            continue;
        dfs(u, s, adj, dp);
    }
    for (auto u : adj[s])
    { // combining for current node
        if (u == e)
            continue;
        // recurrence relation here
    }
}

void solve()
{
    ll n, a, b;
    vl v;
    cin >> n;
    vl adj[n + 1];
    for (ll i = 2; i <= n; i++)
    {
        cin >> a >> b;
        adj[a].emplace_back(b);
        adj[b].emplace_back(a);
    }
    vl dp(n + 1, 0);
    dfs(1, 0, adj, dp);
}
```

Subordinates:

Code Saved.

We use DP on the recursive DFS fn to get number of nodes in subtree of every node (subtree includes the node itself).

No of subordinates is this count excluding the node itself, hence -1.

T.C : $O(n)$

CF - 1528A

Can See : Nice hard question about identifying DP states and transition.

Diameter:

Code saved for both approaches.

The diameter of a tree is the maximum length of a path between two nodes in the tree. (May or may not pass through root)

Obviously, ends of a diamter are always leaves. There may be several maximum-length paths.

Method-1 : DP :

States:

dp[u][0] = max distance b/w any 2 nodes in the subtree of u such that the path b/w those 2 nodes contains u and *has an endpoint at u*

dp[u][1] = max distance b/w any 2 nodes in the subtree of u such that the path b/w those 2 nodes contains u and *doesn't have an endpoint at u*

Base case:

If u is a leaf node, subtree does not have 2 nodes => dp[u][0] = dp[u][1] = 0

Recurrence:

dp[u][0] = 1 + max(dp[v][0]), v belongs to children of u

dp[u][1] : Maximum distance between two distinct children of u, let them be v1 and v2.

then ans = dp[v1][0] + dp[v2][0] + 2

=> dp[u][1] = max(dp[i][0] + dp[j][0]) + 2, where i and j are children of u and i not equal to j

Since we need largest of dp[v][0] of children in both cases, it is a good idea to put dp[v][0] in a vector and sort in descending order.

Final ans = max(max(dp[u][0], dp[u][1])) for all vertices u in the tree.

T.C : Sorting makes it O(n*lgn) though you can also work without sorting in O(n)

Method-2 : 2 BFS :

First, we choose an arbitrary node a in the tree and find the farthest node b from a. Then, we find the farthest node c from b. The diameter of the tree is the distance between b and c.

T.C : O(n)

CF - 1083A

Nice hard question based on diameter DP approach, **look only if time**.

Editorial : Adhish K DP on Trees playlist

Binary Tree :

Postorder + Inorder and Preorder + Inorder **uniquely determine** a tree

But Postorder + Preorder don't