Intro to Tree Algorithms

https://usaco.guide/silver/intro-tree

CP Initiative
joincpi.org

Three Properties of Trees

- A graph is a **tree** iff it is connected and contains N nodes and N−1 edges.

- A graph is a **tree** iff every pair of nodes has exactly one simple path between them.

- A graph is a **tree** iff it is connected and does not contain any cycles.

Tree Terminology

- A **leaf** of a tree is any node in the tree with degree 1.
  - If the tree is rooted, the **root** with a single child is *not* typically considered a leaf, but depending on the problem, this is not always the case.

- A **star graph** has two common definitions. Try to understand what they mean - they typically appear in subtasks.
  - Definition 1: Only one node has degree greater than 1.
  - Definition 2: Only one node has degree greater than 2.

- A **forest** is a graph such that each **connected component** is a tree.

Rooted Trees

- Often, it's helpful to designate an arbitrary node as the **root** of the tree.
- When a tree is rooted, each node has a **parent**, which is the first node on its path to the root.
  - The root does not have a parent. In code, this usually means its parent is set to -1.
- In addition, each node has a **subtree**, which includes all nodes that must pass through it on their path to the root.
- Finally, each node has a **depth**, representing the nodes distance to the root.

DFS on Trees

- Normally, in a DFS function, you consider a starting value and keep track of the visited nodes using a boolean array.

- However, on a tree, **you don't need a visited array**. Just keep track of which node you came from.
    - This is because if you root the tree, the only node you could have visited before is your parent.

DFS Sample Code

```
void dfs(int node, int par){
    for (int to : adj[node]) {
        if (to != par) {
            dfs(to, node);
        }
    }
}
```

Graphs Example Problem

[USACO - Grass Planting]

Grass Planting Solution

USACO - Grass Planting