Tree DP in DSA: Beginner to Advanced Guide

Subtitle: Learn how to apply dynamic programming techniques on tree structures for advanced problem solving.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Tree DP is a **dynamic programming approach applied to tree structures**. Many tree problems require computing values for subtrees and combining results efficiently. This technique is essential for **counting paths, finding diameters, or subtree sums** in trees.

Step 1: Core Concepts

- Trees are **acyclic connected graphs**
- Each node has a **parent-child relationship**
- **DP on trees** involves:
  1. Computing a value for each node based on **child values**
  2. Propagating results **upwards (post-order)** or **downwards (pre-order)**
- Common patterns: **subtree sum, longest path, diameter of tree, number of ways**

Step 2: Tree Representation

- Use an **adjacency list**

```
let tree = {

 0:[1,2],

 1:[0,3,4],

 2:[0],

 3:[1],

 4:[1]

};
```

- Root the tree at any node (commonly node 0)

Step 3: Implementation Example — Subtree Sum

```
let values = [1,2,3,4,5]; // value at each node
let subtreeSum = Array(values.length).fill(0);

function dfs(u, parent){
  subtreeSum[u] = values[u];
  for(let v of tree[u]){
    if(v !== parent){
      dfs(v,u);
      subtreeSum[u] += subtreeSum[v];
    }
  }
}

dfs(0,-1);
console.log(subtreeSum); // sum of subtree for each node
```

Step 4: Implementation Example — Diameter of Tree

```
let diameter = 0;

function dfsDiameter(u, parent){
  let max1=0, max2=0;
  for(let v of tree[u]){
    if(v !== parent){
      let d = dfsDiameter(v,u);
      if(d>max1){ max2=max1; max1=d; }
      else if(d>max2){ max2=d; }
    }
  }
```

```
  diameter = Math.max(diameter, max1+max2);

  return max1+1;

}



dfsDiameter(0,-1);

console.log("Diameter:", diameter);
```

Step 5: Applications

1. **Subtree sum and value propagation**
2. **Tree diameter** (longest path in tree)
3. **Counting paths with constraints**
4. **Dynamic programming problems in competitive programming**

Step 6: Mini Exercises

1. Compute **sum of all nodes in each subtree**
2. Find **diameter** of different tree examples
3. Count **number of nodes at even/odd depth**
4. Solve problems like **maximum path sum in tree**

Step 7: Common Mistakes

- Forgetting to exclude the parent → infinite recursion
- Not properly initializing DP array → incorrect results
- Confusing **pre-order vs post-order traversal**
- Mixing tree representation (adjacency list vs matrix)

Step 8: Practice Plan for Tree DP

Day 1:

- Subtree sum, counting nodes → 3–5 problems

Day 2:

- Diameter, longest path → 3–5 problems

Day 3:

- Mixed Tree DP problems → 5–7 problems

Key Takeaways

- Tree DP helps solve **complex hierarchical problems efficiently**
- Use **post-order DFS** to combine child results
- Practice builds **intuition for subtree-based computations**
- Essential for **competitive programming and advanced graph/tree problems**

Visit haas.dev for more Tree DP guides, problem sets, and interview preparation resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app