Heavy-Light Decomposition (HLD) in DSA: Beginner to Advanced Guide

Subtitle: Learn how to efficiently handle path queries and updates on trees using HLD.

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

Introduction

Heavy-Light Decomposition (HLD) is a technique to **decompose a tree into paths** to efficiently answer **path queries and updates**. It's widely used in **tree path sums, maximum/minimum queries, and segment tree integration**.

Step 1: Core Concepts

- Decompose a tree into **heavy paths** and **light edges**
- **Heavy edge:** connects a parent to its **largest child subtree**
- **Light edge:** all other edges
- This decomposition reduces **tree path queries to O(log n) chains**

Why it matters: HLD allows **efficient path queries using segment trees** on trees.

Step 2: Tree Representation

- Use adjacency list to represent tree:

```
let tree = {
  0:[1,2],
  1:[0,3,4],
  2:[0],
  3:[1],
  4:[1]
};
```

- Root the tree at node 0

Step 3: Key Arrays in HLD

1. parent[u] → parent of node u
2. size[u] → size of subtree rooted at u

3. depth[u] → depth of node u
4. chainHead[u] → head of the heavy path containing u
5. posInBase[u] → position in the **base array** for segment tree

Step 4: Implementation Steps

1. **DFS to compute subtree sizes**
2. **Decompose tree into heavy paths**
3. **Map nodes to base array for segment tree**
4. **Use segment tree for path queries**

```
// Step 1: Compute subtree sizes

function dfs(u, p){

  size[u] = 1;

  for(let v of tree[u]){

    if(v !== p){

      parent[v] = u;

      depth[v] = depth[u]+1;

      dfs(v,u);

      size[u] += size[v];

    }

  }

}



// Step 2: HLD decomposition

let currPos = 0;

function hld(u, p){

  if(chainHead[u] === -1) chainHead[u] = u; // start new chain

  posInBase[u] = currPos++;



  let heavyChild = -1;

  for(let v of tree[u]) if(v !== p)
```

```
    if(heavyChild===-1 || size[v]>size[heavyChild]) heavyChild=v;


  if(heavyChild !== -1){
    chainHead[heavyChild] = chainHead[u]; // continue chain
    hld(heavyChild,u);
  }


  for(let v of tree[u]){
    if(v !== p && v !== heavyChild) hld(v,u); // new chains
  }
}


// Example initialization
let n = 5;
let parent=Array(n).fill(-1), size=Array(n).fill(0), depth=Array(n).fill(0);
let chainHead=Array(n).fill(-1), posInBase=Array(n).fill(0);
dfs(0,-1);
hld(0,-1);
console.log(posInBase, chainHead);
```

Step 5: Applications

1. **Path sum queries in trees**
2. **Maximum/minimum value on path queries**
3. **Subtree updates combined with path queries**
4. **Advanced tree problems in competitive programming**

Step 6: Mini Exercises

1. Compute **sum of values on path** between any two nodes
2. Find **maximum node value on path**
3. Perform **update operations on a node** and propagate queries efficiently
4. Combine **HLD with segment trees for range queries**

Step 7: Common Mistakes

- Miscalculating **heavy child** → wrong decomposition
- Forgetting **chain head updates** → broken chains
- Not maintaining **base array** for segment tree mapping
- Confusing **path queries vs subtree queries**

Step 8: Practice Plan for HLD

Day 1:

- Implement basic HLD decomposition → 2–3 problems

Day 2:

- Integrate with segment tree for path queries → 3–5 problems

Day 3:

- Solve mixed path and update queries → 5–7 problems

Key Takeaways

- HLD decomposes a tree into **heavy paths for efficient queries**
- Essential for **path-based queries in trees**
- Practice **DFS, chain decomposition, and segment tree integration**
- Advanced competitive programming skill for **tree optimization problems**

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

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