# Least Common Ancestor and Binary Lifting

## CS 491 – Competitive Programming

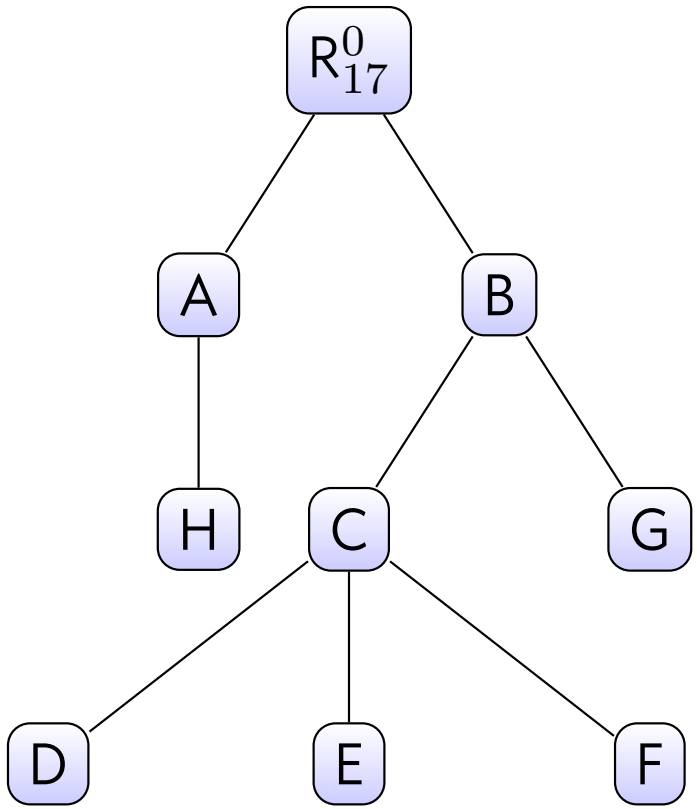
Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Spring 2024

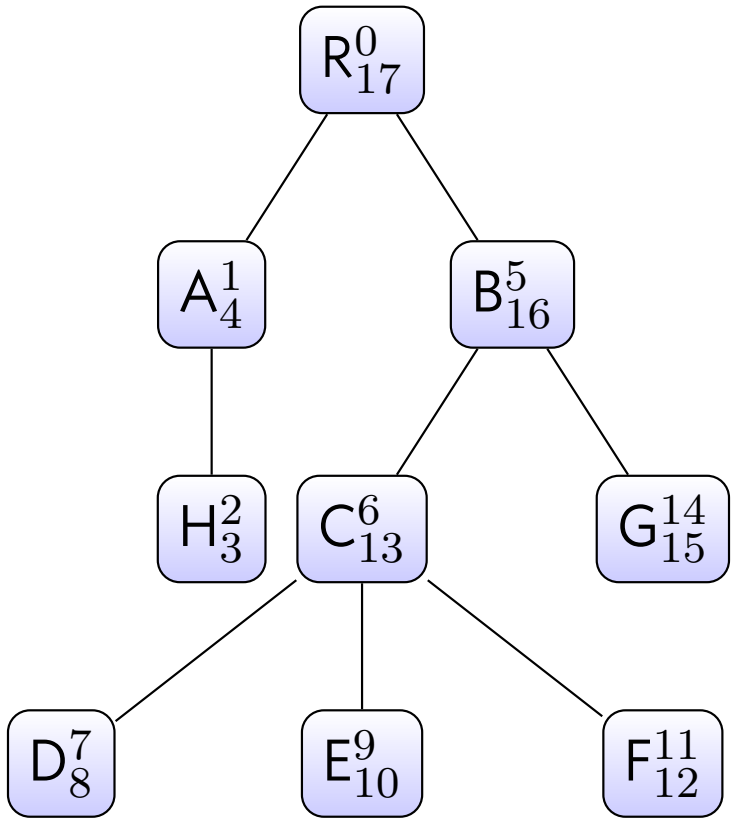
# DFS for Ancestry

```
1  // v is current, p is parent
2  void dfs(int v, int p) {
3      tin[v] = ++timer;
4      up[v][0] = p;
5
6      for (int u : adj[v]) {
7          if (u != p)
8              dfs(u, v);
9
10     tout[v] = ++timer;
11 }
```



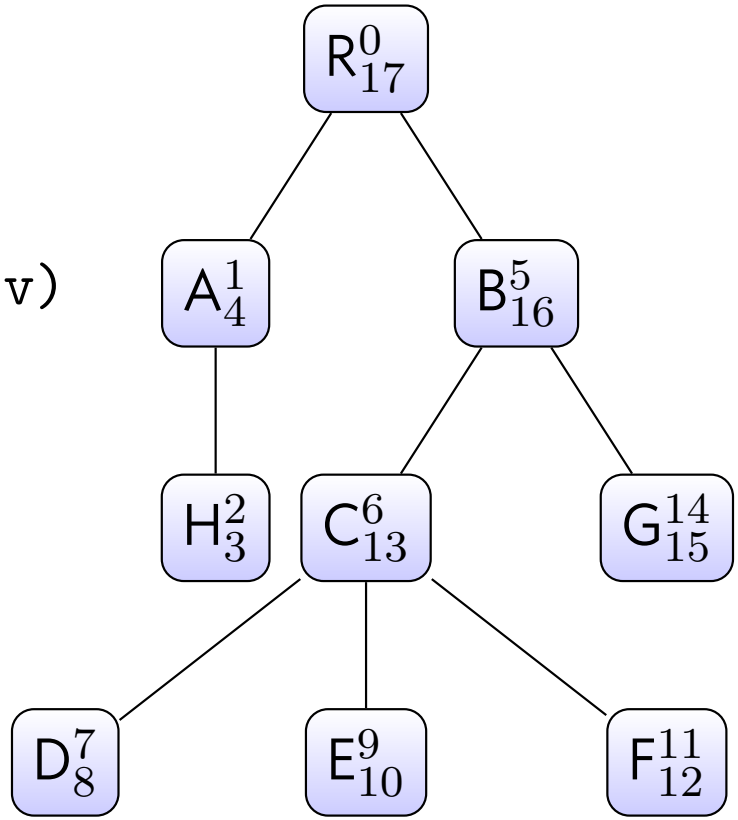
# DFS for Ancestry, Ctd

```
1  // v is current, p is parent
2  void dfs(int v, int p) {
3      tin[v] = ++timer;
4      up[v][0] = p;
5
6      for (int u : adj[v]) {
7          if (u != p)
8              dfs(u, v);
9
10     tout[v] = ++timer;
11 }
```



# Is it an ancestor?

```
1  bool is_ancestor(int u, int v)
2  {
3      return tin[u] <= tin[v]
4        && tout[u] >= tout[v];
5  }
```



# Least Common Ancestor

- We will create a 2D array up to handle this.
  - Size should be $\log n$.

```
l = ceil(log2(n));
up.assign(n, vector<int>(l + 1));
dfs(root, root);

void dfs(int v, int p) {
    tin[v] = ++timer;
    up[v][0] = p;
    for (int i = 1; i <= l; ++i)
        up[v][i] = up[up[v][i-1]][i-1];

    for (int u : adj[v])
        if (u != p) dfs(u, v);

    tout[v] = ++timer;
}
```

# Least Common Ancestor

- Here is the up array for our example. Not very intersting, since the tree is not very deep.

| Node | R | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|---|
| 0 | R | R | R | B | C | C | C | B | A |
| 1 | R | R | R | R | R | R | R | R | R |
| 2 | R | R | R | R | R | R | R | R | R |

- Suppose you have the tree: `0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7` and `3 -> 8 -> 9`. What does the up array look like?

# Answer

- Suppose you have the tree: `0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7` and `3 -> 8 -> 9`. What does the `up` array look like?

| level | steps | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 3 | 8 |
| 1 | 2 | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 2 | 3 |
| 2 | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 0 | 1 |

# Find LCA

```
1  int lca(int u, int v)
2  {
3      if (is_ancestor(u, v))
4          return u;
5      if (is_ancestor(v, u))
6          return v;
7      for (int i = l; i >= 0; --i) {
8          if (!is_ancestor(up[u][i], v))
9              u = up[u][i];
10     }
11     return up[u][0];
12 }
```