6.851 Final Project: Implementing Dynamic LCA

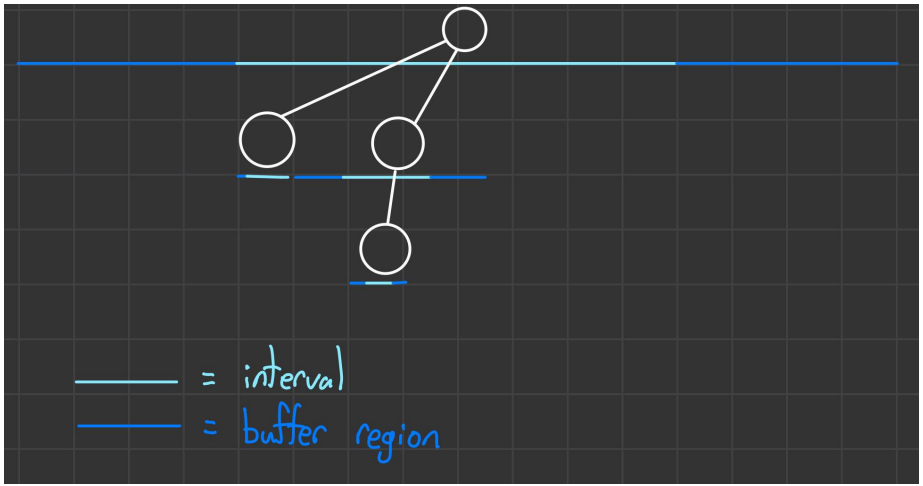
Jonathan Conroy

Background - LCA Problem

- LCA(x, y) = Lowest Common Ancestor of x and y
- Static Tree: Reduce to RMQ
- Dynamic Tree: Gabow's data structure
  - LCA: O(1) worst-case
  - insert_leaf: O(1) amortized possible, O(\log n) amortized implemented

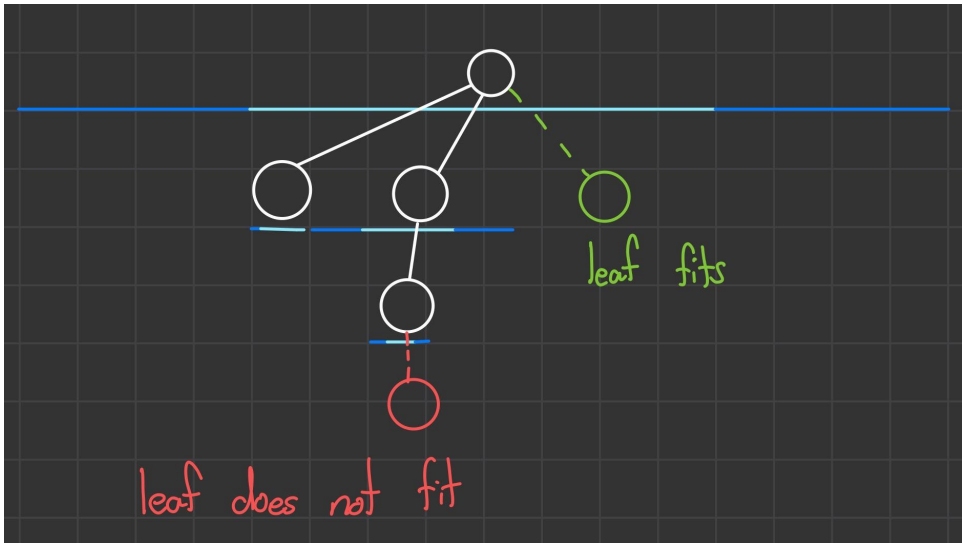
Static Data Structure

- 1. Heavy-light path decomposition: Maintain “compressed tree”(*log n* height)
- 2. Assign “fat preorder” intervals to each node
    - LCA(x, y) must have interval with length greater than |start(x) - start(y)|
- 3. Store “ancestor” tables to compute LCA in *O(1)* time in compressed tree
- *O(n log n)* space and preprocessing time



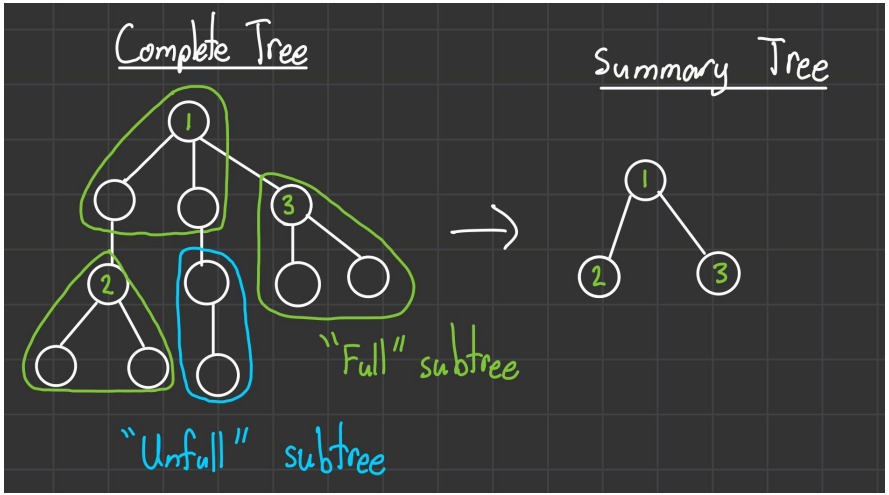
Dynamic Data Structure

- Add a leaf as a new path in the compressed tree
- If a subtree grows too large, reorganize it
- $O(\log^2 n)$ amortized insertion
- $O(n \log n)$ space



Indirection

- Partition nodes into subtrees of size *O(log n)*.
- Maintain “summary” data structure with a node for each full subtree using expensive data structure
- Can solve LCA within subtree in O(1) time



Demo

Testing - Correctness

- Implemented LCA naive algorithm
- Generated random trees (using Prüfer sequences)
    - Random LCA queries on large trees

```
99.991) LCA of 84, 44: 84 and 84
99.992) LCA of 86, 31: 96 and 96
99.993) LCA of 65, 99: 99 and 99
99.994) LCA of 28, 51: 25 and 25
99.995) LCA of 56, 50: 25 and 25
99.996) LCA of 43, 35: 1 and 1
99.997) LCA of 23, 56: 1 and 1
99.998) LCA of 61, 62: 25 and 25
99.999) LCA of 16, 78: 1 and 1
Passed 'multilevel' tests
```

Testing - Performance

Approach	**Construction (ms)**	**Query (μs)**
Naive	2.59	2.90
Static	62.88	0.65
Expensive Dynamic	703.18	0.80
Multilevel Dynamic	3.81	0.49

Table 1: Results of efficiency tests