09-B1

BST Application

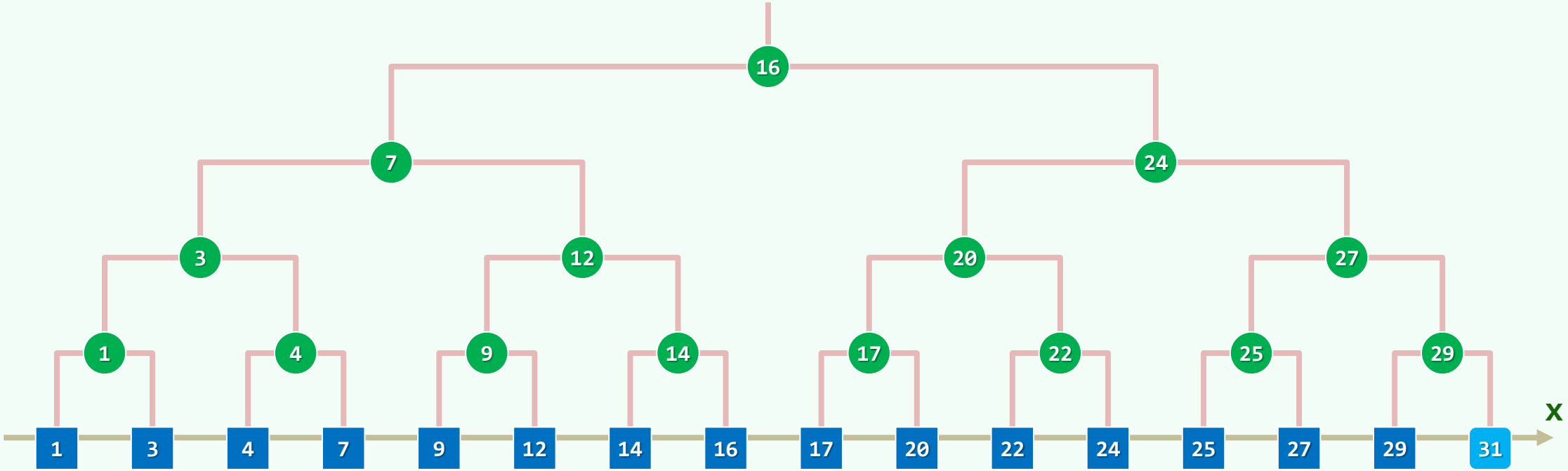
kd-Tree: 1D

邓俊辉

deng@tsinghua.edu.cn

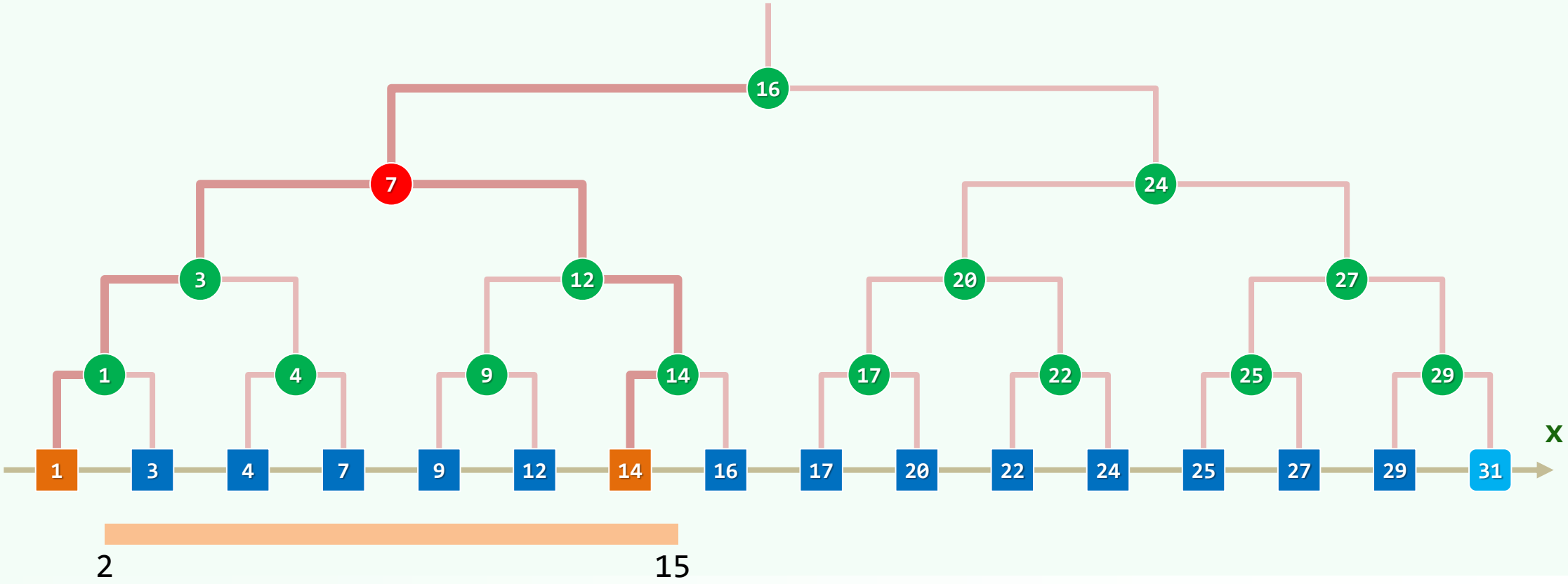
Structure

- For each v, v.key = max{ u.key | u ∈ L-Tree(v) } = v.pred().key
- For each u in L/R-Tree(v), x(u) ≤/> x(v)
- search(x) : returns the maximum key not greater than x



Lowest Common Ancestor

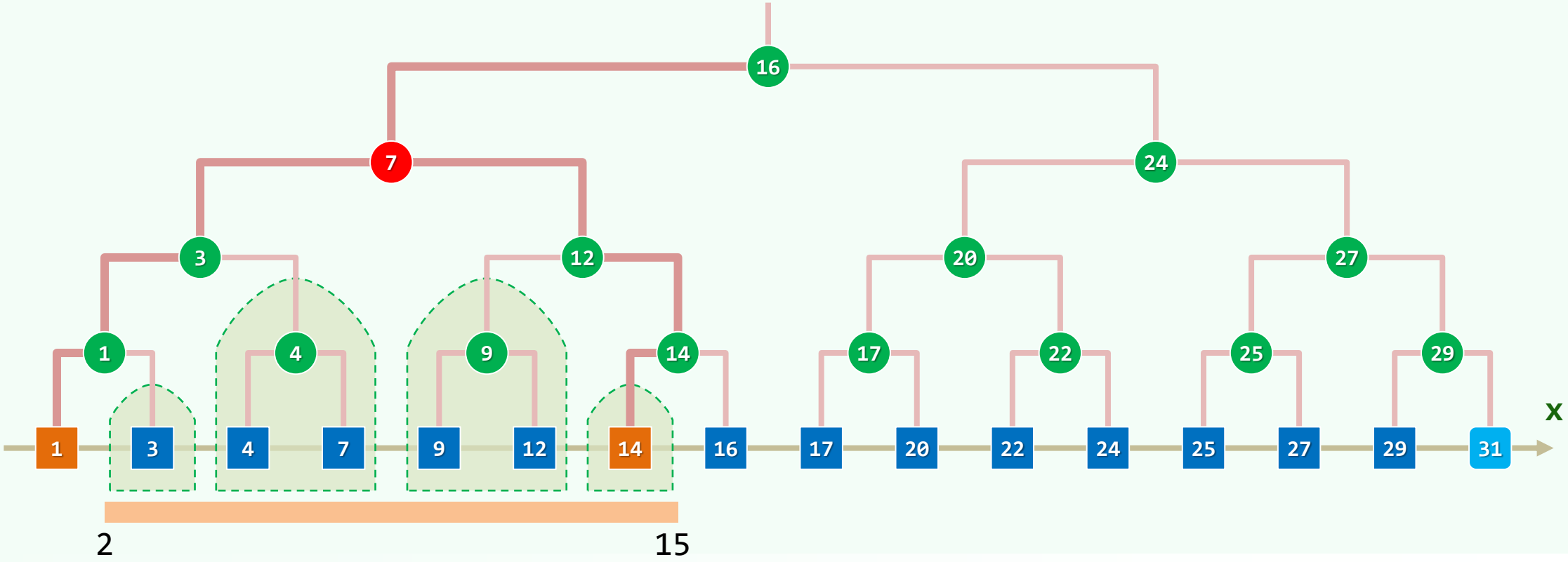
- Consider, as an example, the query for (2, 15] ...
- search(2) = 1 , search(15) = 14 , LCA(1, 14) = 7



Traversal

❖ Starting from the LCA, traverse path(1) and path(14) once more resp.

- All Right/Left-turns along path(1/14) are ignored and
- the Right/Left subtree at each Left/Right-turn is reported



Complexity

❖ **Query:**

$\mathcal{O}(\log n)$

❖ **Preprocessing:**

$\mathcal{O}(n \log n)$

❖ **Storage:**

$\mathcal{O}(n)$