CSE221

Lecture 16:
Splay Trees

Hyungon Moon

Acknowledgment: The content of this file is based on the slides of the textbook as well as the slides provided by Prof. Won-Ki Jeong and Prof. Tsz-Chiu Au.

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Splay Trees

- Two heuristics to make search tree efficient
  - Make search depth shorter (AVL, RB)
  - Make frequently accessed data closer to root
- Splay trees use the latter idea
  - Rearrange tree whenever an element is accessed so that the recently used data is in the root node
- Worst-case O(n) for operation, but O(log n) amortized

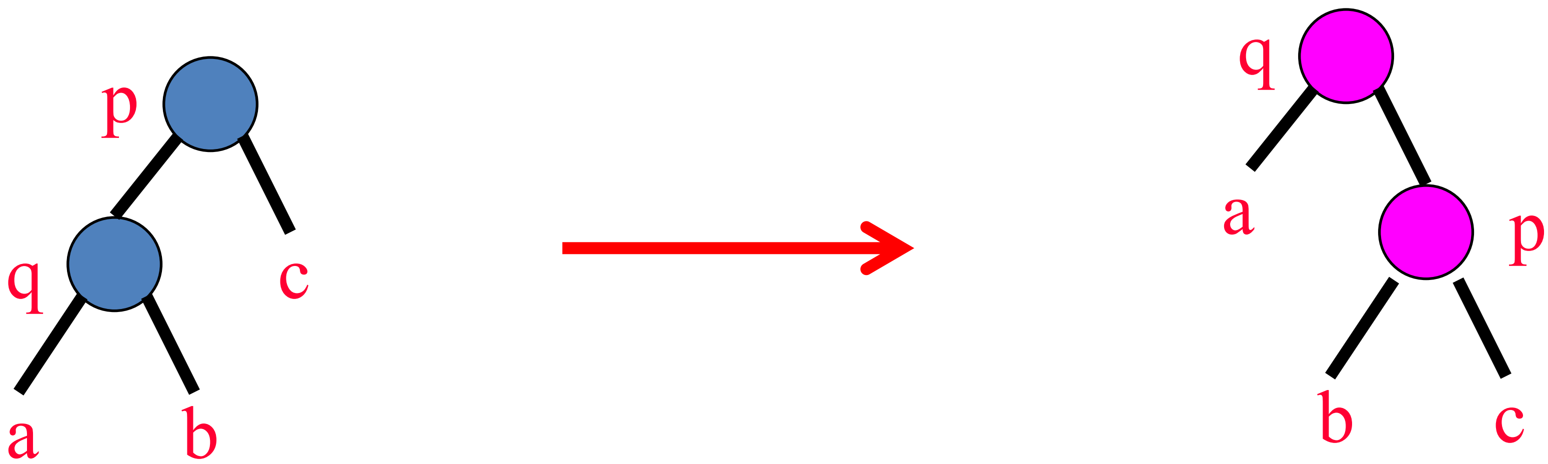
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Splaying

- Designate splay node *q*
  - E.g., accessed node, inserted node …
- Move *q* to the root using a series of splay steps
- In a splay step, *q* moves up the tree by 0, 1, or 2 levels
- Apply to all normal operations on a binary search tree

Splay Step

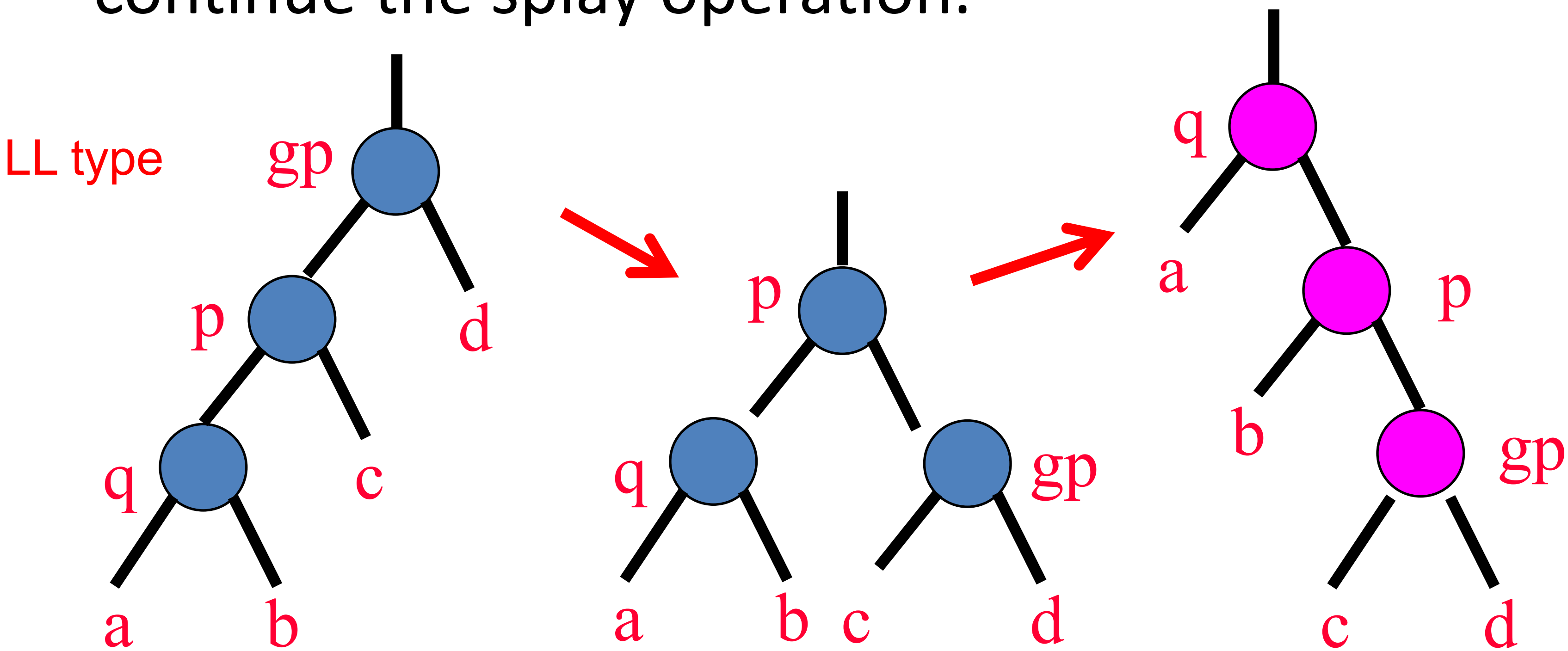
- If *q* = *null* or *q* is the root, stop (splay is over)
- If *q* is at level 2, do a one-level move and terminate the splay operation



- Symmetric: right child of *p*

Splay Step

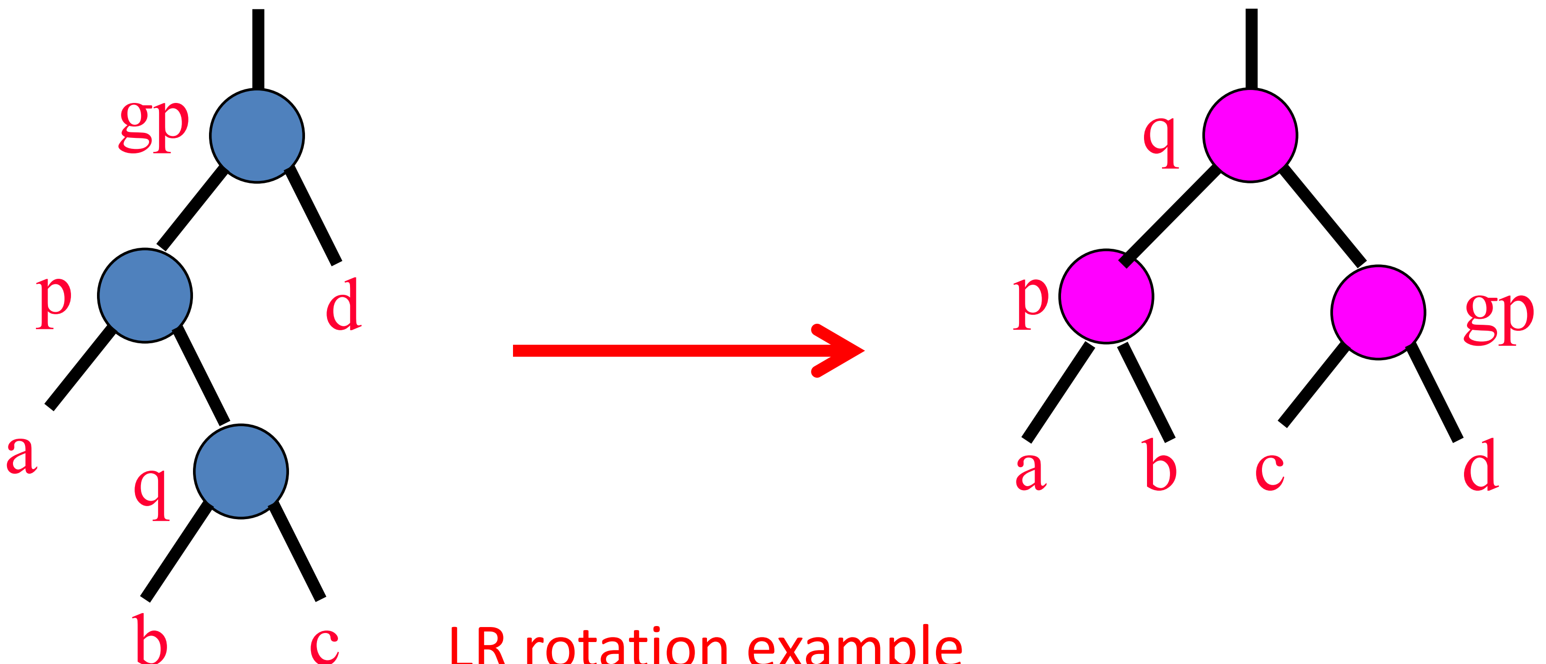
- If *q* is at a level > 2, do a two-level move and continue the splay operation.



- Symmetric: right child of right child of *gp (RR type)*

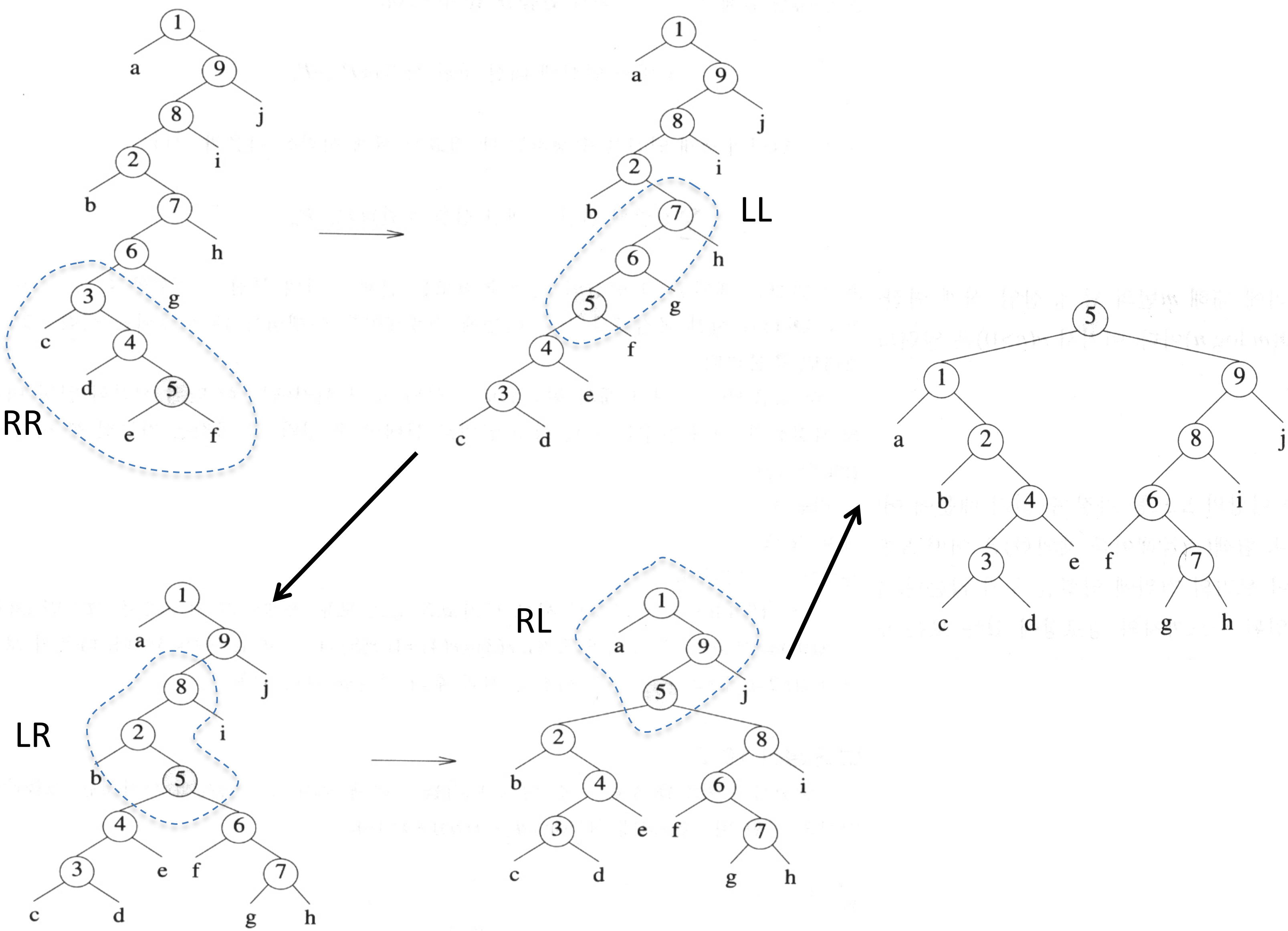
2-Level Move

- LR, RL rotations
  - Similar to AVL/Red-Black



LR rotation example

Splay node : 5

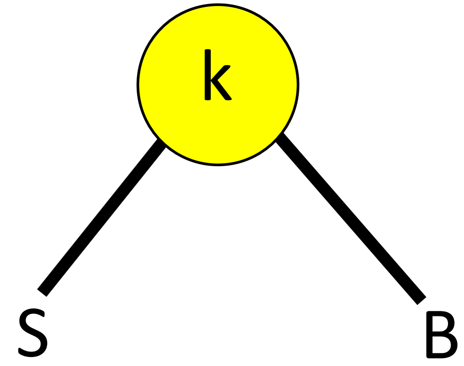


Designating Splay Node

- Search (k)
  - If there is key *k*, the node containing *k* is the splay node
- Insert (k)
  - The newly inserted node is the splay node
- Delete (k)
  - The parent of the physically deleted node is the splay node
  - If *k* is root, there is no splay node

Designating Splay Node

- Split (k)
  - Suppose that the key *k* is actually present in the tree
  - Perform a splay at the node that contains k
  - Then split the tree



Discussion

- Pros
  - Simple to implement when compared with AVL/Red-Black
  - Low memory footprint: no bookkeeping data
  - Optimized for temporal locality
  - O(log n) amortized, faster average performance
- Cons
  - Search gets more expensive since tree rotation can occur
  - No height constraint, the worst time can be O(n)
  - Parallel access is difficult: read-only operation will change the tree

Questions?