Splay Trees

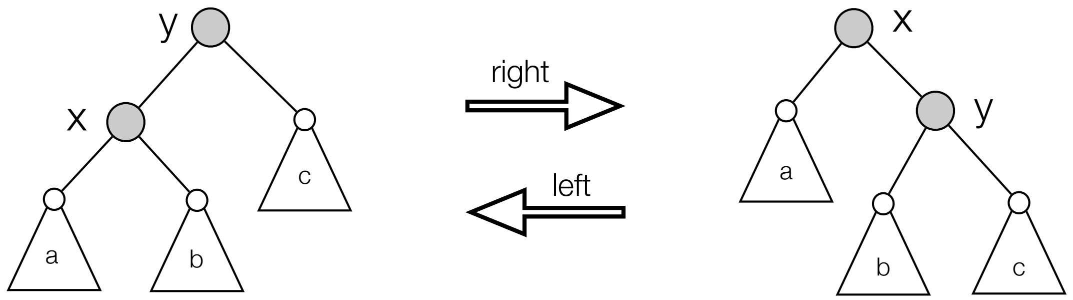
Inge Li Gørtz

Splay Trees

- Self-adjusting BST (Sleator-Tarjan 1983).
  - Most frequently accessed nodes are close to the root.
  - Tree reorganizes itself after each operation.
  - After access to a node it is moved to the root by splay operation.
  - Worst case time for insertion, deletion and search is O(n). Amortised time per operation O(log n).
- Operations. Search, predecessor, sucessor, max, min, insert, delete, join.

Splaying

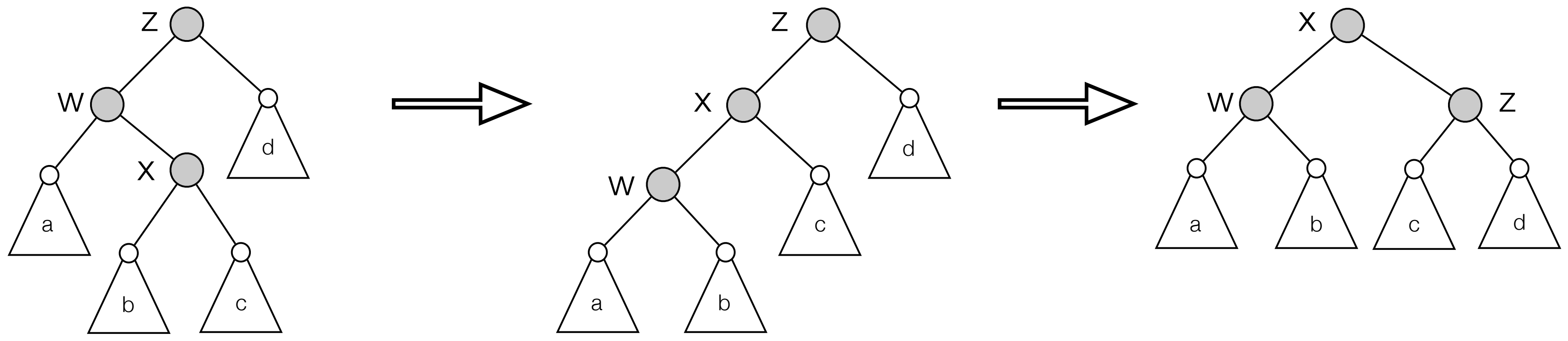
- Splay(x): do following rotations until x is the root. Let y be the parent of x.
    - right (or left): if x has no grandparent.



right rotation at x (and left rotation at y)

Splaying

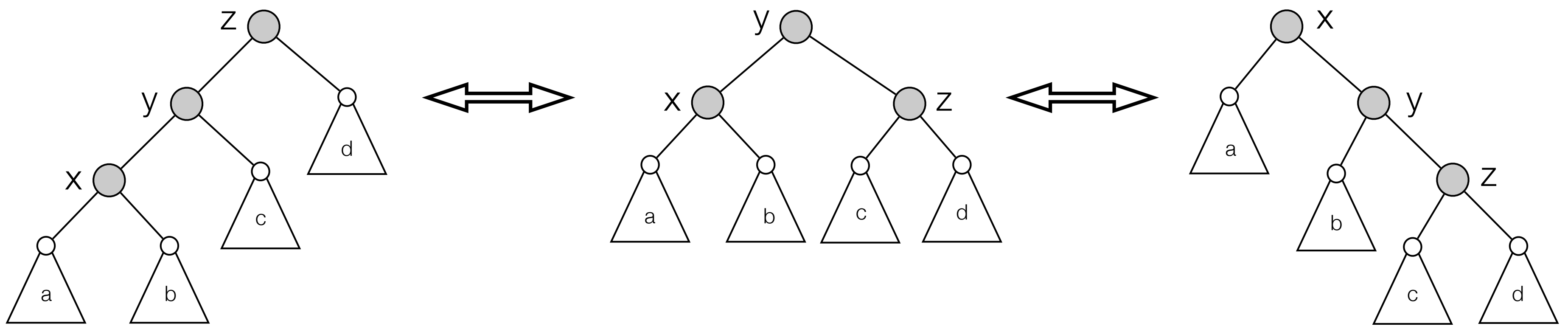
- Splay(x): do following rotations until x is the root. Let p(x) be the parent of x.
  - right (or left): if x has no grandparent.
  - zig-zag (or zag-zig): if one of x,p(x) is a left child and the other is a right child.



zig-zag at x

Splaying

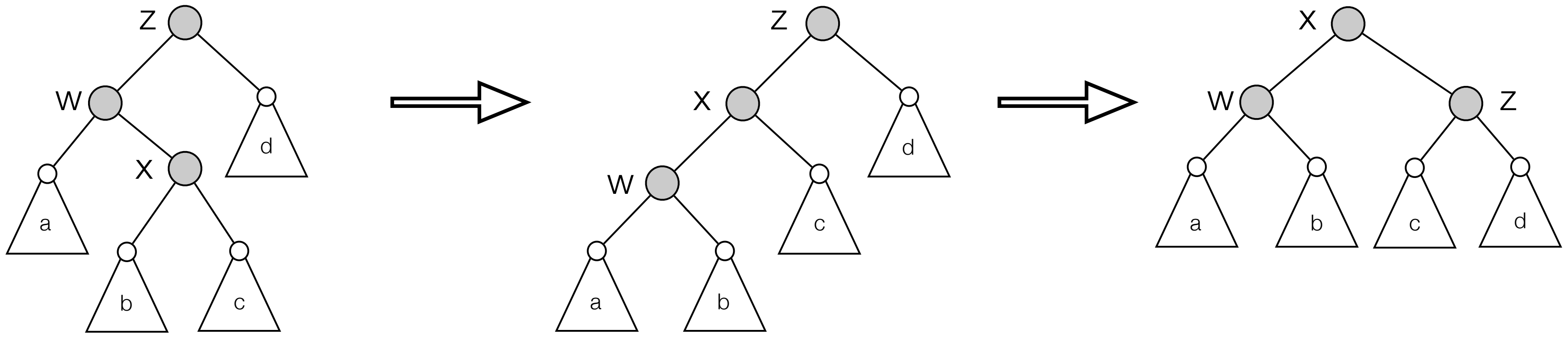
- Splay(x): do following rotations until x is the root. Let y be the parent of x.
    - right (or left): if x has no grandparent.
    - zig-zag (or zag-zig): if one of x,y is a left child and the other is a right child.
    - roller-coaster: if x and p(x) are either both left children or both right children.



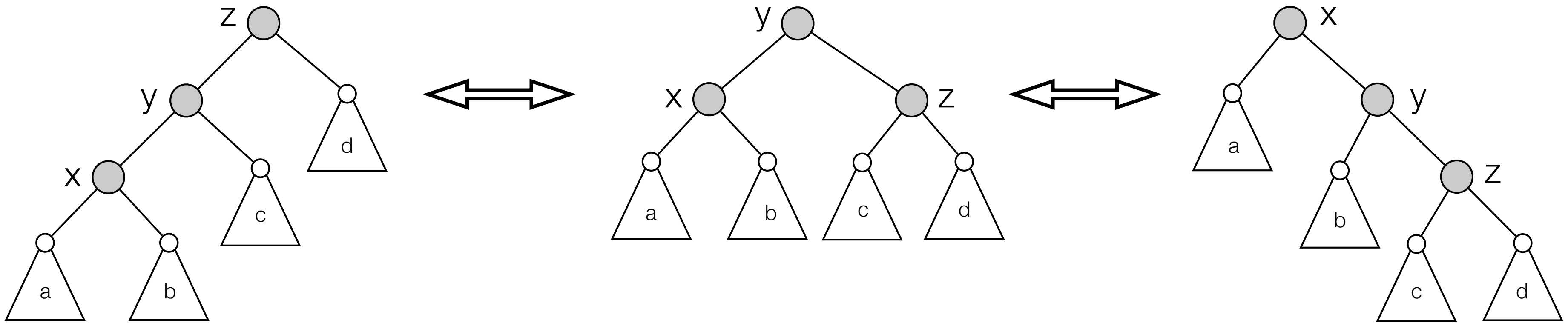
right roller-coaster at x (and left roller-coaster at z)

Splaying

zig-zag at x

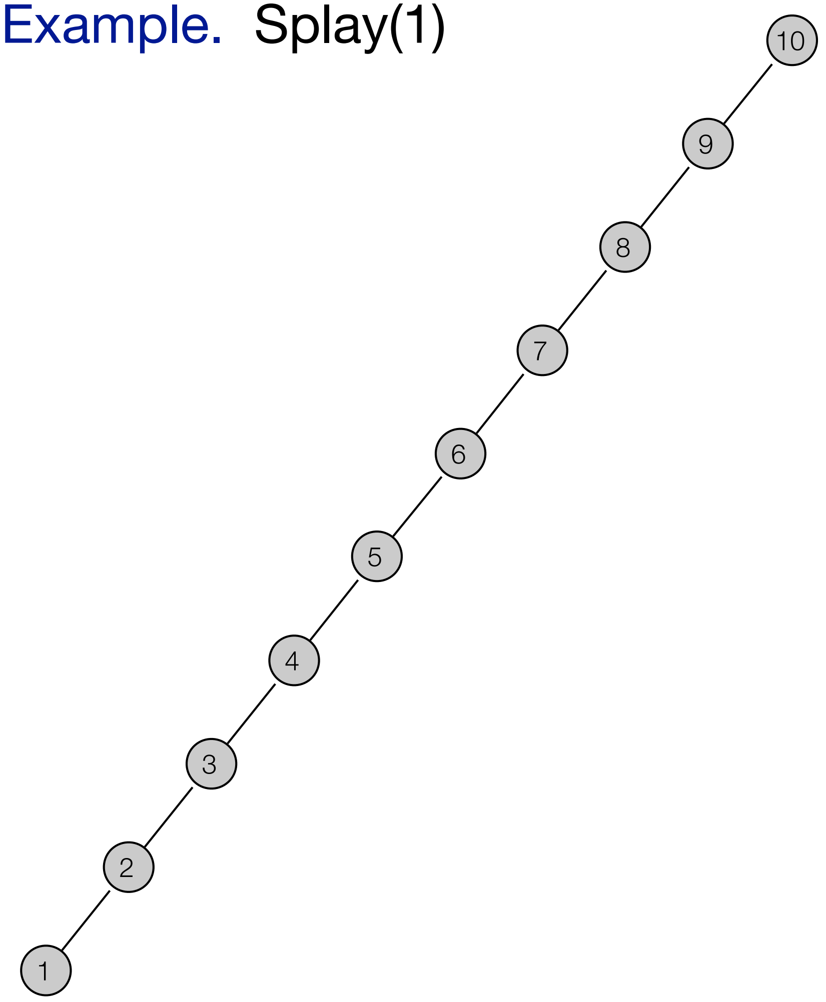


right roller-coaster at x (and left roller-coaster at z)



Splay

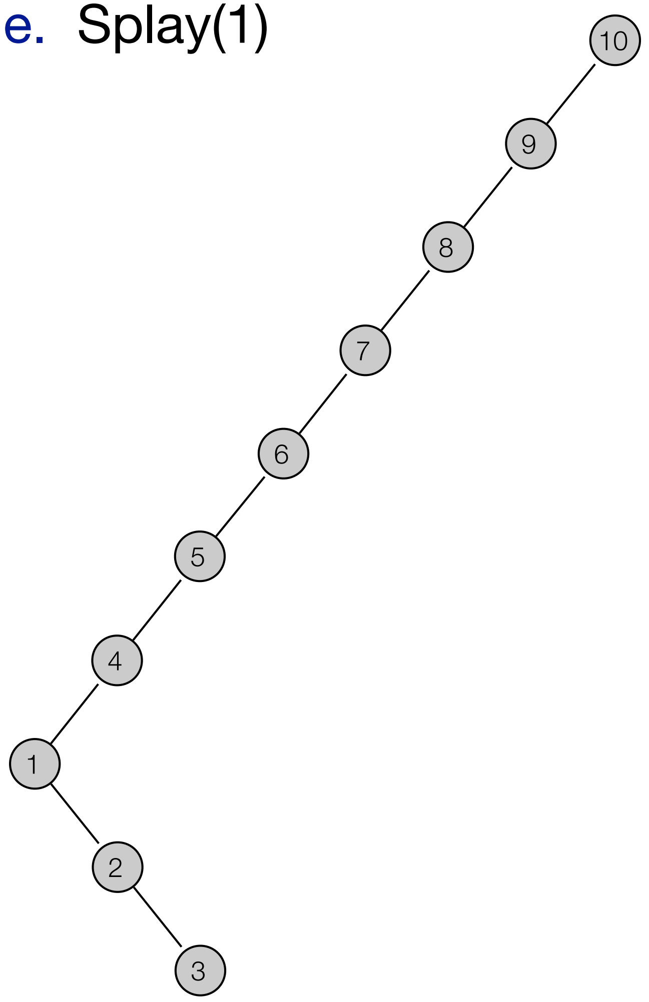
- Example. Splay(1)



right roller-coaster at 1

Splay

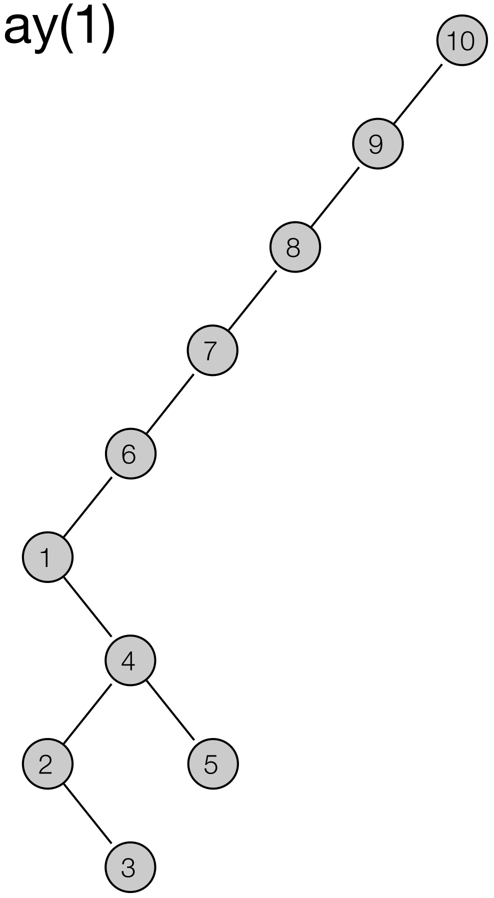
- **Example.** Splay(1)



right roller-coaster at 1

Splay

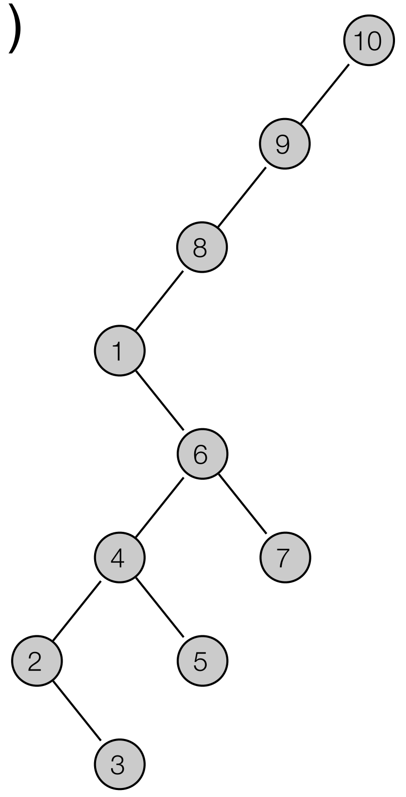
- Example. Splay(1)



right roller-coaster at 1

Splay

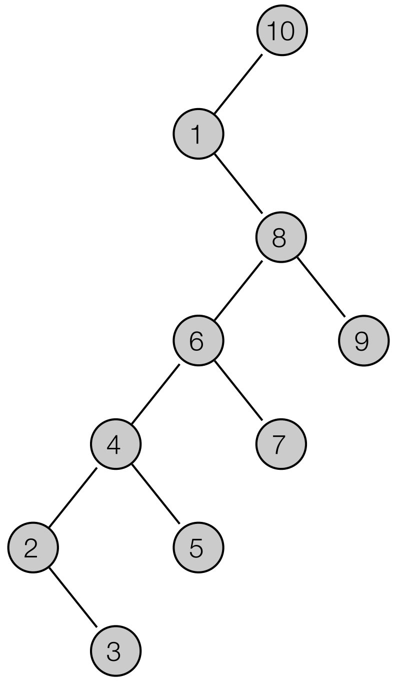
- **Example.** Splay(1)



right roller-coaster at 1

Splay

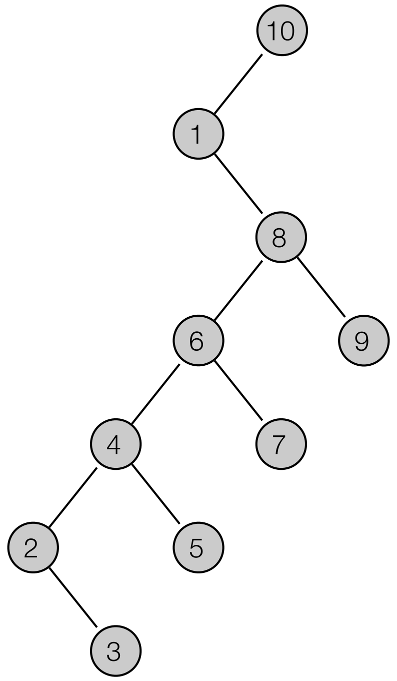
- Example. Splay(1)



right roller-coaster at 1

Splay

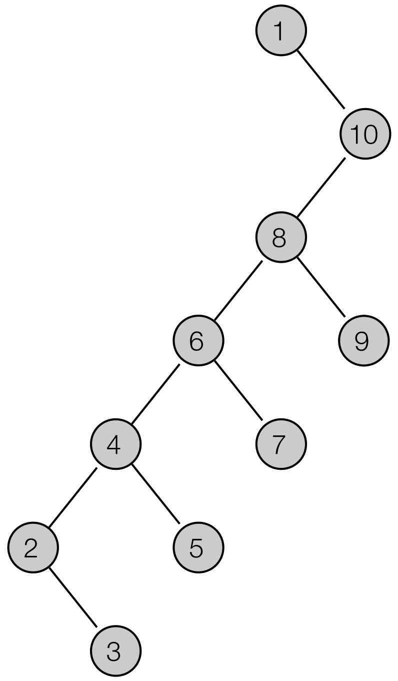
- Example. Splay(1)



right rotation at 1

Splay

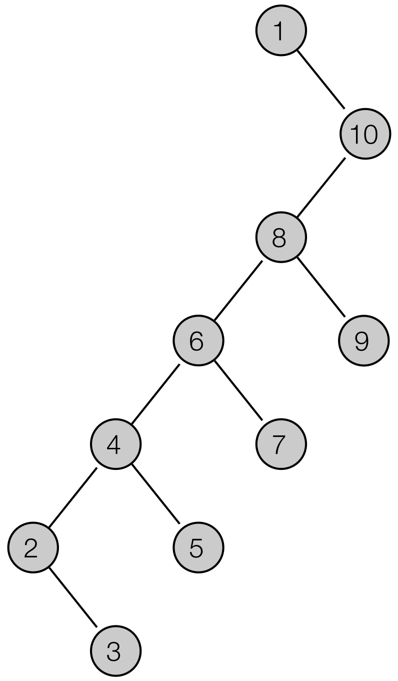
- Example. Splay(1)



right rotation at 1

Splay

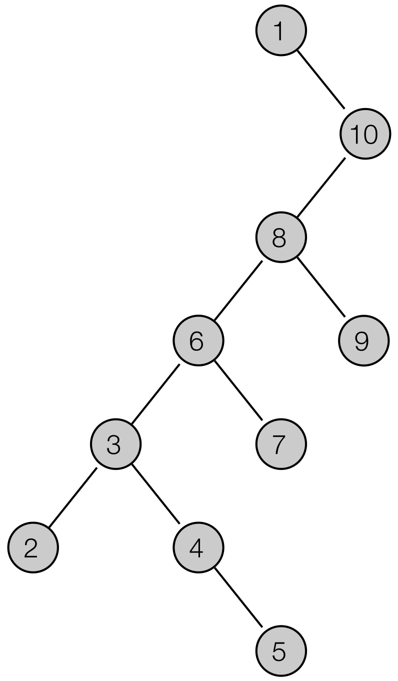
- Example. Splay(3)



zig-zag at 3

Splay

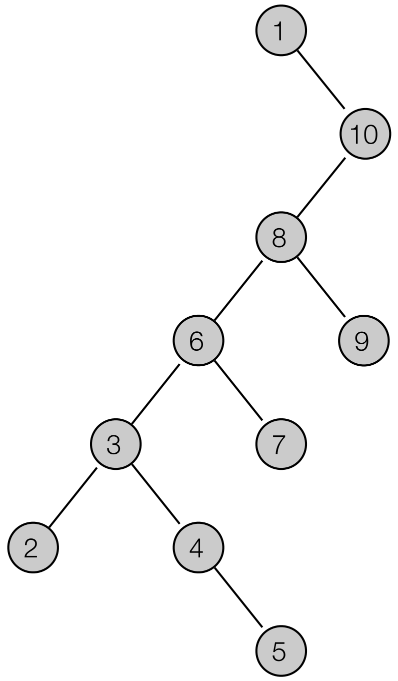
- Example. Splay(3)



zig-zag at 3

Splay

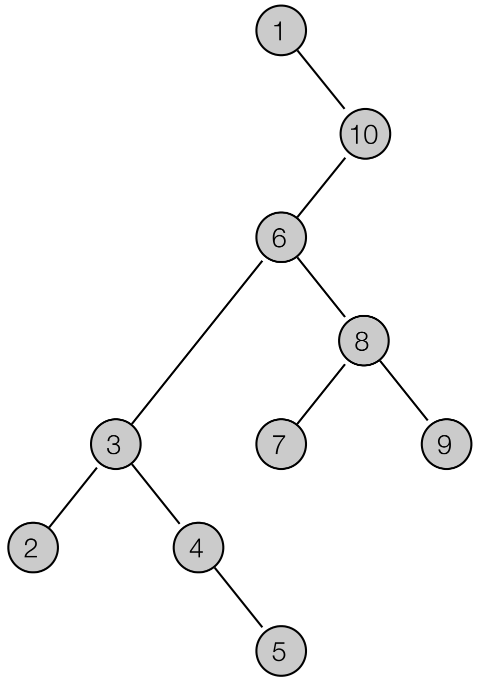
- Example. Splay(3)



roller-coaster at 3

Splay

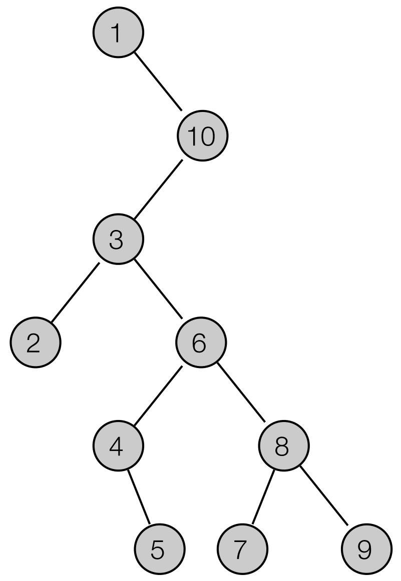
- Example. Splay(3)



roller-coaster at 3

Splay

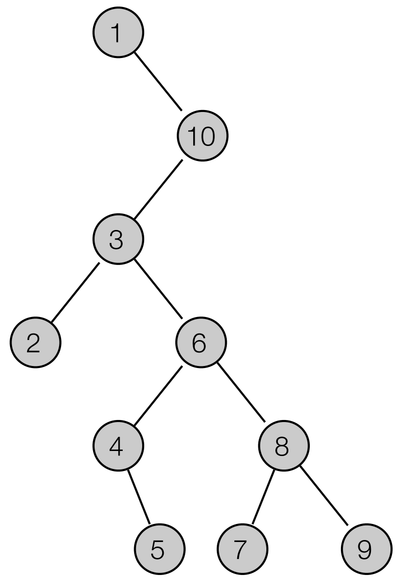
- Example. Splay(3)



roller-coaster at 3

Splay

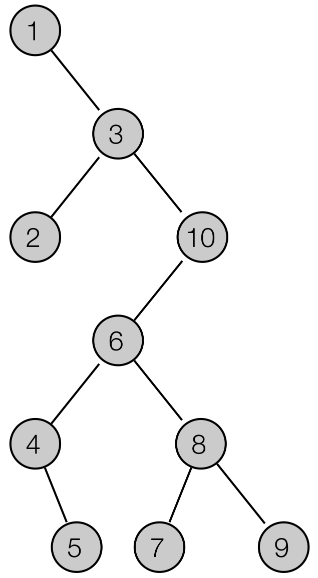
- Example. Splay(3)



zag-zig at 3

Splay

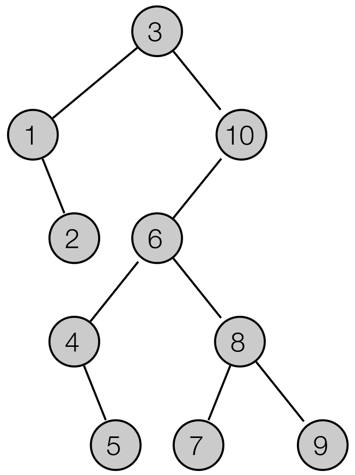
- Example. Splay(3)



zag-zig at 3

Splay

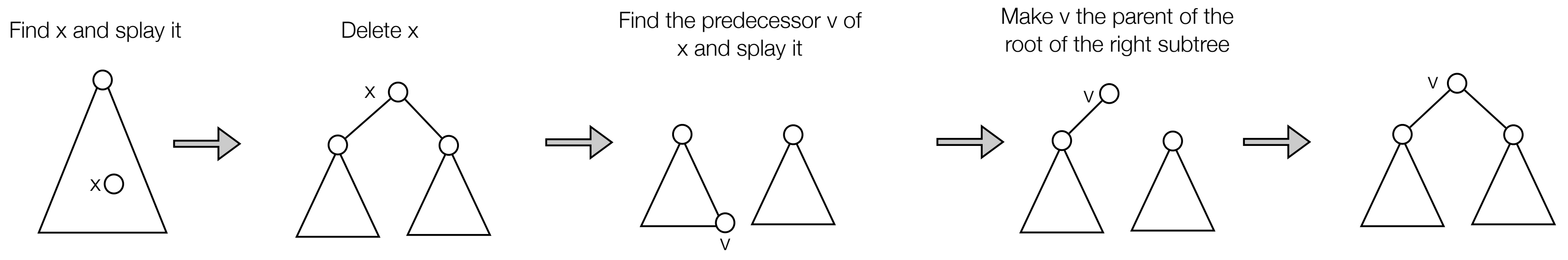
- Example. Splay(3)



zag-zig at 3

Splay Trees

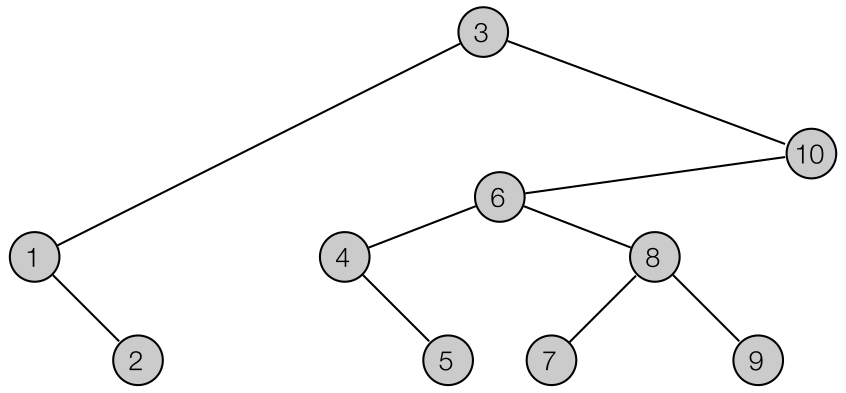
- Search(x). Find node containing key x (or predecessor/successor) using usual search algorithm. Splay found node.
- Insert(x). Insert node containing key x using algorithm for binary search trees. Splay inserted node.
- Delete(x). Find node x, splay it and delete it. Tree now divided in two subtrees. Find node with largest key in left subtree, splay it and join it to the right subtree by making it the new root.



Find x and splay it
x
Delete x
x
Find the predecessor v of x and splay it
v
Make v the parent of the root of the right subtree
v
v

Deletion in Splay Trees

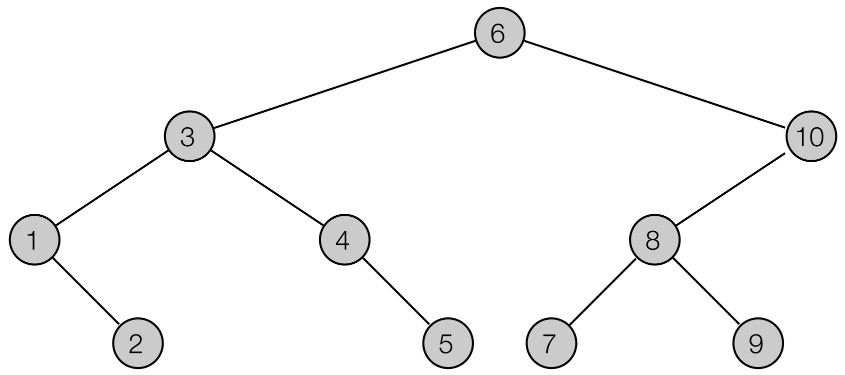
- Delete 6.



splay 6: zag-zig at 6

Deletion in Splay Trees

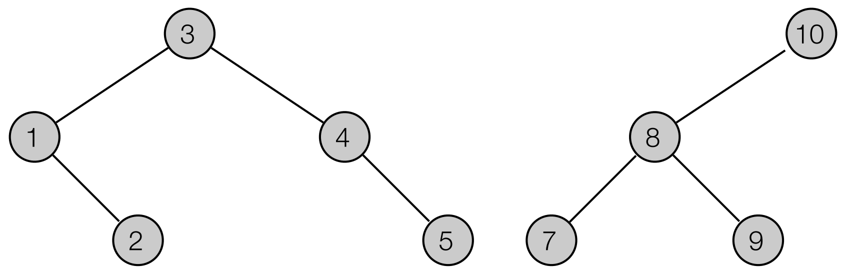
- Delete 6.



delete 6

splay 5

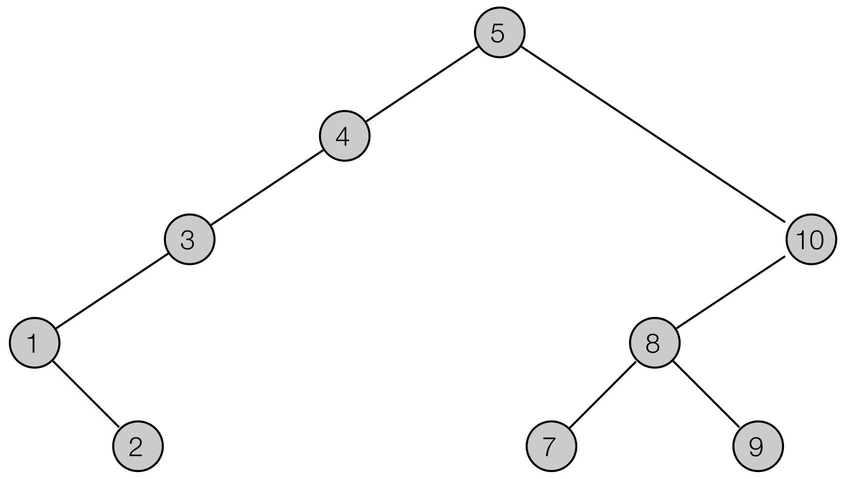
- Delete 6.



delete 6

Deletion in Splay Trees

- Delete 6.



connect

Analysis of splay trees

- Amortized cost of a search, insert, or delete operation is O(log n).
- All costs bounded by splay.

Analysis of splay trees

- Rank of a node.
    - size(v) = #nodes in subtree of v
    - $\mathrm{rank}(v) = \lfloor \lg \mathrm{size}(v) \rfloor$
- Potential function.

$$\Phi = \sum_v \mathrm{rank}(v) = \sum_v \lfloor \lg \mathrm{size}(v) \rfloor$$

- Rotation Lemma. The amortized cost of a single rotation at any node v is at most 1 + 3 rank'(v) - 3 rank(v), and the amortized cost of a double rotation at any node v is at most 3 rank'(v) - 3 rank(v).
- Splay Lemma. The amortized cost of a splay(v) is at most 1 + 3rank'(v) - 3 rank(v).

Splay Lemma Proof

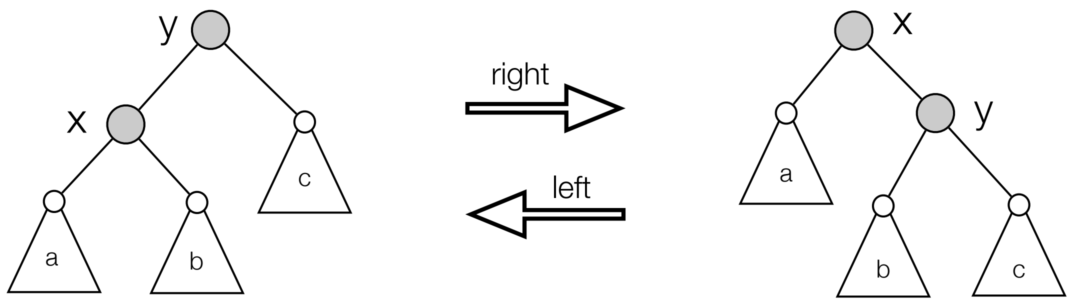
- Rotation Lemma. The amortized cost of a single rotation at any node v is at most 1 + 3 rank'(v) - 3 rank(v), and the amortized cost of a double rotation at any node v is at most 3 rank'(v) - 3 rank(v).
- Splay Lemma. The amortized cost of a splay(v) is at most 1 + 3rank'(v) - 3 rank(v).
- Proof.
  - Assume we have k rotations.
  - Only last one can be a single rotation.

$$\sum_{i=0}^{k} \hat{c}_i \leq \sum_{i=1}^{k-1} \left(r_i(v) - r_{i-1}(v)\right) + (1 + r_k(v) - r_{k-1}(v)) = 1 + r_k(v) - r_0(v)m = O(\lg n)$$

where $r_i(v)$ is the rank of v after the ith rotation.

Rotation Lemma

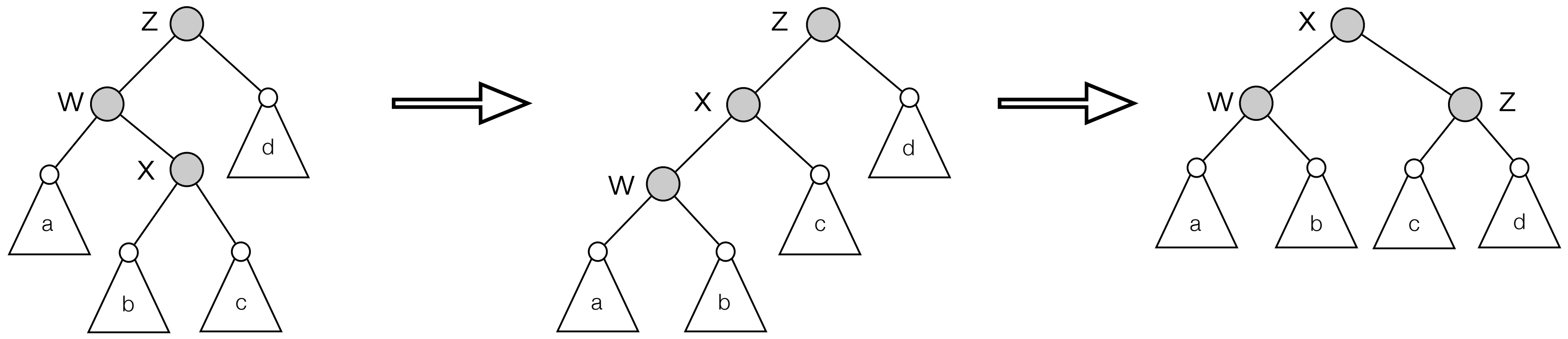
- Proof of rotation lemma: Single rotation.
  - Actual cost: 1
  - Change in potential:
    - Only x and y can change rank.
    - Change in potential at most r'(x) - r(x).
  - Amortized cost ≤ 1 + r'(x) - r(x) ≤ 1 + 3r'(x) - 3r(x).



right rotation at x (and left rotation at y)

Rotation Lemma

- Proof of rotation lemma: zig-zag.
  - Actual cost: 2
  - Change in potential:
    - Only x, w and z can change rank.
    - Change in potential at most 2r'(x) - 2r(x) - 2.
  - Amortized cost: ≤ 2 + 2r'(x) - 2r(x) - 2 ≤ 2r'(x) - 2r(x) ≤ 3r'(x) - 3r(x).



zig-zag at x