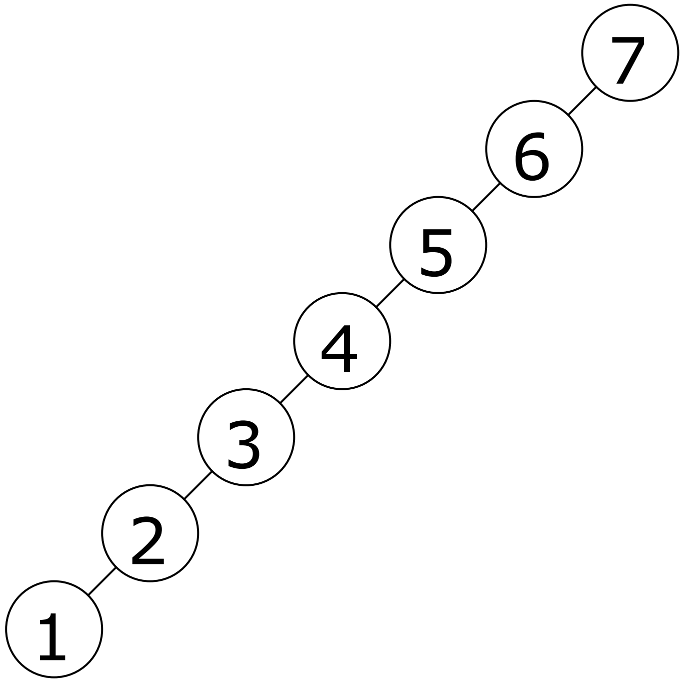
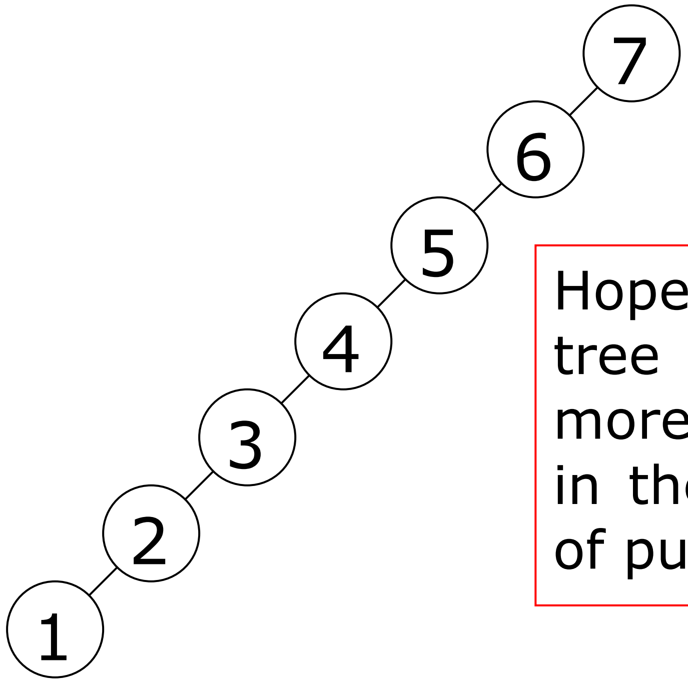
Splay Tree

Consider the following binary search tree

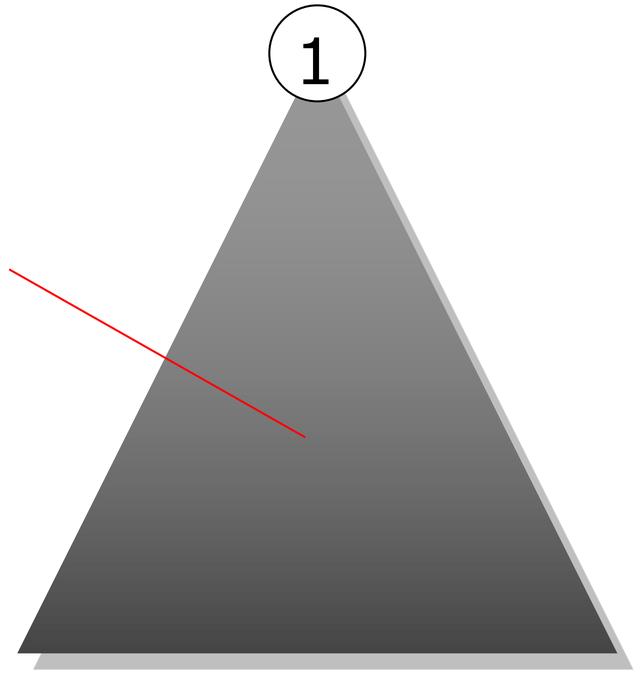


Splay Tree

If node 1 is accessed, we want to push it to the root.

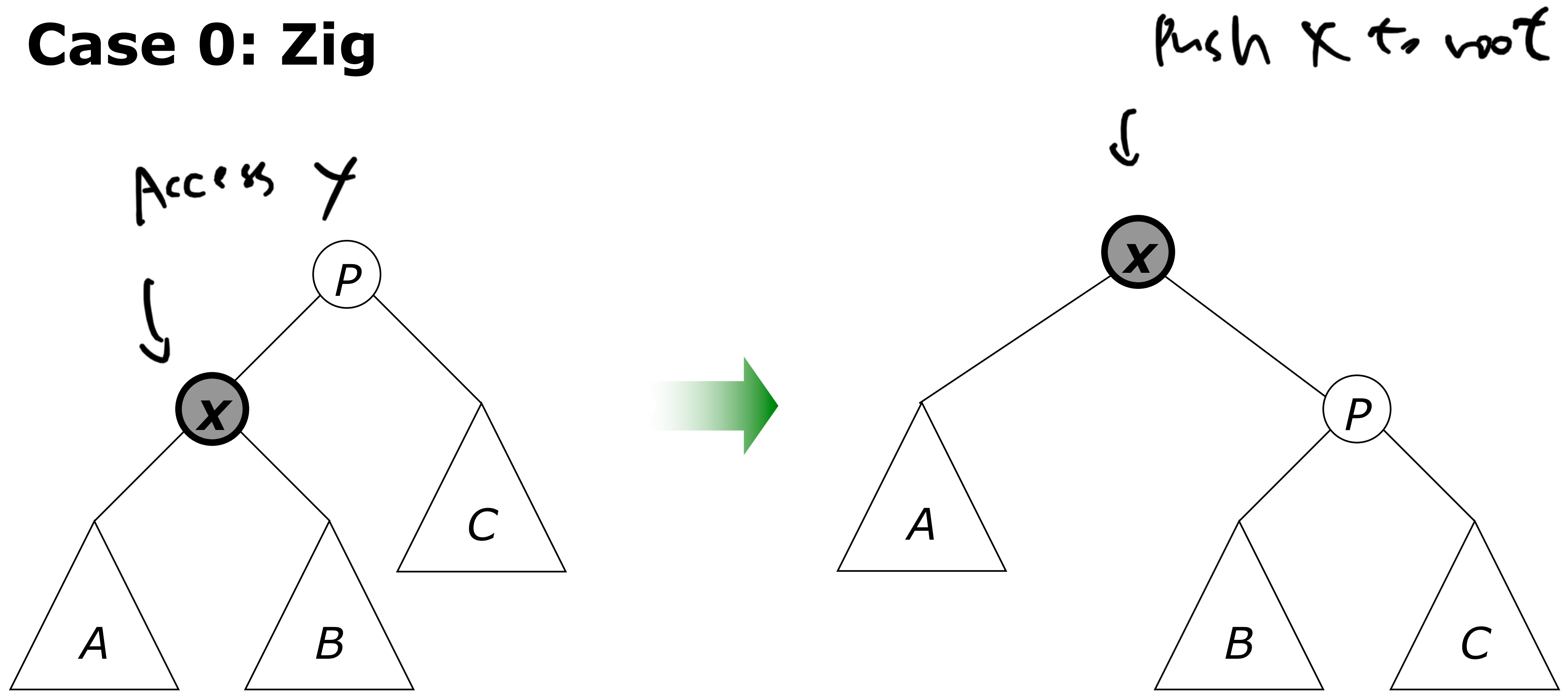


Hopefully, the tree becomes more balanced in the process of pushing.



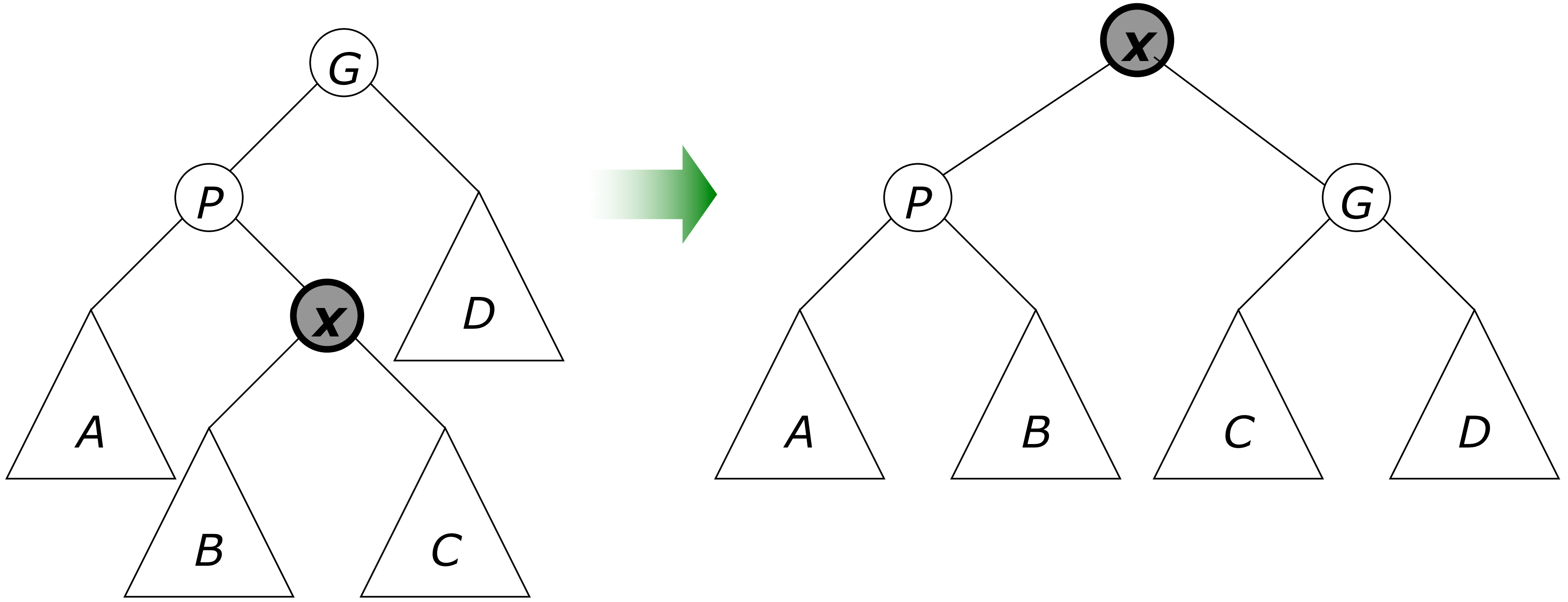
There are three cases of ‘pushing after access.’

Case 0: Zig



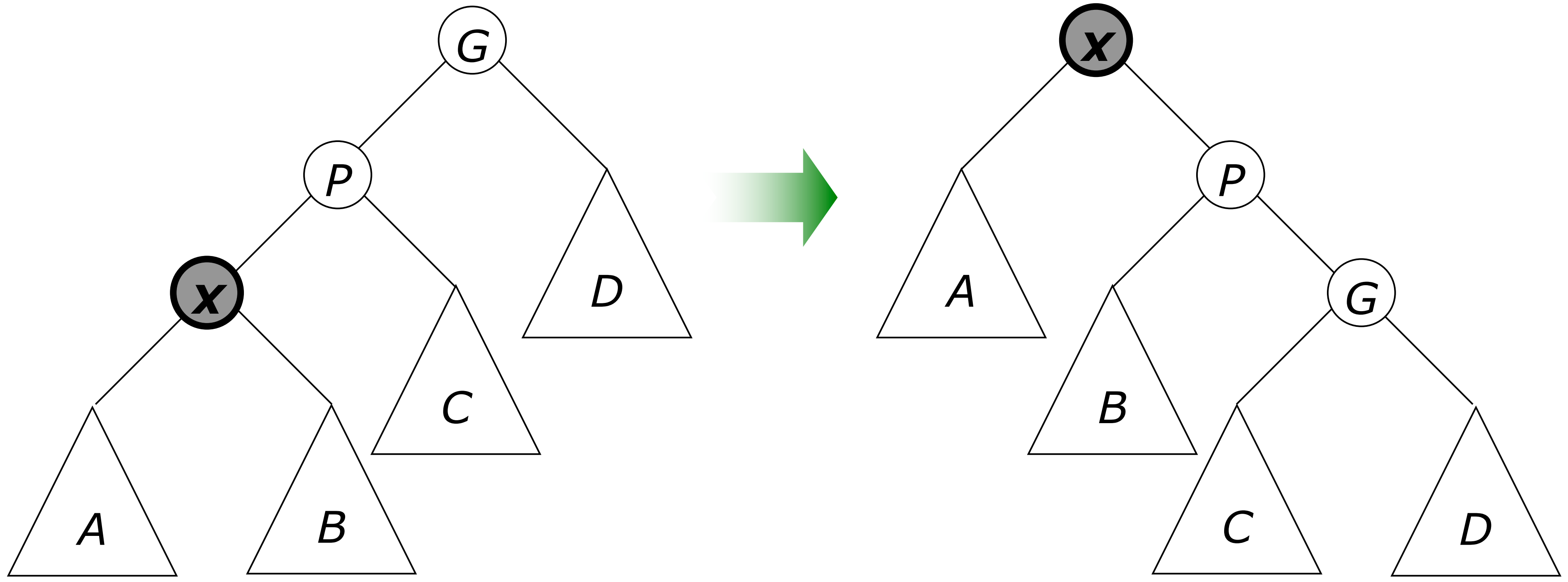
This is exactly a single rotation.

Case 1: Zig-zag

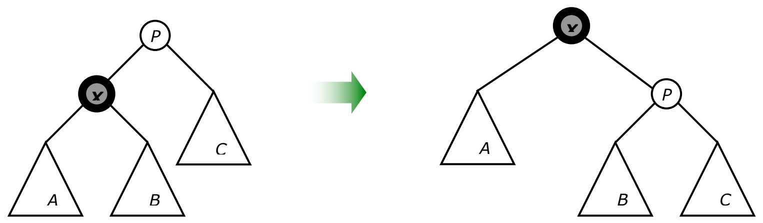


This is exactly a double rotation.

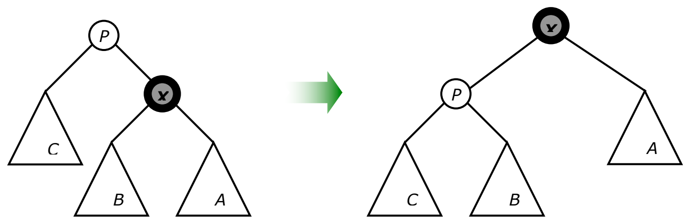
Case 2: Zig-zig



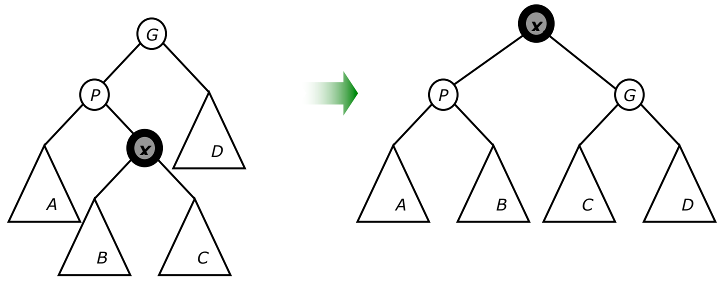
ZIG



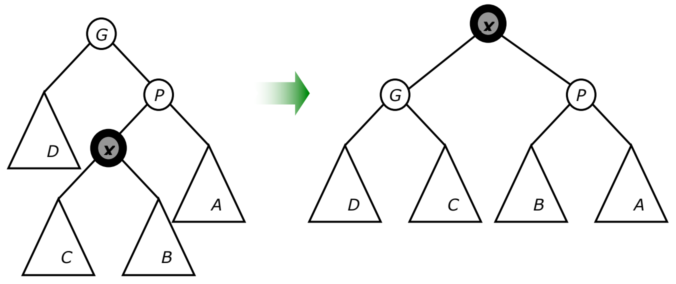
ZIG



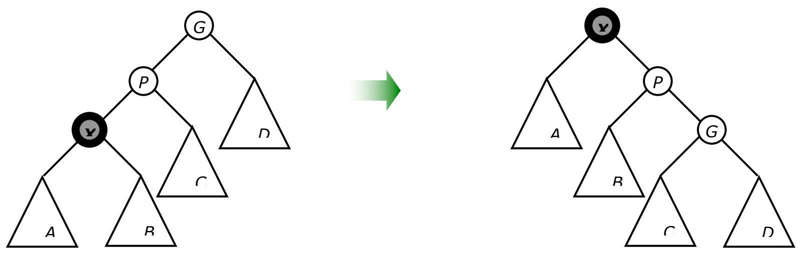
ZIG-ZAG



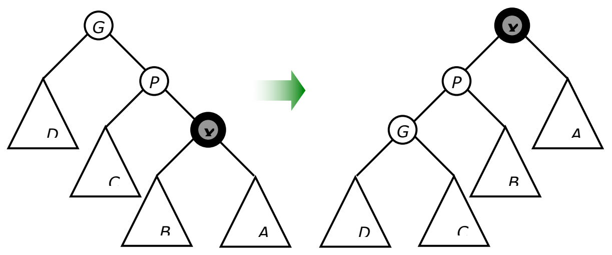
ZIG-ZAG



ZIG-ZIG

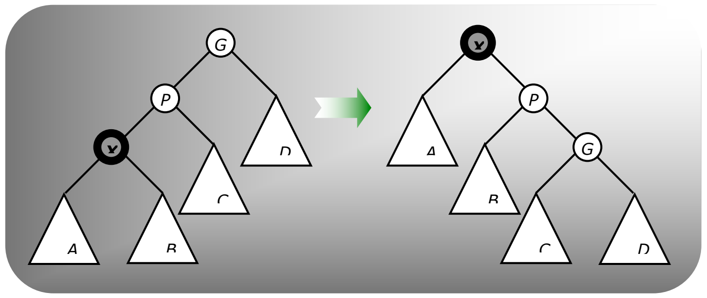


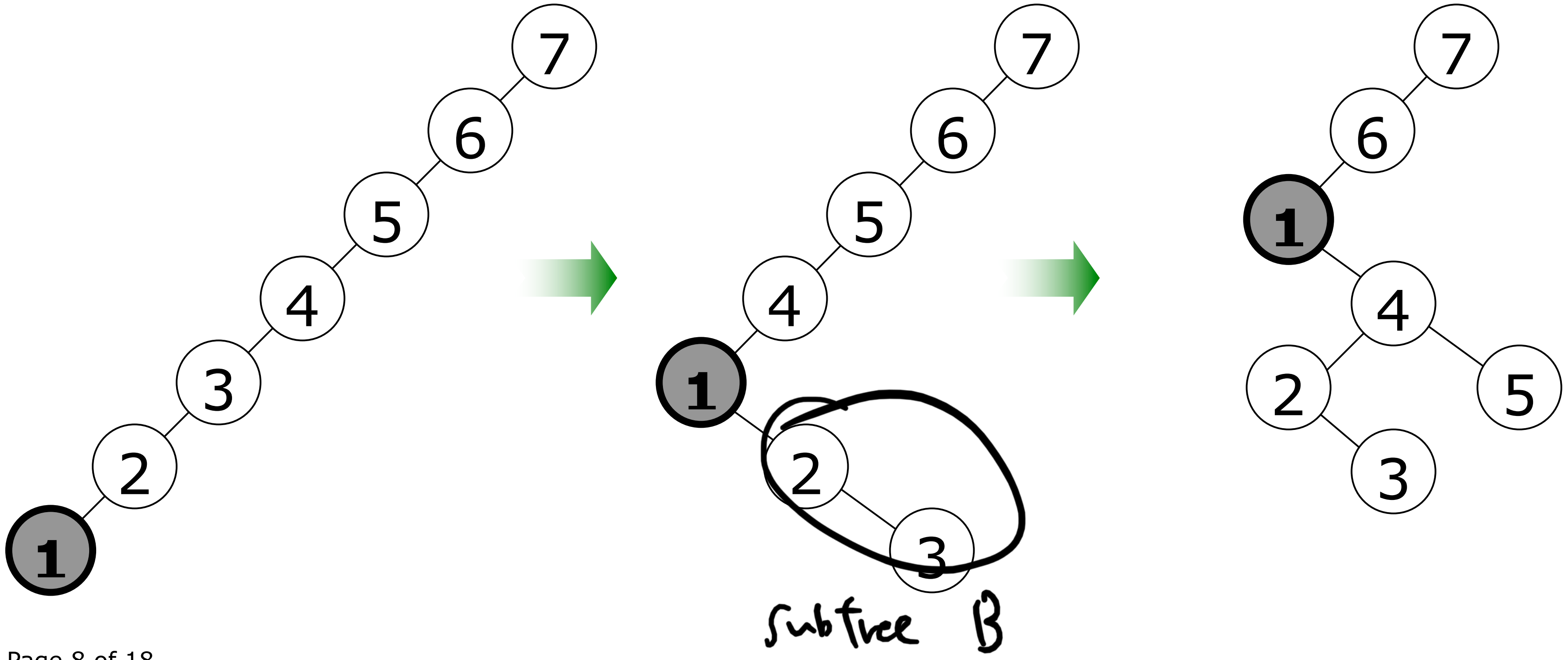
ZIG-ZIG



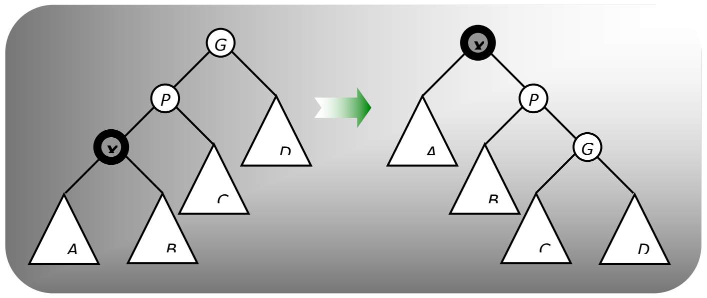
zig · zig

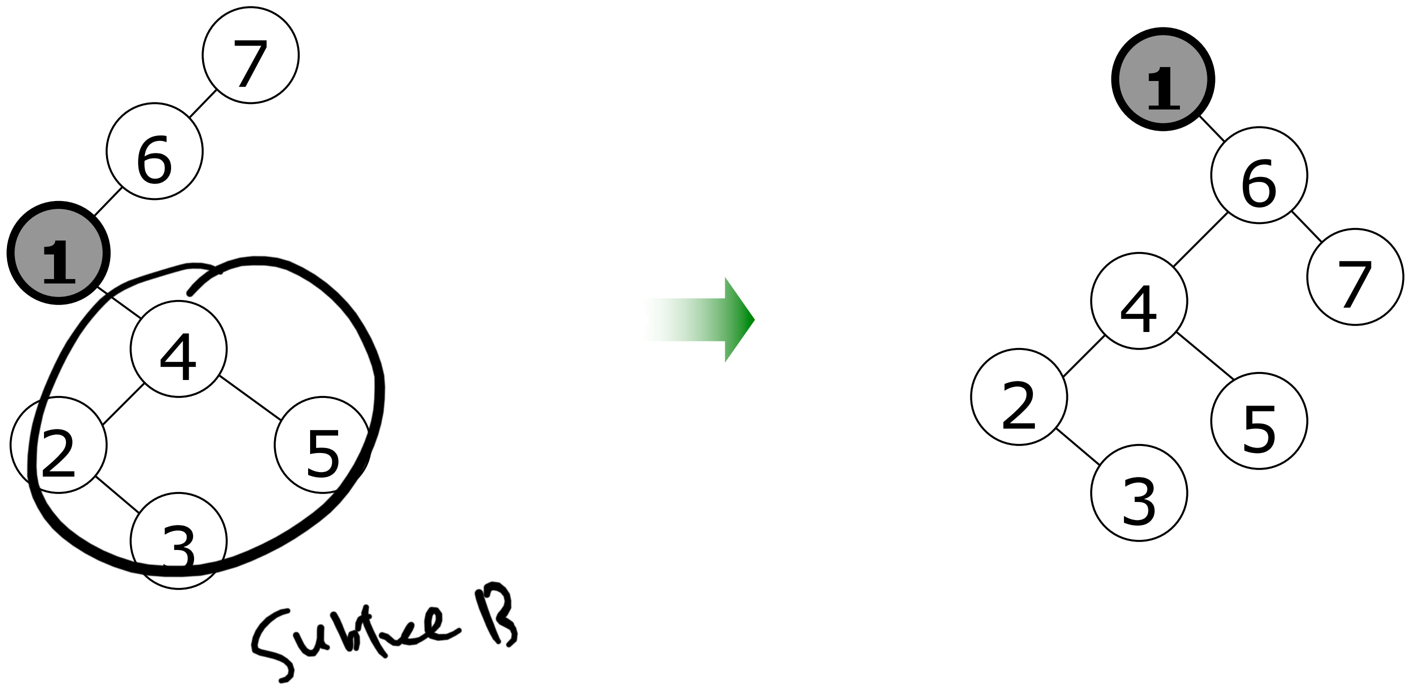
Splaying at Node 1

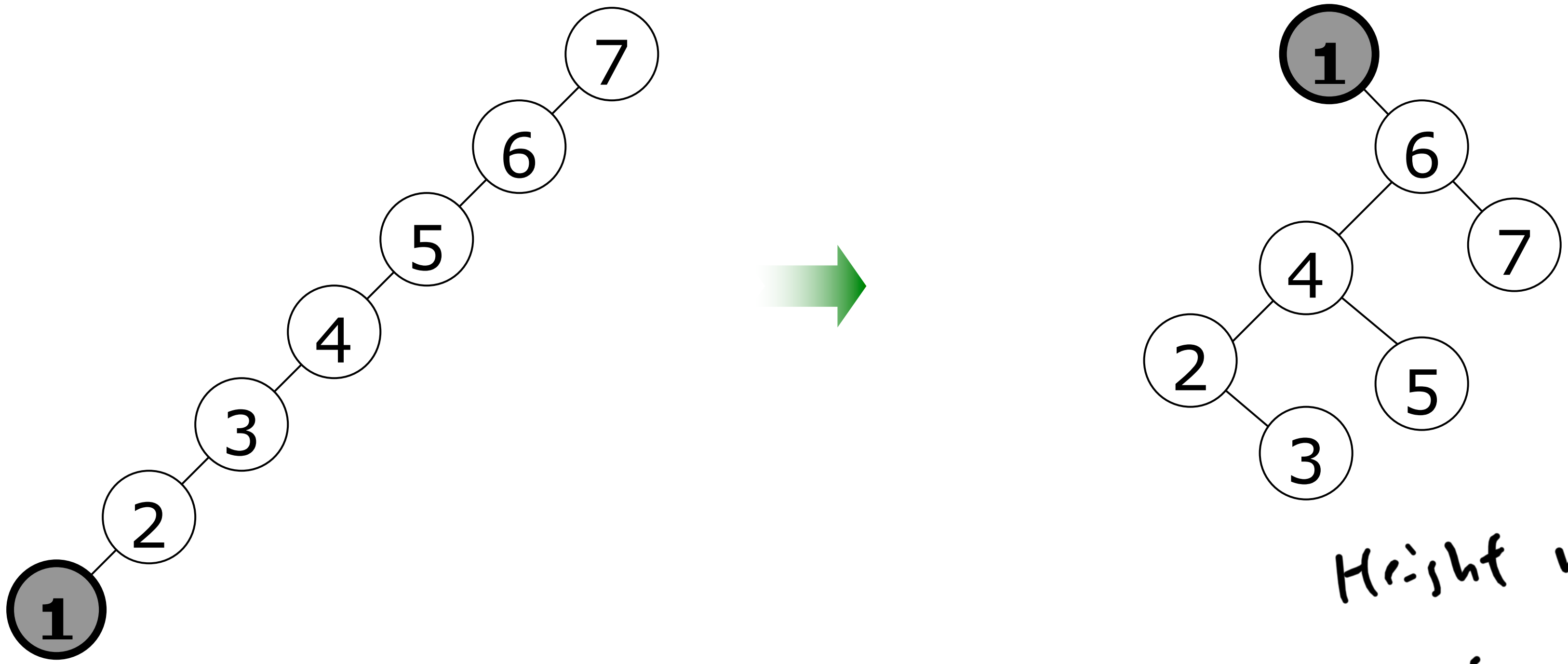




Splaying at Node 1



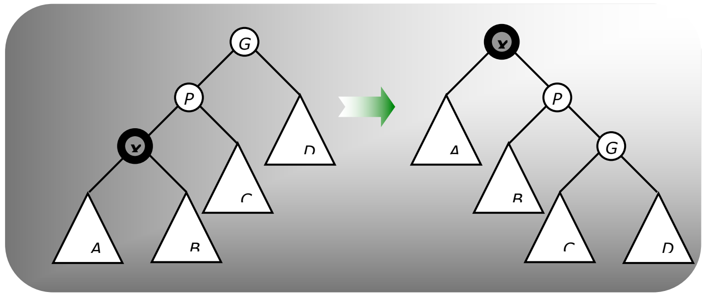


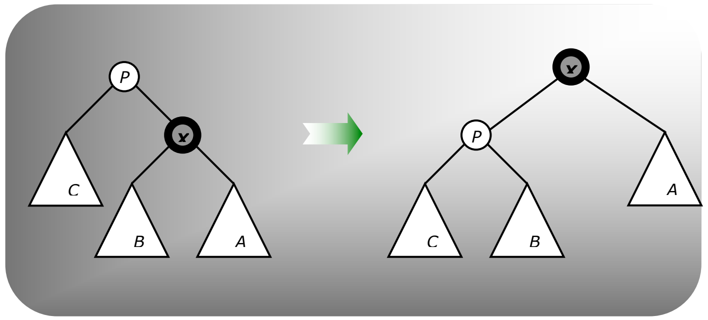


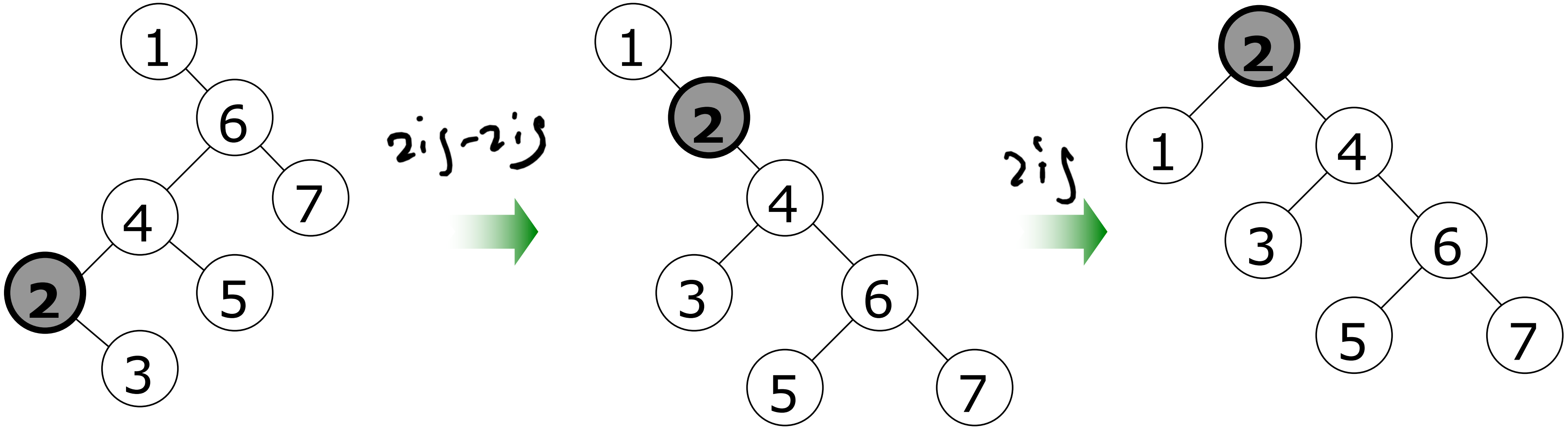
Height reduced
about half

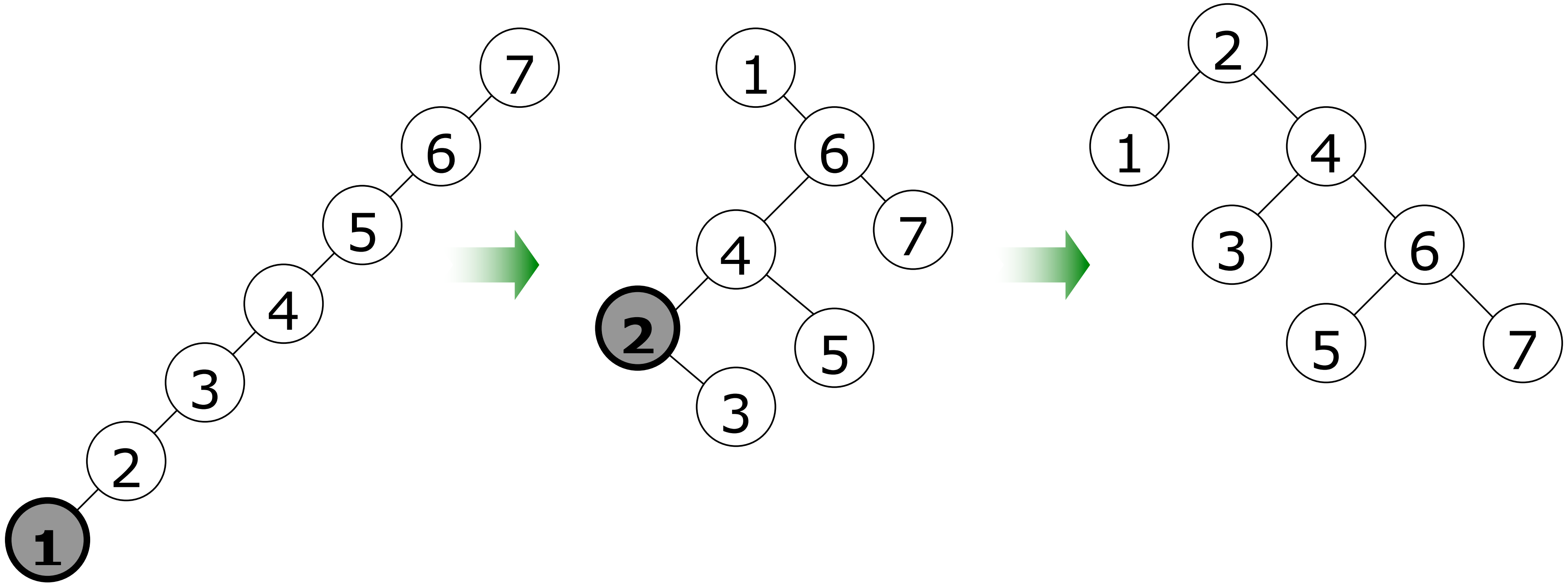
Splaying not only moves the accessed node to the root, but also has the effect of roughly halving the depth of the most nodes on the access path.

Splaying at Node 2



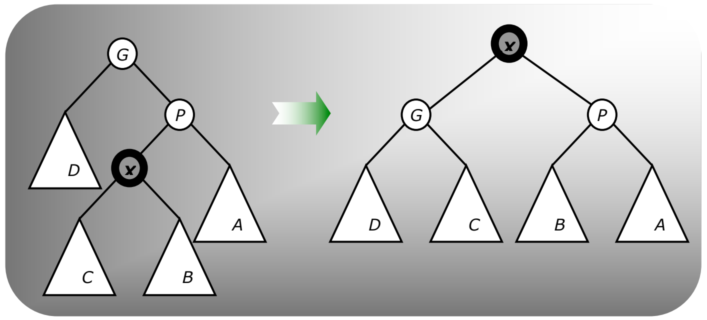


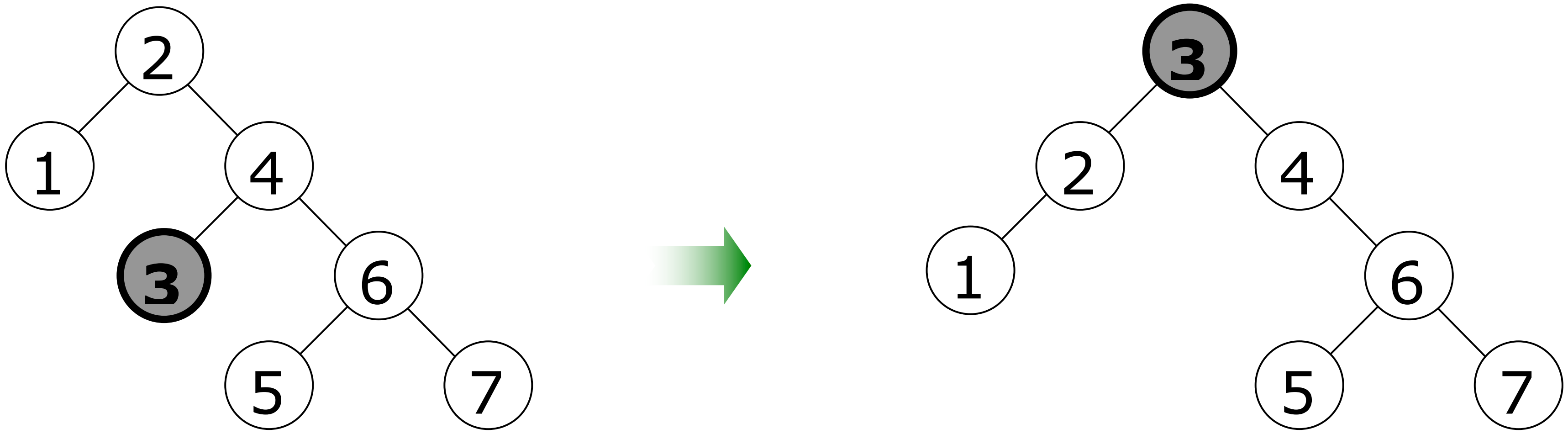




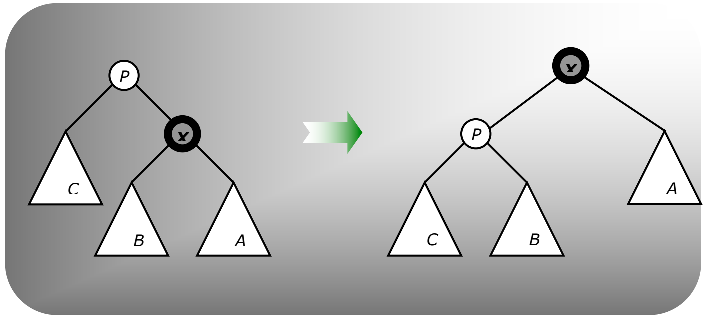
When access paths are long, the rotations tend to be good for future operations.

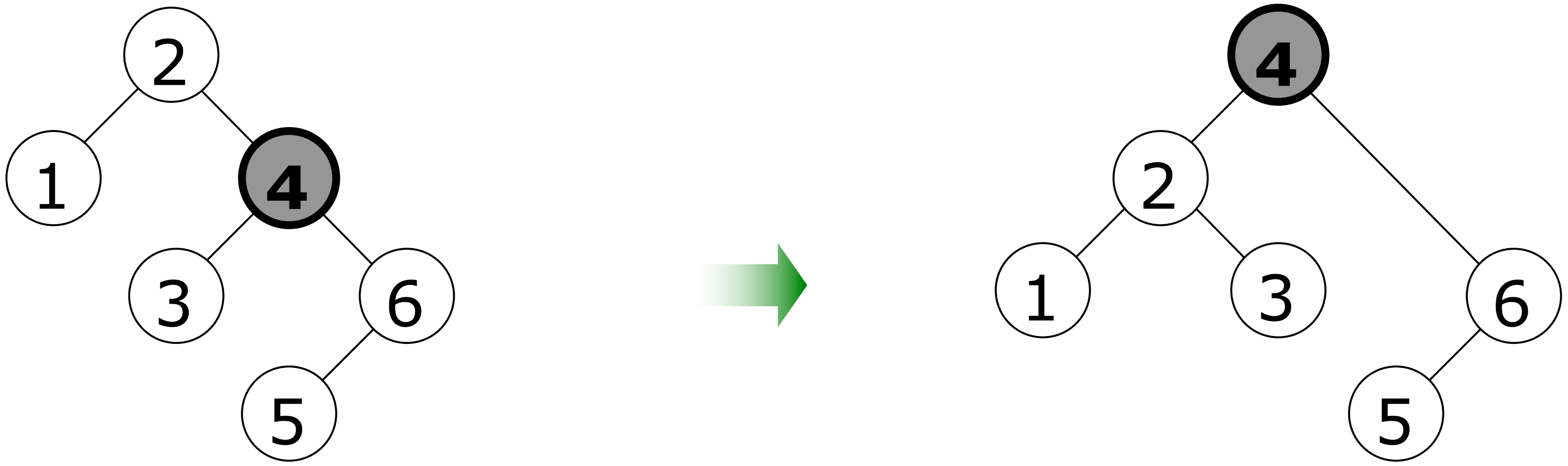
Splaying at Node 3



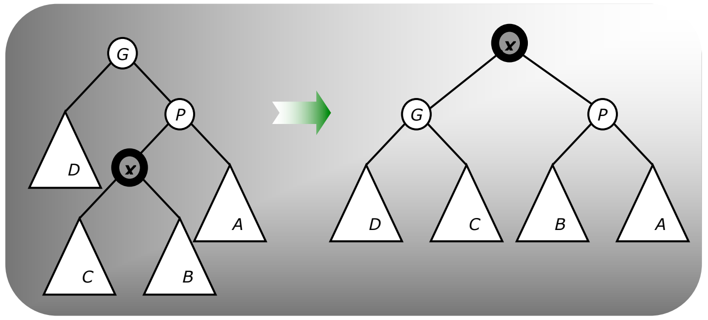


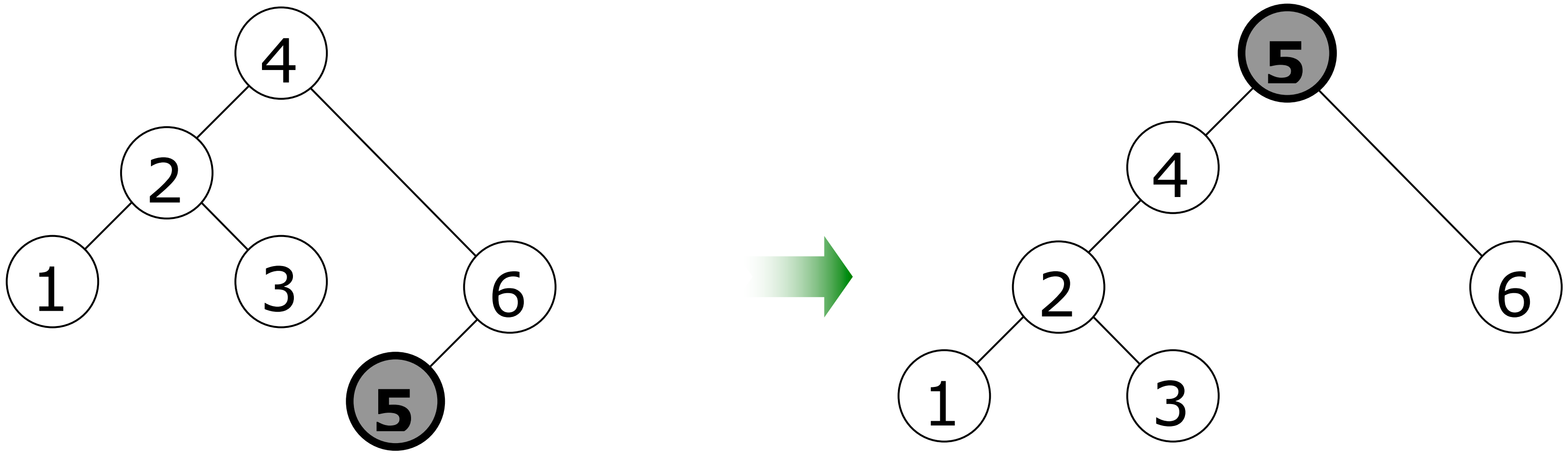
Splaying at Node 4



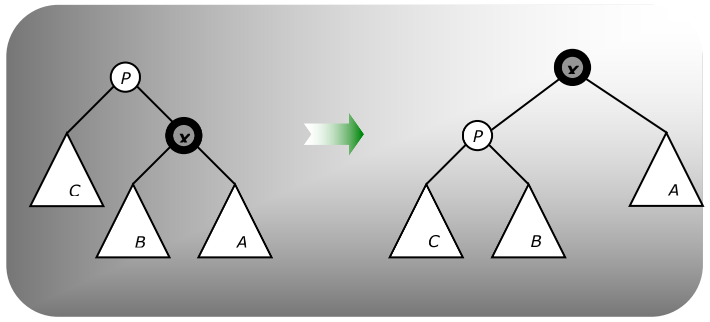


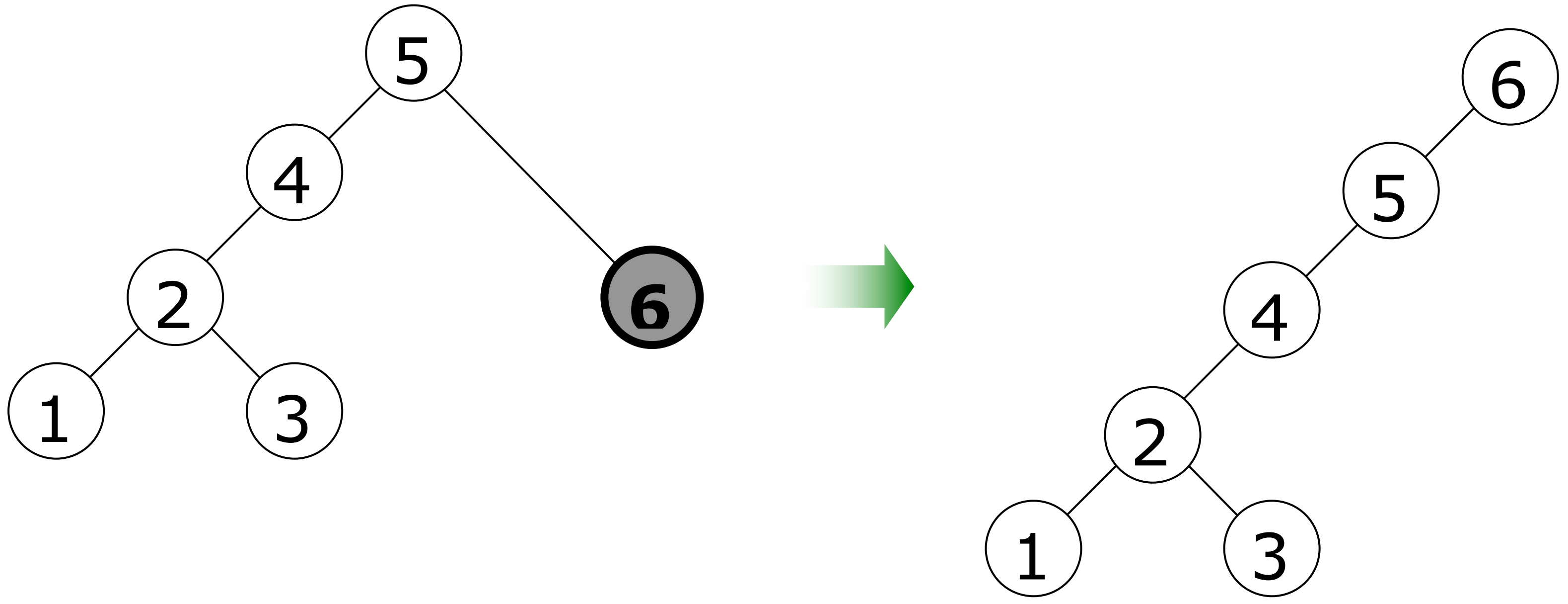
Splaying at Node 5

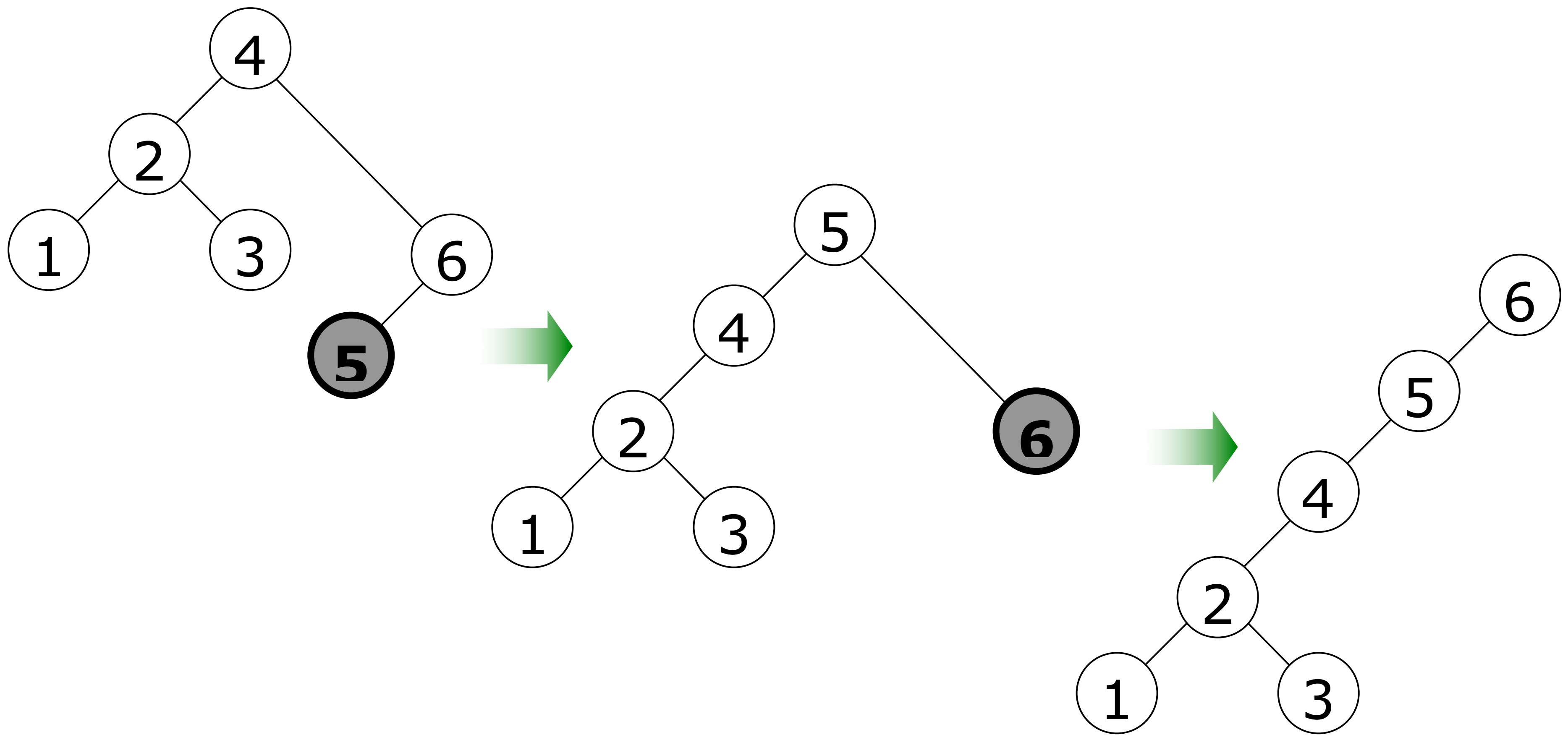




Splaying at Node 6







When accesses are cheap, the rotations are not as good and can be bad.

trie : coding doc

splay tree : searching

The analysis of splay tree is difficult, but splay trees are much simpler to program than AVL trees.