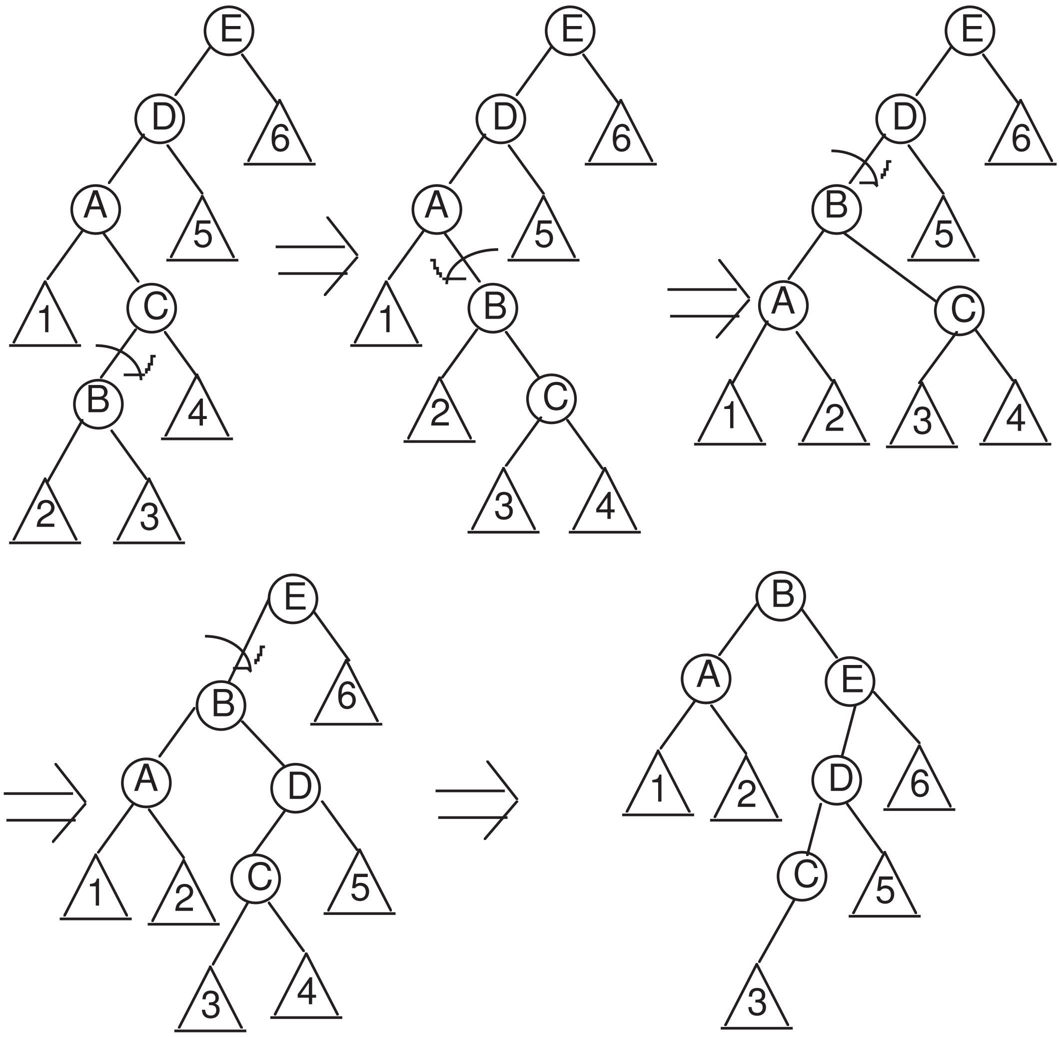
Splay

- Splay trees.
  - Optimal binary search tree does not allow insert/delete.
  - Balanced binary search tree is not optimal.
  - Remember "move-to-front" strategy for linked list (2 times optimal). Similar to "move-to-front", we do this for trees.

But: simple swap with parent can be disastrous. (If all elements have the same probability, search cost is $\Theta(\sqrt{n})$, if we use swaps.)

Try some sort of <u>"move to root"</u> approach: Swap with parent till accessed element is at root.

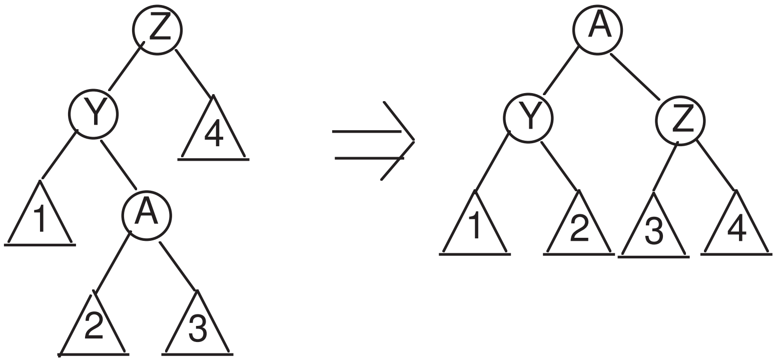
Example. Search (B)



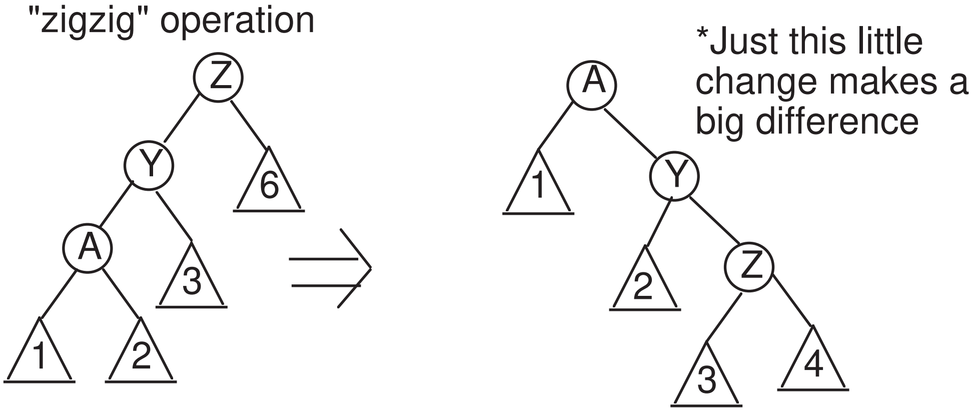
- Independ. problem ⟶ Average Cost ≤ (2 ln 2)*Opt.
- ∃ bad sequences: 1,2,...,n,1,2,...,n,1,...O(n)/Search

A slight change solves this problem. Splaying.

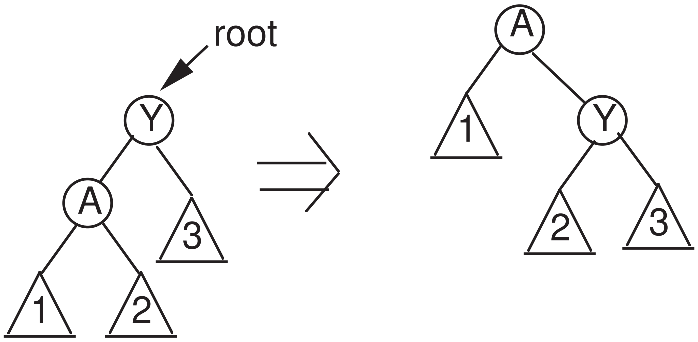
To move A up: <u>"zigzag"</u> operation



"zigzig" operation



"zig" operation



Splaying A $\equiv$ Repeat zigzag, or zigzig as long as possible, by the end do a zig if needed.

Insert: Insert as leaf, then splay it.

Delete: Delete as standard BST, then splay deepest node found.

Search: Spay that element. If unsuccessful, splay the node we end up with.

Theorem. For any sequence S search requests ($S \geq n \log n$), search cost $\leq cS$opt, where opt is the cost on the optimal search tree for that particular sequence of requests.

Theorem. Any sequence of m dictionary operations on "splay tree" (initially empty) uses $O(m \log n)$ time. Here n is the max. number of nodes in the tree.