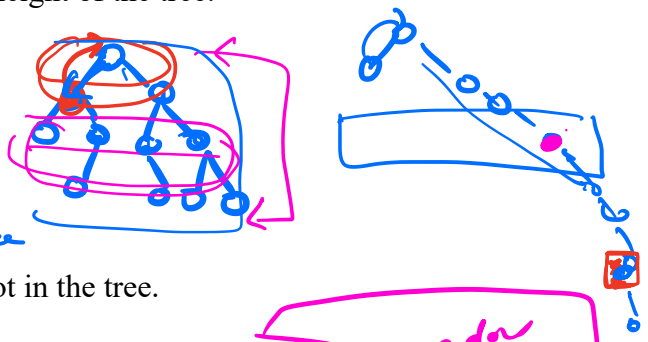
CS314H CSB Lecture 22: Splay Tree

1. Select one.
   - Any BST can become a splay tree once we start splaying operations.
   - We need to first convert a BST to a splay tree and then apply splaying.

2. (True/False)
   Splay operation always compresses the tree, i.e., shortens the height of the tree.

3. What will likely to happen in terms of the tree height?
   - Accessing a shallow node → not much change
   - Accessing a middle node → tend to make tree balanced
   - Accessing a deep node → tends to lengthen the tree



4. We can perform `insert(e)` in two ways. Let's assume e is not in the tree.

   1) Splay e' (where the last element accessed trying to find e).
   Let L and R be the left and right child of e' respectively.
   If e' < e, the new tree would be

```
      e
     / \
    e'  R
   /
  L
```

   If e' > e, the new tree would be

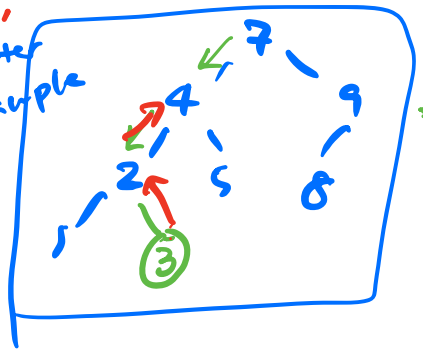
```
      e
     / \
    L   e'
         \
          R
```

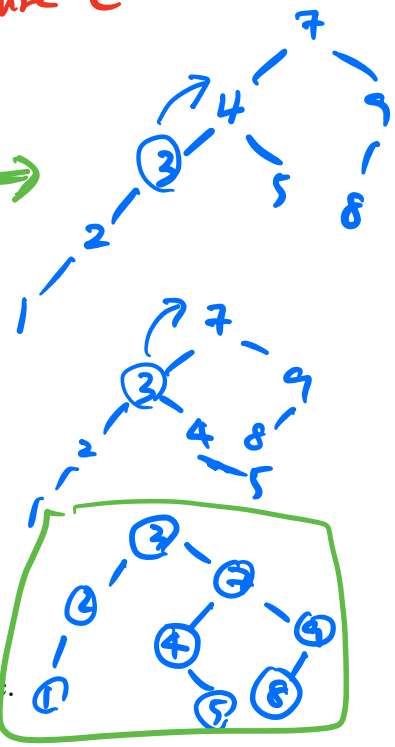
   manipulating pointers top down splay O(h) + O(1)

   2×O(h) + O(1)

   O(h) → traverse down the tree to figure e'

   O(h) → splay on e'





   counter example

   2) First insert e directly into the tree (regular BST way) and splay around e.

```
   e
  / \
 L   R, where L+R are the elements in the tree before the insertion.
```

   O(h) + O(h)

(True/False) The resulting trees using method 1) and 2) always have the same shape.

5. Which approach is better when we can perform splaying in a top-down manner?