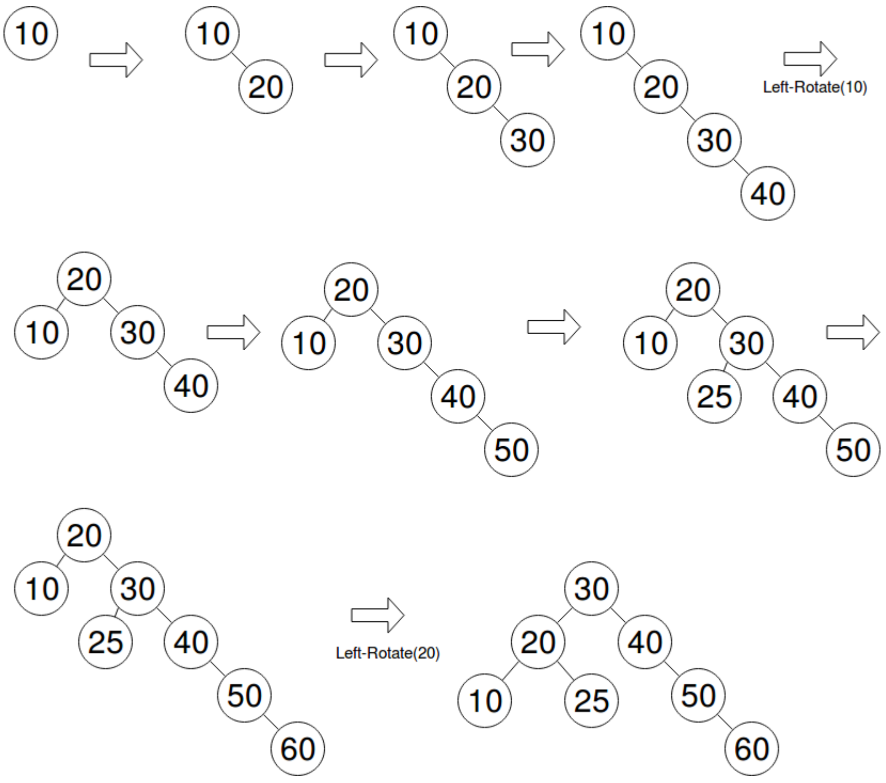
Lab 8 - AVL Trees

April 2, 2018

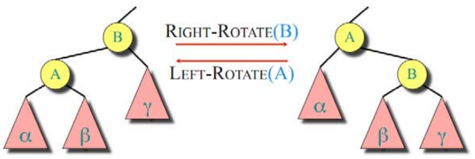
Question

AVL Trees are balanced binary search trees, which satisfy the AVL invariant property that for every node in the tree, $|h_L - h_R| \leq 1 =$ balance factor, where h_L and h_R are the node's left and right children respectively. In this question, you have to write methods for an AVL$_k$ tree, i.e., **an AVL tree with balance factor k**. For example, an AVL$_2$ tree ensures that the invariant $|\mathbf{h_L} - \mathbf{h_R}| \leq \mathbf{2}$ is satisfied for every node of the tree.

Here is an example of an AVL$_2$ tree in which the numbers 10, 20, 30, 40, 50, 25, 60 are inserted:



Rotations



Constraints

- $1 \leq k \leq 5$
- $q \leq 10^5$

Queries

- 1 x : Insert a node with key x into the tree.
- 2 : Print the number of left rotations and number of right rotations performed till now, in that order, as two space separated integers.
- 3 x : Search for the node with key x in the tree and print the number of times you choose to move along a left edge, and the number of times you move along a right edge, in that order, as two space separated integers (#left choices, #right choices). If x is not present in the tree, print ‘$-1 \quad -1$’.

Input Format

- The first line contains the balance factor, k.
- The second line contains an integer q, where q is the number of queries that will follow.
- Each of the next q lines contains a query.

Sample Test Case 1

Input:

```
1
12
1 10
1 20
2
1 30
2
1 40
1 50
1 25
2
3 10
3 50
3 25
```

Output:

```
0 0
1 0
3 1
2 0
0 2
1 1
```

Sample Test Case 2

Input:

```
2
14
1 10
1 20
1 30
2
1 40
2
1 50
2
3 50
1 25
1 60
2
3 60
3 25
```

Output:

```
0 0
1 0
1 0
0 3
2 0
0 3
1 1
```