Announcements

1. Exam 2 ends today! (M-W)

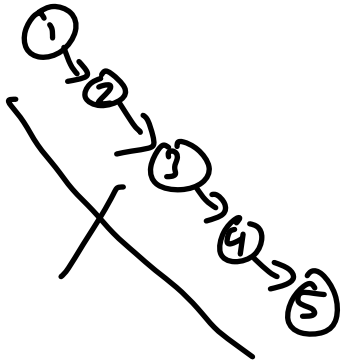
Join Code: **225**

Extra Credit Survey!

due next Wednesday, Goal: 70%, currently at 60%

Warm-Up Question: What happens with a binary tree if I insert the elements in ascending order? Ex. 1, 2, 3, 4, 5

search



Balanced Binary Search Trees (BBSTs)

Learning Objectives

1. Define Balance of a Node
2. Evaluate whether a tree is balanced or not
3. Know how to perform rotations in a tree

Implementation of Lambda Functions

[Capture](inputs){ body }

↳ taking variables from current program state

Example Function

```
1   int big;
2   std::cout << "How big is big? "; // say input is 10
3   std::cin >> big;
4
5   auto isbig = [big](int num) { return (num > big); };
6   std::vector<int> values = {3, 7, 10, 15, 2, 20, 8};
7   std::cout << "Numbers greater than " << big << ": ";
8   for (int v : values) {
9     if(isbig(v)) {
10        std::cout << v << " ";
11    }
12  }
13
14  std::cout << std::endl;
15
16 }
```

Motivation

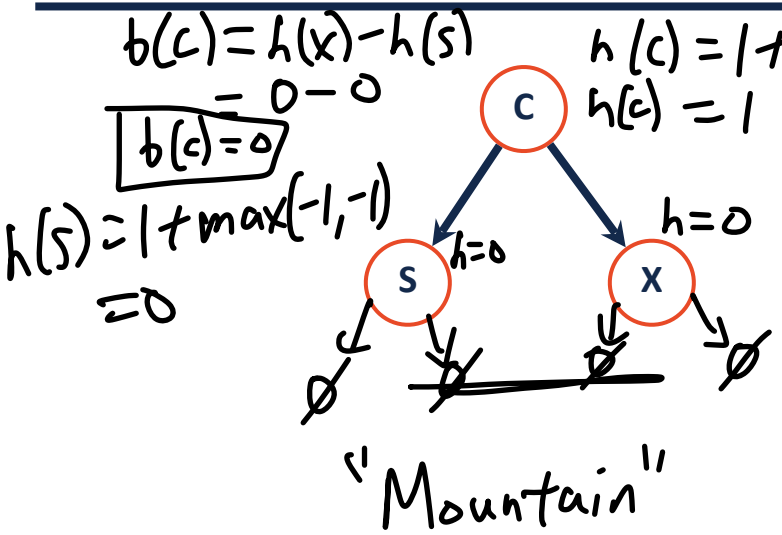
Binary Search Trees

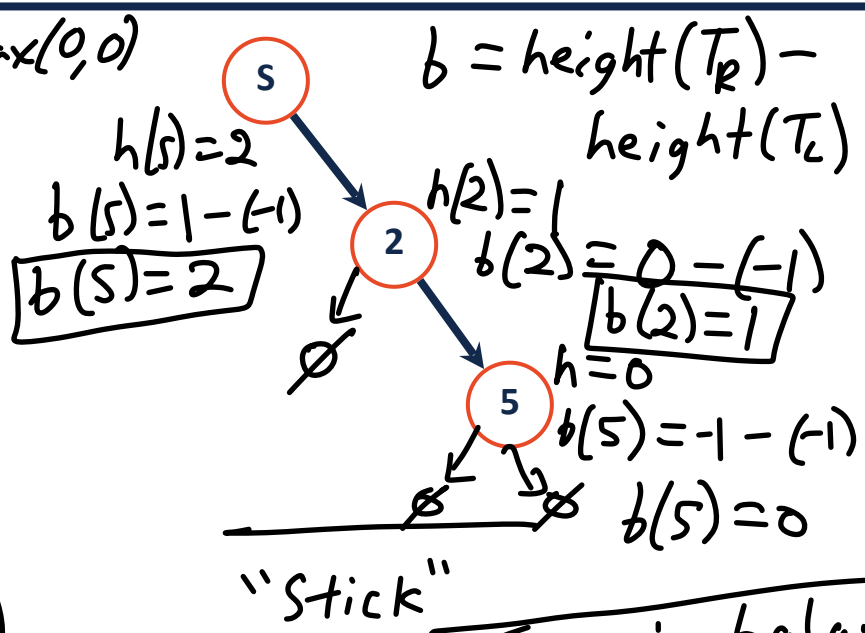
- Find — $O(h) - O(n)$
- Insert — $O(h) - O(n)$
- Remove — $O(h) - O(n)$

$O(\log n)$

↳ for perfect looking trees

Balanced





$h = 1 + \max(h(T_L), h(T_R))$

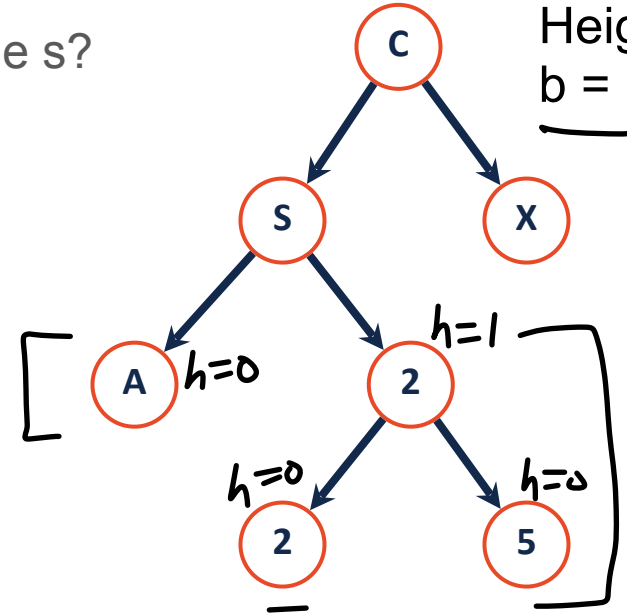
$h(\text{Null}) = -1$

Tree is balanced when $|b| \leq 1$

Balanced

What is the balance of node s?

$b(S) = 1$



Height balance:
$b = height(T_R) - height(T_L)$

Is this tree balanced?

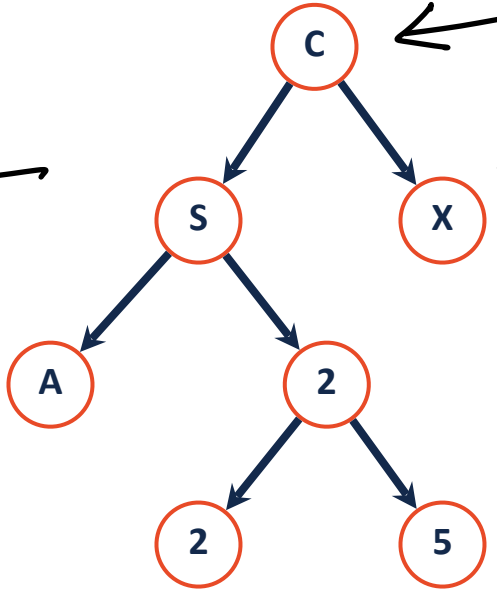
$|b| \leq 1$

Join Code: **225**

Tree Balanced

A tree is height balanced if: |b| <= 1

No!



Imbalance here

h(X) = 0

h(S) = 2

b(C) = h(X) − h(S)

= 0 − 2

= −2

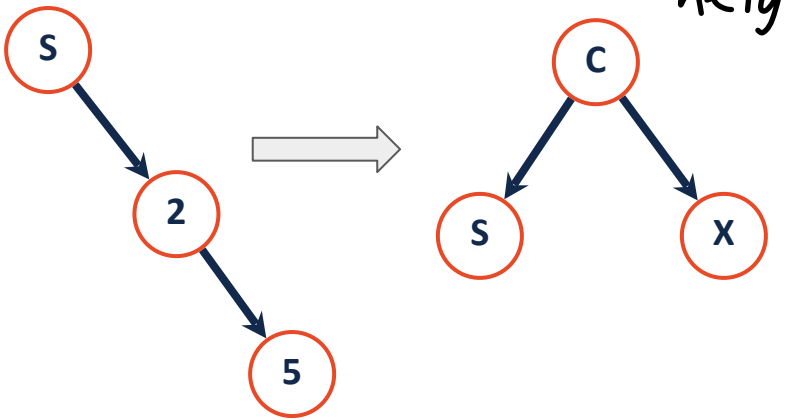
|−2| > 1

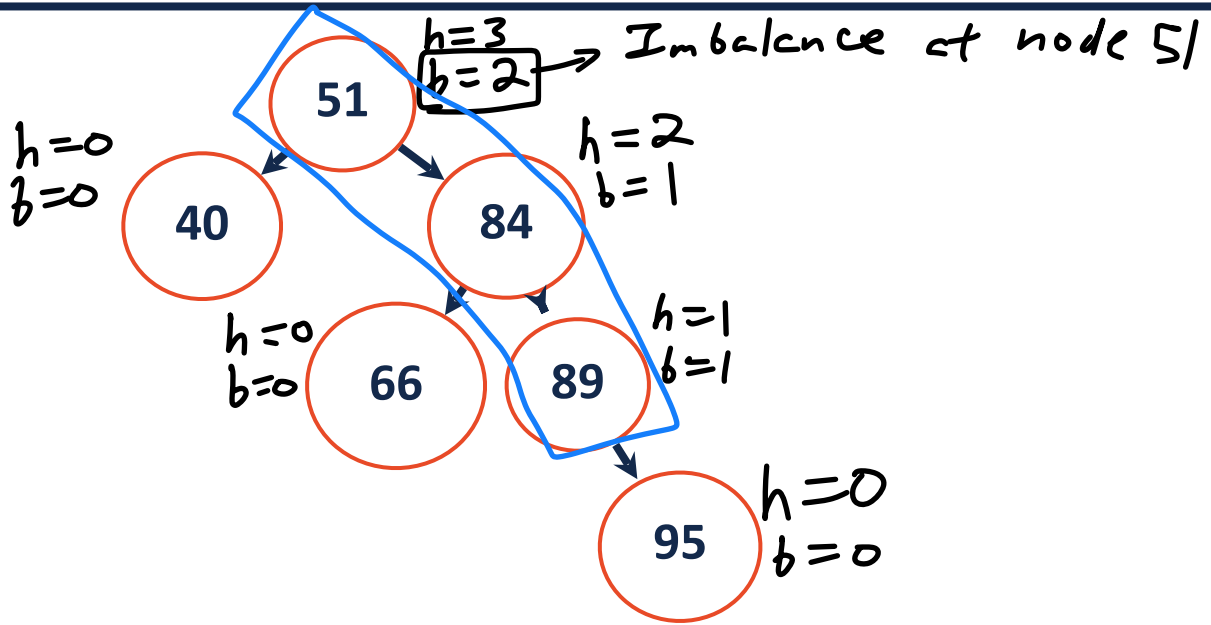
∴ Not balanced

Goal of Rotations

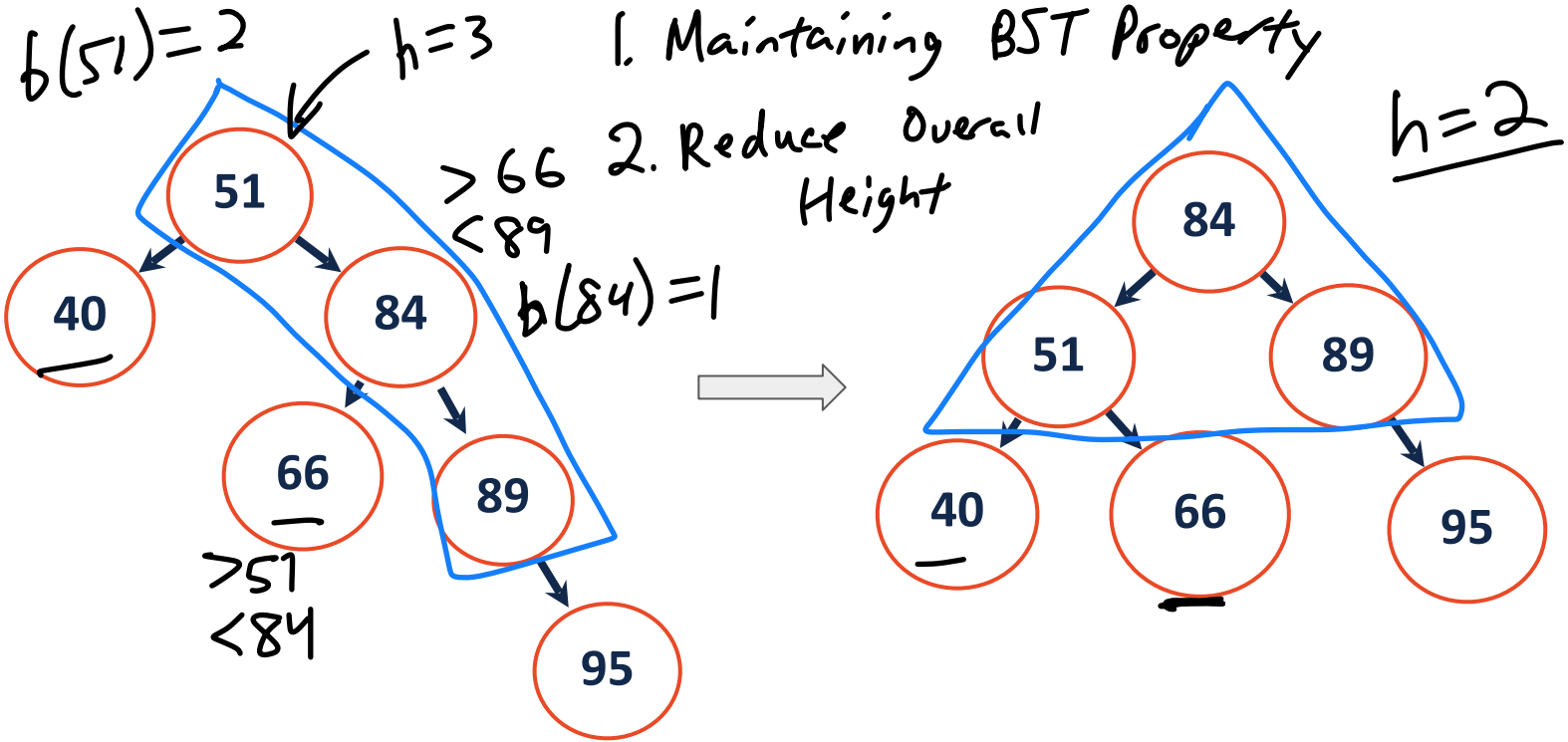
1. BST Property

2. Changes sticks into mountains – reducing overall height of my tree

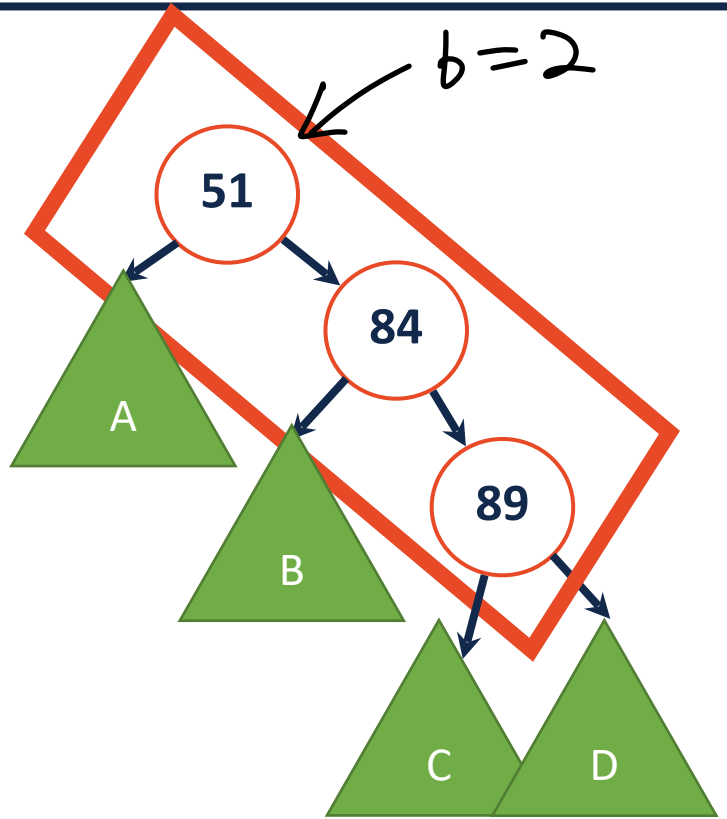



h=3
b=2
Imbalance at node 51
51
h=0
b=0
40
h=2
b=1
84
h=0
b=0
66
89
h=1
b=1
95
h=0
b=0

Left Rotation

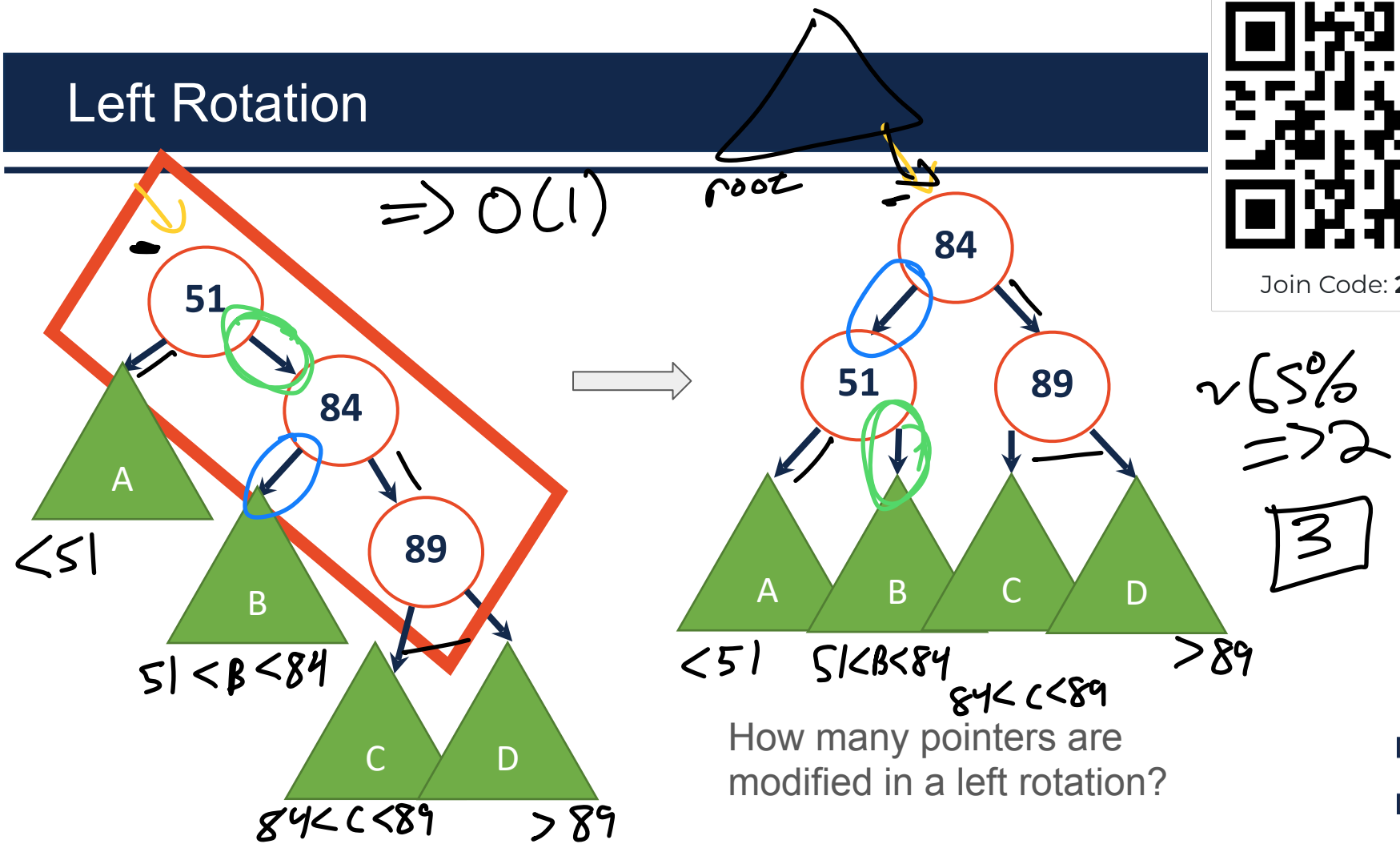


Left Rotation



△ - any balanced binary search tree

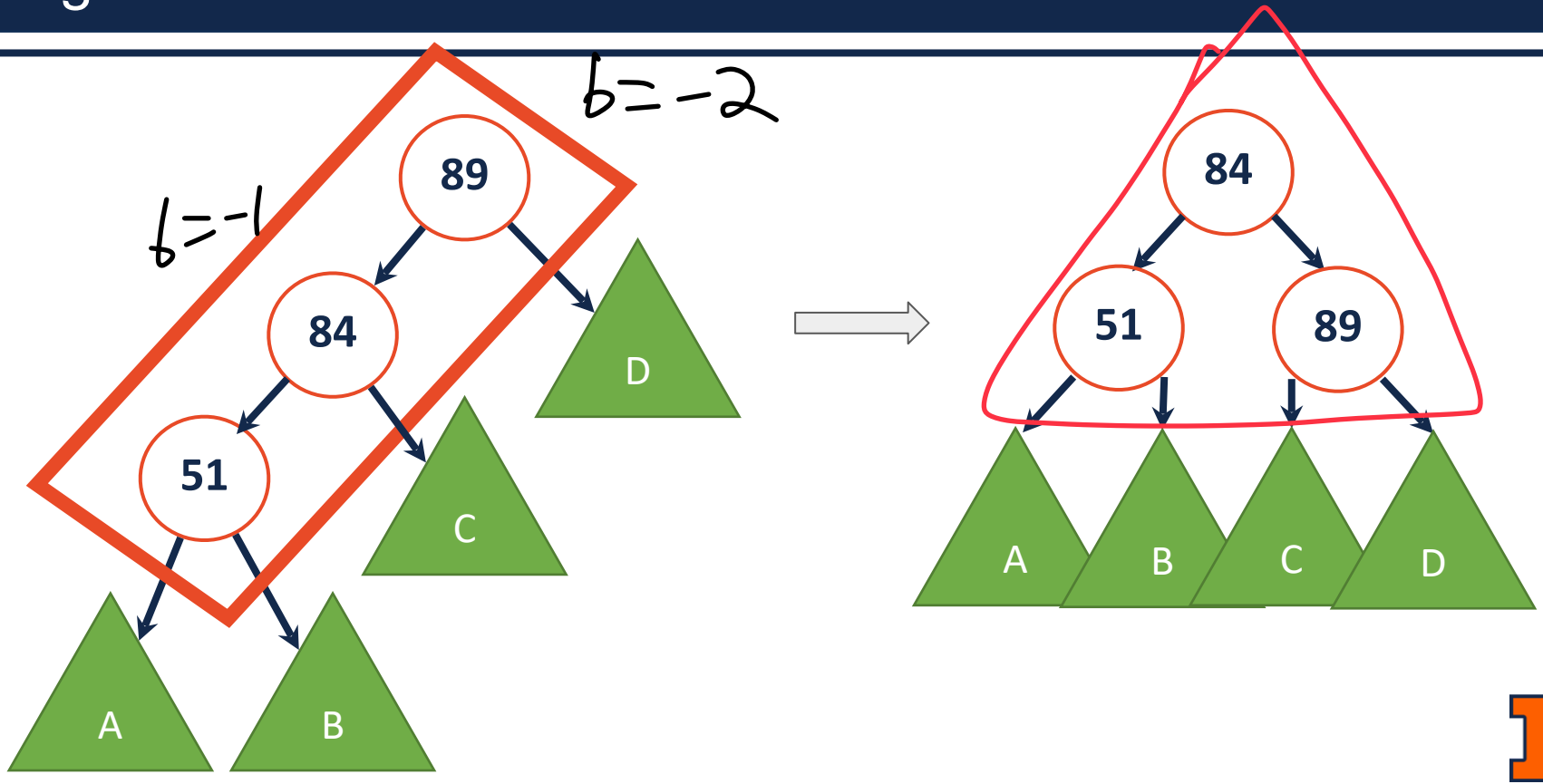
Left Rotation

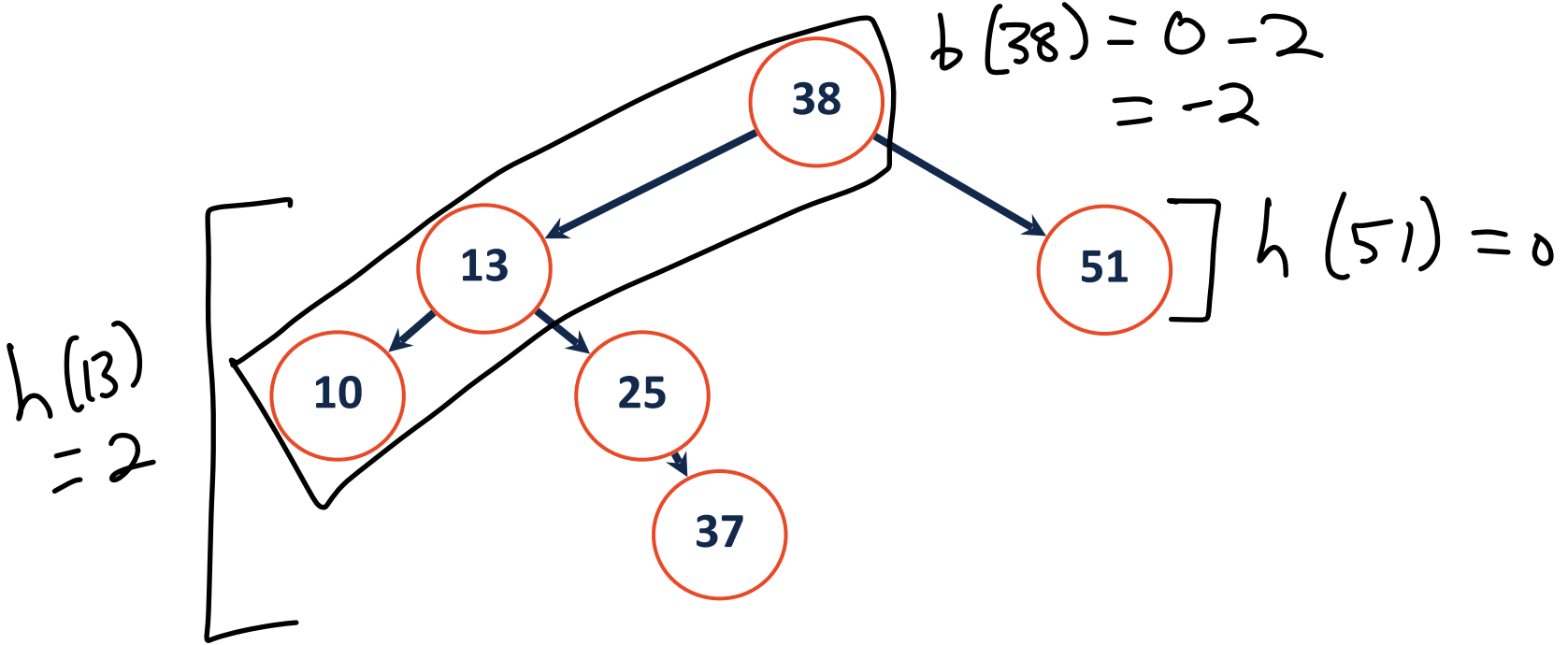


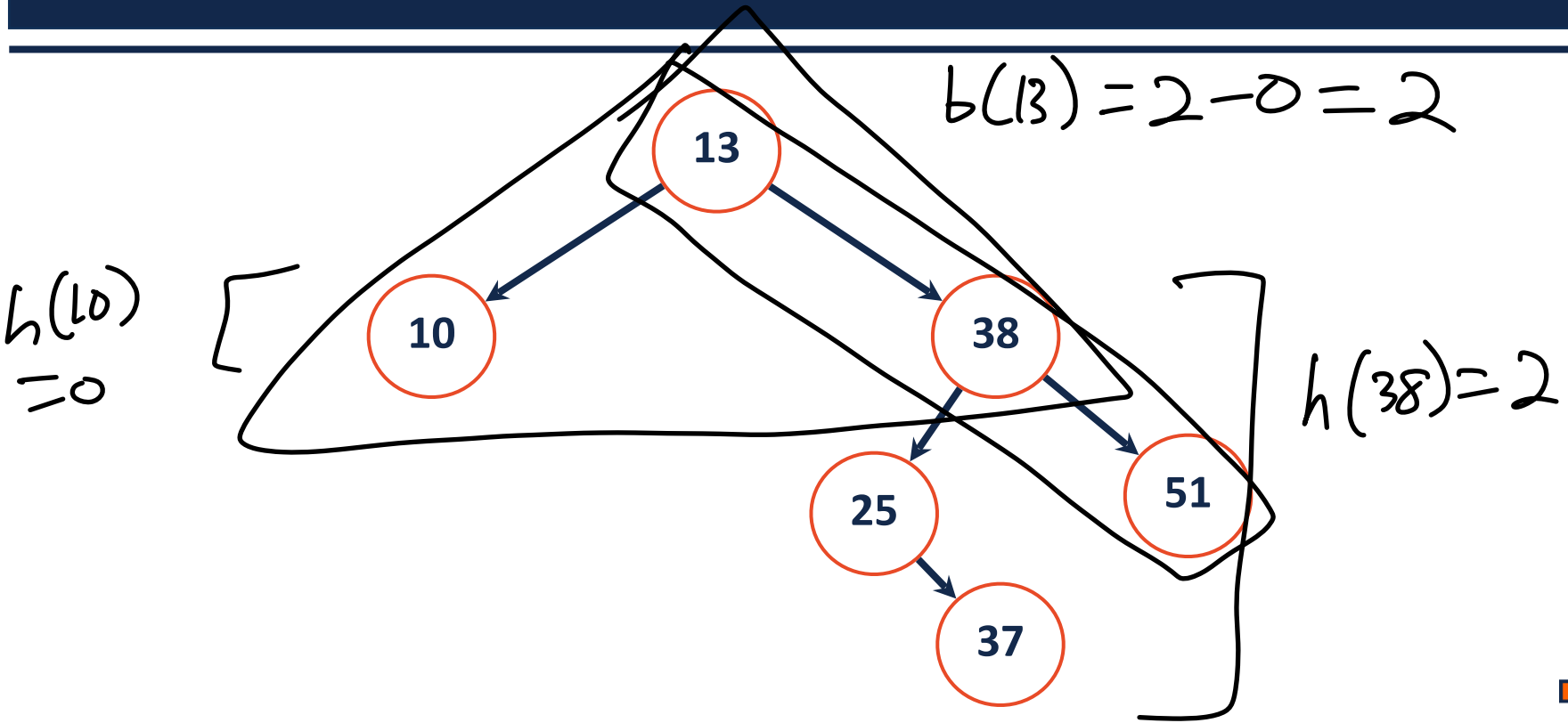
How many pointers are modified in a left rotation?

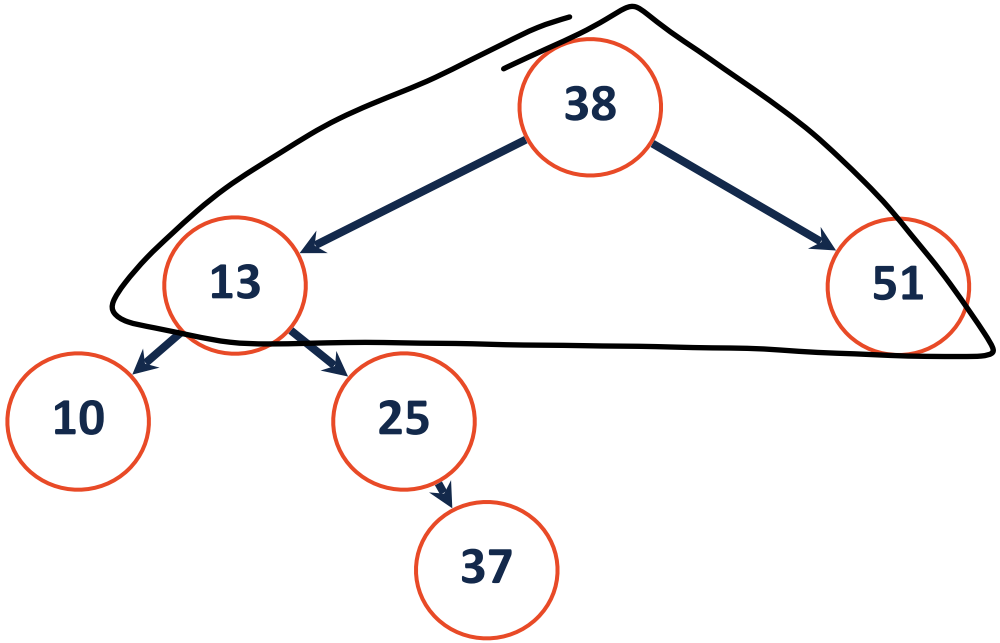
Join Code: **225**

Right Rotation

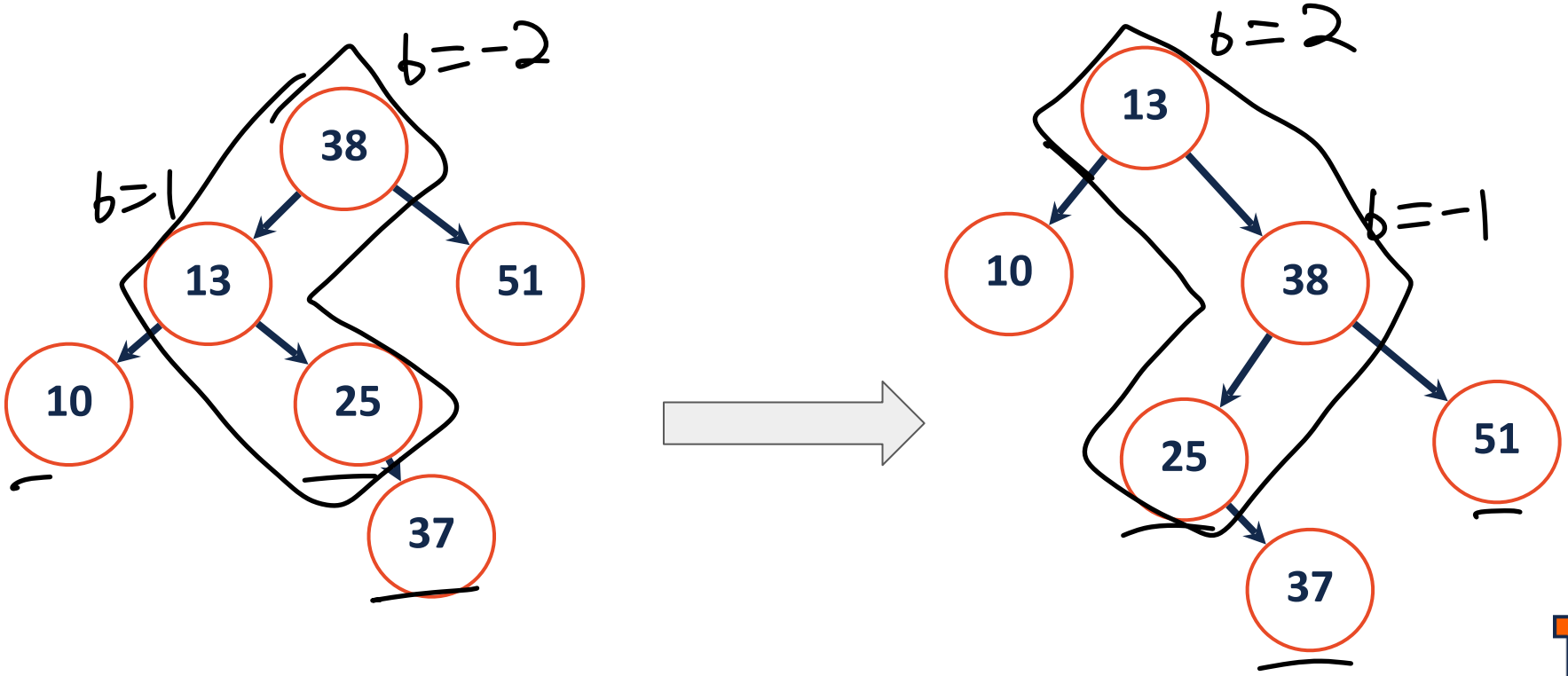



38
13
51
10
25
37
h(38) = 0 - 2
= -2
h(51) = 0
h(13)
= 2

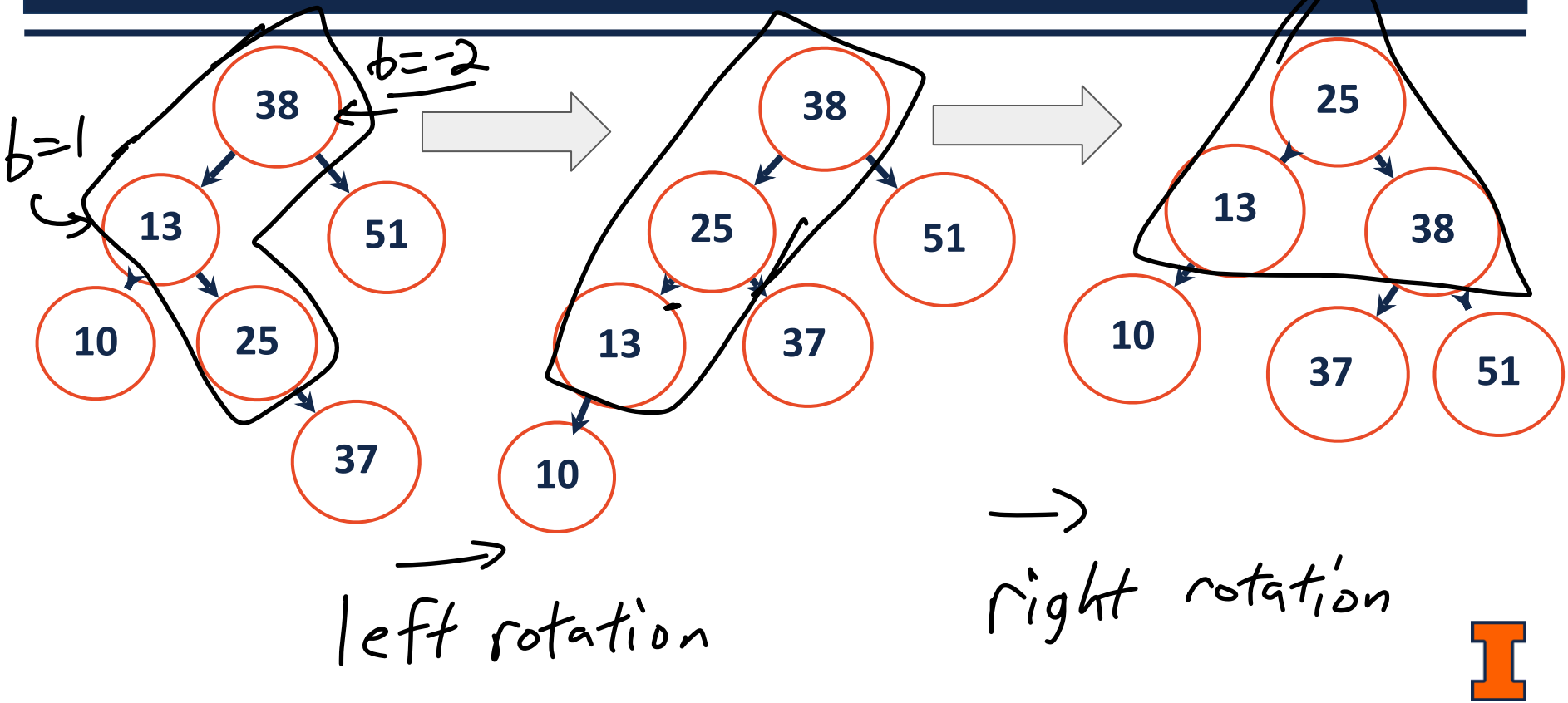

b(13) = 2 − 0 = 2
13
10
38
25
51
37
h(10) = 0
h(38) = 2


38
13
51
10
25
37

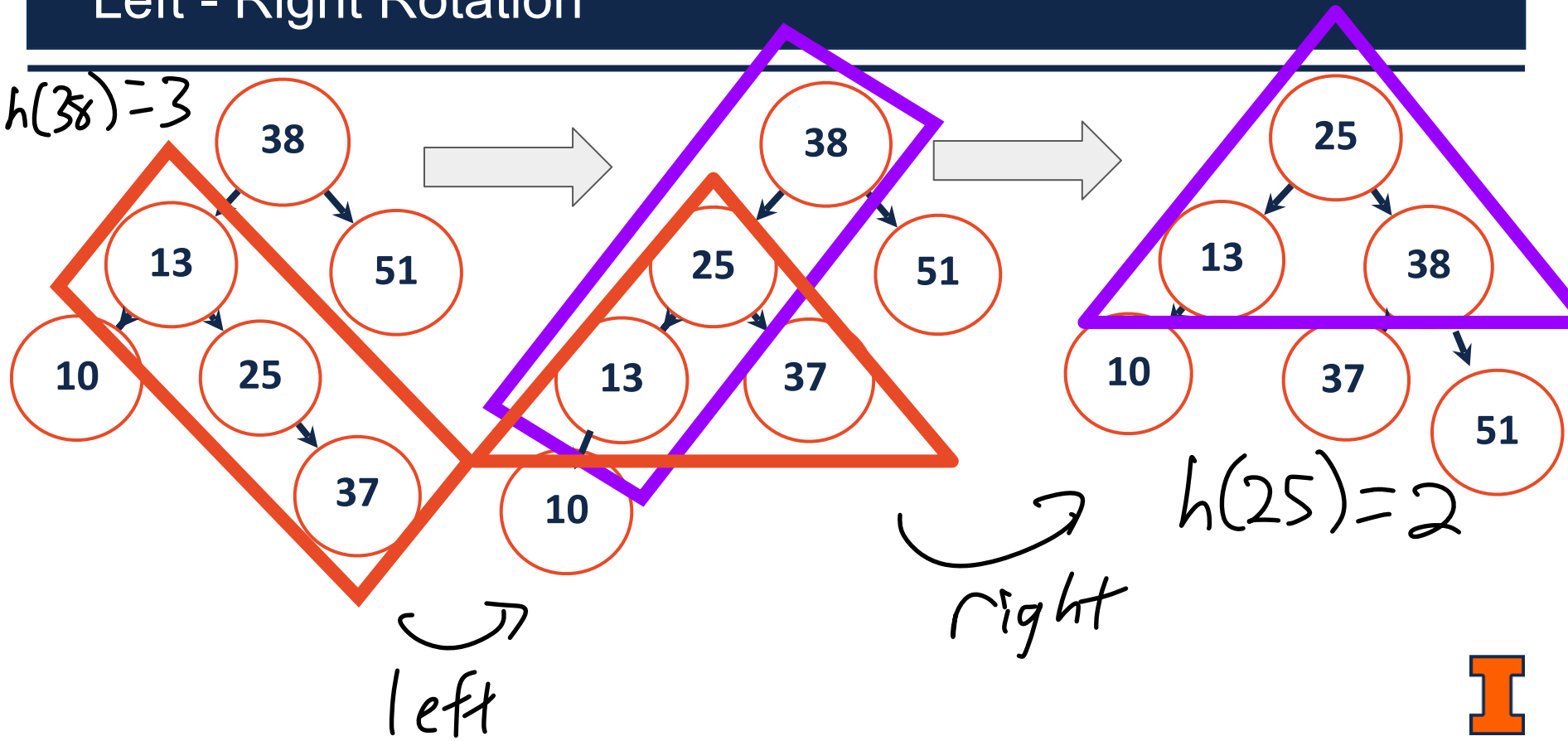
What to do in this case?



Left - Right Rotation



Left - Right Rotation

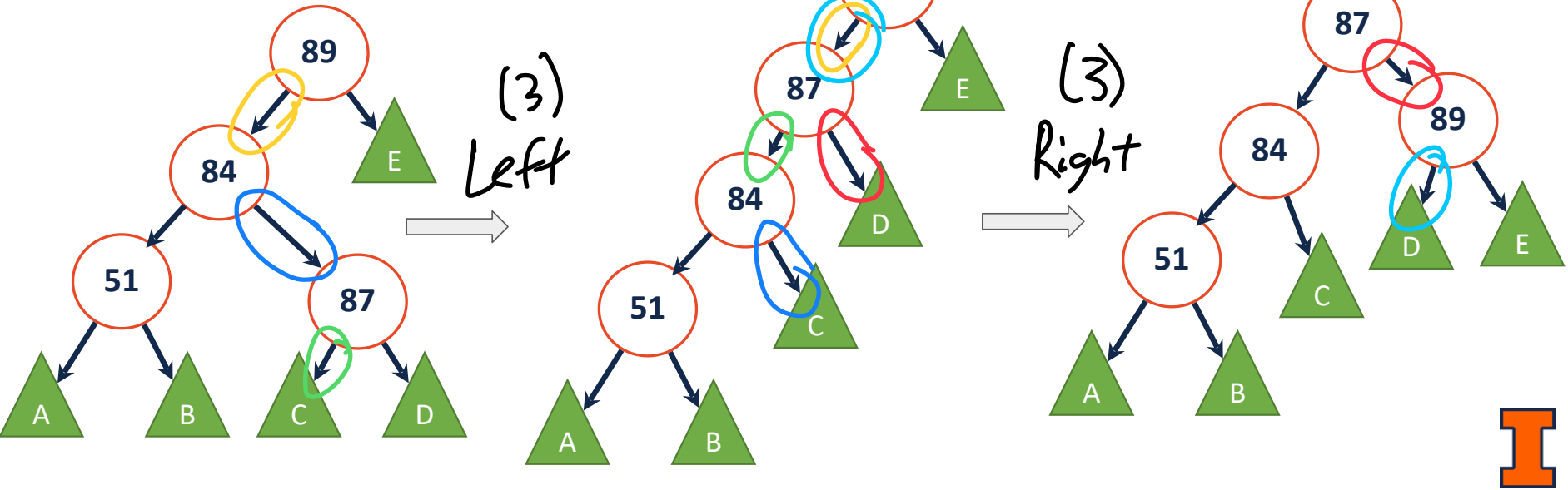


Left-Right Rotation

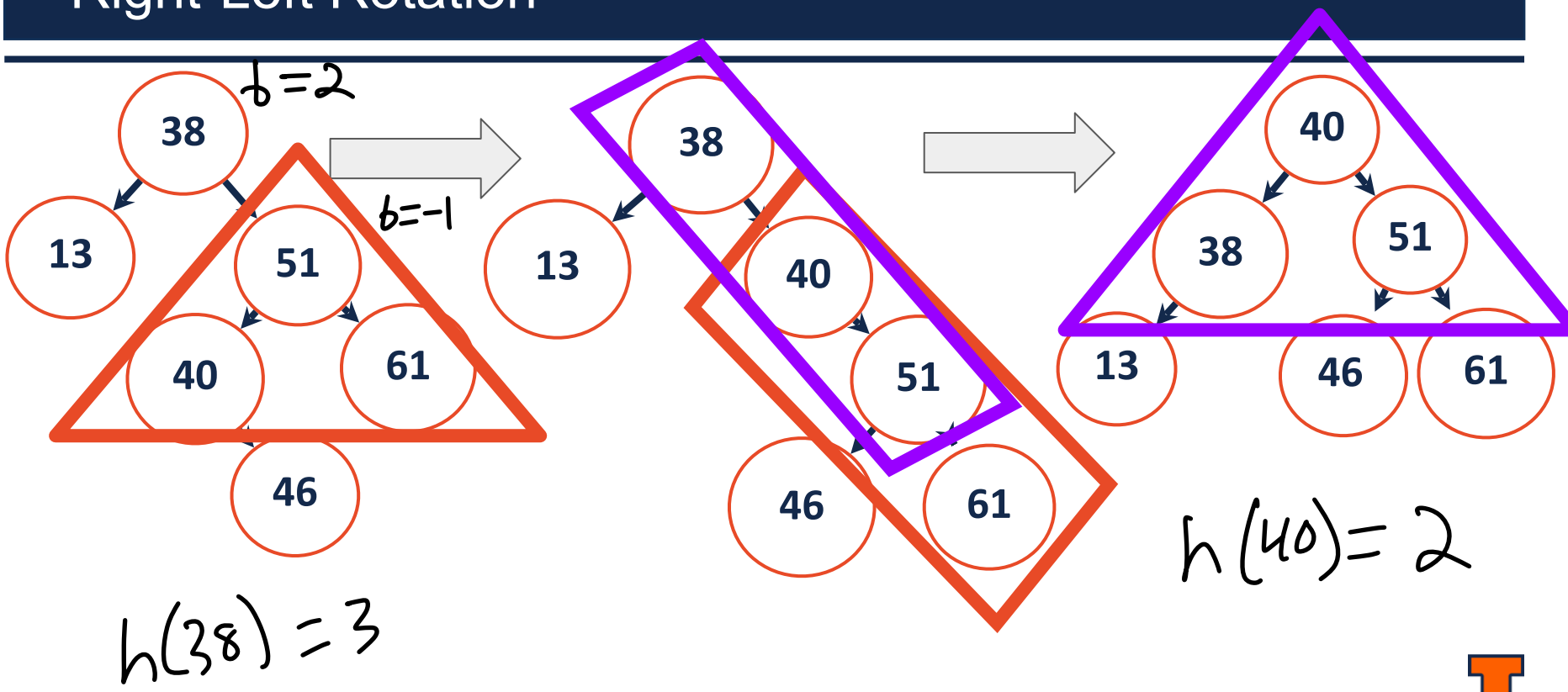
How many pointers are modified in a left rotation?

6

Join Code: **225**



Right-Left Rotation



BBST Rotations Summary

Four kinds of rotations (L, R, LR, RL)

All rotations are local (subtrees are not impacted)

All rotations are constant time: O(1)

BST property maintained

GOAL: $h \rightarrow O(\log n)$

We call these trees: AVL Trees

AVL Trees

Three issues for consideration:

1. Detecting Imbalance - How do I detect an imbalance? ✓ $|b| > 1$
2. Rotations - How do they work? Modifying pointers to make "sticks" → "mountains"
3. Selecting a rotation - Which one do I choose?

balance r -2

T_L -1 Right Rotation

1 Left-Right Rotation

2

T_R 1 Left Rotation

-1 Right-Left Rotation