

# Binary Search Trees: Balance

Daniel Kane

Department of Computer Science and Engineering  
University of California, San Diego

Data Structures  
Data Structures and Algorithms

# Learning Objectives

- Think about the runtime of basic binary tree operations.
- Understand the motivation behind binary search tree balance.
- Implement a rotation.

# Outline

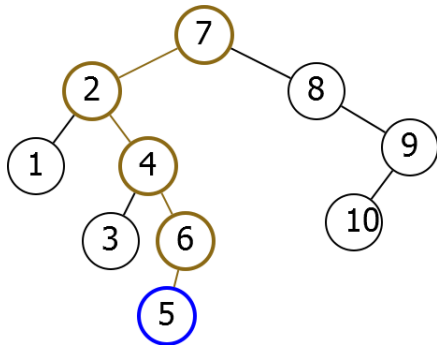
- 1 Runtime
- 2 Balanced Trees
- 3 Rotations

# Runtime

How long do Binary Search Tree operations take?

# Find

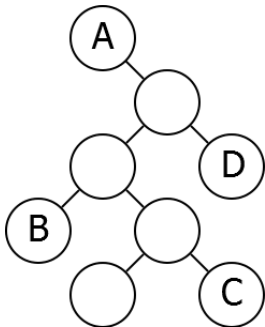
Find(5)



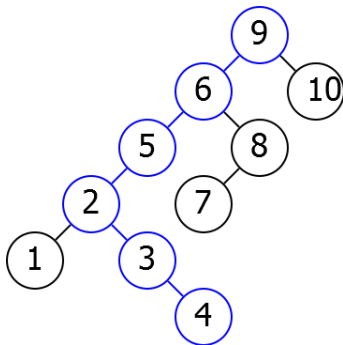
Number of operations =  $O(\text{Depth})$

# Problem

Which nodes will be faster to search for in the following tree?



# Example I



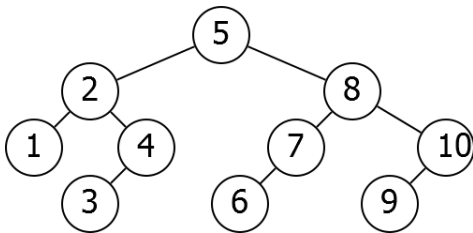
Depth can be as bad as  $n$ .

# Outline

- ① Runtime
- ② **Balanced Trees**
- ③ Rotations



## Example II



Depth can be much smaller.

# Balance

- Want left and right subtrees to have approximately the same size.

# Balance

- Want left and right subtrees to have approximately the same size.
- Suppose perfectly balanced:

# Balance

- Want left and right subtrees to have approximately the same size.
- Suppose perfectly balanced:
  - Each subtree half the size of its parent.
  - After  $\log_2(n)$  levels, subtree of size 1.
  - Operations run in  $O(\log(n))$  time.

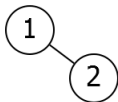
# Problem

Insertions and deletions can destroy balance!

1

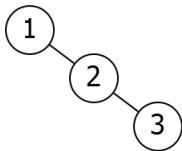
# Problem

Insertions and deletions can destroy balance!



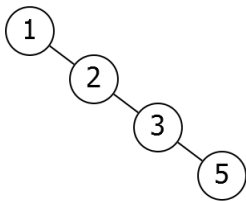
# Problem

Insertions and deletions can destroy balance!



# Problem

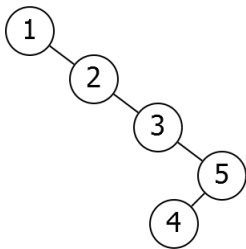
Insertions and deletions can destroy balance!





# Problem

Insertions and deletions can destroy balance!



# Outline

- ① Runtime
- ② Balanced Trees
- ③ Rotations

# Rebalancing

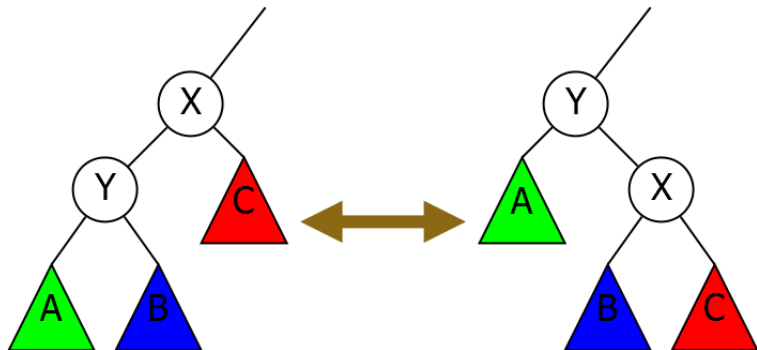
Idea: Rearrange tree to maintain balance.

# Rebalancing

Idea: Rearrange tree to maintain balance.

Problem: How do we rearrange tree while maintaining order?

# Rotations



$$A < Y < B < X < C$$

# Implementation

## RotateRight( $X$ )

$P \leftarrow X.\text{Parent}$

$Y \leftarrow X.\text{Left}$

$B \leftarrow Y.\text{Right}$

$Y.\text{Parent} \leftarrow P$

$P.\text{AppropriateChild} \leftarrow Y$

$X.\text{Parent} \leftarrow Y, Y.\text{Right} \leftarrow X$

$B.\text{Parent} \leftarrow X, X.\text{Left} \leftarrow B$

# Next Time

How to keep a tree balanced. AVL trees.