

# Binary Search Trees: Applications

Daniel Kane

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

Data Structures  
Data Structures and Algorithms

# Learning Objectives

- Compute order statistics in binary search trees.
- Use trees to store and manipulate sequential lists of elements.

# Outline

① Order Statistics

② Color Flips

# Problem

Things you might want to do:

- Return the 7<sup>th</sup> largest element.
- Return the median element.
- Return the 25% percentile element.

# Order Statistics

## Order Statistics

**Input:** The root of a tree  $T$  and a number  $k$

**Output:** The  $k^{th}$  smallest element in  $T$

# Idea

- Need to know which subtree to look in.
- Need to know **how many** elements are in left subtree.

## New Field

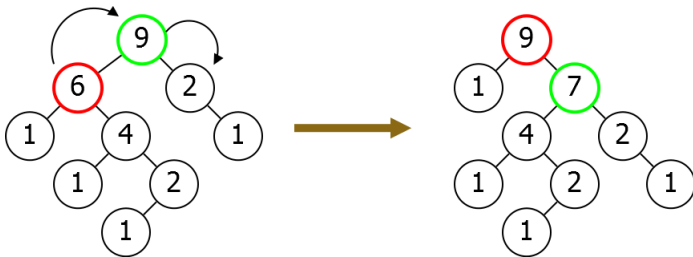
$N$ .Size returns the number of elements in the subtree of  $N$ . Should satisfy:

$$N.Size = N.Left.Size + N.Right.Size + 1,$$

where null nodes have size zero.

# Maintaining Value

When you rotate, you need to recompute sizes.





# Recompute

RecomputeSize( $N$ )

$N.Size \leftarrow N.Left.Size + N.Right.Size + 1$

Rotate

As before

RecomputeSize(Old root)

RecomputeSize(New root)

# Order Statistics

$\text{OrderStatistic}(R, k)$

$s \leftarrow R.\text{Left}.\text{Size}$

if  $k = s + 1$ :

    return  $R$

else if  $k < s + 1$ :

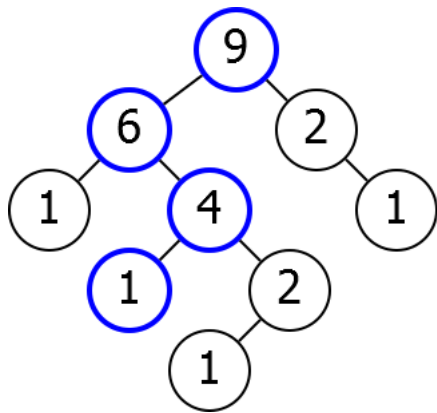
    return  $\text{OrderStatistic}(R.\text{Left}, k)$

else if  $k > s + 1$ :

    return  $\text{OrderStatistic}(R.\text{Right}, k - s - 1)$

# Analysis

Runtime  $O(h)$ .



# Puzzle

How do you compute the rank of the node with a given key?

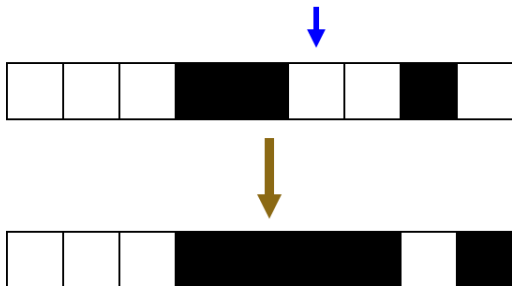
# Outline

① Order Statistics

② Color Flips

# Problem

- Array of squares.
- Each black or white.
- Want to be able to flip colors of all squares after index  $x$ .

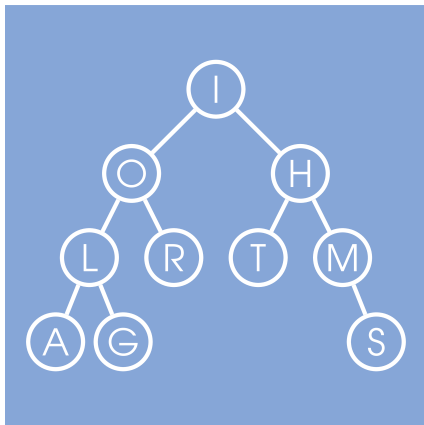


# Operations

- `NewArray( $n$ )` - Creates an array with  $n$  white squares.
- `Color( $m$ )` - Returns color of  $m^{th}$  square.
- `Flip( $x$ )` - Flips the color of all squares of index  $> x$ .

# New Use For Trees

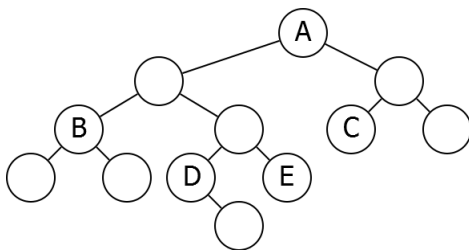
Store elements in sorted order.





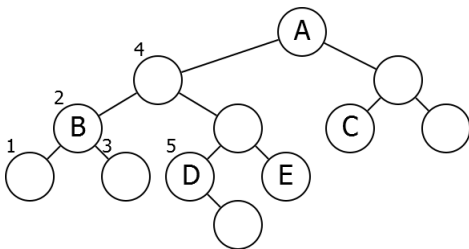
# Problem

Which node represents the 5<sup>th</sup> smallest element in this tree?



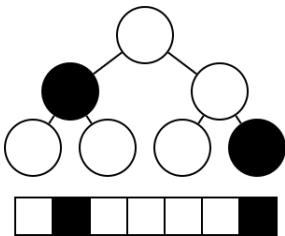
# Problem

Which node represents the 5<sup>th</sup> smallest element in this tree?



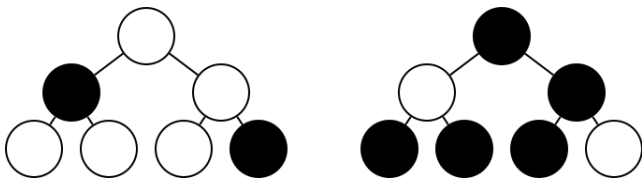
# Idea

Store tree with nodes corresponding to list colors:



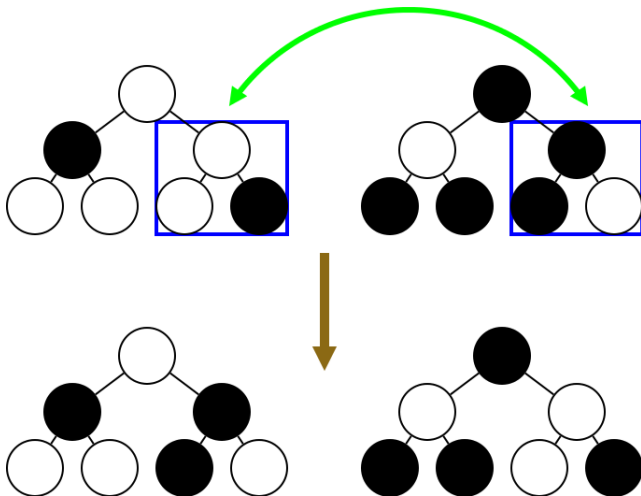
## Idea II

Two trees- one with opposite colors:



# Idea

Flip using merge and split



# Create

## `NewArray( $n$ )`

Create two trees  $T_1, T_2$  with keys  $1 \dots n$ .

Give nodes extra Color field.

All in  $T_1$  have color White

All in  $T_2$  have color Black

# Find

Color(*m*)

$N \leftarrow \text{Find}(m, T_1)$

return  $N.\text{Color}$

# Flip

Flip( $x$ )

$(L_1, R_1) \leftarrow \text{Split}(T_1, x)$

$(L_2, R_2) \leftarrow \text{Split}(T_2, x)$

$\text{Merge}(L_1, R_2) \rightarrow T_1$

$\text{Merge}(L_2, R_1) \rightarrow T_2$



# Moral

Trees can be used for more than searching.  
Can be used to store lists.