



# Data Structures and Algorithms

## Census Problem

# Acknowledgement

- The contents of these slides have origin from School of Computing, National University of Singapore.
- We greatly appreciate support from Dr. Steven Halim for kindly sharing these materials.

# Policies for students

- These contents are only used for students PERSONALLY.
- Students are NOT allowed to modify or deliver these contents to anywhere or anyone for any purpose.



# Recording of modifications

- Currently, there are no modification on these contents.

# Outline

## Motivation: Census Problem

- Abstract Data Type (ADT) Table
- Solving Census Problem with CS1020 Knowledge
- The “performance issue”

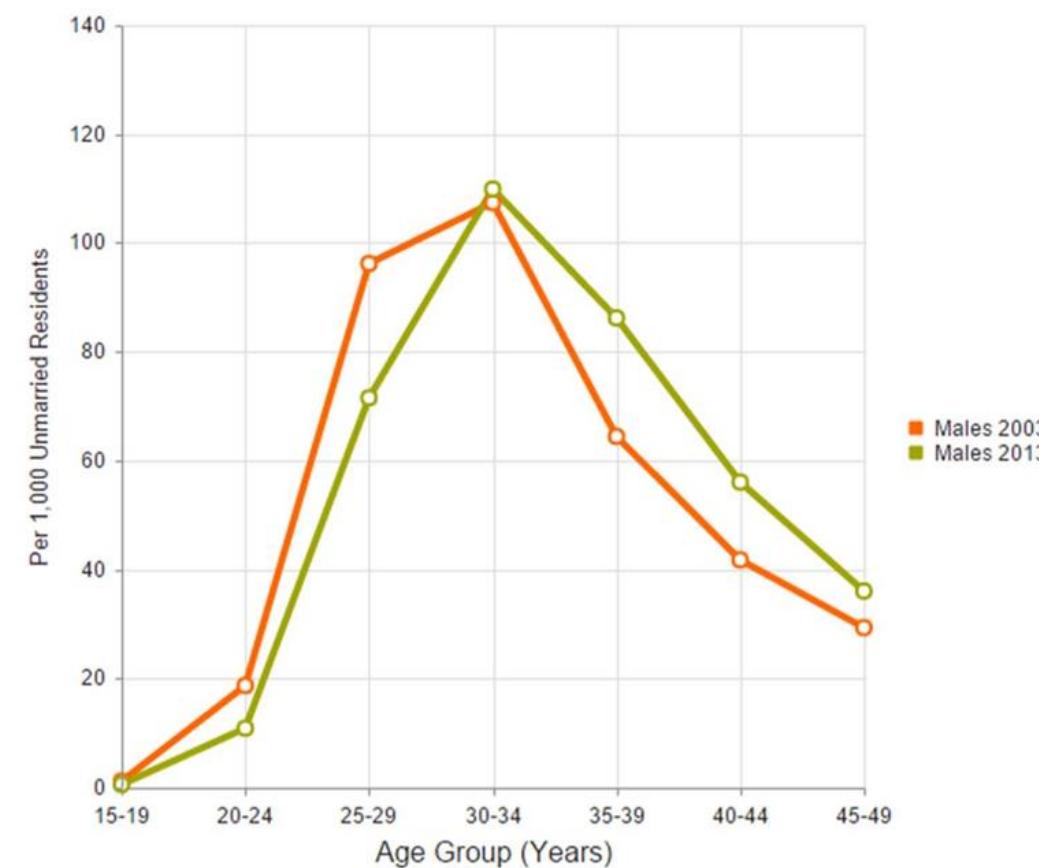
## Binary Search Tree (BST)

- Heavy usage of [VisuAlgo Binary Search Tree Visualization](#)
- Simple analysis of BST operations
- Java Implementation

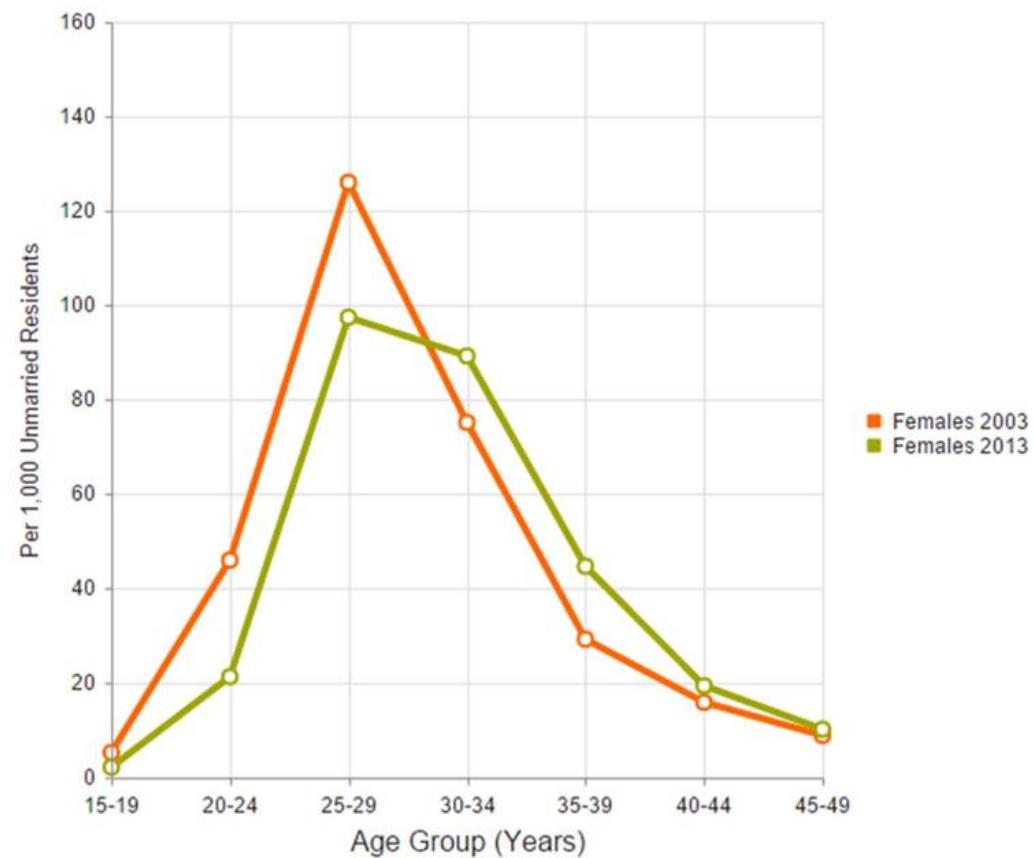
## PS2 Preview

# Census is Important!

Age-Specific Marriage Rates (Males)



Age-Specific Marriage Rates (Females)



Source:

<http://www.singstat.gov.sg>

# Sun Tzu's Art of War

## Chapter 1 “The Calculations”

知彼知己百战不殆

zhī bǐ zhī jǐ bǎi zhàn bù dài

(If you know your enemies and know yourself,  
you will not be imperiled in a hundred battles)

# Your Age (2013 data)

'[ (or ']) means that endpoint is included (closed)

1. [24 ...

$\infty)$

2. [23 ...

24)

3. [22 ...

23)

4. [21 ...

22)

5. [20 ...

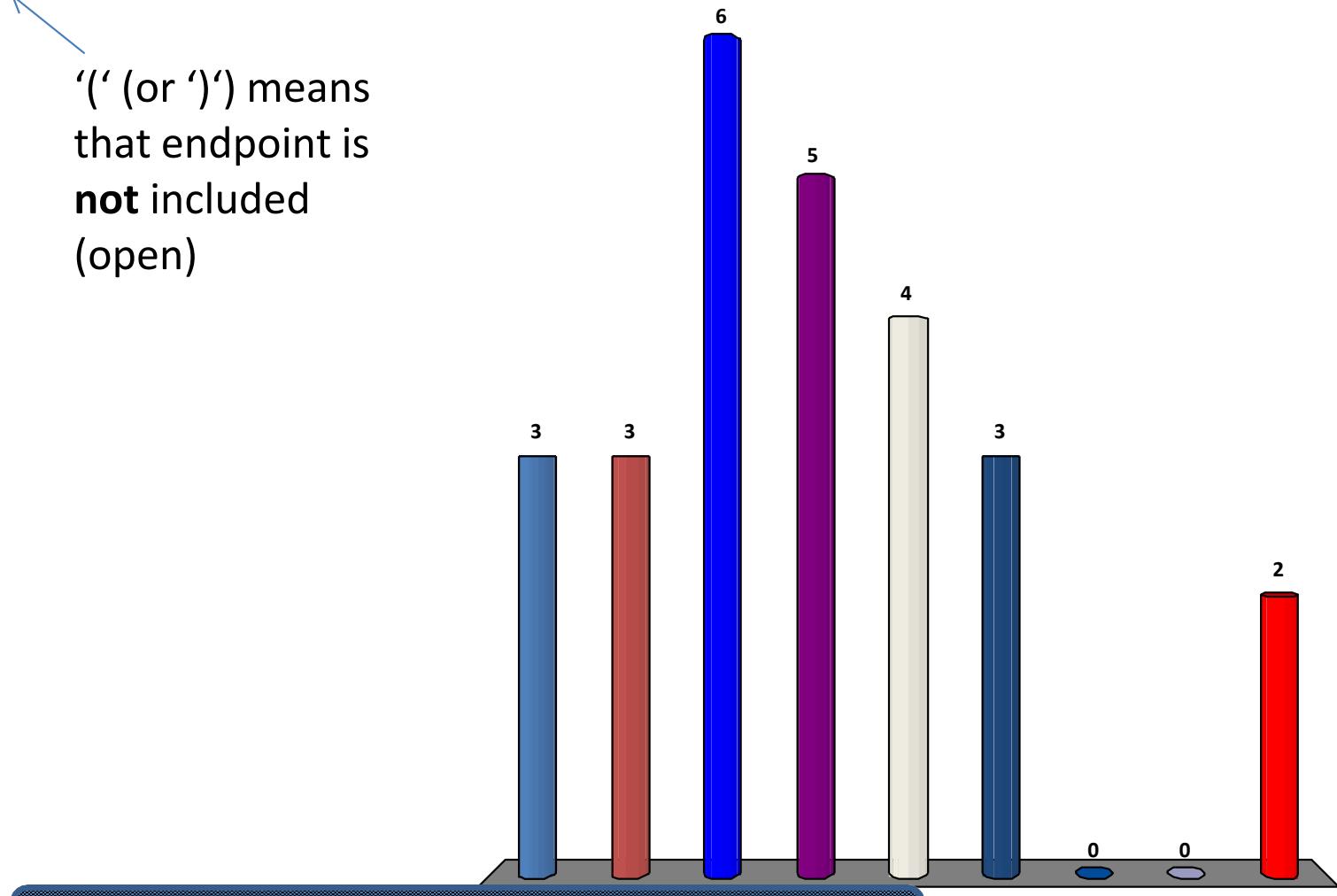
21)

6. [19 ...

20)

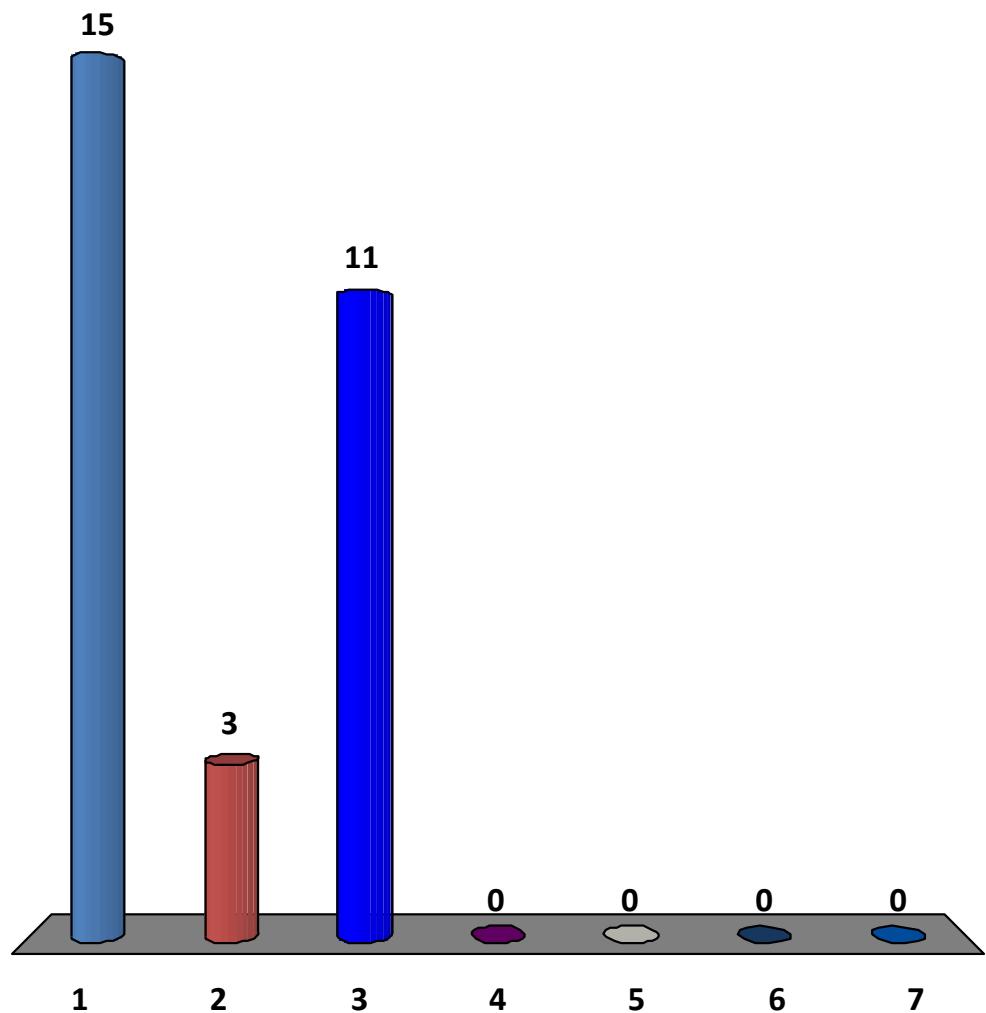
'( (or ')') means that endpoint is **not** included (open)

Mean =  
3.196125



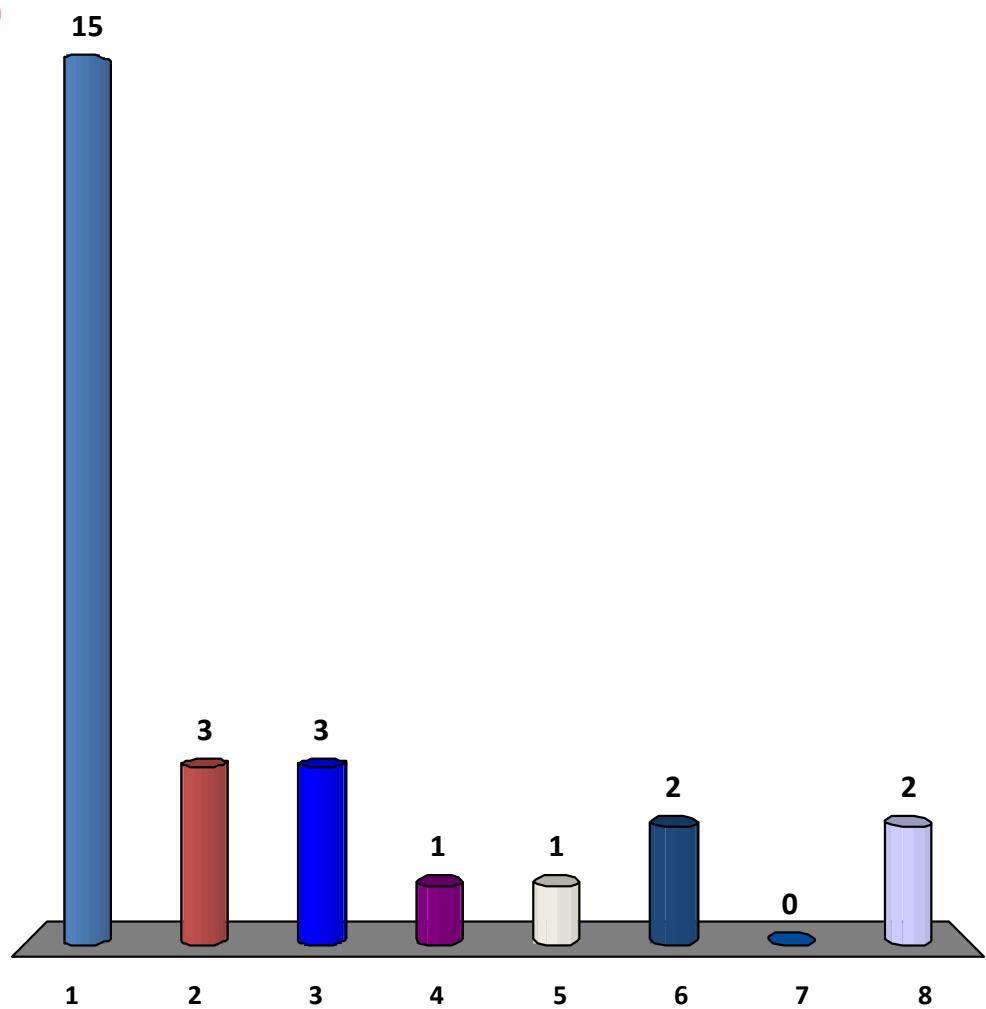
# Your Major (2013 data)

1. Computer Science  
~~(CS) Communications and Media (C&M)~~
2. Comp. Biology (CB)
3. Computer Engineering (CEG/CEC)
4. Information System (IS)
5. Science Maths (SCI)
6. None of the above :O



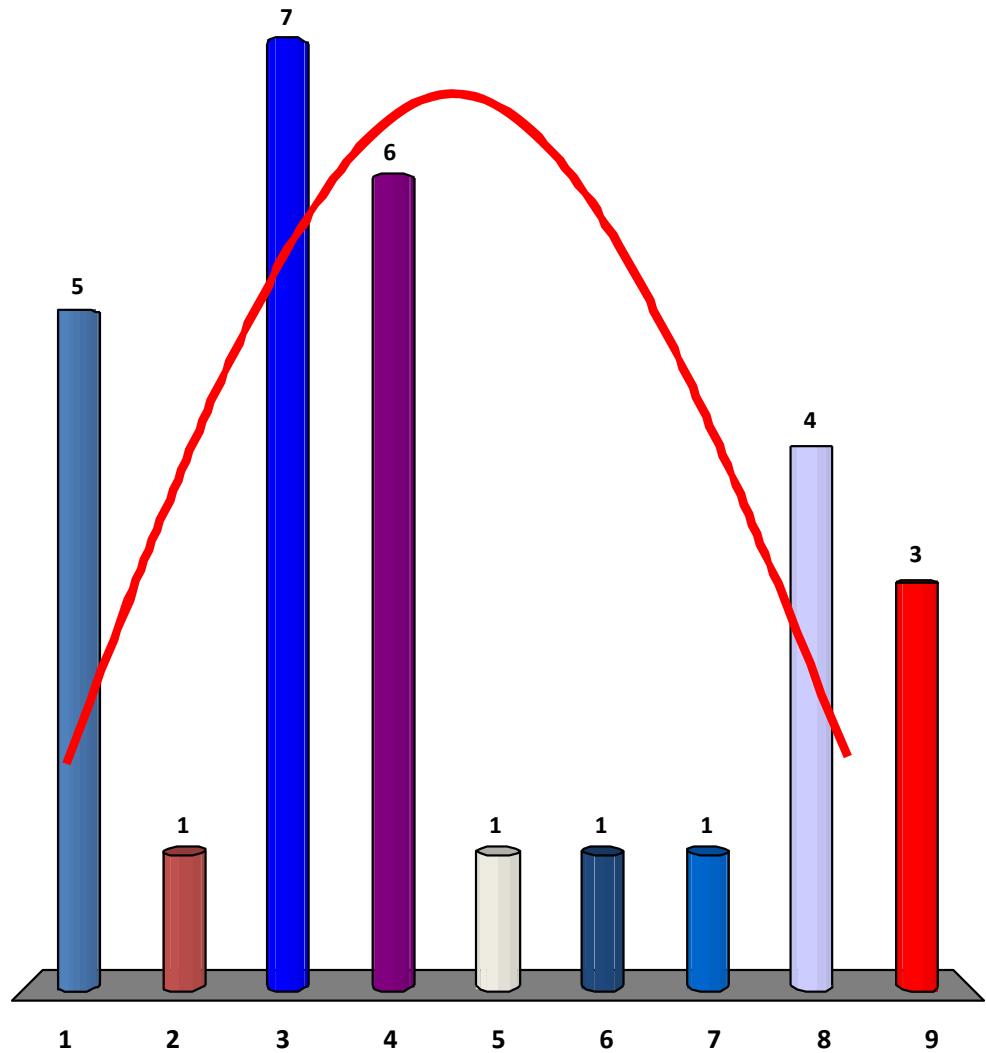
# Your Nationality (2013 data)

1. Singaporean (should be  $\geq 70\%$  according to MOE rules)
2. Chinese
3. Indian
4. Indonesian
5. Vietnamese
6. Malaysian
7. European
8. None of the above



# Your CAP (2013 data)

1. [4.5 ... 5.0]
2. [4.25 ... 4.5)
3. [4.0 ... 4.25)
4. [3.75 ... 4.0)
5. [3.5 ... 3.75)
6. [3.25 ... 3.5)
7. [3.0 ... 3.25)
8. [0.0 ... 3.00)
9. I do not want to  
tell



# What Happen After Census?

Data  
Mining



Statistica  
I  
Analysis

# Abstract Data Type (ADT) Table

Let's deal with one aspect of our census: **Age**

To simplify this lecture, we assume that students' age ranges from [0 ... 100), all integers, and distinct

Required operations:

1. Search whether there is a student with a certain age?
2. Insert a new student (that is, insert his/her age)
3. Determine the youngest and oldest student
4. List down the ages of students in sorted order
5. Find a student slightly older than a certain age!
6. Delete existing student (that is, remove his/her age)
7. Determine the median age of students
8. How many students are younger than a certain age?

# CS1020: Unsorted Array

Index	0	1	2	3	4	5	6	7	
A	5	7	71	50	23	4	6	15	

No	Operation	Time Complexity
1	Search(age)	$O(n)$
2	Insert(age)	$O(1)$
3	FindOldest()	$O(n)$
4	ListSortedAges()	$O(n \log n)$
5	NextOlder(age)	$O(n)$
6	Remove(age)	$O(n)$
7	GetMedian()	$O(n \log n)/O(n)$
8	NumYounger(age)	$O(n \log n)$

# CS1020: Sorted Array

Index	0	1	2	3	4	5	6	7	
A	4	5	6	7	15	23	50	71	

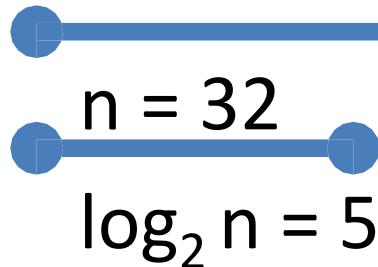
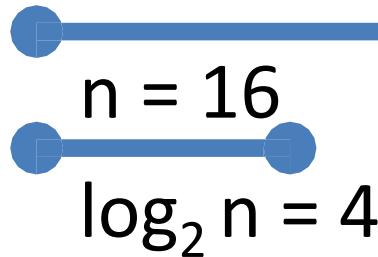
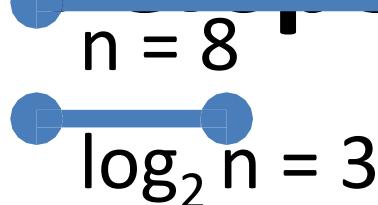
No	Operation	Time Complexity
1	Search(age)	$O(\log n)$
2	Insert(age)	$O(n)$
3	FindOldest()	$O(1)$
4	ListSortedAges()	$O(n)$
5	NextOlder(age)	$O(\log n)$
6	Remove(age)	$O(n)$
7	GetMedian()	$O(1)$
8	NumYounger(age)	$O(\log n)$

# With Just CS1020 Knowledge

No	Operation	Unsorted Array	Sorted Array
1	Search(age)	$O(n)$	$O(\log n)$
2	Insert(age)	$O(1)$	$O(n)$
3	FindOldest()	$O(n)$	$O(1)$
4	ListSortedAge	$O(n \log n)$ S Dynamic data tructure operations	$O(n)$
5	NextOlder(ag)	$O(n)$	$O(\log n)$
6	Remove(age)	$O(n)$	$O(n)$
7	GetMedian()	$O(n \log n) / O(n)$	$O(1)$
8	NumYounger(age)	$O(n \log n)$	$O(\log n)$

# $O(n)$ versus $O(\log n)$ : A

## Perspective



Try larger  $n$ , e.g.  $n = 1000000\dots$

A Versatile, Non-Linear Data Structure

# **BINARY SEARCH TREE (BST)**

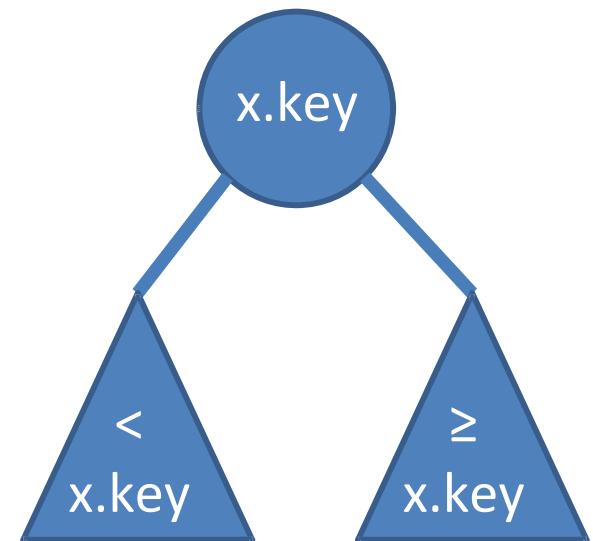
# Binary Search Tree (BST) Vertex

For every vertex  $x$ , we define:

- $x.left$  = the left child of  $x$
- $x.right$  = the right child of  $x$
- $x.parent$  = the parent of  $x$
- $x.key$  (or  $x.value$ ,  $x.data$ ) = the value stored at  $x$

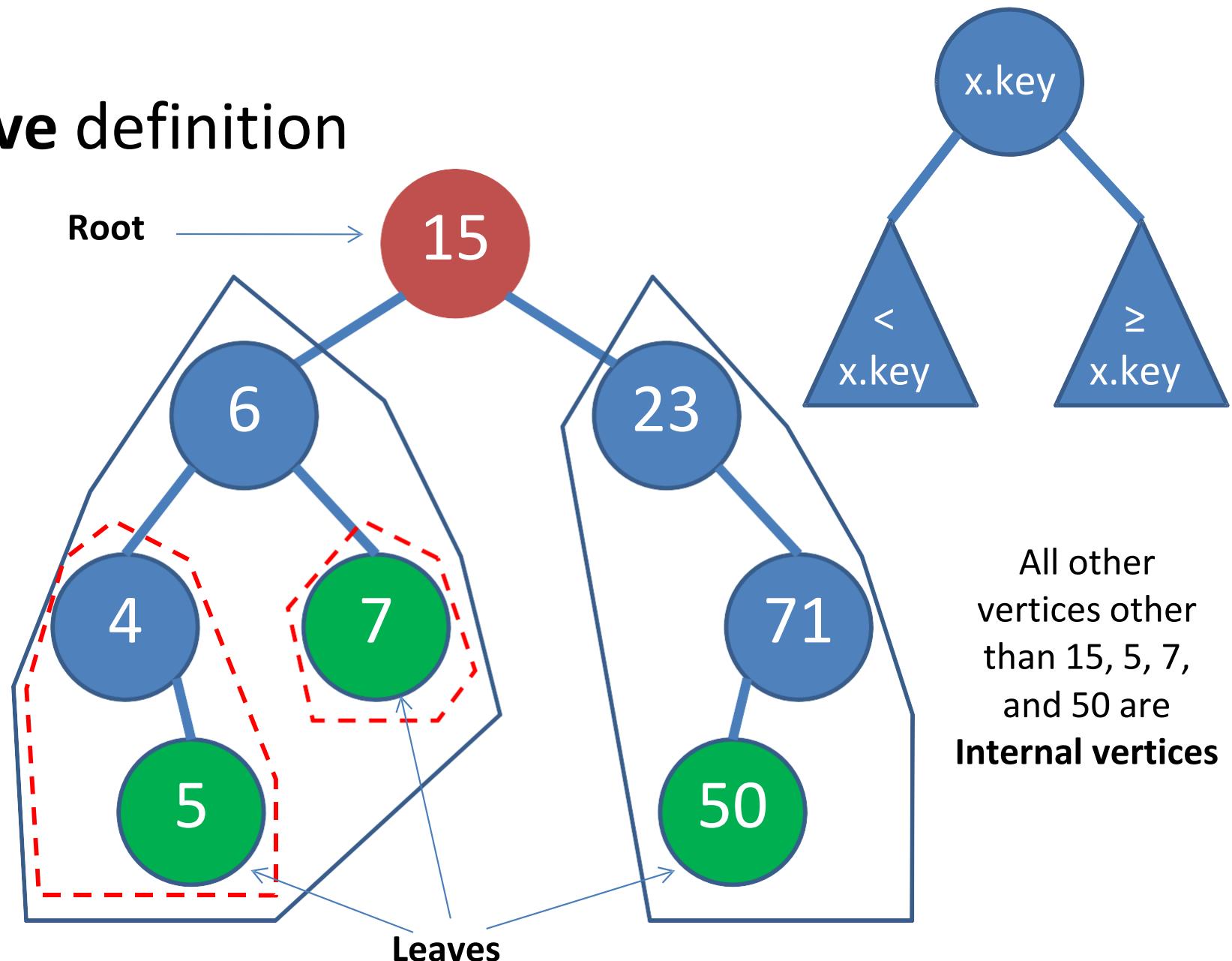
BST Property:

- $x.left.key < x.key \leq x.right.key$
- For simplicity, we assume that the keys are unique so that we can change  $\geq$  to  $>$



# BST: An Example, Keys = Ages

## Recursive definition



# BST: Search/Min/Max Operations

Ask VisuAlgo to perform various search operations on the sample BST, including find min and find max

In the screen shot below, we show **search(5)**

The screenshot shows a binary search tree (BST) with the following structure:

```
graph TD; 15 --> 6; 15 --> 23; 6 --> 4; 6 --> 7; 4 --> 5; 23 --> 50; 23 --> 71;
```

The nodes are colored: 15, 6, 4, and 5 are orange, while 23, 7, 50, and 71 are grey. The path from root 15 to node 5 is highlighted in orange.

On the left, a sidebar menu includes: Create, Search, Insert, Remove, Successor, Predecessor, and In-order Traversal. Below the menu are buttons for Find Min, Find Max, and GO, along with an input field "Enter an integer" containing the value 5.

On the right, a code editor window titled "Search for 5" displays the following pseudocode:

```
Search for 5
Found node 5
if this == null
    return null
else if this key == search value
    return this
else if this key < search value
    search right
else search left
```

At the bottom, there are navigation icons (back, forward, search, etc.) and links for About, Team, and Terms of use.

# BST: Succ/Pred-essor Operations

Ask VisuAlgo to perform Succ/Pred operations  
on the sample BST

In the screen shot below, we show **pred(15)**

The screenshot shows a binary search tree (BST) with the following structure:

```
graph TD; 15((15)) --> 6((6)); 15 --> 23((23)); 6 --> 4((4)); 6 --> 7((7)); 4 --> 5((5)); 23 --> 50((50)); 23 --> 71((71));
```

The nodes 15, 6, 7, and 23 are highlighted in orange, indicating they are part of the search path. The edges between 15 and 6, and between 6 and 7 are orange, while the edges between 7 and 23, and between 4 and 5 are black.

On the left, there is a sidebar with navigation buttons and a list of operations: Create, Search, Insert, Remove, Successor, Predecessor, and In-order Traversal. Below the sidebar are buttons for "15", "GO", and "Enter an integer".

At the bottom, there is a progress bar with "slow" and "fast" ends, and a set of navigation icons (back, forward, etc.).

On the right, there is a code editor window titled "Predecessor(15)" containing the following Java-like pseudocode:

```
if this.left != null return findMax(this.left)
else
    p = this.parent, T = this
    while(p != null && T == p.left)
        T = p, p = T.parent
    if p is null return -1
    else return p
```

Below the code editor, there is a message "Predecessor found!".

# BST: Inorder Traversal Operation

Ask VisuAlgo to perform inorder traversal operation  
on the sample BST

In the screen shot below, we *partial* inorder traversal

The screenshot shows a binary search tree (BST) with nodes containing values 4, 5, 6, 7, 15, 23, 50, and 71. Nodes 4, 5, 6, 7, and 15 are highlighted in orange, indicating they have been visited or are part of the current traversal path. Nodes 23, 50, and 71 are shown in white circles with black outlines, indicating they are yet to be visited. Orange lines connect the nodes 4, 5, 6, 7, and 15, forming the current path of the traversal.

**In-order Traversal**

```
Visit node 15.  
if this is null  
    return  
inOrder(left)  
visit this, then inOrder(right)
```

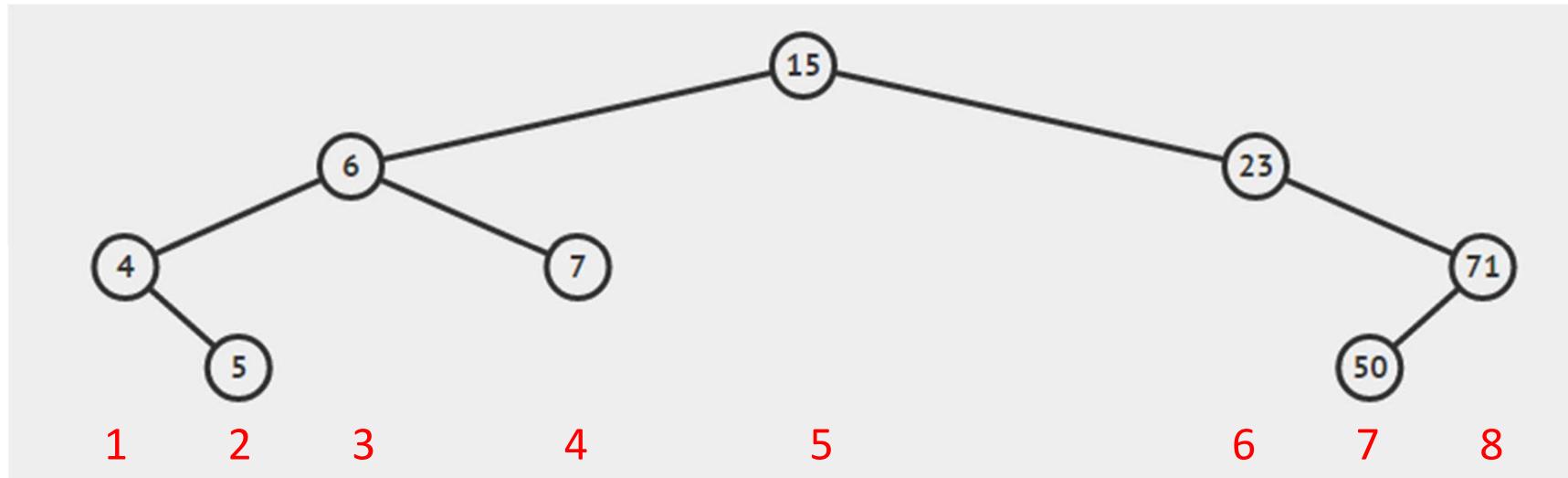
Navigation and interface elements:

- Left sidebar: Create, Search, Insert, Remove, Successor, Predecessor, In-order Traversal.
- Bottom navigation: slow, fast, back, forward, progress bar.
- Bottom right: About, Team, Terms of use.

# BST: Select/Rank Operations

These 2 operations will be added to VisuAlgo BST visualization *soon*; for now, here are the concepts:

- $\text{Select}(k)$  – Return the value  $v$  of  $k$ -th smallest\* element
  - Examples:  $\text{Select}(1) = 4$ ,  $\text{Select}(3) = 6$ ,  $\text{Select}(8) = 71$ , etc (1-based index)
- $\text{Rank}(v)$  – Return the ranking\*  $k$  of element  $v$ 
  - Examples:  $\text{Rank}(4) = 1$ ,  $\text{Rank}(6) = 3$ ,  $\text{Rank}(71) = 8$ , etc
- Details will be discussed in the next lecture



# BST: Insert Operation

Ask VisuAlgo to perform various insert operations  
on the sample BST

In the screen shot below, we show **insert(20)**

The screenshot shows a binary search tree (BST) with the following structure:

```
graph TD; 15 --> 6; 15 --> 23; 6 --> 4; 6 --> 7; 4 --> 5; 23 --> 20; 23 --> 71; 71 --> 50;
```

The node 20 is highlighted in orange, indicating it is the current node being inserted. The path from the root 15 to the insertion point 20 is shown with orange lines.

On the left, there is a sidebar with the following buttons:

- Create
- Search
- Insert** (highlighted in red)
- Remove
- Successor
- Predecessor
- In-order Traversal

Below the sidebar, there is a text input field containing "20" and a "GO" button.

On the right, there is a log window titled "Insert 20" with the following content:

```
20 has been inserted!
if found insertion point
    create new node
    if value to be inserted < this key
        go left
    else go right
```

At the bottom, there are navigation icons (back, forward, search, etc.) and links for "About", "Team", and "Terms of use".

# BST: Delete/Remove Operation (1)

Ask VisuAlgo to perform various delete operations  
on the sample BST (3 cases, this is **delete leaf**)

In the screen shot below, we show **remove(5)** before deletion

The screenshot shows a Binary Search Tree (BST) with the following structure:

```
graph TD; 15 --> 6; 15 --> 23; 6 --> 4; 6 --> 7; 4 --> 5; 23 --> 50; 23 --> 71;
```

The node with value 5 is highlighted in orange, indicating it is the target for deletion. The VisuAlgo interface includes a sidebar with navigation buttons (Create, Search, Insert, Remove, Successor, Predecessor, In-order Traversal) and a main panel with a text input field containing "5" and a "GO" button. A message box displays the command "Remove 5". To the right, a code editor window shows the pseudocode for the remove operation:

```
Remove 5
Node 5 has no children. It is a leaf.
search for v
if v is a leaf
    delete leaf v
else if v has 1 child
    bypass v
else replace v with successor
```

At the bottom, there are playback controls (slow, fast, play, pause, stop) and links to "About", "Team", and "Terms of use".

# BST: Delete/Remove Operation (2)

Ask VisuAlgo to perform various delete operations on the sample BST (this is **delete vertex with one child**)

In the screen shot below, we show **remove(23)** before relayout

The screenshot shows a binary search tree (BST) with the following structure:

```
graph TD; 15 --> 6; 15 --> 71; 6 --> 4; 6 --> 7; 4 --> 5;
```

Nodes are represented as circles with values: 15, 6, 4, 7, 5, 71, and 50. Nodes 15, 71, and 50 have orange outlines, while others have black outlines. Node 23 (value 23) is highlighted with a red background and a red border.

On the left, a sidebar menu has 'Remove' selected. Below it, the number '23' is entered into a text input field, and a 'GO' button is visible. A tooltip says 'Enter an integer or comma-separated array of integers'.

On the right, a panel titled 'Remove 23' contains the following text:

```
Delete node 23 and connect its parent to its right child
```

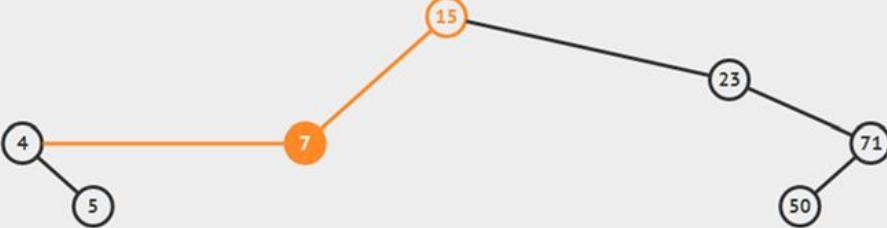
```
search for v
if v is a leaf
  delete leaf v
else if v has 1 child
  bypass v
else replace v with successor
```

At the bottom, there are navigation icons for slow, fast, and a progress bar.

# BST: Delete/Remove Operation (3)

Ask VisuAlgo to perform various delete operations on the sample BST (**delete vertex with two children**)

In the screen shot below, we show **remove(6)** before relayout



Later, we will analyze on why replacing the deleted item with its successor (and delete the old successor works)

Remove 6

```
Replace node 6 with its successor
search for v
if v is a leaf
  delete leaf v
else if v has 1 child
  bypass v
else replace v with successor
```

Create  
Search  
Insert  
Remove  
Successor  
Predecessor  
In-order Traversal

6 GO

Enter an integer or comma-separated array of integers

slow fast

About Team Terms of use

# **ANALYSIS OF BST OPERATIONS**

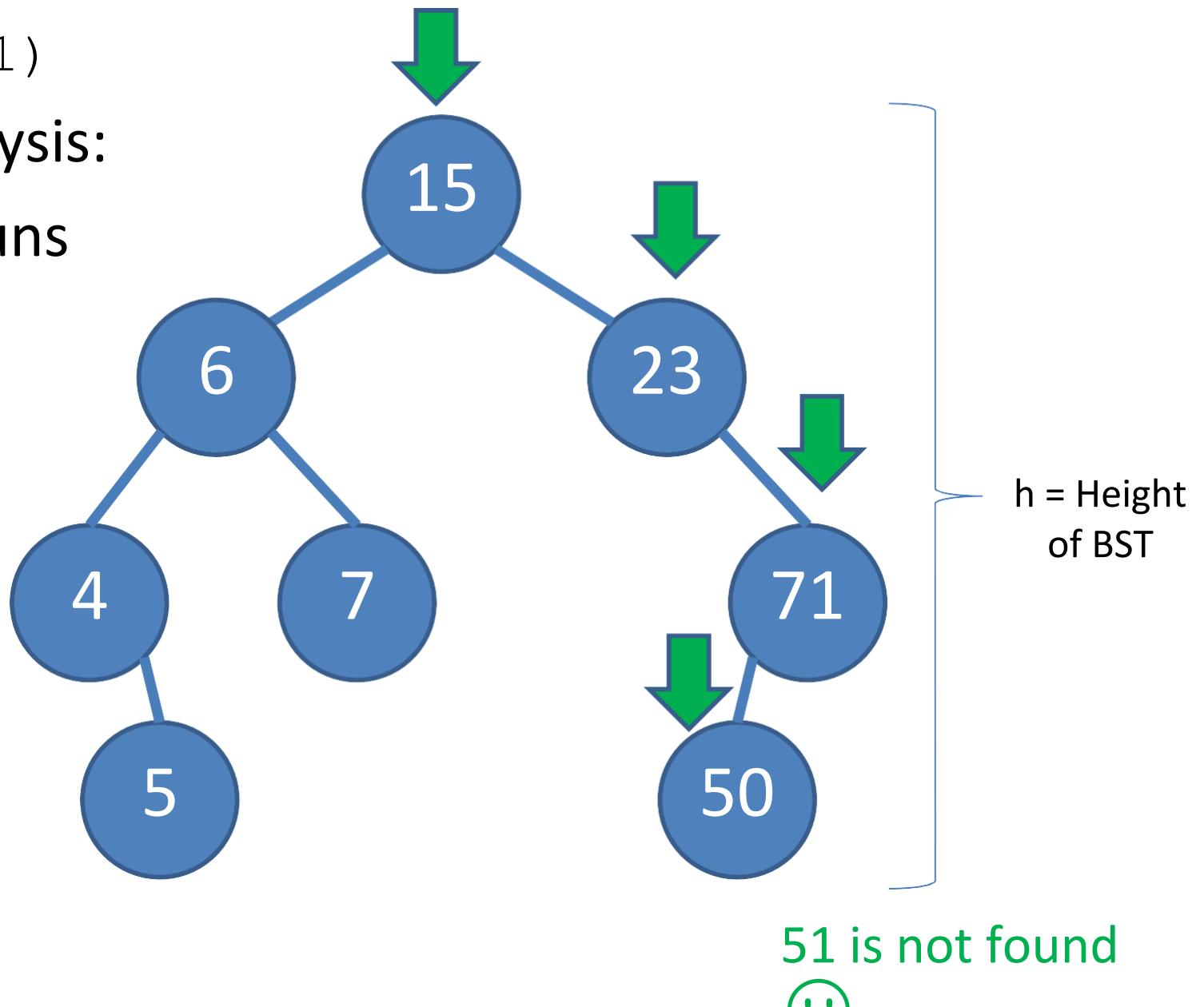
# BST: Search Analysis

search (51)

Quick analysis:

search runs

in  $O(h)$

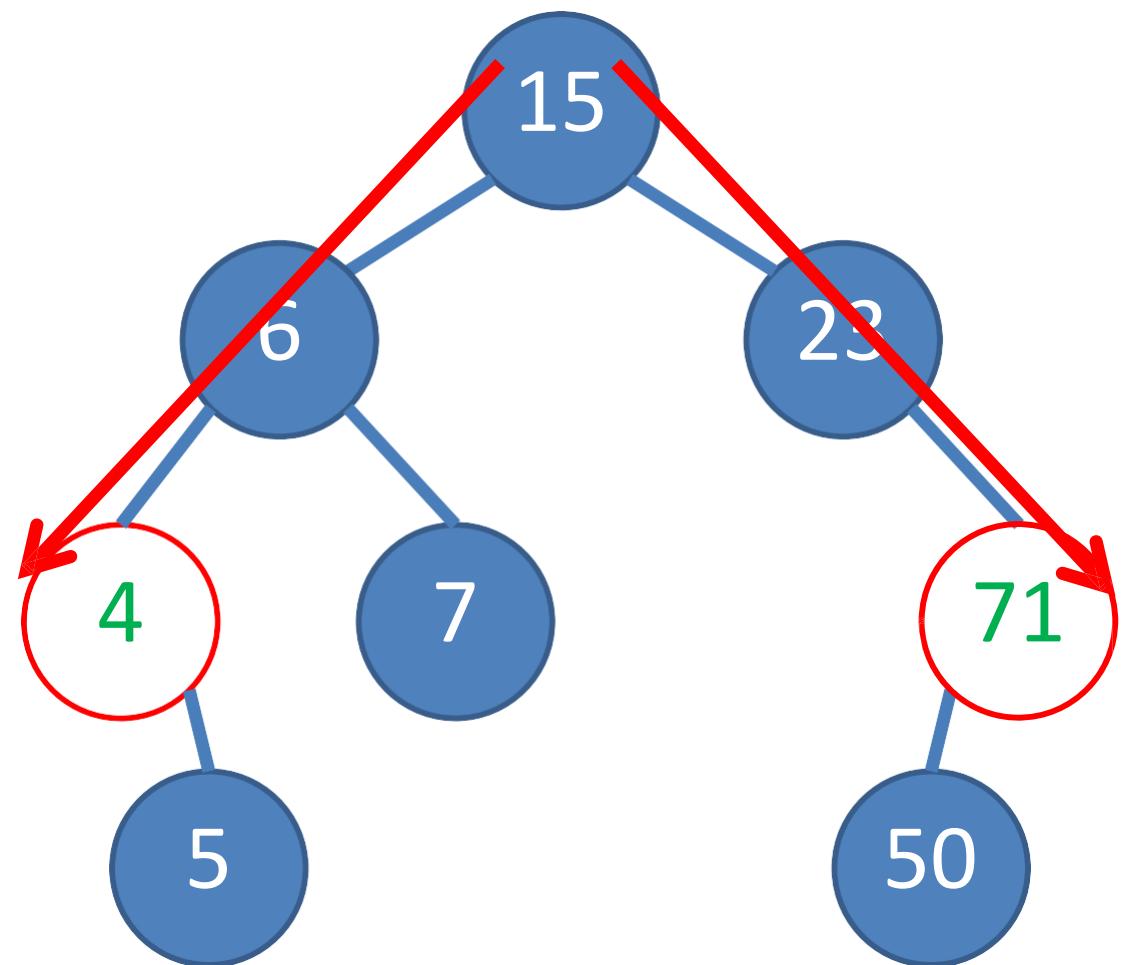


# BST: Find Min/Max Analysis

Quick analysis:

findMin/findMax

also runs in **O(h)**



# BST: Successor/Predcessor

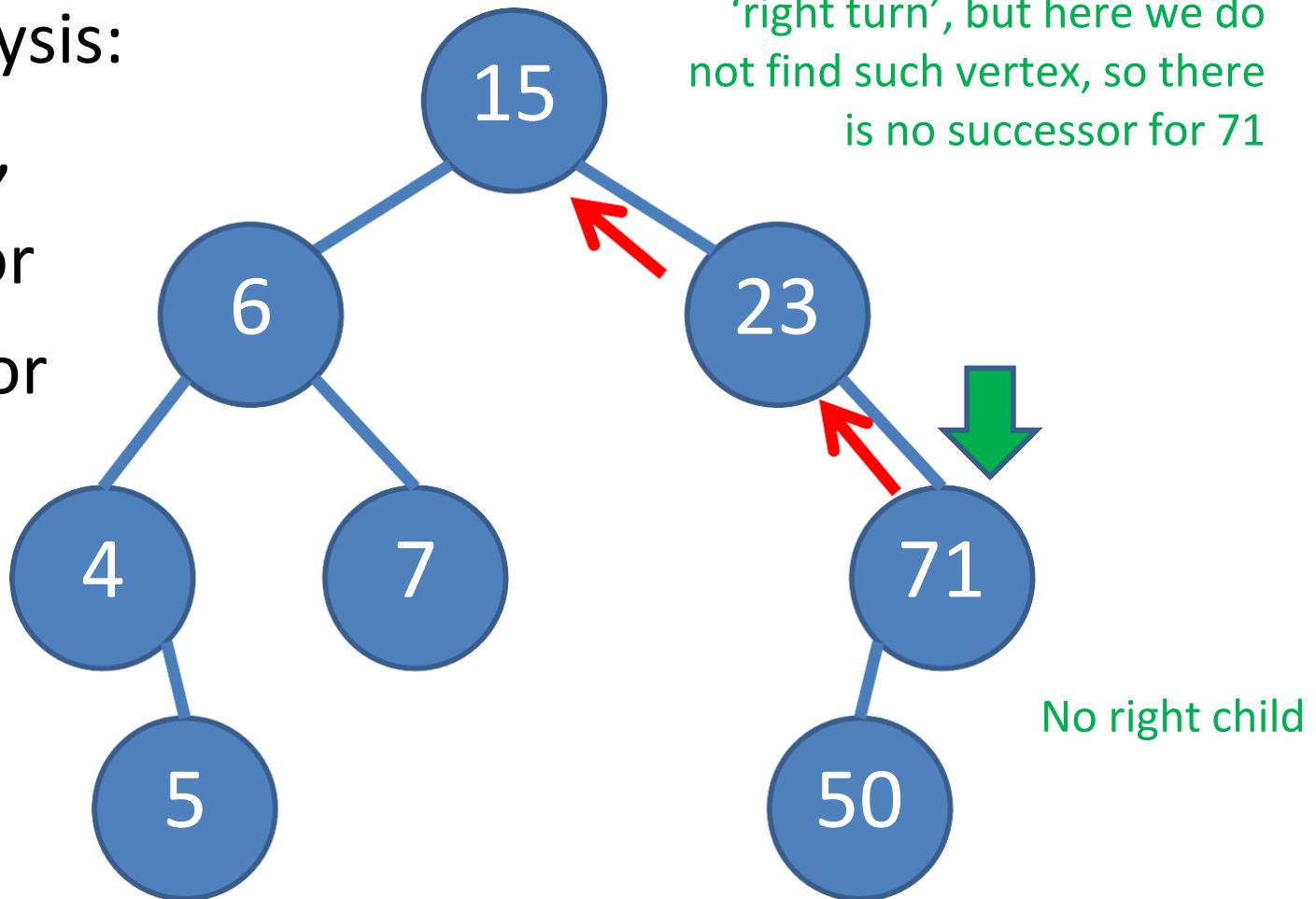
Assumption, we already done an  $O(h)$  search(71) before

Binary SIS

successor (71)

Quick analysis:

$O(h)$  again,  
similarly for  
predecessor



# BST: Inorder Traversal Analysis

Using a *new* analysis technique

Ask this question:

- How many times a vertex is *touched* during inorder traversal from the start until the end?

Answer:

- Three times: from parent and from left + right children (even if one or both of them is/are empty/NULL)
- $O(3n) = O(n)$

# BST: Select/Rank Analysis

We have not explored the operations in detail yet

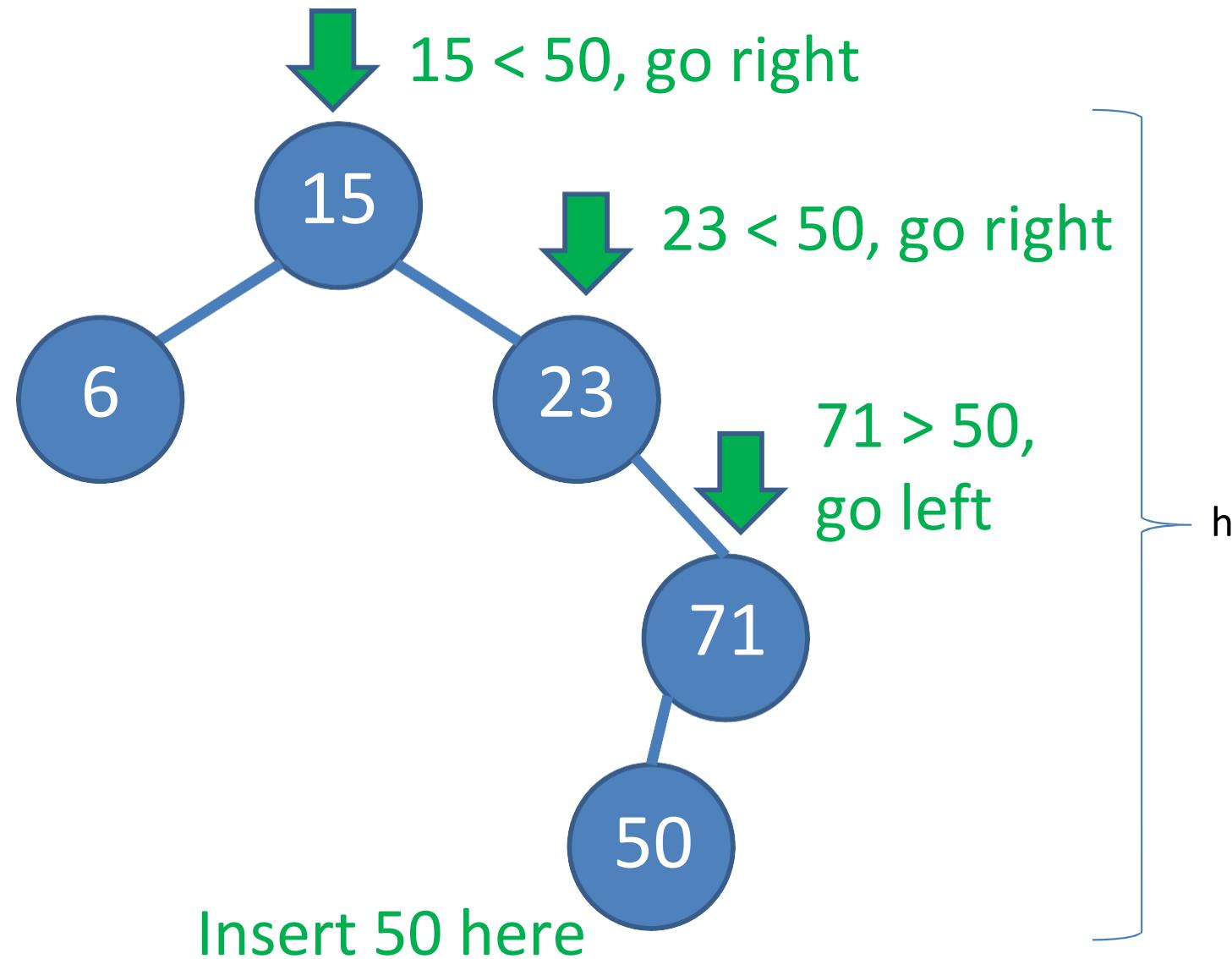
This will be discussed in more details in the next lecture

# BST: Insertion Analysis

insert (50)

Quick analysis:

insert also  
runs in **O(h)**



# Why successor of $x$ can be used for deletion of a BST vertex $x$ with 2 children?

Claim: Successor of  $x$  has at most 1 child!

- Easier to delete and will not violate BST property

Proof:

- Vertex  $x$  has two children
- Therefore, vertex  $x$  must have **a right child**
- Successor of  $x$  must then be the minimum of the right subtree
- A minimum element of a BST has no left child!!
- *So, successor of  $x$  has at most 1 child!* 😊

# BST: Deletion Analysis

Delete a BST vertex  $v$ , find  $v$  in  $O(h)$ , then three cases:

- Vertex  $v$  has no children:
  - Just remove the corresponding BST vertex  $v$   $\in O(1)$
- Vertex  $v$  has 1 child (either left or right):
  - Connect  $v.left$  (or  $v.right$ ) to  $v.parent$  and vice versa  $\in O(1)$
  - Then remove  $v$   $\in O(1)$
- Vertex  $v$  has 2 children:
  - Find  $x = \text{successor}(v)$   $\in O(h)$
  - Replace  $v.key$  with  $x.key$   $\in O(1)$
  - Then delete  $x$  in  $v.right$  (otherwise we have duplicate)  $\in O(h)$

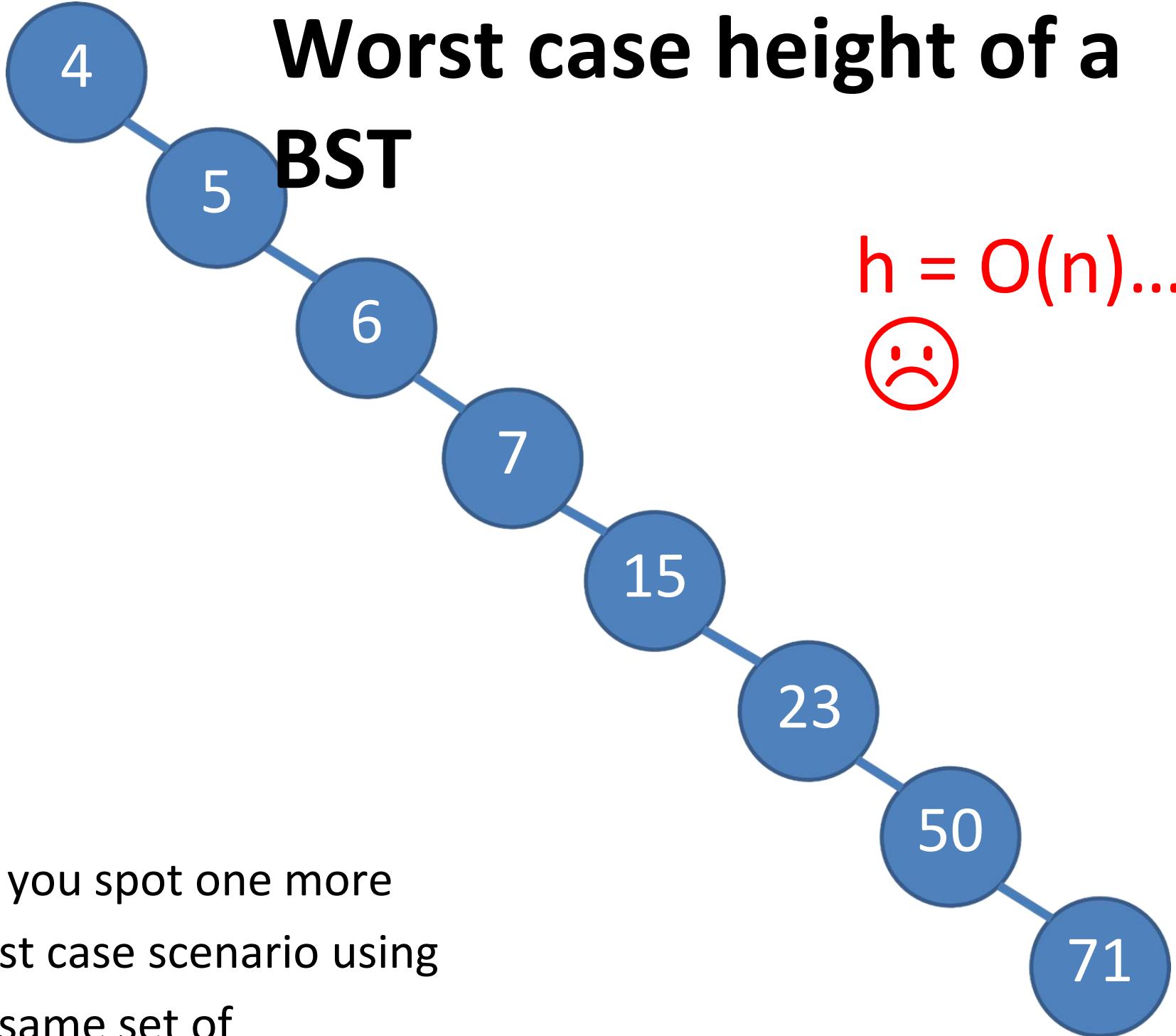
Running time:  $O(h)$

# Now, after we learn BST...

No	Operation	Unsorted Array	Sorted Array	BST
1	Search(age)	O(n)	O(log n)	O(h)
2	Insert(age)	O(1)	O(n)	O(h)
3	FindOldest()	O(n)	O(1)	O(h)
4	ListSortedAges()	O(n log n)	O(n)	O(n)
5	NextOlder(age)	O(n)	O(log n)	O(h)
6	Remove(age)	O(n)	O(n)	O(h)
7	GetMedian()	O(n log n)	O(1)	O(h)
8	Rank(age)	O(n log n)	O(log n)	?

It is all now depends on 'h'... ? next lecture





# Java Implementation

See BSTDemo.java (you can use this for PS2)

Concepts covered:

1. Java Object Oriented Programming (OOP)  
implementation of BST data structure
2. Java Error Handling: Throw & Catch Exception

# The Baby Names Problem (PS2)

Given a list of male and female baby names suggestions (*from your parents, in-laws, friends, yourself, Internet, etc*), your task is to answer some queries (see the next slide)

*This problem is always encountered by every parents with new baby*

(Including the search for baby Joshua name, born on 16 July 2014)



# PS2 Queries

(Note: Unlike this lecture with integer keys, the keys in PS1 are strings)

**Easy:** How many names start with a certain letter?

**Medium:** How many names start with a certain prefix?

*Definition: A prefix of a string  $T = T_0T_1\dots T_{n-1}$  with length  $n$  is string  $P = T_0T_1\dots T_m$  where  $m < n$ .*

**Hard:** Can you do it without Java API library code?

**CS2010R:** How many names have a certain substring?

*Definition: A substring of a string  $T = T_0T_1\dots T_{n-1}$  with length  $n$  is string  $S = T_iT_{i+1}\dots T_{j-1}T_j$  where  $0 \leq i \leq j < n$ .*

**You need efficient DS(es) to answer those queries**

# End of Lecture Quiz 😊

After Lecture 03, I will set a random test mode @ VisuAlgo to see if you understand BST

Go to:

<http://visualgo.net/test.html>

Use your CS2010 account to try the 5 BST questions (medium difficulty, 5 minutes)

Meanwhile, train first 😊

<http://visualgo.net/training.html>