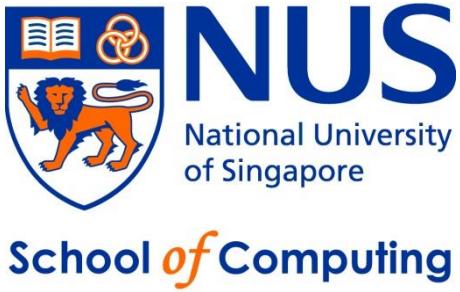
CS2010 – Data Structures and Algorithms II

Lecture 03 – Census Problem

chongket@comp.nus.edu.sg



Admin

- Submit only 1 source file for PSes (use the template given in write up)

- Please submit code for each lower subtask and not just the code for the highest subtask you attempt (**don't assume code for higher level subtask will always work for lower level subtask!**)

Outline

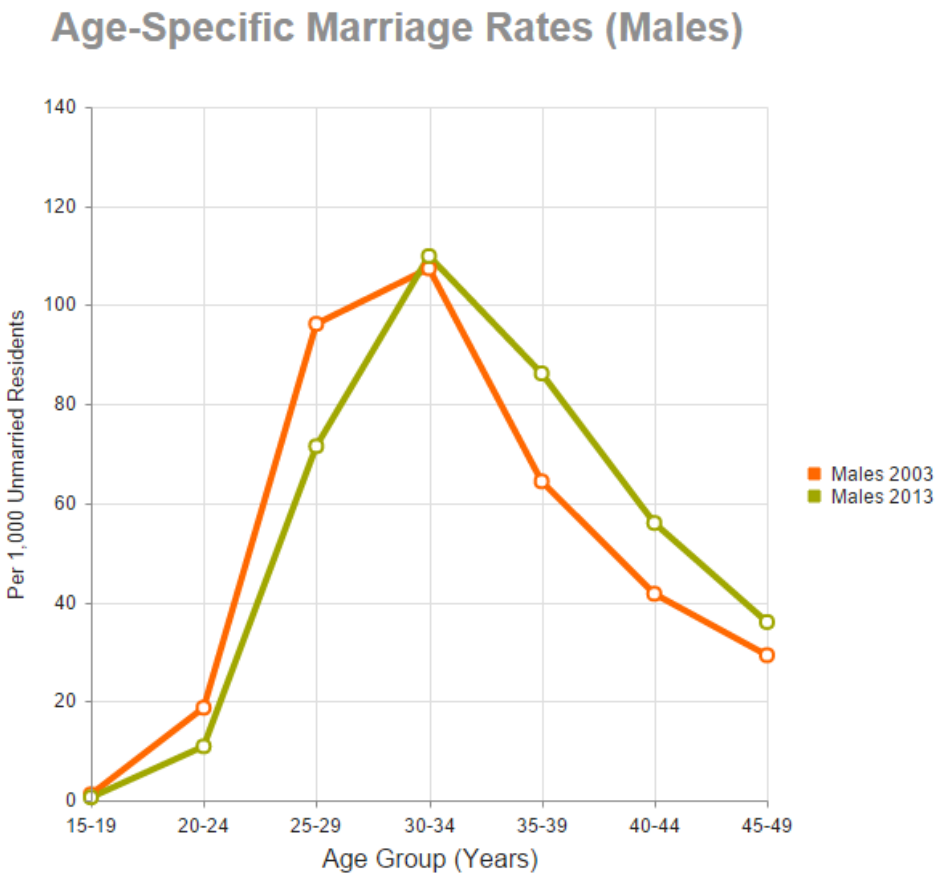
Motivation: Census Problem

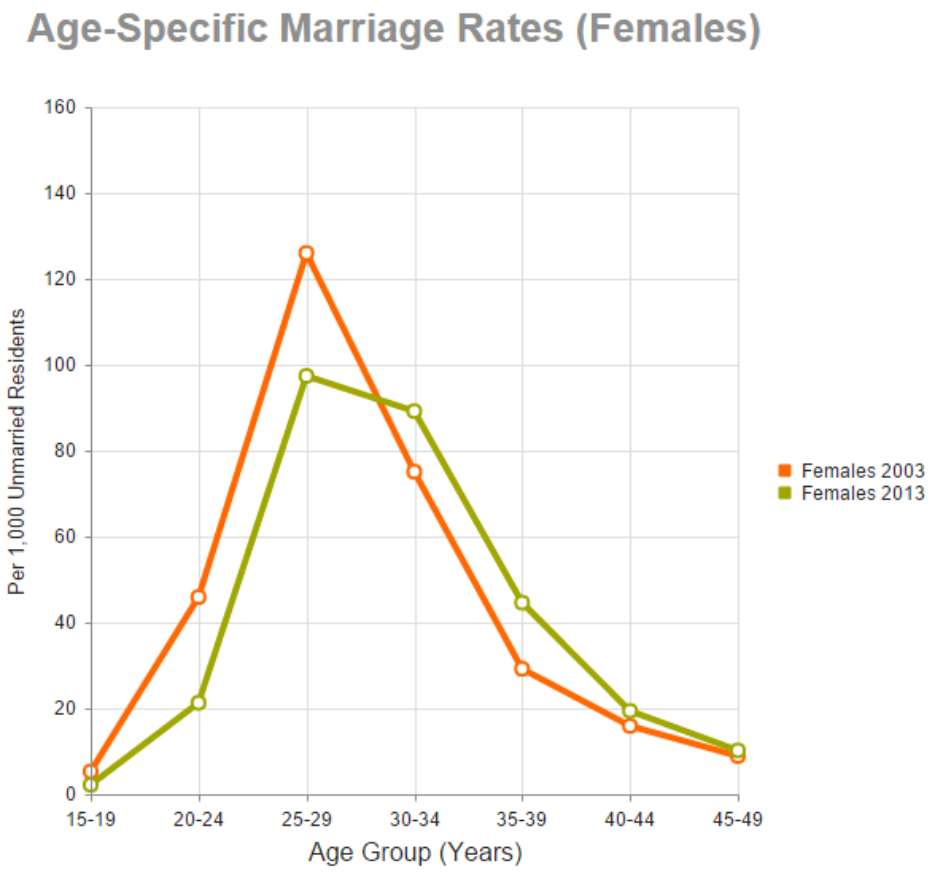
- Abstract Data Type (ADT) Table
- Solving Census Problem with CS1020 Knowledge
- The “performance issue”

Binary Search Tree (BST)

- Heavy usage of VisuAlgo Binary Search Tree Visualization
- Simple analysis of BST operations
- Java Implementation

Census is Important!





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

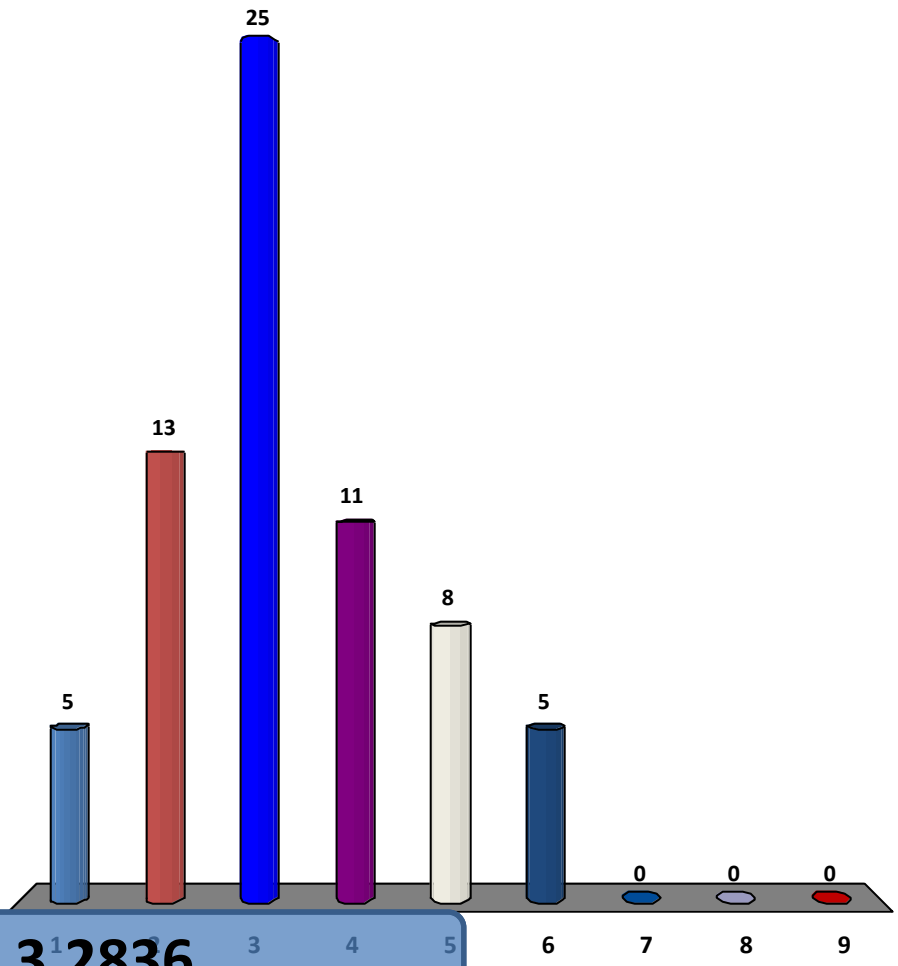
(If you know your enemies and know yourself, you will not be imperiled in a hundred battles)

Your Age (2016 data)

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

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

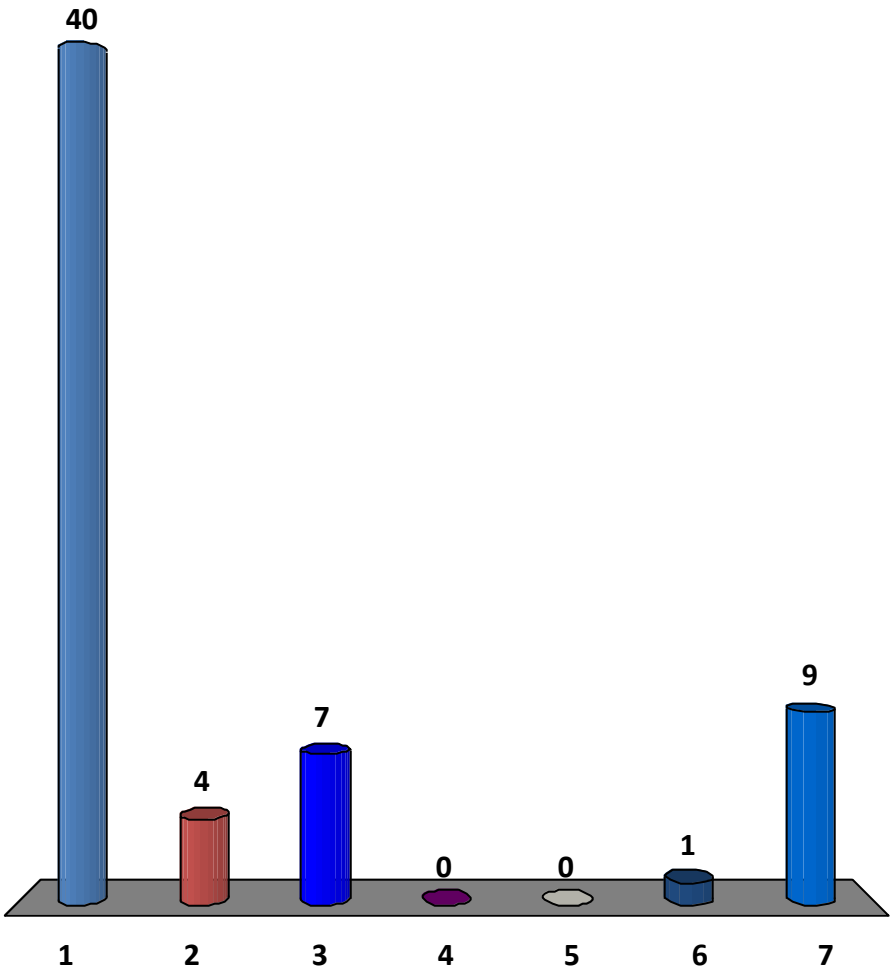
1. [24 … ∞)
2. [23 … 24)
3. [22 … 23)
4. [21 … 22)
5. [20 … 21)
6. [19 … 20)
7. [18 … 19)
8. [17 … 18)
9. [0 … 17)



Mean = 3.2836

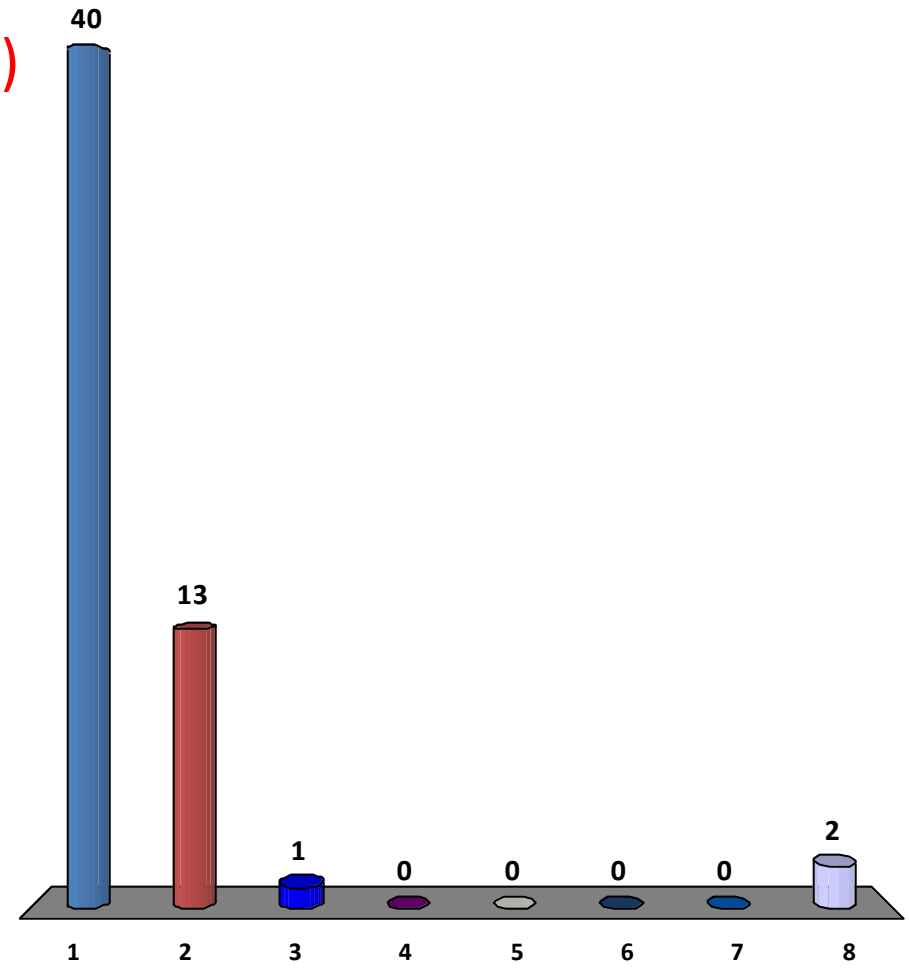
Your Major (2016 data)

1. Computer Science (CS)
2. Business Analytics (BZA)
3. Computer Engineering (CEG/CEC)
4. Comp. Biology (CB)
5. Information System (IS)
6. Science Maths (SCI)
7. None of the above :O



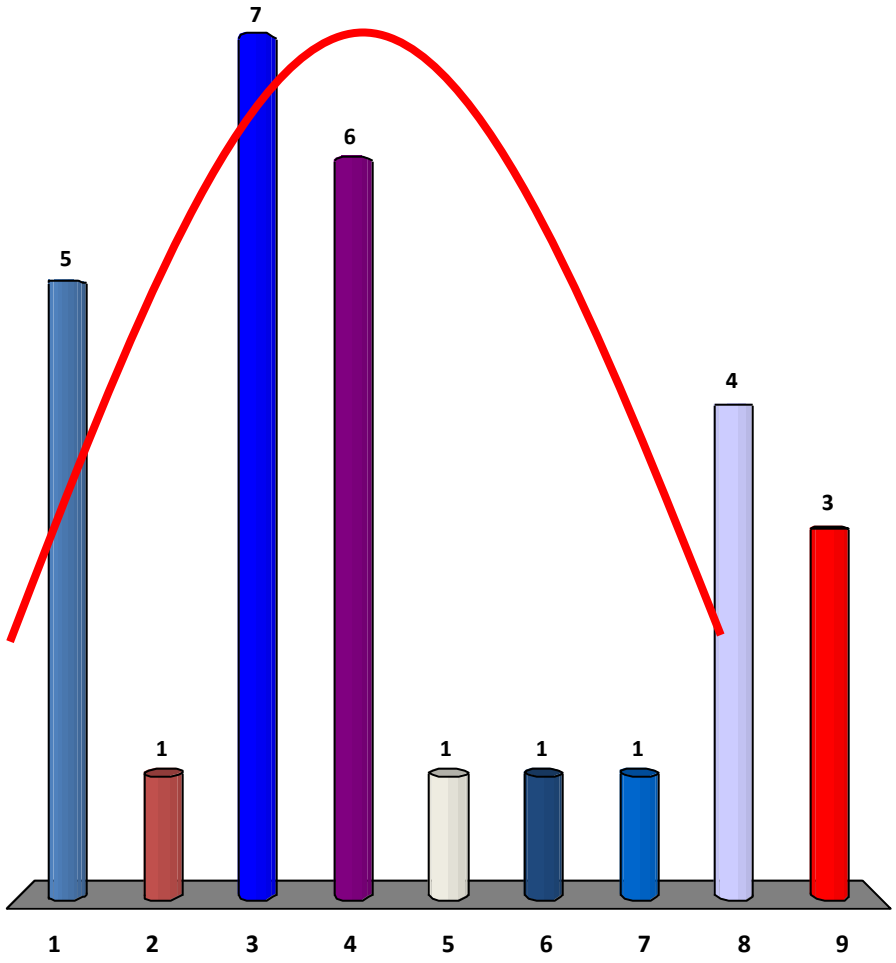
Your Nationality (2016 data)

1. Singaporean (should be ≥ 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) <- old 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

Statistical Analysis

Abstract Data Type (ADT) Table

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

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

Some required operations:

1. Search whether there is a student with a certain age?
2. Insert a new student (insert using 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 (remove using 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()	
5	NextOlder(age)	
6	Remove(age)	
7	GetMedian()	
8	NumYounger(age)	

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)	
2	Insert(age)	O(**N**)
3	FindOldest()	
4	ListSortedAges()	
5	NextOlder(age)	
6	Remove(age)	O(**N**)
7	GetMedian()	
8	NumYounger(age)	

With Just CS1020 Knowledge

No	Operation	Unsorted Array	Sorted Array
1	Search(age)	O(**N**)	
2	Insert(age)	O(**1**)	O(**N**)
3	FindOldest()	O(**N**)	
4	ListSortedAges		
5	NextOlder(age)		
6	Remove(age)		O(**N**)
7	GetMedian()		
8	NumYounger(age)		

Dynamic data structure operations

If N is large, our queries are slow…

O(N) versus O(log N): A Perspective

N = 8

$\log_2$ **N** = 3

N = 16

$\log_2$ **N** = 4

N = 32

$\log_2$ **N** = 5

Try larger **N**, e.g. **N** = 1 000 000…

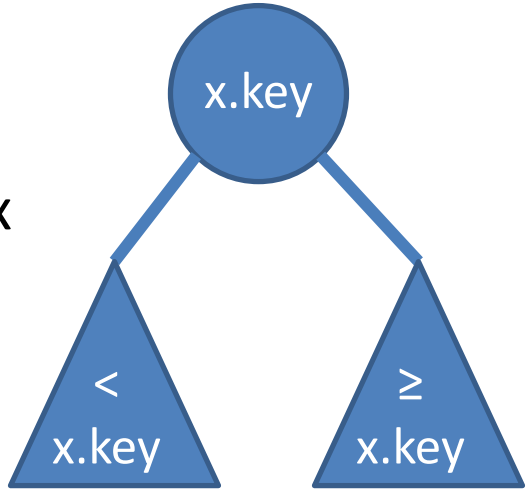
A Versatile, Non-Linear Data Structure

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:

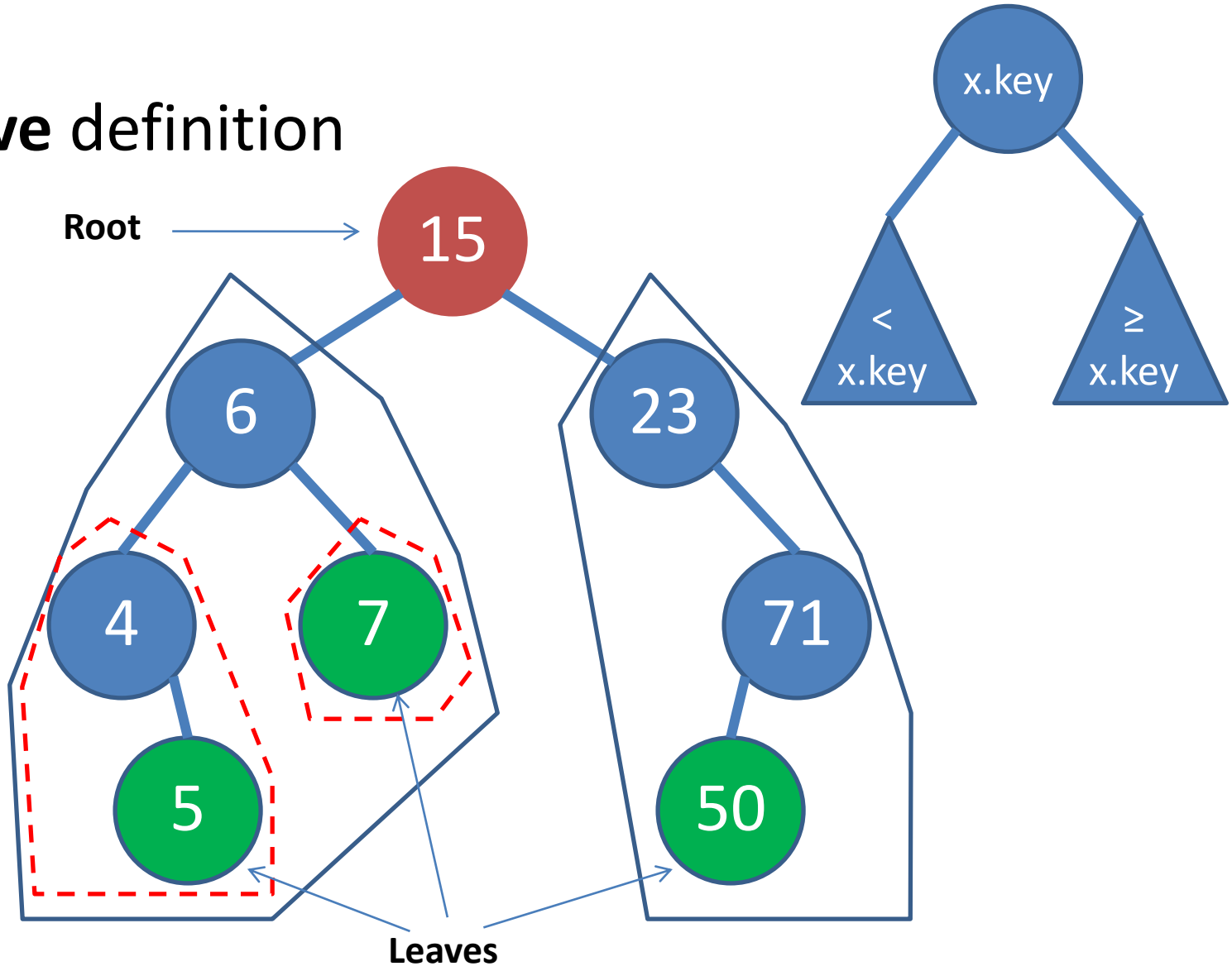
- For every vertex **x** and **y**

  y.key < x.key if **y** is in left subtree of **x**

  y.key ≥ x.key if **y** is in right subtree of **x**
- For simplicity, we assume that the keys are unique so that we can change ≥ to >

BST: An Example, Keys = Ages

Recursive definition

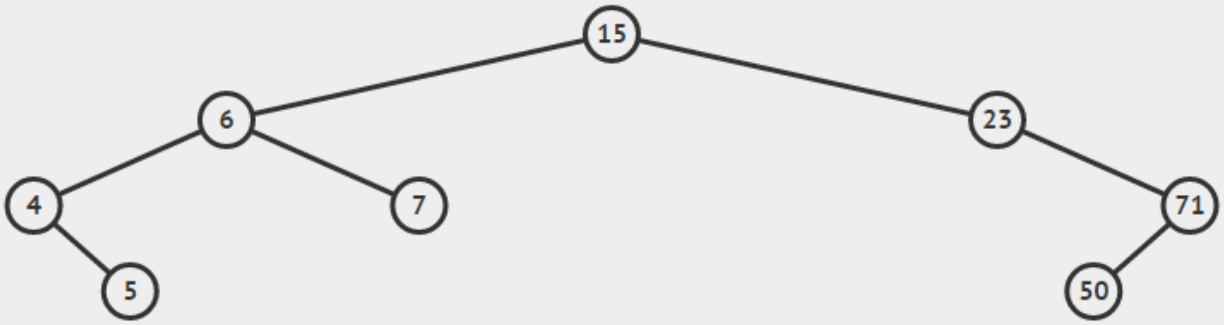


Steven

VisuAlgo - Binary Search T

visualgo.net/bst

en VISUALGO BINARY SEARCH TREE AVL TREE Tutorial Mode



X

View the BST visualisation here! Root vertex does not have a parent. There can only be one root vertex in a BST. Leaf vertex does not have any child. There can be more than one leaf vertex in a BST. All other vertices that are not root nor leaf are called the internal vertices. All vertices have at least 4 attributes: parent, left, right, key/value/data although not all attributes will be used for all vertices.

As we do not allow duplicate integer in this visualization (the full implementation must consider duplicate integers too), notice that for every vertex X, all vertices on the left subtree of X are smaller that X and all vertices on the right subtree of X are greater than X. This is called the 'BST property'.

All available operations on the BST/AVL Tree will be visualized/animated here.

< Prev Next >

Create
Search
Insert
Remove
Successor
Predecessor
Inorder Traversal

slow fast

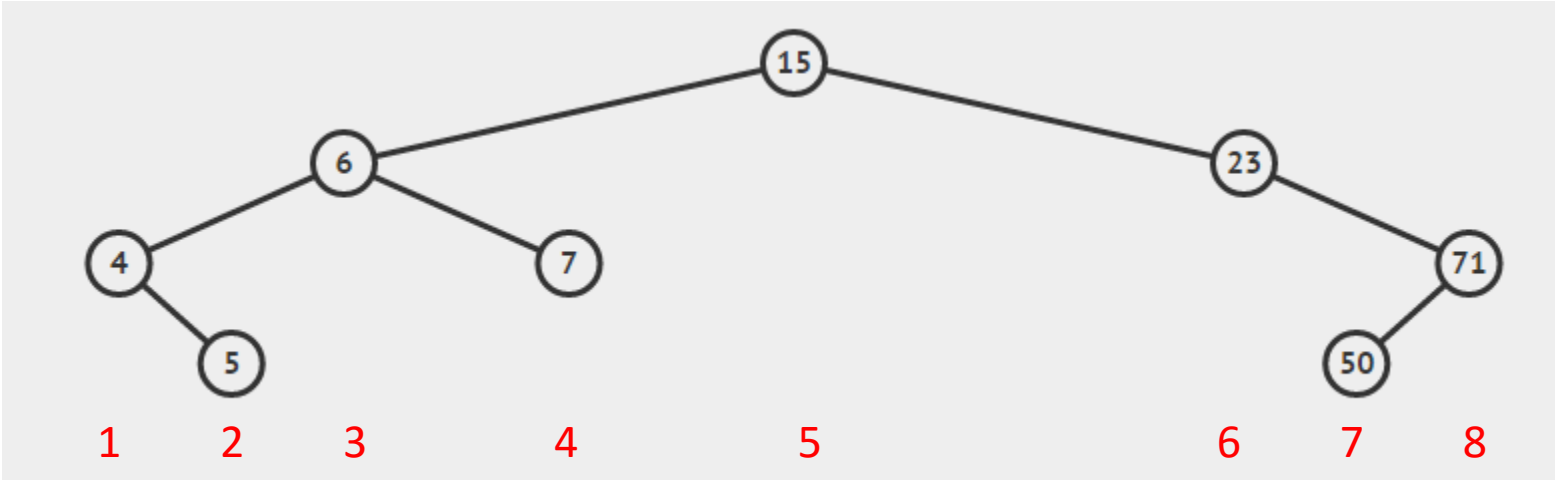
About Team Terms of use

BST: NEW Select/Rank Operations

These 2 operations are not yet in VisuAlgo BST visualization; for now, here are the concepts:

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

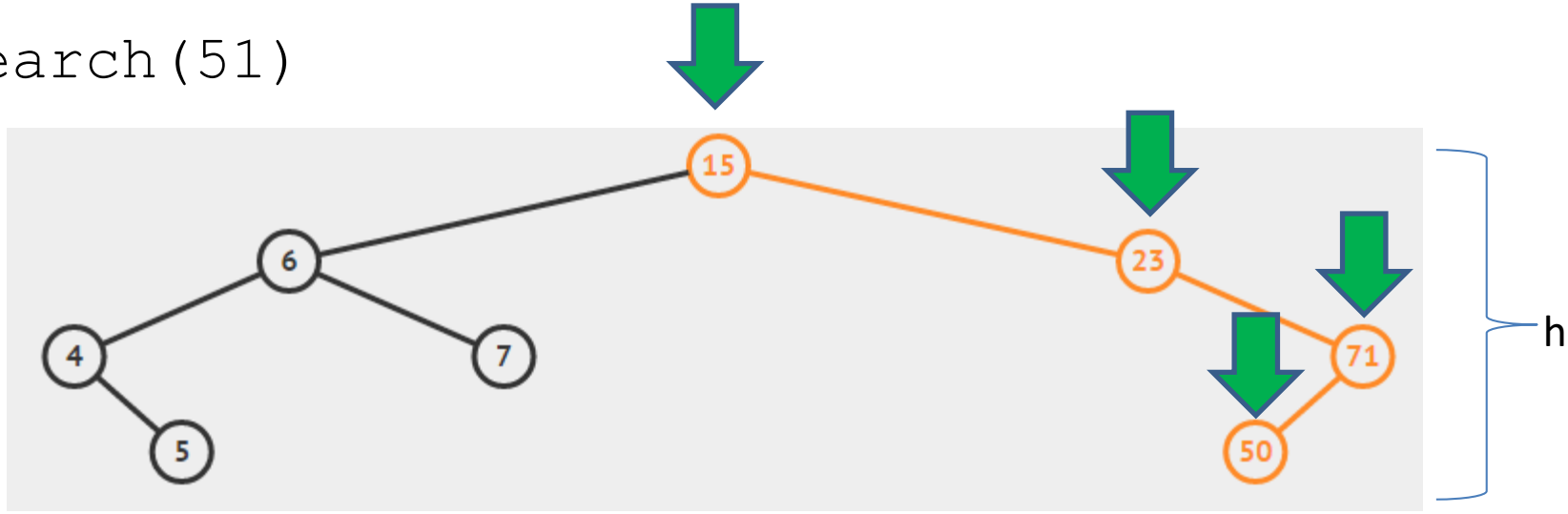
Compare with QuickSelect (tut01)



ANALYSIS OF BST OPERATIONS

BST: Search Analysis

`search(51)`



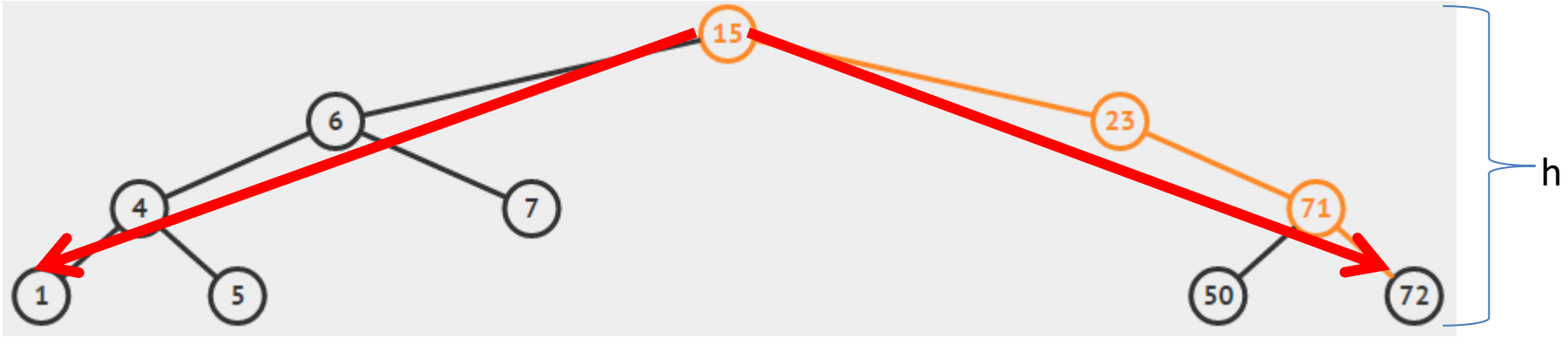
51 is not found ☹

Quick analysis:

`search` runs in **O(h)**

BST: FindMin/FindMax Analysis

`findMin()/findMax()`

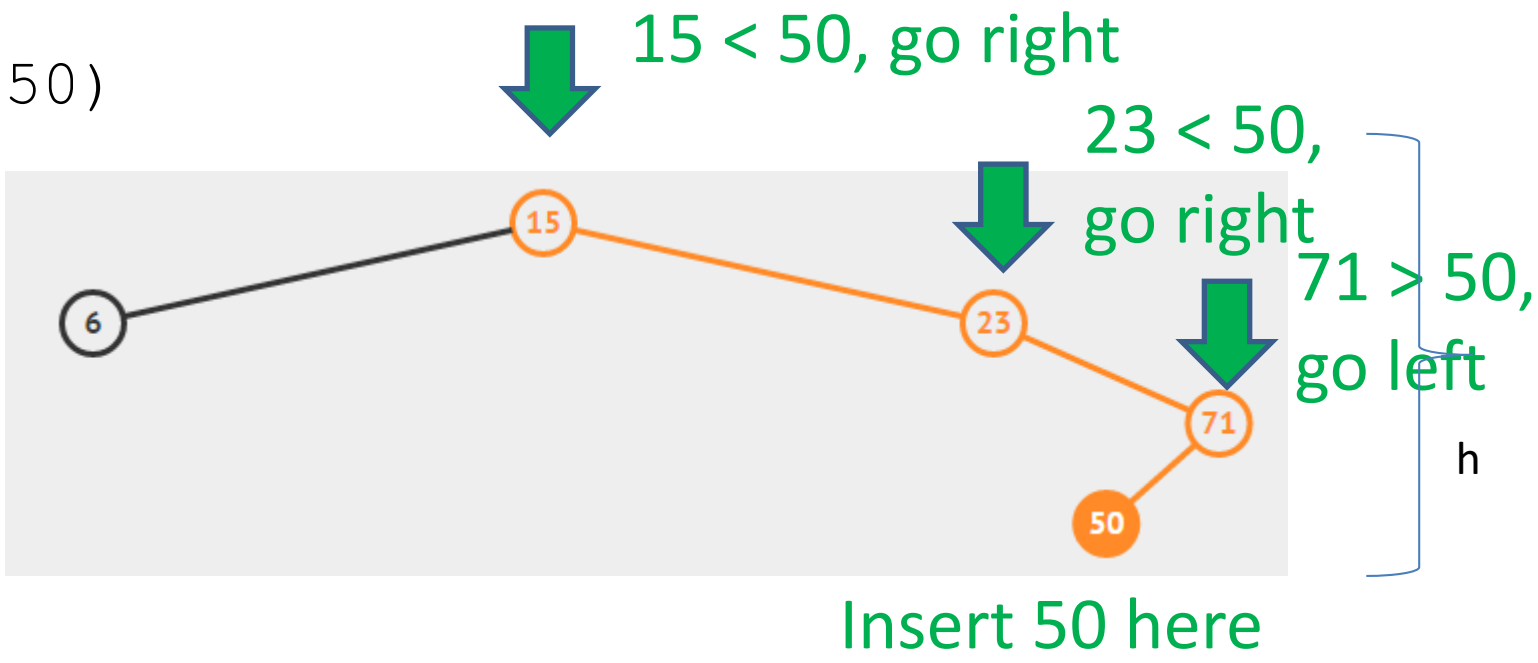


Quick analysis:

`findMin()/findMax` also runs in **O(h)**

BST: Insertion Analysis

`insert(50)`



Quick analysis:

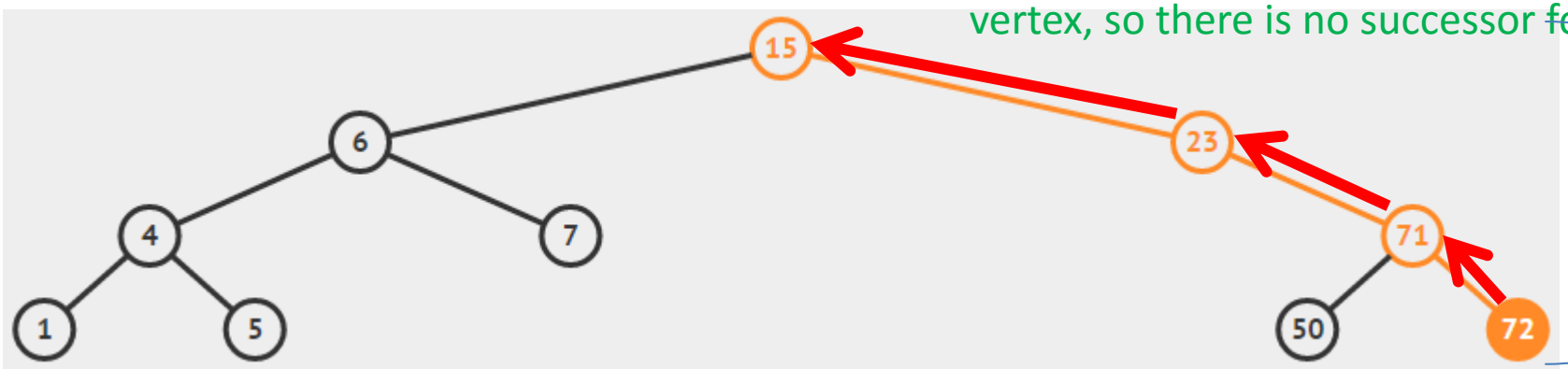
`insert` also runs

in **O(h)**

BST: Successor/Predecessor Analysis

Assumption, we already done an O(h) search(72) before

`successor(72)`



Keep going up until we make a 'right turn', but here we do not find such vertex, so there is no successor for 72

No right child

Quick analysis:

O(h) again, similarly for predecessor

BST: Inorder Traversal Analysis

Using a *new* analysis technique

Ask this question:

- How many times is a vertex *visited* 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(3***N***) = O(***N***)

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** → O(**1**)
- Vertex **v** has 1 child (either left or right):
  - Connect **v**.left (or **v**.right) to **v**.parent and vice versa → O(**1**)
  - Then remove v → O(**1**)
- Vertex **v** has 2 children:
  - Find **x** = successor(**v**) → O(**h**)
  - Replace **v**.key with **x**.key → O(**1**)
  - Then delete **x** in **v**.right (otherwise we have duplicate) → O(**h**)

Running time: O(**h**)

Why is successor of x 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: Select/Rank Analysis

We have not explored the operations in detail yet

This will be discussed in more details in the next lecture

Java Implementation

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

- But as you may have known after doing PS1, this free code will not go far…

Concepts covered:

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

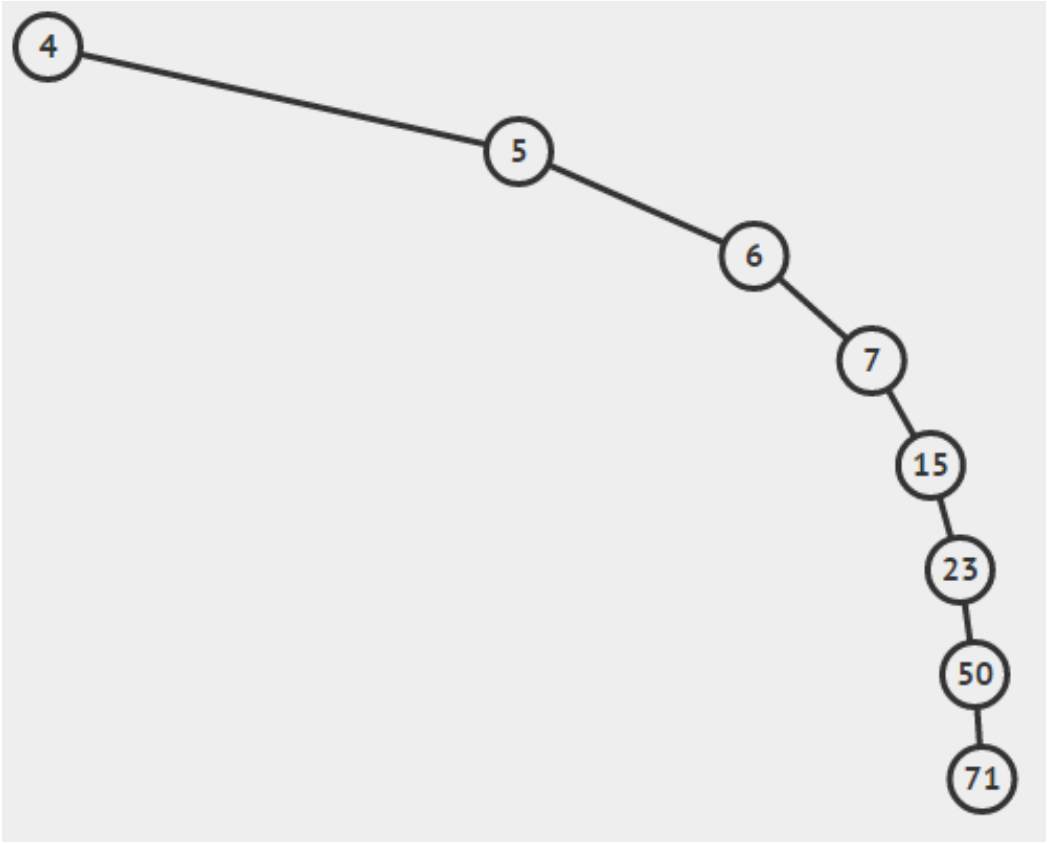
Now, after we learn BST…

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

It is all now depends on **'h'**… → next lecture ☺

Worst case height of a BST

h = O(**N**)… ☹



Can you spot one more worst case scenario using the same set of numbers?

Can we do better?

YES, h = O(log **N**) → next Lecture 04 ☺

Testing/Training BST knowledge on Visualgo ☺

- Go to https://visualgo.net/training
- Select Binary Search Tree and unselect the rest
- Set the question difficulty (go from easy to hard)
- Set the number of questions (try 5 to 10 questions)
- Set a suitable time limit (20 to 60 mins)