CMPT 225

Lecture 13 – Binary Trees and Binary Search Trees

Learning Outcomes

- At the end of the next few lectures, a student will be able to:
  - Define the following data collections:
    - Binary search tree
    - Balanced binary search tree (AVL)
    - Binary heap

    as well as demonstrate and hand-trace their operations
  - Implement the operations of binary search tree and binary heap
  - Implement and analyze sorting algorithms: tree sort and heap sort
  - Write recursive solutions to non-trivial problems, such as binary search tree traversals

Last Lecture

- We saw how to …
  - Define some tree-related terms and concepts

Today's menu

- Describe **binary tree** and its properties
- Describe **binary search tree** and its properties
- Given a **binary search tree**, perform some operations such as:
  - Insert an element (a node containing an element)
  - Retrieve (get) an element

We shall now focus on **2-ary** tree i.e., binary tree

Binary Tree: **N-ary Tree** where **N** = 2

- **Property 1** of Binary tree:

Range of **n** (# of nodes) in a binary tree of height **H**

  - If a binary tree has height **H**, then it can have between **H** and **2^H - 1 nodes** => **n** = **[H .. 2^H – 1]**

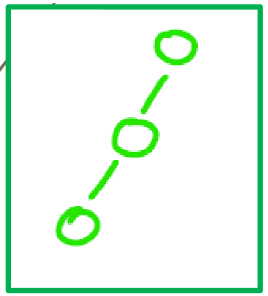
Expressing **n** as a function of height **H**

- Minimum **n** (# of nodes) a binary tree with height **H** can have is: **H**
- Maximum **n** (# of nodes) a binary tree with height **H** can have is: **2^H -1**

Range of **n** nodes in a **binary tree** of height **H**: $[H .. 2^H - 1]$

- For example, if **H** = 3

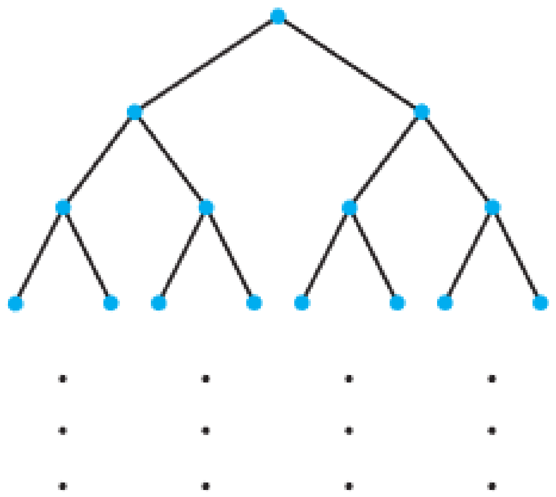
n = 3 **n = 4** **n = 5** **n = 6** **n = 7**



Binary Tree: **N-ary Tree** where **N** = 2

Expressing **n** as a function of height **H**

FIGURE 15-9 Counting the nodes in a full binary tree of height h



Level	Number of nodes at this level	**n** Total number of nodes at this level and all previous levels
1	$1 = 2^0$	$1 = 2^1 - 1$
2	$2 = 2^1$	$3 = 2^2 - 1$
3	$4 = 2^2$	$7 = 2^3 - 1$
4	$8 = 2^3$	$15 = 2^4 - 1$
⋮	⋮	⋮
h	2^{h-1}	$2^h - 1$

Binary Tree: **N-ary Tree** where **N** = 2

- **Property 2** of Binary tree:

  Range of heights of a binary tree with **n** nodes

  - If a binary tree has **n** nodes, then its height **H** can be between **n** and **ceil($\log_2$ (n + 1))**

Expressing height **H** as a function of **n**

- Minimum height **H** a binary tree with **n** nodes can have is: **ceil($\log_2$ (n + 1))**
- Maximum height **H** a binary tree with **n** nodes can have is: **n**

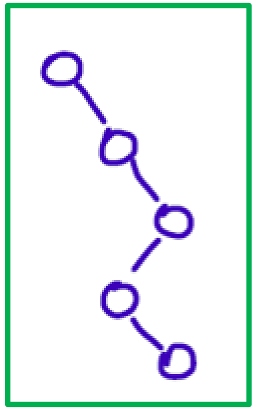
Range of heights **H** of a binary tree with **n** nodes: **[ceil($\log_2$ (n + 1)) .. n]**

- For example, if **n = 5**

H = 3

H = 4

H = 5



How to expressing height **H** as a fcn of **n** in the range [**ceil($\log_2$ (n + 1))** .. n]

Start with **Property 1**: maximum **n** (# of nodes) a binary tree with height **H** can have is: **2^H -1**

$$n = 2^H - 1$$
$$n + 1 = 2^H - 1 + 1$$
$$n + 1 = 2^H$$
$$\log_2 (n + 1) = \log_2 2^H$$
$$\log_2 (n + 1) = H$$

So, where does the function **ceil()** come from?

Well, remember that **n** in the above equation is the maximum **n** (# of nodes) a binary tree with height **H** can have.

But, we can create several binary trees of the same height **H** with different values of **n**, not just its maximum. For example, if **H** = 3, then the maximum **n** is 7, but we can create binary trees of height **3** with other values of **n**, for example, 4, 5, 6 (aside from **n** = **H**). How can the above equation express this?

And we also need to produce an integral value for **H**.

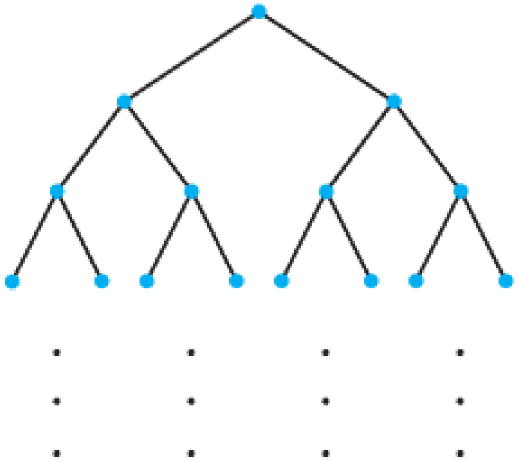
Therefore, using the ***ceiling*** function in the above equation not only produces an integral value for **H**, but also produces the same **H** for various values of **n**.

Binary Tree: **N-ary Tree** where **N** = 2

- **Property 3** of Binary tree:

If a binary tree has height **H** with **H** levels, where each level has the level number **L** (from 1 to **H**), then each level of this binary tree can have a minimum of **1** node to a maximum of **2^{L-1}** nodes.

FIGURE 15-9 Counting the nodes in a full binary tree of height h



Level	Number of nodes at this level	Total number of nodes at this level and all previous levels
1	$1 = 2^0$	$1 = 2^1 - 1$
2	$2 = 2^1$	$3 = 2^2 - 1$
3	$4 = 2^2$	$7 = 2^3 - 1$
4	$8 = 2^3$	$15 = 2^4 - 1$
.	.	.
.	.	.
.	.	.
h	2^{h-1}	$2^h - 1$

What can we do with a Binary Tree?
Binary Tree Operations!

- How would we insert the following elements (in this order) D, E, B, G, A, C into a binary tree?

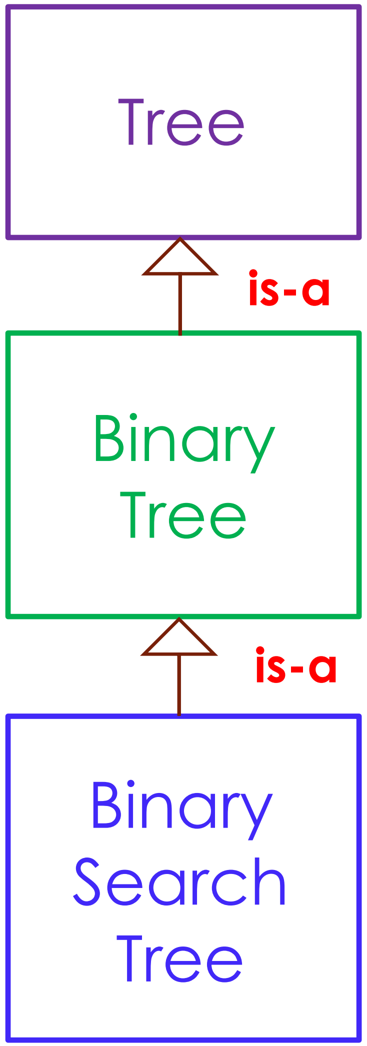
- How would we remove an element from a binary tree?

Binary Search Tree (BST)

- Definition: A Binary Search Tree is a binary tree in which an element stored in a node has a search key value **X** and satisfies the following constraint:

- What about duplication?
- Answer: Commonly stored in right subtree, but it is up to the designer of such data collection (design decision)

Inheritance Relationship (UML class diagram)



Examples of Binary Search Trees

Insert

```
if binary search tree empty
   insert new element in root
otherwise
   if new element < element stored in root
      insert new element into left subtree
   else
      insert new element into right subtree
```

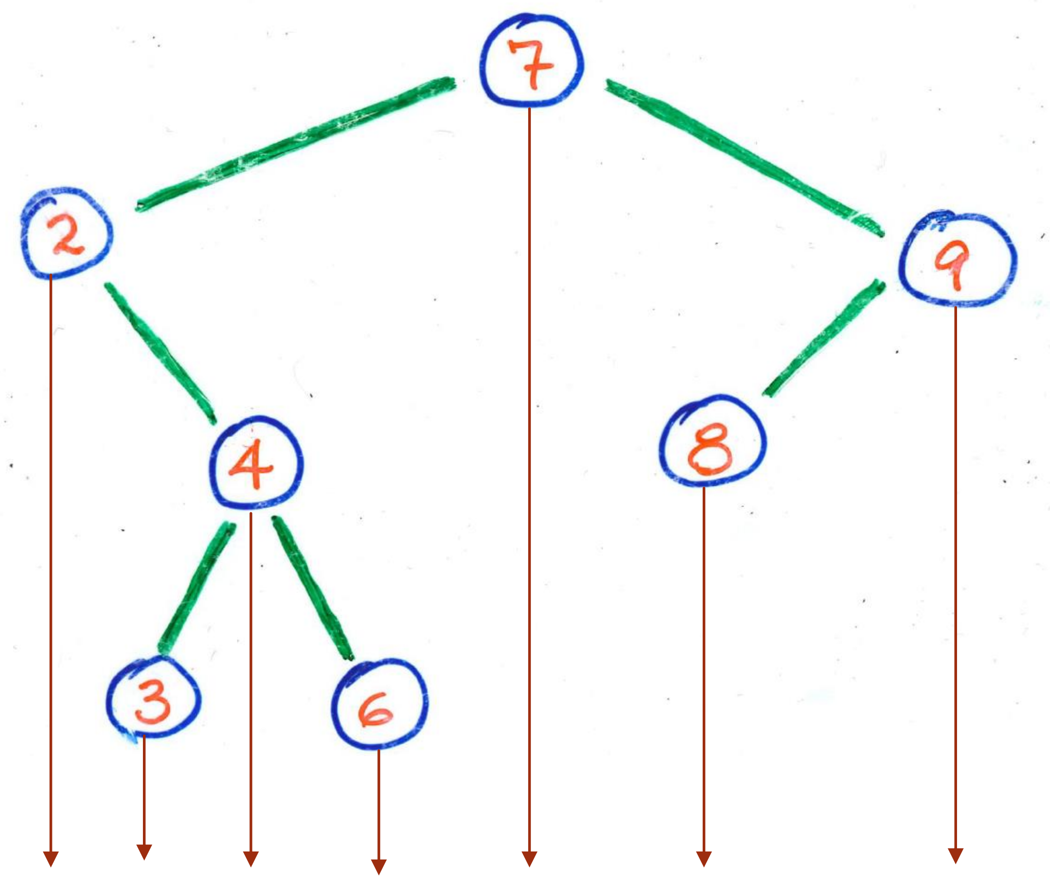
Let's not forget to `elementCount++`

`element` means search key value of element we are inserting

Let’s try!

- Let’s insert 7, 2, 4, 6, 9, 8, 3 into a binary search tree:

Do we still have a binary search tree?
A trick!



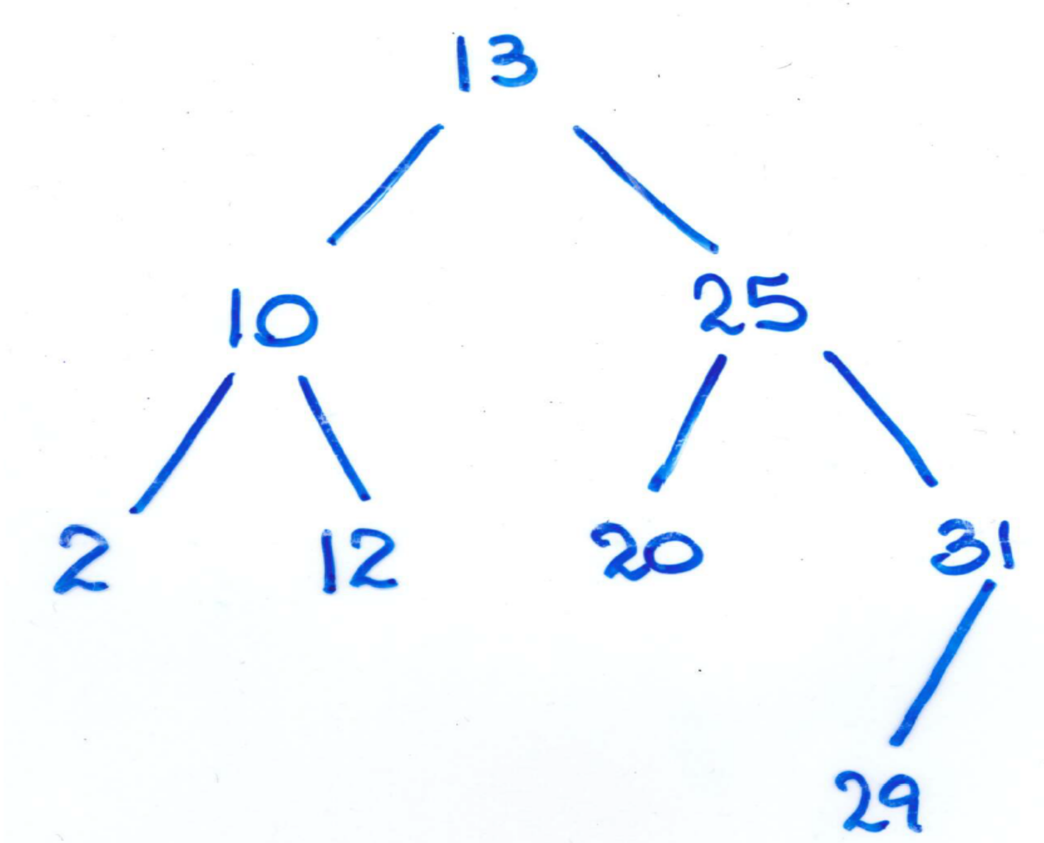
Retrieve (get) - search

```
if binary search tree empty
   target element not there!
if target element == element stored in root
   return element stored in root
otherwise
   if target element < element stored in root
     search left subtree
   else
     search right subtree
```

`element` means search key value of element we are looking for

Let's try!

- Let's retrieve ____ from this binary search tree:



√ Learning Check

- ✓ We can now …
  - ✓ Describe binary tree and its properties
  - ✓ Describe binary search tree and its properties
  - ✓ Given a **binary search tree**, perform some operations such as:
    - ✓ Insert an element (a node containing an element)
    - ✓ Retrieve (get) an element

Next Lecture

- Given a binary search tree (BST), perform some operations such as:
  - ~~Insert an element (a node containing an element)~~
  - ~~Retrieve (get) an element~~
  - Remove an element
  - Find successor of an element
  - Find predecessor of an element
  - Traverse the BST
  - Get the number of elements stored in BST
  - Find minimum element value stored in BST
  - Find maximum element value stored in BST