

MS-CS Course Note (Non-Credit Course 3)

Mark Zhou

July 2024

This is my course note on “Trees And Graphs: The Basics” provided by Colorado University of Boulder. This is a non-credit prep course for an MS-CS degree.

Contents

1	Binary Search Trees	3
1.1	Basic Concepts	3
1.2	The Height of a Binary Search Tree	5
1.3	Basics of Binary Search Tree Quiz	8
1.4	Insertion and Deletion in a Binary Search Tree	10

1 Binary Search Trees

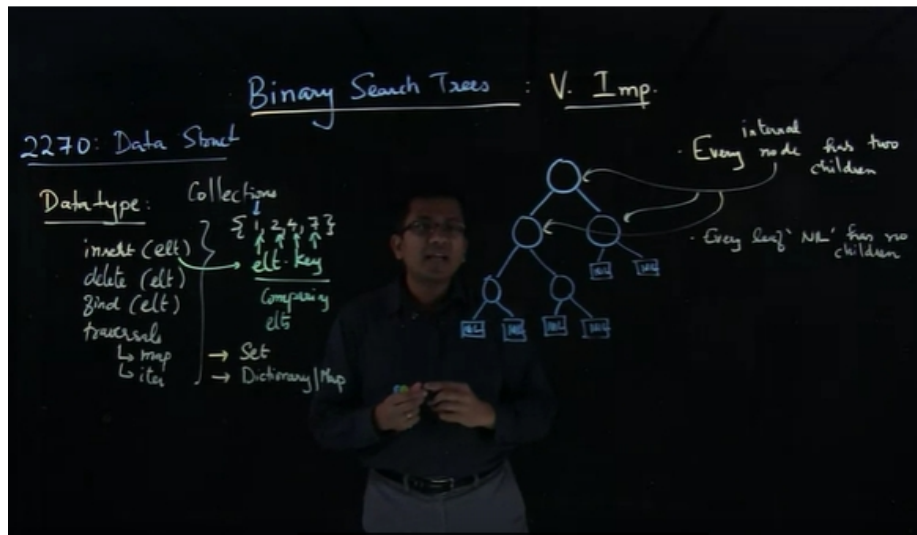
1.1 Basic Concepts

Binary search tree is a binary tree is a kind of data type with set of data elements without repetition.

We can insert, delete, search, and traverse the data elements in a binary search tree.

For each element in it, there will be a key of the element, which will always be a number.

With this setting in place, we can always comparing different elements by comparing their keys, even if the elements are not numbers.



In the figure, we have a binary search tree with some nodes and leaves. Every node has two children nodes and those leaves, which

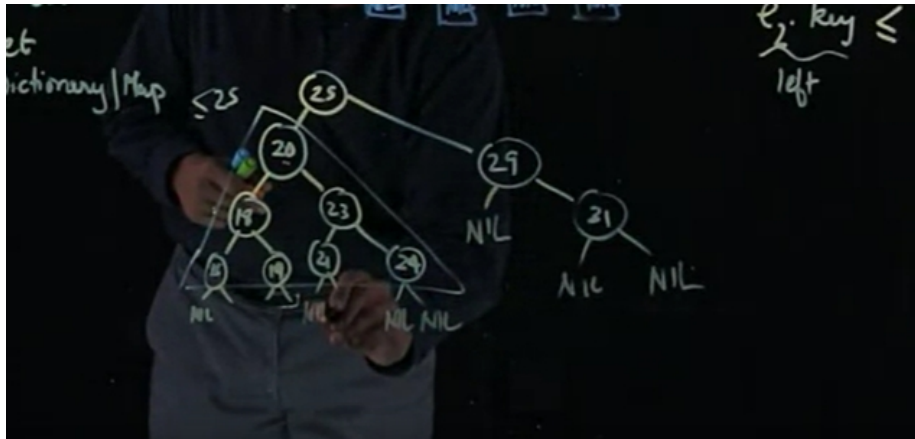
have no children nodes, are called nil nodes.

Every node has an element with a key, and the key of the left child node is always $<$ the key of the parent node, and the key of the right child node is always $>$ the key of the parent node.

The left and right child nodes are also binary search trees.

That is to say, the keys are always in a sorted order regardless of the structure of the tree. When we move the elements around, the keys will be different for each elements, in order to remain in the sorted order.

The leaves have no elements.



When there is a node with the key 25, every node in the left subtree will have a key $<$ 25, and every node in the right subtree will have a key $>$ 25.

The rule will also apply to all those subtrees.

Example:

```
    25
   /  \
  15   50
 / \  / \
10 22 35 70
```

Question:

Binary Search Trees may look similar to Heaps, but it is important to consider their differences.

In a Min-Heap, the smallest element must be the root node of the tree.

In a Binary Search Tree, on the other hand, how would we find the smallest element?

A: We would traverse the left subtree of the root node until we reach a leaf node, which means a node with a NIL as its left child.

1.2 The Height of a Binary Search Tree

The height of a binary search tree is the number of edges on the longest path from the root node to a leaf node.

We will define the height of a leaf node as 0. Then the height of number 25, a.k.a. the root node of the below binary search tree is 2.

Example:

```
    25          -> height = 2
   /  \
  15   50      -> height = 1
 /  \  /  \
10 22 35 70  -> height = 0, since they are leaf nodes
```

Let's assume we have a balanced binary search tree with n internal nodes.

On each layer from the root node, there'll be 2^0 nodes, 2^1 nodes, 2^2 nodes, \dots , 2^h nodes.

The total number of nodes in the tree will be $2^0 + 2^1 + 2^2 + \dots + 2^h = 2^{h+1} - 1$.

So, the height of the tree will be $h = \log_2(n + 1) - 1$.

In the sense of big O notation, the height of a binary search tree is $O(\log_2(n))$, in a balanced binary search tree scenario.

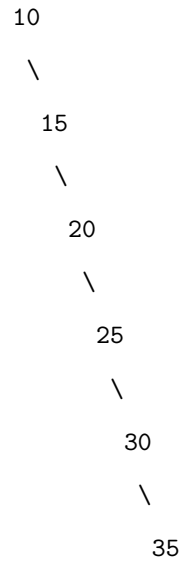
For example, $\log_2^8 = 3$, so the height of a binary search tree with 8 nodes is 3.

$\log_2^{15} = 4$, so the height of a binary search tree with 15 nodes is 4.

In the worst case scenario, where the binary search tree is not balanced, the height of the tree will be $O(n)$.

That is to say, the tree will be a linked list looks like this;

Linked List Example:



In this case, the height of the tree is 6, which is equal to the number of nodes in the tree.

The height of the tree is $O(n)$, which is the worst case scenario.

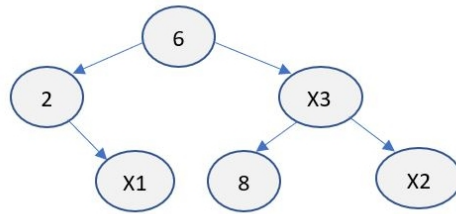
Normally, we will have something in between the best case scenario and the worst case scenario, $O(\log(n)) < height < O(n)$.

1.3 Basics of Binary Search Tree Quiz

1. Consider the following binary search tree below with missing values $X1$, $X2$ and $X3$.

1 / 1 point

Note that the leaves labeled NIL are not shown, but please assume that they exist.



Select all true statements about the tree.

- ☐ $X1$ can be any value less than or equal to 6.
- ☒ $X1$ can be set to the number 5 while remaining a valid binary search tree.

✓ **Correct**
 $X1$ must also be ≥ 2 since it is the right child of 2, and $X1 \leq 6$ since it is in the left subtree of the root 6. Therefore, 5 is a possible value.

- ☐ $X3$ can be any number ≥ 6 .
- ☒ $X3$ can be any number ≥ 8 and $\leq X2$.

✓ **Correct**
Correct

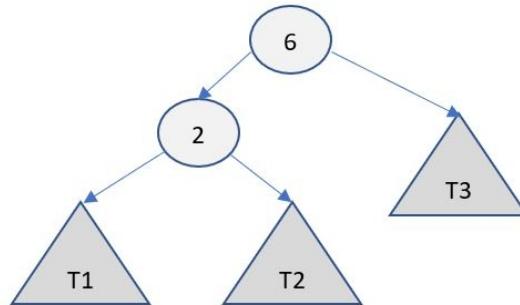
- ☒ $X2$ must have a value ≥ 8 and $\geq X3$.

✓ **Correct**
Correct

- ☒ The height of the root node is 3.

✓ **Correct**
Correct. Note that the root has a longest path of length 3 to a leaf.

2. Consider the following binary search tree with subtrees shown below. Select all true statements about it.



- ☒ Every node in $T1$ must have value ≤ 2 .

✓ **Correct**

Correct since $T1$ is the left subtree of node 2.

- ☒ Every node in $T2$ must have key ≥ 2 and ≤ 6 .

✓ **Correct**

Correct since $T1$ is in the right subtree of node 2 and left subtree of the node 6.

- ☐ If the node with key 25 is found in the tree, we will find it in subtree $T2$.

- ☐ If the node with key -10 is to be found in the tree, it can be found in subtree $T2$.

- ☒ If the node with key 7 is to be found in the tree, it will be found in $T3$.

✓ **Correct**

Correct since $7 > 6$ it will be found in the right subtree of the root node 6.

- ☒ If the height of subtree $T1$ is 4 and that of subtree $T2$ is 2 then the height of node labeled 2 is 5.

✓ **Correct**

Correct since $\max(4, 2) + 1 = 5$

3. Select all correct statements from the list below about binary search trees.

0.6666666666666666
/ 1 point

- ☐ In a fully balanced binary search tree with n total nodes (internal and leaf nodes), where $n = 2^k - 1$ for some k , we will have $(n + 1)/2$ leaves.
- ☒ In the worst case, a binary search tree with n internal nodes can have height n .

✓ **Correct**
Correct. Every node in the tree has a single child in the worst case

- ☐ Assuming that all keys are distinct, the key at the root is the median among all keys of the binary search tree.

可能是这个

You didn't select all the correct answers

1.4 Insertion and Deletion in a Binary Search Tree

We can insert a new element into a binary search tree by comparing the key of the new element with the key of the root node.

If the key of the new element is less than the key of the root node, we will insert the new element into the left subtree.

If the key of the new element is greater than the key of the root node, we will insert the new element into the right subtree.

We will repeat the process until we reach a leaf node.

For example, we have a binary search tree with the following nodes;

Example:

```

      25
     /  \
    15   50
   / \  / \
  10 22 35 70

```

If we want to insert a new element with the key 40, we will compare 40 with 25, and then 40 with 50, and then 40 with 35. Since 40 is greater than 35, we will insert 40 as the right child node of 35.

TBC: Binary search tree: insertion and deletion video.