**Trees**

**Max Depth of Binary Tree:**

Given the root of a binary tree, return *its maximum depth*.

A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.

**Example 1:**

A diagram of a network

Description automatically generated



Approach 1: Recursive DFS -> return 1 + max(DFS(left), DFS(right))

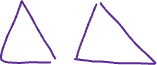
Complexity: time and space O(n) – W.C. traverse unbalanced tree

Base cases

1. Empty tree – return 0
2. 1 node no children – return 1

A diagram of a network

Description automatically generated



A computer screen shot of a code

Description automatically generated

Approach 2: Iterative BFS – use queue. Start queue with root. Advance to next level, replace node in queue with its children. Continue doing this until you reach base which is node has no children.

Complexity: time and space O(n)



Traverse tree level by level



A diagram of a network

Description automatically generated Count number of levels which represents the max



A screenshot of a computer code

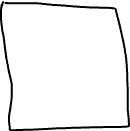
Description automatically generated

Approach 3: Iterative DFS – Use stack. Implement using pre-order traversal.



A diagram of a network

Description automatically generated



A computer screen shot of a code

Description automatically generated

<https://www.youtube.com/watch?v=hTM3phVI6YQ>

**Validate Binary Search Tree:**

Given the root of a binary tree, *determine if it is a valid binary search tree (BST)*.

A **valid BST** is defined as follows:

* The left subtree of a node contains only nodes with keys **less than** the node's key.
* The right subtree of a node contains only nodes with keys **greater than** the node's key.
* Both the left and right subtrees must also be binary search trees.

A diagram of a network

Description automatically generated



**Input:** root = [5,1,4,null,null,3,6]

**Output:** false

**Explanation:** The root node's value is 5 but its right child's value is 4.

Approach: Use DFS recursively. Initialize left and right boundaries as -+inf.

Complexity: time O(n), space O(height)

A screenshot of a computer code

Description automatically generatedA diagram of a network

Description automatically generated



<https://www.youtube.com/watch?v=s6ATEkipzow>

**Symmetric Tree:**

Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

**Example 1:**

A diagram of a tree

Description automatically generated



**Input:** root = [1,2,2,3,4,4,3]

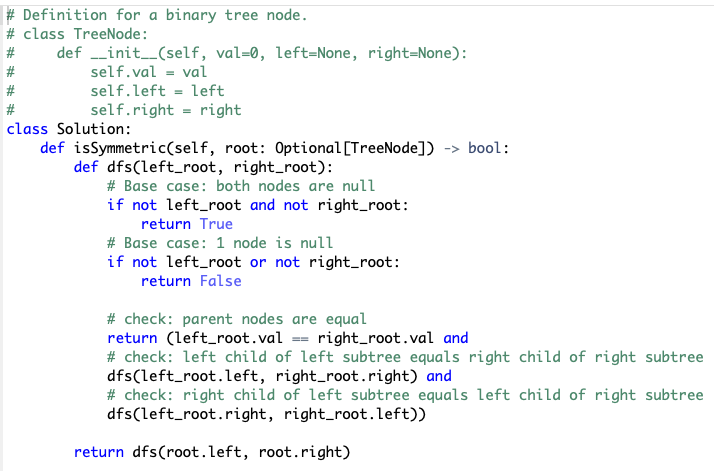
**Output:** true

Approach: Run DFS on left subtree and DFS on right subtree simultaneously

Complexity: time O(n), space O(height)

All the work is performed in the subtrees as the root is always symmetric with itself

* First, check that the parents in the subtree are the same (red)
* Then move to the next level and check the children
  + Check if left child in left subtree equals right child of right subtree (purple)
  + Check if right child in left subtree equals left child in right subtree (green)



<https://www.youtube.com/watch?v=Mao9uzxwvmc>

**Binary Tree Level Order Traversal:**

Given the root of a binary tree, return the level order traversal of its nodes' values. (i.e., from left to right, level by level).

**Example 1:**

A diagram of a tree

Description automatically generated



**Input:** root = [3,9,20,null,null,15,7]

**Output:** [[3],[9,20],[15,7]]

Approach: Traverse tree one level at a time from left to right. Add results to separate list then combine sub-lists for output. Use BFS to do this. Implement with queue and follow FIFO. Initialize queue with root. Pop root and add it to sub-list. Add root’s children from left to right to queue. Repeat until there are no nodes left to traverse so queue is empty.

Complexity: Time and space O(n)

A diagram of a tree

Description automatically generated

7 doesn’t have children to add to q



15 doesn’t have children to add to q

20 has children to add to q



9 doesn’t have children to add to q



A screenshot of a computer code

Description automatically generated

<https://www.youtube.com/watch?v=6ZnyEApgFYg>

**Convert Sorted Array to Binary Search Tree:**

Given an integer array nums where the elements are sorted in **ascending order**, convert it to a**height-balanced** binary search tree.

**Example 1:**

A diagram of a network

Description automatically generated



**Input:** nums = [-10,-3,0,5,9]



**Output:** [0,-3,9,-10,null,5]

**Explanation:** [0,-10,5,null,-3,null,9] is also accepted:

A diagram of a network

Description automatically generated

Approach: Because number in the middle of input array is used as the root, we know that the numbers to the left will be the left subtree and the numbers to the right are the right subtree. So, we can solve problem using recursion with 2 pointers.

Complexity: time O(n), space O(logn)

* Initialize a left and right pointer and find midpoint (used to create node)
* Update pointers so right and midpoint move to left subtree
* Create node from new midpoint and continue to update pointers and create nodes from midpoint until there are no numbers left in left subtree portion of array
* Do the same thing with right subtree

[-10, -3, 0, 5 , 9]



[-10, -3, 0, 5 , 9]



[-10, -3, 0, 5 , 9]



A screenshot of a computer code

Description automatically generated