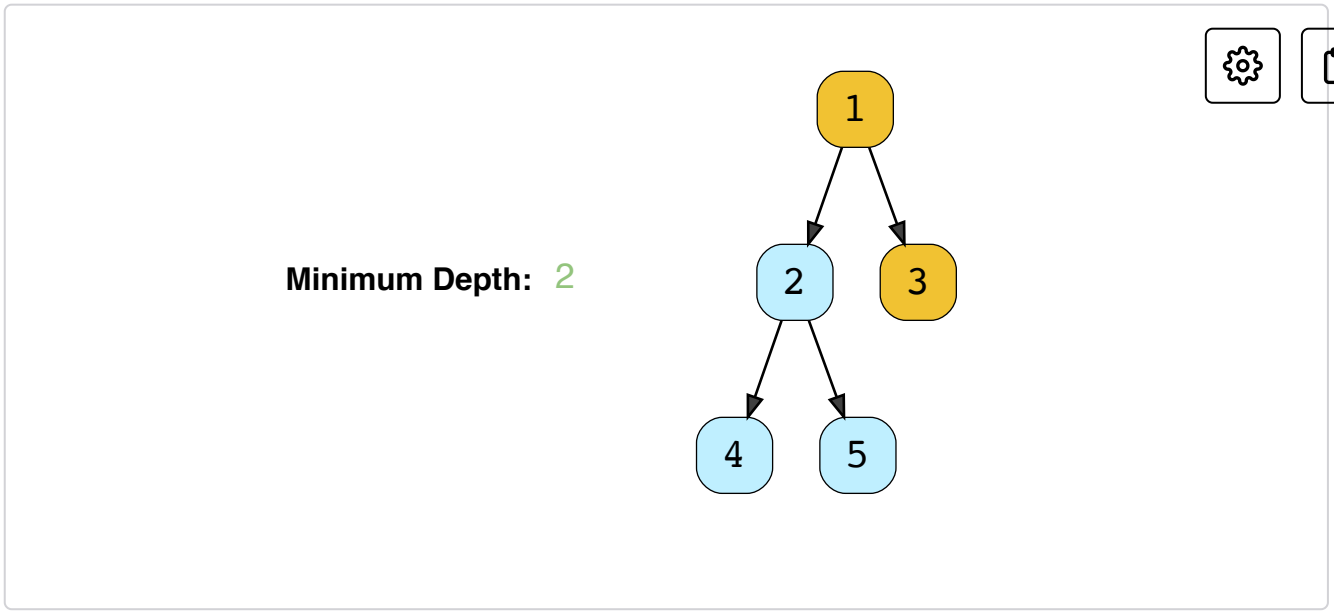
Minimum Depth of a Binary Tree (easy)

We'll cover the following

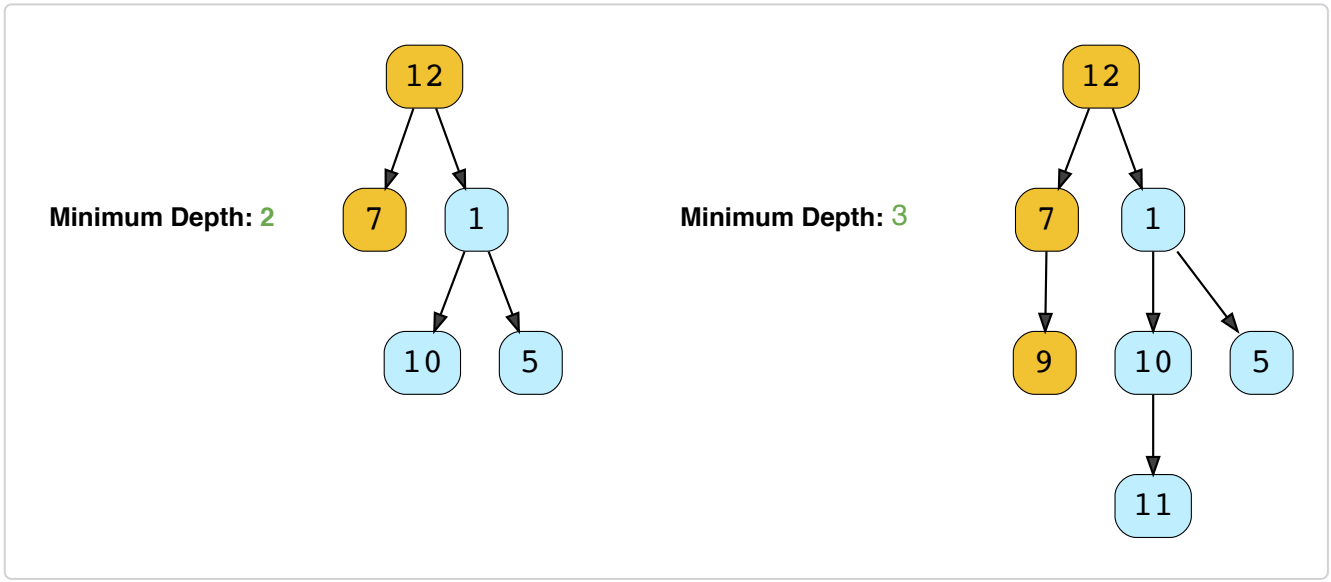
Problem Statement

Find the minimum depth of a binary tree. The minimum depth is the number of nodes along the **shortest path from the root node to the nearest leaf node**.

Example 1:



Example 2:



Try it yourself

Try solving this question here:

Java Python3 JS JS C++

```
10    queue = [root,]
11    while queue:
12      mini_depth+=1
13      size = len(queue)
```

```
    for _ in range(size):
      curr = queue.pop(0)
      if not curr.left and not curr.right:
        return mini_depth
      if curr.left:
        queue.append(curr.left)
      if curr.right:
        queue.append(curr.right)
  return mini_depth

def main():
  root = TreeNode(12)
  root.left = TreeNode(7)
  root.right = TreeNode(1)
  root.right.left = TreeNode(10)
  root.right.right = TreeNode(5)
  print("Tree Minimum Depth: " + str(find_minimu
  root.left.left = TreeNode(9)
  root.right.left.left = TreeNode(11)
  print("Tree Minimum Depth: " + str(find_minimu


main()

```

Output 0.16s

```
Tree Minimum Depth: 2
Tree Minimum Depth: 3
```

Solution

This problem follows the Binary Tree Level Order Traversal (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5726607939469312/) pattern. We can follow the same **BFS** approach. The

only difference will be, instead of keeping track of all the nodes in a level, we will only track the depth of the tree. As soon as we find our first leaf node, that level will represent the minimum depth of the tree.

Code

Here is what our algorithm will look like, only the highlighted lines have changed:

Java Python3 C++ JS

```
    return 0

  queue = deque()
  queue.append(root)
  minimumTreeDepth = 0
  while queue:
    minimumTreeDepth += 1
    levelSize = len(queue)
    for _ in range(levelSize):
      currentNode = queue.popleft()

      # check if this is a leaf node
      if not currentNode.left and not currentNoc
        return minimumTreeDepth

      # insert the children of current node in t
      if currentNode.left:
        queue.append(currentNode.left)
      if currentNode.right:
        queue.append(currentNode.right)



def main():
  root = TreeNode(12)
  root.left = TreeNode(7)
  root.right = TreeNode(1)
  root.right.left = TreeNode(10)
```

```
Output                                                    0.15s

 Tree Minimum Depth: 2
 Tree Minimum Depth: 3
```

Time complexity

The time complexity of the above algorithm is $O(N)$, where 'N' is the total number of nodes in the tree. This is due to the fact that we traverse each node once.

Space complexity

The space complexity of the above algorithm will be $O(N)$ which is required for the queue. Since we can have a maximum of $N/2$ nodes at any level (this could happen only at the lowest level), therefore we will need $O(N)$ space to store them in the queue.

Similar Problems

Problem 1: Given a binary tree, find its maximum depth (or height).

Solution: We will follow a similar approach. Instead of returning as soon as we find a leaf node, we will keep traversing for all the levels, incrementing `maximumDepth` each time we complete a level. Here is what the code will look like:

Java | Python3 | C++ | JS

```
18      maximumTreeDepth += 1
19      levelSize = len(queue)
20      for _ in range(levelSize):
21        currentNode = queue.popleft()
```

```
22
23        # insert the children of current node in t
24        if currentNode.left:
25          queue.append(currentNode.left)
26        if currentNode.right:
27          queue.append(currentNode.right)
28
29    return maximumTreeDepth
30
31
32  def main():
33    root = TreeNode(12)
34    root.left = TreeNode(7)
35    root.right = TreeNode(1)
36    root.right.left = TreeNode(10)
37    root.right.right = TreeNode(5)
38    print("Tree Maximum Depth: " + str(find_maximu
39    root.left.left = TreeNode(9)
40    root.right.left.left = TreeNode(11)
41    print("Tree Maximum Depth: " + str(find_maximu
42
43
44  main()
45
```


Back

Level Averages in a Binary Tree (easy)

Next

Level Order Successor (easy)

Mark as Completed

Ask a Question

Report

an Issue (https://discuss.educative.io/tag/minimum-depth-of-a-binary-tree-easy__pattern-tree-breadth-first-search__grokking-the-coding-interview-patterns-for-coding-questions)