



# Minimum Depth of Binary Tree

# Index	111
📅 CreatedAt	@September 28, 2022
👤 Person	 Ally Hyeseong Kim
🌟 Status	Done
🏷️ Tags	Python Tree
📅 UpdatedAt	@September 28, 2022

## References

LeetCode - The World's Leading Online Programming Learning Platform  
Level up your coding skills and quickly land a job. This is the best place to expand your knowledge and get prepared for your next interview.

 <https://leetcode.com/problems/minimum-depth-of-binary-tree>



### 파이썬 알고리즘 인터뷰

2021 세종도서 학술부문 선정작. 현업과 실무에 유용한 주요 알고리즘 이론을 깊숙이 이해하고, 파이썬의 핵심 기능과 문법까지 상세하게 이해할 수 있는 취업용 코딩 테스트를 위한 완벽 가이드다. 200여 개가 넘는...

 <https://www.aladin.co.kr/shop/wproduct.aspx?ItemId=245495826>



## References

1. DFS
2. BFS

## 1. DFS

```
def dfs(node):
    if not node:
        return 0
    if node.left and node.right:
        return 1 + min(dfs(node.left), dfs(node.right))
    elif node.left:
        return 1 + dfs(node.left)
    elif node.right:
        return 1 + dfs(node.right)
    else:
        return 1

class Solution:
    def minDepth(self, root: Optional[TreeNode]) -> int:
        return dfs(root)
```

- DFS 는 Stack 을 사용하여 구현한다.

## 2. BFS

```

class Solution:
    def minDepth(self, root: Optional[TreeNode]) -> int:
        if root is None:
            return 0
        queue = collections.deque([root])
        depth = 0

        while queue:
            depth += 1
            for _ in range(len(queue)):
                cur_root = queue.popleft()
                if cur_root.left:
                    queue.append(cur_root.left)
                if cur_root.right:
                    queue.append(cur_root.right)
                if not cur_root.left and not cur_root.right:
                    return depth
        return depth

```

- **BFS** 는 **Queue** 를 사용하여 구현한다. → 재귀가 아닌 반복 구조로 구현할 수 있다.