Prompt | Scratchpad | Solutions | Video Explanation

Difficulty: Easy **Category:** Graphs **Successful Submissions:** 57,586+

Depth-first Search

You're given a `Node` class that has a `name` and an array of optional `children` nodes. When put together, nodes form an acyclic tree-like structure.

Implement the `depthFirstSearch` method on the `Node` class, which takes in an empty array, traverses the tree using the Depth-first Search approach (specifically navigating the tree from left to right), stores all of the nodes' names in the input array, and returns it.

If you're unfamiliar with Depth-first Search, we recommend watching the Conceptual Overview section of this question's video explanation before starting to code.

Sample Input

```
graph = A
      /  |  \
     B   C   D
    / \     / \
   E   F   G   H
      / \   \
     I   J   K
```

Sample Output

```
["A", "B", "E", "F", "I", "J", "C", "D", "G", "K", "H"]
```

Hints

Hint 1

The Depth-first Search algorithm works by traversing a graph branch by branch. In other words, before traversing any Node's sibling Nodes, its children nodes must be traversed. How can you simply and effectively keep track of Nodes' sibling Nodes as you traverse them, all the while retaining the order in which you must traverse them?

Hint 2

Start at the root Node and try simply calling the depthFirstSearch method on all of its children Nodes. Then, call the depthFirstSearch method on all children Nodes of each child node. Keep applying this logic until the entire graph has been traversed. Don't forget to add the current Node's name to the input array at every call of depthFirstSearch.

Optimal Space & Time Complexity

O(v + e) time | O(v) space - where v is the number of vertices of the input graph and e is the number of edges of the input graph

Your Solutions | Run Code

Solution 1 | Solution 2 | Solution 3

```python
class Node:
    def __init__(self, name):
        self.children = []
        self.name = name

    def addChild(self, name):
        self.children.append(Node(name))
        return self

    def depthFirstSearch(self, array):
        array.append(self.name)
        for node in self.children:
            node.depthFirstSearch(array)

        return array

# Kunal Wadhwa

# https://github.com/kunal5042
# https://leetcode.com/kunal5042/
# https://www.hackerrank.com/kunalwadhwa_cs
# https://www.linkedin.com/in/kunal5042/

```

Tests | Quick Test | Sandbox

Test Case 1

```
{
  "graph": {
    "nodes": [
      {"children": ["B", "C", "D"], "id": "A", "value": "A"},
      {"children": ["E", "F"], "id": "B", "value": "B"},
      {"children": [], "id": "C", "value": "C"},
      {"children": ["G", "H"], "id": "D", "value": "D"},
      {"children": [], "id": "E", "value": "E"},
      {"children": ["I", "J"], "id": "F", "value": "F"},
      {"children": ["K"], "id": "G", "value": "G"},
      {"children": [], "id": "H", "value": "H"},
      {"children": [], "id": "I", "value": "I"},
      {"children": [], "id": "J", "value": "J"},
      {"children": [], "id": "K", "value": "K"}
    ],
    "startNode": "A"
  }
}
```

Test Case 2

```
{
  "graph": {
    "nodes": [
      {"children": ["B", "C"], "id": "A", "value": "A"},
      {"children": ["D"], "id": "B", "value": "B"},
      {"children": [], "id": "C", "value": "C"},
      {"children": [], "id": "D", "value": "D"}
    ],
    "startNode": "A"
  }
}
```

Custom Output | Raw Output | Submit Code

Yay, your code passed all the test cases!

5 / 5 test cases passed.

- ✓ Test Case 1 passed!
- ✓ Test Case 2 passed!
- ✓ Test Case 3 passed!
- ✓ Test Case 4 passed!
- ✓ Test Case 5 passed!