

Practice Interview

Objective

The partner assignment aims to provide participants with the opportunity to practice coding in an interview context. You will analyze your partner's Assignment 1. Moreover, code reviews are common practice in a software development team. This assignment should give you a taste of the code review process.

Group Size

Each group should have 2 people. You will be assigned a partner

Part 1:

You and your partner must share each other's Assignment 1 submission.

Part 2:

Create a Jupyter Notebook, create 6 of the following headings, and complete the following for your partner's assignment 1:

- Paraphrase the problem in your own words.

Answer:

Given a binary tree, start from the root node and output all paths available traversing from the root to the base leaf nodes of the tree. Each output should be a list starting from the root node value, and following each edge down and printing each of the leaf node values per path.

- Create 1 new example that demonstrates you understand the problem.
Trace/walkthrough 1 example that your partner made and explain it.

My Example:

Input: root = [1,2,3,10,9,8] Output: [[1,2,10],[1,2,9],[1,3,8]]

Partner Example Explained

input = [4, 2, 6, 1, 3, 5, 7] output = [[4, 2, 1], [4, 2, 3], [4, 6, 5], [4, 6, 7]]

This example has a balanced tree with 4 leaf nodes at the base, so there are 4 paths to traverse, starting from the root node = 4, then going through each left and right node down. The first layer root node = 4, the second layer leaf nodes = 2,6, the base layer leaf nodes = 1,3,5,7.

- Copy the solution your partner wrote.

```
In [ ]: # Copy of partner's solution

from typing import List

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def bt_path(root: TreeNode) -> List[List[int]]:
    if root is None:
        return []

    result = []

    def dfs(node: TreeNode, path: List[int]):
        if node is None:
            return

        # we add the current node to the path
        path.append(node.val)

        # if it's a leaf node, add the path to the result
        if node.left is None and node.right is None:
            result.append(path[:]) # Append a copy of the path

        # Recur for left and right children
        dfs(node.left, path)
        dfs(node.right, path)

        # Backtrack
        path.pop()

    dfs(root, [])
    return result
```

- Explain why their solution works in your own words.

Answer:

This solution creates an algorithm that intakes a root node of a tree, and uses depth-first traversal via recursion to visit all the unique root to leaf paths in the tree. This solution works because it begins by appending the root node value to a list to start the path traversal, then uses recursion down the left branches and appends their node values until the base leaf is reached to complete a path. It then pops back up a node to traverse down the right branches and appends their node values until the base is reached again. This process iterates recursively until all node values are visited, which then ensures all root to base leaf node paths are tracked.

- Explain the problem's time and space complexity in your own words.

Time Complexity:

This solution uses recursion to visit all nodes in the tree once, hence the time complexity is $O(n)$.

Space Complexity:

This solution has a space complexity of the maximum depth of the tree to store each path, hence the space complexity is the height of the tree $O(h)$.

- Critique your partner's solution, including explanation, and if there is anything that should be adjusted.

Answer:

The solution and code is very clear, and includes code comments for further explanations. Nothing to adjust from my side. Looks good to me.

Part 3:

Please write a 200 word reflection documenting your process from assignment 1, and your presentation and review experience with your partner at the bottom of the Jupyter Notebook under a new heading "Reflection." Again, export this Notebook as pdf.

Reflection

In this module, we were required to use the algorithms and data structure concepts we learned in class and apply it to solving actual problems with binary trees. We first were given a problem statement with example inputs and outputs, and had to apply the correct data structure to get the required solution. In my case, I had to traverse a binary tree to find the closest duplicated node, and since it was a distance-based question, I



decided to use breadth-first search to implement my solution. Next I reviewed the coding solution of my partner's assignment for a different question, which involved traversing all root to leaf node paths. That solution required a depth-first search approach and was implemented with recursion. I ran through the input examples on their coding solution and found that it worked correctly to produce the expected output. This process allowed me to understand how to solve different types of binary tree problems with different coding algorithms.

Evaluation Criteria

We are looking for the similar points as Assignment 1

- Problem is accurately stated
- New example is correct and easily understandable
- Correctness, time, and space complexity of the coding solution
- Clarity in explaining why the solution works, its time and space complexity
- Quality of critique of your partner's assignment, if necessary

Submission Information

 **Please review our [Assignment Submission Guide](#)**  for detailed instructions on how to format, branch, and submit your work. Following these guidelines is crucial for your submissions to be evaluated correctly.

Submission Parameters:

- Submission Due Date: `HH:MM AM/PM – DD/MM/YYYY`
- The branch name for your repo should be: `assignment-2`
- What to submit for this assignment:
 - This Jupyter Notebook (assignment_2.ipynb) should be populated and should be the only change in your pull request.
- What the pull request link should look like for this assignment:
`https://github.com/<your_github_username>/algorithms_and_data_structures`
 - Open a private window in your browser. Copy and paste the link to your pull request into the address bar. Make sure you can see your pull request properly. This helps the technical facilitator and learning support staff review your submission easily.

Checklist:

- ☐ Created a branch with the correct naming convention.
- ☐ Ensured that the repository is public.
- ☐ Reviewed the PR description guidelines and adhered to them.
- ☐ Verify that the link is accessible in a private browser window.

If you encounter any difficulties or have questions, please don't hesitate to reach out to our team via our Slack at [#cohort-3-help](#). Our Technical Facilitators and Learning Support staff are here to help you navigate any challenges.