

# 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.

```
In [ ]: # Given an list of integers, the solution must effectively traverse a binary tree
# structure and return an list with the list of nodes that is a path to each leaf node.
```

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

```
In [ ]: # Example 1:
# Input: [6, 8, 2, 3, 5, 7, 0, 1, 10]
# Expected Output: [[6,8,7], [6,8,10], [6,2,3,5], [6,2,0,1]]

# Example from assignment 1:
# Input: [1, 2, 2, 3, 5, 6, 7]
# Expected Output: [[1, 2, 3], [1, 2, 5], [1, 2, 6], [1, 2, 7]]

# In the above example is a sufficiently large test case to help better visualize
# the requirement. And the code output meets the need. The one missing part is considering
# the input is a list, we are processing the list items manually.
```

- Copy the solution your partner wrote.

```
In [ ]: from typing import List, Optional

# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val: int = 0, left: Optional['TreeNode']
                 = None, right: Optional['TreeNode'] = None):
        self.val = val
        self.left = left
        self.right = right

def bt_path(root: TreeNode) -> List[List[int]]:
    def dfs(node: TreeNode, path: list[int], paths: list[list[int]]):
        if not node:
            return
        path.append(node.val)
        if not node.left and not node.right:
            paths.append(path.copy())
        dfs(node.left, path, paths)
        dfs(node.right, path, paths)
        path.pop()

    paths = []
    dfs(root, [], paths)
    return paths

root = TreeNode(1)
root.left = TreeNode(2)
root.right = TreeNode(2)
root.left.left = TreeNode(3)
root.left.right = TreeNode(5)
root.right.left = TreeNode(6)
root.right.right = TreeNode(7)

# Call the function
paths = bt_path(root)

# Print the result
print(paths)
```

```
[[1, 2, 3], [1, 2, 5], [1, 2, 6], [1, 2, 7]]
```

- Explain why their solution works in your own words.

```
In [ ]: # Manually setting the nodes compeltely avoids the code to order and assign nodes.
# However, the code iterates through the nodes recursively, to find every path from
# root to every leaf node.
```

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

```
In [ ]: # The worst case:
# Time Complexity is  $O(n^2)$ 
# Space Complexity is  $O(n^2)$ 

# Node traversal is limited to DFS traversal by visiting each node only once,
# however the task to copy path and considering a skewed tree the worst case
```

```
# complexity is  $O(n^2)$ .
```

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

```
In [ ]: # If not for the missing input list to tree conversion the rest of the code covers  
# all intended results. Processing the input list to build the tree while also  
# maintaining covered path list, and updating the list off a metadata seems like  
# an effecient process.
```

## 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 [ ]: # Much of this would have been a simpler process if I was equipped well with basic  
# Python libraries. With my limited knowledge I now understand the rigour that is  
# necessary to build and complete a code solution. Also, the fact that I now know  
# Breadth-First Search (BFS) and Depth-First Search (DFS) while implementing a solution  
# and identified the trade-off makes this an effective learning. This assignment  
# reinforced my understanding of the time and space complexity of each solution.
```

## 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/pull/<pr\\_id>](https://github.com/<your_github_username>/algorithms_and_data_structures/pull/<pr_id>)
  - 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.