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 [ ]:
```
# Your answer here

"""
THE Question One "Check Duplicates in Tree" is to check a binary tree, whether it i
If a duplicate exists, return the duplicate value. If there are multiple duplicates
to the root. If no duplicate exists, return -1.

Trees can have duplicated values. The task is to traverse the tree looking for the
"""
```

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

In [ ]:
```
# Your answer here
"""
Let the new example be the tree below.

                  5
               /    \
```

```
            10      2
           /  \    / \
          10   4  9   9

With the value 5 as the root, the output should be 10, that is the duplicated value

"""
```

- Copy the solution your partner wrote.

In [ ]:

```python
# Your answer here

from collections import deque
class TreeNode(object):
     def __init__(self, val = 0, left = None, right = None):
         self.val = val
         self.left = left
         self.right = right

def is_duplicate(root: TreeNode) -> int:

    if not root:
        return -1
    queue = deque([(root, 0)])
    value_distance = {}
    min_distance = float('inf')
    result = -1

    while queue:
        node, distance = queue.popleft()

        if node.val in value_distance and distance < min_distance:
            min_distance = distance
            result = node.val
        else:
            value_distance[node.val] = distance

        if node.left:
            queue.append((node.left, distance + 1))
        if node.right:
            queue.append((node.right, distance + 1))

    return result


# Building the tree
root = TreeNode(5)
root.left = TreeNode(10)
root.right = TreeNode(2)
root.left.left = TreeNode(10)
root.right.right = TreeNode(4)
root.right.left = TreeNode(9)
root.right.right = TreeNode(9)
```

```
# Calling the function
print(is_duplicate(root))
```

```
10
```

- Explain why their solution works in your own words.

In [ ]:
```
# Your answer here

"""
The solution works by using "deque", that is a efficient appending and popping from

The code starts with the variables necessary:
- The "queue" (class deque) is initialized with the root node obejct and a distance
- The "value_distance" is a dictionary to store the distance of each value from the
- The "min_distance" is initialized to infinity (float('inf')) to keep track of the
- The "result" is initialized to -1 to store the value of the first duplicate found

This loop processes each node in the queue.

    If the node value is already in "value_distance" and the current "distance" is
    "min_distance" and "result" with the node value that will be returned. If not,
    "value_distance" using "node.val" as index and 'distance" and the value.

    Then the code checks if there are children nodes to traverse appending them to

Finally, the variable "result" will contain the duplicated value that is closest to
If there are no duplicated values, the variable will not be updated, therefore it w

"""
```

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

In [ ]:
```
# Your answer here

"""
Time Complexity: to identofy duplicated values, the code uses a  breadth-first sear
So, it will check each node of the tree looking for duplicates; therefore, the comp

Space Complexity: the dictionary will sotre all node values and distances to the ro
If all values are uique (worst case), the dictionary will store all "n" nodes. Ther

"""
```

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

In [ ]:
```
# Your answer here
"""
Any solution have positive points and space for improvement.

- The Breadth-First Search (BFS) is very efficient for this problem to ensure the p
```

```
- The dictionary was also a good chice. The operations are efficient with an averag

As an improvement sugestions:
- Add comments to the code will help readability.
- Optimize the "if" statement inside  the loop checking the "distance" only if the
in the dictionary "value_distance". This will improve the efficienty in extreme sce
values in the tree.

    if node.val in value_distance:
        if distance < min_distance:
            min_distance = distance
            result = node.val

"""
```

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 [ ]:
```
# Your answer here

"""

The Algorithms and Data Structures assignment was exciting.

It gave me a good opportunity to check how much I have learned.
The structure of Assignment 1 was very interesting. Instead of jumping into the cod
the assignment walked me through the mental steps necessary to find a solution that
paraphrasing the problem and thinking about other scenarios before jumping into cod
trial-and-error cycle when coding.

Reviewing my partner’s code was also very interesting. It helped me understand area
For example, it was challenging to look at the code for the first time without comm
algorithm step-by-step using a sheet of paper to follow the logic and look for poin
change to the “if” in the loop. This will not change the algorithm's complexity but
From now on, I will be more careful with my comments to improve reliability for mys
double-checking if my “if” clauses could be improved by eliminating redundancies.

"""
```

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: `23:59 PM - 24/07/2024`
- 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.