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 [ ]:
```
# Determine if there are any duplicate values in the given Binary Tree.
# If duplicates exist, return the closest one to the root; if multiple duplicates are equidistant, return any of them.
If no duplicates exist, return -1.
```

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

In [ ]:
```
# New example:
# # Example binary tree:
#       1
#       / \
#      2   3
#     / \   \
#    4   2   5
#       / \
#      4   6

# Constructing the binary tree
# root = TreeNode(1)
# root.left = TreeNode(2, TreeNode(4), TreeNode(2, TreeNode(4), TreeNode(6)))
# root.right = TreeNode(3, right=TreeNode(5))
# result = is_duplicate(root)
# print(result)  # Output: 2

# Partner example: a binary tree is constructed with 1 as the root, having 2 as both left and right children. Each 2 n
ode further branches out with different values (3, 5 on the left, and 6, 7 on the right).
# The function is_duplicate is called with root1 as input, aiming to find and return the closest duplicate value to th
e root, which in this case is 2.
```

- Copy the solution your partner wrote.

In [ ]:
```
# class TreeNode(object):
def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def enqueue(l, v):
    l.append(v)

def dequeue(l):
    return l.pop(0)

def is_duplicate(root):

    if not root:
        return -1

    queue = [root]
    seen = set()

    while queue:
        node = dequeue(queue)

        if node.val in seen:
            return node.val
        seen.add(node.val)

        # Traverse left and right child nodes
        if node.left:
            enqueue(queue, node.left)
        if node.right:
            enqueue(queue, node.right)

    return -1
```

- Explain why their solution works in your own words.

In [ ]:
```
# My partner's solution utilizes depth-first search (DFS) to traverse the binary tree while maintaining a dictionary t
o track the first occurrence of each node's value and its depth.
# This allows the function to efficiently identify and return the closest duplicate value to the root, meeting the pro
blem's requirements.
```

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

In [ ]:
```
# The time complexity of the solution is O(n), where n is the number of nodes in the binary tree, because each node is
visited once during the depth-first search.
# The space complexity is O(n) in the worst case due to the stack used for DFS and the dictionary to store node values
and depths.
```

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

In [ ]:
```
# Some fo the edge cases are not considered in my partner's solution. For example:
# If the tree is empty (root is None), in which case the function should return -1 indicating no duplicates.
# If all nodes in the tree have unique values, where the function should also return -1 since there are no duplicates
present.
```

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 [ ]:
```
# Firstly, understanding the problem the code aims to solve is essential. This involves carefully reading the problem
statement or requirements documentation to grasp the intended functionality and expected behavior.

# Assessing the structure and organization of the code is important. This includes evaluating whether the code is well
-structured, follows naming conventions, and uses appropriate design patterns or principles. Clear and concise code or
ganization enhances readability and maintainability.

# Analyzing the algorithm and logic implemented is crucial for evaluating efficiency and correctness. This involves ch
ecking the approach taken to solve the problem, considering time and space complexity, and identifying any potential o
ptimizations or inefficiencies.

# Testing and edge case handling should also be reviewed. Ensuring the code handles various input scenarios, including
edge cases and boundary conditions, helps verify its robustness and reliability in different situations.

# Providing constructive feedback is an essential part of code review. This includes pointing out areas for improvemen
t, suggesting alternative approaches, and acknowledging strengths in the code's design or implementation.

# Lastly, clear presentation is key to a successful code review. Providing clear explanation, explaining the rationale
behind suggestions or critiques, and fostering a collaborative environment for discussing improvements contribute to t
he overall quality of the codebase and the development process.
```

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>`
  - 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.