

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.

Given a tree, we want to find the duplicate value closest to the root of the tree if such a value exists. Otherwise we return -1.

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

My example:

Input: [1, 2, 3, 4, 5, 6, 1, 2]

Output: 1

Explanation: There are two duplicate values in the tree (1, 2). We return 1 because it is a level higher than 2 and hence closer to the root.

Partner Example:

Input: [1, 2, 2, 3, 5, 6, 7]

Output: 2

Walkthrough: The root node has children with duplicate values 2 so we return 2 as the answer.

- Copy the solution your partner wrote.

```
In [4]: 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

    seen = set()
    queue = deque([root])

    while queue:
        node = queue.popleft()

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

        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)

    return -1
```

- Explain why their solution works in your own words.

In this solution, we perform a Level Order traversal of the Tree using a Queue while tracking the unique values seen so far in the tree. The first node to have a value that has already been seen is guaranteed to have the duplicate value that is closest to the root and it is returned. If we complete the traversal without finding a duplicate value then no duplicate value exists in the tree and we return -1 as per the requirements of the problem.

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

Time complexity: $O(n)$ where n is the number of nodes in the tree

Explanation: We visit each node in the tree once to check its value.

Space complexity: $O(n)$ where n is the number of nodes in the tree

Explanation: We keep track of unique values in the tree in a set and list of nodes to be visited next in a queue which use $O(n)$ and $O(n/2)$ space respectively in the worst case in which we must traverse the whole tree which does not have any duplicates.

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

The solution is clear, concise and correct. It handles edge cases such as an empty/None tree in the input.

The explanation is clear as well.

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

The problem was quite simple but required an understanding of binary trees and their traversal. For an efficient solution, it was important to come up with an approach that could find all root-to-leaf paths in a single traversal of the tree to avoid duplicate work.

Analyzing the time and space complexity for the solution is non trivial. At first glance one may assume linear time complexity and constant space complexity but it would be incorrect. For space complexity, it is important to keep in mind that recursive function calls implicitly take up space in the function call stack. For time complexity, although returning a slice of a list in Python looks like a single operation, under the hood it requires traversing the list to create a copy to be returned. It is important to also realize that there are two parts to determining the time complexity: the traversal of the tree [$O(n)$] and the creation of the path which happens only in the leaf nodes [best case: $O(1n)$ to worst case: $O(n/2 \lg n)$].

Reviewing/reading someone else's solution/code is always a learning experience which offers the opportunity to learning new things or new ways of doing things. My partner's code was quite simple and hence elegant. However, they may have misunderstood the instructions of the assignment in part and solved all three



problems, which made finding the solution to be reviewed a little difficult.

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.