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

'''Given a list of integers that may contain duplicates. The task is to identify al
from the range starting at 0 up to the maximum number present in the list. If no nu
```

- 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
'''
Example:

Suppose you have the list of integers: [0, 1, 2, 4, 5, 7].

The maximum value in the list is 7, so the range of numbers to consider is from 0 t
The full range of numbers from 0 to 7 is: [0, 1, 2, 3, 4, 5, 6, 7].
The numbers missing from the input list are 3 and 6.
Therefore, the function should return the list of missing numbers: [3, 6].
```

```
If the list were [0, 1, 2, 3, 4, 5, 6, 7], there would be no missing numbers, and t
'''

'''
Let's go through the partner's example step-by-step for the input list lst = [7, 1]

Determine the Maximum Value:
The maximum value in the list lst is 7.

Define the Full Range:
The full range of numbers from 0 to 7 is: [0, 1, 2, 3, 4, 5, 6, 7].

Create a Set from the Input List:
Convert the input list to a set to remove duplicates and allow for efficient lookup

Identify Missing Numbers:
Subtract the set of numbers from the input list from the full range set: {0, 1, 2,
The result is a set of missing numbers: {0, 2, 3, 4, 5, 6}.

Convert to a Sorted List:
Convert the set of missing numbers to a sorted list: [0, 2, 3, 4, 5, 6].

Return the Result:
Since there are missing numbers, the function will return [0, 2, 3, 4, 5, 6].
Therefore, for the input list lst = [7, 1], the output of the function would be [0,
'''
```

- Copy the solution your partner wrote.

In [ ]:
```python
# Your answer here
def missing_num(nums: list) -> int:
    # Determine the maximum value in the list to define the upper limit of the rang
    max_val = max(lst)

    # Create a set from the list to remove duplicates and for O(1) average-time com
    set_lst = set(lst)

    # Create a set of all numbers in the range 0 to max_val
    full_range_set = set(range(max_val + 1))

    # Find the difference between the full range set and the set created from the l
    missing_numbers = full_range_set - set_lst

    # Convert the set of missing numbers to a sorted list
    missing_numbers_list = sorted(list(missing_numbers))

    # If there are no missing numbers, return -1
    if not missing_numbers_list:
        return -1
    else:
        return missing_numbers_list
```

- Explain why their solution works in your own words.

In [ ]:
```
# Your answer here

'''
Efficient Range Comparison and Set Difference:

By comparing the set of numbers in the list to the full range of numbers from 0 to
which numbers are missing. Subtracting the set of numbers in the list from the set
making this approach fast and efficient.

Handling Duplicates:

Using a set automatically removes duplicates, ensuring that each number is only con

'''
```

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

In [ ]:
```
# Your answer here

'''
Time Complexity: O(n log n)
---------------------------

Finding the Maximum Value:
This operation iterates through the list once to find the maximum value. This takes

Creating the Set from the List:
Converting a list to a set involves iterating through the list and adding each elem

Creating the Full Range Set:
This operation involves creating a range from 0 to max_val (inclusive) and then con
Since max_val is the maximum element in the list, m can be at most n (in the worst

Finding the Set Difference:
The set difference operation involves iterating through the elements of one set and
This typically takes O(n) time since each lookup in a set is O(1) on average.

Sorting the Missing Numbers:
sorted(list(missing_numbers)): The missing numbers are converted to a list and sort
where k is the number of missing elements. Hence, this step takes O(n log n) time i

The most time-consuming step is sorting the missing numbers, which takes O(n log n)

Space Complexity: O(n)
----------------------

Space for the List:
The input list itself takes O(n) space.

Space for the Set Created from the List:
set(lst): This set also takes O(n) space, as it stores each unique element from the

Space for the Full Range Set:
set(range(max_val + 1)): This set takes O(m) space, where m is the maximum value in
```

```
Space for the Set Difference:
The set of missing numbers, missing_numbers, in the worst case, can store up to O(n

Space for the Sorted List:
The sorted list of missing numbers, missing_numbers_list, takes O(n) space.

Overall, the space complexity is dominated by the sets and the list used, which giv
'''
```

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

In [ ]:

```
# Your answer here
'''
There are a couple of minor issues to address in the code:

The function accepts a list named nums, but the code uses lst within the function.
The function is designed to return a list of missing numbers, but the docstring imp

Also, any approach to eliminate the need of sorting the set can help to improve the
to use boolean array to find the missing values. The time complexity is O(n), since
'''

def missing_num(nums: list) -> list:
    if not nums:
        return []

    # Determine the maximum value in the list
    max_val = max(nums)

    # Create a boolean array to mark the presence of numbers
    present = [False] * (max_val + 1)

    # Mark the presence of each number in the list
    for num in nums:
        present[num] = True

    # Collect the missing numbers
    missing_numbers = [i for i in range(max_val + 1) if not present[i]]

    return missing_numbers if missing_numbers else -1

# Example usage
print(missing_num([7, 1]))  # Output: [0, 2, 3, 4, 5, 6]
```

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
'''
For Assignment1, I worked on the problem to find the first duplicate in a list usin
the different search techniques used for traversing through tree. I used the breadt
helped in identifying the duplicate or return -1 if none is duplicated. The second
depth first search. It helped in utilising the reusability concepts and understandi

For Assignment 2,  I evaluated my partner's solution for identifying missing number
examples, and breaking down the solution's time and space complexities, I gained a
set operations. Moreover, offering constructive feedback on the strengths and possi
explore algorithmic efficiency and optimization strategies more thoroughly.
'''
```

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.