assignment_2

July 22, 2024

1 Practice Interview

1.1 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.

1.2 Group Size

Each group should have 2 people. You will be assigned a partner

1.3 Part 1:

You and your partner must share each other's Assignment 1 submission.

1.4 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.

[1]:
```
# Your answer here

# My partner's assignment-1 github commit link 'https://github.com/GianneW/
 ↪algorithms_and_data_structures/pull/1/commits' where \
# the her commit hash is '562a16af7ee5f50ac74c7858a6fc5202e8688605', here␣
 ↪henceforth denominated as Her / her / Hers / hers.

# From the commit review perspective we can perceive the two following issues:
# 1 - Different python version (possibly from a different environment) was used␣
 ↪on her code when running it for the last \
# time prior to commit it.
# 2 - She forgot to put her firstname (Wei) in the first step which would␣
 ↪choose give the result '2' instead of '1' to \
# follow up with in the subsequent tasks.
# Following up with was made there so far, her Python function in the 'Starter␣
↪Code for Question 3' is designed to find \
# the missing numbers in a given list of integers numbers.
```

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

[2]:
```
# Your answer here

# My NEW example:
list_of_numbers = [5,0,1]

def missing_numbers(list_of_numbers) -> list:
    n = max(list_of_numbers)
    n_list = list(range(n + 1))
    output_list = [item for item in n_list if item not in list_of_numbers] #␣
↪Here I use list comprehention as a one-liner.

    if not output_list:
        return -1

    return output_list

missing_numbers(list_of_numbers) # Executing the funtion the the same imput of␣
↪hers.

# Here through the list of numbers, this function builds a new complete list of␣
↪integers from zero to maximum number \
# inclusively (n + 1) and then through list comprehention it picks the missing␣
↪values with the `not in` test condition.
```

[2]:
```
[2, 3, 4]
```

- Copy the solution your partner wrote.

[9]:
```
# Your answer here

# Her soluton:
def missing_num(nums) -> int: #nums is a list
  # TODO
  n_list = []
  n = max(nums)
  for i in range(n):
    n_list.append(i)

  output_list = []
  for item in n_list:
    if item not in nums:
      output_list.append(item)

  if output_list == []:
    return -1

  return output_list

missing_num(list_of_numbers)
```

[9]:
```
[2, 3, 4]
```

- Explain why their solution works in your own words.

[4]:
```
# Your answer here

# Her solution works because her function appends a new list of integers from␣
 ↪the range of zero to the mmaximum \
# number of the input list for the function and then her output list will be␣
 ↪appended with a negative logic of found \
# numbers, i.e., the ones that are not in the list of the input list.
```

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

[5]:
```
# Your answer here

# Time Complexity: (O(n.m)), where (n) is the maximum value in the input list␣
 ↪nums and (m) is the number of elements \
# in nums (for the iteration).
# Space Complexity: (O(n)), where n is the maximum value in the input list nums.
# This means the function's runtime scales linearly with the size of nums and␣
 ↪the maximum value in nums, and it \
# uses space proportional to the maximum value in nums.
```

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

[6]:
```
# Your answer here

# The code works as expected but it could elegantly be condensed with␣
 ↪one-liner list comprehention for redability.
# List comprehension provides a more concise way to create lists. It combines␣
 ↪the loop and the conditional logic into \
# a single line of code.
# Also, generally, List comprehension is more efficient and it can be faster as␣
 ↪it is optimized internally by Python as \
# appending is slightly slower due to the repeated calls to the .append()␣
 ↪method, especially in large loops.
```

1.5 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.

1.5.1 Reflection

[7]:
```
# Your answer here

# My reflection
# The process was challenging, as I had to begin by interpreting the questions␣
 ↪based on the hash selection. \
# This required me to grasp the foundations of the classes taught in order to␣
 ↪start coding the correct algorithm \
# for the given logic and input examples. The most challenging part involved␣
 ↪some trial and error, which helped me \
# identify necessary function adjustments, such as using `set()` to keep values␣
 ↪unique. This led to checking seen the \
# values to handle potential duplicates. For each node, the function checks if␣
 ↪its value (`node.val`) is already in the \
# seen set. If it is, a duplicate has been found, and the function immediately␣
 ↪returns this value. The path to leaves \
# problem was indeed more challenging once it requested an extra grasp of how␣
 ↪to make to return all root to leaves in any \
# order. Regarding partner reflection and code readability, she missed using␣
 ↪her first name to reach the correct assignment, \
# but I reviewed what she had done so far, including her code logic structure.␣
 ↪The code is functioning with my own logic \
# replication, which helped restart the main code's idea in terms of algorithm␣
 ↪and results for a linear time and space \
# complexity function.
```

1.6 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

1.7 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.

1.7.1 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: - [X] Created a branch with the correct naming convention. - [X] Ensured that the repository is public. - [X] Reviewed the PR description guidelines and adhered to them. - [X] 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.