Algorithms - Assignment 2 - Leila

Algorithms - Assignment 2 - Part 1

You and your partner should send to each other your Assignment 1 submission.

My partner's Assignment 1 is stored in https://github.com/suzannemichie/UofTDSI_Algorithms_SMC2024

Algorithms - Assignment 2 - Part 2

Create a Jupyter Notebook, create 6 of the following headings, and complete the following about your partner's assignment 1

Algorithms - Assignment 2 - Part 2-1

Paraphrase the problem in your own words.

In an array of integer values, return the misssing integers within the range. The Range has no particular order and can have repeated integers.

Algorithms - Assignment 2 - Part 2-2

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

Leila's example:

Input: lst = [5, 3, 0, 1)

Output: [2, 4]

Suzanne's (my partner) examples:

Suzzane made two examples, while her Example 4 is correct, her example 5 is not correct. Output for her Example 4 should be [0, 3, 6, 8, 9]

Suzanne's Example 4

Input: lst = [1, 2, 4, 5, 7, 10]

Output: [0, 3, 6]

Suzanne's Example 5

Input: lst = [0, 2, 3, 4, 6, 8, 9]

Output: [1, 5, 7]

Algorithms - Assignment 2 - Part 2-3

Copy the solution your partner wrote.

```
### Solution Suzanne (my partner) wrote:

from typing import List

def missing_num(nums: List[int]) -> List[int]:
    n = len(nums)

    # Create a set to store the unique numbers in the given list
    num_set = set(nums)

    # Initialize a list to store the missing numbers
    missing_numbers = []

    # Iterate through the range [0, n] and check for missing numbers
    for i in range(n + 1):
        if i not in num_set:
            missing_numbers.append(i)

    # If there are missing numbers, return the list. Otherwise, return [-1]
    return missing_numbers if missing_numbers else [-1]

# Example usage:
lst1 = [0, 2]
print(missing_num(lst1))  # Output: [1]

lst2 = [5, 0, 1]
print(missing_num(lst2))  # Output: [2, 3, 4]

lst3 = [6, 8, 2, 3, 5, 7, 0, 1, 10]
print(missing_num(lst3))  # Output: [4, 9]

     [1]
     [2, 3]
     [4, 9]


# Example 4
lst4 = [1, 2, 4, 5, 7, 10]
print(missing_num(lst4))  # Output: [0, 3, 6]

# Example 5
lst5 = [0, 2, 3, 4, 6, 8, 9]
print(missing_num(lst5))  # Output: [1, 5, 7]

     [0, 3, 6]
     [1, 5, 7]
```

Algorithms - Assignment 2 - Part 2-4

Explain why their solution works in your own words.

Suzanne's solution is:

- creating a set num_set that stores the unique integers from the input list nums
- then it creates an empty list missing_numbers to store the missing numbers
- then the algorithm iterates from 0 to n (where n is the length of the input list + 1) and checks if each number in this range is present in the num_set
- if a number not found in num_set, is considered a missing number and added to missing_numbers list
- if there are missing numbers in the list, it returns that list; otherwise, returning [-1]

Algorithms - Assignment 2 - Part 2-5

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

The time complexity of Suzanne's code is O(n) and n is the length of the input list nums. This is because the algorithm iterates through a range from 0 to n + 1 once to check for missing numbers, and each check on the set num_set takes O(1) time.

The space complexity of Suzanne's code is O(n), and n is again the length of nums. This is because the code creates a set num_set to store the unique numbers in the input list, and also initializes a list missing_numbers to store the missing numbers. The size of these data structures depends on the number of elements in the input list.

Algorithms - Assignment 2 - Part 2-6

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

Suzanne's code was correct and delivering correct outputs. However, if working with significantly large inputs and range of missing integers are extensive, an imprvement in the code would optimize the storage of missing ingeres, means reducing the space complexity. Instead of storing the missing numbers in a list and returning the list, we could modify the function to return the missing numbers as it finds them. If so, we can avoid storing all missing numbers in memory, but still returning the outputs.

```
## Suzanne's code modified:

from typing import List

def missing_num(nums: List[int]):
    num_set = set(nums)

    # Iterate through the range [0, n] and yield missing numbers
```

Algorithms - Assignment 2 - Part 3

Please write a 200 word reflection documenting your studying process from assignment 1, and your presentation and reviewing experience with your partner at the bottom of the Juypter Notebook under a new heading "Reflection." Again, export this Notebook as pdf.

```
lst1 = [0, 2]
```

Algorithms - Assignment 2 - Part 3 - Reflection

(1) My assigned problem was Problem 1 which was "Finding duplicate values in a Binary Tree". It involved understanding how the algorithm uses a recursive approach to check for duplicates while traversing the tree. By testing the code with a sample tree, I learned about the importance of recursion in solving tree-related problems and the necessity of thorough testing to ensure the correctness and effectiveness of the solution. This exercise helped me to understand algorithms in the context of binary trees and proved the value of systematic analysis and testing in writing reliable codes.

(2) The reviewing process of my partner's Assignment 1 "Finding missing Integers in a List" made me study the problem in full as if it was my own. I briefly thought of the potential solution(s) and with that in mind I tested my partner's inputs by executing her code. The outputs were correct. I tried a few new inputs with known outputs and used her code, all outputs were correct. I concluded her code is right even though such conclusions cannot be generalized for inputs of any length and order. I then thought if there are any other ways of achieving the same by taking less time and/or space.