UofT DSI Algorithms

Assignment 1

Submitted by Suzanne M Chalambalacis 22-Feb-2024

Part 1:

You will be assigned one of three problems. You can find the problem description in q[assigned number].md. They are based-off problems from Leetcode.

```
(hash("suzanne michie")%3)+1
```

```
3
```

Question Three: Missing Number in Range

You are given a list containing `n` integers in the range `[0, n]`. Return a list of numbes that are missing from the range `[0, n]` of the array. If there is no missing number, return -1. Note, all the integers in the list may not be unique.

Examples

Example 1

Input: `lst = [0, 2]`

Output: [1]

Example 2

Input: `lst = [5, 0, 1]`

Output: [2, 3, 4]

Example 3

Input: `lst = [6, 8, 2, 3, 5, 7, 0, 1, 10]`

Output: [4, 9]

Starter Code

```
def missing_num(nums: List) -> int:
    # TODO
```

Part 2:

In a Jupyter Notebook (.ipynb) file, create 6 headings are write down the following:

Paraphrase the problem in your own words

Answer: Given a list of n integers within the range [0, n], the task is to provide a list of numbers that are absent from the array's range [0, n]. If there are no missing numbers, the function should return -1. It is important to note that the integers in the list may not be unique.

In the .md file containing your problem, there are examples that illustrate how the code should work. Create 2 new examples that demonstrate you understand the problem.

Answer:

Example 4

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

Output: [0, 3, 6]

Example 5

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

Output: [1, 5, 7]

Code the solution to your assigned problem in Python (code chunk). Try to find the best time and space complexity solution!

Answer:

```python
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]
```

```python
# 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]
```

Explain why your solution works

Answer:

The solution works by using a set to efficiently check for the presence of numbers in the given list.

The code creates a set (`num_set`) to store the unique numbers in the given list (`nums`). It then iterates through the range `[0, n]` using a loop, where `n` is the length of the input list `nums` . Inside the loop, it checks whether each number `i` is present in the set `num_set` . If `i` is not in the set, it means `i` is a missing number, and it is appended to the `missing_numbers` list. After the loop completes, the function checks if there are any missing numbers. If there are, it returns the list of missing numbers. Otherwise, it returns `[-1]` .

Explain the problem's time and space complexity

Answer:

Time Complexity: The time complexity of the provided solution is O(n), where `n` is the length of the input list `nums` . The main factor contributing to the time complexity is the loop that iterates through the range `[0, n]` . The code also performs a constant-time operation by checking whether the current number is in the set `num_set` .

Space Complexity: The space complexity of the solution is also O(n). The primary space usage comes from the set `num_set` created to store the unique numbers in the input list.

Explain the thinking to an alternative solution (no coding required, but a classmate reading this should be able to code it up based off your text)

Answer:

An alternative solution could involve using the mathematical property of the sum of the first n numbers. The sum of the first n numbers can be expressed as n * (n + 1) / 2.

1. Calculate Expected Sum: Determine the expected sum of the first n numbers using the formula: expected_sum = n * (n + 1) / 2.
2. Calculate Actual Sum: Compute the sum of the elements in the given list (nums). This can be done with a loop or a built-in sum function.
3. Identify Missing Number(s): Subtract the actual sum from the expected sum. The result will be the sum of the missing number(s).
4. Return Result: If there are missing numbers, return the list of missing numbers. Otherwise, return -1.

This approach can be more memory-efficient than the previous one, as it doesn't require the creation of a set.