Homework: Time Complexity and Function Testing

In this assignment, you'll write three Python functions to solve the same problem but with different constraints. You'll also analyze the time complexity of your solutions. You will create and submit two files: `hw3.py` and `test_hw3.py`. The `hw3.py` file should contain all the functions described in the assignment, while the `test_hw3.py` file should include unit tests to verify the correctness of these functions.

Problem Statement

You are given two lists, `list1` and `list2`, that have the following properties:

- They contain unique integers.
- Both lists are unordered.
- Both lists have the same length, (n).

Your task is to:

1. First, generate two random lists.
2. Write two functions to find the number of common items between these two lists using different approaches.
3. Mesure the execution time of the two functions for different list sizes.

Part1: Generating Random Lists

Write a Python function `generate_lists(size)` that:

- Takes an integer `size` as input.
- Creates two random lists `list1` and `list2` of unique integers with length equal to `size`. Use the `random.sample()` function to create these lists. `random.sample(population, k)` return a `k` length list of unique elements chosen from the population sequence.
- Returns `list1` and `list2`.

Example of `random.sample()` usage:

```python
import random

size = 10
print(random.sample(range(size*2), size)) #Prints a list of 10 unique random elements, each between 0 and 19
```

Part 2: find the number of common items (No collections) and analyze the running time

Once you have generated the two lists, your task is to write a Python function `find_common(list1, list2)` that:

- Takes list1 and list2 as input.
- Returns the number of common items between these two lists.
- **Constraint:** You are not allowed to use any Python collections (such as lists, sets, dictionaries, strings, tuples, etc.).

Additional instructions: Analysis

- For each line of your function, determine the number of operations it performs.
- Time Complexity: Derive the overall time complexity of your function using asymptotic notation (Big-O).

Part 3: find the number of common items and analyze the running time

Once you have generated the two lists, your task is to write a Python function `find_common_efficient(list1, list2)` that:

- Takes list1 and list2 as input.
- Returns the number of common items between these two lists.
- **You are allowed to use any Python collections (like sets, dictionaries, etc.) to make your code more efficient.**

Additional instructions: Analysis

- For each line of your function, determine the number of operations it performs, e.g.:
- Time Complexity: Derive the overall time complexity of your function using asymptotic notation (Big-O).

```
def sum_n(n):
  """Returns the sum of the first n integers"""
                           # Costs:
  total = 0                # 1
                           #
  for num in range(n):     # n*
    total = total + num    #     2
                           #
  return total             #+1
                           #-------
                           # 1 + 2n + 1 = 2n+2 = O(n)
```

Part 4: Measuring Execution Time

Write a function `measure_time()` to measure the execution time of `find_common` and `find_common_efficient` for different input sizes: `10, 100, 1000, 10000, 20000`. Then create and display a table showing the execution time for `find_common` and `find_common_efficient` with the different list sizes. When you run the measure_time function, you should get an output similar to this:

```
  List Size    find_common Time (s)    find_common_efficient Time (s)
-----------  ----------------------  ------------------------------
         10                  [Time]                          [Time]
        100                  [Time]                          [Time]
```

```
         1000              [Time]              [Time]
        10000              [Time]              [Time]
        20000              [Time]              [Time]
```

Part 5: Testing Functions Using Unit Tests

In this part, you will create file `test_hw3.py` that contains unit tests to verify the correctness of the following functions `generate_lists`, `find_common`, and `find_common_efficient`. Write your own unittests to further test expected behaviors. For example, you need to test whether `generate_lists` produces lists of the correct size and with unique elements.

Submitting

Submit the following files:

- hw3.py
- test_hw3.py

Students must submit to Gradescope individually within 24 hours of the due date (homework due dates are typically Tuesday at 11:59 pm EST) to receive credit.