HOMEWORK 4
RECURSION AND DYNAMIC PROGRAMMING *

10-607 COMPUTATIONAL FOUNDATIONS FOR MACHINE LEARNING

START HERE: Instructions

- **Collaboration Policy**: Please read the collaboration policy in the syllabus.
- **Late Submission Policy:** See the late submission policy in the syllabus.
- **Submitting your work:** You will use Gradescope to submit answers to all questions and code.
  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, please use the provided template. Submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Each derivation/proof should be completed in the boxes provided. To receive full credit, you are responsible for ensuring that your submission contains exactly the same number of pages and the same alignment as our PDF template.
  - **Latex Template:** `https://www.overleaf.com/read/dtjscgjtszwt`
  - **Programming:** You will submit your code for programming questions on the homework to Gradescope. After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). Unless otherwise specified, you are only permitted to use the Python Standard Library modules and `numpy`.
- **Materials:** The data and reference output that you will need in order to complete this assignment is posted along with the writeup and template.

Question	Points
Recursive Sequences	6
Recursion	2
Code to Equation	6
Total:	14

*Compiled on Monday 11th November, 2024 at 01:59

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

- ● Matt Gormley
- ○ Marie Curie
- ○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

- ● Henry Chai
- ○ Marie Curie
- ⊗ Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- □ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- □ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

10-606	10-6067

1 Recursive Sequences (6 points)

Take the recursive sequence defined here:

$$f(n) = n, \forall 0 \leq n \leq 3$$
$$f(n) = f(n-1) + f(n-2) - \frac{f(n-4)}{f(n-3)}, \forall n \geq 4$$

Download `10607_f23_hw4.zip` and implement `sequence1`, `sequence2`, and `sequence3` in the file `hw4.py`.

- Implement this sequence recursively in `sequence1()`.
- Implement the sequence iteratively in `sequence2()`.
- Implement the sequence recursively with memoization in `sequence3()` .

For each of the questions, you need to fill in the function bodies (in place of `# YOUR CODE HERE`). To test your code locally on a subset of Gradescope tests, run the script `hw4_tests.py`. After you are done, you will submit your code to Gradescope, where we wil run your code on a full suite of tests and the autograder will assign your code points based on whether it passes the tests.

We also provided a file `sequence_time.py` that contains code for you to observe the speed of the techniques you have implemented for the sequence. Run the script and take a minute to observe and reason about the relative amount of time each technique uses.

1. (2 points) **Short answer:** Find the time complexity of the naive recursive implementation of the sequence in `sequence1()`. Observe that it is very bad, and explain why that is the case based on your code.

Solution

Time complexity for naive recursive sequence(1) = 4.909 ms = $O(2^n)$
Recursive calls itself multiple times, and growth of the tree exponentially.
It is very bad because of its inefficiency. Extremely slow.

2. (2 points) **Short answer:** Find the time complexity of your iterative implementation of the sequence (`sequence2()`) based on n.

Solution

$O(n)$

3. (2 points) **Short answer:** Find the time complexity of your recursive implementation with memoization (`sequence3()`) based on n.

Solution

$O(n)$

2 Recursion (2 points)

Download `10607_f23_hw4.zip` and implement the following questions in the file `hw4.py`.

- Implement the functions `recursive_solution` (left hand side) and `static_solution` (right hand side) of the following equation:

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$$

- In Python, we commonly use the `len()` function to determine the length of a string. Turns out, recursion is a viable way to find the length of a string. Implement the function `string_length` without using Python's built in `len()`. You will not earn any points if you use Python's built in function.
- The sum of a finite geometric series is defined as

$$S_n = ar^0 + ar^1 + \cdots + ar^{n-1} = \sum_{k=1}^{n} ar^{k-1} = \begin{cases} a\left(\frac{1-r^n}{1-r}\right) & r \neq 1 \\ an & r = 1 \end{cases}$$

  Implement the sum both recursively in `geometric_sum` (left hand side) and statically in `geometric_sum_definition` (right hand side).

For each of the questions, you need to fill in the function bodies (in place of `# YOUR CODE HERE`). To test your code locally on a subset of Gradescope tests, run the script `hw4_tests.py`. After you are done, you will submit your code to Gradescope, where we wil run your code on a full suite of tests and the autograder will assign your code points based on whether it passes the tests.

1. (2 points) If you were implementing the Python library, how would you go about implementing the `len()` function for a string? Argue for your solution based on its computational complexity and space complexity.

Solution

```
def len(s):
    count = 0
    for character in s:
        count += 1
    return count
```

3 Code to Equation (6 points)

Given each recursive function, write the corresponding function mathematically.

1. (3 points) **Short answer:** Function `f` where n is an integer. Assume $n \geq 1$. Write an expression for $f(n)$.

```
def f(n):
    if n == 1:
        return 2/5
    return 2 * n / (n + 4)
```

Solution

$$f(n) = \begin{cases} \frac{2}{5} & \text{if } n = 1 \\ \frac{2n}{(n+4)} & \text{if } n > 1 \end{cases}$$

2. (3 points) **Short answer:** Function `g` where n is an integer. Assume $n \geq 0$. Write an expression for $f(n)$.

```
def g(n):
    if 0 <= n <= 3:
        return n
    if n % 3 == 0:
        return g(n-3) + g(n-2)
    elif n % 2 == 0:
        return g(n-1) + g(n-2)
    else:
        return g(n-1) + g(n-3)
```

Solution

$$g(n) = \begin{cases} n & \text{if } 0 \leq n \leq 3 \\ g(n-3) + g(n-2) & \text{if } n \bmod 3 = 0 \\ g(n-1) + g(n-2) & \text{if } n \bmod 2 = 0 \text{ \& } n \bmod 3 \neq 0 \\ g(n-1) + g(n-3) & \text{if } n \bmod 2 \neq 0 \end{cases}$$

4 Collaboration Questions

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found in the syllabus.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details.
3. Did you find or come across code that implements any part of this assignment? If so, include full details.

Your Answer

1. Did not receive any help whatsoever.
2. I did not give any help to anyone.
3. I did not come across code that implements any part of the assignment.