MATH 157: Mathematics in the world

Homework 3 (Due February 20th, 2018 at 12:30PM)

Each problem has the same weight, and the highest 5 out of 6 will be scored.

Problem 1

In class we demonstrated that the number of trailing zeros in $n!$ is given by the function

$$f(n) = \sum_{i=1}^{\infty} \left\lfloor \frac{n}{5^i} \right\rfloor$$

Let $g(n)$ denote the number of trailing zeros in the number

$$\prod_{m=1}^{n} m!.$$

1. Implement f in Python. Print the value of $f(100)$ in order to confirm our computation from class.
2. Express $g(n)$ in terms of the function f.
3. Use your answer to implement g in Python. What is the value $g(100)$?
4. Substitute f into g to arrive at a double sum expression. Exchange the sums to arrive at a new expression for g.
5. Use your answer to the previous part to compute $g(100)$ by hand.

   You should not be evaluating any long sums by hand, computer, or calculator. Try to simplify things as much as possible, and make use of the formula

   $$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

   (Hint: After exchanging the sums, consider the outer one. Given that $n = 100$, how many values of the outer iterable do you need? Try to evaluate the inner sum separately for each of the outer values that matter. If you are not sure how to do that, write the first few elements of the series and look for a pattern.)

Problem 2

1. Write a function `is_prime` which accepts an integer and determines whether it is prime. [1]

2. Write a function `factorial` [2] and use it to find all $n \in \{1, \ldots, 15\}$ such that $n! + 1$ is prime.

Problem 3

1. Is $4^{543} + 543^4$ a prime number?

2. What is the last digit of $7^{7^{2015}}$ (in base 10)?

3. Prove the following divisibility criterion for 13: Split off the last digit. Add four times of it to the number that is left. The result is divisible by 13, if and only if the first number is.
   Note: you may need to repeat the process several times to get a number small enough to be able to say if it is divisible by 13.

4. Extra Credit: Invent a new way of testing a number is divisible by 13. Explain why you think it is better than the method above. (I am accepting answers like: because this is what I invented and etc.)

Problem 4

The aim of this exercise is to show you how to plot a function using Python. There are a number of plotting solutions for Python, but one of the most common ones is Matplotlib[3]. We will also benefit from a numeric computation library called Numpy[4].

1. Execute the following Python program to confirm that both Matplotlib and Numpy are installed on your system.

```
import matplotlib
import numpy
```

[1] If you are using Python 2, there is an important difference between the functions `range` and `xrange`. The former always allocates a list of the numbers in the range, which may be an issue depending on memory constraints. The advantage of `xrange` is that it produces a special object, called a *generator*, which allows us to loop through the values in the range without constructing a list of them in memory first.

If this sounds confusing, the bottom line is you should always write `for i in xrange(...)` and not `for i in range(...)` in Python 2.

In Python 3, the `range` function is much smarter and you don't need to worry about using it in a for loop.

[2] If you are using Python 3, you can use `math.factorial` instead.

[3] `http://matplotlib.org/`

[4] `http://www.numpy.org/`

If you get an error and are not sure how to proceed, I suggest using a prepackaged Python distribution such as Canopy or Anaconda (see Homework 1).

2. Plotting with Python is very simple. For example, consider the following program which plots $f(x) = x^2$, $g(x) = \log(x)$, and $h(x) = \lfloor x \rfloor$.

```
import math
import matplotlib.pyplot as plt
import numpy as np

f = lambda x: x**2
g = math.log
h = math.floor

f_domain = np.arange(-3, 3, 0.01)
g_domain = np.arange(0.01, 3, 0.01)
h_domain = f_domain

plt.plot(f_domain, map(f, f_domain))
plt.plot(g_domain, map(g, g_domain))
plt.plot(h_domain, map(h, h_domain))

plt.show()
```

   If you would like to save a figure for later use, replace `plt.show()` with `plt.savefig("filename.png")`.

   For more information about plotting, take a look at the following tutorial – `http://matplotlib.org/users/pyplot_tutorial.html`.

   Recall the Kill Bill function

$$S(n) = 2(n - 2^{\lfloor \log_2 n \rfloor}) + 1.$$

   Write a program which plots $S(n)$ for $n = 1, \ldots, 1000$ and include the plot in your write-up.

3. The floor function $\lfloor \cdot \rfloor$ satisfies the inequality

$$x - 1 \leq \lfloor x \rfloor \leq x$$

   for all real values x.

   To gain some visual intuition about the inequality, plot each of the three terms on the same pair of axes. Include the plot in your write-up.

4. Recall that the number of trailing zeros in $n!$ is given by the function

$$f(n) = \sum_{i=1}^{\infty} \left\lfloor \frac{n}{5^i} \right\rfloor.$$

Use the right side of the inequality above to obtain an upper bound for f. Simplify your answer by using the geometric series formula[5].

5. Use the left side of the inequality to obtain a lower bound for f. Be careful to count the number of non-zero terms in the sum defining f.

6. Plot f together with its two bounds on the same pair of axes. Use the domain $\{1, \ldots, 1000\}$. Include the plot in your write-up.

Problem 5

1. Use the Sieve of Eratosthenes to write a function `primes_up_to` which given an integer $n \geq 2$ returns all primes in the interval $[2, n]$.

2. Write a function `pi` which returns the number of primes less or equal to n. In number theory, this is known as the *prime-counting function* π.

3. Write a function `pi_list` which given an integer $n \geq 2$, returns the list of values

$$\pi(2), \pi(3), \ldots, \pi(n).$$

It is very easy to define `pi_list` as follows.

```python
pi_list = lambda n: [pi(i) for i in range(2, n+1)]
```

The challenge here is to use a single call to `primes_up_to`, and not one for every value $2 \leq i \leq n$. [6]

The prime counting function π lies at the center of one of the most celebrated results in number theory, the Prime number theorem. The result states $\pi(n)$ is asymptotic[7] to $n/\ln(n)$ meaning that

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1.$$

---

[5] `http://en.wikipedia.org/wiki/Geometric_series`

[6] Hint: Imagine you have constructed a list `l` of the values $\pi(i)$ for $2 \leq i \leq p-1$ for some prime p. The next prime after p is q. What are the values $\pi(p), \ldots, \pi(q-1)$, and how can you update the list `l` accordingly?

Recall that the plus sign performs concatenation when applied to lists in Python. For example, `[0,1] + [2,3]` returns `[0,1,2,3]`.

[7] More generally, we say that two functions f and g are *asymptotic*, denoted by $f \sim g$, if

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1.$$

4. Plot $\pi(n)$ and its approximation $\dfrac{n}{\ln n}$ on the same pair of axes. Use $2 \leq n \leq 10^6$ as your domain.

5. Produce a second plot of the following ratio over the same domain:

$$\frac{\pi(n)}{n/\ln n}.$$

Problem 6

Recall the *Sum the digits!* problem from class. We showed that the sequence a_t defined recursively as

$$a_0 = 2015!, \qquad a_{t+1} = S(a_t)$$

converges to $a = 9$.

For the sake completeness, here is how the argument goes. First, we observe that $S(n) \equiv n \pmod 9$ (this is analogous to the divisibility criterion for 9). It follows that the limit $a = \lim_t a_t$ is congruent to 2015! mod 9. On the other hand, 9 divides 2015!, so $9|a$. Next, note that $S(n)$ is smaller than n for $n \geq 10$ (to be verified below). This implies that the limit can be either $a = 0$ or $a = 9$. But if $n > 0$, then $S(n) > 0$, so the solution $a = 0$ can be discarded. We conclude that $a = 9$.

In fact, the argument we presented here generalizes to compute the limit of any recursive sequence defined in terms of S. Let m is a positive integer and $0 \leq r < 9$ be the remainder of m modulo 9. Then the limit of the sequence

$$a_0 = m, \qquad a_{t+1} = S(a_t)$$

is

$$a = \lim_t a_t = \begin{cases} r & \text{if } r > 0, \\ 9 & \text{if } r = 0. \end{cases}$$

Below we will use the following definition of S. Given a number $n = \sum_{i=0}^{\ell} c_i 10^i$ where all c_i are digits, we set

$$S(n) = \sum_{i=0}^{\ell} c_i.$$

1. The number of digits (base 10) of a positive integer n is given by

$$1 + \lfloor \log_{10} n \rfloor.$$

Use this fact to show that

$$S(n) \leq 9(1 + \lfloor \log_{10} n \rfloor).$$

Hint: The largest digit is 9.

2. Find the smallest integer n_1 such that

$$9(1 + \lfloor \log_{10} n_1 \rfloor) < n_1.$$

   Hint: The number is sufficiently small (less than 20) to find it by hand. If this sounds hard, you can also write a simple program to do that.

   Assuming that this inequality holds for $n \geq n_1$, explain why $S(n) < n$ for $n \geq n_1$.

3. Conclude that $S(n) < n$ as long as $n \geq 10$. (Note this implies the limit is a single digit.)

   Hint: Check manually the values of n not covered by the previous exercise.

4. Implement the function S in Python.

5. Use your implementation of S and the `factorial` function above, to find the smallest $\ell \geq 1$ such that

$$\underbrace{S \circ \cdots \circ S}_{\ell}(2015!) = 9.$$

Remark. Note that Python has no trouble working with very large numbers such as 2015!. In this case, 2015! has 5786 digits and certainly does not lie in the range of a standard 64-bit integer.

This convenience is due to the fact Python uses *arbitrary precision* integers out of the box. See `http://en.wikipedia.org/wiki/Arbitrary-precision_arithmetic`.