CSC110 Fall 2022 Assignment 2: Logic, Constraints, and Wordle!

Yehyun Lee

October 12, 2022

Part 1: Conditional Execution

Complete this part in the provided `a2_part1_q1_q2.py` and `a2_part1_q3.py` starter files. Do **not** include your solutions in this file.

Part 2: Proof and Algorithms, Greatest Common Divisor edition

1. We have to use range(1, m + 1) instead of range(1, n + 1). Here's the following reason:

   Because the precondition is $1 \leq m \leq n$, we have the possibility of n having a greater positive integer value than m positive integer value. In other words, to have the largest possible number of common divisors for m and n, having n will exceed m when using range(1, n + 1). So this won't work.

   If we use range(1, m + 1) instead of range(1, n + 1), the largest common divisor will stop at m. So we need to use range(1, m + 1) to have the largest possible number of common divisors for both m and n.

   This is because if we use range(1, n + 1), we would have a greater number than m, since precondition $1 \leq m \leq m$. And since, we learned from lecture 9 that properties of divisibility includes $\forall$ n, d $\in \mathbb{Z}^+$, $d \mid n \implies d \leq n$, properties of divisibility state that if one positive integer divides second positive integer, then first is less than or equal to second; having greater possible divisors than first positive integers, in this case, having greater possible divisors than m when using range(1, n + 1) violate the properties of divisibility.

   Thus, we need to use range(1, m + 1) instead of using range(1, n + 1).

2. This is because we always don't have an empty collection, but in fact, we have at least one value inside the collection all the time.

   Remember when we proved the definition of properties of divisibility in class(also reference course note 4.6), this is also true: $\forall$ n $\in \mathbb{Z}$, $1 \mid n$. And because 1 always divides any n value, it must be included in the collection. See how range(1, m + 1) starts from 1, meaning common_divisors will always include 1 at least.

   Since there's 1 in common_divisors, this is not an empty collection. Thus, we do not need to check whether calling max(common_divisors) will cause an error for collection that is not an empty set, but contains 1 at least or more values.

3. *Proof.*

   Let n, m, d $\in \mathbb{Z}$, assume there exists $k_0 \in \mathbb{Z}$, such that $m = dk_0$ and $m \neq 0$.

We want to prove that (d | n ⇔ d | n % m), which we'll do in 2 parts: first proving that (d | n ⇒ d | n % m), and then proving that (d | n % m ⇒ d | n).

Part 1: Proving d | n ⇒ d | n % m

Assume that d | n, i.e., that there exists $k_1 \in \mathbb{Z}$ such that n = dk_1. We want to prove that d | n % m.

Proof:

We want to show d | n % m.
We want to use Quotient-Remainder Theorem, and use the given property.

Remember, according to Quotient-Remainder Theorem,
dividend / divisor = quotient R remainder.

From here, move divisor to right side. (As you know we don't multiply the remainder here.)
dividend = (divisor * quotient) + remainder.

This is n = qd + r from the Quotient-Remainder Theorem.

Since, d | n % m = d | b, where b is remainder,
n = dividend, m = divisor, a = quotient, b = remainder.

Changing equation accordingly: n = (m * a) + b

Now, we know r = b = n - ma, as we move the (m * a) to other side.
So, n % m = r = b = n - ma.

Thus, d | (n % m) = d | r = d | (n - am).

From the given property, we can replace (n - am). (Now, you can start to see the similarity here.)
d | (n + m(-a)).
d | (an + bm).

And now,
since d | n ∧ d | m ⇒ d | (an + bm) is true, we know d | n ⇒ d | (n + m(-a)) is also true, as integer coefficient of a and b does not really matter in this case, and thus makes us have same (n + m) as the given property.

Thus, d | n ⇒ d | n % m is true.

Part 2: Proving d | n % m ⇒ d | n

Assume that d | n % m, we want to prove that d | n.

Proof:

According to properties of divisibility, there exists $k_1 \in \mathbb{Z}$, such that n = dk_1.

So we know, $n = dk_1$ for the $d \mid n$.
Since we have proved in Part 1 that $n \% m$ is $n - ma$, we know $dk_2 = n - ma$.

So when we substitute $n = dk_1$,
$dk_2 = n - ma$ is
$dk_2 = dk_1 - ma$.

Since $dk_2 = n - ma$,
$n - ma = dk_1 - ma$.

And again substitute $n = dk_1$,
$dk_1 - ma = dk_1 - ma$.

Since this is equivalent, we can conclude that $d \mid n \% m \Rightarrow d \mid n$ is true.

Thus, $d \mid n \% m \Rightarrow d \mid n$ is true.

Since we proved $(d \mid n \Leftrightarrow d \mid n \% m)$ is true, we can conclude that the whole statement is true.

□

4. According to Quotient-Remainder Theorem, we can use equation: $n = qd + r$.
If $r == 0$ because $n \% m == 0$, the equation is: $n = qd$.
Since $n = qd$ or now $n = qm$, n is divisible by m, and so m divides n, thus, if r is 0, n is divided by m, so, we can return m when $r == 0$.

(Note that we use m because when r is 0, m divides n, but another reason is because m is smaller than n, and is largest possible number that can divide both m and n. Remember, according to properties of divisibility, $\forall\ n, d \in \mathbb{Z}^+, d \mid n \implies d \leq n$, meaning we cannot have a greater integer than dividend when looking for greatest common divisor.)

We use range(1, r + 1), because, r is the greatest possible common divisor that we can have when we're trying to find the range of numbers smaller than range(1, m + 1). Remainder of $n \% m$ is the greatest possible common divisor we can have because $n = qd + r$ is equivalent to $r = n - qd$, and so we can switch position of n and m, and replace r instead of n, so gcd(n, m) is equivalent to gcd(m, r). So now, r is the greatest possible common divisor instead of m, and now instead of using range(1, m + 1), we can use range(1, r + 1).
Also, since the precondition require us to have number at least greater or equal to 1, we start from 1 as a min, then r + 1 for max. (Note: We have to add 1, to include r.)

```
def gcd(n: int, m: int) -> int:
    """Return the greatest common divisor of m and n.

    Preconditions:
    - 1 <= m <= n
    """
    r = n % m

    if r == 0:
        return m
    else:
        possible_divisors = range(1, r + 1)
        common_divisors = {d for d in possible_divisors if divides(d, n) and divides(d, m)}
        return max(common_divisors)
```

Part 3: Wordle!

Complete this part in the provided `a2_part3.py` starter file. Do **not** include your solutions in this file.