Sophia Su

Professor Hunter Alzate

Discrete Math for CS

The Euclidean Algorithm Task 7

Task 7.) Part a.) Learn about the Euclidean algorithm and how it works, what it does, what it's efficiency is, number of recursive calls, etc. and write about which Tasks it could've been used in for this programming Assignment.

The Euclidean algorithm is named after Euclid, who described this method in a mathematical treatise called *The Elements*. It is a method for quickly finding the greatest common divisor of two numbers. He based it on the principle that the GCD of two numbers divides their difference.

When we divide the larger number (dividend a) by the smaller number (divisor b), we get a remainder. If the remainder is zero, then the algorithm terminates, otherwise, we update the divisor to be the new dividend, and the remainder to be the new divisor. We continue this process until the remainder becomes zero, and at this point, the last non-zero remainder is the GCD.

The efficiency of the Euclidean algorithm, or how quickly it finds the GCD of two numbers, is pretty high due to its low time complexity. It can find the GCD efficiently for large numbers. The time and space complexity is expressed as $O(\log \min(a, b))$, where a and b are the input numbers.

This time complexity is also the approximate number of recursive calls. It is, of course, dependent on the input numbers. In this case, the upper bound and average case have the same time complexity - when a and b are consecutive numbers, it is approximately $(\log \min(a, b))$. With each step of the algorithm, the size of the numbers

decreases exponentially. The lower bound is $O(1)$, when one number is 0 or if a and b are relatively prime. But typically, the recursive calls made by the Euclidean algorithm are roughly proportional to the number of digits in the smaller number, since each recursive call will reduce the size of the problem by about one constant factor.

The tasks that the Euclidean algorithm could be used for include Task 3, the greatest_common_divisor and Task 5, relatively_prime. The latter calls the former. The Euclidean algorithm's base case is when b is 0, at which point it returns a as the GCD. If a is 0 at the start, then b is the GCD. If a or b is 0, the time complexity is $O(1)$, since one operation determines b is GCD (no recursive calls).

Part b.) Rewrite your code for Task 5 using the Euclidean algorithm as a subroutine.

```
def euclidean_gcd(a, b):
    """Return the greatest common divisor of a and b using the Euclidean algorithm.
    Args:
        a (int): The first integer.
        b (int): The second integer.
    Returns:
        int: The gcd of a and b.
    """
    if b == 0:
        return a
    else:
        return euclidean_gcd(b, a % b)

def relatively_prime(a, b):
    """Check if a and b are relatively prime.
    Args:
        a (int): The first integer.
        b (int): The second integer.
    Returns:
        bool: True is a and b are relatively prime. False otherwise.
    """
    return euclidean_gcd(a, b) == 1
```