Ryan Richardson

Design and Documentation

10/14/23

Project 07

Original Code:

import random

import cProfile

def guess():

    return random.randint(2, 5000)

def isPrime(x):

    for i in range(x):

        for j in range(x):

            if i \* j == x:

                return False

    return True

def findPrimes(num):

    primes = []

    for i in range(num):

        p = guess()

        while not isPrime(p):

            p = guess()

        primes += [p]

    return primes

cProfile.run('print(findPrimes(100)[:10])')

This is the initial program I tested on 100 numbers. The result the cProfile returned was:

A screenshot of a computer

Description automatically generated

Cumulative time is what we want to look at for each part of the program. Since findPrimes() is where everything is executed in our program, it is expected to show the total time. The guess() function call is nearly zero so that hints it is not the problem and not in need of being optimized. When we look at the isPrime() function it is taking up almost 100% of the total computation time of the program and shows this is where the issue lies. Each time isPrime() is called in findPrimes(), it uses two nested for loops to check if the number is prime which is very inefficient. After changing the function to:

def isPrime(x):

    if x <= 1:

        return False

    for i in range(2, x):

        if (x % i) == 0:

            return False

    return True

The output of cProfile is as follows:  
A screen shot of a computer program

Description automatically generated

It reduces the isPrime() function calls from 98.005 seconds to just 0.038 with a more efficient algorithm which contains only one for loop. In addition, extra conditions can be made such as if a number is even to immediately return false with this algorithm:

def isPrime(x):

    if x == 2:

        return True

    if x <= 1 or x % 2 == 0:

        return False

    for i in range(3, x):

        if (x % i) == 0:

            return False

    return True

The cProfile output is:

A screen shot of a computer

Description automatically generated

This makes isPrimes() slightly faster once again but not nearly as drastically as the difference between the first and second algorithm. By profiling this code, I was able to see exactly where the inefficiency lied and how my changes were affecting the code run time.